WEBSAPE: STRUCTURALLY AWARE AND PROGRESSIVELY EXPERIENCED WEB AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Current web agents lack mechanisms to autonomously acquire structural knowledge of website architecture or to accumulate operational experience over time, and consequently regress to blind exploration on complex tasks. To address this, we present **WebSAPE**, a Structurally Aware and Progressively Experienced web-agent framework. WebSAPE maintains a centralized knowledge store consisting of (i) per-site textual topology guides synthesized from website exploration, summarizing page functions, reusable operations, and expected destinations, and (ii) an experience repository of operational lessons distilled from historical trajectories. A progressive learning pipeline is built to update the knowledge one-shot and incrementally. At runtime, both artifacts are jointly retrieved and injected into the agent's context to facilitate its execution. Additionally, we introduce BizArena, a high-fidelity benchmark for web agents in enterprise productivity environments, with 100 curated tasks spanning email, calendar, chat, project management, and cross-app workflows within self-hosted environments under deterministic, instrumented evaluation. Experiments indicate that our WebSAPE surpasses the state-of-the-art academic and industrial methods on the WebArena ($+13.6$ pp), WebArena-Lite ($+10.3$ pp), WebVoyager ($+2.4$ pp), and BizArena ($+4.0$ pp) benchmarks, demonstrating that integrating structural priors with progressively distilled experience yields markedly more capable web agents. *Code and benchmark are publicly released at* `https://github.com/papersubmissionstore/WebSAPE`.

1 INTRODUCTION

The advent of large language models (LLMs) has catalyzed autonomous web agents (Prabhu et al., 2026; Yang et al., 2025; Browser-Use Team, 2024) capable of navigating interfaces and executing workflows (Zhou et al., 2024; He et al., 2024), holding the potential to fundamentally transform how people interact with digital services by automating complex, time-consuming online tasks on behalf of users. Nevertheless, these agents lack mechanisms to autonomously acquire structural understanding of target websites or to accumulate operational experience from previous executions, forcing them to approach every task from scratch (Xue et al., 2025). Recent research mitigates this via tool-learning (Prabhu et al., 2026), memory augmentation (Wang et al., 2025b), world models (Gu et al., 2025), and trajectory synthesis (Xu et al., 2025). Despite their merits, these methodologies share two limitations: they lack a priori structural information of target websites, and their acquired static experience lacks introspective mechanisms to progressively refine strategies across deployments. These shortcomings motivate **WebSAPE**, a Structurally Aware and Progressively Experienced web-agent framework, which uniquely integrates *structural knowledge* to obviate blind exploration with continuously distilled *operational experience*.

Specifically, WebSAPE autonomously crawls target websites via breadth-first search (BFS) to synthesize a structured topology guide, and concurrently maintains experience distilled from historical trajectories. Both artifacts together form the *WebSAPE Knowledge*, which is jointly retrieved and injected into the agent's context at runtime. To handle complex cross-site workflows under bounded LLM context, WebSAPE additionally adopts a hierarchical orchestration that dispatches sub-tasks, equipping both the parent planner and each child executor with the topology guides and experience entries pertinent to their scope. A progressive learning pipeline further extracts and contextually retrieves actionable insights, and updates the knowledge store in one-shot or incrementally. High-cost

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sites trigger topology refreshes, while high-cost or failed tasks trigger experience refreshes. Furthermore, to rigorously evaluate WebSAPE and address the scarcity of enterprise benchmarks, we introduce **BizArena**, a high-fidelity, deterministic testbed for web agents. BizArena features synthesized environments spanning email, calendar, chat, and project management, hosting 100 curated tasks that demand persistent state modifications and complex intra-app and cross-app orchestration. Leveraging a containerized infrastructure for millisecond-level resets, BizArena ensures strict reproducibility and state isolation. Crucially, its dual-constraint protocol audits both functional success and operational safety, penalizing unintended side effects like data corruption or accidental deletion.

Our principal contributions are summarized as follows:

- We propose the WebSAPE agent, which progressive learns beneficial knowledge through (i) a topology exploration pipeline that constructs structured navigation guides and (ii) a progressive experience learning method that distills operational lessons from historical trajectories, both unified under a knowledge workflow that couples pre-learning with incremental updates.
- We introduce BizArena, a deterministic 100-task benchmark spanning email, calendar, chat, and project management. It features atomic state isolation for strict reproducibility and a dual-constraint protocol that audits both functional success and operational safety to penalize unintended side effects.
- We demonstrate that the proposed WebSAPE agent achieves state-of-the-art performance across widely adopted benchmarks, including WebArena, WebArena-Lite and WebVoyager, as well as on our novel BizArena benchmark.

2 Related Works

Web Agent Architectures and Skill Acquisition. Early autonomous web agents primarily utilized reinforcement learning for instruction following (Shi et al., 2017; Liu et al., 2018). LLMs later catalyzed prompting-driven strategies, enabling multi-step sequential planning (Yao et al., 2023; Wei et al., 2022). Recent advancements scale these architectures to realistic environments via multimodal models (Zheng et al., 2024a; Hong et al., 2024), explicit navigation fine-tuning (Lai et al., 2024), and streamlined observation-action alignments (Yang et al., 2025), with proprietary systems transitioning into production (Anthropic, 2024; OpenAI, 2025). However, these frameworks generally assume deterministic execution, rendering them brittle to UI occlusions and Shadow DOM complexities. To improve robustness, recent work abstracts reusable functional primitives a posteriori from agent interaction, generating API-like tool abstractions (Zheng et al., 2025; Prabhu et al., 2026) or discrete programmatic skills (Wang et al., 2025a). Concurrently, trajectory synthesis pipelines augment the training corpus by replaying public web tutorials (Xu et al., 2025) or by structured exploration of target sites (Gandhi & Neubig, 2026). WebSAPE diverges by discovering structural knowledge via topology learning and progressively distilling operational experience across deployments, rather than relying solely on interaction-driven extrapolation or computationally expensive fine-tuning.

Memory and Experience in Agents. Iterative learning mechanisms are increasingly central to agent frameworks. Reflexion (Shinn et al., 2023) introduced verbal self-reflection for single-episode improvement, while subsequent architectures dynamically abstract and retrieve complex workflow patterns from historical trajectories (Wang et al., 2025b; Zheng et al., 2024b). WebEvolver (Fang et al., 2025) optimizes both world models and agents via a co-evolutionary framework, and recent multimodal agents systematically distill complex experiences into generalized programs (Sarch et al., 2024). WebSAPE distinguishes itself in two ways: (1) it maintains a structurally evolving experience repository across explicitly epochs—enabling robust cross-domain knowledge accumulation over localized reflections—and (2) it strictly couples this amassed operational experience with foundational topological knowledge, integrating an epistemic dimension conventionally ignored by purely memory-centric approaches.

Web Agent Benchmarks. The web agent evaluation landscape has expanded across diverse domains: simulated e-commerce (WebShop (Yao et al., 2022)), self-hosted functional environments (WebArena (Zhou et al., 2024)), live-website navigation (WebVoyager (He et al., 2024)), and enterprise IT administration (WorkArena, WorkArena++ (Drouin et al., 2024; Boisvert et al., 2024)).

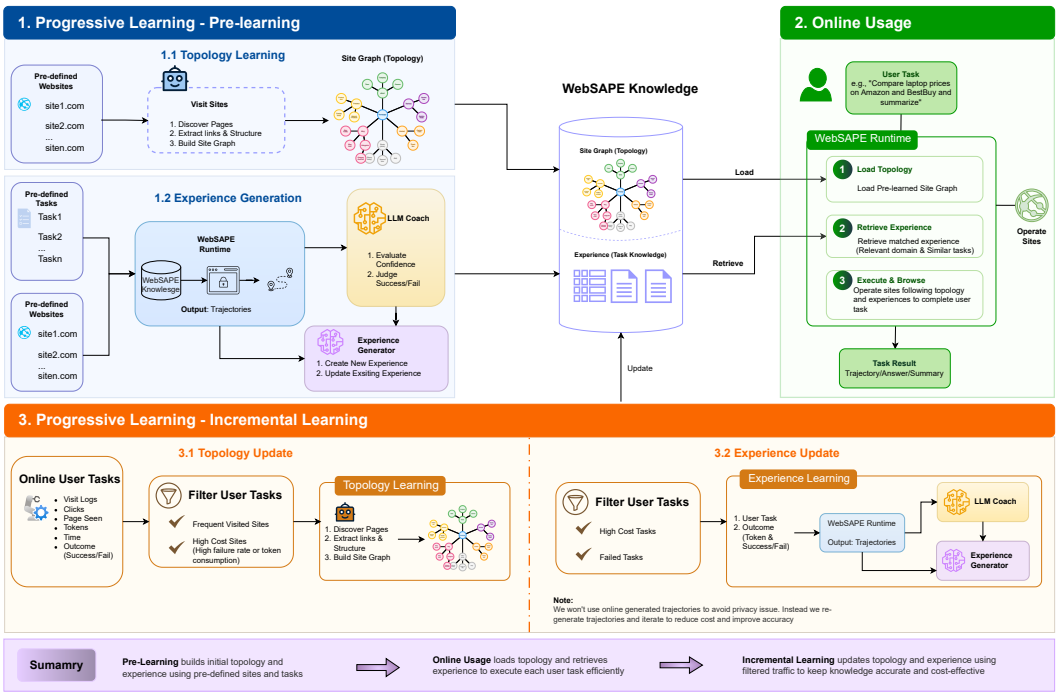


Figure 1: Overall framework of the proposed WebSAPE method.

However, a high-fidelity benchmark dedicated to the core enterprise productivity suite—spanning email, messaging, and project management—remains notably absent. BizArena fills this gap by instantiating a deterministic, containerized environment. Beyond expanding domain coverage, it addresses key limitations in existing benchmarks by enforcing atomic state isolation to prevent session-dependent leakage and introducing a safety-critical protocol to penalize unintended side effects.

3 METHOD

Figure 1 illustrates the WebSAPE framework. At its core, WebSAPE maintains a centralized *WebSAPE Knowledge* (Section 3.1) store comprising two complementary artifacts—a *site topology* graph that encodes each target website's structural prior, and an *experience repository* of operational lessons distilled from historical task trajectories—which together operationalize the "structurally aware" and "progressively experienced" aspects of WebSAPE, respectively. The shared knowledge is built in two ways: pre-learning with pre-defined tasks and sites to bootstrap the initial knowledge; incremental learning using conditionally filtered traffic to update both topology and experience. We illustrated how knowledge is built with *progressive learning* (Section 3.2). In *online usages* (Section 3.3), each user task is executed by retrieving the relevant topology and experience and dispatched through a hierarchical orchestration, in which the parent planner and children executors are based on an optimized Eko framework Fellou AI (2025).

3.1 WEBSAPE KNOWLEDGE

Site topology graph. The site topology graph is the structural component of the WebSAPE Knowledge: it encodes how a website is organized as a navigable state space, providing the agent with explicit structural awareness of the target site. Concretely, for each target site, WebSAPE maintains a directed graph $G = (V, R)$, where each vertex $v \in V$ represents a discrete page state defined by its canonical URL and a set of mapped interactive elements, and each directed edge $r = (v_i, v_j, a) \in R$ encodes the action a that triggers the transition from v_i to v_j, augmented by the accessibility-tree delta $\Delta(v_i, v_j)$. The graph is compiled into a *topology guide* that an LLM can consume directly.

Experience repository. Operational experience is rigidly structured as tuples $e = (s, \sigma, L)$, where s denotes the relevant web domain(s), σ provides a descriptive summary, and $L = \{\ell_1, \ldots, \ell_m\}$ comprises actionable lessons. The aggregate experience E accumulates prior knowledge while continuously integrating updates derived from recently processed trajectories. While the topology graph furnishes invariant structural bedrock about a website's layout, the experience repository encapsulates evolving operational intuition that complements it.

3.2 Progressive Learning

WebSAPE initials a progressive learning framework in which both topology and experience can be updated incrementally. This is crucial as when we use WebSAPE for online serving, the pre-built knowledge is limited. While in Figure 1, we also see pre-learning which allow us to fill knowledge one time as a bootstrap. WebSAPE is able to build knowledge progressively beacause *topology learning* and *experience generation* are site and task independent. *Topology learning* constructs the site topology graphs, and *experience generation*, which uses the agent itself together with an LLM coach to seed the experience repository.

Topology exploration and guide synthesis. Starting from a root page, WebSAPE performs bounded breadth-first exploration of the website's interaction graph. At each state, it observes the accessibility structure, identifies actionable controls, executes a selected subset, and records the resulting transitions. To limit repeated exploration, each control is assigned a semantic key derived from its action type and normalized accessible label. Controls already attempted in the current state are excluded, while controls whose semantic keys occurred at shallower exploration depths are suppressed. Besides, we further adopt a filter to remove controls that are evidently instance-specific or transient, such as individual messages, timestamps, and notification counters. Moreover, an LLM-based functional selector then retains controls representing reusable website operations, while a state-local ranker selects a bounded number of actions for expansion. We use a maximum exploration depth of $D = 3$ and a branch budget of $b = 15$ in our experiments. Finally, an LLM model synthesizes the explored graph into a textual navigation guide organized around page functions, available operations, and expected destinations. The guide preserves the operational structure of the graph while omitting redundant exploration traces and low-level state details. Please refer to Appendix A.1 for more details.

Experience generation. We formulate the experience space E as a learnable parameter space, and thus view experience generation as a parameter learning process. Given a user task $t \in T$, WebSAPE retrieves the current experience state E_i to generate a task-specific trajectory τ_t:

$$\tau_t = \text{WebSAPE}(t; E_i) \tag{1}$$

To generate effective experience, we employ an LLM acting as a *Coach* to provide supervisory feedback that indicates how well the generated trajectory τ_t aligns with the user task t, **without using ground-truth annotations**. For each τ_t, $\mathcal{L}_{coach}$ produces a success indicator c_t and an explanatory rationale r_t. The objective is to maximize task success over the task distribution T. Instead of computing a true gradient, WebSAPE leverages an LLM to generate experience update:

$$\Delta E_i = G(E_i, t, \tau_t, \mathcal{L}_{coach}(\tau_t)) \tag{2}$$

where G denotes the Experience Generator, and E_i represents the experience state at the i-th iteration. Here, we treat $\mathcal{L}_{coach}$ as the loss function, where successful trajectories with confident rationales yield low loss, and unsuccessful ones yield high loss. This allows experience to be iteratively refined in a weight-tuning-like manner, progressively reducing task execution errors. The update follows an experience-gradient-descent abstraction:

$$E_{i+1} = E_i + \eta \Delta E_i \tag{3}$$

where η acts as a gating filter implemented by an LLM that determines whether an experience update is reliable enough to be incorporated. Concretely, it governs which lessons in the experience state E_i should be added, removed or modified, thereby ensuring the stable evolution of experience over time. The effectiveness of the learning loop is analyzed in section 5.3

Incremental update. In *online usage* (Section 3.3), we collect telemetry based on real user tasks —visit logs, click streams, page states, token cost, and final outcomes. Topology is created, updated on frequently visited sites and high-cost sites (e.g., sites with elevated failure rates or token consumption). Similar filters are also used to retain high-cost and unresolved tasks, which are then re-executed with the existing experience E_i to generate trajectories. Emitting updates ΔE_i that are merged into next epoch's experience to achieve cost saving and accuracy improve. Through this synergy between pre-learning and incremental updates, WebSAPE commits continuous self-evolution, systematically narrowing the performance gap as the volume of operational deployment scales.

3.3 Online Usage: Per-Task Inference

Every incoming user task t traverses two runtime stages:

(1) **Knowledge retrieval.** WebSAPE agent first parses t to identify the target websites and retrieves the corresponding topology guides from the knowledge base, encapsulating each within structured XML demarcation for injection into the system prompt. In parallel, it retrieves the experience repository for the k most contextually salient entries $E_k \subset \mathrm{E}$ based on a normalized embedding similarity score (we set $k = 3$ empirically). Together, the loaded topology and the retrieved experience tuples form the runtime context furnished to the agent prior to execution.

(2) **Hierarchical execute.** Equipped with both the topology guides and the retrieved experience tuples, the WebSAPE runtime drives the browser to complete user tasks. As injecting full topology guides for every potentially relevant website into a single monolithic agent context would precipitously exhaust the context window, the runtime is hierarchically orchestrated: a centralized *parent planner* ingests the user objective together with topology and experience covering the entire task, and segments it into focused sub-tasks dispatched to *child executors*; each executor receives only the topology guide and experience tuples scoped to its sub-task. As a result, both the parent planner and every child executor operate with structural and experiential knowledge appropriate to their respective scopes, while the planner aggregates concise executor summaries and triggers continuations or re-planning until the task is fulfilled.

4 The BizArena Benchmark

BizArena is a high-fidelity benchmark tailored for evaluating web agents within enterprise productivity environments (Figure 2). While existing benchmarks have advanced agentic capabilities in general web navigation (Zhou et al., 2024; He et al., 2024) and IT administration (Drouin et al., 2024; Boisvert et al., 2024), they offer limited coverage of the **interconnected enterprise stack**. This limitation is substantial, given that employees spend over 30% of their working hours operating within these environments (Microsoft, 2023). We design the benchmark around four fundamental pillars: (1) comprehensive coverage of core workplace functions (Email, Calendar, Chat, and Project management); (2) a diverse task suite encompassing both deep intra-app and cross-app state transitions; (3) self-hosted web environments supported by a deterministic, containerized infrastructure; and (4) an evaluation protocol that rigorously audits both functional success and operational safety. The BizArena benchmark is publicly released at `https://github.com/papersubmissionstore/WebSAPE`.

4.1 Deterministic & Instrumented Infrastructure

BizArena provides self-hosted, containerized environments featuring synthesized web apps that closely mimic core enterprise software. These environments are designed to balance high-fidelity interaction with strict reproducibility: (1) **High-Fidelity but Controllable Environments.** The environments preserve authentic UI layouts and interaction patterns, allowing agents to navigate realistic state transitions without the unpredictability, UI drift, and rate limits inherent in live production systems. Furthermore, all underlying website data and event logs are systematically recorded, forming a fully controllable and trackable environment for agent evaluation. (2) **Fast State Reset.** To mitigate state leakage across trials, where residual data from prior episodes contaminates subsequent evaluations, BizArena provides a rapid reset mechanism. This restores the environment to an invariant seed state in milliseconds before every query, ensuring unbiased performance benchmarking. (3) **Temporal Snapping.** Workplace tasks often rely on relative temporal references (e.g., "follow

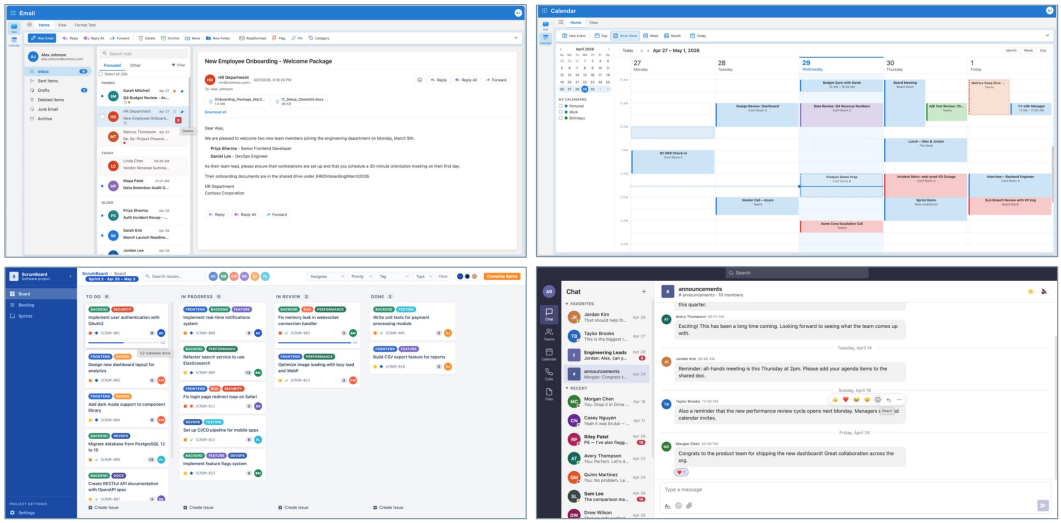


Figure 2: Synthesized enterprise productivity apps in BizArena.

up on yesterday's chat"). We implement relative temporal snapping, which synchronizes the internal system clock with the seed data's timeline. This prevents time-sensitive tasks from becoming logically inconsistent as real-world time progresses.

4.2 Task Synthesis and Multi-App Workflows

BizArena features 100 curated tasks requiring persistent state modifications (e.g., `Create`, `Reply`, `Schedule`, `Assign`) rather than mere information retrieval. The suite comprises 33 Email/Calendar tasks, 16 Chat tasks, and 25 Project Management tasks. The remaining 26 cross-app tasks demand orchestration across the entire stack. With trajectories averaging 8 steps (extending to 10+ for cross-app workflows), tasks require agents to perform complex synthesis—such as extracting a postmortem from a chat thread, filing a project issue, and notifying stakeholders via email. This closely reflects the true complexity of real-world professional workflows.

4.3 Evaluation: Functional Success vs. Operational Safety

To rigorously assess agent performance, BizArena employs a deterministic, state-based evaluation protocol. By directly inspecting backend databases and system logs, our framework evaluates agents across two complementary dimensions: functional correctness and operational safety.

Functional Correctness & Granular Progress. We evaluate whether the agent successfully achieved the user's intent. Because enterprise workflows involve long-horizon trajectories, we provide both a strict *Success Rate* (binary) and a granular *Completion Score* (partial). Formally, for task q, let $\mathcal{G}_q = \{g_{q,1}, g_{q,2}, \ldots, g_{q,n_q}\}$ denote its n_q rubric checks. The completion score $C(q)$ is the fraction of satisfied checks, while the binary success $S(q)$ is conjunctive:

$$C(q) = \frac{1}{n_q} \sum_{i=1}^{n_q} \mathbb{I}(g_{q,i} \text{ is satisfied}), \quad S(q) = \prod_{i=1}^{n_q} \mathbb{I}(g_{q,i} \text{ is satisfied}) \tag{4}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Thus, $S(q) = 1$ if and only if every check in $\mathcal{G}_q$ is satisfied; otherwise, $S(q) = 0$. The completion score instead assigns partial credit for satisfied checks, enabling precise diagnosis of where agents falter.

Preservation constraints. We partition $\mathcal{G}_q$ into goal-achievement checks $\mathcal{P}_q$, which assess the intended task outcomes, and, when applicable, preservation checks $\mathcal{N}_q$, which detect unintended state changes, such as leaving non-target messages, conversations, or project records unchanged. This way, $\mathcal{G}_q = \mathcal{P}_q \cup \mathcal{N}_q$. Because binary success is conjunctive, a failed check in either subset makes $S(q) = 0$; hence, every preservation check has veto power over success, and satisfying

Table 1: Success rate (%) across benchmarks. Top rows report the numbers published in each paper; bottom rows demonstrate the performance with the *identical LLM* model, i.e., Claude Opus 4.5.

Setting	Method	WebArena	WebArena-Lite	WebVoyager	BizArena
Published configurations	WebVoyager (He et al., 2024)	–	–	59.1	–
	WILBUR (Lutz et al., 2024)	–	–	52.6	–
	Agent-E (Abuelsaad et al., 2024)	–	–	73.1	–
	AgentOccam (Yang et al., 2025)	43.1	–	54.3[†]	–
	AgentTrek (Xu et al., 2025)	22.4	–	–	–
	ASI (Wang et al., 2025a)	40.4	–	–	–
	AWM (Wang et al., 2025b)	35.5	–	–	–
	Hybrid Agent (Song et al., 2025)	38.9	–	–	–
	Plan-and-Act (Erdogan et al., 2025)	48.2	57.6	81.4	–
	SkillWeaver (Zheng et al., 2025)	29.8	–	–	–
	WebPilot (Zhang et al., 2025)	37.2	–	–	–
	WebRL (Qi et al., 2025)	–	49.1	–	–
	Go-Browse (Gandhi & Neubig, 2026)	45.4	–	–	–
	R-WoM (Mei et al., 2026)	34.6	–	–	–
	WALT (Prabhu et al., 2026)	50.1	–	–	–
	Web-Shepherd (Chae et al., 2026)	–	39.4	–	–
Identical LLM Claude Opus 4.5	AgentOccam (Yang et al., 2025)	44.7	53.9	57.4	51.0
	WALT (Prabhu et al., 2026)	57.0	63.0	68.7	59.0
	Browser-Use (Browser-Use Team, 2024)	57.4	62.4	81.7	73.0
	Claude Code (Anthropic, 2025)	60.3	64.2	76.7	69.0
	WebSAPE (ours)	**63.7**	**67.9**	**83.8**	**77.0**
	w/o Exp.	57.8	63.0	81.3	69.0
	w/o Exp., w/o Topo.	55.5	61.2	79.9	65.0

[†] AgentOccam reported its result on an 129-task subset of WebVoyager, on which WebSAPE achieves 78.3%.

additional goal-achievement checks cannot compensate for its violation. Both subsets contribute to $C(q)$, so a failed preservation check also lowers completion.

5 Experiments

5.1 Main Results

Comparison with published performance. In the experiments, we evaluate WebSAPE against a broad set of state-of-the-art web agents from both academia (e.g., WALT (Prabhu et al., 2026), AgentOccam (Yang et al., 2025), Plan-and-Act (Erdogan et al., 2025)) and industry (e.g., Browser-Use (Browser-Use Team, 2024), Claude Code (Anthropic, 2025)) on the commonly used benchmarks *WebArena* (Zhou et al., 2024), *WebArena-Lite* (Liu et al., 2025), *WebVoyager* (He et al., 2024). The top rows in Table 1 shows numbers published in each method's paper. It can be seen that the proposed WebSAPE outperforms the best performance in the existing literature by $+13.6$ pp, $+10.3$ pp, and $+2.4$ pp on WebArena, WebArena-Lite and WebVoyager, respectively.

Performance with identical LLM model. To disentangle agent framework contributions from LLM capacity, we re-evaluate the open-sourced SOTA academic[1] and industrial baselines with the *identical* Claude Opus 4.5 model as WebSAPE. As shown in the bottom rows of Table 1, WebSAPE reaches **63.7%** on WebArena, ahead of Claude Code (60.3%), Browser-Use (57.4%), WALT (57.0%), and AgentOccam (44.7%). It also reaches **67.9%** on WebArena-Lite, **83.8%** on WebVoyager, and **77.0%** on BizArena, outperforming all compared methods.

5.2 BizArena benchmark

Table 2 presents the results of success rate and completion scores on BizArena, with the *identical* Claude Opus 4.5. WebSAPE achieves the highest Overall Success Rate (**77.0%**) and Completion

[1] Plan-and-Act (Erdogan et al., 2025) trains its own models, so we did not replace them with other LLM.

Table 2: The proposed BizArena benchmark with success rate (%) and completion score (%) (in parentheses) metrics. All methods use the *identical LLM* of Claude Opus 4.5.

Method	Chat	Email & Calendar	Project Mgmt	Cross-App	Overall
AgentOccam	12.5 (59.6)	60.6 (73.7)	72.0 (77.3)	42.3 (72.1)	51.0 (71.9)
WALT	50.0 (72.6)	57.6 (63.1)	**88.0** (91.3)	38.5 (62.2)	59.0 (71.5)
Browser-Use	81.3 (87.7)	**78.8 (88.1)**	76.0 (84.5)	**57.7 (82.9)**	73.0 (85.8)
Claude Code	68.8 (91.2)	75.8 (81.8)	**88.0 (92.9)**	42.3 (79.2)	69.0 (85.4)
WebSAPE (ours)	**93.8 (97.9)**	**78.8** (85.1)	**88.0** (91.3)	53.8 (80.6)	**77.0 (87.5)**

Score (**87.5%**), outperforming all four baselines on both metrics. Its overall success rate exceeds Browser-Use and Claude Code by 4.0 and 8.0 percentage points, respectively. WebSAPE leads Chat with **93.8%** success and **97.9%** completion, ties Browser-Use for the highest Email & Calendar success rate (**78.8%**), and ties WALT and Claude Code for the highest Project Management success rate (**88.0%**), while attaining 91.3% completion in Project Management. On Cross-App tasks, WebSAPE reaches 53.8% success and 80.6% completion, outperforming AgentOccam, WALT, and Claude Code on both metrics. Together, these results highlight WebSAPE's strong task-completion performance across enterprise productivity workflows. Additionally, WebSAPE achieves the best success rate with the least agent steps (reported in Table 5 in *Appendix* B), indicating that the WebSAPE knowledge effectively guides the execution of web agent.

5.3 Ablation Studies

Numerical results. In Table 1, we also ablate the experience (Exp.) and topology guides (Topo.) on WebArena, WebArena-Lite, WebVoyager and the proposed BizArena. Starting from the full WebSAPE (**63.7%** on WebArena, **67.9%** on WebArena-Lite, **83.8%** on WebVoyager, **77.0%** on BizArena), removing experience alone drops WebArena to 57.8% (-5.9%), WebArena-Lite to 63.0% (-4.9%), WebVoyager to 81.3% (-2.5%), and BizArena to 69.0% (-8.0%), demonstrating that progressively distilled lessons carry tangible value even on top of a strong topology prior. Further removing the topology guides yields 55.5% on Webarena (-2.3%), 61.2% on WebArena-Lite (-1.8%), 79.9% on WebVoyager (-1.4%), and 65.0% on BizArena (-4.0%), indicating that the structural guides remain a dominant contributor on both settings. This indicates that the joint contribution of structural topology and experience significantly advance our WebSAPE.

Analysis on Progressive Experience. Figure 3 visually demonstrates the individual contributions of the proposed topology guide and progressive experience. Specifically, the blue curve illustrates how experience evolves within the progressive learning framework. At each sequential epoch i, the Experience Generator (G) analyzes execution trajectories and their evaluations by the LLM coach from epoch $i-1$ to construct an updated experiential store E_i, which is inherited by subsequent tasks. As shown in Figure 3, on WebArena-Lite the success rate climbs from 63.0% at Epoch 0 (no experience) to a convergence plateau of 67.9% at Epoch 3 as experience is refined across epochs. Crucially, this experience learning process **does not rely on any ground-truth** supervision; the LLM Coach alone steers WebSAPE in an improving direction. As a result, the progressively distilled experience brings WebSAPE remarkably close to the performance achieved with ground-truth supervision (red dashed line in Figure 3). This analysis verifies the self-evolution capability of the proposed experience learning loop.

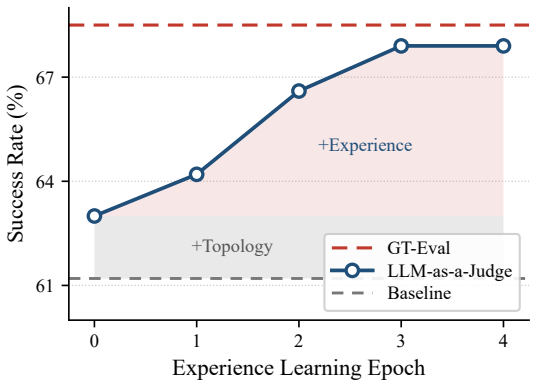


Figure 3: Success rate with topology guide and progressive experience epochs on WebArena-Lite.

Case study. We use a case to demonstrate the effectiveness of WebSAPE, when executes task *"Show me products under $30 in 'men shoes' category"* in WebArena-Lite, WebSAPE loads the shopping topology, which records explicit selector entries such as *"click '#ui-id-77'* $\rightarrow$ *Clothing, Shoes & Jewelry"*, allowing the agent to reach the target category in deterministic click. In parallel, the retrieved experience describes *"directly construct the URL with a consolidated price parameter (e.g.,* `?price=x-y`*)"* for price filtering; the agent applies this verbatim, issuing a single *navigate_to* with *"?price=0-30"* appended to the URL, and obtains 523 correctly filtered products. Compared with WebSAPE w/o Exp., w/o Topo., the full WebSAPE finishes the task successfully while consuming 29% fewer browse actions and 53% fewer input tokens. The case crisply illustrates the effectiveness of the two artifacts: the topology localizes *where* to act, while the experience refines *how* to compose actions efficiently.

5.4 Generalizability of Experience via Cross-Validation

To verify the generalizability of WebSAPE experience, we use a two-direction cross-validation design within WebArena (WA). We partition its WebArena into *non-overlapping* subsets WA-Lite ($\mathcal{Q}_L$) and WA w/o Lite ($\mathcal{Q}_R$), with $\mathcal{Q}_L \cap \mathcal{Q}_R = \emptyset$. Each experience store is evaluated on both test splits. In Table 3, rows indicate the learning set used for experience distillation, while columns report success rate on the test splits. For each test split, we also report the baseline without experience and the same-split reference (where experience is distilled from the test split itself, thus containing learning–test overlap). The off-diagonal results directly assess the generalizability and transferability of operational experience across non-overlapping task sets.

Table 3: Cross-validation of experience across non-overlapping WebArena splits ($\mathcal{Q}_L \cap \mathcal{Q}_R = \emptyset$).

Experience-learning set	**Test: WA w/o Lite**		**Test: WA-Lite**	
	SR (%)	Δ	SR (%)	Δ
None (w/o Exp.)	56.3	–	63.0	–
WA-Lite ($\mathcal{Q}_L$)	**60.9**	**+4.6**	67.9*	+4.9
WA w/o Lite ($\mathcal{Q}_R$)	63.4*	+7.1	**69.1**	**+6.1**

*Same-split reference with learning–test task overlap. **Bold** marks held-out cross-task transfer to *unseen* tasks.

On WA w/o Lite, experience learned from WA-Lite yields 60.9% success, compared with 56.3% without experience (Table 3), achieving a substantial 4.6-percentage-point gain on unseen tasks. The same-test-set reference, which uses experience learned on WA w/o Lite, reaches 63.4%, showing that the cross-task experience recovers the majority of the performance gain. In the reverse direction, experience learned from WA w/o Lite achieves 69.1% success on WA-Lite, improving by 6.1 percentage points over the 63.0% result without experience and by 1.2 points over the 67.9% same-test-set reference learned on WA-Lite. This higher performance is likely attributable to the larger scale of WA w/o Lite (647 tasks versus 165 in WA-Lite), as distilling from a broader collection of tasks exposes the agent to richer and more diverse operational patterns that benefit downstream execution. Together, these bidirectional results demonstrate the strong generalizability of WebSAPE's distilled experience, confirming that it captures portable operational heuristics rather than overfitting to specific training tasks.

6 Conclusion

We presented **WebSAPE**, a Structurally Aware and Progressively Experienced web-agent framework that maintains a unified *WebSAPE Knowledge* store of per-site topology graphs and an experience repository, jointly retrieved at runtime via a lightweight hierarchical orchestration and refreshed by a progressive pipeline from conditionally filtered live traffic. We further release **BizArena**, a deterministic 100-task benchmark spanning email, calendar, chat, project management, and cross-app workflows, which we believe will help the community optimize web agents for enterprise scenarios. Across WebArena, WebArena-Lite, WebVoyager, and BizArena, WebSAPE attains state-of-the-art performance under both per-baseline configurations and the identical Claude Opus 4.5 model.

AI USE STATEMENT

Generative AI tools were used only to aid or polish the writing of this paper, including grammar, spelling, and typographical corrections. They were not used to generate research ideas, conduct experiments, analyze results, or substantively produce the scientific content. The authors reviewed and take full responsibility for the manuscript.

ETHICS STATEMENT

The authors have read and adhered to the ICLR Code of Ethics. This work does not involve human subjects or personally identifiable data, and we are not aware of any ethical concerns raised by the research, experiments, benchmark, or release.

REPRODUCIBILITY STATEMENT

We anonymously release the complete source code and the proposed BizArena benchmark at `https://github.com/papersubmissionstore/WebSAPE`. The repository includes the implementation, benchmark tasks and environments, evaluation procedures, configurations, and instructions necessary to reproduce the reported results.

REFERENCES

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.

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. `https://www.anthropic.com/news/3-5-models-and-computer-use`, 2024. Accessed: 2025-09-07.

Anthropic. Claude Code. `https://github.com/anthropics/claude-code`, 2025. Accessed: 2026-09-08.

Anthropic. Pricing. `https://platform.claude.com/docs/en/about-claude/pricing`, n.d. Claude API documentation. Accessed: 2026-09-09.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault de Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. WorkArena++: Towards compositional planning and reasoning-based common knowledge work tasks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5996–6051, 2024.

Browser-Use Team. Browser-use: Enable AI to control your browser. `https://github.com/browser-use/browser-use`, 2024. Accessed: 2025-01-27.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-shepherd: Advancing PRMs for reinforcing web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL `https://openreview.net/forum?id=G2kMroO9UV`.

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

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, 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*, 2025.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. WebEvolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8970–8986, 2025.

Fellou AI. Eko: A production-ready agentic workflow framework, 2025. URL https://fellou.ai/eko/docs/. Accessed May 2026.

Apurva Gandhi and Graham Neubig. Go-Browse: Training web agents with structured exploration. In *International Conference on Learning Representations*, 2026.

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.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 6864–6890, 2024.

Wenyi Hong et al. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. AutoWebGLM: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5295–5306, 2024.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*, 2018.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Song XiXuan, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. VisualAgentBench: Towards large multimodal models as visual foundation agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2snKOc7TVp.

Michael Lutz, Arth Bohra, Manvel Saroyan, Artem Harutyunyan, and Giovanni Campagna. WILBUR: Adaptive in-context learning for robust and accurate web agents. *arXiv preprint arXiv:2404.05902*, 2024.

Kai Mei, Jiang Guo, Shuaichen Chang, Mingwen Dong, Dongkyu Lee, Xing Niu, and Jiarong Jiang. R-WoM: Retrieval-augmented world model for computer-use agents. In *International Conference on Learning Representations*, 2026.

Microsoft. Will AI fix work? 2023 Work Trend Index annual report. https://www.microsoft.com/en-us/worklab/work-trend-index/will-ai-fix-work, 2023. Accessed: 2024-05-01.

OpenAI. Computer-using agent. https://openai.com/index/computer-using-agent/, 2025.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, Silvio Savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *International Conference on Learning Representations*, 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Gabriel Herbert Sarch, Lawrence Jang, Michael J Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yueqi Song, Frank F Xu, Shuyan Zhou, and Graham Neubig. Beyond browsing: Api-based web agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11066–11085, 2025.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. AgentTrek: Agent trajectory synthesis via guiding replay with web tutorials. In *International Conference on Learning Representations*, 2025.

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. In *Second Conference on Language Modeling*, 2025.

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 *International Conference on Learning Representations*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

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

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23378–23386, 2025.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024a.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, 2024b.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024.

A Appendix

A.1 Topology Construction Details

State and transition representation. An interface state is identified from a normalized page location, active overlay context, salient accessibility landmarks, and the set of actionable controls. Abstractly,

$$v = \mathrm{Hash}\big(u, o, \mathcal{L}, \{(a_j, \ell_j, s_j)\}_{j=1}^{m}\big)\,, \tag{5}$$

where u is the normalized URL, o describes an active overlay, $\mathcal{L}$ denotes salient landmarks, and (a_j, ℓ_j, s_j) are the action type, normalized accessible label, and verified interaction target of control j. Volatile URL fields, timestamps, counters, and other transient text are normalized to avoid fragmenting equivalent states. A transition records the source and target states, action type, interaction target, before/after locations, and a simplified accessibility delta.

Depth-aware semantic novelty. For a candidate control e, WebSAPE defines

$$k(e) = \begin{cases} (a(e), \mathrm{norm}(\ell(e))), & \ell(e) \neq \emptyset, \\ (a(e), s(e)), & \text{otherwise}, \end{cases} \tag{6}$$

where norm lowercases the label, normalizes whitespace, and removes common interface terms. At depth $d > 0$, a control is eligible only if its target has not already been explored and its semantic key did not occur at a shallower depth:

$$\mathrm{eligible}(e, d) = \mathbb{I}[s(e) \notin S_v]\,\mathbb{I}[s(e) \notin S_{\mathrm{global}}]\,\mathbb{I}\left[k(e) \notin \bigcup_{j<d} K_j\right]. \tag{7}$$

This is a heuristic normalization procedure rather than embedding-based clustering.

Candidate filtering and exploration budget. Given candidate controls C at depth d, the crawler (i) removes previously explored targets and shallower-depth semantic keys; (ii) filters obvious transient or content-specific controls; (iii) applies LLM-based functional selection; (iv) ranks the survivors for the current state; and (v) expands at most b actions.

Table 4: Default topology-exploration parameters.

Parameter	Default
Maximum exploration depth D	3
Branch budget b	15
Maximum candidate controls per state	50

Topology-guide synthesis. States are grouped by exploration depth and summarized by page location, functional label, available actions, interaction targets, destination states, and accessibility changes. Failed or unvalidated transitions are omitted, and repeated row-like operations may be represented by a common action pattern. A language model then synthesizes the summary into a navigation guide whose entries describe the operation available in a state and its expected result. For example, a graph transition from a product listing through the action "Men's Shoes" to a filtered category state is rendered as a guide entry indicating that selecting "Men's Shoes" opens the corresponding category view.

B Step-Count Analysis

In addition to the success rate reported in Table 2, we report two per-query step counts, averaged over recorded trajectories on the proposed BizArena benchmark. A *tool-call step* (S_{tool}) counts one `tool_call` entry. An *all-entry step* (S_{all}) counts every progress entry, including reasoning, tool calls, tool results, errors, status messages, and planning events. Table 5 shows that WebSAPE has the lowest overall averages among the compared methods under both definitions in most scenarios and also achieves the least average steps, i.e., 24.46 tool-call steps and 107.09 all-entry steps per query. These thanks to the topology guides that identify relevant pages and interaction targets, and the distilled experience that provides effective operation experience to avoid redundant exploration and repeated trial-and-error.

Table 5: Average tool-call steps (S_{tool}) and all-entry steps (S_{all}) per query on BizArena with Claude Opus 4.5. Fewer steps are better for both definitions.

Method	Chat		Email & Calendar		Project Mgmt		Cross-App		Overall	
	$S_{\text{tool}} \downarrow$	$S_{\text{all}} \downarrow$	$S_{\text{tool}} \downarrow$	$S_{\text{all}} \downarrow$	$S_{\text{tool}} \downarrow$	$S_{\text{all}} \downarrow$	$S_{\text{tool}} \downarrow$	$S_{\text{all}} \downarrow$	$S_{\text{tool}} \downarrow$	$S_{\text{all}} \downarrow$
AgentOccam	23.44	119.19	19.91	101.39	33.60	169.84	41.77	210.73	29.58	149.78
WALT	20.56	101.81	58.97	267.39	31.44	154.88	51.65	254.19	44.04	209.34
Browser-Use	**17.81**	92.31	23.12	124.97	31.52	152.64	**28.65**	154.31	25.81	134.29
Claude Code	37.88	93.56	67.94	162.31	56.28	137.96	70.23	170.46	60.74	147.19
WebSAPE (ours)	18.69	**80.44**	**19.00**	**84.45**	**25.88**	**108.38**	33.56	**151.00**	**24.46**	**107.09**

C LLM Cost Analysis

We additionally analyze WebSAPE's LLM inference cost per query. Let I_q, R_q, W_q, and O_q denote uncached input, cache-read input, cache-creation input, and output tokens recorded for query q, respectively. Using Anthropic's published rates of \$5, \$0.50, \$6.25, and \$25 per million tokens, with the 5-minute cache-write rate (Anthropic, n.d.), the cost in USD and its mean over a query set $\mathcal{D}$ are

$$C_q = \frac{5I_q + 0.5R_q + 6.25W_q + 25O_q}{10^6}, \qquad \overline{C}_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} C_q. \tag{8}$$

The input buckets are disjoint: cached tokens are not included in I_q, and output tokens are charged once.

Table 6: Average LLM inference cost (USD) per query for WebSAPE with Claude Opus 4.5. Claude Code is included as a representative cutting-edge industrial-agent reference. Lower is better.

Benchmark	WebSAPE	Claude Code (ref.)
WebArena-Lite	**1.5535**	1.5840
BizArena (overall)	**2.1825**	2.7321
Chat	1.5030	**1.4810**
Email & Calendar	**2.2890**	2.9085
Project Mgmt	**1.8605**	2.5815
Cross-App	**2.7750**	3.4230

As shown in Table 6, WebSAPE costs \$1.5535 per query on WebArena-Lite and \$2.1825 per query on BizArena overall. Across BizArena categories, its average cost ranges from \$1.5030 on Chat to \$2.7750 on Cross-App, reflecting differences in task complexity and interaction length. These costs are lower than the representative cutting-edge industrial agent, Claude Code (Anthropic, 2025), at the benchmark level: Claude Code costs \$1.5840 on WebArena-Lite and \$2.7321 on BizArena overall. At the category level, WebSAPE is also lower-cost on Email & Calendar, Project Management, and Cross-App; Chat is the only exception, where WebSAPE costs 1.5% more (\$1.5030 versus \$1.4810). Thus, WebSAPE's performance gains do not require higher overall LLM inference cost than this industrial-agent reference.