SCALABLE BEHAVIOUR CLONING ON BROWSER USING VIA SKILL DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Browser demonstrations encode procedural knowledge about where to navigate, which information to preserve, and how to verify task completion. Reusing this knowledge requires separating the procedure from the incidental details of a recorded session. We present BROWSERBC, a framework that distills browser trajectories into structured natural-language skills and organizes them in a skill graph. Each skill captures applicability conditions, execution steps, progress checks, completion evidence, and recovery guidance. The executor interprets these instructions against the live page, enabling skills to be produced and consumed by different models. Evaluations on WebArena-Hard and ClawBench show improved task completion under task-family and rubric-informed, case-matched guidance, respectively, with shorter interactions on WebArena-Hard. Cross-model experiments demonstrate skill reuse across executors, while real-world scenario tests show benefits from greater skill availability and graph organization in both success and token efficiency. Desktop cases illustrate how procedural guidance supports navigation and completion checks while revealing persistent execution difficulties. Together, these findings support skill distillation as a way to accumulate and share procedural knowledge across agents.

1 INTRODUCTION

Web browsers connect code hosting, enterprise dashboards, research tools, and everyday online services. Reliable browser agents could make these workflows accessible through natural-language instructions. Progress in grounded web interaction (Yao et al., 2022; Deng et al., 2023) has motivated increasingly realistic environments that test long-horizon execution, visual understanding, and enterprise workflows (Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024; Chezelles et al., 2025).

A browser task requires decisions whose consequences extend beyond the current page. An agent must find an entry point, preserve constraints across steps, and verify that the requested change has taken effect. A single observation rarely reveals the complete procedure, making prior experience valuable for these decisions. Planning and long-context reasoning help connect observations across steps (Gur et al., 2024), while element grounding and operational knowledge remain challenges (Zheng et al., 2024; Xie et al., 2024). We focus on the procedural knowledge that can guide these decisions: where to go, what to preserve, and what to verify before moving on.

Human browsing traces offer a source of this knowledge. A demonstrated path reveals how a user navigates, carries information between steps, responds to feedback, and checks the outcome. Existing resources capture complementary evidence through action demonstrations, dialogue-conditioned navigation, and trajectories synthesized from web tutorials (Deng et al., 2023; Lù et al., 2024; Xu et al., 2025). Making these procedural relationships explicit would allow another agent to draw on the demonstrated experience.

Raw trajectories, however, mix task progress with repeated attempts, coordinates, DOM details, and transient state. Replaying these details ties a procedure to its recorded page. We take a structured natural-language *skill* as the reusable unit of demonstrated behavior. A skill describes when a procedure applies, how to advance it, what evidence establishes completion, and how to recover from failures. For example, a reporting skill preserves the relationship between filters, aggregation, and output checks while the executor locates the current controls. Procedural constraints remain explicit, and concrete actions adapt to live observations.

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.

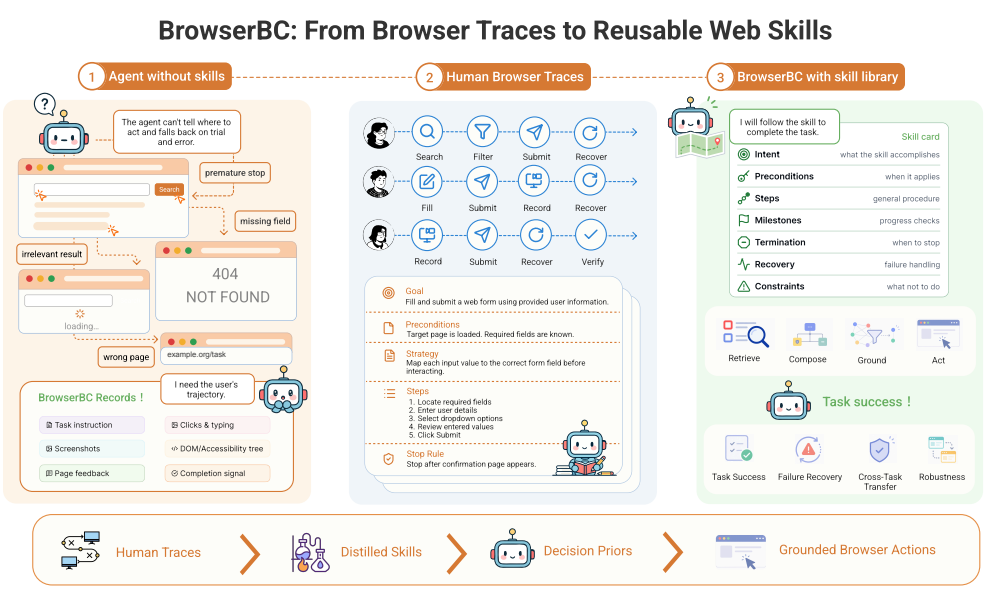


Figure 1: **Motivation of BrowserBC.** BROWSERBC turns human browsing traces into reusable natural-language skills, providing procedural guidance for navigation, state preservation, and completion checks.

Accumulating this knowledge requires both sharing skills across agents and organizing them within a library. Prior work shows how textual reflections and lessons guide later attempts (Shinn et al., 2023; Zhao et al., 2024), and how executable skill libraries preserve action procedures (Wang et al., 2023; 2025b). Natural-language cards let one model distill experience for another to execute. As demonstrations accumulate, overlapping procedures require consolidation, while specialized variants must retain their applicability conditions. These needs motivate a skill graph that records relationships among cards as the library grows.

We propose **BROWSERBC** to implement this approach. It cleans and segments browser trajectories into behavioral evidence, distills structured skill cards, and organizes them through consolidation and specialization. During execution, selected cards supply procedural guidance; the agent grounds each step in the live page and checks its effects. This separation allows a demonstrated procedure to be retained and reused across executors.

We evaluate task-family guidance on WebArena-Hard and rubric-informed, case-matched guidance on ClawBench. The results show improved task completion and fewer WebArena-Hard tool calls. Further studies examine cross-model reuse, skill availability and graph organization in real-world GitLab cases, and procedural guidance in controlled desktop tasks.

Contributions. This paper makes the following contributions:

- We formulate browser demonstration reuse through structured natural-language skills that make applicability, progress, completion, and recovery explicit.
- We develop BROWSERBC, combining evidence abstraction, skill distillation, and graph-based library organization with an execution interface shared across models.
- We provide empirical evidence for improved task completion and interaction efficiency under matched guidance, cross-model skill reuse, and the benefits of skill availability and graph organization.

2 RELATED WORK

Web agents and demonstrations. WebArena and VisualWebArena evaluate executable web workflows, while OSWorld extends agent evaluation to desktop applications (Zhou et al., 2024; Koh et al.,

2024; Xie et al., 2024). Mind2Web provides demonstrations for grounding and action prediction; WebLINX records dialogue-conditioned navigation; AgentTrek synthesizes trajectories from web tutorials (Deng et al., 2023; Lù et al., 2024; Xu et al., 2025). These resources provide actions together with their surrounding context. BROWSERBC extracts procedural relationships into contextual guidance, while the executor grounds each action in the live page. This complements learning to predict actions directly from demonstrations.

Skills and reusable experience. Reflexion stores verbal feedback for later attempts, and ExpeL extracts natural-language knowledge from past tasks (Shinn et al., 2023; Zhao et al., 2024). These methods establish a basis for reusing experience in textual form. Voyager maintains executable code skills, and programmatic skill induction constructs and verifies skills for digital tasks (Wang et al., 2023; 2025b). The latter reports gains over a text-skill counterpart in its evaluation. BROWSERBC uses a browser-specific textual schema with explicit preconditions, progress checks, terminal evidence, and recovery rules. We evaluate this interface through matched guidance, cross-model reuse, and comparisons of skill availability and library organization on real-world GitLab cases.

Agent planning and execution. SeeAct studies element grounding, while WebAgent combines planning, long-context reading, and program synthesis (Zheng et al., 2024; Gur et al., 2024). WALT extracts callable website tools, and AgentFold manages interaction history through context folding (Prabhu et al., 2026; Ye et al., 2026). Our focus is demonstration-derived procedural knowledge supplied to the executing policy. Appendix E discusses additional benchmarks, learning approaches, and cross-agent experience reuse.

3 METHOD

BROWSERBC is a *skill-centric* behavior cloning framework that converts browser demonstrations into structured natural-language procedures. Here, behavior cloning transfers demonstrated procedures into an external skill library, which conditions the agent's actions on the current observation. The framework comprises behavioral evidence abstraction (§3.3), procedural skill distillation (§3.4), skill-library construction (§3.5), and skill-conditioned execution (§3.6).

3.1 MOTIVATION

A procedure links actions to the states they must establish. A report depends jointly on its date range, aggregation period, and status filter; a completed form still requires confirmation that submission succeeded. Demonstrations reveal these dependencies, which a flat action log leaves implicit.

Skill cards make these dependencies explicit. Preconditions describe when a procedure applies, progress milestones state what should be observed along the way, and terminal evidence specifies the outcome needed before stopping. Recovery guidance links observed failures to corrective actions. The executor grounds these instructions in the controls and operations available on the current page.

3.2 PROBLEM FORMULATION

We denote a browser trajectory as $\tau = (x, e_1, \ldots, e_T, y)$, where x is the task instruction, $e_t = (o_t, a_t, f_t)$ comprises the observation, action, and execution feedback, and y is the outcome when available. Feedback includes page transitions, validation messages, and completion signals. Standard behavior cloning fits an action policy to demonstrations. Here, demonstrations are distilled into a *skill library* $\mathcal{L} = \{s_i\}_{i=1}^{N}$, where each s_i expresses a procedure supported by one or more trajectories.

Given a target instruction $x^\star$ and observation history $h_t^\star = o_{\leq t}^\star$, the agent receives a compact subset of skills and predicts its 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 operator $\mathcal{R}$ determines the supplied guidance, while π_θ maps the task, current observations, and guidance to an action. The cards specify which values to preserve and which states to verify. Because they are stored separately from the policy, the same card can guide different executors. Section 4.1 describes the task-family and case-matched guidance used in the evaluation.

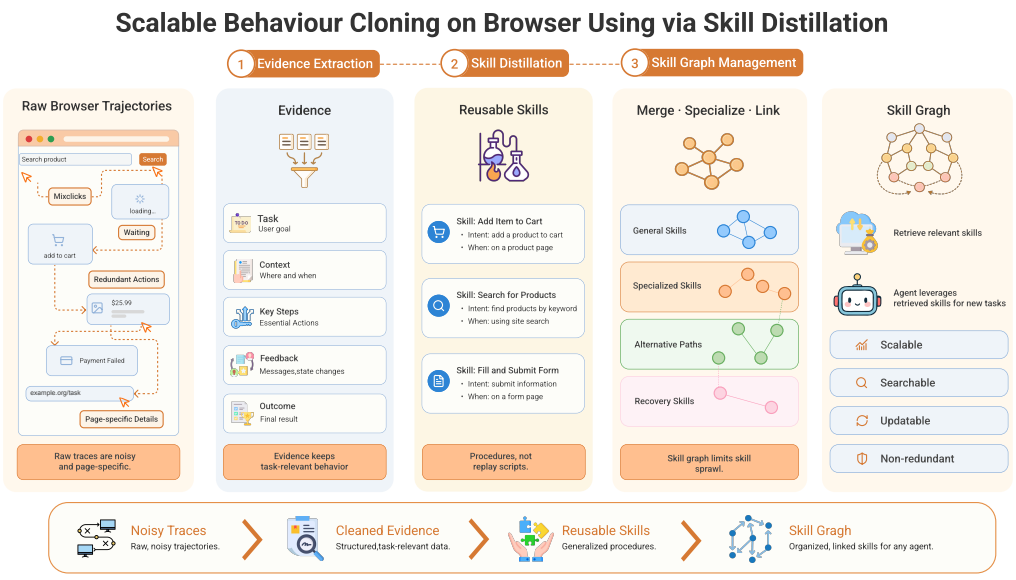


Figure 2: **Overview of BROWSERBC.** Trajectories are abstracted into evidence units and distilled into natural-language skills. A graph records relationships among skills. Selected cards guide execution, while the agent identifies concrete actions from the live page.

3.3 BEHAVIORAL EVIDENCE ABSTRACTION

Raw trajectories mix task progress with repeated attempts, idle waits, misclicks, and transient page content. The first stage normalizes events into a common representation, filters interactions that do not contribute to progress, and segments the trace into semantically coherent sub-procedures. Segmentation follows meaningful changes in task state so that each segment retains the context needed to interpret the demonstrated behavior.

For each segment, we extract an *evidence unit*

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

where c^{pre} and c^{post} summarize the contexts before and after the segment, b summarizes the behavior, and f and y retain feedback and the available outcome. Contexts preserve task-relevant affordances, input fields, constraints, and success or failure signals. Session-specific coordinates, selectors, identifiers, and private values are removed from reusable descriptions.

The pre- and post-contexts distinguish an attempted operation from an observed outcome. Clicking Save is evidence of an action; observing the updated record is evidence of its effect. A validation message can similarly reveal that a required field was omitted. These observations provide the evidence for progress checks, stopping conditions, and recovery instructions.

3.4 PROCEDURAL SKILL DISTILLATION

Given evidence units $\mathcal{U}_s$ and optional metadata m_s, such as task category or website family, the distiller produces a candidate card $\hat{s} = D(\mathcal{U}_s, m_s)$. A card contains intent, applicable scope, preconditions, execution steps, progress milestones, terminal evidence, failure modes, recovery strategies, negative constraints, and provenance. Together, these fields describe how to apply a procedure and verify its outcome.

The distiller is instructed to express reusable procedural relationships and remove session-specific artifacts. The content audit checks for checker-only answers, stale literals, and private runtime values. Task-provided values remain inputs to the current execution. ClawBench construction additionally supplies evaluation rubrics, human demonstrations, and available completion evidence to the distiller; content filtering does not remove this supervision (§4.1).

A form-filling card, for example, instructs the executor to identify semantic fields, preserve task-provided values, check validation feedback, and verify acceptance. A successful trajectory supports one viable procedure; multiple attempts supply evidence about preconditions and recovery. Provenance links each card to its source demonstrations.

3.5 Skill-Library Construction

Repeated demonstrations can yield overlapping procedures. Each candidate skill can be added as a new entry, merged with an existing entry, or registered as a specialization. Compatibility is assessed in terms of intent, preconditions, steps, effects, and terminal evidence. Compatible descriptions can share a card, whereas different applicability conditions or conflicting constraints motivate separate entries. Consolidation groups compatible evidence within a shared procedure. Merging pools evidence for the same procedure; specialization preserves a shared procedure while restricting its applicability. Thus, cards with similar action sequences can remain separate when their preconditions or required completion evidence differ.

We organize the library as a skill graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$. Nodes are skill cards, and edges describe temporal dependencies, specializations, alternative procedures, or incompatibilities. For example, form filling can be associated with payment and profile-update specializations, along with recovery procedures for validation failures. The card remains the unit exposed to the executing model.

The graph supports consolidation and revision as experience accumulates, while the execution interface remains a selected textual procedure. The matched-guidance evaluation tests the utility of this interface. Section 4.4 examines library growth and the contribution of graph organization; automatic retrieval quality is a separate empirical question (§4.6).

3.6 Skill-Conditioned Execution

The general retrieval operator $\mathcal{R}$ selects guidance using semantic relevance and, when available, compatibility with the browser context. Our evaluation supplies task-family guidance on WebArena-Hard and a matched case-level or domain-level card on ClawBench. These settings test execution with relevant guidance; they do not separately measure automatic retrieval quality in a large library.

The selected cards are inserted into the agent's context. The execution prompt instructs the agent to prioritize the live observation when it conflicts with a card. Preconditions help determine applicability, progress milestones support intermediate checks, and terminal evidence guides stopping. If an expected transition fails, recovery rules suggest a response that the executor must ground in the current state.

The card therefore carries procedural knowledge, while the policy chooses concrete actions. The distiller–executor study tests this separation by varying the producer and consumer of guidance on the same task set.

4 Experiments

4.1 Experimental Setup

We examine whether procedural guidance improves task completion, reduces interaction length, and transfers across executors. The studies span self-hosted web tasks, live-site workflows, real-world GitLab cases, and controlled desktop tasks. Table 1 summarizes the benchmark and desktop settings; §4.4 describes the collected GitLab cases.

Evaluation settings. WebArena-Hard denotes our 258-task evaluation on self-hosted WebArena sites (Zhou et al., 2024). Its task-group counts match the released WebArena-Verified Hard subset: 57 GitLab, 56 Shopping, 55 Shopping-admin, 42 Reddit, and 48 multi-site tasks (ServiceNow, 2026). We use the original WebArena programmatic checkers with common normalizations and manual adjudication. Appendix A.1.4 describes the scoring procedure. ClawBench evaluates workflows on live production websites using an agentic evaluator and human-reference criteria (Zhang et al., 2026). Our evaluation contains 152 tasks, reported in eight task groups. The original ClawBench paper describes 153 tasks; the maintained V1 corpus contains 152 after a task removal (TIGER-AI-Lab,

2026). The desktop study consists of 30 constructed Ubuntu cases inspired by OSWorld (Xie et al., 2024).

Table 1: Evaluation settings and scoring. The desktop study uses a constructed case set rather than the full OSWorld benchmark.

Setting	Environment	Verification	Evaluation focus
WebArena-Hard	Self-hosted web	Programmatic checkers	Task completion in a self-hosted environment
ClawBench	Live production websites	Agentic pass/fail evaluator	Live-site workflows and state changes
OSWorld-style cases	Ubuntu desktop apps	State-based verifiers	Procedural guidance on desktop tasks

Agents and comparison. Unless otherwise stated, both the distiller and executor use Claude-Sonnet-4.6. The skill-off and skill-on conditions use the same executor, action interface, and step budget. Skill-on adds demonstration-derived procedural guidance to the executor's context. This comparison measures the benefit of supplying the additional information as a skill card. The cross-model study (§4.3) varies distiller and executor separately. Success rate is the fraction of evaluated tasks scored as successful; aggregate rates weight each task equally. Reported deltas are differences between the displayed success rates in percentage points.

Skill construction and supervision. WebArena-Hard skills are distilled from demonstrations associated with the same task family. ClawBench uses an iterative protocol: cases are first run without the skill library, successful demonstrations are distilled, and later runs receive a matched case-level or domain-level skill. Distillation inputs include the task description, evaluation rubric, verified human demonstrations, and available completion evidence. Each subsequent ClawBench run receives one rubric-informed, matched card. Appendix A.1 describes the construction and execution protocols. Both evaluations concern matched guidance; their relation to held-out transfer is discussed in §4.6.

Scoring and content controls. WebArena-Hard uses programmatic checks, with manual adjudication when required before aggregation. The same checker normalizations and pass/fail standard apply to both conditions. ClawBench uses an agentic pass/fail evaluator whose records include the applied rubric and supporting evidence. The skill-content audit removes checker-only answers, stale literals, and private runtime values, or treats them as non-copyable inputs. The cards retain the rubric-informed supervision described above.

4.2 Main Results

Table 2 reports results by task group for the same browser agent without skills (Base) and with the BrowserBC skill library.

WebArena-Hard. With matched skills, overall success increases from 60.5% $(156/258)$ to 81.4% $(210/258)$, a $+20.9$-point difference in the displayed rates. Gains occur in every task group and are largest in Multi-site ($+31.2$ points) and Shopping ($+28.6$ points). Appendix C illustrates the resulting changes in navigation and state verification.

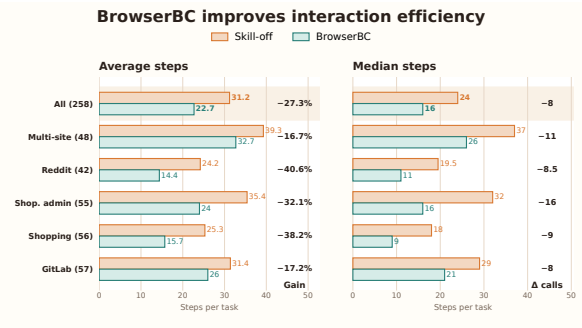
ClawBench. The skill-off agent solves $50/152$ tasks (32.9%), while the skill-conditioned agent solves $104/152$ (68.4%), a $+35.5$-point difference. All eight task groups improve under the rubric-informed, case-matched protocol. The improvement extends across the live-site workflows evaluated under this protocol.

4.3 Analysis and Ablations

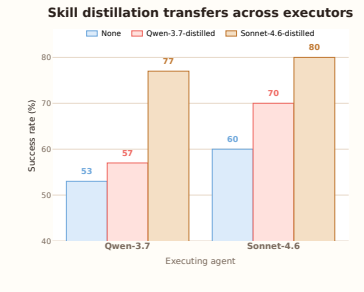
Interaction length. Across all WebArena-Hard tasks, mean tool calls decrease from 31.2 to 22.7 (-27.3%), and the median decreases from 24 to 16 $(-8$ calls). Figure 3a reports the task-group breakdown. Calls are aggregated over all tasks, rather than restricted to those solved in both conditions.

Table 2: Performance comparison across WebArena-Hard and ClawBench. All Base and BROWSERBC results are reported as success rates (SR, %), and Δ SR denotes the absolute improvement in success rate.

Task group / category	# Cases	Base (Skill-Off)	BROWSERBC	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	57	24.6	64.9	+40.3
Finance	6	50.0	100.0	+50.0
Work	17	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	16	25.0	56.2	+31.2
Pets	11	27.3	54.5	+27.2
All	**152**	**32.9**	**68.4**	**+35.5**



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



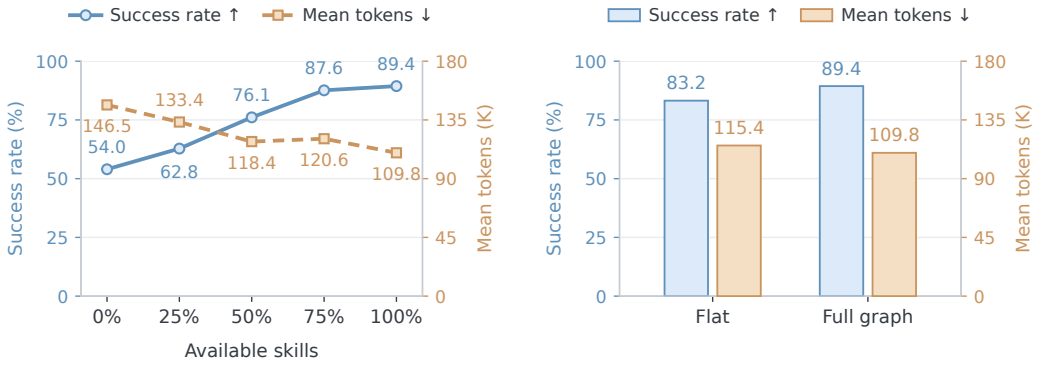
(b) Decoupling the distiller from the executor.

Figure 3: BROWSERBC improves interaction efficiency and transfers distilled skills across executors. Left: BROWSERBC reduces the average number of 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.

Higher success thus accompanies shorter online interactions; total cost also includes distillation, storage, and skill context.

Distiller–executor separation. We vary the model that distills a skill and the model that executes it on 30 read-only WebArena-Hard tasks, using the same source trajectories. Figure 3b shows six conditions: two executors, each receiving no skill, a Qwen-3.7-plus-distilled skill, or a Sonnet-4.6-distilled skill. The figure abbreviates Qwen-3.7-plus as Qwen-3.7 and reports rounded success percentages.

Sonnet-distilled skills improve both executors. The Qwen executor reaches 77% and the Sonnet executor reaches 80%, compared with 53% and 60% without skills. Qwen-distilled skills yield 57% and 70%, respectively. Both skill sources improve both executors, supporting cross-model reuse on this task set. One task corresponds to about 3.3 percentage points, limiting the resolution of the comparison between guided executors.



(a) Growth with additional skills.

(b) Effect of graph organization.

Figure 4: **Skill availability and library organization on 113 GitLab cases collected from real-world scenarios.** Left: success rate and mean token usage as the fraction of available skills increases; 0% provides no skills. Right: a flat library without graph organization versus the full graph-organized library. Blue uses the left axis for success rate; orange uses the right axis for mean token usage (K denotes thousands). Higher success and lower token usage are preferred.

Adapting guidance to the live state. An always-follow variant, which prevents the agent from overriding the supplied skill based on the live state, reaches 77.5% on WebArena-Hard, compared with 81.4% for BROWSERBC. This comparison supports allowing the executor to adapt the supplied procedure to current observations. The desktop cases below examine how procedural guidance interacts with execution errors.

4.4 LIBRARY GROWTH AND GRAPH ORGANIZATION

We collect 113 GitLab cases from real-world scenarios to examine how the availability of skills affects task performance and token usage. We evaluate without skills and with 25%, 50%, 75%, and 100% of the skill library. The evaluation set remains fixed; the fractions specify the proportion of skills made available.

Growth with additional skills. Figure 4a reports success rate and mean token usage on the 113 cases. Increasing the available skill fraction raises success from 54.0% without skills to 89.4% with the full library, a gain of 35.4 percentage points. Mean token usage decreases from 146.5K to 109.8K, a 25.1% reduction. Token usage rises from 118.4K at 50% to 120.6K at 75%, while success continues to improve. The full library therefore combines the highest success rate with the lowest mean token usage among the tested fractions. The intermediate token increase shows that higher success need not coincide with lower token usage at every library fraction; the overall efficiency gain is clearest when comparing the full library with the skill-off baseline.

Contribution of graph organization. On the same 113 GitLab cases, we compare a flat library without graph organization against the full graph-organized library (Figure 4b). Success increases from 83.2% to 89.4%, a gain of 6.2 percentage points, while mean token usage decreases from 115.4K to 109.8K (−4.9%). The graph-organized library therefore combines higher task success with lower token usage in this comparison.

4.5 BEYOND THE BROWSER: OSWORLD-STYLE CASE STUDY

We apply task-family skills to 30 controlled Ubuntu desktop cases with fixed initial states and independent state-based verifiers. The cases cover settings, file and text editing, terminal commands, application configuration, dialogs, and office applications. Skills distilled from successful task-family demonstrations describe procedures that the executor grounds in the current desktop state. The base agent solves 11 cases; matched skills enable 6 additional successes, while 13 remain unsolved in both

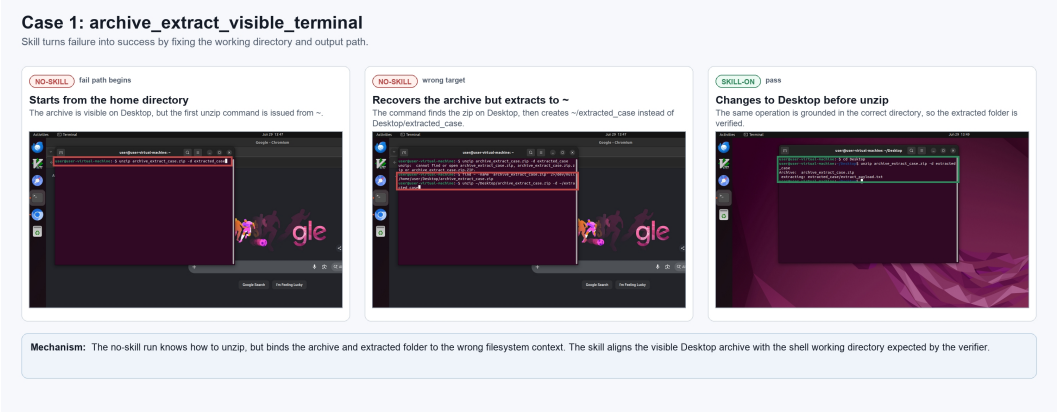


Figure 5: **A desktop case: archive extraction.** The skill-off agent knows the extraction command but uses the wrong filesystem context. The matched skill establishes the Desktop working directory and checks the extracted output.

settings. This diagnostic study is not an OSWorld leaderboard result or a full-benchmark performance estimate.

Establishing the correct filesystem context. In Figure 5, the skill-off agent knows the extraction command but first runs it from the home directory. It later locates the Desktop archive yet creates the extracted folder under the wrong filesystem context. The matched skill specifies the missing procedure: establish the Desktop working directory, extract the archive there, and check the resulting folder. The skill makes the required working context and output checks explicit. This distinguishes successful command execution from task completion: the extracted files must also appear in the requested location.

Avoiding unnecessary recovery. In the script case, both conditions succeed. The skill-off agent assumes a home-directory path and then searches for the script; the matched skill supplies the Desktop path and avoids recovery. Together, the archive and script cases illustrate more reliable state changes and shorter successful procedures. Execution failures leave 13 cases unsolved, and extra verification can lengthen runs. Appendix D provides the script screenshots, protocol details, and outcome taxonomy.

4.6 Scope of the Evidence

The WebArena-Hard and ClawBench results establish the utility of matched guidance under the reported supervision. They do not isolate card structure or abstraction from additional task-relevant information. ClawBench skills also use evaluation rubrics. The GitLab study tests skill availability and graph organization on 113 cases; larger libraries and task-family or website holdouts would test scalability and transfer beyond these settings. A complete efficiency assessment must include distillation and execution costs. The desktop study provides diagnostic cases rather than a full-benchmark estimate.

5 Conclusion

BrowserBC converts browser demonstrations into natural-language skills with explicit applicability, progress checks, completion evidence, and recovery guidance. Under task-family guidance on WebArena-Hard and rubric-informed, case-matched guidance on ClawBench, skills improve task completion and reduce WebArena-Hard tool calls. Cross-model experiments demonstrate reuse across executors, and real-world GitLab tests show benefits from greater skill availability and graph organization. Desktop cases show how procedures address path and completion errors while exposing persistent execution failures. These results support separating procedural knowledge from its executor and motivate testing reuse across larger libraries and held-out tasks.

AI USE STATEMENT

Large language models were used to assist with language polishing, including correcting grammar, refining phrasing, and improving the clarity and readability of the manuscript. The authors reviewed and revised the suggested edits to ensure that they accurately conveyed the intended technical meaning. All experimental results and implementation details reported in the paper were checked and confirmed by the authors. The authors take full responsibility for the accuracy and integrity of the final manuscript, including its technical claims and conclusions.

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. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0b82662b6c32e887bb252a74d8cb2d5e-Abstract-Datasets_and_Benchmarks_Track.html.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, et al. Web-shepherd: Advancing PRMs for reinforcing web agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5bac36213824c6e0b0a173df8e2276a6-Abstract-Conference.html.

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. URL 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. URL 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. URL 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. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e0c50c0e78db8d9d881c79cac775e499-Abstract-Conference.html.

Hang Gao and Yongfeng Zhang. Memory sharing for large language model based agents, 2024. URL 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. URL 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. URL 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. URL 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. URL 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. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9760f18785acf535321e50405fd36578-Abstract-Conference.html.

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. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/666c1861d709bd84e20b6e0e02a2c223-Abstract-Conference.html.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue, 2024. URL 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*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL 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. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5b175f9e93873e3a10a6ce43dbb82e05-Abstract-Conference.html.

ServiceNow. WebArena-Verified. Software repository, 2026. URL https://github.com/ServiceNow/webarena-verified.

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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f7c4783621a20f5f11a316cd249e0252-Abstract-Conference.html.

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

TIGER-AI-Lab. ClawBench: Can AI agents complete everyday online tasks? Software repository, 2026. URL https://github.com/TIGER-AI-Lab/ClawBench.

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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4d239452d41ded07139801d7c966c629-Abstract-Conference.html.

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. URL 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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/cc7ae529e945226b0d52ea4ac478c4f3-Abstract-Conference.html.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks, 2025b. URL 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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/af043aee9cb137785d93195c6cf4cd96-Abstract-Conference.html.

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. URL 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. URL 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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2852977ade3d2a70eceb78cdef91f4b3-Abstract-Conference.html.

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*, 2022. URL https://papers.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Rui Ye, Zhongwang Zhang, Kuan Li, et al. Agentfold: Long-horizon web agents with proactive context folding. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fa948624dfde013671e72c1a7ca4aebc-Abstract-Conference.html.

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. URL https://arxiv.org/abs/2311.05657.

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Zhuofeng Li, Xingwei Qu, Zhengkang Guo, Yuanzhe Shen, Dingjie Song, Han Zhou, Tuney Zheng, Xian Wu, Hao Yu, Songcheng Cai, Yi Lu, Yunzhuo Hao, Minyi Lei, Liang Chen, Kai Zou, Huifeng Yin, Wendong Xu, Dongfu Jiang, Ping Nie, Jiaheng Liu, Wenhu Chen, and Kelsey R. Allen. ClawBench: Can AI agents complete everyday online tasks?, 2026. URL https://arxiv.org/abs/2604.08523.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL 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. URL 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. URL https://arxiv.org/abs/2307.13854.

A APPENDIX

Appendix guide

This appendix details the skill representation, benchmark-specific protocols, and browser case demonstrations. Appendix D presents the controlled desktop study.

A.1 REPRESENTATION AND EVALUATION PROTOCOLS

A.1.1 SKILL OBJECT FORMAT

Across the web experiments, skills are represented as structured natural-language cards. Each card specifies the task family it applies to, its preconditions, a concise procedure, progress and completion evidence, common failure modes, recovery or abort rules, and provenance. The card supplies procedural knowledge, while live observations determine concrete labels, element identities, page availability, validation messages, and submission state.

In WebArena-Hard, this format is specialized to self-hosted browser tasks with programmatic checkers. A skill records the task pattern, the relevant site state, the semantic operations for advancing the task, and the observable evidence required before stopping. For example, a skill may say to construct a report with a particular aggregation period and status filter, or to verify a committed file state, but it does not prescribe fixed coordinates or a fixed action sequence.

In ClawBench, the same skill format is adapted to live websites where terminal states may be visible in the page, reflected in a submitted request, or judged by an agentic evaluator. The skill therefore emphasizes state preservation, pre-submission checks, false terminal states, and recovery order. Across both benchmarks, skills are rendered as agent-facing procedural guidance: they state which target state must be established and how completion should be verified, without copying stale coordinates, DOM selectors, account tokens, verification codes, or other private runtime values.

A.1.2 WEBARENA-HARD HARNESS

WebArena-Hard is evaluated in the benchmark's self-hosted environment using the WebArena task definitions and original programmatic checkers. Unless stated otherwise in the main text, the skill-off and skill-on conditions use the same executor, browser action interface, authentication state, and step budget. Skill-on adds the distilled procedural guidance.

WebArena-Hard uses task-family guidance: skills are distilled from demonstrations associated with the evaluated task family. During execution, a retrieved skill is appended as procedural guidance rather than as a mandatory plan: the prompt explicitly instructs the agent to trust the live browser observation whenever it conflicts with the skill. In administrative tasks, for example, the skill may specify report construction while the executor identifies the current page controls and filter state.

A.1.3 CLAWBENCH HARNESS

ClawBench is evaluated on live production websites with a fresh browser state per case. The results in Table 2 use the final 152-case evaluation set and aggregate outcomes by task category. The cards supply procedural guidance that the agent interprets against the live page.

The skill-on pipeline is iterative and uses rubric-informed, matched guidance. Cases first run without a skill library; successful demonstrations are then distilled into reusable skills. On later runs, each case receives a single matched case-level or domain-level skill. This one-skill-per-case intervention evaluates execution with targeted procedural guidance. A fresh browser state resets the interaction session; it does not define a held-out split of the source demonstrations.

The distillation step draws on the benchmark task description, evaluation rubric, verified human demonstrations, and observable completion evidence when available. The resulting skills emphasize state preservation and terminal evidence. For instance, if a successful human path preserved upstream search filters or booking constraints before opening a final contact, checkout, creation, or submission

page, the skill expresses those constraints as checks to repeat in the live browser before final submission.

A.1.4 Scoring and Skill-Content Controls

WebArena-Hard uses the benchmark's original programmatic checkers, whereas ClawBench uses an agentic pass/fail evaluator that applies the benchmark rubric to the completed trajectory. WebArena outcomes are aggregated after common checker normalizations and manual adjudication when required, under the same pass/fail standard for both conditions. The ClawBench evaluator records its justification, the applied rubric rules, and the supporting evidence for audit.

The paired skill-off and skill-on conditions use the same executor, action interface, and step budget. Skill-on additionally receives the constructed procedural guidance. Skills are restricted to procedural content: they may describe what state to establish, which task-provided values must be preserved, what progress looks like, and what terminal evidence confirms completion, but they may not encode checker-only answers or stale runtime literals. Private or volatile values, such as account tokens, verification codes, payment details, generated identifiers, and session-specific contact information, are redacted or treated as non-copyable runtime values.

B Trajectory and Skill Representations

This appendix visualizes the representations used by BrowserBC and gives WebArena case demonstrations. A source browser trajectory is distilled into a Markdown skill and used to guide a fresh agent rollout. The boxes specify the representation fields, and the figures illustrate individual rollouts.

Representation sketches. The following schemas show the normalized trajectory view and the distilled skill view. Angle-bracket fields are schema placeholders. The former preserves task evidence; the latter abstracts that evidence into reusable procedural guidance.

Demonstration card

```
## Metadata
site: <site>
intent: <user request>
## Step sequence
- url: <page>
  observation: <visible state>
  action: <browser action>
  evidence: <why it matters>
## Stop record
answer: <result>
stop_signal: <checked evidence>
```

Skill card

```
## Metadata
pattern: <task pattern>
goal: <what to accomplish>
## Procedure
- precondition: <site state>
  navigate: <target page>
  act: <browser action>
  verify: <why it works>
## Stop rule
answer: <result type>
stop_signal: <completion
evidence>
```

C WebArena Case Demonstrations

The case demonstrations compare a base rollout with a skill-conditioned rollout for the same task. Each panel uses real browser screenshots from the recorded rollout and a short caption describing the behavior encoded by the trajectory or skill.

C.1 Additional Results

This section summarizes the scoring and interaction-efficiency checks behind the WebArena-Hard aggregate reported in the main text.

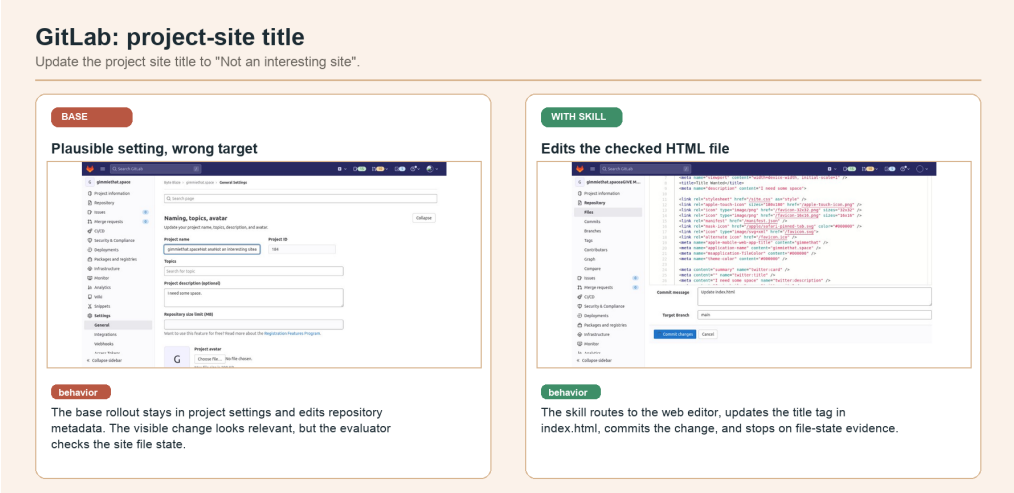


Figure 6: GitLab case demo. The base rollout changes a plausible project setting, while the skill-conditioned rollout edits the checked HTML file and commits the target state.

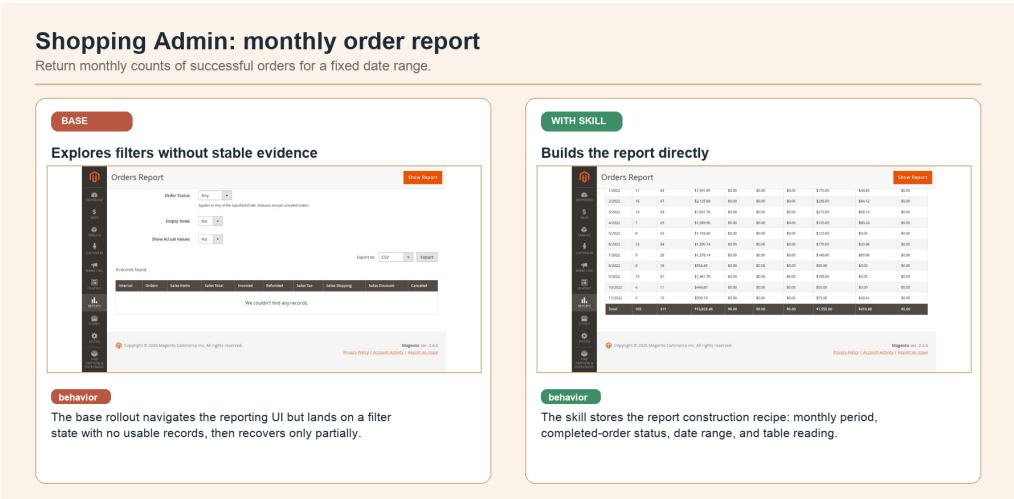


Figure 7: Shopping Admin case demo. The skill captures the report-construction procedure: date range, monthly aggregation, completed-order filter, and table reading.

Audited scoring. The final WebArena-Hard totals are 156 / 258 solved tasks for the skill-off baseline and 210 / 258 solved tasks for BROWSERBC, corresponding to 60.5% and 81.4% success respectively. Manual adjudication, when required, is applied before aggregation under the same pass/fail standard used by the benchmark checker. The totals aggregate one audited outcome per task.

Interaction efficiency. The interaction-efficiency summary in Figure 3a shows that the all-task median decreases from 24 to 16 tool calls, and mean tool calls decrease in every WebArena-Hard task group under skill conditioning. These measurements summarize online interaction length under the reported skill-off and skill-on conditions.

Qualitative appendix material. The qualitative material describes task context, the supplied skill, and observed execution behavior. The examples illustrate individual mechanisms; the aggregate comparison is given in Table 2.

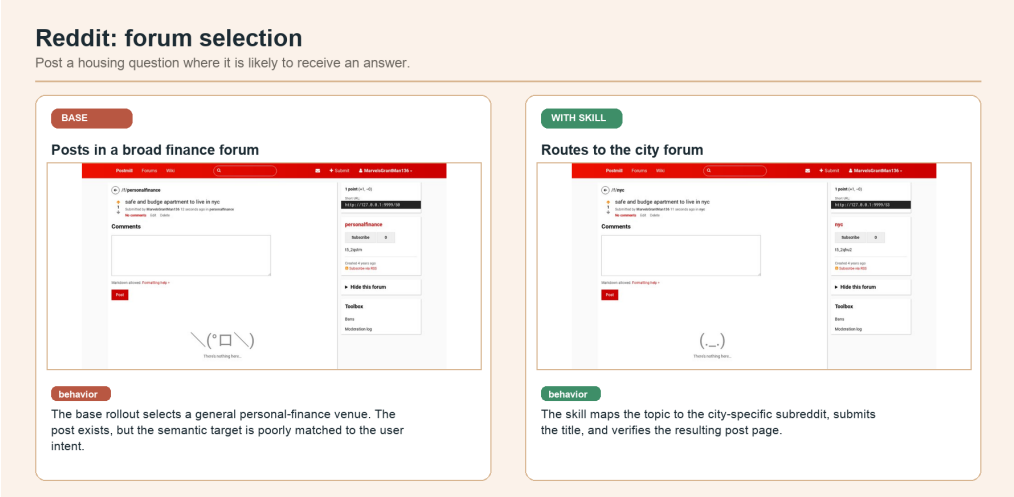


Figure 8: Reddit case demo. The skill encodes semantic forum selection and routes the housing question to the city-specific forum before verifying the created post.

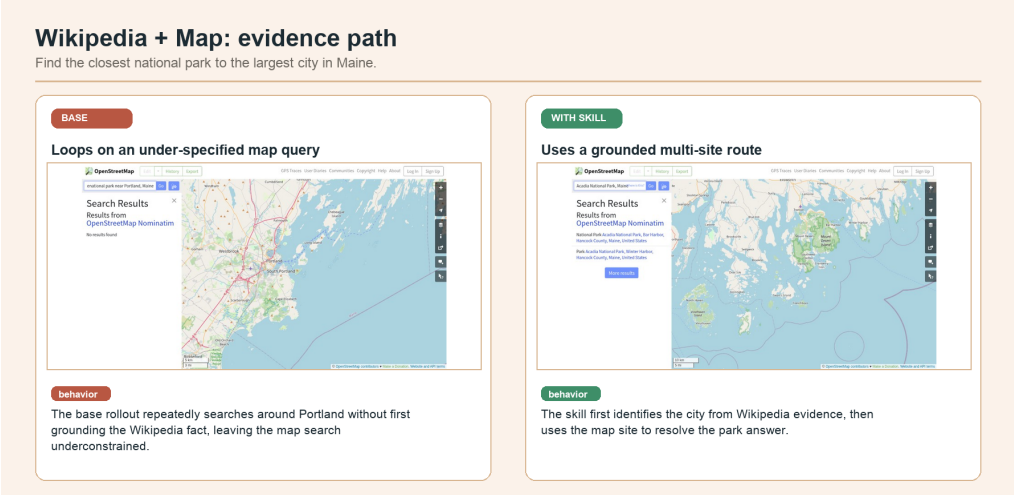


Figure 9: Wikipedia and map case demo. The skill preserves the cross-site evidence order: ground the city fact first, then query the map site for the park answer.

Table 3: Qualitative evidence in the appendix.

Material	Scope	Fields
Case-level evidence		
OSWorld-style cases	Ubuntu desktop	Initial state, procedure, and outcome taxonomy.
Browser examples	WebArena tasks	Matched skill, agent actions, and terminal evidence.
Execution analysis	Browser and desktop cases	Path errors, state verification, and recovery.

D OSWORLD-STYLE DESKTOP CASE STUDY

Framing and protocol. The desktop study uses the same skill fields as the browser experiments: preconditions, state transitions, progress evidence, terminal evidence, and recovery rules. It evaluates task-family guidance on a controlled case set, rather than the full OSWorld benchmark.

We construct 30 Ubuntu desktop cases covering settings changes, file and text editing, terminal commands, application configuration, modal dialogs, and office applications. Each case has a fixed initial state and an independent state-based verifier. Skills are distilled from successful demonstrations at the task-family level: a skill may describe that a setting must be opened, changed, persisted, and verified, or that a file must be edited and saved, but it does not encode coordinates or a replay script. We include cases where the skill matches the task, cases that the base agent already solves, and difficult cases where the procedure is known but execution fails. The outcome categories distinguish baseline successes, additional successes with guidance, and cases that remain unsolved.

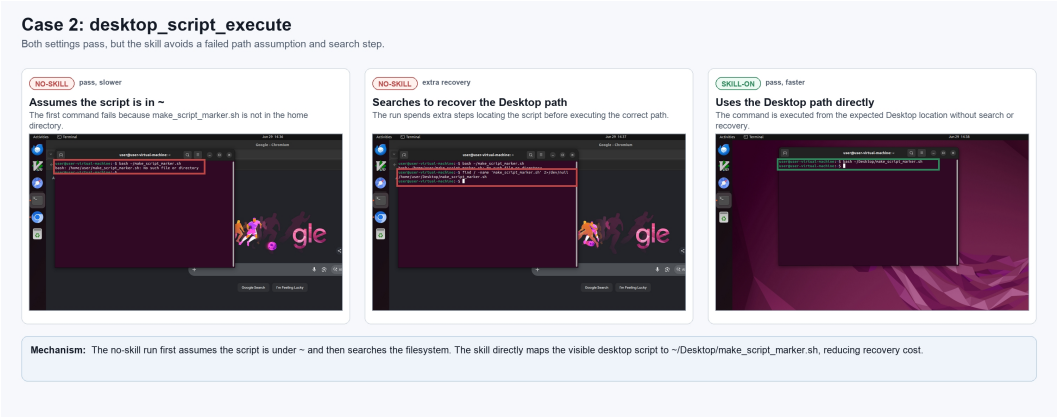


Figure 10: Desktop script execution. Both conditions succeed. The skill-off agent initially searches from the home directory, while the matched skill identifies the Desktop path and avoids the ensuing recovery. The archive-extraction case appears in Figure 5 in the main text.

Table 4: Outcome taxonomy for the OSWorld-style diagnostic study. Counts denote cases in this controlled case set: 11 are solved without the skill, 6 additional cases are solved with the skill, and 13 remain unsolved in both settings.

Case type	Count	Interpretation
Solved without skill	11	The base agent completes the task without the supplied skill.
Skill-enabled gain	6	The agent succeeds with the matched skill but not without it.
Unsolved	13	Failures persist due to focus, modal, file-picker, or recovery-state issues.

Case interpretation. The archive-extraction case in Figure 5 illustrates how a procedure can specify the filesystem context needed for the requested output. Section 4.5 describes the action sequence and the role of its completion check.

The desktop script case (Figure 10) illustrates reduced recovery effort. Both conditions succeed, but the skill-off run initially assumes a home-directory location and spends extra steps searching for the Desktop path. The matched skill supplies the Desktop path, avoiding this failed assumption and the subsequent recovery.

Scope. The archive and script cases show how a skill can supply a useful filesystem procedure and reduce avoidable recovery. Correct guidance still requires reliable execution: 13 cases remain unsolved in both conditions, with failures involving focus, modal dialogs, file pickers, or recovery state. Additional verification can also make a successful trajectory longer than necessary. These cases show why procedural guidance must be interpreted against the current GUI and filesystem state.

E Extended Related Work

Web and computer-use environments. WebShop studies grounded interaction in a simulated storefront (Yao et al., 2022). WorkArena and WorkArena++ target enterprise knowledge work, while BrowserGym provides a shared interface for web-agent research (Drouin et al., 2024; Boisvert et al., 2024; Chezelles et al., 2025). These settings complement the executable workflows in WebArena and the visually grounded tasks in VisualWebArena (Zhou et al., 2024; Koh et al., 2024). OSWorld exposes challenges in GUI grounding and operational knowledge across desktop applications (Xie et al., 2024). The controlled desktop cases in our study examine how explicit procedures interact with these execution difficulties.

Learning and interaction strategies. WebVoyager studies multimodal web interaction, while OpenWebVoyager develops agents through iterative exploration and feedback (He et al., 2024a;b). ScaleCUA and OpenCUA study computer-use data and supervision at larger scales (Liu et al., 2026; Wang et al., 2025a). Other approaches improve agents through reinforcement learning and information seeking (Li et al., 2026; Wu et al., 2025; Vattikonda et al., 2025), process rewards (Chae et al., 2025), hierarchical or structured exploration (Yang et al., 2025; Gandhi & Neubig, 2026), and test-time interaction scaling (Shen et al., 2025). These approaches address agent learning and exploration. BrowserBC focuses on representing demonstration-derived procedures and supplying them through the executor's context.

Modularity and experience transfer. Agent Lumos separates planning, grounding, and execution into modules (Yin et al., 2024). Memory sharing studies reuse across language agents (Gao & Zhang, 2024), and Open X-Embodiment provides evidence for transfer from heterogeneous robot demonstrations (Open X-Embodiment Collaboration et al., 2024). These lines of work motivate separating knowledge production from its use. Our cross-model experiment varies the producer and consumer of textual skills on browser tasks, while the real-world GitLab study examines their availability and organization.