Scalable Behaviour Cloning on Browser Using via Skill Distillation

Internet users collectively perform an enormous range of skilled work through web browsers, from software development and document editing to search, forms, and enterprise workflows. This activity is a highly scalable but under-exploited source of reusable browser skills. We study scalable behavior cloning for browser agents via skill distillation, converting user interaction trajectories into compact natural-language skills that agents can read, retrieve, reuse, and compose directly. We show that simple trajectory summarization alone can improve benchmark performance through behavior cloning, and further introduce a distributed framework for scalable skill collection and reuse across users and tasks. This suggests that the scalability of browser agents may come less from manually designed tasks and more from the collective skills already expressed by internet users. Our project is available at: https://lab.einsia.ai/browserbc.

Date: June 27, 2026

Einsia

1 Introduction

Web browsers have become the universal execution interface for modern digital work. Developers move between code hosting, continuous integration, and documentation; operators maintain dashboards and enterprise workflows; researchers search and transfer information across online tools; and everyday users complete tasks in email, calendars, shopping sites, and software-as-a-service applications. A browser agent that can reliably operate this interface would therefore provide a general execution layer for digital labor. Early work such as WebShop showed how language agents could act in grounded web environments (Yao et al., 2023), while Mind2Web broadened the problem toward generalist agents over real websites (Deng et al., 2023). More recent benchmarks, including WebArena and VisualWebArena, make the setting increasingly realistic by requiring long-horizon interaction, multimodal observation, element grounding, and task-level evaluation (Zhou et al., 2024; Koh et al., 2024), and WorkArena and BrowserGym extend this line toward common knowledge-work tasks and standardized research infrastructure (Drouin et al., 2024; Chezelles et al., 2025).

Despite this progress, contemporary web agents remain strikingly inefficient on real websites, and we argue that the bottleneck is *decision-making* rather than *operation*. Today's agents are no longer poor at operating a browser: they parse rendered and structured observations, locate buttons and input fields, and execute clicks, keystrokes, navigation, and submissions. The difficulty is not *whether* to click but *where* to click. A website is effectively a large maze in which the agent observes only its local surroundings, without knowing the global site structure, the relative cost of alternative paths, which entry points are reliable, or which state signals task completion. Acting under this partial observability, even a capable agent is forced to explore by trial and error—entering plausible but irrelevant pages, oscillating among search results, stopping just short of the answer, or omitting a critical field in a long form. These are failures of decision under incomplete information, not of low-level control. This view is consistent with evidence that the hard part of web interaction lies above flat action imitation: WebAgent couples planning, long-context understanding, and program synthesis (Gur et al., 2024), SeeAct identifies grounding as a central bottleneck (Zheng et al., 2024), and OSWorld reports the same pattern for open-ended computer-use tasks beyond the browser (Xie et al., 2024).

The prior that agents lack is, however, abundantly available in human behavior. Every day, people complete real tasks in browsers, and they rarely re-derive each step from scratch: they act with priors about where entry points are, how to read and narrow search results, which fields of a form matter, and what feedback confirms completion. These priors are seldom written on the page but are implicit in the path a person

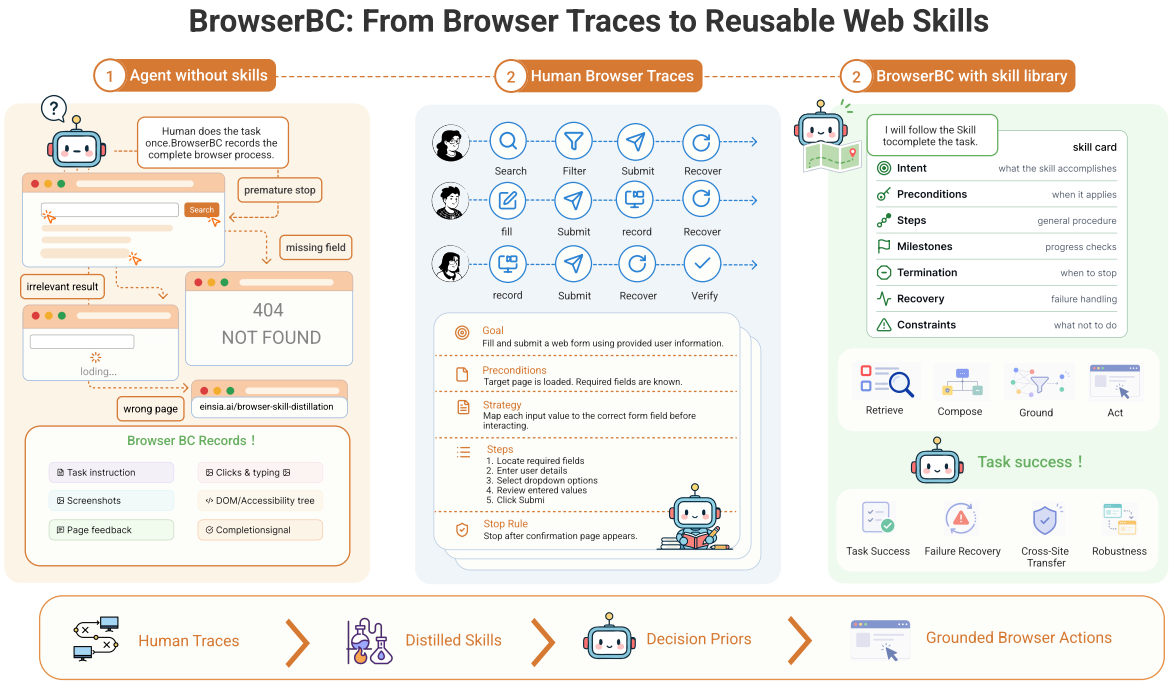


Figure 1 Motivation of BrowserBC. BrowserBC turns human browsing traces into reusable natural-language skills, providing agents with decision priors for navigating unfamiliar websites.

takes, so interaction traces are not merely action logs but large-scale records of human decision-making under partial observability. Such traces are increasingly available at scale: Mind2Web supervises grounding and next-step prediction from demonstrations over real websites (Deng et al., 2023), WebLINX adds multi-turn, dialogue-conditioned navigation traces (Lù et al., 2024), and AgentTrek synthesizes trajectories from web tutorials (Xu et al., 2025).

Raw traces, however, are a poor unit of reusable supervision. They are long, noisy, redundant, and tightly coupled to incidental page state: low-level targets depend on coordinates, DOM structure, transient identifiers, login state, and page versions, so directly cloning clicks or next-step actions learns brittle page-specific patterns rather than transferable competence. We argue that the reusable unit of browser behavior is neither an individual action nor an entire trajectory, but a transferable natural-language *skill*: a structured description of a subtask strategy that states its intent, applicability, preconditions, execution steps, progress milestones, termination evidence, and failure-recovery procedures. This view builds on evidence that language agents benefit from persistent verbal experience, ranging from storing textual feedback (Shinn et al., 2023) and extracting reusable lessons from prior trials (Zhao et al., 2024) to maintaining an expanding skill library for open-ended exploration (Wang et al., 2023) and inducing skills above the level of raw action sequences (Wang et al., 2025b). Representing experience as natural language yields a property we consider central: the model that *produces* a skill and the model that *executes* it can be decoupled. A capable model or a person performs a task once, the run is distilled into a skill, and a different, smaller, and cheaper model can then read that skill and carry out the same family of tasks. Experience recorded once can thus be broadcast to all agents, charting a route to general web browsing that does not require training every agent to be stronger.

Building on these observations, we propose **BrowserBC**, a scalable framework that converts browser interaction traces into a retrievable, composable, and executable library of natural-language skills. BrowserBC cleans and semantically segments each trace, distills the resulting evidence into structured skill cards that retain transferable procedural knowledge while stripping volatile or leaking detail, and organizes candidate skills into a skill graph so that growth proceeds through consolidation rather than unbounded accumulation. At inference time, a lightweight retriever supplies a small set of relevant skills as context, and the agent grounds them into element selection and executable actions on the live page. The resulting formulation

is complementary to modular agent-training frameworks such as Agent Lumos (Yin et al., 2024) and to standardized web-agent environments such as BrowserGym (Chezelles et al., 2025).

Contributions. This paper makes the following contributions:

- We reframe the inefficiency of capable web agents as a decision problem under incomplete information, and argue that human interaction traces constitute large-scale human decision data whose reusable unit is a transferable natural-language skill rather than a click or a full trajectory.
- We propose BrowserBC, a scalable framework that distills browser traces into a skill library organized as a skill graph and decouples skill provenance from skill execution, so that experience recorded once is reusable across models.
- We validate the framework on challenging web-agent benchmarks, measuring next-action prediction, end-to-end task success, failure recovery, skill reuse, cross-site generalization, and robustness to page perturbations.

2 Related Work

Web and computer-use agents. Research on browser agents has advanced along three coupled fronts: environments, benchmarks, and end-to-end policies. Early work established grounded language-agent interaction in controlled settings, from WebShop in a simulated storefront (Yao et al., 2023) to Mind2Web over real websites (Deng et al., 2023). WebArena and VisualWebArena subsequently made evaluation realistic by requiring long-horizon interaction, multimodal perception, element grounding, and programmatic success checks (Zhou et al., 2024; Koh et al., 2024), and WorkArena and BrowserGym extended this line toward enterprise knowledge work and a shared research infrastructure (Drouin et al., 2024; Boisvert et al., 2025; Chezelles et al., 2025). A parallel effort strengthens the policy itself: WebVoyager and OpenWebVoyager study multimodal agents under realistic browsing (He et al., 2024a,b), SeeAct isolates grounding as the central bottleneck for vision-language agents (Zheng et al., 2024), and WebAgent couples planning, long-context reading, and program synthesis (Gur et al., 2024). Beyond the browser, OSWorld generalizes the problem to open-ended computer use and argues for robust observation–action abstractions over narrow memorization of UI state (Xie et al., 2024), while large-scale efforts such as ScaleCUA and OpenCUA characterize how competence scales with data and supervision (Liu et al., 2026; Wang et al., 2025a). More recent systems push capability through reinforcement learning and information-seeking objectives (Li et al., 2026; Wu et al., 2025; Vattikonda et al., 2025), process reward modeling (Chae et al., 2025), structured or hierarchical exploration (Yang et al., 2025; Gandhi and Neubig, 2026; Shen et al., 2025), and tool or context abstractions (Prabhu et al., 2026; Ye et al., 2026). The recurring theme is that browser competence improves with scale, abstraction, and richer interaction. Our work is orthogonal to all three fronts: rather than proposing new environments, rewards, or training objectives, we treat the trajectories already produced by ordinary internet users as a scalable supervision signal and ask how to convert them into reusable procedural knowledge.

Learning from web demonstrations. A substantial body of work uses human or agent trajectories as supervision. Mind2Web shows that demonstrations over real websites can supervise action grounding and next-step prediction (Deng et al., 2023); WebLINX collects multi-turn, dialogue-conditioned navigation traces from real-world use (Lù et al., 2024); and AgentTrek synthesizes trajectories from web tutorials as an alternative route to scalable data (Xu et al., 2025). Such traces implicitly encode how people decompose work, locate pages, select elements, recover from mistakes, and recognize task completion. The central difficulty is one of representation: raw traces are long, noisy, and tightly coupled to incidental page state, so directly cloning coordinates, selectors, or next-step actions tends to capture brittle, page-specific correlations rather than transferable competence. We share the goal of exploiting demonstrations at scale, but we take the reusable unit of supervision to be a transferable natural-language *skill* rather than a single action or an entire trajectory, and we deliberately abstract away coordinates and selectors during distillation so that the resulting guidance survives changes in page version and authentication state.

Skills and reusable experience. Our formulation builds on evidence that language agents benefit from persistent verbal experience. Reflexion converts outcome feedback into textual self-notes that improve later attempts (Shinn et al., 2023), ExpeL extracts reusable lessons across trials (Zhao et al., 2024), and Voyager maintains an expanding library of executable skills for open-ended embodied exploration (Wang et al., 2023). Related work induces programmatic skills that represent behavior above the level of raw action sequences (Wang et al., 2025b), while modular frameworks such as Agent Lumos decouple planning, grounding, and execution into reusable components (Yin et al., 2024). A complementary observation is that recorded experience can be reused beyond the agent that produced it: memory-sharing schemes pool useful experiences across language agents (Gao and Zhang, 2024), and robotics efforts such as Open X-Embodiment show that heterogeneous demonstrations collected across platforms can support transfer (Open X-Embodiment Collaboration et al., 2024). We differ from these lines in both source and representation: we distill *human* browser trajectories spanning many users, websites, and tasks into an auditable skill library, and—because each skill is a self-contained natural-language description—we decouple the model that *produces* a skill from the model that *executes* it, so that experience recorded once can be reused across different and cheaper models, with many trajectories contributing to one skill and one trajectory yielding several skills.

3 Method

BrowserBC is a *skill-centric* behavior cloning framework for browser agents. Instead of imitating demonstrations as flat sequences of low-level actions, BrowserBC distills user interaction trajectories into reusable, natural-language *skills*. Each skill encodes a transferable procedure for a class of browser tasks: when it applies, which information must be preserved across steps, how progress is made, how completion is verified, and how common failures are diagnosed and recovered from. At inference time, the agent retrieves the skills semantically relevant to the current task, conditions its action generation on them, and grounds every resulting action in the live browser state.

This design shifts the unit of behavior cloning from an atomic action to a reusable skill: a trajectory is treated not as a script to be replayed verbatim but as *evidence* of the procedural knowledge underlying a successful interaction. The distinction is particularly consequential on the web, where the same semantic operation may be realized through markedly different layouts, DOM structures, widgets, and interaction flows—an action that is optimal on one page may be meaningless or even harmful on another. By cloning skills rather than surface actions, BrowserBC transfers browser competence across websites and tasks while substantially reducing its dependence on brittle, page-specific details. Moreover, because each skill is a self-contained natural-language object, the model that *distills* a skill need not be the model that *executes* it: competence recorded once can be reused across different and cheaper agents.

Figure 2 summarizes the four stages of BrowserBC: behavioral evidence abstraction (§3.2), procedural skill distillation (§3.3), skill-library construction (§3.4), and skill-conditioned execution (§3.5). The first three stages address three obstacles that separate raw demonstrations from reusable competence—noisy and non-transferable trajectory logs, the brittleness of action replay, and the uncontrolled growth of an ever-expanding demonstration store—while the last stage puts the resulting skills to work. We first formalize the skill-centric view that ties these stages together.

3.1 Problem Formulation

We denote a browser trajectory as $\tau = (x, e_1, \ldots, e_T, y)$, where x is a natural-language task instruction, $e_t = (o_t, a_t, f_t)$ is the t-th event comprising the observation o_t, the user action a_t, and the execution feedback f_t (e.g., page transitions, validation messages, or completion signals), and y is the task outcome when available.

A standard behavior cloning objective fits a low-level policy $\pi(a_t \mid x, o_{\leq t})$ that maps the task and interaction history directly to the next action. In browser environments, however, low-level actions are weak carriers of transferable behavior: the same intent may map to different selectors, coordinates, or input mechanisms across pages, while visually similar actions may carry different procedural meaning depending on the task state. Cloning raw actions therefore captures page-specific correlations rather than the reusable skills that explain *why* a demonstration succeeds.

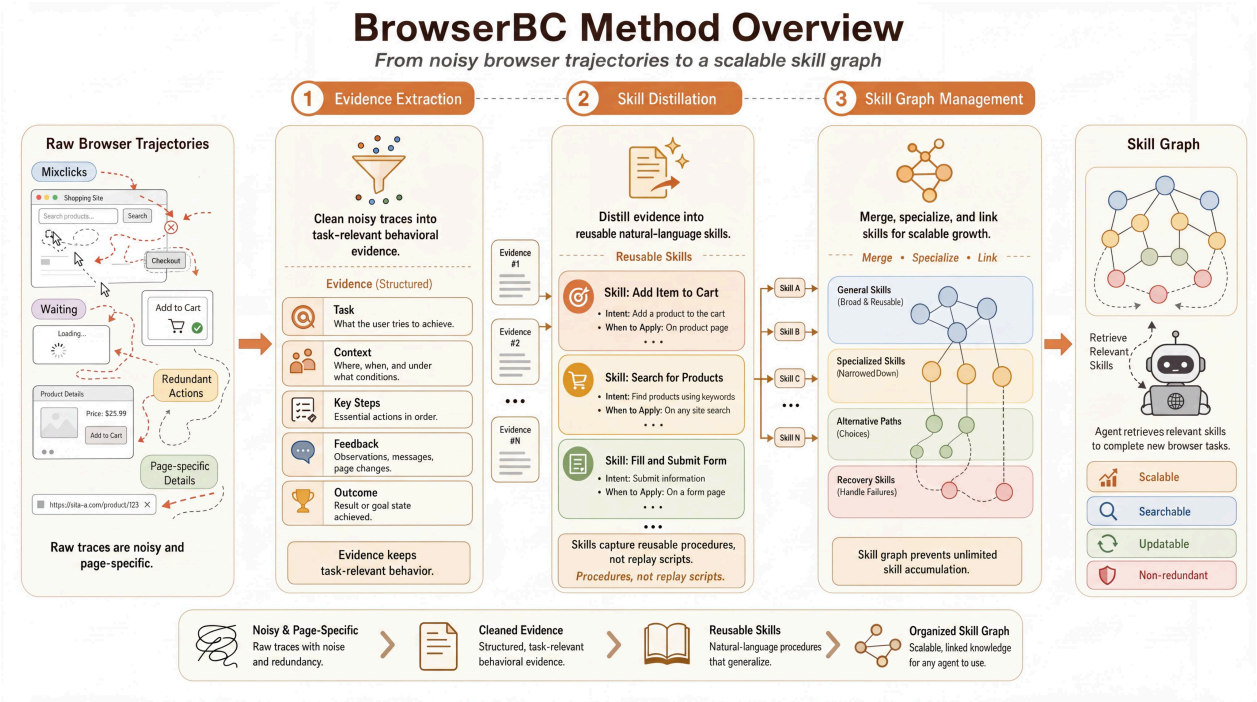


Figure 2 Overview of BrowserBC. BrowserBC cleans and segments browser traces, distills them into structured natural-language skills, organizes the skills into a graph, and retrieves relevant skills to guide browser-agent execution on new tasks.

We instead cast behavior cloning over a *skill library* $\mathcal{L} = \{s_i\}_{i=1}^{N}$, where each s_i is a natural-language procedure distilled from one or more trajectories. Given a new instruction $x^\star$ and browser history $h_t^\star = o_{\le t}^\star$, the agent retrieves a compact, task-relevant subset of skills and predicts the next action through a *skill-conditioned* policy:

$$\mathcal{R}_t = \mathcal{R}(x^\star, h_t^\star, \mathcal{L}), \quad |\mathcal{R}_t| \ll |\mathcal{L}|, \qquad a_t \sim \pi_\theta(\cdot \mid x^\star, h_t^\star, \mathcal{R}_t). \tag{1}$$

The retrieved skills $\mathcal{R}_t$ do not replace perception or grounding; they supply *procedural constraints* on what the agent should attend to, which values it must preserve, what intermediate progress looks like, when a task should terminate, and how to recover when a step fails. Crucially, the library $\mathcal{L}$ is decoupled from the policy π_θ that consumes it: skills are distilled once and serve as model-agnostic guidance, so the executing agent can differ from—and be substantially cheaper than—the one used during distillation. In this view, demonstrations supervise the agent through reusable strategies and explicit completion criteria rather than through one-step imitation of individual actions.

3.2 Behavioral Evidence Abstraction

Raw trajectories must be cleaned and abstracted before any skill can be distilled from them. They are noisy and heterogeneous, interleaving misclicks, idle waits, repeated attempts, and transient page states with the few interactions that actually drive task progress; they are also saturated with volatile or privacy-sensitive content—exact coordinates, DOM selectors, transient identifiers, login state, and personal text—that is irrelevant or even harmful to retain. Distilling skills from such logs without first isolating the task-relevant signal would propagate noise and non-transferable artifacts into every downstream skill.

The first stage therefore converts raw trajectories into structured *evidence*. We normalize each trajectory into a common event representation, filter out interactions that do not contribute to task progress, and segment the remainder into semantically coherent sub-procedures—rather than arbitrary fixed-length windows—so that each segment corresponds to a meaningful unit of behavior. For each segment we extract an *evidence unit*

$$u = \big(x,\ c^{\mathrm{pre}},\ b,\ c^{\mathrm{post}},\ f,\ y\big), \tag{2}$$

comprising the instruction x, the contexts c^{pre} and c^{post} that summarize the state immediately before and after the segment, a summary b of the demonstrated behavior, the feedback f observed during the segment, and the outcome y. The context summaries preserve task-relevant information—visible affordances, required input fields, user-provided values, applicable constraints, and success or failure signals—while discarding non-transferable artifacts. This abstraction *lifts* demonstrations from page-specific action traces into procedural evidence that generalizes across browser states, forming the clean substrate from which skills are distilled.

3.3 Procedural Skill Distillation

Even on clean evidence, cloning behavior as a replayable trace is fundamentally brittle on the web. Layouts, DOM structures, page versions, and authentication states change constantly, so a procedure pinned to specific coordinates or selectors breaks as soon as the page shifts; conversely, the genuinely transferable content of a demonstration—*what* to accomplish and *how* to tell that it is progressing and complete—is never made explicit by a raw action sequence. The challenge is to extract this reusable procedural knowledge while deliberately stripping away the volatile and leak-prone details that do not transfer.

Given a set of evidence units $\mathcal{U}_s$ and optional metadata m_s (e.g., task category, website family, or source identifiers), BrowserBC distills an agent-readable skill $\hat{s} = D(\mathcal{U}_s, m_s)$. A skill is a structured natural-language *card* with a fixed set of fields: intent, applicable scope, preconditions, execution steps, progress milestones, terminal evidence, failure modes, recovery strategies, negative constraints, and provenance. These fields make a skill directly consumable by a language-model agent while keeping it human-auditable and easy to update. The distillation model is explicitly instructed to retain reusable procedural structure while removing session-specific artifacts—private text, exact coordinates, DOM paths, or one-time authentication state—and a leakage audit ensures that no task-specific answer or evaluation-checker content is silently written into a card unless such details are genuinely essential to the task semantics. Crucially, a distilled skill specifies *what* should be accomplished and *how* to reason about progress toward it, not how to replay a particular demonstration: a form-filling skill, for instance, instructs the agent to identify fields by their semantic labels and to preserve task-provided values, rather than to reuse a specific coordinate or selector that happened to work in the source trajectory. Skills may be induced from a single successful trajectory—which already captures the structure of one viable solution—or consolidated from multiple attempts, where successful runs strengthen the execution guidance while failed runs expose missing preconditions and motivate explicit recovery strategies.

3.4 Skill-Library Construction

Distilling one skill per trajectory does not by itself yield a scalable system. If every demonstration produces an independent entry, the library quickly degenerates into a redundant, ever-growing pile of overlapping and sometimes mutually conflicting skills, in which retrieval becomes diffuse and any update risks touching unrelated entries. Scalability requires that the library be *compressed and structured* as it grows, not merely accumulated.

To this end, whenever a candidate skill $\hat{s}$ is produced, BrowserBC decides whether to *add* it as a new entry, *merge* it into an existing skill, or register it as a *specialization* of a more general one. Two skills are merged when they share a compatible intent, preconditions, steps, effects, and terminal evidence, and are kept separate when they apply under different conditions, require different information, or encode mutually conflicting constraints. We organize the resulting library as a lightweight *skill graph* $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, whose nodes are distilled skills and whose edges encode simple relations among them—temporal dependency, specialization, alternative procedures for a shared subgoal, or incompatibility under the same state. This lets a generic procedure (e.g., form filling) link to its specializations (such as payment or profile updates) and to the recovery skills that handle its characteristic failures, without collapsing them into a single flat entry. The organization makes BrowserBC scalable along three axes: it consolidates repeated demonstrations into reusable nodes instead of accumulating isolated examples; it restricts retrieval and updates to the relevant regions of the skill space; and it supports incremental refinement, since a new trajectory updates only the affected skills and their immediate neighbors.

3.5 Skill-Conditioned Execution

At inference time, the retrieval operator $\mathcal{R}$ in Eq. (1) selects the most relevant skills by their semantic similarity to the task and, when available, their compatibility with the current browser context. We deliberately avoid introducing a specialized retrieval architecture or an explicit symbolic planner: retrieval simply exposes the agent to the prior knowledge most likely to be useful at the current step. The selected skills $\mathcal{R}_t$ are then inserted into the agent's context as natural-language guidance, and the agent grounds them into executable actions against the live page.

Skills are thus neither executable scripts nor demonstrations to be replayed verbatim: they bias the agent toward distilled behavior patterns, while each concrete action is still chosen with respect to the current state. When applying a form-filling skill, for example, the agent identifies the required fields by their labels, transfers the task-provided values into them, monitors for validation messages, and stops once a success confirmation appears—reading the concrete fields and layout from the current page rather than copying them from any source trajectory. Because this guidance is purely textual and model-agnostic, the executing agent may be a different and lighter-weight model than the one used for distillation. This combination preserves the data efficiency of behavior cloning while avoiding the brittleness of action replay, allowing a single skill to guide behavior across different websites, interaction mechanisms, and underlying models.

4 Experiments

4.1 Experimental Setup

We evaluate BrowserBC on three benchmarks that probe complementary axes of browser and computer-use competence, progressing from controlled reproducibility to real-world volatility and finally to cross-substrate generalization (Table 1). WebArena-Hard (Zhou et al., 2024) comprises 258 human-verified tasks across six self-hosted sites with original programmatic checkers; its stable and reproducible environment makes it our primary setting for controlled comparison and mechanistic analysis. ClawBench complements this with tasks on *live production websites*, combining layout volatility, write-heavy state changes, and human-grounded agentic verification; because the same page may change between runs, it constitutes our strongest test of whether BrowserBC skills encode transferable procedural knowledge rather than brittle, page-specific action traces. Finally, OSWorld-style desktop tasks move interaction off the browser onto applications, files, dialogs, and persistent system state, serving as a diagnostic study of whether the skill abstraction transfers across interaction substrates.

Table 1 Summary of the three evaluation benchmarks. The three settings span controllability, real-world fidelity, and cross-substrate generalization, and jointly test whether BrowserBC clones reusable procedural knowledge rather than surface actions.

Benchmark	Environment	# Tasks	Verification	Primary stress axis
WebArena-Hard	Self-hosted web (6 sites)	258	Programmatic checkers	Controllability / reproducibility
ClawBench	Live production websites	152	Agentic pass/fail evaluator	Real-world volatility / write-heavy state
OSWorld (case study)	Ubuntu desktop apps	30	State-based verifiers	Cross-substrate generalization

Unless otherwise stated, all comparisons hold the agent, action interface, and step budget fixed, varying only whether retrieved BrowserBC skills are inserted into the agent's context. To control leakage, skills are restricted to procedural content, and any checker-specific or answer-bearing values are audited and redacted before evaluation; identical checker normalizations are applied to the skill-off and skill-on runs.

4.2 Main Results

Table 2 reports task-group results on WebArena-Hard and per-category results on ClawBench for the identical browser agent with skills off (base) and with the BrowserBC skill library.

WebArena-Hard. BrowserBC raises overall task success from 60.5% (156/258) to 81.4% (210/258), a +20.9-point absolute gain aggregated across task groups. The improvement is largest on Multi-site and

Table 2 Performance comparison across WebArena-Hard and ClawBench. For consistency, all Base and BrowserBC results are reported as success rates (SR, %).

Task group / category	# Cases	Base(Skill-Off) SR	BrowserBC SR	Δ SR
WebArena-Hard				
GitLab	57	64.9	86.0	+21.1
Shopping	56	60.7	89.3	+28.6
Shopping (admin)	55	56.4	70.9	+14.5
Reddit	42	78.6	85.7	+7.1
Multi-site	48	43.8	75.0	+31.2
All	**258**	**60.5**	**81.4**	**+20.9**
ClawBench				
Daily	52	24.6	64.9	+40.3
Finance	6	50.0	100.0	+50.0
Work	21	47.1	76.5	+29.4
Dev	18	33.3	66.7	+33.4
Academic	14	50.0	78.6	+28.6
Travel	13	38.5	76.9	+38.4
Social	18	25.0	56.2	+31.2
Pets	11	27.3	54.5	+27.2
All	**153**	**32.9**	**68.4**	**+35.5**

Δ SR denotes the absolute improvement in success rate, measured in percentage points.

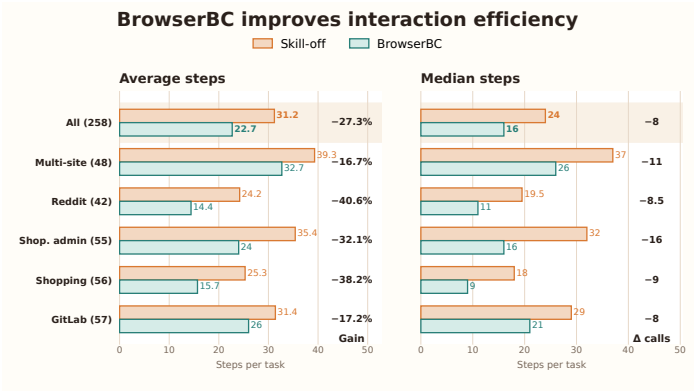
Shopping tasks, where procedural navigation, multi-step search, and result interpretation are most demanding, whereas the smaller gains on Shopping-admin and Reddit indicate that skills help most when the bottleneck is procedural uncertainty rather than low-level execution fidelity.

ClawBench. On ClawBench, the skill-free baseline solves 50/152 tasks (32.9%), closely matching the difficulty profile reported for frontier agents, while BrowserBC solves 104/152 tasks (68.4%)—a +35.5-point absolute improvement that roughly doubles the number of solved tasks and is broadly distributed across all eight categories. This gain is notable precisely because ClawBench removes the conditions under which a skill library could succeed through memorization: as tasks run on live websites whose layout, availability, and flows change between runs, any skill encoding exact coordinates, DOM selectors, or cached page state would degrade rather than help. The large improvement therefore indicates that BrowserBC skills transfer at the level of *what to accomplish and how to verify completion*. The result is further consistent with ClawBench's write-heavy emphasis: skills that supply explicit preconditions, the state-changing operation, and the terminal evidence confirming submission directly address the completion-recognition failures that the agentic evaluator penalizes. Finally, the magnitude of this gain on a live, unsaturated benchmark—larger in relative terms than the WebArena gain—suggests that procedural priors are most valuable where the base agent's bottleneck is procedural uncertainty on unfamiliar real-world sites rather than low-level grounding.

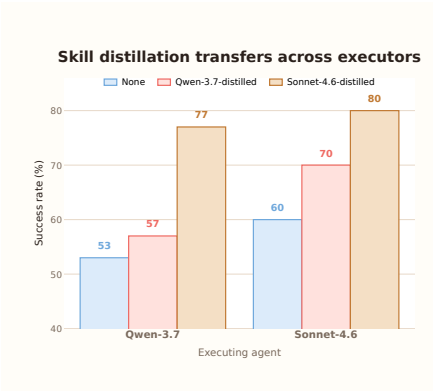
4.3 Analysis and Ablations

Skills reduce interaction length. Beyond success rate, BrowserBC shortens the interaction required to solve a task. Across all WebArena-Hard tasks, the mean number of agent tool calls drops from 31.2 to 22.7 (−27.3%) and the median from 24 to 16 (−8 steps), broken down by task group in Figure 3a. This reduction is consistent with the role of skills as procedural priors: they curtail exploratory navigation and repeated page inspection while leaving low-level grounding to the live browser state.

Decoupling the distiller from the executor. A central design claim of BrowserBC is that a skill is a model-agnostic object: it can be distilled once and reused by a different, cheaper agent at execution time (§3.1). We test this by crossing the model that *distills* skills with the model that *executes* them on 30 read-only WebArena-Hard tasks, using identical trajectories and the same evaluation protocol. We visualize the six



(a) Interaction efficiency on WebArena-Hard by task group.



(b) Decoupling the distiller from the executor.

Figure 3 BrowserBC improves interaction efficiency and transfers distilled skills across executors. Left: BrowserBC reduces average steps in every group and lowers the overall median by 8 tool calls. Right: on WebArena-Hard tasks, bars are grouped by the *executing* agent and colored by the *distilling* model.

resulting configurations as a grouped bar chart (Figure 3b), grouping by executor and coloring by skill source, which renders the two effects directly comparable: *within* each executor group the bars rise sharply only for Sonnet-distilled skills, while *across* groups the Sonnet-distilled bars are of similar height. Two findings emerge. First, skill quality is determined primarily at distillation time: Sonnet-4.6-distilled skills improve both executors substantially (+24 and +20 points), whereas Qwen-3.7-distilled skills yield only marginal gains. Second, high-quality skills transfer across executors: the small agent equipped with Sonnet-4.6-distilled skills reaches 77%, approaching the large agent's 80%, directly substantiating the "distill once, reuse cheaply" premise.

Skills are calibrated priors, not mandatory scripts. Finally, we examine whether retrieved skills should be followed unconditionally. An always-follow variant, which executes the retrieved skill without allowing the agent to override it against the live state, reaches only 77.5% on WebArena-Hard, below BrowserBC's 81.4%. A skill is thus best treated as a calibrated prior that the agent may revise when live observations contradict it, rather than a rigid program—a property we probe more sharply through the mismatched-skill controls in the desktop setting (§4.4).

4.4 Beyond the Browser: OSWorld Case Study

Goal and framing. The preceding results establish BrowserBC within browser benchmarks. We now use OSWorld-style desktop tasks to ask a different question: whether the same skill abstraction remains meaningful when interaction moves from web pages to a general computer-use environment. We treat this not as a full OSWorld benchmark but as a diagnostic transfer study that identifies which part of BrowserBC transfers, which part does not, and what this reveals about natural-language skills in GUI agents. The transferable object is not a browser-specific action sequence but a *procedural prior*—a compact description of preconditions, semantic state transitions, progress milestones, terminal evidence, and recovery strategies—which in desktop tasks ranges over applications, files, dialogs, window focus, and persistent settings, separating procedural knowledge from low-level GUI execution more sharply than browser-only environments.

Setup. We construct 30 OSWorld-style Ubuntu desktop cases spanning system settings, file manipulation, text and structured-data editing, application configuration, terminal interaction, browser/media dispatch, modal dialogs, and office applications, each with a fixed initial state and an independent state-based verifier. Skills are distilled at the task-family level from successful GUI demonstrations—recording what state must be established, which application-level operation produces it, what intermediate evidence indicates progress, and what final evidence should be checked—rather than as coordinate traces. Beyond matched skill-on and no-skill runs, we add two controls: easy cases where the base agent already succeeds, so that not all success is

Table 3 Mechanistic interpretation of the OSWorld-style case study. Counts summarize selected desktop cases and controls; they diagnose *when* skills help rather than estimate a full OSWorld success rate.

Regime	Count	Mechanistic interpretation
High procedural uncertainty	17	Base agent reaches a relevant surface but lacks a reliable task schema; the skill supplies the missing procedure and terminal evidence.
Low procedural uncertainty	2	Task is shallow enough that the base agent already finds the action path; the marginal value of a skill is small.
High execution fragility	10	Procedure is not the bottleneck; failure stems from focus, modal grounding, file-picker state, or recovery.
Poor skill calibration	3	A plausible but wrong prior confidently moves the agent away from the target state; retrieval and binding matter.

attributed to skills, and mismatched-skill controls supplying an irrelevant or wrongly parameterized skill, to test whether skills behave as useful priors or as brittle instructions.

Results overview. Table 3 summarizes the case-level outcomes, which should be read as diagnostic evidence rather than a leaderboard score. The key observation is not merely that 17 cases improve with matched skills, but that outcomes organize around three underlying factors—procedural uncertainty, execution fragility, and skill calibration—revealing that BrowserBC-style skills primarily reduce epistemic uncertainty about the workflow while leaving low-level control uncertainty largely unaddressed.

Finding 1: skills act as task schemas. The clearest positive cases are those in which the base agent partially understands the goal but fails to preserve the full schema—opening a settings page without toggling the correct state, or opening a terminal without executing the command. The distilled skill repairs this by externalizing a human-like routine of establishing context, performing the state-changing operation, persisting it, and verifying terminal evidence—the same mechanism observed on WebArena, now ranging over applications and files.

Finding 2: a boundary between knowledge and control. In the still-hard cases the semantic routine is often correct, yet failure occurs at the level of GUI control: the wrong window receives input, a modal is not grounded, or a path resolves unexpectedly. This indicates that skill distillation clones procedural knowledge but does not by itself resolve perception, focus management, or low-level action reliability.

Finding 3: skills can mislead when miscalibrated. A timezone skill that selects London fails for UTC+0 under daylight saving, whereas the matched skill correctly selects Reykjavik; an unrelated skill, in turn, performs confident but irrelevant actions. This turns the limitation into a concrete design requirement for evidence-aware retrieval and parameter binding.

Scope. Overall, the portable component of BrowserBC is the representation of a skill as a semantic state-transition model, which transfers whenever the task family exhibits repeated procedural structure, intermediate progress can be recognized semantically, and the final state can be independently verified. Its value is highest when the missing ingredient is procedural structure, and limited when the missing ingredient is low-level GUI grounding or when retrieval supplies the wrong prior.

5 Conclusion

In this work, we presented a framework that converts raw agent trajectories into a reusable skill library through three stages: Behavioral Evidence Abstraction, Procedural Skill Distillation, and Skill-Library Construction. By decoupling skill distillation from skill execution, our approach separates the question of *what* constitutes a good skill from *which* agent ultimately uses it, allowing skill quality to be fixed once at distillation time and transferred across heterogeneous executors. Experiments show that distilled skills consistently improve downstream task success and that stronger distillers yield skills that benefit even weaker executing agents.

These results indicate that explicitly distilling procedural skills from past trajectories is an effective way to accumulate and reuse experience across tasks and agents. We see several promising directions for future work, including extending the library to longer-horizon and multi-agent settings, and studying how skills can be continually refined and pruned as the agent encounters new environments.

6 Contributions and Acknowledgements

Zheng Jiang[*], Yuzhao Peng[*], Houde Qian[*], Kaisen Yang[*,†], Boshi Zhang[*], Youjie Zheng[*], Bohan Lyu, Bingxiang He, Eren Cai, Calvin Xiao, Qinhuai Na[‡]

[*]Equal contribution. Names are sorted in alphabetical order of the last name.
[†]Project lead.
[‡]Corresponding author.

This project is founded and supported by Navers Lab, Einsia.AI.

References

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks, 2025. https://arxiv.org/abs/2407.05291.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, et al. Web-shepherd: Advancing PRMs for reinforcing web agents. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=G2kMroO9UV.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The browsergym ecosystem for web agent research, 2025. https://arxiv.org/abs/2412.05467.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. https://arxiv.org/abs/2306.06070.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024. https://arxiv.org/abs/2403.07718.

Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=IpzRWE52yw.

Hang Gao and Yongfeng Zhang. Memory sharing for large language model based agents, 2024. https://arxiv.org/abs/2404.09982.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis, 2024. https://arxiv.org/abs/2307.12856.

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, 2024a. https://arxiv.org/abs/2401.13919.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization, 2024b. https://arxiv.org/abs/2410.19609.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024. https://arxiv.org/abs/2401.13649.

Kuan Li, Zhongwang Zhang, Huifeng Yin, et al. Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=HuP16O5SJf.

Zhaoyang Liu, JingJing Xie, Zichen Ding, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=yBFUqdJFZn.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue, 2024. https://arxiv.org/abs/2402.05930.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. https://arxiv.org/abs/2310.08864.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, et al. WALT: Web agents that learn tools. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=cgIDqcJcoI.

Junhong Shen, Hao Bai, Lunjun Zhang, et al. Thinking vs. doing: Improving agent reasoning by scaling test-time interaction. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=un1TRwNgiv.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. https://arxiv.org/abs/2303.11366.

Dheeraj Vattikonda, Santhoshi Ravichandran, Emiliano Penaloza, et al. How to train your LLM web agent: A statistical diagnosis. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=67xkPEM3bZ.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. https://arxiv.org/abs/2305.16291.

Xinyuan Wang, Bowen Wang, Dunjie Lu, et al. Opencua: Open foundations for computer-use agents. In *Advances in Neural Information Processing Systems*, 2025a. https://openreview.net/forum?id=6iRZvJiC9Q.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks, 2025b. https://arxiv.org/abs/2504.06821.

Jialong Wu, Baixuan Li, Runnan Fang, et al. Webdancer: Towards autonomous information seeking agency. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=quJdphBcdP.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. https://arxiv.org/abs/2404.07972.

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, 2025. https://arxiv.org/abs/2412.09605.

Qianlan Yang, Xiangjun Wang, Danielle Perszyk, and Yu-Xiong Wang. Self-guided hierarchical exploration for generalist foundation model web agents. In *Advances in Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=9twwDW60Bw.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023. https://arxiv.org/abs/2207.01206.

Rui Ye, Zhongwang Zhang, Kuan Li, et al. Agentfold: Long-horizon web agents with proactive context folding. In *International Conference on Learning Representations*, 2026. https://openreview.net/forum?id=IuZoTgsUws.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. Agent lumos: Unified and modular training for open-source language agents, 2024. https://arxiv.org/abs/2311.05657.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. https://arxiv.org/abs/2308.10144.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024. https://arxiv.org/abs/2401.01614.

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, 2024. https://arxiv.org/abs/2307.13854.