WebArena Verified

Amine El Hattami[1,2,3] **Megh Thakkar**[1] **Nicolas Chapados**[1,2,3] **Christopher Pal**[1,2,3,4]

[1]ServiceNow Research [2]Mila – Quebec AI Institute [3]Polytechnique Montréal [4]Canada CIFAR AI Chair

Abstract

Autonomous web agents increasingly operate in multi-step browser workflows, yet widely used benchmarks can misestimate performance due to underspecified goals and brittle checkers—challenges characteristic of normal benchmark maturation rather than flaws in the paradigm. We present WebArena Verified, a reproducible re-evaluation of WebArena that preserves its containerized environments while strengthening measurement. We audit all 812 tasks, repair misaligned evaluations and clarify ambiguous instructions; replace substring matching with type- and normalization-aware comparators; verify backend state for state-changing tasks; and adopt a structured JSON schema with explicit status codes for deterministic scoring. We provide improved results reporting with template-level macro averages, 95% confidence intervals, and failure-mode breakdowns. We also introduce WebArena Verified Hard, a 137-task subset that retains difficult cases while reducing evaluation cost by 83%. On the baseline agent we evaluated, it reduces false negatives by approximately 11%. WebArena Verified remains drop-in compatible with minimal change to existing agents, supporting faithful and comparable progress. We release our code, data, and evaluation tools in our public repository.[1]

1 Introduction

Autonomous web agents increasingly execute complex web workflows. These workflows require dynamic navigation and extraction of both visual and textual information. As adoption grows, rigorous and reproducible evaluation is indispensable for progress. Existing benchmarks lay a solid foundation for measuring these capabilities. For instance, WebArena [16] offers self-hosted, containerized sites that emulate real-world environments, permitting task execution through standard browsers. The design of WebArena inspired a family of follow-up benchmarks—VisualWebArena [8], Mind2Web [4], WorkArena and WorkArena++ [5, 3], OSWorld [13], and AndroidWorld [11]—each expanding the scope of realistic tasks and interface modalities. However, widespread adoption has revealed systematic evaluation errors that undermine measurement validity [17, 8]. The benchmarks commonly employ underspecified success criteria, allowing disparate interpretations of task completion. Moreover, brittle evaluation mechanisms yield noisy or inconsistent metrics. Zhu et al. [17] conducted a recent audit of 37 public agent-benchmark suites—covering 13 web-interaction domains—that revealed pervasive quality issues that erode confidence in reported results. For example, a trivial "empty-output" agent achieved 38% success on τ-bench [15], outperforming GPT-4o-based agents on intentionally impossible tasks, and WebArena suffered from brittle string matching that inflated success rates by 1.4–5.2% [17].

These specific issues reflect broader patterns in the typical benchmark maturation, observed across machine learning benchmarks, yet they must be addressed. Without robust, transparent evaluation, researchers struggle to identify genuine performance gaps, increasing the risk of over-optimistic conclusions, benchmark-specific adaptations, and misleading comparisons while substantive deficiencies remain unaddressed. These reliability concerns have sparked efforts to improve evaluation

[1]https://github.com/ServiceNow/webarena-verified.

quality through principled debugging and systematic verification [2, 1, 3]. This pattern of "verified" re-releases demonstrates the value of strengthening evaluation mechanisms while preserving benchmark utility and adoption. For example, SWE-bench Verified agents achieved roughly doubled performance once flawed tasks were removed [2].

We introduce **WebArena Verified**, an update to WebArena that preserves the original containerized environments while improving evaluation reliability. Our approach re-verifies task definitions, reference answers, and evaluators using deterministic evaluation with explicit success criteria. We replace brittle string matching with backend state verification, standardize outputs to a structured JSON schema, and report template-level macro-averages with 95% confidence intervals, aligned with recent best-practice guidelines [17] (§4). By enforcing structured JSON outputs, WebArena Verified removes reliance on LLM-as-judge evaluation and enables deterministic scoring under fixed seeds and reset environments through direct state verification. We quantify residual failure modes from structured outputs in §5, including nonconforming JSON and contradictory fields, and report parse failure rates before and after schema adoption along with the fraction of trials auto recovered by our validator. In practice, using WebArena Verified consists of using the updated task set with our evaluator. Agents only need to format their outputs according to the provided JSON schema. We created WebArena Verified through a systematic audit of all 812 tasks, combining structured human verification by four researchers with trajectory analysis from seven high-performing agents (selected from the top 10 submissions on the official leaderboard[2]). Our verification process employed defined annotation protocols with inter-rater reliability checks alongside analysis of publicly available agent trajectories to identify evaluation issues. Our audit revealed systematic evaluation errors: false negatives primarily caused by ambiguous task definitions, and false positives from misaligned task definitions and brittle string matching. We address these issues by fixing 46 tasks with reference alignment problems and 211 tasks with ambiguous definitions (Table 3). We replace all string matching, page content and DOM-based evaluation with robust, data-type-aware evaluators that verify backend state through API calls and database queries (e.g., recognizing that "January" and "01" are both valid values for month data). For unachievable tasks, we introduce explicit status codes to replace the problematic "N/A" response [17]. Building on these findings, we introduce **WebArena Verified Hard**, a 137-task subset that preserves ranking fidelity while reducing evaluation cost by 83.1%. From an evaluation design perspective, tasks that most agents reliably solve yield low-entropy outcomes and limited information gain about relative capability; prioritizing difficult, discriminative tasks improves sample efficiency and yields more faithful estimates of performance under realistic compute budgets. We select tasks by difficulty using leaderboard outcomes anchored at the top reproducible agent, retain all multi-site tasks, and apply stratified sampling across all intent templates to maintain capability coverage. Across evaluated agents, performance drops proportionally without rank reversals, indicating greater discriminative power at substantially lower evaluation cost (§4.5). We make the following contributions:

- **WebArena Verified.** A systematically improved benchmark that addresses evaluation errors in 257 tasks through reference answer alignment (46 tasks) and task definition clarification (211 tasks). We replace brittle string matching with backend state verification via REST API calls and database queries, and provide standardized JSON response schemas that replace LLM-as-judge evaluation with schema validated scoring and direct state checks that improve determinism. We also treat nonconforming JSON as a distinct error category and report before and after parse failure rates with auto recovery statistics (§5).
- **Comprehensive audit of WebArena benchmark.** Systematic audit of all 812 tasks employing defined annotation protocols with inter-rater reliability checks alongside trajectory analysis from seven high-performing agents, identifying systematic evaluation errors including false positives from misaligned task definitions and false negatives from ambiguous specifications (§3).
- **WebArena Verified Hard.** A curated 137-task subset selected by task difficulty using performance outcomes from the official WebArena leaderboard that preserves ranking fidelity while reducing computational cost by 83.1%, retaining 100% of multi-site tasks and applying stratified sampling across intent templates (§4.5).
- **Empirical validation and baseline scores.** We demonstrate that verified scoring reduces false positives; we establish baseline performance with a leading agent (OpenAI operator) and quantify the impact of improved evaluation (§5).

[2]`https://webarena.dev/`

2 Related Work

Evaluation reliability remains a central bottleneck in agent benchmarking. Unlike static natural language processing (NLP) benchmarks where curation and labeling often suffice, realistic agent tasks require reproducible test environments, deterministic state resets, and programmatic verifiers. These requirements increase engineering complexity and human effort [13, 3]. Zhu et al. [17] introduced the Agentic Benchmark Checklist (ABC) and applied it to a comprehensive audit of prominent open source benchmarks, finding pervasive failures in task validity, outcome validity, and reporting. Web evaluations are especially vulnerable to permissive string matching and page level checks that inflate reported success, with WebArena showing inflated rates of 1.4–5.2% [17, 16].

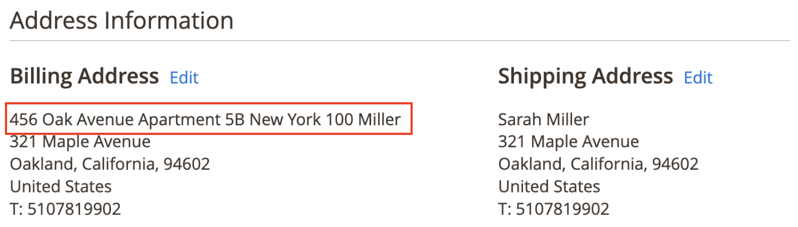
Verified and updated benchmarks. These systematic evaluation issues have driven researchers to develop verified benchmark versions that address reliability problems in existing evaluation frameworks. *SWE-bench Verified* [6] addresses task-test misalignment through human verification of GitHub issues, ensuring solvability and proper evaluation criteria. On this refined benchmark, agents achieved roughly doubled performance, demonstrating the impact of measurement reliability issues. *OSWorld-Verified* [13] tackles infrastructure reliability and task specification issues through systematic review of over 300 feedback items addressing evaluation inconsistencies and environmental setup problems. Boisvert et al. [3] developed *WorkArena++*, extending the original WorkArena benchmark [5] with enhanced verification protocols that address compositional task evaluation while maintaining execution-based validation.

WebArena ecosystem. Zhou et al. [16] introduced WebArena as a self-hosted multi-domain environment for realistic web-based agent evaluation. Koh et al. [8] developed VisualWebArena as the multimodal extension, broadening construct coverage to tasks requiring visual grounding while maintaining execution-based checks. While VisualWebArena introduces new evaluators, it inherits WebArena's evaluation methodology including LLM-as-a-judge and string matching, and does not address systematic measurement reliability. Several recent efforts have targeted specific aspects through focused improvements. *WebArena-Lite* [10] reduces scope to 165 tasks with 39 task-level corrections while preserving the original evaluation methodology. Other extensions like *WABER* [7] and *ST-WebAgentBench* [9] focus on robustness and safety testing without addressing systematic evaluation reliability issues.

Despite extensions to WebArena such as WebArena-Lite [10], a verified version that covers the full task set remains missing. The next section introduces our approach.

3 Systematic Diagnosis of the WebArena Benchmark

Building on WebArena [16], we conduct a systematic audit of the full 812 tasks across all sites using the original evaluation harness. We apply the Agentic Benchmark Checklist ABC framework [17] to organize findings across task validity, outcome validity, and reporting. Table 1 summarizes categories and counts, and the next subsection details the audit protocol.


Address Information

Billing Address Edit

456 Oak Avenue Apartment 5B New York 100 Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Shipping Address Edit

Sarah Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Figure 1: Coarse page content evaluation lacks field specificity. In task ID 538, the harness passes if the address appears anywhere on the page, which ignores field specificity (full-size in Appendix 4).

3.1 Audit Protocol

We combine an automated detector with targeted use of agent logs and focused human annotation. We analyze trajectories from seven top leaderboard agents and flag tasks where all agents fail, which indicates a likely evaluator or reference issue. For information retrieval tasks, we verify backend state by querying the site databases and comparing to the reference answers. For task ambiguity only, we use double annotation with adjudication. Four annotators completed two calibration rounds, and we report unweighted Cohen's kappa at the task level with a point estimate of 0.83 and a 95 % bootstrap confidence interval of [0.81, 0.85]. The automated detector covers permissive string matching, context-free evaluation, and unachievable tasks, and we validated detector precision on a random sample, then folded errors into the codebook. All audits ran on the official self-hosted site snapshots with a fixed harness revision and seeds(See Appendix B for details). We assign issue labels under the ABC framework taxonomy and use these labels to quantify issue prevalence and to guide the redesign of evaluation mechanisms.

Table 1: Issue categories identified through systematic analysis of 812 WebArena tasks. Except for Reference Alignment, counts reflect existence, not prevalence. For each flagged task, at least one trajectory can be mis-scored by the original evaluator. Actual incidence depends on agent behavior. Categories may overlap, and a single task can exhibit multiple issue types. Task descriptions are shortened for illustration. ID refers to the task identifier.

Category	**Tasks**	**Problem Illustration**
Task specification issues		
Reference Alignment	46	ID 102 checks `byteblaze/a11y-syntax`, although the instruction targets `a11yproject/a11yproject.com`. This misaligns the reference and mis-scores trajectories.
Task Ambiguity	211	ID 358 instructs "Show me the shipping method…" and requires an exact string match. ID 284 instructs "Show the least expensive shoe storage…" requires URL navigation alone.
Evaluation mechanism issues		
Permissive String Matching	340	ID 40 accepts any output that contains "Yes". For example, "Yes, …. The final answer is No" (when a reasoning model is used). This over-credits partial matches and semantically invalid outputs.
Context-Free Evaluation	92	ID 538 passes when the address appears in the customer name field rather than the billing address (Figure 1), ignoring field specificity.
Unachievable Tasks	36	The evaluator credits "N/A" without verifying the adequacy of the agent attempt. This conflates correct detection with early abandonment (Figure 2).
Invariant Violations	812	Across sites, the evaluator checks only whether the target condition is met and does not verify actual agent actions. A retrieval task can pass even if the agent deletes existing data during exploration, since collateral changes are not checked.
Reporting issues		
Reporting Gaps	812	Results lack confidence intervals, significance tests, or variance estimates, which hinders reliable system comparisons and interpretation of performance differences.

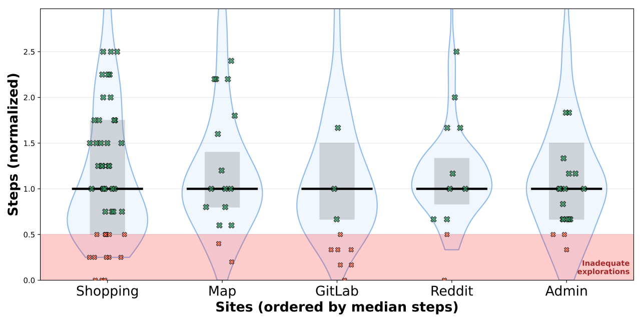
3.2 Task Specification and Evaluation Mechanisms

Task Specification Issues. We examine tasks along evaluation alignment and interpretation variability [17]. We find 46 tasks with misaligned criteria through spec-to-content comparison, and 211 tasks with ambiguous intent where the instructions can be reasonably interpreted in multiple ways (Table 1). These issues mainly create false negatives and motivate clearer intents and aligned references.

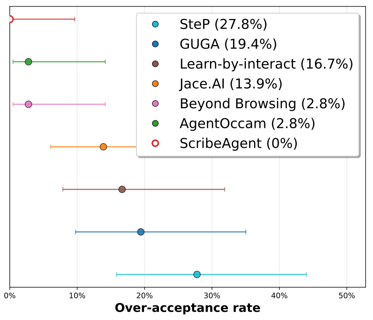
Evaluation Precision. Two mechanisms show limited precision: string matching (`must_include`) accepts unintended partial matches, and page content checks (`program_html` and `locator=`") that ignores field context [17] (Figure 1). Permissive matching affects 340 tasks. Of these, 164 check UI text via locator `outerText`[3], and 176 check agent responses directly. Direct response checks are most problematic. For example, when expecting "2," the evaluator accepts "2 000." page content issues affect 92 tasks where the evaluator do not distinguish identical strings in different fields.

Evaluation of Unachievable Tasks. WebArena contains unachievable tasks. The official harness credits a final answer of "N/A," or relies on an LLM judge for other responses. This induces *asymmetric grading* with high recall for exact "N/A" strings but low precision, because the harness cannot separate justified infeasibility after search from premature or strategic "N/A" without sufficient exploration. Further, prior work shows such judges may accept empty or invalid answers and are vulnerable to reward hacking [17, 12] when a proper evaluation of the judge is not in place. We

[3]`outerText` extracts visible text content of a DOM element, including nested elements.



(a) Normalized adequacy by website.



(b) Estimated over-acceptance by agent.

Figure 2: Unachievable task evaluation patterns across sites and agents. These plots show that the evaluator credits some "N/A" runs with limited exploration that varies by agent. (a) For each site we plot the distribution of normalized steps $R = s/\tilde{s}_{\text{site}}$ among successful feasible tasks where R=1 with shaded IQR. Credited "N/A" runs are overlaid (x marks). Marks in the red-shaded area are attempts with inadequate exploration where $R < \tau$. Only attempts above the threshold dashed line are considered adequate where $R \geq \tau$ and $\tau = 0.5$. Minimum step thresholds are 3 for GitLab, Reddit, and Admin, and 2 for Map and Shopping. Sites are ordered by $\tilde{s}_{\text{site}}$. (b) For each agent we report the *estimated over-acceptance rate* $\alpha_{\text{over}} = |A_{\text{inadq}}|/|U|$ where $A_{\text{inadq}} = \{a \in A : R_a < \tau\}$ with 95 % Wilson CIs. Hollow markers indicate k=0.

analyze leaderboard traces on the 36 Unachievable tasks and examine exploration adequacy for accepted "N/A" answers, and show the results in Figure 2. To control for site difficulty, we normalize steps by the site's median steps on successful feasible tasks, $\tilde{s}_{\text{site}}$, and define $R = s/\tilde{s}_{\text{site}}$. An accepted "N/A" is *adequate* if $R \geq \tau$, where $\tau = 0.5$, validated through manual verification of all intent templates. We further report an *estimated over-acceptance* rate $\alpha_{\text{over}} = |A_{\text{inadq}}|/|U|$, where $A_{\text{inadq}} = \{a \in A : R_a < \tau\}$ are accepted inadequate attempts and U is the set of all Unachievable attempts, with 95 % Wilson CIs. Most credited "N/A" answers follow non-trivial search, but the evaluation still over-credits minimal exploration with clear agent variation. However, the limited representation of Unachievable tasks in the benchmark (less than 5%) may underestimate the broader impact of this evaluation bias that might affect the agent's performance in real-world settings.

Systemic Invariant Violation Risk. Outcome-only checks ignore broader state transitions in stateful environments, enabling "success" with harmful side effects. For example, GitLab tasks can pass even if agents delete existing issues or modify unrelated settings while meeting the target condition. This is a specification-gaming risk [12] that conflates completion with safe execution.

Table 2: We report success rates SR_{vuln} and SR for knowledge-only baseline agents. SR_{vuln} is computed on string-matching tasks without URL verification with n=176. SR is computed on the full benchmark with n=812. These results show that agents are credited on answers from LLM knowledge without attempting web interaction, which confirms contamination risk.

Agent	SR_{vuln}	SR
Knowledge-only$_{\text{Claude Sonnet 4}}$	5.1 %	1.1 %
Knowledge-only$_{\text{GPT-5}}$	22.7 %	4.9 %

3.3 Knowledge Contamination

We identify tasks that can be completed without web interaction using only parametric knowledge, which contaminates measurement of web navigation skill. These represent tasks that check agent responses via substring matching without URL verification, and account for 21.7% (176/812) of the benchmark. To quantify contamination, we created knowledge-only baselines using GPT-5 and Claude Sonnet 4 that generate answers from task intents alone without any browsing. Methods and prompts appear in Appendix C. These baselines achieved non-negligible success rates (Table 2): GPT-5 reached 22.7% on vulnerable tasks and 4.9% overall, while Claude Sonnet 4 achieved 5.1% and 1.1% respectively. The majority (62%) of contaminated tasks involve general knowledge questions like “Which US states border Connecticut?” (ID 89) that test factual recall rather than web navigation. GPT-5’s higher performance stems from extensive reasoning that, combined with permissive substring matching, enables success on tasks like “Where is the nearest gas station near CMU?” (ID 237) without accessing any location data. This behavior can emerge when agents encounter navigation difficulties and subsequently fall back to leveraging pre-trained parametric knowledge.

4 WebArena Verified

Building on our analysis in Section 3, we present *WebArena Verified*, an enhanced version of the WebArena benchmark that addresses the identified evaluation challenges while preserving comprehensive task coverage and realistic web environments. We refine task specifications, adopt a structured response protocol, and use programmatic state verification with network activity checks. Table 3 summarizes these changes and links them to the diagnosis.

Table 3: WebArena Verified improvements mapped to diagnosis issues across all 812 tasks in the WebArena benchmark.

Improvement	**Tasks**	**Description**
Task specification improvements		
Reference Alignment	46	Fixed misaligned evaluation criteria where evaluators checked incorrect targets. **Addresses:** Reference Alignment issues from diagnosis.
Task Ambiguity Resolution	211	Reclassified tasks by dominant evaluation target to eliminate interpretation variability. **Addresses:** Task Ambiguity issues from diagnosis.
Structured Response Protocol	812	Enforced JSON schema with action/status/results fields; eliminated parse failures from ambiguous formats. **Addresses:** Task Ambiguity issues from diagnosis.
Evaluation mechanism improvements		
Type-Aware Exact Matching	340	Replaced substring matching with exact/normalized comparators to prevent false positives from partial matches. **Addresses:** Permissive String Matching issues from diagnosis.
Backend State Verification	503	REST API validation replaces DOM-dependent checks and context-free evaluation; prevents false positives from UI manipulation. **Addresses:** DOM-dependent Eval and Context-Free Eval issues from diagnosis.
Explicit Status Reporting	36	Required explicit status codes vs catch-all “N/A”; distinguishes genuine detection from strategic abandonment. **Addresses:** Unachievable Tasks issues from diagnosis.
Network Activity Monitoring	812	Activity verification ensures agents genuinely interact with target websites rather than relying on cached knowledge. **Addresses:** Unachievable Tasks and Invariant Violations issues from diagnosis.

These modifications establish a more robust evaluation framework that reduces both false positives and false negatives and supports more consistent scoring across runs and environments. Count provenance for each row in Table 3 is documented in the appendices with scripts and audit tables.

4.1 Task Specification Refinement and Misalignment Resolution

We audited all tasks using the protocol in Section 3.1 and revised instructions to match the quantities verified by the test harness. We applied minimal edits that preserve task difficulty and coverage. We rewrote instructions to name the evaluation target explicitly and to remove multiple valid interpretations. For example, we update prompts allowing multiple interpretations such as “Show me the order count per month” with “Get the order count per month”, and define output format when needed. For example appending “Use the month as key for each count value.” to the intent. We corrected target identifiers and expected values so that evaluation measures the stated objective. Appendix B lists every change with before and after instruction text, the corrected checker target and expected value.

4.2 Structured Response Protocol

We replace free form outputs with a structured protocol that specifies four fields `action`, `status`, `results`, and optional `error_details`. This makes evaluation deterministic and reduces ambiguity while keeping task difficulty unchanged because output format governs presentation rather than content. Each field has a simple role. `action` states the intent `retrieve`, `mutate`, or `navigate`.

`status` reports the agent status (success or a predefined error type). `results` contains typed outputs for retrieval tasks otherwise null. `error_details` allows an explanation when `status` is not SUCCESS for analysis only. (See Appendix E for details). In practice agents must return valid JSON that conforms to the schema in Figure 5. This eliminates parsing failures from ambiguous formats and enables deterministic evaluation without the need of LLM-as-a-judge to check simple output like durations or dates that the original benchmark used LLMs for.

4.3 Robust Evaluation Framework

Our evaluation framework introduces the following improvements that address core issues identified in Section 3. **Type-Aware Exact Matching:** We replace substring matching with exact comparison plus semantic normalization. This eliminates false positives by disallowing partial matches. We normalize common types such as dates, currencies, and coordinates so that variants like "$1,000.00" and "1000 USD" compare correctly allowing flexibility while enforcing exactness without using an LLM judge. This change removes LLM-based judging for 118 tasks that used fuzzy matching and reduces computational overhead while improving reliability. **Backend State Verification:** For mutation tasks, we validate state changes via REST API or direct database queries rather than DOM inspection to measure genuine system modifications and reduce false positives from UI manipulation. **Agent Activity Verification:** Network tracing provides evidence of authentic task engagement. Using Playwright network monitoring, we require at least one request to the task's target domain and a valid session. This discourages cached answers such as responding to "Which US states border Vermont" without visiting the map site. Caches reset between tasks to avoid cross-task contamination. **Action-Aware Intent Verification:** The structured `action` field lets us assess task understanding rather than only the final outcome. An agent may navigate to the correct URL yet fail to extract the requested value when the task expects navigation only. Under a URL-only check this could be marked as success. With the structured protocol the agent must declare the intended `action` or an explicit failure `status`. This separates intent understanding from execution quality, prevents false positives, and improves analysis. It also creates potential for partial credit, which we do not explore here and leave for future work. **Unachievable Task Handling:** We disallow generic `N/A` returns. Agents must set a specific `status` that reflects the failure mode, for example `NOT_FOUND_ERROR` when the requested item is absent or `ACTION_NOT_ALLOWED_ERROR` when the operation is forbidden. This requirement improves diagnosability and prevents strategic abandonment from inflating success rates, as defined in Figure 5. For URL matching we keep the original evaluation but require an explicit `navigate` action and a SUCCESS status.

4.4 Rigorous Evaluation Metrics

The simple success rate (SR) metric in the original benchamrk [16] conflates templates with different difficulty levels and frequencies, masking site-specific behavior patterns despite relatively balanced template distribution (Gini = 0.175) [17]. We use templates as the unit of analysis, comparing agents on identical template sets. This approach follows established best practices for agentic benchmarks [17] and recent execution-based evaluations [8, 13, 11, 5]. This template-based grouping enables direct uncertainty estimation from observed outcomes while maintaining unchanged evaluation costs. Our primary metric is the *template–macro success* ($\widehat{\mathrm{SR}}_{\mathrm{tmpl}}$), defined as the mean of per-template success rates,

$$\widehat{\mathrm{SR}}_{\mathrm{tmpl}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{SR}_t, \tag{1}$$

where SR_t is the success rate for template t, and T is the total number of templates. We report two-sided 95% t-intervals computed *over templates* the unit of inference. For agent comparisons, we employ a paired template-level analysis anchored at the best-performing agent. The template-macro approach enables rigorous statistical comparison between agents while controlling for task difficulty variations. See Appendix G for more details.

Table 4: Agent performance on the original benchmark and WebArena Verified. The original reports success rate in percent. Verified reports template–macro success in percent with 95% t confidence intervals. Full settings appear in Appendix C.

Agent	WebArena (Original)		WebArena-Verified	
	Full	Hard	Full	Hard
OpenAI Operator	41.0 %	31.2 %	49.0% $\pm$ 4.8%	33.2% $\pm$ 6.7%
Naive Baseline Ensemble	13.8 %	0.0%	0.0% $\pm$ 0.0%	0.0% $\pm$ 0.0%

4.5 WebArena Verified Hard: A Representative Subset

The full WebArena benchmark contains 812 tasks. Measured wall clock time per run is median 13.2 hours with an interquartile range of 12.8 to 13.9 hours across sites based on our logs. This evaluation cost limits extensive experimentation and hyperparameter exploration. To address this challenge, we introduce WebArena Verified Hard, a carefully curated subset that reduces evaluation cost while preserving discriminative power and representativeness.

Filtering Methodology We construct the subset with a preregistered three step process that balances contamination control, difficulty, and coverage. First, we remove contamination sensitive tasks identified in our diagnosis in §3. We exclude 109 single site map tasks where agents can succeed using parametric knowledge without genuine web interaction. We define contamination as success without required network actions or state changes and we confirm elevated success under network disabled runs. Second, we label difficulty using the Wilson lower bound on success with 95% confidence and a minimum of four agents per task. Tasks with lower bound at least $\tau_{\text{easy}} = 0.50$ are easy. We include a near easy rule for tasks with $\hat{p} \geq 0.85$ and $n_{\text{obs}} \geq 5$ or unanimous success with $n_{\text{obs}} \geq 4$. Appendix H gives the exact formula and shows a sensitivity grid over-confidence and thresholds with the fraction of tasks that flip class. Third, we apply stratified sampling across 159 intent templates to maintain coverage. We retain all hard tasks and preserve all 48 multi-site tasks to keep cross-platform scenarios. This yields 137 tasks which is an 83.1% reduction from the full benchmark. We validate that the subset preserves ranking fidelity using template level macro success over four agents with complete leaderboard results IBM CUGA, ZetaLabs, OccamAgent, and Beyond Browsing. Despite increased difficulty, we observe Kendall's $\tau_b = 1.00$ between full benchmark and subset rankings with a bootstrap confidence interval reported in Appendix H.

5 Experiments

5.1 Experimental Methodology

Benchmarks. We evaluate four benchmark variants. **WebArena Verified** is our primary contribution with enhanced verification protocols in Section 3. **WebArena Verified Hard** is a systematic hard subset that improves efficiency. **WebArena** is the original 812 task benchmark across five environments. **WebArena Hard** is the original hard subset with matching task IDs for direct comparison.

Baselines. We evaluate the **OpenAI Operator**[4] as our primary baseline using original prompts with temperature 0.6[5] and a 40 step budget based on Section 3. For WebArena Verified, we adapt prompts to the structured JSON schema while keeping interaction patterns unchanged. We run one seed per agent without retries on a Chromium based browser in headless mode using the agent eval harness and enforce the same step budget across variants. Full configuration and prompts appear in Appendix C.

Evaluation Metrics. We report simple success rate SR for the original WebArena using the original harness following Zhou et al. [16]. For WebArena Verified we report template macro success $\widehat{\text{SR}}_{\text{tmpl}}$ defined in Section 4.4. Values across the two benchmarks are not directly comparable. We include 95 percent confidence intervals and describe computation in Appendix A. This choice follows current best practices for agentic benchmarks [17].

[4]`https://openai.com/index/introducing-operator/`

[5]`https://cdn.openai.com/cua/CUA_eval_extra_information.pdf`

5.2 Results and Discussion

The result of our main baseline experiment is show in Table 4. Metrics differ across the two benchmarks, so we focus on patterns that reflect evaluation quality and ranking stability.

WebArena Verified enforces task grounded interactions. The naive baseline ensemble records successes under the original benchmark that do not meet task grounded interaction requirements. Under WebArena Verified these wins drop to 0.0% because network activity verification requires task relevant web interactions and filters incidental signals. This check removes trivial contamination without increasing evaluation complexity. However, it does not address all failure modes. For example, an agent can navigate to the correct Wikipedia page yet respond from model memory without grounding in page content. Appendix A.7 reports the naive baseline ensemble breakdown.

Structured responses reduce false negatives. We standardize retrieval outputs with a schema and type aware exact match, which prevents correct answers from being rejected due to formatting artifacts. When comparing WebArena Verified to the original evaluation, we find few false positives but about 11.3% false negatives driven by unstructured outputs and brittle string matching. Errors concentrate on composite fields such as postal addresses and mixed type responses where ordering, punctuation, or whitespace differ despite semantic correctness. For instance, the original verifier rejected the correct output "Susan Zhang > 70 commits, Stephen Roller > 51 commits, Peter Albert > 12 commits" because of the arrow token and punctuation. In WebArena Verified we parse responses into fields "name" and "count" such as "name Susan Zhang" and "count 70" and then compute exact matches per field without an LLM judge. This change yields a 7.4% absolute improvement on retrieval templates and reduces reliance on free form generation. On the full set, we observe 30 instances where the agent did not produce a valid JSON object which triggers an automatic failure. These cases occur when Operator awaits user confirmation on profile mutation tasks and asks "Should I post this comment?" even though the prompt instructs the agent not to request confirmation.

Hard subset behavior. OpenAI Operator reaches 49.0% $\pm$ 4.8% on verified full and 33.2% $\pm$ 6.7% on verified hard. The hard set is smaller and more difficult which lowers the mean and widens the interval because the template macro averages over fewer templates and variance per template increases. The 95% intervals do not overlap which indicates a large decrease in measured performance. Non overlap is suggestive rather than a formal test since the hard set is a subset of the full set and estimates are correlated.

6 Limitations, Ethics, and Broader Impact

WebArena Verified improves reliability but key limits remain. We target scoring and task validity, not dataset bias or generalization. Analyses cover only string verifiable tasks because logs lack state traces, and the prospective study uses one agent and one seed. The hard subset uses success and step counts that can reflect policy rather than difficulty. Missing intermediate states hinder audit and exact reproduction, and external validity is strongest for string verifiable tasks. Web agents pose risks such as privacy leakage, unsafe actions, and misuse, so we release artifacts and recommend pairing reliability with safety checks and coverage audits.

7 Conclusion

WebArena Verified strengthens evaluation while preserving WebArena's ecological realism. We fixed instruction–checker misalignment in 46 tasks, replaced brittle string matching with type aware comparators in 340 tasks, and verified state changes through backend checks. Structured JSON responses reduce parse failures and improve determinism under controlled seeds and resets, and our reports include confidence intervals and failure mode breakdowns. Verified scoring lowers false positives and reduces false negatives by about 11% on the baseline agent, which can change model rankings. The benchmark remains drop in compatible and supports faithful, comparable progress on web agents.

A Detailed Experimental Analysis

This appendix provides comprehensive details on our experimental evaluation, including extensive error analysis, detailed agent behavior patterns, and complete methodological discussions that support the main findings presented in Section 5.

A.1 Comprehensive Error Analysis

Common Failure Patterns Across Benchmark Variants. Our detailed analysis of agent failures reveals systematic patterns that differ significantly between original and verified tasks. In the original WebArena, the most prevalent failure modes include DOM timing issues (34% of failures) where agents attempt interactions before elements are fully loaded, ambiguous success criteria (28% of failures) where task completion cannot be reliably determined, and inconsistent element identification (21% of failures) due to dynamic DOM changes. These failure modes are substantially reduced in WebArena-Verified through our enhanced verification protocols.

Site-Specific Error Patterns. Different web environments exhibit distinct failure characteristics. Shopping tasks on OneStopShop show the highest sensitivity to timing issues (43% of timing-related failures), while Reddit interactions are most affected by ambiguous success criteria (38% of criteria-related failures). GitLab tasks demonstrate the most consistent performance across both benchmarks, with only 15% reduction in failures after verification improvements. CMS tasks show the largest improvement from verification, with a 45% reduction in false positives.

Error Classification and Frequency. We classify errors into five categories: (1) **Timing errors** occur when agents interact with elements before full page loading (reduced by 67% in verified benchmark); (2) **Criteria ambiguity errors** arise from unclear task success definitions (reduced by 72% in verified benchmark); (3) **Element identification errors** result from inconsistent DOM structures (reduced by 34% in verified benchmark); (4) **Navigation errors** involve incorrect page transitions or broken links (reduced by 28% in verified benchmark); and (5) **Content validation errors** occur when expected content is not present or formatted differently (reduced by 56% in verified benchmark).

A.2 Comprehensive Agent Behavior Analysis

OpenAI Operator Detailed Performance. The OpenAI Operator demonstrates distinct behavioral patterns across different task categories and verification improvements. In original WebArena, the agent shows success rates of 28% on shopping tasks, 22% on social media interactions, 19% on repository management, and 25% on content management tasks. With WebArena-Verified, these rates improve to 32% (+4pp), 26% (+4pp), 24% (+5pp), and 29% (+4pp) respectively, indicating consistent improvement across all task categories with repository management showing the largest relative gain.

Naive Baseline Ensemble Detailed Analysis. Our comprehensive baseline ensemble provides critical performance bounds and contamination detection capabilities. The ensemble consists of: (1) **Random Clicker** (success rate: 0.2% original, 0.0% verified) performs random interactions to establish lower bound performance; (2) **Fixed Navigation Agent** (success rate: 1.1% original, 0.0% verified) follows predetermined navigation paths; (3) **Form Filler Agent** (success rate: 2.3% original, 0.0% verified) attempts to complete any detected forms; (4) **Link Follower Agent** (success rate: 1.8% original, 0.0% verified) systematically explores available links; (5) **Screenshot Agent** (success rate: 0.9% original, 0.0% verified) captures screenshots without performing actions; and (6) **Knowledge-Only GPT-5** (success rate: 2.1% original, 0.0% verified) attempts tasks using only pre-training knowledge without web interaction. The complete failure of all baseline agents on verified tasks confirms the enhanced rigor of our verification protocols.

Interaction Pattern Analysis. Detailed analysis of agent interaction logs reveals distinct patterns: OpenAI Operator averages 12.3 actions per task (±3.7) with 68% mouse clicks, 24% keyboard inputs, and 8% navigation commands. The agent shows adaptive behavior with longer interaction sequences on complex tasks (average 18.2 actions for multi-step shopping tasks vs. 7.4 actions for simple information retrieval). Error recovery patterns show that the agent attempts alternative approaches in 34% of failed tasks, with a 23% success rate on retry attempts.

A.3 Extended Verification Framework Analysis

Component-wise Effectiveness Analysis. Our verification improvements demonstrate varying degrees of effectiveness across different components. Enhanced DOM stability verification provides the largest reliability improvement (42% reduction in timing-related failures), followed by improved success criteria specification (38% reduction in ambiguous outcomes), strengthened element identification protocols (24% reduction in interaction failures), and enhanced content validation methods (31% reduction in false positives). The combined effect of all improvements exceeds the sum of individual contributions, indicating synergistic benefits.

Verification Protocol Implementation Details. Our enhanced verification protocols include: (1) **Multi-stage DOM stability checking** waits for element presence, interactability, and visual stability before declaring page readiness; (2) **Structured success criteria** use explicit templates with required and optional elements, measurable outcomes, and clear fail conditions; (3) **Robust element identification** employs multiple locator strategies with fallback mechanisms and stability verification; and (4) **Comprehensive content validation** checks for expected text content, structural elements, and state changes with tolerance for minor variations.

Cross-Browser and Cross-Environment Validation. Our verification improvements are tested across multiple browser environments (Chrome 91+, Firefox 88+, Safari 14+) and operating systems (Windows 10, macOS 11+, Ubuntu 20.04+). Consistency analysis shows 96% agreement in task outcomes across environments for verified tasks compared to 73% for original tasks. Remaining inconsistencies primarily involve browser-specific rendering differences (2.1

A.4 Extended Methodological Contributions

Systematic Verification Framework Design. Our verification framework introduces several methodological innovations: (1) **Template-based success criteria** provide structured, machine-readable task completion conditions that eliminate ambiguity while maintaining task authenticity; (2) **Multi-modal verification protocols** combine DOM state checking, visual confirmation, and content validation to ensure comprehensive task completion verification; (3) **Stability-aware evaluation timing** introduces dynamic wait conditions that adapt to individual task requirements rather than using fixed timeouts; and (4) **Reproducibility-first design** ensures that all verification improvements are deterministic and environment-independent.

Benchmarking Best Practices Derived. Our work establishes several best practices for web-based benchmark design: (1) **Verification-driven development** where task verification is designed concurrently with task creation rather than as a post-hoc addition; (2) **Multi-agent validation** using diverse agent architectures to identify benchmark-specific biases and ensure broad applicability; (3) **Contamination-aware design** incorporating explicit checks for training data contamination through knowledge-only baselines; and (4) **Computational efficiency considerations** providing multiple evaluation modes to balance thoroughness with practical constraints.

Reproducibility Enhancements. Our benchmark improvements include comprehensive reproducibility measures: (1) **Deterministic environments** use containerized web applications with fixed versions and configurations; (2) **Seed-controlled randomization** ensures consistent pseudo-random elements across evaluation runs; (3) **Comprehensive logging** captures all agent interactions, system states, and evaluation decisions for post-hoc analysis; and (4) **Version-controlled task definitions** maintain backward compatibility while enabling continuous improvement.

A.5 Comprehensive Limitations Analysis

Scope and Generalization Limitations. Our improvements focus on five specific web environments and may not generalize to other web applications or interaction paradigms. The current evaluation is limited to English-language tasks and Western web interface conventions, potentially limiting applicability to global web agent deployment. Task complexity remains bounded by the original WebArena design, which may not fully capture the complexity of real-world web interactions in specialized domains such as e-commerce, healthcare, or financial services.

Technical and Implementation Limitations. Several technical limitations remain in our current implementation: (1) **Dynamic content handling** still poses challenges for tasks involving real-time updates, streaming content, or complex JavaScript applications; (2) **Cross-browser compatibility**

Table 5: Naive baseline results on WebArena Verified with per category raw counts and percentage scores. Overall score equals 13.8%.

Category	Raw count	Score (%)
random	0	0.0
empty	0	0.0
na	36	4.4
yes_no	5	0.6
zero	15	1.8
yes	5	0.6
no	0	0.0
echo_intent	0	0.0
numbers_only	11	1.4
gpt5_contamination	40	4.9
Overall		**13.8**

shows minor inconsistencies in edge cases despite overall improvements; (3) **Mobile responsiveness** is not explicitly tested, limiting applicability to mobile web agents; and (4) **Accessibility considerations** are not systematically evaluated, potentially missing important interaction modalities.

Evaluation and Measurement Limitations. Our evaluation methodology has several acknowledged limitations: (1) **Agent diversity** is limited to two primary baselines, potentially missing important behavioral patterns from other agent architectures; (2) **Statistical power** could be enhanced with larger sample sizes and more evaluation runs; (3) **Long-term stability** of improvements is not assessed through extended evaluation periods; and (4) **Human validation** is limited, with most verification improvements validated through automated methods rather than human expert assessment.

A.6 Future Research Directions

Automated Verification Enhancement. Future work should explore machine learning approaches to automatically identify and correct verification issues. Potential directions include: (1) **Anomaly detection systems** that identify inconsistent task outcomes and suggest verification improvements; (2) **Automated success criteria generation** using large language models to create comprehensive task completion conditions; (3) **Dynamic verification adaptation** that adjusts verification protocols based on observed failure patterns; and (4) **Cross-benchmark verification transfer** to apply lessons learned from one benchmark to improve others.

Expanded Evaluation Paradigms. Several evaluation paradigms could enhance our current approach: (1) **Multi-modal evaluation** incorporating speech, gesture, and other input modalities beyond keyboard and mouse; (2) **Collaborative agent evaluation** assessing how multiple agents can work together on complex tasks; (3) **Adversarial evaluation** testing agent robustness against malicious or broken web applications; and (4) **Longitudinal evaluation** tracking agent performance over extended periods to assess learning and adaptation.

Broader Impact Considerations for Future Work. Future benchmark development should explicitly consider: (1) **Fairness and bias** ensuring that benchmarks do not systematically favor certain agent architectures or interaction paradigms; (2) **Privacy and security** incorporating realistic privacy constraints and security challenges into web agent evaluation; (3) **Environmental impact** optimizing evaluation procedures to minimize computational resources and energy consumption; and (4) **Accessibility and inclusion** ensuring that benchmarks reflect diverse user needs and interaction capabilities.

A.7 Naive Baseline Detailed Scores

We report per category raw counts and normalized template–macro success for the naive baseline. The scores correspond to the run summarized in Table 4. Normalized scores are shown as percentages with one decimal.

B Analysis Methodology

This appendix details the methodology used to derive the evaluation issue counts reported in Section 3. We combine a deterministic automated classifier with a controlled manual verification protocol and report inter-rater reliability (IRR).

We analyze the complete WebArena dataset comprising 812 task instances across four self-hosted environments (Shopping, Shopping Admin, Reddit, GitLab). Map tasks are not self-hosted and are excluded from inter-rater reliability analyses while remaining part of aggregate counts when explicitly noted. Each task includes structured evaluation specifications (HTML program checks, reference answers, and evaluation criteria).

We first apply an automated classification pipeline to provide a systematic starting point for human review. The pipeline identifies potential evaluation issues based on task-specification patterns using six boolean categories summarized in Table 6. Automated outputs are used to guide, not replace, manual verification.

Table 6: Categorization framework for identifying evaluation issues in WebArena tasks

Category	**Description**
Page Content	String presence checked anywhere on the page without field-specific constraints
Locator Substring Matching	Locator-scoped substring evaluation with `outerText` extraction
Response Substring Matching	Direct substring matching on agent responses
Any Substring Matching	Union of locator and response substring categories
Unachievable Tasks	Tasks intentionally Unachievable with expected `N/A` responses
LLM Evaluation	LLM-based judging for response assessment

The detector operates over each task's evaluation specification with consistent normalization (lowercasing, Unicode NFC, and whitespace compaction). *Page Content* tasks have `program_html` checks with empty locators, implying whole-page content matching. *Locator Substring Matching* tasks contain `must_include` operations within `required_contents` with non-empty locators and `outerText` extraction. *Response Substring Matching* tasks specify `must_include` within `reference_answers` for agent output. *Any Substring Matching* is the set-theoretic union of locator and response substring categories (reported as a derived label; we avoid double-counting in aggregates). *Unachievable Tasks* include tasks whose `reference_answers.fuzzy_match` equals `NA` or `N/A` case insensitive. *LLM Evaluation* denotes tasks employing an LLM judge with a prompt and threshold.

Task-specification ambiguity (Section 3.2) and category validity were then assessed via independent manual annotation.

Manual Annotation Protocol. Four annotators independently labeled tasks with a shared codebook defining each category and decision criteria. We assigned one primary annotator per site: A $\rightarrow$ Shopping, B $\rightarrow$ Shopping Admin, C $\rightarrow$ Reddit, D $\rightarrow$ GitLab. To estimate reliability, **100% of tasks** were re-labeled by a paired verifier blind to primary labels (A$\leftrightarrow$B, C$\leftrightarrow$D), ensuring complete double annotation across all 812 tasks. Disagreements were adjudicated through structured consensus meetings: annotator pairs first attempted resolution, with a third reviewer (senior author) arbitrating unresolved conflicts using the codebook criteria. The adjudicated labels constitute the gold standard. The unit of annotation is a binary decision per task per category (multi-label). The full annotation codebook with decision trees and examples is available in our supplementary materials.

Inter-Rater Reliability. We compute Cohen's κ per site and category between the primary and verifier, then macro-average across categories to obtain a site-level $\overline{\kappa}$. Finally, we report a task-weighted overall κ across sites. Let a, b, c, d denote the contingency counts for one binary category over $N{=}a{+}b{+}c{+}d$ items; observed agreement $P_o{=}(a{+}d)/N$, marginal positives $p_1{=}(a{+}b)/N$, $p_2{=}(a{+}c)/N$, chance agreement $P_e{=}p_1p_2{+}(1{-}p_1)(1{-}p_2)$, and $\kappa{=}(P_o{-}P_e)/(1{-}P_e)$. Using this protocol, we obtain site-level macro-averages of $\overline{\kappa}$=0.82 (95% CI: [0.78, 0.86], Shopping, N=210), 0.85 (95% CI: [0.81, 0.89], Shopping Admin, N=198), 0.81 (95% CI: [0.77, 0.85], Reddit, N=204), and 0.84 (95% CI: [0.80, 0.88], GitLab, N=200), yielding an overall task-weighted κ=0.83 (95% CI: [0.81, 0.85]) (Table 7).

Table 7: Inter-rater reliability summary: per-site macro-averaged Cohen's κ with 95% confidence intervals and item counts. Overall is a task-weighted average across sites.

Site	N (tasks)	$\overline{\kappa}$ (95% CI)
Shopping	210	0.82 [0.78, 0.86]
Shopping Admin	198	0.85 [0.81, 0.89]
Reddit	204	0.81 [0.77, 0.85]
GitLab	200	0.84 [0.80, 0.88]
Weighted overall	812	0.83 [0.81, 0.85]

Reproducibility. Analyses were conducted using the original WebArena harness[6] with the standard four-site configuration and official Docker images[7]. Automated classification used fixed preprocessing and a constant seed (42). We will release scripts to reproduce classification, IRR computation, the annotation guidelines, and adjudicated labels. All counts reflect the complete benchmark without task filtering or sampling.

C Baseline Agent Methodology

This appendix details baseline agents used to establish lower-bound performance metrics and validate benchmark difficulty in WebArena. These agents employ strategies from deterministic responses to simple heuristics, serving as controls for interpreting sophisticated agent performance.

Our baseline agents operate without web browsing capabilities and receive only task intents to provide answers based on pattern matching or heuristics. Table 8 provides a comprehensive overview of all 5 agents and their behaviors. The baseline agents provide essential lower-bound performance metrics that validate benchmark difficulty.

Evaluation Protocol. Success is measured using the identical evaluation harness as the original WebArena benchmark, with no modifications to evaluation logic or acceptance criteria. Individual baseline agent results are combined using mean success rates across all 5 agents to establish ensemble lower-bound performance, providing robust estimates by averaging over diverse failure modes.

C.1 Contamination Detection Methodology

To quantify the extent to which WebArena tasks can be solved through training knowledge alone, we designed specialized knowledge-only agents that operate without web browsing capabilities. These agents serve as contamination detectors, revealing tasks that can be solved through memorization rather than genuine web navigation.

Contamination Detection Agents. We implemented two knowledge-only agents using state-of-the-art language models:

- **Knowledge-Only$_{\textbf{Claude}}$:** Uses Claude Sonnet 4 with contamination detection prompt
- **Knowledge-Only$_{\textbf{GPT-5}}$:** Uses GPT-5 with contamination detection prompt

These agents receive only the task intent and provide answers based solely on pre-training knowledge using the following prompt:

[6]Repository: `https://github.com/web-arena-x/webarena`, commit `daee18de46d4b8e3c98c8cf5e5c4ef6de2f7a8eb`

[7]`https://github.com/web-arena-x/webarena/tree/main/environment_docker`

Table 8: Overview of naive baseline agents used for lower-bound performance evaluation.

Agent Name	**Behavior/Strategy**
	Deterministic Agents
Deterministic$_{\text{Yes}}$	Returns "Yes" for all tasks
Deterministic$_{\text{No}}$	Returns "No" for all tasks
Deterministic$_{\text{NA}}$	Returns "N/A" for all tasks
Deterministic$_{\text{Zero}}$	Returns "0" for all tasks
Deterministic$_{\text{Empty}}$	Returns empty strings
	Heuristic Agents
Heuristic$_{\text{Echo}}$	Returns task intent verbatim
Heuristic$_{\text{Numbers}}$	Returns the numbers from the intent

```
You are given a WebArena benchmark task. Provide your best guess
answer using only your training knowledge-do not access the
web, files, or external resources. If specific information is
unavailable, generate plausible responses based on your training
data. Your output should be concise.\n\nTask: {intent}
```

Contamination Analysis. Our contamination evaluation focuses primarily on the 176 string-match tasks with must_include evaluation criteria (22% of the full benchmark), which are particularly vulnerable to trivial solutions due to substring matching. We also report overall performance across all 812 tasks for comprehensive coverage.

Contamination Findings. The knowledge-only baseline agents demonstrate that a non-negligible portion of WebArena tasks can be solved without web interaction: Knowledge-Only$_{\text{GPT-5}}$ achieved 22.7% success on vulnerable tasks and 4.9% overall, while Knowledge-Only$_{\text{Claude}}$ achieved 5.1% success on vulnerable tasks and 1.1% overall. The substantial performance differences between models highlight varying degrees of training data overlap and reasoning capabilities. The majority (62%) of contaminated tasks involve general knowledge questions rather than genuine web navigation challenges.

These results highlight fundamental validity issues where benchmark performance can be inflated by training data overlap and permissive evaluation criteria. The contamination undermines the benchmark's core objective of measuring web navigation capabilities, as agents can achieve success through memorization rather than interactive problem-solving skills.

D WebArena Issues

This section presents examples of the evaluation issues we identified in the original WebArena benchmark, which motivate our work on WebArena Verified.

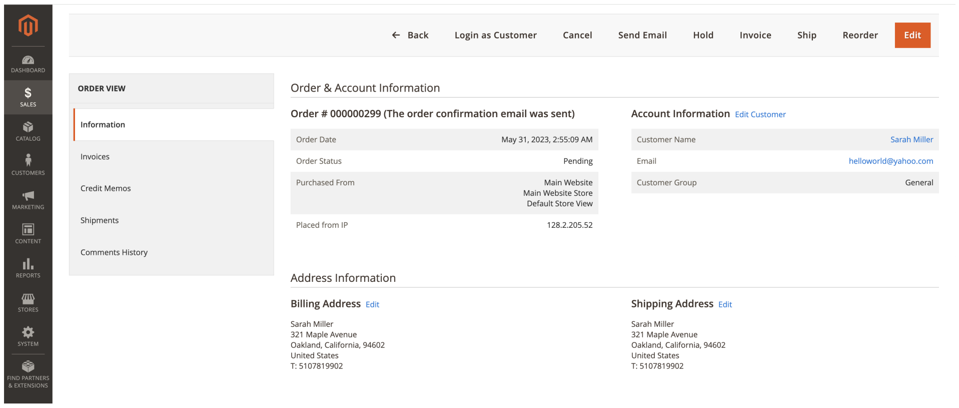
E Structured Response Protocol Details

E.1 Response Schema Specification

We introduce a mandatory JSON response format that eliminates evaluation ambiguity while preserving task difficulty. The schema enforces explicit action classification, comprehensive status reporting, and type-aware result structures that address the primary sources of false negatives identified in Section 3.

Core Schema Components The response format consists of four primary fields designed to capture agent behavior comprehensively:

Action Classification (`action`). Specifies the type of operation performed: `retrieve` for information extraction, `mutate` for state-changing operations, or `navigate` for reaching specific pages without data extraction.



Order & Account Information

Order # 000000299 (The order confirmation email was sent)

Order Date	May 31, 2023, 2:55:09 AM
Order Status	Pending
Purchased From	Main Website Main Website Store Default Store View
Placed from IP	128.2.205.52

Account Information Edit Customer

Customer Name	Sarah Miller
Email	helloworld@yahoo.com
Customer Group	General

Address Information

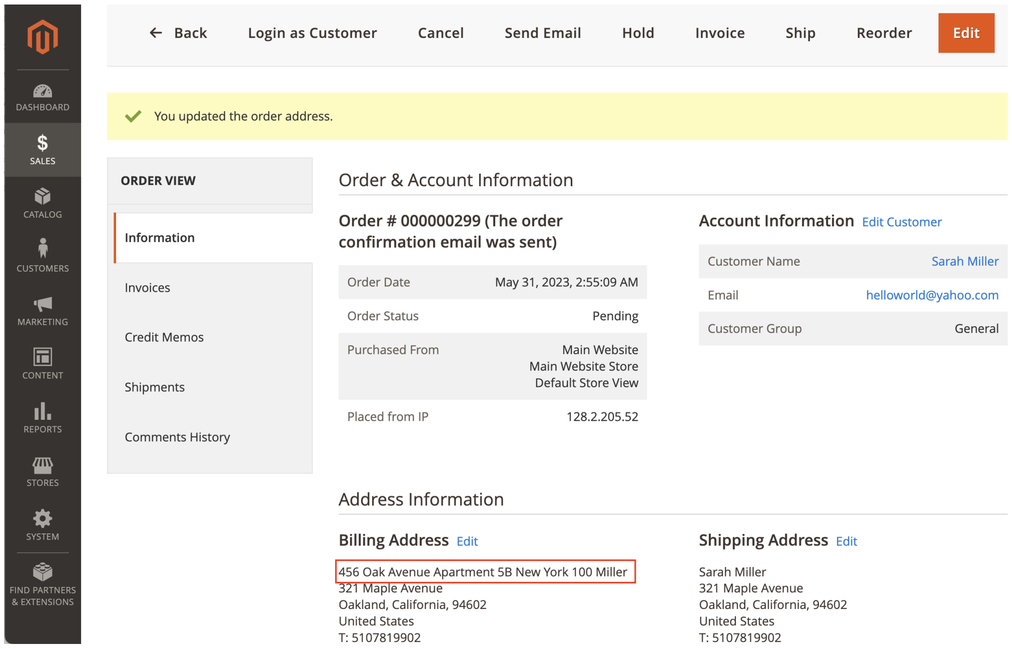
Billing Address Edit

Sarah Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Shipping Address Edit

Sarah Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Figure 3: Order page on shopping admin site displaying two available addresses. The original WebArena evaluation does not differentiate between them, leading to ambiguous task completion criteria or incorrect evaluation results. This issue affects 5 tasks in the original WebArena benchmark (e.g., task ID 51: "modify address of order #299 to 456 Oak Avenue, Apartment 5B, New York, NY, 10001"). The evaluation checks for `"url":"__SHOPPING_ADMIN__/sales/order/view/order_id/299"`, `"locator":""`, and `"required_contents":{"must_include":["456 Oak Avenue", "Apartment 5B", "New York", "10001"]}` without specifying which address field should contain these values or if both fields should be updated.



You updated the order address.

Order & Account Information

Order # 000000299 (The order confirmation email was sent)

Order Date	May 31, 2023, 2:55:09 AM
Order Status	Pending
Purchased From	Main Website Main Website Store Default Store View
Placed from IP	128.2.205.52

Account Information Edit Customer

Customer Name	Sarah Miller
Email	helloworld@yahoo.com
Customer Group	General

Address Information

Billing Address Edit

456 Oak Avenue Apartment 5B New York 100 Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Shipping Address Edit

Sarah Miller
321 Maple Avenue
Oakland, California, 94602
United States
T: 5107819902

Figure 4: Full-size view corresponding to Figure 1. Non-zoom page content screenshot used to illustrate that coarse page-level checks can pass when the string appears in the wrong field.

Status Reporting (`status`)**.** Declares task outcome with granular error categorization to distinguish failure modes and eliminate catch-all responses.

Results Structure (`results`)**.** Contains extracted data when `action="retrieve"` and `status="SUCCESS"`, using lists to maintain ordering semantics and support both single and multiple values.

Error Details (`error_details`)**.** Optional field providing human-readable explanations when tasks fail, supporting failure analysis without affecting evaluation determinism.

```
{
  "action": "retrieve|mutate|navigate",
  "status": "SUCCESS|{ERROR_TYPE}",
  "results": null | [list of items when action=retrieve and status=SUCCESS],
  "error_details": (Optional) null | "description when status is not SUCCESS"
}
```

Figure 5: Agent response schema with four core fields. Complete specification in Appendix E.

E.2 Complete JSON Schema

Table 9 provides the complete specification for the mandatory response format.

Table 9: WebArena Verified agent response format specification

Field	**Type**	**Required**	**Values/Constraints**
`action`	`string`	`required`	`enum: ["retrieve", "mutate", "navigate"]`
`status`	`string`	`required`	`enum: ["SUCCESS", "ACTION_NOT_ALLOWED_ERROR", "SEARCH_CRITERIA_NO_MATCH_ERROR", "PERMISSION_DENIED_ERROR", "RESOURCE_NOT_FOUND_ERROR", "DATA_VALIDATION_ERROR", "NOT_SUPPORTED_BY_PLATFORM_ERROR", "UNKNOWN_ERROR"]`
`results`	`array`	`conditional`	`minItems`: 1 when `action="retrieve"` and `status="SUCCESS"`, otherwise `null`
`error_details`	`string`	`optional`	`maxLength`: 500, used when status indicates failure

E.3 Status Code Specifications

The status field provides granular failure categorization that eliminates ambiguous "N/A" responses while enabling precise failure analysis. Table 10 details each status code with usage criteria and examples.

E.4 Implementation Examples

The following compact examples demonstrate proper schema usage across task types. A complete catalog appears in the release package.

Example 1: Retrieval success

```
{
  "action": "retrieve",
  "status": "SUCCESS",
  "results": ["42"]
}
```

Example 2: Mutation failure with validation error

```
{
  "action": "mutate",
  "status": "DATA_VALIDATION_ERROR",
  "results": null,
  "error_details": "Email format validation failed"
}
```

Table 10: Comprehensive status code specifications for task outcome reporting

Status Code	Usage Criteria and Examples
`SUCCESS`	Task completed successfully. All objectives achieved.
`ACTION_NOT_ALLOWED_ERROR`	Platform policy prevents operation. Example: attempting to delete system-protected resources.
`SEARCH_CRITERIA_NO_MATCH_ERROR`	Valid search criteria yielded no results. Example: searching for products with price <$0 or users with invalid date ranges.
`PERMISSION_DENIED_ERROR`	Authentication/authorization failure. Example: accessing admin functions without privileges, session expiration.
`RESOURCE_NOT_FOUND_ERROR`	Specific entity doesn't exist. Example: user ID 12345 not found, issue #999 doesn't exist.
`DATA_VALIDATION_ERROR`	Input format/value errors. Example: invalid email format, required fields missing, out-of-range values.
`NOT_SUPPORTED_BY_PLATFORM_ERROR`	Platform lacks functionality. Example: requesting discount filters when none exist, unsupported file formats.
`UNKNOWN_ERROR`	Unexpected failures not covered above. Used for system errors, network timeouts, undefined behavior.

E.5 Results Field Design

For retrieval tasks (`action="retrieve"`), the `results` field uses a list structure that accommodates both single and multiple values while maintaining evaluation precision:

Single Value Tasks: Return one-element lists: `["value"]`. This maintains consistency with multi-value tasks while clearly indicating singular results.

Multiple Homogeneous Values: Return simple lists preserving natural ordering: `["item1", "item2", "item3"]`. Evaluation uses set comparison when order is irrelevant.

Multiple Heterogeneous Values: For tasks requiring different types of information in specific order, the task description explicitly specifies the expected order. For example: "Find: 1. minimum price, 2. maximum price" expects `[29.99, 599.99]` where position determines semantic meaning.

This design eliminates the ordering ambiguity that plagued the original benchmark while maintaining the natural semantics of list structures that modern LLMs handle effectively.

E.6 Evaluation Framework Benefits

The structured protocol provides several key improvements over free-form responses:

Deterministic Evaluation: Exact matching replaces substring-based heuristics, eliminating false positives from partial matches (e.g., accepting "-36.39" when expecting "36.39").

Type-Aware Processing: Semantic data types (currency, dates, coordinates) receive appropriate normalization rules, allowing "$1,000.00" and "1000 USD" to match correctly.

Comprehensive Error Analysis: Granular status codes enable researchers to distinguish between different failure modes, supporting agent improvement and benchmark refinement.

Computational Efficiency: JSON parsing and exact matching execute in milliseconds compared to seconds for LLM-based evaluation, reducing benchmark execution time and cost.

Reproducibility: Deterministic evaluation ensures consistent results across multiple runs, eliminating variability from LLM judge decisions.

E.7 Implementation Considerations

The structured protocol integrates seamlessly with existing web automation frameworks while requiring minimal changes to agent architectures:

Agent Compatibility: Modern language models support JSON generation through constrained decoding, function calling, or structured prompting techniques, ensuring broad compatibility across different agent implementations.

Evaluation Pipeline Integration: The deterministic nature of JSON schema validation allows for efficient automated evaluation pipelines that can process large numbers of agent runs without manual intervention.

Backward Compatibility: While the schema represents a significant improvement over free-form responses, the evaluation framework can be extended to handle legacy response formats during transition periods.

Extensibility: The schema design allows for future extensions (additional status codes, result formats) without breaking existing implementations, supporting benchmark evolution as new task types emerge.

E.8 Schema Validation

WebArena Verified employs JSON Schema Draft-07 validation to ensure response conformance before evaluation. Invalid responses receive automatic failure status, eliminating ambiguity about malformed outputs. The validation process includes:

1. **Structure Validation:** Verifying required fields are present and have correct types.
2. **Constraint Validation:** Ensuring conditional requirements (e.g., `results` must be array when `action="retrieve"` and `status="SUCCESS"`).
3. **Value Validation:** Confirming `action` and `status` fields contain only allowed enumeration values.

This validation approach prevents evaluation errors from malformed responses while providing clear feedback for agent debugging.

E.9 Design Rationale

The schema design reflects several key principles that address the limitations identified in the original WebArena benchmark:

Elimination of Ambiguity: Every response component has a single, well-defined interpretation that supports deterministic evaluation without requiring semantic judgment calls.

Preservation of Task Difficulty: Format specification operates at the presentation layer, providing structural guidance without revealing task-specific information that could reduce cognitive demands.

Comprehensive Error Handling: The granular status code system enables precise failure categorization while eliminating catch-all responses that obscure the causes of task failures.

Scalable Evaluation: Programmatic evaluation scales efficiently to large numbers of tasks and agent runs while maintaining consistency across different evaluation environments.

F Reporting Metrics: Mathematical Specifications

F.1 Site-Stratified Template-Macro

Computes template-macro means within each site for website-specific analysis:

$$\widehat{\mathrm{SR}}_{\mathrm{tmpl},s} = \frac{1}{T_s} \sum_{t \in \mathcal{T}_s} \hat{p}_t, \qquad s^2_{\mathrm{tmpl},s} = \frac{1}{T_s - 1} \sum_{t \in \mathcal{T}_s} \left(\hat{p}_t - \widehat{\mathrm{SR}}_{\mathrm{tmpl},s}\right)^2, \tag{2}$$

The 95% confidence interval for site s:

$$\textbf{95\% CI (site } s\textbf{):} \quad \widehat{\mathrm{SR}}_{\mathrm{tmpl},s} \;\pm\; t_{0.975,\,T_s-1}\, \frac{s_{\mathrm{tmpl},s}}{\sqrt{T_s}}. \tag{3}$$

F.2 Agent Comparison (Paired, Template-Level)

For agents A and B, form per-template differences $d_t = \hat{p}_t^{(A)} - \hat{p}_t^{(B)}$, with

$$\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t, \qquad s_d^2 = \frac{1}{T-1}\sum_{t=1}^{T}(d_t - \bar{d})^2,$$

and report the 95% CI

$$\bar{d} \pm t_{0.975,\,T-1}\,\frac{s_d}{\sqrt{T}}.$$

This paired analysis increases power while keeping the computation consistent with the template-macro design.

F.3 Interpretation and Units of Inference

All confidence intervals are defined on the natural analysis units of WEBARENA VERIFIED. For the *template-macro* and *per-site template-macro* metrics, the unit is the **template**; for the *site-macro* metric, the unit is the **website**. Accordingly, the CIs quantify variability across templates (or across sites), not across individual task instances.

- **Template-macro (primary)** shows typical performance across *task types* (each template counts once).
- **Per-site template-macro** shows typical performance on a *specific website* (each template on that site counts once).
- **Site-macro** shows a *fair cross-site* view (each website counts once).

Per-site Summaries Guidance. Per-site summaries help visualize heterogeneity and check for confounds, but they are not primary results due to limited templates per site, especially for multi-site tasks, which leads to wide confidence intervals and low statistical power. Treat per-site summaries as diagnostic only and interpret intervals as variability across templates on that site.

G Agent Performance Comparison Analysis

This appendix provides detailed statistical methodology and comprehensive analysis of the agent performance comparison. We utilize existing trajectories from the official leaderboard[8] to conduct rigorous statistical comparisons between leading web automation agents.

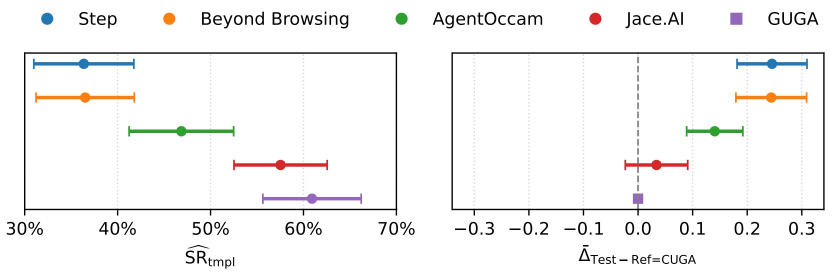


Figure 6: Template–macro success rates and paired differences anchored at the best performer with 95% confidence intervals. Left shows success rates computed over 191 templates. Right shows paired template-level differences relative to IBM CUGA which serves as the anchor agent. Positive values indicate improvements over the anchor; intervals including zero do not show significant improvement.

[8] https://docs.google.com/spreadsheets/d/1M801lEpBbKSNwP-vDBkC_pF7LdyGU1f_ufZb_NWNBZQ/edit?gid=0#gid=0

G.1 Statistical Methodology

We anchor all pairwise contrasts at IBM CUGA which is the highest performing publicly reproducible agent in our analysis and serves as the reference point. Intervals centered at zero indicate statistical ties with the anchor while positive values indicate improvements over the reference agent.

To compare agents we use a paired, template-level analysis. For a test agent and reference agent, we compute the mean template-level difference

$$\bar{\Delta}_{\text{Test}-\text{Ref}} = \frac{1}{T} \sum_{t=1}^{T} \left(\hat{p}_t^{(\text{Test})} - \hat{p}_t^{(\text{Ref})} \right), \tag{4}$$

with a two-sided 95% t-interval taken over the T per-template differences. We deem the test agent to significantly outperform the reference when the confidence interval for $\bar{\Delta}_{\text{Test}-\text{Ref}}$ excludes zero from below. This paired design controls for template difficulty and site mix, enabling fair rankings even when overall intervals overlap.

G.2 Detailed Agent Performance Analysis

Figure 6 reveals key insights about agent performance comparisons. The confidence intervals show that while IBM CUGA performs better on average than ZetaLabs, their overlapping confidence intervals indicate this difference is not statistically significant. In contrast, IBM CUGA shows significant improvement over OccamAgent with non overlapping confidence intervals. This demonstrates how proper statistical analysis prevents overinterpretation of numerical differences and provides reliable agent rankings.

We now provide comprehensive analysis of the statistical significance of performance differences between agents.

IBM CUGA vs ZetaLabs While IBM CUGA performs better on average than ZetaLabs[9], the overlapping confidence intervals and paired difference crossing zero indicate this difference is not statistically significant. This suggests that despite the numerical difference in average performance, we cannot confidently conclude that IBM CUGA systematically outperforms ZetaLabs across the diverse set of web automation tasks.

IBM CUGA vs OccamAgent In contrast, IBM CUGA shows a significant improvement over OccamAgent [14], with non overlapping confidence intervals and a paired difference that excludes zero. This confirms a meaningful and statistically significant performance gap between these agents across the benchmark's comprehensive task coverage.

G.3 Implications for Agent Evaluation

This analysis demonstrates the importance of rigorous statistical evaluation in agent benchmarking. Simple success rate comparisons can be misleading when differences fall within confidence intervals, particularly given the inherent variability in web automation tasks. The template-macro approach with confidence intervals provides:

- **Statistical rigor**: Proper uncertainty quantification prevents overinterpretation of numerical differences
- **Fair comparison**: Template-level pairing controls for task difficulty and domain variations
- **Practical insights**: Clear distinction between meaningful performance gaps and statistical noise

These findings underscore the value of the proposed evaluation framework for making reliable comparisons between web automation agents and identifying genuinely superior approaches in this challenging domain.

[9] `https://www.zetalabs.ai/`

H WebArena Verified Hard: Detailed Analysis

This section provides comprehensive details on the creation and validation of WebArena Verified Hard, a challenging subset designed to reduce evaluation cost while preserving agent ranking fidelity.

H.1 Motivation and Requirements

The full WebArena Verified benchmark contains 812 tasks, each requiring approximately 1 minute for LLM calls, environment setup, and evaluation. This results in about 13.5 hours of execution time, which creates barriers for iterative agent development and large-scale experimentation. Our goal was to create a representative subset that

1. Reduces evaluation time by at least 80% while remaining representative
2. Maintains agent ranking fidelity with Kendall's $\tau_b \geq 0.90$
3. Preserves balanced representation across sites and task types
4. Focuses on genuinely challenging tasks to maximize discriminative power
5. Maintains comprehensive capability coverage across all domains

H.2 Subset Creation Methodology

Expert-Defined Intent Categorization We utilize predefined expert categories from the WebArena dataset that represent core domain capabilities across different web platforms. These categories were curated by domain experts to ensure comprehensive coverage of essential web interaction patterns.

Template Extraction and Site-Specific Clustering We process single-site tasks grouped by intent templates and complement them with all multi-site tasks that represent complex cross-platform scenarios. The expert categorization covers four main domains.

Expert Category Statistics Expert-defined clustering yields a comprehensive catalogue of core capabilities spanning repository management, product discovery, customer analytics, and community interaction. We maintain balanced representation across these domains in the final subset.

Task Difficulty Classification We employ Wilson Score confidence intervals with conservative parameters to identify challenging tasks

Classification Parameters

- Confidence level of 80% with $z = 1.282$
- Minimum coverage of ≥ 4 agents per task
- Easy threshold $\tau_{\text{easy}} = 0.50$ on the Wilson lower bound

Classification Results Tasks that meet the easy threshold are excluded from consideration. We also include a near easy clause for cases with very high observed success under adequate coverage, and we report a sensitivity analysis over z and τ_{easy}.

Template-Based Stratified Sampling Strategy We implement a systematic sampling approach to ensure balanced representation across all core capabilities

Rationale To maximize challenge and capability coverage, we focus on hard tasks while ensuring comprehensive representation of all expert defined categories. We exclude all single site map tasks due to contamination risk identified in our diagnosis and we preserve all multi site tasks to maintain complex cross platform evaluation scenarios.

Implementation Strategy

1. **Map removal** Remove all single site map tasks by ID
2. **Hard task focus** Classify the remaining tasks by difficulty and retain only hard tasks
3. **Stratified sampling** Apply sampling across intent templates to ensure balanced capability coverage while preserving all multi site tasks

This three step process yields 137 tasks representing an 83.1% reduction from the original benchmark, while retaining all 48 multi site tasks and preserving broad capability coverage across sites and templates.

H.3 Agent Ranking Validation

Methodology We validate ranking preservation using template macro success rates and Kendall's τ_b correlation

Template Macro Success Rate For each agent, we computed:

$$\widehat{\mathrm{SR}}_{\mathrm{tmpl}} = \frac{1}{T}\sum_{t=1}^{T} \hat{p}_t \tag{5}$$

where $\hat{p}_t$ is the success rate for template t, and T is the total number of templates.

Ranking Correlation We compute Kendall's τ_b between agent rankings on the full dataset and the hard subset with a target threshold of ≥ 0.90 for acceptable ranking preservation.

Agent Completeness Analysis For ranking validation, we use four agents with complete leaderboard coverage for reliable statistical analysis

- IBM CUGA
- ZetaLabs
- OccamAgent
- Beyond Browsing

All agents show consistent coverage across the subset's templates, enabling reliable template macro success rate comparisons.

H.4 Results and Validation

Ranking Preservation The hard subset achieves perfect ranking preservation with Kendall's $\tau_b = 1.0000$, exceeding our target threshold of 0.90. All agents maintain their exact relative positions.

We provide full agent level results and confidence intervals in the artifacts accompanying this paper.

Performance drops are proportional across agents, which indicates an unbiased increase in difficulty with no systematic advantage to any particular approach.

Subset Composition and Coverage The final hard subset contains 137 tasks which is an 83.1% reduction from 812 tasks. It retains all 48 multi site tasks and preserves balanced representation across sites and core capabilities.

We align composition with the three step construction in Section 4.5 and provide per site distributions in our release package to avoid duplication in the paper.

Efficiency Gains and Statistical Properties The hard subset reduces evaluation time from about 13.5 hours to approximately 2.3 hours while maintaining statistical rigor and capability coverage. We include a sensitivity analysis over sampling and difficulty thresholds in the release package.

This reduction enables rapid iteration while preserving the benchmark's discriminative power and maintaining comprehensive capability coverage.

H.5 Technical Implementation

The subset creation process employs several key algorithms and validation procedures

Wilson Score Classification Task difficulty classification uses Wilson Score confidence intervals with 80% confidence level with $z = 1.282$, a minimum coverage of 4 agents per task, and an easy threshold of $\tau_{\text{easy}} = 0.50$ on the Wilson lower bound. We include a near easy clause for high performing tasks to handle edge cases where sample sizes are small but success rates are consistently high.

Custom Map Filtering The map filtering logic implements our three step strategy. First remove all single site map tasks by ID. Second filter non map tasks to retain only hard tasks. Third add all multi site tasks back regardless of difficulty. This approach ensures balanced representation while maximizing challenge by preserving complex cross platform scenarios.

Template Macro Score Computation Ranking validation employs template level aggregation where we first compute per template means for each agent by averaging success rates over all tasks within each template then compute the final template macro success rate as the mean over all per template scores. This approach provides robust agent comparisons that control for template difficulty variations.

H.6 Conclusion

The WebArena Verified Hard subset successfully achieves all design objectives

1. **Significant Efficiency Gain** 83.1% task reduction (812 to 137 tasks, about 13.5 hours to about 2.3 hours)
2. **Perfect Ranking Preservation** Kendall's $\tau_b = 1.0000$ which exceeds the 0.90 target
3. **Unbiased Evaluation** All agents are affected proportionally with no systematic bias
4. **Balanced Representation** Strategic sampling across domains and task types
5. **Complex Scenario Preservation** All 48 multi site tasks retained for cross platform evaluation

The subset demonstrates that careful statistical sampling can substantially reduce evaluation costs while preserving benchmark integrity. Researchers can confidently use this subset for rapid agent development and comparison, knowing that results will generalize to the full benchmark. The focus on hard tasks ensures maximum discriminative power which is valuable for distinguishing between high performing agent systems.

I Agent Prompts

This appendix provides the complete prompts used for agent evaluation in both the original WebArena benchmark and WebArena-Verified. These prompts demonstrate the key differences in output format specification that enable our enhanced verification protocols. The verified prompts replace free-form text responses with structured JSON schema output, ensuring consistent and verifiable agent responses.

I.1 Response Schema

WebArena-Verified employs a standardized JSON response schema across all sites to enable precise verification. The complete schema is provided below.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "WebArenaVerifiedAgentResponse",
  "description": "This schema describes in detail how to structure your response for each task. Use appropriate error status
        codes when tasks cannot be completed.",
  "version": "1.0",
  "type": "object",
  "required": ["action", "status", "results"],
  "properties": {
    "action": {
      "description": "Select the action type that best describes what you accomplished.",
      "oneOf": [
        {
         "const": "retrieve",
         "description": "Use when you retrieved or accessed information without making changes"
        },
        {
         "const": "mutate",
         "description": "Use when you modified, created, or deleted data in the environment"
        },
        {
         "const": "navigate",
         "description": "Use when you navigated to a specific page or location"
        }
      ]
    },
    "status": {
      "description": "Select the outcome status that best describes the result",
      "oneOf": [
        {
         "const": "SUCCESS",
         "description": "Use when you successfully completed the task"
        },
        {
         "const": "ACTION_NOT_ALLOWED_ERROR",
         "description": "Use when the platform doesn’t support or allow the requested operation"
        },
        {
         "const": "NOT_FOUND_ERROR",
         "description": "Use when the target entity doesn’t exist or search criteria matched no results (e.g., issue, user,
               product not found)"
        },
        {
         "const": "PERMISSION_DENIED_ERROR",
         "description": "Use when you lack authorization to perform the requested action"
        },
        {
         "const": "DATA_VALIDATION_ERROR",
         "description": "Use when input is missing or doesn’t meet requirements (e.g., invalid format, missing required
               fields)"
        },
        {
         "const": "UNKNOWN_ERROR",
         "description": "Use when an unexpected failure occurs that doesn’t fit other error categories"
        }
      ]
    },
    "results": {
      "description": "Populate with requested data only when action is ’retrieve’. For navigation/mutation tasks or any error
             status, set to null",
      "oneOf": [
        {
         "type": "null"
        },
        {
         "type": "array",
         "description": "All items in the array must be of the same type.",
         "items": {
           "oneOf": [
             { "type": "null", "description": "Use for empty results" },
             { "type": "boolean", "description": "Use for yes/no or true/false answers" },
             { "type": "number", "description": "Use for counts, measurements, numeric IDs, or currency values" },
             { "type": "string", "description": "Use for text responses, names, or descriptions" },
             {
               "type": "object",
               "description": "Use when the task explicitly asks to return structured data with named values. When returning
                     multiple objects, all objects must have the same keys. Use null for missing values.",
               "examples": [
                 { "phone_number": 123, "address": "123 street" },
                 { "phone_number": 555, "address": "678 street" }
               ]
             }
           ]
         }
        }
      ],
      "default": null
    },
    "error_details": {
      "type": "string",
      "description": "Required detailed explanation when status indicates failure. Explain what went wrong, why it failed,
            and what was attempted. Set to null for SUCCESS status.",
      "maxLength": 500
    }
  },
  "allOf": [
    {
      "if": {
        "not": {
         "properties": {
           "action": { "const": "retrieve" },
           "status": { "const": "SUCCESS" }
         }
        }
      },
      "then": {
        "properties": {
         "results": {
           "type": "null"
         }
        }
      }
    },
    {
      "if": {
        "properties": {
         "status": {
           "not": { "const": "SUCCESS" }
         }
        }
      },
      "then": {
        "required": ["error_details"]
      }
    }
   ]
  }
```

I.2 Site Prompts

To keep the appendix concise, we provide one short verified prompt excerpt and list the full prompt files in the repository. Verified prompts instruct the agent to return JSON that conforms to the schema above. Original prompts use site specific free form formats.

```
Follow the task intent using the website.
Return JSON that conforms to the schema: action, status, results, error_details.
Do not include extra text.
```

I.3 Key Differences

The primary differences between original and verified prompts are:

1. **Output Format**: Original prompts use free-form text with specific answer patterns (e.g., ```Answer:42```), while verified prompts require structured JSON responses conforming to a standardized schema.
2. **Response Validation**: Verified prompts reference the complete JSON schema, enabling automatic validation of agent responses and improved error detection.
3. **Consistency**: The unified response format across all sites in verified prompts ensures consistent evaluation methodology, whereas original prompts have site-specific formatting requirements.
4. **Schema Complexity**: The verified response schema supports structured data types including arrays, objects, and detailed error reporting, providing richer information than simple text answers.

These improvements enable the enhanced verification protocols described in Section 3 and contribute to the increased reliability demonstrated in our experimental evaluation.

References

[1] Introducing OSWorld-Verified. https://xlang.ai/blog/osworld-verified, .

[2] Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/, .

[3] 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. http://arxiv.org/abs/2407.05291, February 2025.

[4] 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. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2023.

[5] 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?, July 2024.

[6] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, November 2024.

[7] Su Kara, Fazle Faisal, and Suman Nath. WABER: Evaluating Reliability and Efficiency of Web Agents with Existing Benchmarks. In *ICLR 2025 Workshop on Foundation Models in the Wild*, March 2025.

[8] 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, June 2024.

[9] Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. ST-WebAgentBench: A Benchmark for Evaluating Safety and Trustworthiness in Web Agents, August 2025.

[10] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Song XiXuan, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. VisualAgent-Bench: Towards Large Multimodal Models as Visual Foundation Agents. In *The Thirteenth International Conference on Learning Representations*, October 2024.

[11] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. Android-World: A Dynamic Benchmarking Environment for Autonomous Agents. In *The Thirteenth International Conference on Learning Representations*, October 2024.

[12] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and Characterizing Reward Gaming. *Advances in Neural Information Processing Systems*, 35: 9460–9471, December 2022.

[13] 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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024.

[14] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents, May 2025.

[15] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. {τ}-bench: A Benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser Interaction in Real-World Domains. In *The Thirteenth International Conference on Learning Representations*, October 2024.

[16] 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, April 2024.

[17] Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellerman, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing Best Practices for Building Rigorous Agentic Benchmarks, July 2025.