WebNavigator: Global Web Navigation via Interaction Graph Retrieval

Xuanwang Zhang[1,2,3,*] **Yuteng Han**[1,2,*] **Jinnan Qi**[1,2]
Xinyu Liu[1,2] **Mulong Xie**[3] **Zhen Wu**[1,2,†] **Xinyu Dai**[1,2]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]School of Artificial Intelligence, Nanjing University, China
[3]Fellou AI

zhangxuanwang@smail.nju.edu.cn wuz@nju.edu.cn

Abstract

Despite significant advances in autonomous web navigation, current methods remain far from human-level performance in complex web environments. We argue that this limitation stems from Topological Blindness, where agents are forced to explore via trial-and-error without access to the global topological structure of the environment. To overcome this limitation, we introduce WebNavigator, which reframes web navigation from probabilistic exploration into deterministic retrieval and pathfinding. WebNavigator constructs Interaction Graphs via zero-token cost heuristic exploration offline and implements a Retrieve-Reason-Teleport workflow for global navigation online. WebNavigator achieves state-of-the-art performance on WebArena and Online-Mind2Web. On WebArena multi-site tasks, WebNavigator achieves a 72.9% success rate, more than doubling the performance of enterprise-level agents. This work reveals that Topological Blindness, rather than model reasoning capabilities alone, is an underestimated bottleneck in autonomous web navigation. Code is available at `https://github.com/fate-ubw/webNavigator`.

1 Introduction

Despite achieving superhuman proficiency in automated code generation and complex mathematical theorem proving [1, 2], autonomous agents continue to struggle with human-level performance in dynamic web navigation, particularly in complex cross-site scenarios [3–5]. Web navigation agents operate by receiving a natural language intent and issuing a sequence of actions, observing only the current page state at each step [3]. The predominant approach, exemplified by ReAct [6], follows a single linear trajectory where the agent selects each action based solely on the current observation and interaction history, without any awareness of the broader environment. The research community has converged on two dominant paradigms to address this limitation.

Paradigm 1: Online Exploratory Search. Methods in this paradigm attempt to expand an agent's observational scope at inference time through best-first search, Monte Carlo Tree Search, or reflection-guided planning [7–9]. While these approaches acquire broader environmental context than reactive baselines, they remain constrained to local subtree expansion. Each expansion step extends the agent's planning horizon incrementally by accumulating observations through interaction history, without ever revealing the global topological structure of the environment. As a result, these methods represent sophisticated trial-and-error within a bounded horizon rather than principled global planning.

[*]Equal contribution. [†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Paradigm 2: Learned Internal Planning and World Models. A second line of work attempts to internalize environmental knowledge through model parameters or learned world models. Methods in this category train transition models to predict state changes [10] or elicit latent environmental knowledge from LLM priors [11–13]. Recent studies reveal that learned world models are prone to counterfactual hallucinations, fabricating page elements and states that do not exist, comprising the largest share of prediction failures, while methods relying on parametric priors find that leading LLMs perform near random chance when predicting web interaction outcomes from internal knowledge alone [14, 15]. We argue that this brittleness stems from a fundamental information deficit. Because the global navigational structure of a website remains latent behind individual page observations, models are forced to fill this gap through parametric approximation rather than grounded environmental knowledge.

Despite these efforts, both paradigms continue to fall short of human-level performance on complex web navigation tasks. The academic community often attributes such shortcomings to insufficient multi-step reasoning capabilities. However, we argue this view is incomplete. Beyond reasoning capacity, both paradigms share a more fundamental and systematically overlooked structural bottleneck, which we term Topological Blindness. Specifically, agents are forced to plan with only a local information subset, comprising current observations, interaction history, and internal priors, while remaining blind to the global topological structure of the website.

We posit that web agents, much like expert human users, require a persistent mental map of the environment to achieve optimal planning. Even domain experts cannot efficiently navigate an unfamiliar enterprise system on their first encounter. Once a human expert has pre-explored a system, navigation becomes deterministic. The expert knows precisely which pages exist, how to reach them, and which path leads most directly to the goal. This observation motivates our central insight. In structured web environments, navigation should be reframed from a probabilistic reasoning challenge into a deterministic retrieval and pathfinding problem. We introduce WebNavigator, a two-phase framework comprising Offline Interaction Graph Construction and Online Retrieval-Augmented Navigation. **Offline Interaction Graph Construction:** WebNavigator employs a heuristic engine to systematically formalize the website's topological structure into a directed Interaction Graph $\mathcal{G}$. Traditional crawlers parse only static hyperlinks and cannot discover page states that emerge through user interaction, whereas our engine actively interacts with dynamic elements to capture the full navigable structure of the website. Crucially, this engine builds the Interaction Graph with zero-token cost, requiring no LLM involvement and only a homepage URL as input. Subsequently, all nodes are embedded and indexed into a vector database [16], transforming the Interaction Graph into a retrievable knowledge base. **Online Retrieval-Augmented Navigation:** During inference, the Global-View Navigator (Figure 1) implements a three-stage workflow that locates the optimal target page and teleports the agent to it: (1) **Retrieve.** The agent issues a navigation query, and the Navigator retrieves the top-k relevant observations from the pre-constructed knowledge base. (2) **Reason.** A multimodal selector identifies the optimal observation. (3) **Teleport.** A pathfinding algorithm computes and executes the shortest trajectory to the target observation at zero-token cost. WebNavigator shifts agents from blind exploration to global planning with only six actions. The Global-View Navigator serves as a modular component that can be integrated into various existing frameworks.

We evaluate WebNavigator on WebArena and Online-Mind2Web [3, 17]. WebNavigator achieves state-of-the-art performance in both WebArena and Online-Mind2Web. Our results demonstrate that WebNavigator achieves a 50.0% success rate on the most challenging multi-site tasks in WebArena, doubling the previous SOTA under identical configurations. Remarkably, leveraging Gemini-2.5-Pro, WebNavigator establishes a new performance ceiling of 72.9% on multi-site tasks, more than doubling the enterprise-level agent CUGA [18]. Furthermore, our experiments across 136 real-world websites in Online-Mind2Web confirm the robust generalization of our paradigm. Collectively, these results identify Topological Blindness as a fundamental and overlooked bottleneck that constrains the potential of autonomous web agents.

2 Problem formulation

We formalize the web navigation task as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$ [3, 8], where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, $\mathcal{T}$ the transition function, and $\mathcal{R}$ the reward function. An agent receives

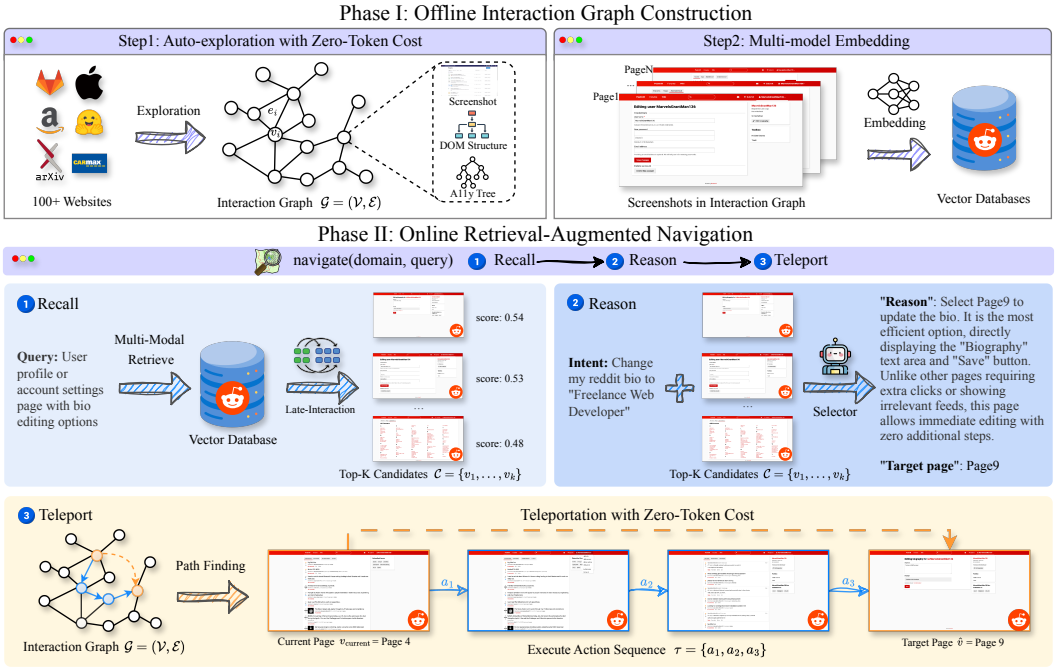

Figure 1: Overview of WebNavigator. WebNavigator resolves Topological Blindness via a two-phase paradigm. (1) **Offline Interaction Graph Construction**. A heuristic auto-exploration engine discovers dynamic page observations at zero-token cost and indexes all observations into a vector database. (2) **Online Retrieval-Augmented Navigation**. The Global-View Navigator implements a three-stage workflow: **Retrieve** top-k candidates from the Interaction Graph via multimodal retrieval; **Reason** to identify the optimal target page; and **Teleport** by computing and executing the shortest path within the Interaction Graph, achieving globally optimal navigation.

an intent i and an initial observation o_0. At each time step t, the agent generates an action $a_t \in \mathcal{A}$ based on the current state s_t, where s_t typically encodes the intent i, current observation $o_t \in \mathcal{O}$, and interaction history $h_t = (o_0, a_0, \ldots, a_{t-1})$:

$$a_t \sim \pi_\theta(a_t \mid i, o_t, h_t) \tag{1}$$

where θ represents the internal knowledge encoded in LLM parameters. After executing action a_t, the environment transitions to a new underlying state $s_{t+1} \in \mathcal{S}$ according to the transition function $\mathcal{T}(s_t, a_t)$ and returns a subsequent observation o_{t+1} to the agent. This process continues until the agent issues a termination command or exceeds the step limit, receiving a reward $r \in \{0, 1\}$ based on the functional correctness of task completion [3].

We define **Topological Blindness** as the condition where π_θ relies solely on the local information subset $\{o_t, h_t, \theta\}$ without access to the complete observation space $\mathcal{O}$ and transition function $\mathcal{T}$. This structural blindness forces agents into inefficient trial-and-error exploration. We argue that augmenting the policy with environmental knowledge resolves this limitation:

$$a_t \sim \pi_\theta(a_t \mid i, o_t, h_t, \mathcal{O}, \mathcal{T}) \tag{2}$$

With complete environmental information, agents can in principle achieve globally optimal planning within model capacity bounds. However, providing complete $\mathcal{O}$ and $\mathcal{T}$ to agents remains impractical in real-world scenarios. This raises our central question: *Can we construct a compact environmental representation that approximates $(\mathcal{O}, \mathcal{T})$ to enable global planning in web navigation?*

3 WebNavigator

We decompose web navigation into the exploration and local execution stages [13]. The exploration stage involves navigating across web pages to locate task-relevant observations, such as exploring

from the homepage to find the product editing page. The local execution stage involves performing specific interactions within identified pages, such as filling out forms to update product information. Complex tasks alternate between these two stages multiple times. In this paper, we focus on achieving globally optimal planning in the exploration phase.

Previous work has shown that modeling observation transitions $T' : (o_t, a_t) \rightarrow o_{t+1}$ is sufficient to simulate web environments [14]. Building on this insight, we introduce the **Interaction Graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to capture observation transitions. Nodes $v \in \mathcal{V}$ represent unique page observations, directed edges $e = (v, a, v') \in \mathcal{E}$ represent observation transitions, where executing action $a \in \mathcal{A}$ at node v induces a transition to node v'. Each node v is grounded via an observation mapping ϕ, where $o_v = \phi(v) = (o_v^{\text{vis}}, o_v^{\text{str}})$ denotes the multimodal observation comprising a screenshot o_v^{vis} and structural metadata o_v^{str} including DOM trees and accessibility trees. Given the Interaction Graph $\mathcal{G}$ that encodes the observation space $\mathcal{O}$ and transition dynamics T', agents can achieve globally navigation in the exploration phase:

$$a_t \sim \pi_\theta(a_t \mid i, o_t, h_t, \mathcal{G}) \tag{3}$$

By conditioning on $\mathcal{G}$, agents convert web navigation from probabilistic exploration into deterministic pathfinding, effectively addressing Topological Blindness. WebNavigator instantiates this paradigm through two phases: Offline Interaction Graph Construction and Online Retrieval-Augmented Navigation.

3.1 Phase I:Offline Interaction Graph Construction

Constructing Interaction Graphs $\mathcal{G}$ requires discovering dynamic web states that emerge through user interactions. Traditional crawlers parse static hyperlinks to discover URL-addressable pages but miss interaction-triggered states, thereby capturing only a narrow subset of the complete observation space $\mathcal{O}$ [19]. Recent LLM-based exploration methods [20] suffer from probabilistic reasoning and context-window constraints, leading to redundant revisiting, incomplete coverage, and prohibitive token costs.

Heuristic Auto-Exploration Engine. As shown in Figure 1 phase I, we develop a heuristic auto-exploration engine based on breadth-first search (BFS) that systematically interacts with dynamic elements to explore the observation space $\mathcal{O}$. Each discovered node $v \in \mathcal{V}$ is uniquely indexed by hashing its structural components $\text{Hash}(o_v^{\text{str}}, \text{url}_v)$, where each web element is identified via deterministic XPath rather than dynamically assigned BIDs, guaranteeing stable interaction across browser sessions.[1] The engine captures comprehensive representations including DOM structures, accessibility trees, and screenshots. Naive BFS is inefficient because web interactions typically induce only local changes. Child pages inherit most elements from their parents, resulting in redundant exploration. Inspired by the human strategy of avoiding re-clicking explored elements, we design an Adaptive BFS algorithm that leverages structural differencing between DOM trees. At each exploration step, we compute the structural difference between the current node v and its parent v_{parent} to identify newly added elements. We then extract interactive elements from these differential elements. This strategy significantly reduces exploration overhead by focusing on newly emerged interactive elements, which are substantially fewer than the total number of elements in practice. To reach interaction-triggered nodes such as toggled menus, the algorithm simultaneously constructs $\mathcal{G}$ while utilizing it for pathfinding. Additionally, the engine employs a configurable block list to exclude hazardous operations and external links from exploration. Complete details are provided in Appendix Section C.

Graph Indexing for Retrieval. Upon completing exploration, all nodes $v \in \mathcal{V}$ are embedded and indexed in a persistent database for multimodal retrieval during online navigation. Details about retrieval and navigation integration are described in Section 3.2.

3.2 Phase II: Online Retrieval-Augmented Navigation

Directly providing the complete Interaction Graph $\mathcal{G}$ to LLMs is impractical due to context window constraints. Summarizing observations via LLMs not only adds token overhead but also risks losing critical information. WebNavigator resolves this through the Global-View Navigator, which implements a three-stage workflow: (1) **Retrieve:** identifying top-k candidate nodes $\mathcal{C} \subset \mathcal{V}$ via

[1] XPath is a W3C standard language: `https://www.w3.org/TR/xpath/`

multimodal retrieval given navigation query q, (2) **Reason:** selecting the optimal target node $v^* \in \mathcal{C}$ from the candidates, and (3) **Teleport:** computing and executing the shortest path to v^*.

Retrieve. Recent studies in optical compression have demonstrated that vision-language models can process rendered text with significantly fewer tokens than text-only models without sacrificing reasoning performance [21, 22]. Building on this insight, WebNavigator adopts screenshots as optically compressed observations for multimodal retrieval, as they provide more compact representations than prohibitively long and noisy DOM trees. Given a navigation intent i, the agent aims to locate the optimal target observation $o^* \in \mathcal{O}$ where the task can be completed. To identify the corresponding node in $\mathcal{G}$, the agent formulates a navigation query q based on intent i to specify the target page functionality (e.g., for intent "Edit product X's price to $50", query "page to edit product information") and encodes it into multi-vector representation $\mathbf{q} \in \mathbb{R}^{n \times d}$ using the same multimodal embedding model from Phase I, where n denotes the number of tokens and d is the embedding dimension per token [16]. This query embedding $\mathbf{q}$ is then compared with the pre-indexed screenshot embeddings $\{\mathbf{v}_j \in \mathbb{R}^{m_j \times d}\}_{j=1}^{|\mathcal{V}|}$ by computing the late-interaction similarity score [23]:

$$s_{\text{late}}(\mathbf{q}, \mathbf{v}_j) = \frac{1}{n} \sum_{i=1}^{n} \max_{\ell \in \{1,\ldots,m_j\}} \mathbf{q}_i \cdot \mathbf{v}_{j,\ell}^{\top} \tag{4}$$

where m_j denotes the number of tokens in screenshot, $\mathbf{q}_i \in \mathbb{R}^d$ denotes the i-th token embedding in the query, and $\mathbf{v}_{j,\ell} \in \mathbb{R}^d$ denotes the ℓ-th token embedding in screenshot of node v_j. Unlike dense single-vector retrieval, late-interaction computes fine-grained token-level similarities, enabling more precise semantic matching while maintaining computational efficiency [16]. The retrieval process identifies the top-k candidates $\mathcal{C}$ with the highest similarity scores:

$$\mathcal{C} = \{v_1, \ldots, v_k\}, \text{ where } v_i = \arg\max_{v \in \mathcal{V} \setminus \{v_1,\ldots,v_{i-1}\}} s_{\text{late}}(\mathbf{q}, \mathbf{v}) \tag{5}$$

Reason. Since retrieval is based on semantic similarity, the top-k candidates $\mathcal{C}$ may contain visually similar but functionally distinct pages. To identify the optimal target node, we leverage a multimodal LLM as a zero-shot reasoner (prompt detailed in Appendix Section L.4) to select the most likely candidate $\hat{v}$ by analyzing the visual observations $\mathcal{O}_{\mathcal{C}} = \{o_{v_j}^{\text{vis}}\}_{j=1}^{k}$ along with the intent i:

$$\hat{v} = \arg\max_{v \in \mathcal{C} \cup \{\emptyset\}} P_\theta(v \mid i, \mathcal{O}_C) \tag{6}$$

where P_θ represents the model's probability over the candidate set $\mathcal{C}$. When $\hat{v} = \emptyset$, indicating no candidate sufficiently matches the navigation intent, the Navigator bypasses the Teleport stage and the agent falls back to React-style low-level actions to continue task execution. As shown in Section E.2, such cases are rare and do not necessarily lead to task failure. By selecting from the top-k candidates rather than engaging in generative exploration, this approach fundamentally reduces navigation complexity from generation to verification [24, 25].

Teleport. Once the target node $\hat{v}$ is determined, the Global-View Navigator computes the shortest path from the current observation v_{current} to $\hat{v}$ by invoking the pathfinding mechanism used in Section 3.1:

$$\tau = (a_1, \ldots, a_m) = \text{ShortestPath}(v_{\text{current}}, \hat{v}, \mathcal{G}) \tag{7}$$

The Navigator executes this action sequence τ via XPath-grounded actions, transitioning the agent to the target observation with zero-token cost.

Overall, through the Retrieve-Reason-Teleport workflow, the Global-View Navigator approaches globally optimal planning by transforming probabilistic exploration into deterministic retrieval and pathfinding over the pre-constructed graph. From the agent's perspective, this entire workflow is abstracted into a single high-level action that accepts a navigation query and enables teleportation to the task-relevant page, providing an LLM-friendly interface that eliminates the need to manage complex navigation logic.

3.3 WebNavigator Agent Design

Unlike prior approaches that fragment capabilities across numerous atomic primitives [11, 3], we aggregate cross-domain planning, task decomposition, and environment navigation into a high-density yet LLM-friendly interface. WebNavigator encapsulates the entire Retrieve-Reason-Teleport

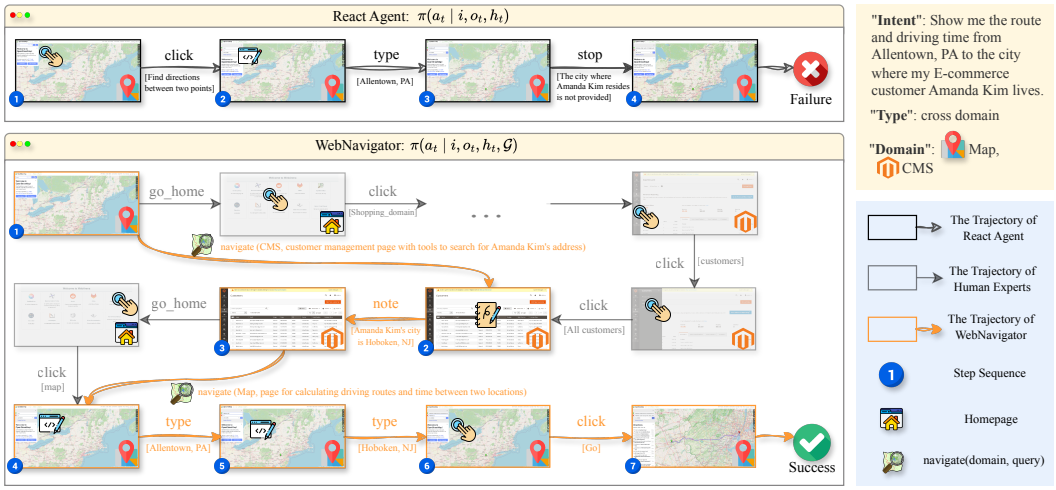

Figure 2: Trajectory comparison on a multi-site task (WebArena 760), which requires retrieving a specific customer address from the CMS to plan a route on the Map. WebNavigator achieves human-level planning via `navigate(domain, query)` actions, whereas the ReAct baseline prematurely terminates due to Topological Blindness. The human expert trajectory is the gold standard.

workflow into a unified action denoted as `navigate(domain,query)`. This design establishes a paradigm of **capability aggregation**. The `query` parameter enables the agent to focus on one navigation subgoal at a time, naturally decomposing complex tasks into sequential steps. The `domain` parameter enables cross-domain planning by allowing dynamic switching between Interaction Graphs. The LLM specifies where to navigate, while the underlying workflow handles how to reach the target as a black box. In contrast to previous methods that fragment tab management capability into explicit actions such as `tab_focus`, `new_tab`, and `tab_close` [3], WebNavigator offers a significant advantage by subsuming these capabilities within the Retrieve-Reason-Teleport workflow. The underlying browser state is automatically managed while shielding the LLM from these mechanics. Methods such as SteP require designing domain-specific actions for each website, including `find_commits` for GitLab and `search_customer` for e-commerce [11]. WebNavigator externalizes environmental knowledge into the Interaction Graph as queryable external memory. This design eliminates the need to handcraft specialized actions for every domain, allowing a generic interface to operate uniformly across websites. Centered on the `navigate(domain,query)` action, we augment with five primitives for local execution. This yields the most compact action space among existing methods (Table 8), reducing decision complexity and improving action selection reliability. Complete action definitions and prompts are detailed in Appendix Section L.3.

4 Experiment

To comprehensively evaluate whether WebNavigator addresses Topological Blindness, we evaluate it on three benchmarks: a controlled environment (WebArena) and two real-world benchmarks (Online-Mind2Web and WebVoyager).

Benchmarks. We primarily evaluate on **WebArena** [3], the most widely used benchmark for web navigation. WebArena comprises 812 carefully designed tasks spanning five representative websites: e-commerce (Shopping), social forums (Reddit), collaborative development (GitLab), content management systems (CMS), and mapping services (Map). WebArena employs programmatic validation mechanisms developed by human experts to assess functional correctness, enabling reliable measurement of task completion. To assess generalization capabilities beyond these controlled environments, we extend our evaluation to the real-world benchmark **Online-Mind2Web** [17], which consists of 300 diverse tasks across 136 live websites. We further include **WebVoyager** [4] as an additional benchmark for end-to-end navigation on live real-world websites.

Table 1: Main results on WebArena and Online-Mind2Web. Act # denotes the action space size. Success rate (SR) for different website domains. **Bold** and underlined values indicate the best and second-best performance among non-enterprise agents. Methods and scores marked with $*$ are our reproduced results. Models marked with † are finetuned.

Method	Model	Act #	WebArena							Online-Mind2Web
			SR(%)	Multisite	Shopping	CMS	Reddit	GitLab	Map	SR(%)
Webarena* [3]	GPT-4o	12	21.1	8.3	24.6	20.3	22.6	22.2	18.4	-
Browsergym [26]	GPT-4o	20	31.4	-	-	-	-	-	-	-
Paradigm 1: Online Exploratory Search										
Tree search [7]	GPT-4o	12	19.2	-	28.1	16.5	10.5	13.3	25.8	16.6*
Exact [8]	GPT-4o	16	-	-	-	-	-	23.5	-	-
Branch and browse [9]	GPT-4o	12	35.8	18.8	34.6	26.4	50.9	36.7	46.8	30.3*
WebPilot [27]	GPT-4o	-	37.2	-	36.9	24.7	65.1	39.4	33.9	-
Auto-Eval [25]	GPT-4-Preview	12	20.2	-	-	-	-	-	-	22.0*
Paradigm 2: Learned Internal Planning & World Models										
WebDreamer [10]	GPT-4o	12	-	-	-	-	-	-	-	35.0
WMA [14]	GPT-4o	12	16.6	-	-	-	-	-	-	21.0*
SteP [11]	GPT-4-Turbo	27	33.0	-	37.0	24.0	59.0	32.0	30.0	17.4*
Plan-and-Act [13]	Llama-70B†	8	45.7	-	-	-	-	-	-	-
Agent-e [28]	GPT-4o	8	-	-	-	-	-	-	-	27.0
AgentOccam [12]	GPT-4-Turbo	8	43.1	14.6	40.6	45.6	61.3	37.8	46.8	-
AgentOccam* [12]	GPT-4o	8	42.9	25.0	33.7	46.7	60.4	44.4	40.4	19.6*
Enterprise-Level Autonomous Agents										
CUGA [18]	-	-	61.7	35.4	58.3	62.6	75.5	61.7	64.2	-
New Paradigm: Retrieve and Navigate										
WebNavigator (Ours)	Qwen3-VL-32B-Instruct	6	47.8	43.8	44.9	45.1	75.5	50.6	44.0	39.7
WebNavigator (Ours)	GPT-4o	6	49.9	50.0	44.4	48.6	73.6	42.2	51.4	41.3
WebNavigator (Ours)	Claude-Sonnet-4	6	57.1	50.0	**51.9**	58.2	**85.9**	50.0	51.4	38.7
WebNavigator (Ours)	Gemini-2.5-Pro	6	**63.3**	**72.9**	**51.9**	**66.5**	**85.9**	**62.2**	**53.2**	**52.7**

Baselines. We compare WebNavigator against methods including WebArena [3], Browsergym [26], Tree Search [7], Exact [8], Branch-and-browse [9], WebPilot [27], Auto-Eval [25], WebDreamer [10], WMA [14], SteP [11], Plan-and-Act [13], Agent-e [28], AgentOccam [12] and CUGA [18].

Implementation Details. To ensure a fair comparison, we strictly align our experimental setup with previous works [17, 12]. Specifically, we reproduce key baselines using GPT-4o as the unified backbone model. Detailed experimental parameters are provided in Appendix Section B.

Main Results. As shown in Table 1, WebNavigator substantially outperforms previous state-of-the-art methods on WebArena and Online-Mind2Web. Specifically, on WebArena, WebNavigator achieves 47.8% with Qwen3-VL-32B-Instruct, 49.9% with GPT-4o, 57.1% with Claude-Sonnet-4, and 63.3% with Gemini-2.5-Pro, surpassing WebPilot (37.2% from Paradigm 1) and Plan-and-Act (45.7% from Paradigm 2). Most significantly, WebNavigator achieves breakthrough performance on multi-site tasks, the most challenging cross-domain setting in WebArena. With GPT-4o and Claude-Sonnet-4, WebNavigator achieves a 50.0% success rate, surpassing AgentOccam by 100% relative improvement over its 25.0%. Compared with the enterprise-level agent, WebNavigator with Gemini-2.5-Pro achieves a 72.9%, more than twice the performance of the CUGA system. Overall, WebNavigator establishes a new performance ceiling on WebArena's multi-site tasks. On Online-Mind2Web, WebNavigator achieves 52.7% with Gemini-2.5-Pro and 41.3% with GPT-4o, establishing state-of-the-art performance on this challenging benchmark. Crucially, these results validate the generalization capability of WebNavigator across 136 diverse real-world websites. WebNavigator also outperforms the reproduced WebVoyager agent in our additional real-world evaluation reported in Appendix Section G.

Necessity of the Selector. We evaluate the Selector at both task and query levels. At the task level, we remove the Selector and directly navigate to the top-1 retrieved page. At the query level, we separately measure whether the Retriever recalls a valid target and whether the Selector identifies the final navigation target. Detailed metric definitions and query statistics are provided in Appendix Section H.

Table 2: Task-level Selector ablation.

Backbone	Top-1	+ Selector	Gain
GPT-4o	37.5%	50.0%	+12.5
Claude-Sonnet-4	41.7%	50.0%	+8.3
Gemini-2.5-Pro	54.2%	72.9%	+18.7

Table 3: Query-level Selector analysis.

Backbone	Top-1 Acc.	Recall@30	Selector Acc.
GPT-4o	22.97%	71.62%	75.82%
Claude-Sonnet-4	51.02%	95.92%	96.72%
Gemini-2.5-Pro	63.83%	95.74%	92.00%

As shown in Tables 2 and 3, adding the Selector improves multi-site success by 8.3–18.7 percentage points across all three backbones. The Retriever achieves high Recall@30, exceeding 95% with Claude-Sonnet-4 and Gemini-2.5-Pro, which shows that the relevant page is highly likely to be included among the retrieved candidates. However, its Top-1 Accuracy remains only 22.97–63.83%, showing that retrieval alone cannot reliably identify the final target. The Selector directly raises target-selection accuracy from 22.97–63.83% to 75.82–96.72%, confirming its importance in the Reason stage.

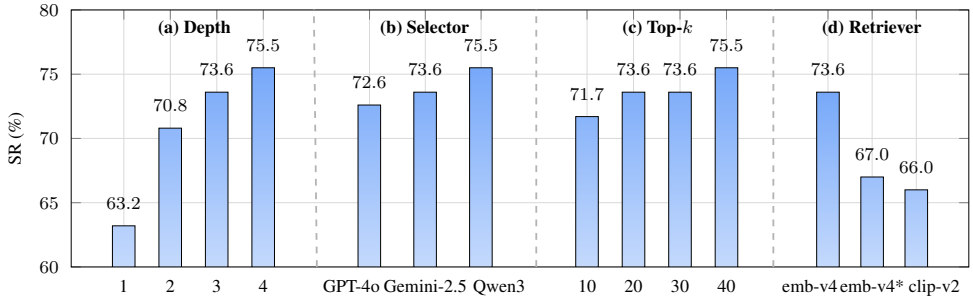

Figure 3: Ablation study on depth, top-k, selector model, and retriever on the Reddit domain in WebArena. Unless otherwise specified, the default configuration uses depth=3, top-k=30, Gemini-2.5-Flash as the selector, and Jina-embedding-v4 as the retriever. The Selector group compares GPT-4o, Gemini-2.5-Flash, and Qwen3-VL-8B. The Retriever group compares Jina-embedding-v4, Jina-embedding-v4 (dense), and Jina-clip-v2. $*$ denotes dense embedding mode.

Necessity of the Interaction Graph. To rigorously isolate the effect of topology awareness, we replace the Interaction Graph with an empty graph while keeping all other settings fixed. We rerun the 514 tasks previously solved by full WebNavigator with Gemini-2.5-Pro, and the empty-graph variant fails on 95. Manual analysis attributes 38.9% of these regressions to Topological Blindness. This controlled result shows that removing topology awareness directly reintroduces failures in locating task-relevant pages, demonstrating the importance of the Interaction Graph for global navigation. Detailed experimental settings and failure classifications are provided in Appendix K.

Structural Foundations of Topological Blindness. The magnitude of performance improvement across domains directly correlates with the severity of Topological Blindness in each environment. Multi-site navigation represents the peak of Topological Blindness, where agents are blind to information in external domains. Consider WebArena task 760: "Show the route from Allentown, PA to where customer Amanda Kim lives." As shown in Figure 2, without global awareness, the react agent is misled by local cues, leading to premature termination. In contrast, WebNavigator achieves human-level planning by acquiring cross-domain knowledge, encoded in $\mathcal{G}_{\text{CMS}}$ and $\mathcal{G}_{\text{Map}}$. Specifically, WebNavigator teleports to the customer page in CMS to retrieve Amanda Kim's city, then switches to Map to query the driving directions, thus completing the task.

Reddit represents a wide and shallow topology with over 90 forums. React agents can only observe a fraction of paths from the homepage, leading to low-probability guesses. Consider WebArena task 681, which requires posting a GAN-related repository to a relevant subreddit. In this case, an agent lacking global awareness is predisposed to converge on a locally plausible candidate such as `/f/coolgithubprojects`. In contrast, WebNavigator identifies the optimal targets such as `/f/deeplearning` and `/f/MachineLearning` by leveraging the complete forums encoded in $\mathcal{G}_{\text{Reddit}}$. This global visibility enables WebNavigator to achieve its most significant gain on Reddit. In deep, complex environments such as CMS, Shopping, and GitLab, agents become trapped by misleading surface pages that mask deeper, task-relevant pages. WebNavigator enables direct

navigation to optimal pages, bypassing these traps with substantial gains on Shopping, CMS, and GitLab. Performance on the Map domain converges across models, where Gemini-2.5-Pro exhibiting no significant margin over GPT-4o or Claude-Sonnet-4. This uniformity is attributed to the constrained observation space of the Map domain ($|\mathcal{V}_{\text{Map}}| = 16$), which mitigates the impact of Topological Blindness. In contrast, other domains contain more than 100 nodes, detailed in Table 5.

5 Analysis

5.1 Impact of Environmental Knowledge Completeness and Information Bandwidth

We investigate the underlying mechanisms that enable WebNavigator to resolve Topological Blindness. Specifically, we conduct controlled experiments on the Reddit domain in WebArena.

Knowledge Completeness. We investigate how the completeness of environmental knowledge affects agent performance. We control knowledge completeness by varying exploration depth from 1 to 4, with fixed GPT-4o, $k = 30$, Gemini-2.5-Flash selector, and Jina-embedding-v4 retriever. As shown in the Depth group of Figure 3, the success rate rises from 63.2% at depth 1 to 70.8% at depth 2. This upward trend persists at greater depths, reaching 73.6% at depth 3 and 75.5% at depth 4. At depth 1, the Interaction Graph captures only surface-level entry points, leaving many task-relevant pages undiscovered. When depth reaches 2, the Interaction Graph achieves sufficient coverage of task-relevant subreddits, marking a critical transition. Depths 3 and 4 provide additional gains by handling tasks requiring deeper navigation, though with smaller incremental improvements. These results confirm that sufficient knowledge completeness is necessary to address Topological Blindness.

Information Bandwidth. We investigate how the information bandwidth affects agent performance. The retrieval top-k parameter controls the number of candidates transferred from the knowledge base to the selector. Larger k expands the decision space of the Selector but increases computational overhead, while insufficient k risks excluding the optimal target. We vary k from 10 to 40 while keeping all other parameters fixed. As shown in the Top-k group of Figure 3, performance improves as k increases. At $k = 40$, WebNavigator achieves 75.5%, surpassing our main configuration of $k = 30$. More importantly, even with the narrowest bandwidth at $k = 10$, WebNavigator achieves 71.7% success rate, outperforming all non-enterprise baseline methods in Table 1. This robustness under constrained bandwidth demonstrates that knowledge completeness, rather than the retrieval channel width, is the primary determinant of performance.

Task Simplification via Verification Asymmetry. A fundamental principle holds that verifying a solution is inherently easier than generating one [25, 17]. WebNavigator instantiates this principle in web navigation by transforming trajectory generation into candidate selection. Rather than constructing a navigation path from scratch, the agent selects the optimal page from a pre-retrieved candidate set. We evaluate Selectors ranging from leading proprietary models to the lightweight Qwen3-VL-8B-Instruct. As shown in the Selector group of Figure 3, the 8B model achieves a 75.5% success rate, performing on par with GPT-4o and Gemini-2.5-Flash. This consistency across model scales confirms that the WebNavigator successfully offloads navigation complexity from model reasoning to structured knowledge retrieval.

Retrieval Granularity. We examine whether the granularity of multimodal retrieval affects the performance of WebNavigator. Specifically, we compare late interaction retrieval, which leverages token-level alignment via multi-vector representations, against traditional dense methods. We evaluate Jina-embedding-v4 in both late interaction and dense mode (denoted by $*$ in Figure 3), and Jina-clip-v2 [29] in dense mode only. As shown in the Retriever group of Figure 3, late interaction retrieval achieves a 73.6% success rate, substantially outperforming dense embedding approaches at 66-67%. Dense embeddings compress entire screenshots into fixed-size vectors, discarding fine-grained spatial cues such as specific buttons, form fields, or menu structures. In contrast, late interaction preserves local visual semantics through token-level matching, enabling the retriever to identify specific page regions that satisfy the navigation intent. This confirms that web navigation requires fine-grained retrieval rather than global compression.

5.2 Error Analysis

We manually inspect all failed trajectories of WebNavigator with Gemini-2.5-Pro across multiple WebArena domains, categorizing each failure into one of four types.

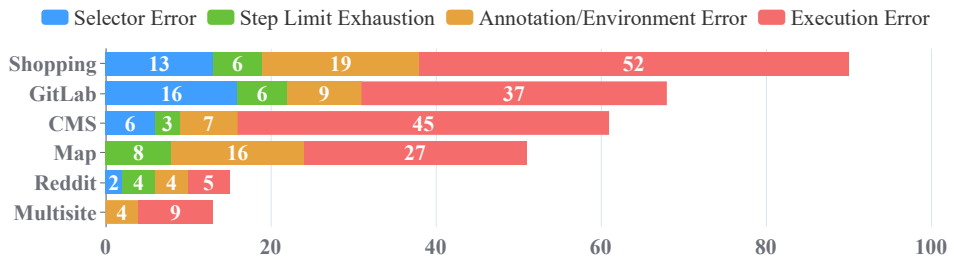

Figure 4: Failure analysis on WebArena (Gemini-2.5-Pro).

Retrieval Failure occurs when the Selector returns `None` or selects an incorrect target page. **Execution Failure** occurs when the agent reaches the target page but fails due to LLM limitations in local reasoning, including wrong answers, incorrect output format, missed constraints, or misjudged task completion. **Step Limit Exhaustion** occurs when the agent exhausts the step budget before completing the task. **Annotation/Environment Error** occurs when ground-truth annotations contain ambiguity or errors, or when the environment behaves unexpectedly. As shown in Figure 4, retrieval failure accounts for only 12.4% of all failures, indicating that the retrieval and pathfinding pipeline reliably handles navigation across the vast majority of tasks. Notably, multi-site tasks record zero retrieval failures, confirming that cross-domain retrieval is robust even in the most challenging setting. The dominant failure mode is execution failure at 58.7%, where the agent reaches the correct page but fails due to LLM limitations in local reasoning. Step limit exhaustion accounts for a further 9.1%, reflecting that even leading models cannot complete local tasks within the budget. Together, LLM-side failures constitute 67.8% of all failures, confirming that residual failures concentrate on LLM capability limitations rather than framework design. The remaining 19.8% corresponds to annotation and environment errors, which affect only 7.3% of all WebArena tasks and fall outside the agent's control.

6 Related Work

Paradigm 1: Online Exploratory Search. This paradigm attempts to mitigate Topological Blindness through search-based algorithms during task execution. Specifically, these approaches employ best-first search and Monte Carlo Tree Search (MCTS) guided by model-based value functions [30, 7]. To enhance reliability, reflective mechanisms leverage contrastive reflection or multi-agent debate to correct navigation errors [25, 8]. Additionally, other works utilize hierarchical structures to optimize local execution [27, 9]. However, such exploration is transient and environment-agnostic, discarding the acquired structural knowledge after each episode. Unlike these reactive exploration approaches, our framework persists environmental knowledge into a reusable interaction graph.

Paradigm 2: Learned Internal Planning and World Models. This paradigm attempts to mitigate Topological Blindness by learning environmental knowledge within model parameters or building world models. One significant direction is the construction of explicit environmental simulators, which learn world models to predict state transitions [10, 14]. Another approach focuses on structural planning, utilizing policy stacks or planning trees to recursively break down long-term objectives [11, 12]. To improve quality, specialized architectures separate high-level planning from low-level execution [13]. Despite their efficiency, these methods rely on probabilistic inference often yielding hallucinated transitions in novel environments. In contrast, our Retrieve-and-Navigate paradigm avoids internal simulations by grounding decisions in pre-constructed environmental knowledge.

7 Conclusion

This work introduces WebNavigator, a novel paradigm that overcomes Topological Blindness by transforming probabilistic exploration into deterministic retrieval and pathfinding, achieving global planning with only 6 actions. WebNavigator achieves state-of-the-art performance on WebArena and Online-Mind2Web. On WebArena multi-site tasks, WebNavigator demonstrates a performance ceiling of 72.9%. These results reveal that Topological Blindness represents an underestimated bottleneck in autonomous web navigation.

Acknowledgments and Disclosure of Funding

The authors declare no conflict of interest. This work is supported by the NSFC (No. 62376120, 62576163) and the Fundamental Research Funds for the Central Universities (No. 2026300382).

References

[1] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[2] Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, November 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09833-y.

[3] 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. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[4] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 6864–6890. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.371.

[5] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vázquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, pages 11642–11662. OpenReview.net, 2024.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[7] Jing Yu Koh, Stephen Marcus McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

[8] Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. Exact: Teaching AI agents to explore with reflective-mcts and exploratory learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[9] Shiqi He, Yue Cui, Xinyu Ma, Yaliang Li, Bolin Ding, and Mosharaf Chowdhury. Branch-and-browse: Efficient and controllable web exploration with tree-structured reasoning and action memory. *arXiv preprint arXiv:2510.19838*, 2025.

[10] Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

[11] Paloma Sodhi, SRK Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked llm policies for web actions. *arXiv preprint arXiv:2310.03720*, 2023.

[12] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for llm-based web agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[13] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, pages 15419–15462. OpenReview.net, 2025.

[14] Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[15] Surgan Jandial, Yinong Oliver Wang, Andrea Bajcsy, and Fernando De la Torre. On the fine-grained planning abilities of vlm web agents. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25347–25380, 2025.

[16] Michael Günther, Saba Sturua, Mohammad Kalim Akram, Isabelle Mohr, Andrei Ungureanu, Bo Wang, Sedigheh Eslami, Scott Martens, Maximilian Werk, Nan Wang, et al. jina-embeddings-v4: Universal embeddings for multimodal multilingual retrieval. In *Proceedings of the 5th Workshop on Multilingual Representation Learning (MRL 2025)*, pages 531–550, 2025.

[17] Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.

[18] Sami Marreed, Alon Oved, Avi Yaeli, Segev Shlomov, Ido Levy, Offer Akrabi, Aviad Sela, Asaf Adi, and Nir Mashkif. Towards enterprise-ready computer using generalist agent. *arXiv preprint arXiv:2503.01861*, 2025.

[19] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10290–10305. Association for Computational Linguistics, 2025.

[20] Jiali Cheng, Anjishnu Kumar, Roshan Lal, Rishi Rajasekaran, Hani Ramezani, Omar Zia Khan, Oleg Rokhlenko, Sunny Chiu-Webster, Gang Hua, and Hadi Amiri. Atlas: Actor-critic task-completion with look-ahead action simulation. *arXiv preprint arXiv:2510.22732*, 2025.

[21] Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. Glyph: Scaling context windows via visual-text compression. *arXiv preprint arXiv:2510.17800*, 2025.

[22] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

[23] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM, 2020. doi: 10.1145/3397271.3401075.

[24] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[25] Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024.

[26] De Chezelles, Thibault Le Sellier, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F Xu, Siva Reddy, Quentin Cappart, et al. The browsergym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467*, 2024.

[27] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 23378–23386. AAAI Press, 2025. doi: 10.1609/AAAI.V39I22.34505.

[28] Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.

[29] Andreas Koukounas, Georgios Mastrapas, Sedigheh Eslami, Bo Wang, Mohammad Kalim Akram, Michael Günther, Isabelle Mohr, Saba Sturua, Nan Wang, and Han Xiao. jina-clip-v2: Multilingual multimodal embeddings for text and images. *arXiv preprint arXiv:2412.08802*, 2024.

[30] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, pages 62138–62160. OpenReview.net, 2024.

[31] Kevin Larson and Mary Czerwinski. Web page design: Implications of memory, structure and scent for information retrieval. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 25–32, 1998.

A Limitations and Societal Impacts

A.1 Limitations

Extension to Computer Use and Mobile Environments. This work focuses on browser-based web navigation and does not extend to computer use or mobile navigation scenarios. Computer use and mobile navigation are pursued as separate research directions within the community, as they involve fundamentally different interaction paradigms, including desktop GUI manipulation and touch-based interfaces. Adapting WebNavigator to these environments is a promising direction for future work.

Browser Harness Coverage of Corner Cases. The current browser harness does not yet handle all corner cases that arise in real-world web navigation. For example, observations may be captured before a page has fully loaded, resulting in incomplete accessibility trees. In addition, dynamically injected Browser element IDs (BIDs) may become invalid by the time an action is executed, causing action execution to fail when the target element can no longer be located by its BID. Although such cases are rare in practice, improving harness coverage to handle them remains an important direction for future engineering work.

Upfront Graph Construction. WebNavigator requires an Interaction Graph to be constructed before task execution on a new website. Although graph construction involves no LLM calls and the resulting graph is fully reusable across tasks, this prerequisite step adds an initial time investment before deployment. We plan to address this by developing a dedicated SDK for web developers that automates graph construction and maintenance, making WebNavigator a lightweight infrastructure layer for large-scale web navigation deployment.

Limited Support for HTML canvas elements. WebNavigator relies on accessibility trees for element grounding, which makes it ineffective on websites where core functionality is rendered via HTML canvas elements, such as browser-based games or interactive visualization tools. These websites expose no semantic accessibility structure, leaving pure visual approaches as the only viable option. Supporting such environments requires fundamentally different grounding mechanisms and is left for future work.

Manual Selection of Exploration Depth. The exploration depth parameter controls the coverage of the Interaction Graph and is currently set manually. Although Section B.2 provides principled guidance for depth selection, an automatic method that adapts depth to each website would further reduce deployment overhead. We leave the development of such an adaptive depth selection mechanism for future work.

A.2 Broader Impact

WebNavigator advances the capability of autonomous web agents to complete complex, goal-oriented tasks across diverse real-world websites, with the potential to assist users in automating daily web-based work. However, we acknowledge the potential risks associated with autonomous web interactions, including the possibility of malicious dual-use such as automating CAPTCHA bypassing or unauthorized data scraping. We emphasize that future deployment must incorporate strict usage policies and human-in-the-loop mechanisms, and we call on the research community to establish shared safety standards for autonomous web automation.

B Implementation Details and Hyperparameter Settings

We conduct all experiments using an RTX A6000 (48GB) server with 8 GPUs. Table 4 and Table 5 summarize the experimental hyperparameters and per-domain graph construction statistics, respectively. For WebArena, we conduct a systematic exploration up to depth 3, yielding Interaction Graphs that provide comprehensive coverage of the five websites. For Online-Mind2Web, which encompasses 136 diverse real-world websites, we adopt a conservative exploration depth of 1. This strategy ensures broad coverage across all 136 websites within a limited time frame, capturing the majority of functional entry points for each website. Despite this conservative setting, WebNavigator

achieves SOTA performance as shown in Table 1, demonstrating that even shallow exploration enables effective generalization from controlled environments to real-world websites. We use the same depth-1 configuration for WebVoyager.

B.1 Experimental Configuration

The global hyperparameters shared across all experiments are listed in Table 4. The selector model, retriever, and inference settings remain fixed across all domains and benchmarks.

Table 4: Global hyperparameters for WebNavigator.

Parameter	Value
Observation	Acc. Tree + Image
Selector	Gemini-2.5-Flash
Selector Input	Intent i + top-k candidate screenshots
Retriever	Jina-embedding-v4
Top-k	30
Temperature	0.5
Top-p	0.95
Max Steps	20

B.2 Graph Construction and Statistics

Table 5 reports the exploration depth, resulting graph size, branching factor, and construction time for each WebArena domain. Notably, the Interaction Graphs constructed by our adaptive BFS are highly sparse and tree-like, with branching factors ranging from 1.10 to 1.98. This sparsity is consistent with modern web design principles, where navigation hierarchies guide users through sequential functional pathways rather than densely interconnected structures.

Table 5: Per-domain graph construction statistics on WebArena.

Parameter	Map	CMS	Shopping	GitLab	Reddit
Exploration Depth	1	2	2	2	3
Graph Nodes	16	122	570	838	225
Graph Edges	19	138	627	1,659	295
Branching Factor	1.19	1.13	1.10	1.98	1.31
Construction Time (min)	2.1	23.1	122.1	316.8	31.1

Selection of Exploration Depth. The exploration depth D is set differently per domain, guided by the principle of Broad, Shallow Information Structures [31], which demonstrates that modern websites concentrate functional entry points at shallow depths. Websites with functionally simple interfaces and shallow navigation, such as Map, require only $D = 1$ to capture primary functionality. In contrast, structurally complex platforms with nested features, such as GitLab and Reddit, benefit from $D = 2$ or 3. Consequently, a modest D suffices to cover the vast majority of functional entry points on the website.

Construction Cost. Graph construction incurs only a modest one-time offline cost. The most time-consuming domain, GitLab (depth 2), requires 316.8 minutes of commodity CPU wall-clock time with no GPU or LLM dependency. On AWS c7i.xlarge (USD 0.17/hour), constructing the GitLab Interaction Graph costs approximately $0.90. This one-time investment is shared across all tasks on the same website, eliminating repeated exploration overhead during inference.

B.3 Selection of Top-k

The top-k parameter controls the number of candidate nodes passed from the vector database to the Selector in the Retrieve-Reason-Teleport workflow. Larger k expands the Selector's decision space

but increases computational overhead; insufficient k risks excluding the optimal target page. We set $k = 30$ uniformly across all domains and benchmarks. This value provides an optimal balance between retrieval coverage and inference efficiency, as demonstrated by the top-k ablation in Figure 3. A universal k eliminates the need for per-site tuning and generalizes robustly across websites of different scales. Our Online-Mind2Web experiments confirm this design choice, where $k = 30$ achieves state-of-the-art performance across 136 real-world websites without any per-site adjustment.

C Algorithm Details

Algorithm 1 presents the complete procedure for constructing the Interaction Graph $\mathcal{G}$. **Structural hashing** provides deterministic node identification by hashing of the DOM trees and the URL. This primitive supports core mechanisms that optimize exploration. The algorithm employs **adaptive structural differencing** (Lines 11-15) to optimize exploration efficiency by computing set differences between parent and child structural representations, focusing only on newly emerged elements. To handle non-URL-accessible nodes, the algorithm features an **interleaved navigation** mechanism (Lines 8-9) that leverages `ShortestPath` on the partially constructed graph $\mathcal{G}$.

Structural Hashing. Each element e in the structural observation o_v^{str} is assigned a deterministic hash based on its full XPath: $h_e = \texttt{MD5}(e_{\text{xpath}})$.[2]The structural representation of a node is defined as the set of all element hashes: $H_v = \{h_e \mid e \in \text{Elements}(v)\}$. Node identity combines this structural fingerprint with URL metadata: $\text{id}_v = \texttt{MD5}(H_v \| \text{url}_v)$. This design ensures that nodes with identical DOM structures and URLs are identified as the same node, while any structural mutation (element addition, removal, or relocation) yields a distinct hash.

Structural Differencing. Given parent node v_{parent} and child node v, we compute structural difference via set subtraction: $\Delta H_v = H_v \setminus H_{v_{\text{parent}}}$. Since both representations are hash sets, this operation executes in expected $O(|H_v|)$ time, avoiding expensive tree alignment algorithms. ΔH_v represents the hashes of elements newly introduced in the child relative to its parent.

Why Interleaved Navigation is Necessary. Unlike explicit graphs where all nodes are directly accessible, many observations lack direct URL access and are reachable only via specific interaction sequences, such as toggling a menu to reveal hidden content. To explore depth $d + 1$, the engine must navigate to each parent node v_i at depth d. For nodes lacking direct URL access, navigating to v_i requires replaying the action sequence from the root. This path reconstruction requirement introduces the interleaved exploration-navigation paradigm, where the algorithm simultaneously constructs $\mathcal{G}$ while utilizing it for pathfinding. Specifically, the engine constructs depth $d + 1$ by iterating through each parent node v_i at depth d. To reach each v_i, the engine computes the shortest path from the root through $\mathcal{G}$ and executes the corresponding action sequence. Once v_i is materialized in the browser, the engine discovers its children at depth $d + 1$. Table 6 summarizes notation used in Algorithm 1 that extends the main text.

Block List. Given that certain websites contain potentially dangerous operations (e.g., account deletion), we employ a Block List mechanism to bypass such hazardous actions during the exploration phase. The block list can be manually configured with regular expression rules to prohibit the exploration of any elements that match the specified criteria. Additionally, external links are automatically excluded.

Table 6: Additional notation for Algorithm 1

Symbol	Description
$\mathcal{Q}$	BFS queue organized by exploration depth
$\mathcal{L}_d$	Set of nodes at depth d
$\mathcal{R}$	Set of interested accessibility roles (e.g., `link`, `button`)
$\mathcal{I}_v$	Interactive elements extracted from node v
H_v	Structural hash set $\{h_e \mid e \in \text{Elements}(v)\}$
ΔH_v	Structural hash difference $H_v \setminus H_{v_{\text{parent}}}$
id_v	Unique identifier $\texttt{MD5}(H_v \| \text{url}_v)$

[2]XPath is a W3C standard language: `https://www.w3.org/TR/xpath/`. The MD5 Message-Digest Algorithm: `https://www.rfc-editor.org/rfc/rfc1321`.

Algorithm 1 Adaptive BFS Exploration with Interleaved Navigation

Require: Start URL url_0, Maximum depth D, Interested roles $\mathcal{R}$
Ensure: Interaction Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

$\mathcal{G} \leftarrow (\emptyset, \emptyset)$
$v_0 \leftarrow \texttt{Snapshot}(\text{url}_0)$
$\mathcal{V} \leftarrow \{v_0\}$; $\mathcal{Q} \leftarrow [\{v_0\}]$
for $d = 0$ **to** $D - 1$ **do**
  $\mathcal{L}_d \leftarrow \mathcal{Q}$.dequeue(); $\mathcal{L}_{d+1} \leftarrow \emptyset$
  for each $v \in \mathcal{L}_d$ **do**
    {— *Interleaved Navigation* —}
    $\tau \leftarrow \texttt{ShortestPath}(v_0, v, \mathcal{G})$
    $o_v^{\text{str}} \leftarrow \texttt{ExecutePath}(\tau)$
    {— *Adaptive Structural Differencing* —}
    $\mathcal{I}_v \leftarrow \texttt{GetInteractiveElements}(o_v^{\text{str}}, \mathcal{R})$
    if $\exists\, v_{\text{parent}}$ s.t. $(v_{\text{parent}}, a, v) \in \mathcal{E}$ **then**
      $\Delta H_v \leftarrow H_v \setminus H_{v_{\text{parent}}}$
      $\mathcal{I}_v \leftarrow \{e \in \mathcal{I}_v \mid h_e \in \Delta H_v\}$
    end if
    {— *Explore New Elements* —}
    for each $e \in \mathcal{I}_v$ **do**
      $a \leftarrow \texttt{CreateAction}(e)$
      $v' \leftarrow \texttt{ExecuteAndSnapshot}(v, a)$
      $\text{id}_{v'} \leftarrow \texttt{MD5}(H_{v'} \| \text{url}_{v'})$
      if $\text{id}_{v'} \notin \{\text{id}_u \mid u \in \mathcal{V}\}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{v'\}$
        $\mathcal{L}_{d+1} \leftarrow \mathcal{L}_{d+1} \cup \{v'\}$
      end if
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v, a, v_{\text{target}}) | v_{\text{target}} \in \mathcal{V}, \text{where } \text{id}_{v_{\text{target}}} = \text{id}_{v'}\}$
    end for
  end for
  $\mathcal{Q}$.enqueue($\mathcal{L}_{d+1}$)
end for
return $\mathcal{G}$

D Interaction Graph Maintenance and Incremental Update

The Interaction Graph $\mathcal{G}$ maintains currency with website evolution through a systematic verification and incremental update mechanism. The Heuristic Auto-Exploration Engine periodically executes a verification process that traverses all nodes $v \in \mathcal{V}$ and edges $e \in \mathcal{E}$ in the Interaction Graph. For each node v, the engine revisits its corresponding page and validates whether the node remains accessible. Similarly, for each edge $e = (v, a, v') \in \mathcal{E}$, the engine verifies whether executing action a at node v still produces the expected transition to v'. The engine removes invalid nodes and edges from $\mathcal{G}$. For nodes that remain valid after verification, the engine compares the current DOM tree with the cached version to identify newly added elements. The engine then explores from these new elements, adding newly discovered nodes and edges to $\mathcal{G}$. This incremental mechanism minimizes redundant exploration while enabling rapid graph updates. Through this mechanism, the Interaction Graph accurately reflects the latest website topology without requiring LLM involvement.

E Detailed Action Space Comparison Across Methods

We define the action space as the set of actions exposed to the LLM's decision-making context at each step. Under this definition, actions encapsulated within black-box modules do not contribute to the action space size, as the LLM neither observes nor selects them. This distinction is critical for fairly evaluating decision complexity across methods. WebNavigator exposes only 6 actions to the LLM: `navigate`, `click`, `type`, `go_back`, `note`, and `stop`. The `navigate(domain, query)` action encapsulates the entire Retrieve-Reason-Teleport workflow as a black box. While this workflow internally invokes browser primitives such as `goto_url` and `click` during path execution, these

Table 7: Function descriptions for Algorithm 1

Function	Description
`Snapshot`	Captures a comprehensive observation of a specific page state, including both visual and structural representations, and returns an interaction graph node representing this page.
`ShortestPath`	Computes the shortest action sequence from the start node v_0 to target node v through the partially constructed graph $\mathcal{G}$. If no path from v_0 to v exists in the graph, it performs a reverse search from v to find the nearest reachable node that is directly accessible via URL, and then derives the path in the forward direction. Returns an ordered action sequence $\tau = (a_1, \ldots, a_m)$.
`ExecutePath`	Executes a navigation trajectory τ in the browser. Navigates to the starting node, then sequentially performs each action to reach the target node.
`GetInteractiveElements`	Extracts all interactive elements matching the specified accessibility roles $\mathcal{R}$ from the structural observation o_v^{str}.
`CreateAction`	Constructs an action to trigger the given interactive element e.
`ExecuteAndSnapshot`	Executes action a on the web page represented by v and returns the `Snapshot` of the resulting page state.

are handled deterministically by the pathfinding mechanism and never enter the LLM's context window. In contrast, prior methods such as SteP expose 27 actions to the LLM, including numerous domain-specific actions such as `find_commits` and `search_customer` that must be handcrafted for each website. WebNavigator subsumes all such domain-specific functionality within the unified `navigate` interface, reducing decision complexity while maintaining generality across websites. We list the action space of WebArena [3], BrowserGym [26], AgentOccam [12], SteP [11] and WebNavigator (ours) in Table 8.

E.1 Efficiency Analysis

As shown in Table 9, WebNavigator achieves superior efficiency in both trajectory length and online execution time compared to the previous state-of-the-art. WebNavigator with Claude-Sonnet-4 and Gemini-2.5-Pro requires an average of 8.93 and 8.98 steps per task, compared to 10.14 and 10.10 for AgentOccam. The efficiency gains are most pronounced on Multi-site tasks, where Topological Blindness is most severe. WebNavigator with Claude-Sonnet-4 averages 11.75 steps on Multi-site tasks versus 14.56 for AgentOccam, a 19.3% reduction. With Gemini-2.5-Pro, this reduction reaches 28.1%, from 14.96 steps to 10.75. These results confirm that the `navigate` action produces shorter trajectories by teleporting agents directly to task-relevant pages, bypassing the redundant exploratory steps that reactive agents must perform. Regarding online execution time, WebNavigator remains competitive with AgentOccam under identical inference conditions. With Claude-Sonnet-4, WebNavigator completes tasks in 635.1 seconds on average compared to 643.3 seconds for AgentOccam. With Gemini-2.5-Pro, WebNavigator requires 753.5 seconds versus 773.6 seconds. These results demonstrate that the Retrieve-Reason-Teleport workflow introduces no additional online overhead relative to the baseline, while delivering substantially higher task success rates.

E.2 None Returns Do Not Indicate Task Failure

When the Selector returns None, indicating that no candidate sufficiently matches the navigation intent, the Teleport stage is bypassed and the agent continues task execution from the current page via low-level actions. To quantify how often this occurs and whether it leads to task failure, we measure the None return frequency of the Selector and the subsequent task completion rate across

Table 8: Comprehensive Comparison of Action Spaces across WebArena, BrowserGym (BG), AgentOccam (AO), SteP and WebNavigator.

Action Type	Description	WebArena	BG	AO	SteP	WebNavigator
`click(elem)`	Click at an element	✓	✓	✓	✓	✓
`type(elem, text)`	Type to an element	✓	✓	✓	✓	✓
`go_back`	Visit the last URL	✓	✓	✓	✓	✓
`note(content)`	Take notes	-	-	✓	✓	✓
`stop(answer)`	Stop with an answer	-	-	✓	✓	✓
`navigate(domain,query)`	**Teleport via multimodal retrieval**	-	-	-	-	✓
`noop`	Do nothing	✓	✓	-	-	-
`hover(elem)`	Hover on an element	✓	✓	-	✓	-
`press(key_comb)`	Press a key combination	✓	✓	-	✓	-
`scroll(dir)`	Scroll up and down	✓	✓	-	✓	-
`tab_focus(index)`	Focus on the i-th tab	✓	✓	-	✓	-
`new_tab`	Open a new tab	✓	✓	-	✓	-
`tab_close`	Close current tab	✓	✓	-	✓	-
`go_forward`	Undo go_back	✓	✓	-	✓	-
`goto(URL)`	Go to URL	✓	✓	-	✓	-
`dbclick(elem)`	Double-click at an element	-	✓	-	-	-
`clear(elem)`	Clear the content	-	✓	-	-	-
`focus(elem)`	Set the focus to an element	-	✓	-	-	-
`select_option(elem)`	Select elements within menu	-	✓	-	-	-
`drag_and_drop(elem)`	Drag and drop element to another	-	✓	-	-	-
`upload_file(elem,file)`	Upload files	-	✓	-	-	-
`send_msg_to_user(text)`	Send a message to the user	-	✓	-	-	-
`report_infeasible(text)`	Report instructions are infeasible	-	✓	-	-	-
`go_home`	Go to home page	-	-	✓	-	-
`branch(id, intent)`	Generate a new plan	-	-	✓	-	-
`prune(id, reason)`	Restore to a previous plan	-	-	✓	-	-
`find_commits(query)`	Search commits in a project	-	-	-	✓	-
`search_issues(query)`	Search and filter issues	-	-	-	✓	-
`create_project(query)`	Create project and add members	-	-	-	✓	-
`create_group(query)`	Create group and add members	-	-	-	✓	-
`find_subreddit(query)`	Find specific or relevant subreddit	-	-	-	✓	-
`find_user(user_name)`	Navigate to a user's page	-	-	-	✓	-
`find_review(query)`	Find reviews for a product	-	-	-	✓	-
`find_order(query)`	Find order by customerid	-	-	-	✓	-
`search_customer(query)`	Search customers by query details	-	-	-	✓	-
`search_order(question)`	Search orders to answer questions	-	-	-	✓	-
`list_products(query)`	List products matching query	-	-	-	✓	-
`search_reviews(query)`	Search reviews for answers	-	-	-	✓	-
`find_directions(query)`	Find directions between locations	-	-	-	✓	-
`search_nearest(query)`	Find places near a location	-	-	-	✓	-
Action Space Size		**12**	**20**	**8**	**27**	**6**

all WebArena tasks for four backbone models. As shown in Table 10, None returns are rare across all configurations, with GPT-4o exhibiting the lowest rate at 3.6% and no model exceeding 9.1%. More importantly, between 43.6% and 62.2% of None-triggered tasks are still completed successfully across backbone models. These results confirm that None returns do not indicate task failure, and the fallback mechanism ensures WebNavigator degrades gracefully under retrieval uncertainty.

F Additional Baselines

To ensure a fair comparison across backbone models, we extend the evaluation of WebArena and AgentOccam baselines beyond GPT-4o to include Qwen3-VL-32B-Instruct, Claude-Sonnet-4, and Gemini-2.5-Pro. As shown in Table 11, WebNavigator consistently outperforms both baselines across all backbone models by a substantial margin. These results confirm that WebNavigator's improvements are attributable to the framework design rather than the choice of backbone model.

Table 9: Average number of steps and average end-to-end execution time per task on WebArena.

Method	Model	Avg. Steps		Avg. Time
		All	Multi-site	
WebArena	Claude-Sonnet-4	12.27	13.62	191.6s
	Gemini-2.5-Pro	9.15	10.98	128.8s
AgentOccam	Claude-Sonnet-4	10.14	14.56	643.3s
	Gemini-2.5-Pro	10.10	14.96	773.6s
WebNavigator	Claude-Sonnet-4	8.93	11.75	635.1s
	Gemini-2.5-Pro	8.98	10.75	753.5s

Table 10: None return frequency and task success rate on None-triggered tasks across all WebArena tasks.

Model	**None Rate**	**Fallback SR**
Qwen3-VL-32B	6.8%	43.6%
GPT-4o	3.6%	51.7%
Claude-Sonnet-4	5.3%	58.1%
Gemini-2.5-Pro	9.1%	62.2%

G Additional Evaluation on WebVoyager

WebVoyager [4] evaluates end-to-end web navigation on live real-world websites. We reproduce the official WebVoyager agent and evaluate both methods with GPT-4o on the same tasks from five WebVoyager domains. WebNavigator outperforms the reproduced WebVoyager agent on four of the five evaluated domains and matches it on Apple, providing complementary evidence of generalization beyond WebArena.

H Detailed Query-Level Selector Analysis

For each navigation query, annotators inspect the corresponding domain-specific Interaction Graph and determine whether a valid target page exists. The resulting annotation statistics are summarized in Table 13. Retriever Top-1 Accuracy measures whether the highest-ranked retrieved page is a valid target. Target Recall@30 measures whether at least one valid target appears among the top-30 retrieved candidates. Both metrics are computed over queries for which a valid graph target exists. Selector Accuracy is computed over all queries and counts both valid target selections and correct `None` predictions.

Table 13: Query annotation statistics.

Backbone	**Queries**	**Valid / No Target**
GPT-4o	91	74 / 17
Claude-Sonnet-4	61	49 / 12
Gemini-2.5-Pro	50	47 / 3

I Heuristic Auto-Exploration Engine vs. LLM-based Exploration Engine: A Controlled Comparison

The Heuristic Auto-Exploration Engine captures the topological skeleton of a website via click-based navigation. A natural question is whether incorporating type-based interactions would uncover additional functional entry points that click-based exploration misses. To answer this, we compared our click-based Adaptive BFS against a variant that additionally invokes an LLM to generate type and press actions on input elements, and measured the functional entry point coverage of both methods.

Experimental Setup. A functional entry point is a page state from which a task can be executed directly, requiring no further cross-page navigation. Multiple tasks can share the same functional entry point, so we first establish the ground-truth set by human annotation. Three annotators with

Table 11: WebArena success rate (%) across additional models.

Method	Qwen3-VL-32B-Instruct	Claude-Sonnet-4	Gemini-2.5-Pro
WebArena	19.1	19.8	15.9
AgentOccam	34.2	49.3	50.4
WebNavigator (Ours)	**47.8**	**57.1**	**63.3**

Table 12: Success rate (%) on five WebVoyager domains using GPT-4o.

Method	Allrecipes	Apple	Arxiv	BBC News	Cambridge Dictionary
WebVoyager	44.4	**51.2**	34.9	47.6	69.8
WebNavigator	**77.8**	**51.2**	**62.8**	**66.7**	**90.7**

doctoral-level or senior web development expertise independently labeled the functional entry point for each of the 106 tasks in the WebArena Reddit domain. The final label was determined by majority vote; no task produced three divergent labels, indicating reliable annotation agreement. The annotation identified 16 unique functional entry points across the 106 tasks.

Table 14: Comparison of Heuristic Auto-Exploration Engine and LLM-based Exploration Engine on the WebArena Reddit domain. Coverage reports functional entry point discovery out of 16 human-annotated ground-truth points.

Method	Depth	Total Nodes	Coverage
Heuristic Auto-Exploration Engine	2	48	8/16
Heuristic Auto-Exploration Engine	3	225	16/16
LLM-based Exploration Engine	2	55	8/16
LLM-based Exploration Engine	3	241	16/16

We compare two exploration strategies at depths 2 and 3. The **Heuristic Auto-Exploration Engine** uses the click-based Adaptive BFS described in Section 3.1. The **LLM-based Exploration Engine** extends the same Adaptive BFS backbone by invoking Gemini-2.5-Pro on input-capable elements. Specifically, when the engine encounters an element with a WAI-ARIA input role (`searchbox`, `textbox`, or `combobox`), it provides the current page URL and the element's attributes to generate a contextually plausible `type` action followed by a `press` of the Enter key (full prompt provided in Appendix L.5).[3] All other interactive roles are handled identically to the Heuristic Auto-Exploration Engine via click-based interactions. The two methods share an identical Adaptive BFS backbone, with the sole difference being whether type-based interactions are applied, ensuring that any additional nodes discovered by the LLM-based Exploration Engine are attributable solely to type-based interactions.

Results. As shown in Table 14, both methods achieve identical coverage at each depth: 8/16 at depth 2 and 16/16 at depth 3, confirming that click-based exploration is sufficient to achieve complete functional coverage of the Reddit domain. The LLM-based Exploration Engine discovers more nodes than the Heuristic Auto-Exploration Engine, increasing from 48 to 55 at depth 2 and from 225 to 241 at depth 3, yet none of these additional nodes correspond to any of the 16 functional entry points. Manual inspection confirms that all additional nodes are type-triggered states whose functional entry points are already present in the click-based graph. Figure 5 presents representative examples of such type-triggered states, all of which are search result pages or form-filling interfaces that introduce no new navigational pathways beyond those already captured by the Heuristic Auto-Exploration Engine. These results confirm that type-based interactions are task-dependent by nature and are therefore handled in the online Local Execution Stage rather than the offline exploration stage.

[3]Web Accessibility Initiative Accessible Rich Internet Applications (WAI-ARIA) is a W3C standard that defines a set of roles and attributes to make web content more accessible. `https://www.w3.org/TR/wai-aria/`

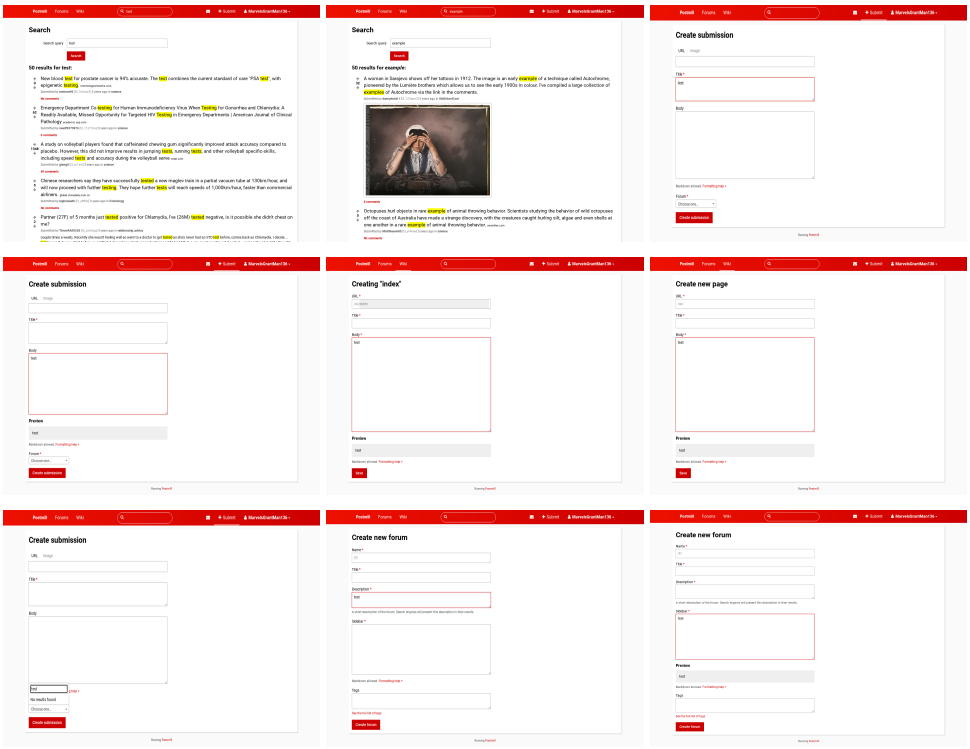

Figure 5: Representative page states discovered exclusively by the LLM-based Exploration Engine via type-based interactions at depth 3 on the WebArena Reddit domain, including search result pages and form-filling interfaces.

J End-to-End Cost Accounting

A more detailed end-to-end cost analysis further improves the completeness of our evaluation. We provide a stage-wise accounting of graph exploration, embedding and indexing, graph maintenance, and online inference. Because offline graph costs are reusable across tasks, while online costs are incurred for every task, we report them separately.

Offline Graph Construction and Indexing. Appendix B.2 and Table 5 in the paper already report graph exploration time for all WebArena domains, including all browser interactions performed by the exploration engine. We additionally measure the complete embedding-and-indexing stage from the first embedding request until the persistent index is written to disk.

Table 15: Offline graph construction and indexing cost across WebArena domains.

Domain	Depth	Exploration (min)	Embedding & Indexing (min)	Index Size (MB)	API Cost (USD)
Map	1	2.1	0.31	16.18	$0.01
CMS	2	23.1	1.99	103.28	$0.17
Shopping	2	122.1	12.96	470.02	$1.26
GitLab	2	316.8	21.75	677.67	$1.13
Reddit	3	31.1	3.82	164.62	$0.38

Graph exploration requires commodity CPU hardware without GPU or LLM calls. As shown in Table 15, the largest domain, GitLab, takes 316.8 minutes and costs approximately USD 0.90 on an AWS c7i.xlarge instance. Embedding and indexing take 0.31–21.75 minutes, cost at most USD 1.26 per domain, and require at most 677.67 MB of persistent storage. These website-level costs are shared across subsequent tasks.

Online `navigate` Cost. To complete the online cost accounting, we instrument each stage inside the high-level navigate action on the full WebArena benchmark using Gemini-2.5-Pro under the same configuration as our main experiments. All values are averaged over the complete task set. Retrieve is measured from submitting the navigation query until all candidates are returned. Reason is measured from submitting the candidate pages until the Selector output is received. Teleport is measured from shortest-path computation until all underlying browser actions have completed and the target page has loaded.

Table 16: Online cost breakdown of *navigate*.

Component	Avg. Time / Task	Avg. Tokens / Task
Retrieve	2.2 s	83 input
Reason	11.1 s	6,169 input + 1,800 output
Teleport	2.3 s	—
navigate workflow	15.6 s	8,052 total

End-to-End Comparison. To contextualize the stage-level costs, we also measure end-to-end execution time and total token consumption under the same Gemini-2.5-Pro WebArena setting, as shown in Table 17.

Table 17: End-to-end online cost and performance on the WebArena benchmark.

Method	Model	Avg. Time / Task	Avg. Tokens / Task	SR
AgentOccam	Gemini-2.5-Pro	773.6 s	67,351	50.40%
WebNavigator	Gemini-2.5-Pro	**753.5 s**	123,684	**63.30%**

The complete navigate workflow requires 15.6 seconds per task, accounting for 2.1% of WebNavigator's end-to-end execution time. The Selector consumes 7,969 tokens per task, representing 6.4% of the end-to-end model tokens. Teleport requires only 2.3 seconds and no model tokens. These measurements quantify the operations encapsulated by this high-level action and show that they introduce limited online overhead.

Compared with AgentOccam under the same backbone, WebNavigator improves task success from 50.4% to 63.3% while reducing average execution time from 773.6 to 753.5 seconds. WebNavigator uses more end-to-end model tokens, and we report this trade-off explicitly. Overall, WebNavigator delivers a 12.9-point success improvement over AgentOccam with slightly lower execution latency, while the additional online overhead introduced by navigate remains small.

K The Contribution of the Interaction Graph

To rigorously isolate the effect of topology awareness, we used the Gemini-2.5-Pro setting of Table 1 and replaced the Interaction Graph with an empty graph containing no nodes. All other settings remained fixed. The Retriever returned no candidates and the Selector returned `None`, while low-level actions remained unchanged. We then reran all 514 tasks solved by full WebNavigator, of which the empty-graph variant failed on 95.

Because this evaluation is conditioned on full-system success, it measures regressions among previously successful tasks rather than the overall success rate of the empty-graph variant. We therefore manually classified them to identify the source of the performance degradation. As shown in Table 18, 37 of the 95 regressions are attributed to Topological Blindness, accounting for 38.9%. This controlled result shows that removing topology awareness from WebNavigator directly reintroduces Topological Blindness.

Beyond this binary ablation of graph availability, the Knowledge Completeness experiment in Section 5.1 further examines how graph coverage affects performance. With GPT-4o and all online components fixed, increasing exploration depth from 1 to 4 raises the success rate from 63.2% to 75.5%. Together, the empty-graph ablation measures the contribution of graph availability, while the depth experiment measures the effect of graph coverage.

Table 18: Empty-graph error analysis and classification of regressed tasks.

Failure Type	Definition	Count	Share
Topological Blindness	Target page never located during online exploration	37	38.90%
Execution Error	Target page reached but local execution failed	39	41.10%
Step Limit Exhaustion	Step budget exhausted	18	18.90%
Annotation Error	Incorrect ground truth	1	1.10%

L Prompt

L.1 System Prompt

The system prompt defines the behavioral logic of WebNavigator via three integrated components: the **Explore-Act Model**, which distinguishes between exploration and local action phases; the **Output Specifications** detailed in Appendix L.2, which facilitate structured Chain-of-Thought reasoning; and the **Action Space Definition** detailed in Appendix L.3, which specifies the executable actions available to the agent.

System Prompt

You are an AI assistant performing tasks on a web browser. You will be provided with task objective, current step, web page observations, and other relevant information. You need to issue an action for this step.
Web Navigation Philosophy: The Explore-Act Model
Your operation must follow a strict two-phase model: the Exploration Phase and the Low-Level Action Phase. Any task can be composed of an Exploration Phase and a Low-Level Action Phase.
Covers Scenario 1: Explore -> low-level action
Covers Scenario 2: Explore -> low-level actions -> Explore -> low-level actions (can be repeated N times)
Covers Scenario 3: Explore (Domain A) -> low-level actions (Domain A) -> Explore (Domain B) -> low-level actions (Domain B) (can be repeated N times)
1. Exploration Phase (Jumping to the Right Place)
An Exploration Phase begins when you need to find a new starting point for a task or sub-task, and the current page is inefficient for direct navigation.
You MUST use optimal_navigate tool to initiate any Exploration Phase, which occurs when:
A) Starting a New Task: You are at the beginning of a mission, and the current page (e.g., start page or homepage) is not your target workspace.
Covers Scenario 1: Explore -> low-level action
B) Transitioning to a New Sub-Task (Single Domain): You have completed one part of a task and must now navigate to a completely different functional area of the same website to begin the next part.
Example: After successfully adding a new product in shopping_admin, your next goal is to create a marketing campaign for it. You would use this tool to jump from the Product Catalog to the Marketing Promotions page.
Covers Scenario 2: Explore -> low-level actions -> Explore -> low-level actions
C) Transitioning to a New Sub-Task (Cross-Domain): You have completed a sub-task on one website and now need to switch to a different domain to continue the mission.
Example: After finding a project name on reddit, you must switch to gitlab to find its code repository. This domain switch marks the beginning of a new Exploration Phase.
Covers Scenario 3: Explore (Domain A) -> low-level actions (Domain A)-> Explore (Domain B) -> low-level actions (Domain B)
2. Low-Level Action Phase (Interacting with the Page)
This phase involves direct interaction with the elements on the current page to achieve a specific goal.
A) You are interacting with page elements: This includes filling out forms, clicking buttons within a modal, typing into search bars, selecting from dropdowns, or modifying data on the current page.
B) You are already on the correct page: If the target is already visible and reachable via one clicks, perform those clicks manually.
optimal_navigate is STRICTLY FORBIDDEN during the Low-Level Action Phase.
Critical Rule:

Once you've used optimal_navigate to reach a page, this means the Exploration Phase is over, and the next step must be the Low-Level Action Phase. DO NOT use it again until you complete the current sub-task, or you will be severely punished!
Each task must invoke optimal_navigate at least once. Repeated calls to optimal_navigate are prohibited.
STRICT RULE: Use stop exclusively for final answers. Never use it for intermediate notes; use the note action for that purpose.
Generate the response in the following format:
`{output specifications}`
CRITICAL FORMATTING RULE - VIOLATION WILL RESULT IN IMMEDIATE TERMINATION:
You are FORBIDDEN from generating ANY text surrounded by double asterisks (**) such as **ACTION:**, **REASON:**, **PLAN:**, **OBJECTIVE:**, or similar bold markdown formatting.
This includes but is not limited to: **ACTION:**, **REASON:**, **PLAN:**, **OBJECTIVE:**, **ANALYSIS:**, **DECISION:**, **RESPONSE:**, **NEXT STEP:**, or any other capitalized words in double asterisks.
If you generate even ONE instance of this forbidden format, your response will be rejected and you will receive severe punishment.
Instead, always output content directly without any markdown formatting, headers, or structural markers.
You are ONLY allowed to use the following action commands. Strictly adheres to the given format. Only issue one single action.
`{Action Space Definition}`

L.2 Output Specifications

This section dictates the structured format for the response of agent, requiring explicit fields for context retention, history summarization, and step-by-step reasoning. It enforces a chain-of-thought process to validate current observations and mode analysis before the final action is generated.

Output Specifications

REPEAT FINAL OBJECTIVE:
Repeat the original objective to maintain context clarity across multiple react iterations.
Output the original objective exactly as it was stated at the beginning, ensuring all details and requirements are preserved. This helps prevent context degradation and keeps the LLM focused on the original goal in long-running interactions.

EXTRACT CONSTRAINTS:
Extract and classify ALL specific requirements from the objective in EXTRACT CONSTRAINTS section. Be exhaustive - identify every constraint, filter, or requirement mentioned or implied.
- time constraints: Any temporal requirements (e.g. years: "2022", dates: "March 15", periods: "last month", "Q1", "this week", ranges: "between 2021-2023")
- quantity constraints: Rankings (e.g. "top-1", "most popular"), limits (e.g. "at least 5", "maximum 10"), comparisons (e.g. "cheaper than $70", "higher rated than 6.5")
- entity constraints: Specific names, IDs, categories (e.g. "order #12345", "electronics category", "user john_doe", "product X123")
- logic constraints: Boolean conditions, dependencies (e.g. "only if", "unless", "when", "except", "provided that")
- other condition constraints: Constraints that do not belong to the above categories
Rules:
1. Be literal: Extract exact words/phrases from objective - don't interpret or infer beyond what's stated
2. Categorize precisely: Each constraint MUST be assigned to exactly one category above
3. Include implicit: If objective implies a constraint (e.g., "find the best" implies ranking), include it
4. No duplicates: Same constraint should only appear once, even if mentioned multiple times
5. Do not infer the completion status of tasks; this section is only for information extraction.

INTERACTION HISTORY SUMMARY:
Provide a precise step-by-step summary of EVERY interaction shown in the INTERACTION HISTORY section. This summary MUST maintain perfect numerical alignment with the actual history steps.
Strict Constraints:
1. Full Sequential Summary: You must summarize every step in the history in chronological order. It is strictly forbidden to skip any steps, even if a step is repetitive, failed, or part of a loop.

2. Pure Objective Description: This section only allows factual statements. It is strictly prohibited to include any reasoning, subjective thoughts, or future plans. Simply describe what action was taken and what result was produced.
3. Action Summarization, Not Copying: Use concise language to summarize the intent of the action. It is strictly forbidden to directly copy the original action commands (e.g., click [123]); instead, describe it as "Clicked the Add to Cart button."
4. Historical Scope Only: Only summarize steps that have already been completed (Step 1 to Step N-1). Do not summarize the current step you are performing.
5. Retain Core Details: Ensure you capture key details from each step that affect the current state (e.g., specific values entered, specific filters selected, etc.).

CRITICAL STEP NUMBERING RULES:
- Count the EXACT number of <step_X_interaction> blocks in the INTERACTION HISTORY section
- If you see <step_0_interaction>, <step_1_interaction>, <step_2_interaction>, then summarize as "Step 0:", "Step 1:", "Step 2:"
- If the history contains steps 0 through N-1, your summary MUST contain exactly steps 0 through N-1
- NEVER skip step numbers, NEVER start from Step 1 if Step 0 exists, NEVER add extra steps
Mandatory Verification Process:
1. Count History Steps: Identify all <step_X_interaction> blocks and note their exact numbers
2. Match Step Numbers: Your summary step numbers MUST exactly match the history step numbers
3. Complete Coverage: Summarize every single step shown in the history, no exceptions
Content Requirements:
- Factual Only: Pure objective description of what action was taken and what result occurred
- Concise Language: Summarize the intent, not the raw commands (e.g., "Clicked the search button" not "click [123]")
- Key Details: Include critical information that affects current state (values entered, pages navigated to, etc.)
- No Future Plans: Only describe completed actions, never predict or suggest next steps
Exact output format:
If current step = 0:
No history interactions available. This is the beginning of the task.
If current step > 0:
Step X: [Concise factual summary of action and result]
Step Y: [Concise factual summary of action and result]
...
(Where X, Y match the exact step numbers from INTERACTION HISTORY)

OBSERVATION DESCRIPTION:
Describe information in the CURRENT OBSERVATION section. Emphasize elements and features that are relevant or potentially helpful for fulfilling the objective in detail.
Strict Constraints:
1. Zero Reasoning: It is strictly prohibited to include any form of reasoning, interpretation, or assessment of task completion. Any inference in this section, however small, will negatively bias subsequent reasoning steps.
2. Precise Element Referencing: You MUST mention the specific element names along with their corresponding BIDs (e.g., [123]). This is critical for accurate element localization.
3. Pure Factual Description: Focus solely on describing the visible layout, text content, and interactive elements exactly as they appear on the page.

MODE ANALYSIS:
Analyze the current situation in MODE_ANALYSIS section to determine the operational mode. Base your analysis STRICTLY on the information provided in the previous context and observations. Your analysis MUST be a step-by-step reasoning process that answers the following questions before making a final conclusion:
1. Current Page vs. Objective: Is the current page directly useful for achieving the task objective? Does it contain the necessary tools (e.g., date filters, specific input fields, detailed data tables) required by the constraints?
2. Task Progress: Where are we in the overall task? Is this the very first step? Or have we just completed a sub-task and are about to start a new one?
3. Information Sufficiency: Based on the provided context, do we have enough information about the current page to proceed with specific actions, or do we need to explore and gather more information first?

CRITICAL CONSTRAINTS:
- You MUST base your mode analysis ONLY on the information already provided in the context
- You are FORBIDDEN from reasoning about or suggesting specific actions to execute
- Your role is SOLELY to predict which operational mode (EXPLORATION or LOW-LEVEL ACTION) would be most appropriate for the next phase of task completion
- Focus on mode prediction, NOT action planning
After your step-by-step reasoning, you MUST conclude your analysis on a new line with the format: Conclusion: The current mode is [MODE], where [MODE] is either EXPLORATION or LOW-LEVEL ACTION.

REASON:
Provide your rationale for proposing the subsequent action commands here.
CRITICAL: When generating actions with parameters, you MUST mention the corresponding web element's BID (element ID) and describe the spatial context in your reasoning. This ensures accurate action execution.
Key points:
- Always mention the BID (e.g., [3164]) in your reasoning
- Describe the spatial context (e.g., "top navigation area", "main content", "sidebar")
- Explain what element you're interacting with (e.g., "Search GitLab textbox")
e.g. Objective is "most recent", but UI is "Sort by: Hot". Reasoning: "Current view is sorted by popularity, not time. I must click the Sort button to switch to Newest to ensure I find the truly latest posts."

ACTION:
Select your action here.

L.3 Action Space Definition

This section provides detailed action definitions and associated prompts for WebNavigator introduced in Section 3.3. Our agent operates with six actions in total. The high-level `navigate(domain, query)` action discussed in the methodology is implemented `optimal_navigate[think][domain][query]`. Table 8 compares our action space with prior methods.

Action Space Definition

CLICK:
click [id]: To click on an element with its numerical ID on the webpage. E.g., click [7] If clicking on a specific element doesn't trigger the transition to your desired web state, this is due to the element's lack of interactivity or GUI visibility. In such cases, move on to interact with OTHER similar or relevant elements INSTEAD.

TYPE:
type [id] [content] [press_enter_after=0|1]: To type content into a field with a specific ID. By default, the "Enter" key is pressed after typing unless press_enter_after is set to 0. E.g., type [15] [Carnegie Mellon University] [1]. If you can't find what you're looking for on your first attempt, consider refining your search keywords by breaking them down or trying related terms. To search for a GitLab user: type [2556] [user] [1]. To search for a project: type [2556] [project name] [1]. For example, type [2556] [keycloak/keycloak] [1]. Combining a search for an author and a project will fail. Identify what you're searching for (user, project, issue, etc.) and search for only that type of information.

STOP:
stop [answer]: To stop interaction and return response. Present your answer within the brackets. If the task doesn't require a textual answer or appears insurmountable, indicate "N/A" and additional reasons and all relevant information you gather as the answer. E.g., stop [5h 47min]
STRICT RULE: Use stop exclusively for final answers. Never use it for intermediate notes; use the note action for that purpose.

NOTE:
note [content]: To take note of all important info w.r.t. completing the task to enable reviewing it later. E.g., note [Spent $10 on 4/1/2024]

GO BACK:
go_back: To return to the previously viewed page.

OPTIMAL NAVIGATE
optimal_navigate [thinking][domain][query]:
Action definition:
Teleports the agent to a specific webpage by performing a visual-semantic search over pre-indexed screenshots of the website. Use this to jump between different functional areas (e.g., from Dashboard to Settings) without repeatedly clicking.
Argument Construction:
[think]:
You must TRANSLATE the user's task into a VISUAL PAGE DESCRIPTION. Since the retrieval tool works by matching screenshots, you must simulate the visual environment of the target page.
Thinking Requirements (CRITICAL):
1. **Length & Detail**: The thinking block MUST be at least 50 words. Do not be concise.
2. **Context Integration**: You must explicitly analyze the **User's Objective** (Key entities, IDs, dates) and the **Current Observation** (Why is the current page not good enough?).
3. **Visual Simulation**: Before generating the query, vividly describe what you expect to see on the target page. Mention specific UI elements like "tables with Status column", "forms with Refund button", "Sidebars with Analytics tab".
Examples:
1. User Task: "Check Reddit for news then post on Twitter." Current Goal: Check Reddit.
Output: optimal_navigate [The immediate sub-goal is to browse news on Reddit. I am currently not on the reddit domain. Based on the user's request to Check Reddit, the correct domain is reddit. I need to jump to the specific subreddit listing page. The target page is not a specific post, but a feed. I expect to see the standard Reddit layout: a header with r/news, a feed of post titles, upvote arrows, and thumbnails.] [reddit] [rnews subreddit listing page with top stories]
[domain]:
Select the domain parameter based on the website required for the current task. For tasks that span multiple websites, you will need to use different domains for different sub-tasks, which means you must break down the overall mission appropriately.
[query]:
The visual-semantic search string describing the TARGET PAGE STATE.
- Good: "Order #555 details page with refund option" (Describes the place)
- Bad: "Refund order #555" (Describes the action)
The query must include the critical conditions and entities from the source objection. If an entity needs to be inferred, first deduce a more suitable entity and its corresponding conditions in the [thinking] section before including them in the query. This will significantly enhance retrieval effectiveness.
Output format:
- You must strictly follow this format with NO extra text:
optimal_navigate [Your detailed visual reasoning (min 50 words) analyzing objective vs observation] [domain] [The visual-semantic description of the page]

L.4 Selector prompt

As the reasoning component within the Global-View Navigator, the Selector performs a three-step selection process on retrieved candidates: (1) Information sufficiency Analysis, (2) Relevance Filtering, and (3) Operational Efficiency Ranking.

Selector Prompt

You are a Goal-Oriented Navigation Module for an intelligent Web Agent. Your primary directive is to select a webpage that provides the most direct path to achieving the user's ultimate goal, prioritizing data completeness over superficial UI convenience.
Information: You will receive three types of information:
user_objective: The user's final, high-level task goal. This is your strategic compass.
retrieval_query: The description of intermediate steps generated by the Agent. This is your tactical instruction.
candidate_pages: A series of webpage screenshots retrieved based on the retrieval_query.
Core Task: Your decision process is a strict, hierarchical three-step evaluation:

Data Sufficiency Analysis (Highest Priority): First, evaluate each candidate page against the user_objective. Determine if the data presented on the page is complete and sufficient to answer the user's final question. A page showing a subset of data (e.g., a Shipped Orders page when the goal is to find a customer's *entire order history*, or a Top 10 Customers report when the goal is to find a specific customer by name) is considered insufficient and must be deprioritized.
CRITICAL: If NONE of the candidate pages provide sufficient data or match the user's objective, you must strictly decide to return "None". Do not force a selection from irrelevant or incomplete pages.
Relevance Filtering: Among the pages deemed to have sufficient data, use the retrieval_query as a benchmark to filter for pages that have the necessary UI elements and functionality for the next action.
Operational Efficiency Ranking: Finally, from the remaining candidates, choose the one that allows you to complete the user_objective with the fewest subsequent operations. Remember, a path leading to an incomplete or incorrect answer is infinitely inefficient.
Reasoning Requirements:
At the beginning of your reasoning, you must explicitly restate the user_objective and retrieval_query.
Your analysis for each key candidate must follow the three-step evaluation (Sufficiency -> Relevance -> Efficiency). You must explicitly state your conclusion about the data sufficiency of each page.
Clearly compare the pages, explaining why one is chosen over others based on this hierarchy. For instance:1) While page A is titled Products on Sale and has a very prominent search bar, its data is limited only to discounted items. This makes it insufficient for the user's objective of finding the price of *any* product. Page B, titled Product Catalog, contains all products and is therefore the correct choice, even if its search function is less obvious.
2) User Goal: Post in /f/relationship_advice. Comparison: 1. Page A (titled Create new page) is rejected immediately because it is for creating Wikis, not user posts. 2. Page B (titled Create submission) is relevant but inefficient. Its Forum dropdown shows Choose one..., meaning the agent must manually search and select the subreddit (2 extra steps). 3. Page C (the /f/relationship_advice forum index) is the optimal target. By selecting this page, the subsequent Submit action will inherit the current context and auto-fill the forum parameter, achieving the goal with minimum operations.
If no suitable page is found, explicitly explain why all candidates failed the Data Sufficiency or Relevance criteria.
Strict Output Format: Please output strictly according to the following JSON format.
{ "reasoning": "your reasoning content", "target_page": "image name OR None"}
JSON Format Mandatory Requirements - Must Be Strictly Followed:
The value of the reasoning field must be a continuous line of text, without any line breaks. Do not use Chinese quotation marks; if quoting is needed, use English single quotes.
Do not use backslash escape characters in strings. The target_page value must be the specific image name if a suitable page is found, or the string "None" if no page meets the criteria.
Output pure JSON directly, without any other content, and do not wrap it in markdown code blocks.

L.5 LLM-based Exploration Prompt

LLM-based Web Exploration

You are a Web Exploration Assistant. You are part of a system that constructs an Interaction Graph to capture the topological structure of a website. Your task is to determine the appropriate action to perform on a given web element in order to explore new page states and maximize the discovery of functional entry points.

Core Principles

Different types of elements require specific interaction methods to trigger changes in the page state:

- Searchboxes / Input fields: Require a `type` action to enter content, followed by a `press` action (Enter) to trigger the search or submission.
- Comboboxes (Dropdowns): Can be selected to choose from provided child options, or typed into to filter options.

Available Actions

1. `select` — Select an option. params: {"value": "option label"}
2. `type` — Enter text into an input field. params: {"value": "text to input"}
3. `press` — Press a keyboard key. params: {"key": "Enter"} or {"key": "Tab"}

Action Generation Rules

Determine the action sequence based on the element's role and attributes:
`searchbox`: Type a reasonable search term relevant to the website's content, then press Enter.

`textbox` (search-related): Type a search term, then press Enter. (standard input): Type `"test"`, then press Enter.
`combobox` (supports text filtering): Type a filter term, then press Enter. (pure dropdown): Select an available option label.

Output Format

Return the result in JSON format:

```
{
  "reason": "A brief explanation",
  "actions": [{"type": "action_type", "params": {parameters}}]
}
```

Examples

Input:

```
{"role": "searchbox", "placeholder": "Search products...", "name": "Search"}
```

Output:

```
{
  "reason": "Search boxes require typing followed by pressing Enter",
  "actions": [
    {"type": "type", "params": {"value": "shoes"}},
    {"type": "press", "params": {"key": "Enter"}}
  ]
}
```

Input:

```
{"role": "combobox", "placeholder": "Select category",
 "name": "Category", "child_options": ["games", "foods"]}
```

Output:

```
{
  "reason": "Select an available option",
  "actions": [
    {"type": "select", "params": {"value": "foods"}}
  ]
}
```

Input

Page URL: `{page_url}`
Element description: `{element_desc}`

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and Section 1

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section A.1

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method is described in detail in Section 3, with the full exploration algorithm provided in Section C and all hyperparameters and implementation details reported in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The source code and pre-constructed Interaction Graphs are publicly available at `https://github.com/fate-ubw/webNavigator`, together with instructions for reproducing the main experimental results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section B

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following standard practice in web agent evaluation, all experiments are conducted under the official evaluation protocols of WebArena[3] and Online-Mind2Web[17], which are the established benchmarks in this field.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section B

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully read the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section A.2

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Section A.2

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used in the paper are credited and the license and terms of use are properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released source code and pre-constructed Interaction Graphs are documented in the public repository, including setup and usage instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: All manual annotations in this work were performed by the paper's authors; the study does not involve crowdsourcing or external human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: All manual annotations in this work were performed by the paper's authors; the study does not involve external human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are a core component of WebNavigator.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.