CLIFT: Conformal Self-Verification for Web Agent Training and Test-Time Scaling

Yifan Zhang **Yutong Dai** **Viraj Prabhu** **Zhiyuan Hu** **Ran Xu** **Zeyuan Chen**
Salesforce AI Research
{yifan.zhang, yutong.dai, viraj.prabhu}@salesforce.com
{zhiyuan.hu, ran.xu, zeyuan.chen}@salesforce.com

Abstract

Open-source web agents are now strong enough to execute realistic browser tasks, but training them with reinforcement learning still depends on weak supervision: binary task success is too sparse for credit assignment, while frontier-language-model judges are too expensive to call at every step and cannot be assumed available at deployment. We introduce CLIFT, a training and test-time scaling method built around conformal self-verification. During training, the agent answers natural-language verification questions about its own rollouts; a Compositional Conformal Certifier keeps only question signals whose URL-conditional evidence agrees with a training-time judge, assigns signed trust weights through polarity-aware lift, and blends the resulting verifier score into per-step rewards in a way that never subtracts from the judge baseline. At test time, the same certified bank is frozen and reused as structured evidence for Conformal Trajectory Selection (CTS): the agent samples a greedy rollout and one or more diverse retries, the self-verifier summarises each URL trace, and a conservative majority-vote rule chooses whether to swap away from the current incumbent without calling any external judge. This single mechanism supports three settings. On WebArena Infinity, CLIFT achieves state-of-the-art performance among open-source web agents. On VisualWebArena, a bank trained with the open model transfers to GPT-5.5 at test time and reaches state-of-the-art performance under the canonical harness. On Online Mind2Web, without training an agent on the benchmark, translating the certified question bank improves a live-web agent in zero-shot evaluation. Together these results position conformal self-verification as a way to turn costly judge feedback into a reusable training signal and a judge-free test-time scaling signal.

1 Introduction

Open-source web agents have reached the point where they can operate realistic browser interfaces, but the verifier remains the bottleneck for improving them with reinforcement learning. A binary success signal is easy to deploy but too sparse for credit assignment: when all rollouts in a GRPO group [Shao et al., 2024] receive the same outcome, the advantage estimate collapses. A frontier language model judge [Bai et al., 2022, Liu et al., 2023] provides richer feedback, but it is costly during training and cannot be assumed available at test time. This tension is especially visible in web-agent benchmarks: WebArena-Infinity (WAI) [Zhou, 2026] uses programmatic per-app verifiers, VisualWebArena (VWA) [Koh et al., 2024] uses outcome-based URL/DOM checks, and live-web suites such as Online-Mind2Web (OM2W) [Xue et al., 2025] depend on noisy LLM judges. The policy is often not the only limiting object; the way we score and select its rollouts can dominate the observed success rate.

CLIFT asks whether the judge's feedback can be turned into a reusable verifier that the agent carries with itself, rather than a separate process reward model [Lightman et al., 2024, Wang et al., 2024] or a post-hoc best-of-N scorer [Cobbe et al., 2021, Snell et al., 2025]. CLIFT maintains a bank of

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

Table 1: **Feedback mechanisms for web-agent RL and test-time scaling.** Dense = step-level feedback beyond binary outcome; URL = URL-state grounding; Cert. = certified trust; J-free = judge-free test time; TTS = test-time scaling; Xfer = reusable bank transfer.

Approach	Dense	URL	Cert.	J-free	TTS	Xfer
Outcome RLVR / GRPO (trajectory outcome) [Shao et al., 2024]	×	×	×	✓	×	×
Stratified GRPO (structural strata) [Zhu et al., 2025]	✓	×	×	✓	×	×
Rich-feedback self-distillation (teacher feedback) [Hübotter et al., 2026]	✓	×	×	✓	✓	×
Process / verifier reward models (learned PRM) [Lightman et al., 2024]	✓	×	×	✓	×	×
Post-hoc search / BoN (sample then select) [Snell et al., 2025]	×	×	×	✓	✓	×
Conformal routing / judging (uncertainty gate) [Badshah et al., 2026]	×	×	✓	×	×	×
CLIFT (URL-certified bank for training and CTS)	✓	✓	✓	✓	✓	✓

natural-language verification questions about web-agent behaviour and certifies which questions are reliable at which URL states. The same bank is used in two places (Figure 2). During training, certified bank evidence is blended with the comparative judge to produce a denser per-step reward and URL-stratified advantages. At test time, the trained agent runs a greedy trajectory plus diverse retries; the frozen certified bank summarizes each URL trace; and a conservative trajectory selector, Conformal Trajectory Selection (CTS), emits one trajectory per task. The common object is not a new model but a calibrated verifier: a set of questions, URL scopes, polarity signs, and conformal trust weights that can be reused after training.

The certifier is the part that makes this more than prompt engineering. Each verification question carries a URL scope and a polarity sign indicating whether YES is evidence of progress or failure. The Compositional Conformal Certifier (CCC) runs a Mondrian adaptive conformal tracker [Gibbs and Candès, 2021, Vovk, 2012] per question and URL granularity to estimate whether the question separates high- and low-judge rollouts there. Certified questions receive signed trust weights. During training this gives a verifier reward that can only add evidence on top of the judge baseline; unlike a linear blend, it cannot pull the reward below what the judge alone would give. During test-time scaling, the same signed evidence is frozen and presented to a comparative selector over two completed trajectories. In both cases, the question bank is a compact interface between expensive judge feedback and the agent's own rollouts.

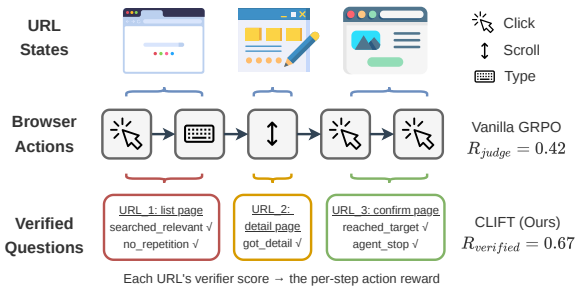


Figure 1: **CLIFT concept.** Vanilla GRPO broadcasts one trajectory-scalar reward; CLIFT scores each visited URL with verification questions and inherits the URL score to actions taken there. Full pipeline: Figure 2.

We evaluate this idea in three complementary ways. On WAI, where programmatic verifiers provide dense training supervision, CLIFT improves a trainable open Gemma-4 agent and reaches the strongest public 9-app score we are aware of, with training and CTS on the same open backbone. On VWA, the bank created during Gemma-4 training serves as a test-time verifier for GPT-5.5, showing that the learned verification knowledge transfers to a stronger closed backbone and reaches state-of-the-art performance under the canonical harness. On OM2W, no agent is trained at all: the certified bank is translated and recalibrated, then used by CTS to improve live-web test-time scaling. Together these support one claim: conformal self-verification is useful both as a training signal and as a reusable, judge-free test-time scaling signal. Related work is discussed in §7.

2 CLIFT: Conformal Lift-calibrated Introspective Filtering of Trajectories

CLIFT is a GRPO-family policy-gradient method [Shao et al., 2024] that changes how rollouts are scored and reused. It turns natural-language verification questions into a conformally certified URL-aware verifier, uses that verifier to add non-negative per-step evidence during training, and

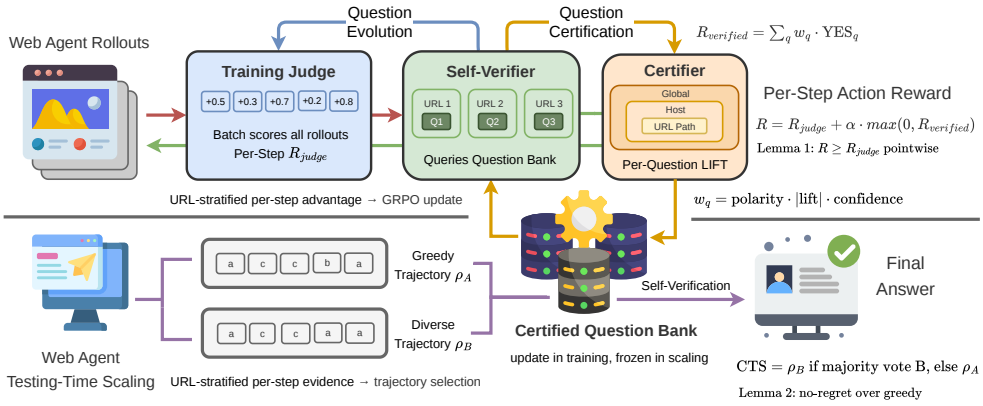


Figure 2: **CLIFT overview.** *Training:* rollouts are scored by a comparative judge and by a self-verifier that queries a polarity-tagged question bank. The CCC certifies question weights by URL tier, yielding a signed verifier score that is asymmetrically added to the judge reward and then converted into URL-stratified token advantages. *Test-time scaling:* the trained agent emits a greedy rollout and diverse retries; the frozen certified bank produces per-URL evidence for each pairwise comparison, and a conservative repeated-vote rule decides whether to keep the incumbent or swap.

freezes the same bank for CTS at test time. Thus both halves of Figure 2 share one object: questions with URL scopes, polarity signs, and certified trust weights.

2.1 The Conformal Verifier

A verification question (VQ) is a YES/NO check on the agent's behaviour. Each question q has a URL pattern π_q and polarity $\sigma_q \in \{-1, 0, +1\}$ indicating progress, failure, or ambiguity. At training and test time, the agent's base model answers applicable questions with the task adapter detached. Answers at URL state u are aggregated into

$$R_i^{VQ}(u) = \text{clip}_{[-1,1]}\Big(\sum_{q \in Q^{\text{cert}}(u)} w^q_{\tau^\star(u)} \cdot \mathbf{1}[y_i^q(u) = \text{YES}] \Big), \tag{1}$$

where $y_i^q(u)$ is the verifier's answer to q for rollout i at u, $Q^{\text{cert}}(u)$ contains certified in-scope questions, and $w^q_{\tau^\star(u)}$ is the most-specific certified trust weight (Eq. 4). Positive weights indicate progress, negative weights indicate failure, and an empty set gives $R_i^{VQ}(u) = 0$.

Certification. The same question can be reliable on one app and useless on another, so the Compositional Conformal Certifier (CCC) keeps a Mondrian-ACI tracker [Vovk, 2012, Gibbs and Candès, 2021] for each question and URL tier $\tau \in \{\text{global}, \text{host}, \text{URL-path}\}$. The non-conformity score is polarity-aware:

$$\begin{aligned} s_k^{q,\tau} &= \mathbf{1}\big[\,\text{pred}_q(y_k^q) \neq z_k\,\big], \quad z_k = \mathbf{1}[\hat{R}_k^{\text{judge}} > \theta_j], \\ \text{pred}_q(\text{YES}) &= \mathbf{1}[\sigma_q{=}{+1}], \quad \text{pred}_q(\text{NO}) = \mathbf{1}[\sigma_q{=}{-1}]. \end{aligned} \tag{2}$$

Here k indexes judge-scored observations of q at tier τ, with answer y_k^q and judge score $\hat{R}_k^{\text{judge}} \in [0, 1]$ binarised at θ_j. Positive questions conform when YES agrees with a high judge score; negative questions conform when YES agrees with a low score. Neutral questions do not certify. With $\text{lift}^q_{k,\tau}$ the mean judge-score difference between YES and NO examples at tier τ, a tier earns trust and, if $\sigma_q \neq 0$ and $t^q_\tau > 0$, is certified with signed weight

$$t^q_\tau = \big|\text{lift}^q_{k,\tau}\big| \cdot \rho^q_{k,\tau}, \tag{3}$$

$$w^q_\tau = \sigma_q\, t^q_\tau \;\in\; [-1, +1], \tag{4}$$

where $\rho^q_{k,\tau} \in [0, 1]$ discounts low observation counts, chance-level agreement with the judge, and held-out sign flips (Appendix A.3), and $w^q_\tau = 0$ otherwise. At URL u we query the most specific certified tier, falling back URL-path → host → global. The ACI tracker's running miscoverage $\alpha^q_{k,\tau}$

against $\alpha^\star$ is a diagnostic outside Eqs. 3–4, so "certified" denotes empirical per-tier reliability, with ACI controlling each tracker's long-run miscoverage [Gibbs and Candès, 2021] (worked example: Appendix A.5).

Bank growth. The bank starts from a judge-grounded seed set of VQs, each tagged with a polarity. At periodic training intervals, a CEGAR-style loop [Clarke et al., 2000] proposes a small batch of new URL-scoped questions separating top and bottom rollouts. New questions enter uncertified. A sanity veto demotes polarity to neutral when observed lift contradicts the tag:

$$\sigma_q \leftarrow 0 \quad \text{if} \quad n^q \geq n_{\text{veto}} \ \wedge \ \operatorname{sgn}\big(\text{lift}^q\big) \neq \sigma_q^{\text{LLM}} \ \wedge \ \big|\text{lift}^q\big| > \delta_{\text{veto}}, \tag{5}$$

with global-tier n^q and lift^q, proposal-time tag σ_q^{LLM}, and thresholds $n_{\text{veto}}, \delta_{\text{veto}}$. The same lift estimate therefore certifies questions and catches polarity mistakes.

2.2 Training: Per-step Rewards and URL-stratified Advantages

For each task, one comparative judge call scores a GRPO group of G rollouts from human-readable action traces and returns trajectory scores R_i^{judge}, per-step scores $r_t^{i,\text{judge}}$, and the rationales CEGAR uses; the judge is training-only. In parallel, the self-verifier answers bank questions for each rollout and URL state. The training reward is

$$\tilde{r}_t^i \ = \ r_t^{i,\text{judge}} \ + \ \beta \cdot \max\big(0,\, R_i^{VQ}(u_t^i)\big), \tag{6}$$

where u_t^i is the URL state at step t of rollout i and β controls the verifier contribution. Since the verifier term is clipped at zero, $\tilde{r}_t^i \geq r_t^{i,\text{judge}}$: certified evidence can strengthen the judge signal, but cannot drag it down.

URL-stratified per-token advantage. We discount step rewards into returns $G_t^i = \sum_{\ell=t}^{T_i-1} \gamma^{\ell-t}\, \tilde{r}_\ell^i$ with discount γ, then group-normalise by the return mean $\bar{G}_c$ and standard deviation $\sigma_{G,c}$ of each cell $c_t^i = (\text{group},\, t,\, u_t^i)$:

$$\widehat{A}_t^i = \frac{G_t^i - \bar{G}_{c_t^i}}{\sigma_{G,c_t^i} + \varepsilon}. \tag{7}$$

Advantages are broadcast to tokens over tool-call spans; tokens outside those spans receive zero gradient. The GRPO update is unchanged, but credit assignment becomes URL-local.

2.3 Test Time: CTS (Conformal Trajectory Selection)

At deployment the external training judge is dropped. The policy emits candidate rollouts under a small budget K: a greedy incumbent ρ_A and one or more diverse retries ρ_B. CLIFT then uses the frozen bank and base self-verifier in a pairwise selector to decide whether to keep the incumbent or swap to the retry, avoiding brittle step-wise selection. For $K{>}2$, the same selector is applied as a tournament chain that maintains one incumbent and still emits a single trajectory.

For each rollout $\rho \in \{\rho_A, \rho_B\}$, let $U(\rho) = (u_0, u_1, \ldots, u_{T_\rho})$ be its sequence of distinct URL states (consecutive duplicates collapsed). The verifier produces a per-state evidence record at every $u_j \in U(\rho)$:

$$E_j(\rho) = \Big\{ \big(q,\ w^q_{\tau^\star(u_j)},\ \text{lift}^q_{\tau^\star(u_j)},\ p^q(\rho, u_j)\big) \ : \ q \in Q^{\text{cert}}(u_j),\ \pi_q \sim u_j \Big\}, \tag{8}$$

where $\pi_q \sim u_j$ denotes a URL-pattern match, weights and lifts are frozen at the end of training, and $p^q(\rho, u_j)$ is the verifier's YES probability on the partial trajectory. We keep this as structured evidence rather than a single scalar.

The evidence sheets, action traces, and final states are passed to the self-verifier in a comparative prompt, which outputs **A** (keep greedy), **B** (swap), or **TIE**.

To prevent verifier variance from causing harmful swaps, each pairwise comparison uses a small odd number m of independent self-verifier calls and applies a conservative majority rule:

$$\text{CTS}(\rho_A, \rho_B) = \begin{cases} \rho_B & \text{if at least } \lceil (m+1)/2 \rceil \text{ calls vote B,} \\ \rho_A & \text{otherwise (including ties and majority-A votes).} \end{cases} \tag{9}$$

The asymmetric default is the safety valve: ties and uncertainty keep ρ_A. If, whenever ρ_A succeeds and ρ_B fails, the m votes are independent given the pair and each votes B with probability at most $\frac{1}{2} - \delta$ ($\delta > 0$), then (Proposition 1, Appendix A.4)

$$\mathbb{E}[\mathrm{SR}(\mathrm{CTS})] \;\geq\; \mathbb{E}[\mathrm{SR}(\rho_A)] - \epsilon(m, \delta), \qquad \epsilon(m, \delta) = e^{-2m\delta^2}, \tag{10}$$

and a K-candidate chain loses at most $(K-1)\,\epsilon(m, \delta)$; the bound limits harmful swaps only. Here m is the number of verifier votes in a pairwise comparison, whereas K denotes the rollout budget reported in the result tables.

Training and test time thus share one object, per-URL-state certified bank evidence, aggregated into rewards in training and into a comparative choice at test time.

3 Experimental Design

WAI is the primary training setting: its self-hosted SaaS-style apps expose programmatic verifiers that can support online RL and public-board comparison. VWA is the controlled transfer setting: its Reddit, Shopping, and Classifieds environments use outcome-based URL/DOM checks, so we test whether a bank learned while training an open model remains useful when applied to GPT-5.5 at inference. OM2W is the live-web transfer setting: it contains 300 public-domain tasks judged by an LLM rubric, and no reported policy is trained on OM2W data.

Table 2: Experimental roles. OM2W supplies no reported policy-training data.

Benchmark	Training role	Eval role
WAI	train Gemma-4	in-domain + board
VWA	train bank	transfer to GPT-5.5
OM2W	none	zero-shot transfer

3.1 Task synthesis and bank construction

Training tasks are synthesized without reading evaluation tasks. WAI seeds come from app descriptions; VWA seeds come from SynthAgent [Wang et al., 2026] site exploration. Two frontier LLMs expand the seeds, and a shared filter removes malformed, duplicate, hardcoded, pure-lookup, and trivial-navigation tasks. The reported checkpoint trains on $1{,}886$ WAI tasks and $1{,}272$ VWA tasks across 16 LoRA shards. OM2W is excluded from policy training and used only for bank transfer and CTS. Appendix A.2 gives CLIFT training details.

3.2 Per-benchmark protocols

The three benchmarks probe complementary properties of the same verifier bank: WAI tests whether certified self-verification can *train* a stronger open agent, VWA tests whether the trained-on-open bank *transfers* to a frozen closed backbone, and OM2W tests whether the same bank *recalibrates* for live-web zero-shot selection; prompts are in Appendix A.1.

WAI. We train Gemma 4 31B + LoRA with the full URL-path / host / global CCC hierarchy. The comparison isolates three effects: the base browser-use backbone, vanilla GRPO with the same comparative judge but no certified bank, and CLIFT with certified step-level feedback. We then apply CTS to the trained checkpoint to measure whether the frozen bank gives an additional no-regression test-time selection gain on the same in-domain tasks.

VWA. We use the bank learned during open-model training as structured evidence for a stronger closed backbone. GPT-5.5 is not updated and sees the same completed-task interface; the only added component is CTS over finished trajectories. This setting isolates whether the bank stores transferable web-interaction knowledge rather than merely improving the Gemma-4 policy that produced it.

OM2W. We translate and recalibrate the WAI/VWA bank for live-web domains, then run CTS directly at inference. Reported rows use upstream Gemma-4 or GPT-5.5 weights with no OM2W-specific LoRA. Question translation rewrites benchmark-specific URL patterns into the live-web equivalents, drops predicates that no longer apply, and adds OM2W-specific failure-mode questions for bot-block pages, hallucinated extraction, partial-filter application, and wrong-final-answer stops. The recalibration pass scores each translated or new question on 120 judge-labelled OM2W rollouts (60 successful, 60 failed; Appendix A.7) using the same lift-based certifier as training, so any reported gain comes from cross-benchmark transfer of the certified question taxonomy plus the calibration recipe, not from policy training on OM2W.

Table 3: **WAI leaderboard slice.** Success rates on the public 9-app comparison set; values are percentages without percent signs. The CLIFT +CTS column reports the trained Gemma-4 backbone with $K=3$ Conformal Trajectory Selection (one selected trajectory per task). Overall is the task-weighted success rate over the 9 apps and matches the CLIFT +CTS row of Table 6.

App (n)	**Gemini 3 Flash + BU**	**Qwen 3.5+**	**Kimi K2.5**	**Gemma-4 (base, ours)**	**Gemma-4 + CLIFT +CTS ($K=3$, ours)**
Elation Clinical (120)	82.0	54.0	50.0	69.2	**91.7**
Elation Prescription (120)	81.0	42.0	23.0	70.0	**91.7**
GitLab Plan & Track (140)	64.0	37.0	39.0	59.3	**69.3**
Gmail (60)	75.0	57.0	70.0	60.8	**78.3**
Handshake (200)	50.0	50.0	50.0	47.0	**56.0**
Linear Acct Mgmt (120)	**73.0**	66.0	54.0	53.3	64.2
PayPal Wallet (140)	89.0	71.0	71.0	82.1	**90.7**
Super-human (120)	50.0	26.0	15.0	45.8	**52.5**
Xero Invoicing (120)	**81.0**	56.0	52.0	75.0	79.2
Overall (9 apps)	70.1	50.7	46.4	61.8	**74.6**

4 Result Analysis

We report results in the order of §3.2, from the most direct supervision to none. §4.1 asks whether certified self-verification improves an open, trainable agent on WAI; §4.2 asks whether the bank learned during that training remains useful for a stronger closed backbone on VWA; and §4.3 asks whether the same bank can be translated and recalibrated for live-web selection on OM2W, where no policy is trained. §4.4 then separates the contributions of training and test-time selection. Every CTS row emits one selected trajectory per task, scored like a single rollout.

4.1 Training with certified feedback

The public-board comparison in Table 3 is the main WAI headline: CLIFT-trained Gemma-4 with $K=3$ CTS reaches the highest 9-app success rate among the published systems we compare against (Claude Opus 4.6 is the training-time judge; CTS itself calls no external judge). The gain is not concentrated in one easy app: CLIFT wins 7 of 9 apps, is competitive on the rest, and improves the same Gemma-4 browser-use backbone by 12.8 percentage points overall. Gemini 3 Flash leads on Linear and Xero, and the larger 110-task Gmail variant is reported separately in Figure 4. This is the in-domain training claim: the verifier bank is not merely a post-hoc reranker; it changes the online RL trajectory enough for a 31B open model to exceed the public frontier reference on the matched WAI slice.

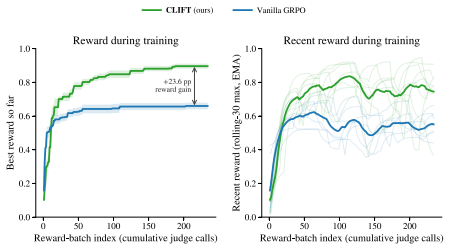


Figure 3: **WAI training reward.** CLIFT separates from vanilla GRPO under the same judge and browser-use stack.

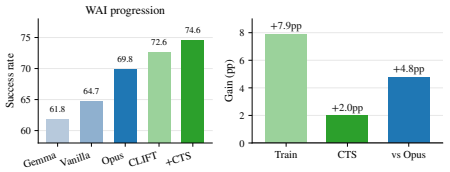


Figure 4: **WAI progression and gains.** Operating points and deltas.

The training trace in Figure 3 shows that the gap is not just a larger evaluation rollout budget: once the certified bank supplies signed per-step evidence, training separates from vanilla GRPO before selection. Reading WAI through training dynamics and aggregate success, rather than a single leaderboard number, clarifies CTS's role: it operates on an already-improved policy, so a small additional lift is expected. Figure 4 orders the operating points: base Gemma-4, vanilla GRPO, Opus 4.6 (on common task IDs), CLIFT, and CLIFT +CTS; the CTS bar marks the final conservative selection point rather than a pure reranking lift. Training supplies the large WAI jump; CTS preserves it without per-app regression (examples: Appendix A.9) and adds a small lift. Larger test-time-scaling effects appear on VWA and OM2W below.

4.2 Bank transfer to a closed backbone

Table 4: **VWA success rates.** C/S/R are Classifieds, Shopping, and Reddit; Avg is weighted by env size. For CTS rows, K denotes rollout budget and one selected trajectory is emitted per task.

Method	C	S	R	Avg.
SGV [Andrade et al., 2026]	52.0	**57.0**	33.0	50.2
WALT [Prabhu et al., 2026]	**64.1**	53.4	39.0	52.9
Gemma-4 base	41.5	28.5	24.3	30.9
Gemma-4 + CLIFT	44.9	30.7	31.0	34.4
Gemma-4 + CLIFT+CTS (K=2)	45.3	32.6	31.4	35.6
GPT-5.5 + CLIFT	60.7	43.1	47.1	48.5
GPT-5.5 + CLIFT+CTS (K=4)	62.4	51.0	**50.0**	**53.7**
Human	91.1	88.4	87.1	88.7

VWA is the controlled transfer benchmark between the two training settings. Its three Docker environments have deterministic URL/DOM outcome verifiers, but their visual grounding, page layouts, and navigation routines differ from WAI's SaaS apps. We therefore use VWA to ask whether a bank learned while training an open Gemma-4 agent can still provide useful evidence when the evaluated policy is GPT-5.5, the executor weights are frozen, and only the certified question bank is reused at inference. The published baselines in Table 4 follow the VisualWebArena comparison protocol [Koh et al., 2024]; lower published baselines are omitted from the main table for space.

The Gemma rows isolate the training effect on an open backbone: CLIFT improves the base model before any closed-model transfer is introduced, and CTS adds a smaller additional lift. This mirrors the WAI pattern, but at lower absolute success rates because VWA's visual and navigation demands are less aligned with the training harness. The GPT-5.5 rows test the transfer claim directly. The policy weights are unchanged; only the certified bank evidence and CTS selection are added at inference. Adding CTS with rollout budget $K{=}4$ raises the weighted average to 53.7, narrowly above WALT's 52.9 reference, with gains on all three environments. The strongest message is not that VWA is solved (the human row remains far above every agent) but that a bank constructed while training an open model remains useful when the underlying executor is a closed model. On identical candidate sets, random selection gains nothing over the anchor, whereas CTS gains 2.2 points (paired $p = 0.003$) and is statistically indistinguishable from a bank-free GPT-5.5 vote (Appendix A.8).

4.3 Cross-benchmark transfer

OM2W is our hardest setting: 300 live-web tasks across roughly 30 public domains, judged end-to-end by an LLM rubric rather than DOM matching. We use it as a pure transfer benchmark: no reported policy is fine-tuned on OM2W data. The only adaptation is at the verifier layer: questions mined from the WAI/VWA banks are filtered, rewritten, expanded for live-web failure modes, and recalibrated on OM2W rollouts. Table 5 compresses the main bank evidence. The transferred taxonomy does not collapse under the noisier live-web judge; instead, 100 of 132 questions certify, including both positive progress signals and negative failure-mode signals. This is the strongest evidence that the bank is a reusable artifact rather than a benchmark-local prompt list.

Table 5: **Transferred bank on OM2W.** Questions are reused from WAI/VWA or expanded, then recalibrated on OM2W rollouts; the final certified bank has 75 positive-lift and 25 negative-lift questions with mean lift 0.34.

Origin	Construction	$\lvert\mathcal{Q}\rvert$	Certified
WAI bank	keep/rewrite	36	33 (91.7%)
VWA bank	keep/rewrite	48	31 (64.6%)
OM2W expansion	live-web checks	48	36 (75.0%)
Final bank	**132 qs**	**132**	**100 (75.8%)**

Figure 5 reports realised success: every "+ ours" row emits one selected trajectory per task, not an oracle union over samples. On the open backbone, the transferred bank and CTS raise Gemma-4 from 40.0 to 45.0, enough to exceed the GPT-5.5 low-effort baseline on the same judge (43.7). On the stronger GPT-5.5 high-effort backbone, the same test-time mechanism raises success from 49.7 to 61.0 at rollout budget $K{=}4$, a $+11.3$pp gain; on the 180 tasks outside the bank's calibration sample the gain is larger (45.6 to 60.0, paired McNemar $p < 10^{-5}$; Table 9). This is where test-time scaling is most visible: unlike WAI, where training already moves the greedy policy close to its ceiling, OM2W leaves substantial room for the selector to choose among distinct completed trajectories.

The cross-benchmark result also clarifies what transfers. The policy does not: the Gemma rows use upstream weights and the GPT-5.5 rows keep the closed model fixed. What transfers is the certified question taxonomy plus the calibration recipe, which turns prior judge feedback into structured evidence for a new live-web environment. Because the bank is just a list of URL-scoped, polarity-tagged questions with calibrated trust weights, recalibration on a new domain only requires a sample of judge-scored rollouts; no policy update is needed to begin selecting trajectories. The same property explains why one bank can serve both an open and a closed backbone: the selector reads evidence about the trajectory, not about the model that produced it. Detailed bank construction and transfer tables are in Appendix A.6.

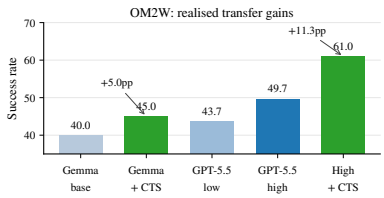


Figure 5: **OM2W transfer gains.** Realised SR; K is rollout budget.

4.4 Ablation study

Table 6: **WAI ablation summary.** Success rates are shown without percent signs. All training rows use Claude Opus 4.6 as the comparative judge with the same browser-use backbone; only the bank/CTS components vary. Steps are averaged over successful tasks in available logs; CTS improves/ties/regresses 5/4/0 apps with a 17.8% swap rate pooled over all tasks.

Variant	Isolated change	SR	Δ vs base	Avg solve steps
Gemma-4 base	no RL, greedy rollout	61.8	–	15.4
Vanilla GRPO	scalar comparative-judge reward	64.7	+2.9	18.5
CLIFT	certified step reward + URL advantages	72.6	+10.8	16.0
CLIFT +CTS	add conservative trajectory selection	**74.6**	**+12.8**	15.5
Opus 4.6	closed-model reference on the common task IDs	69.8	+8.0	–

Table 6 consolidates the ablations. Base Gemma-4 is the no-RL reference; vanilla GRPO isolates scalar judge-reward training; CLIFT adds certified step feedback; and CLIFT +CTS adds only test-time selection. Training accounts for most of the WAI gain, while CTS improves five apps, ties four, and never lowers the single-rollout CLIFT anchor.

The decomposition further shows that the gain is not explained simply by additional interaction or inference-time search. Vanilla GRPO improves the base model by only 2.9 points while increasing the average number of steps on successful trajectories from 15.4 to 18.5. In contrast, CLIFT reaches a 10.8-point improvement with an average of 16.0 steps, suggesting that the certified URL-local feedback primarily improves credit assignment rather than encouraging longer trajectories. CTS then adds a further 2.0 points while reducing the average to 15.5 steps. Together with its 17.8% swap rate and zero per-app regressions, this supports the intended division of labor: CLIFT produces the stronger policy during training, while CTS intervenes selectively at test time when the certified bank provides sufficient evidence to prefer an alternative rollout.

5 Discussion and future work

Beyond the headline numbers, two questions shape how CLIFT should be read: what makes the certified bank more than a list of generated questions, and how CLIFT behaves when its evidence is weak. Both matter for using CLIFT outside the settings studied here, where the judge, the benchmark, and the URL structure may all differ from those the bank was certified on. We discuss both, then outline how the bank could become executable, which would make its checks easier to audit and to carry across environments.

What makes the bank non-trivial. Proposing questions is easy; knowing which answers to trust, where, and in which direction is not. Most proposals carry no signal: after training only 44 of 992 WAI and 21 of 1,565 VWA questions keep non-zero trust (Appendix A.6), so an uncertified bank would mostly inject noise. A question informative on one app may be useless on another, so trust is estimated per URL tier and polarity is checked against observed lift rather than taken from

the proposer (Eq. 5). The evidence is also used conservatively, only adding to the judge reward (Eq. 6) and swapping only on a repeated majority (Proposition 1); a random choice among the same candidates gains nothing (Appendix A.8).

Failures are visible. On WAI and VWA, many URL states accumulate enough observations for signed trust weights; on OM2W, the noisier live-web judge leaves more questions uncertified. This refusal is a diagnostic rather than a silent collapse: it says the current judge, state partition, or question bank does not support reliable distillation, and keeps noisy verifier evidence out of both the training reward and the CTS selector. The certified fraction of a bank can therefore serve as a cheap health check: when it drops after a change of judge, benchmark, or URL partition, the new setting needs recalibration before its evidence is trusted, which is the step the OM2W transfer performs explicitly.

Toward executable checks. A natural next step is to replace natural-language questions with executable browser-state assertions written by a coding agent. The agent would learn not only to act on the web, but also to write compact checks that can be executed during rollouts and reused for test-time selection, reducing dependence on a separate self-verification bank while preserving auditability. The certifier would apply unchanged, since an assertion is simply another question whose trust is earned from observed lift, and an executed check returns the same answer on the same state, removing verifier sampling noise from that estimate.

6 Limitations

CTS assumes that a task can be attempted more than once from the same state. WAI and VWA reset between attempts, so candidates can run in parallel; on live websites retries are sequential, multiplying latency by the rollout budget (Table 11), and a purchase or a sent message can change state irreversibly, after which another attempt is neither independent nor safe. Selection there suits read-only or reversible tasks or a sandboxed replica, and training, which we run only in sandboxes, faces the same constraint. We also train one open-weight backbone with a frontier judge whose biases calibration detects only through observed disagreement; the transfer results do not show one bank to be universal across web domains; and stronger open agents carry misuse risk that calls for rate limits and domain-specific safeguards.

7 Related work

CLIFT sits at the intersection of web-agent benchmarks, RL credit assignment, test-time search, and reliable verifier-based selection. The closest systems usually improve one component of this loop: stronger web interfaces for agents, denser reward signals for training, or calibrated decision rules for deployment. CLIFT instead studies how one certified verification bank can connect these components, carrying judge feedback from online RL into test-time trajectory selection.

Web agents and test-time search. Language agents combine reasoning, browser actions, and tools [Nakano et al., 2021, Yao et al., 2023b, Shinn et al., 2023, Schick et al., 2023, Qin et al., 2024]; search methods add deliberation or tree expansion at inference time [Wei et al., 2022, Yao et al., 2023a, Zhou et al., 2024a, Snell et al., 2025, Song et al., 2024]; and modern web, software, desktop, and embodied benchmarks stress realistic interfaces [Zhou et al., 2024b, Koh et al., 2024, Deng et al., 2023, Yao et al., 2022, Shridhar et al., 2021, Yang et al., 2024, Xie et al., 2024]. CLIFT differs from prompt-level search because its selector reuses a certified per-URL bank learned during RL training, not just more samples.

RL credit assignment and process rewards. GRPO and related RLVR methods improve LLMs from scalar outcomes [Schulman et al., 2017, Shao et al., 2024, DeepSeek-AI, 2025, Qi et al., 2025], while agent variants refine credit assignment through stratification, anchor states, success rates, or Monte-Carlo values [Zhu et al., 2025, Feng et al., 2025, Chen et al., 2025, Kazemnejad et al., 2025]. Process-verifier and rich-feedback methods provide denser supervision [Cobbe et al., 2021, Lightman et al., 2024, Wang et al., 2024, Hübotter et al., 2026, Madaan et al., 2023, Agarwal et al., 2024], and LLM judges provide it for open-ended generations [Bai et al., 2022, Liu et al., 2023, Zheng et al., 2023, Rafailov et al., 2023]. CLIFT keeps GRPO but changes the reward object: judge feedback is augmented by a certified self-verifier and normalised by URL state.

Conformal reliability and specification mining. Conformal prediction gives finite-sample reliability guarantees [Vovk et al., 2005, Angelopoulos and Bates, 2021, Gibbs and Candès, 2021, Vovk, 2012], with recent adaptations to language modeling, decision making, off-policy prediction, LLM-agent routing, and LLM judging [Quach et al., 2024, Lekeufack et al., 2024, Kuipers et al., 2024, Si et al., 2026, Badshah et al., 2026]. Those works calibrate outputs, policies, or judgments; CLIFT calibrates the *verification questions* themselves, using URL scope, polarity, and a Mondrian-ACI trust weight. A CEGAR-style loop [Clarke et al., 2000] turns judge rationales into reusable specifications for both training and test-time selection.

8 Conclusion

CLIFT turns expensive judge feedback into a reusable verifier for web agents: conformally certified URL-state evidence densifies training reward, then the frozen bank supports judge-free Conformal Trajectory Selection. Across WAI, VWA, and OM2W, the same bank helps train an open model, transfers to a stronger closed model, and recalibrates for live-web selection without policy fine-tuning. A trajectory score is consumed once; a certified bank can be inspected, recalibrated, and reused.

The bank is the durable artefact behind all three results. Each certified question carries a URL scope, a polarity sign, and a trust weight derived from observed lift, with provenance back to the judge rationale that introduced it. The bank is therefore small enough to ship with the agent and auditable when a result looks off; when a question's lift collapses on a new distribution, the certifier withholds trust rather than amplifying noise. What the judge taught the agent to notice thus persists from training reward to test-time selection, from open to closed backbones, and from sandboxed verifiers to live-web judges, and stronger future judges can extend it without discarding earlier rounds.

Acknowledgments and Disclosure of Funding

We thank our colleagues at Salesforce AI Research for helpful discussions and for the compute and infrastructure support that made the online browser-agent training in this work possible. We are grateful to the anonymous reviewers and the area chair for their constructive feedback, which improved the clarity of the paper. We also thank the maintainers of WebArena-Infinity, VisualWebArena, and Online-Mind2Web for releasing their benchmarks and evaluation harnesses, and the developers of the open-source tools this work builds on, including browser-use, vLLM, and TRL.

This work was funded by Salesforce, where all authors are employed at Salesforce AI Research. The authors have no other funding or competing interests to disclose.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Moises Andrade, Joonhyuk Cha, Brandon Ho, Vriksha Srihari, Karmesh Yadav, and Zsolt Kira. Let's think in two steps: Mitigating agreement bias in MLLMs with self-grounded verification. In *International Conference on Learning Representations*, 2026.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Sher Badshah, Ali Emami, and Hassan Sajjad. SCOPE: Selective conformal optimized pairwise LLM judging. In *International Conference on Machine Learning*, 2026. arXiv:2602.13110.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Yuhan Chen, Yuxuan Liu, Long Zhang, Pengzhi Gao, Jian Luan, and Wei Liu. STEP: Success-rate-aware trajectory-efficient policy optimization. arXiv:2511.13091, 2025.

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Computer Aided Verification*, 2000.

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

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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 *NeurIPS Datasets and Benchmarks*, 2023.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025.

Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 2021.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. arXiv:2601.20802, 2026.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *International Conference on Machine Learning*, 2025.

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. In *ACL*, 2024.

Tom Kuipers, Renukanandan Tumu, Shuo Yang, Milad Kazemi, Rahul Mangharam, and Nicola Paoletti. Conformal off-policy prediction for multi-agent systems. In *IEEE Conference on Decision and Control*, 2024.

Jordan Lekeufack, Anastasios N. Angelopoulos, Andrea Bajcsy, Michael I. Jordan, and Jitendra Malik. Conformal decision theory: Safe autonomous decisions from imperfect predictions. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332, 2021.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, Silvio Savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *International Conference on Learning Representations*, 2026. arXiv:2510.01524.

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

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *ICLR*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Yu Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *ICLR*, 2021.

Wenwen Si, Sooyong Jang, Insup Lee, and Osbert Bastani. Conformal constrained policy optimization for cost-effective LLM agents. In *AAAI Conference on Artificial Intelligence*, 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Annual Meeting of the Association for Computational Linguistics*, pages 7584–7600, 2024.

Vladimir Vovk. Conditional validity of inductive conformal predictors. In *Asian Conference on Machine Learning*, 2012.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics*, pages 9426–9439, 2024.

Zhaoyang Wang, Yiming Liang, Xuchao Zhang, Qianhui Wu, Siwei Han, Anson Bastos, Rujia Wang, Chetan Bansal, Baolin Peng, Jianfeng Gao, Saravan Rajmohan, and Huaxiu Yao. SynthAgent: Adapting web agents with synthetic supervision. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2511.06101.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, 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 *NeurIPS*, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2504.01382.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *NeurIPS*, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023a.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *NeurIPS Datasets and Benchmarks*, 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *International Conference on Machine Learning*, 2024a.

Shuyan Zhou. WebArena-Infinity: Generating browser environments with verifiable tasks at scale. `https://webarena.dev/webarena-infinity/`, March 2026.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024b.

Mingkang Zhu, Xi Chen, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Stratified GRPO: Handling structural heterogeneity in reinforcement learning of LLM search agents. arXiv:2510.06214, 2025.

A Appendix

This appendix contains evaluation prompts, the CLIFT-only training setup, detailed bank-transfer accounting for OM2W, and case studies drawn from evaluation logs.

A.1 Evaluation Prompts

WAI browser-use agent prompt. The WAI browser-use rollout code uses the following system and user templates (abridged only by replacing runtime fields with braces).

```
You are a browsing agent completing a task on a web app.
Each turn you see a simplified accessibility snapshot of the current page
listing interactable elements with numeric ids.  Call exactly ONE tool per
turn:  click(element_id), type_text(element_id, text, press_enter=True),
scroll(direction), goto(url), go_back(), press(key), hover(element_id),
stop(answer).
Do not narrate.  Emit a single tool call.
Task:  {intent}
Current URL: {url}
Page title:  {title}
Accessibility snapshot:  {snapshot}
Recent action history:  {history}
```

VWA Gemma-4 agent prompt. The VWA evaluation harness uses the Gemma-4 browser prompt below.

```
You are an autonomous intelligent agent tasked with navigating a web browser.
You will be given web-based tasks.
The user's objective:  This is the task you're trying to complete.
The current web page screenshot has numbered interactable elements.
The observation lists interactable elements as [id] [tagType] [text].
Use browser control tools to navigate and interact with pages.
Rules:  issue only valid actions; issue one tool call at a time; reason step
by step before acting; use element ids; verify before stop; do not stop
prematurely; do not construct internal URLs with goto.
OBJECTIVE: {objective}
URL: {url}
OBSERVATION: {observation}
PREVIOUS ACTION: {previous_action}
```

OM2W agent and WebJudge prompts. The OM2W rollout prompt uses the same browser-agent pattern, but for live websites and final-answer tasks:

```
You are a web browsing agent.  You will be given a task objective and must
interact with a real website to complete it.
Each turn you receive a screenshot with numbered element IDs, an
accessibility tree, the current URL, and your previous action.
Call exactly one tool per turn.  Do not construct URLs; click links on the
page.  Never stop on the first step.  For find/show/look-up tasks, navigate
to the relevant page first.  Verify before calling stop.
OBSERVATION: {observation}
URL: {url}
OBJECTIVE: {objective}
PREVIOUS ACTION: {previous_action}
```

The canonical OM2W judge (WebJudge; Xue et al., 2025) first extracts key points, scores snapshots, and then applies the final judging prompt:

```
You are an expert in evaluating the performance of a web navigation agent.
Given the user's task, the agent's action history, key points for task
completion, and important web pages in the trajectory, determine whether the
agent completed the task and achieved all requirements.
Important criteria:  filters must be applied correctly; exact ranges must
match; submission/display actions must be completed; invalid or empty
retrieved information may still be success if the required action was
correctly performed.
Respond in two lines:
Thoughts:  {reasoning}
Status:  "success" or "failure"
```

A.2 CLIFT Training Setup

This subsection documents the training setup for the reported CLIFT checkpoint, organised into the four moving parts of the system: the backbone and the task pool that drives RL rollouts; the compute budget and shard layout; the per-rollout reward and verification loop; and the certification rule that turns the running bank into the deployment object loaded by CTS at test time. The concrete hyperparameter values used by all four stages are listed separately in §A.3.

Backbone and shard layout. The reported CLIFT checkpoint trains Gemma-4-31B-it with a rank-8 LoRA adapter. We use 16 single-node shards, each with its own browser workers, reward pipeline, and LoRA adapter. After training, the per-shard adapters are averaged element-wise and merged into the base model for evaluation. OM2W is excluded from policy training entirely; we use it only for bank transfer and CTS at evaluation.

Seed pool. The training pool is synthesised from scratch without reading any evaluation task. WAI seeds (186 tasks) are drawn from each application's schema and user-facing documentation; VWA seeds (206 tasks) come from a separate SynthAgent crawl that probes the three Docker environments

via header-only navigation. Both seed sets describe what the apps *can do*, not what the public benchmarks *ask the agent to do*.

Frontier-LLM expansion. Two frontier LLMs (Claude Opus 4.6 and GPT-5.5) expand each seed into a draft pool of roughly $20\times$ candidates per seed. Each expansion call sees only the seed's schema or crawl trace and a small list of formatting constraints; it never sees the public benchmark task list, evaluator code, or any reference answer.

Four-stage filter. A shared filter then removes (i) *malformed* tasks that fail to parse or execute against a sandbox replay, (ii) *near duplicates* under embedding-based dedup at cosine 0.92, (iii) *hard-coded* tasks whose answers are already present verbatim in the seed schema, and (iv) *trivial-navigation* tasks that resolve in a single URL change without any UI interaction or form input. What remains is the $1{,}886$ WAI and $1{,}272$ VWA training tasks used for all reported runs.

Train/eval isolation. At no point during synthesis, filtering, sharding, or RL training did the pipeline read WAI, VWA, or OM2W evaluation task descriptions, evaluation HTML/DOM snapshots, or evaluation reward signals. The eval suites are first loaded only after all training runs and adapter averaging are complete, and only by the read-only evaluator.

Compute resources. Training ran as shard-parallel single-node jobs on H200 machines; at peak we used eight nodes, each with $8\times$H200 GPUs, and completed 16 shard jobs across the WAI and VWA training pools. Within each node, resources were split between vLLM serving, LoRA optimization, and browser-rollout processes. Closed-backbone VWA and OM2W evaluations used GPT-5.5 API inference for rollouts and CTS candidates, but did not fine-tune the closed model.

Rollout and verification loop. Each prompt produces a GRPO group of $G = 8$ browser rollouts. One comparative batch judge call scores the group at step level and supplies reasoning for CEGAR. In parallel, the base Gemma-4 model answers verification questions with the LoRA adapter detached, so verifier answers come from the same backbone without using the trained task adapter.

Certification, reward, and deployment object. For each question, CCC maintains global, host, and URL-path Mondrian-ACI trackers (target $\alpha^\star = 0.10$, reported as a diagnostic) and lift-based trust weights (Eqs. 3–4). Certified verifier evidence is added through the asymmetric per-step blend, so it can support the judge reward but cannot subtract from it. The resulting URL-stratified returns are written to token advantages over tool-call spans, and the frozen certified bank is the deployment object reused by CTS at test time.

A.3 Reported CLIFT Hyperparameters

The method section uses symbolic quantities; the concrete values used for the reported runs are: judge threshold $\theta_j = 0.5$; GRPO group size $G = 8$; ACI miscoverage target $\alpha^\star = 0.10$ (a tracker diagnostic, not a term in Eqs. 3–4); saturation-retirement count $n_{\mathrm{sat}} = 100$, above which a question whose YES-rate has saturated is retired to zero trust; trust discount $\rho^q_{k,\tau} = \big(1 - 1/\sqrt{n^q_{k,\tau} + 1}\big) \cdot \max(0,\, 2p^q_{k,\tau} - 1) \cdot g^q_\tau$, where n counts observations, $p = 1 - \bar{s}^{q,\tau}$ is the rate at which the polarity-implied prediction agrees with the judge, and $g \in [0.25, 1]$ demotes a tier whose held-out lift disagrees in sign with its calibration lift; CEGAR interval $M = 60$ training steps; CEGAR batch size 3 to 5 questions; polarity-veto thresholds $n_{\mathrm{veto}} = 50$ and $\delta_{\mathrm{veto}} = 0.05$; reward blend $\beta = 0.15$; return discount $\gamma = 0.95$; CTS vote count $m = 3$ with self-verifier temperature 0.7; greedy incumbent temperature 0; diverse retry temperature 1.0; and seed-bank sizes 281 WAI questions and 89 VWA questions before filtering and certification.

A.4 Conservativeness of CTS

Proposition 1. *Let ρ_A, ρ_B be two completed rollouts with success indicators $S_A, S_B \in \{0, 1\}$, and let CTS apply Eq. (9) with odd m. Assume that, conditional on (ρ_A, ρ_B) with $S_A = 1$ and $S_B = 0$, the m votes are independent and each equals B with probability at most $\frac{1}{2} - \delta$, $\delta > 0$. Then $\mathbb{E}[S_{\mathrm{CTS}}] \geq \mathbb{E}[S_A] - e^{-2m\delta^2}$. For a tournament chain over K candidates in which the*

incumbent is compared with each retry in turn, under the same assumption at every comparison, $\mathbb{E}[S_{\mathrm{CTS}}] \geq \mathbb{E}[S_A] - (K-1)\, e^{-2m\delta^2}$.

Proof. CTS changes the emitted outcome only when it swaps and $S_A \neq S_B$, so

$$\begin{aligned}\mathbb{E}[S_{\mathrm{CTS}}] - \mathbb{E}[S_A] &= \Pr[\text{swap}, S_A{=}0, S_B{=}1] - \Pr[\text{swap}, S_A{=}1, S_B{=}0] \\ &\geq -\Pr[S_A{=}1, S_B{=}0]\ \Pr[\text{swap} \mid S_A{=}1, S_B{=}0].\end{aligned}$$

Given $S_A = 1, S_B = 0$, the number X of B votes is a sum of m independent indicators with $\mathbb{E}[X] \leq m(\frac{1}{2} - \delta)$. A swap requires $X \geq (m+1)/2 > m/2$, hence $X - \mathbb{E}[X] > m\delta$, and Hoeffding's inequality gives $\Pr[\text{swap} \mid S_A{=}1, S_B{=}0] \leq e^{-2m\delta^2}$. Bounding $\Pr[S_A{=}1, S_B{=}0] \leq 1$ proves the pairwise claim. For the chain, let $S^{(c)}$ be the success of the incumbent after c comparisons; applying the pairwise bound to each comparison gives $\mathbb{E}[S^{(c)}] \geq \mathbb{E}[S^{(c-1)}] - e^{-2m\delta^2}$, and summing over the $K-1$ comparisons yields the result. □

The assumption constrains only votes against a successful incumbent; TIE votes count toward keeping it. The bound treats the m self-verifier samples, drawn at temperature 0.7 (Appendix A.3), as independent; with correlated votes it approaches the single-vote case $m{=}1$.

A.5 Worked Example: One Question's Lifecycle

We follow one question through the pipeline: *"Did the agent produce a non-empty final result before timing out?"*

Proposal. The question entered the WAI training bank with global scope ($*$) and positive polarity, so YES is evidence of progress. Like every new question, it carried zero weight until it had been observed.

Observation. At every URL state of every rollout, the base model answers it with the task adapter detached. Each answer is one YES/NO call on the same weights that serve the policy, with no external API call.

Certification. Its trust comes from measured separation, not from how it was proposed. On the transferred bank's calibration sample (60 judge-passed and 60 judge-failed OM2W trajectories), the verifier answers YES with probability 0.83 on passing and 0.22 on failing trajectories. That is a lift of $+0.62$, well above the discrimination floor, so the question certifies with trust weight 0.62. A question with lift near zero, such as the always-true and never-firing checks the certifier rejects, would keep weight 0.

Training use. Wherever the question is certified, a YES answer at URL state u adds its weight to $R^{VQ}(u)$ (Eq. 1). The one-sided blend $\beta \max(0, R^{VQ})$ raises that step's reward (Eq. 6), and the URL-stratified return turns it into an advantage on the tokens of that step (Eq. 7).

Test-time use. The frozen question contributes a record $(q, w, \mathrm{lift}, p(\mathrm{YES}))$ at each matching URL state of each candidate (Eq. 8). For example, a candidate that times out without an answer shows low $p(\mathrm{YES})$ on this certified question, and that is the evidence the comparative vote reads before deciding whether to swap (Eq. 9).

A.6 Bank Details and OM2W Transfer

This appendix tracks the construction, certification, and transfer of the verification banks that drive both training and test-time selection. For each benchmark we report the seed bank size, the size of the candidate bank after CEGAR-driven growth across training shards, the certified subset retained by the conformal certifier, and (for OM2W only) the subset that survives filtering and rewriting for live-web reuse. The same bank object is read by the training reward (§2.2) and by CTS at test time (§2.3); the numbers below describe how that object grows and shrinks at each stage.

WAI and VWA training banks. The WAI training bank starts with 281 seed verification questions derived from app schemas and user documentation; the VWA training bank starts with 89 seed questions from SynthAgent site exploration of the three Docker environments. Both banks grow through the CEGAR refinement loop, in which the judge proposes 3 to 5 new questions every

M=60 training steps targeting rollouts that the existing bank fails to discriminate, and shrink through certification: questions whose lift falls below 0.02 at all tiers, or whose polarity is contradicted by observed lift, are demoted to neutral or dropped. After cross-shard merging, the WAI candidate bank totals 992 unique questions, of which 44 retain non-zero trust weights at one or more URL tiers and form the certified WAI bank used by the training reward and by CTS at test time. The VWA candidate bank totals 1,565 unique questions with 21 certified at the end of training. The 36 WAI and 48 VWA questions reported in Table 7 are the portion of each bank that the filter/rewrite stage judged general enough to reuse on live-web OM2W: certified questions kept as-is (7 WAI, 10 VWA) plus rewritten variants of certified or promising uncertified questions (29 WAI, 38 VWA). The remainder stay in their original training banks because their URL patterns or DOM references are app- or environment-specific and do not transfer.

Representative certified questions. To make the bank concrete, we walk through six representative certified questions drawn from the merged WAI and VWA banks, using the same prompt-style format as the agent and judge templates above. Each block shows the question identifier, the URL pattern at which it applies (`*` = global; the bank also contains per-host and per-URL-path refinements such as `*google.com*` and `*reddit.com/r/*` that share the same text but track separately under Mondrian-ACI), the observed trust weight w (signed by polarity, magnitude scaled by lift), and the number of judge-scored observations n that supported certification, followed by the YES/NO question that is asked of every rollout at runtime.

Q.1: Reaching content-bearing pages (WAI, w=+0.54). The single strongest signal in the bank. Rollouts that visit a page containing task-relevant content are far more likely to receive a high judge score than rollouts that loiter on app homepages or navigation menus, and this pattern holds across all 9 WAI apps.

```
Identifier:  made_meaningful_progress_toward_goal
URL pattern:  *
Trust weight:  w = +0.54, observations:  n = 82.
Did the agent reach at least one page or state that contains information
directly relevant to answering or completing the task (not just the app
homepage or navigation menus)?
```

Q.2: Constructive non-destructive behaviour (WAI, w=+0.54). A two-part check that distinguishes "did nothing harmful" from "did something productive". Rollouts that take destructive shortcuts (delete, cancel, revoke) without a constructive counterpart fail this question even when the final URL looks plausible.

```
Identifier:  avoided_destructive_unintended_actions
URL pattern:  *
Trust weight:  w = +0.54, observations:  n = 82.
Did the agent both (a) avoid performing destructive actions (delete, cancel,
revoke) not required by the task AND (b) take at least one constructive
action toward the task goal?
```

Q.3: Source-grounded final answer (WAI, w=+0.27). Targets a common failure mode in information-retrieval tasks: emitting an answer without ever opening the page that actually contains the requested fact. Certified at high n, so the verifier has stable evidence at the global tier.

```
Identifier:  extracted_info_from_correct_page
URL pattern:  *
Trust weight:  w = +0.27, observations:  n = 520.
For information-retrieval tasks, did the agent extract its final answer from
the page that actually contains the requested information?
```

Table 7: **OM2W bank construction and recalibration.** Questions come from WAI/VWA banks plus an OM2W-specific failure-mode expansion.

Origin	Stage	Questions	Certified	Mean lift
WAI bank	kept as-is	7	7	0.46
WAI bank	rewritten	29	26	0.46
VWA bank	kept as-is	10	4	0.31
VWA bank	rewritten	38	27	0.28
OM2W expansion	live-web checks	48	36	0.30
Total	–	**132**	**100**	**0.34**

Q.4: End-state coherence (WAI, w=+0.26). A simple URL coherence check that catches "wandered off" rollouts in which the agent ends on an unrelated section of the app. Cheap to answer for the base verifier (it only needs the final URL and the task description) and stable across apps.

```
Identifier:  final_page_matches_task_context
URL pattern:  *
Trust weight:  w = +0.26, observations:  n = 238.
Is the final URL or page the agent ended on relevant to the task objective
rather than an unrelated section of the app?
```

Q.5: Active search behaviour (VWA, w=+0.20). A VWA-specific predicate. Distinguishes agents that submit a query and then act on the results from agents that issue a search and freeze on the results page without clicking through. Especially discriminative on the Classifieds and Shopping environments.

```
Identifier:  used_search_and_acted_on_results
URL pattern:  *
Trust weight:  w = +0.20, observations:  n = 66.
Did the agent both submit a search query AND subsequently take a meaningful
action on the results (e.g., clicking a result link rather than just
observing the search page)?
```

Q.6: Navigation depth (VWA, w=+0.13). Many VWA tasks require going past the search-results page to a specific listing or product detail page to extract a price, image, or description. This question rewards depth of navigation rather than breadth and is the highest-trust VWA question whose URL pattern is already general enough to translate to live-web OM2W without rewriting.

```
Identifier:  visited_target_detail_page
URL pattern:  *
Trust weight:  w = +0.13, observations:  n = 200.
Did the agent navigate to a specific item detail page (product page, listing
page, or submission page) relevant to the task, beyond the search results
page?
```

These six illustrate three properties the certifier rewards: the question must discriminate between high- and low-judge rollouts (positive lift), it must be answerable from the trajectory by the base self-verifier (not too low-level for the LoRA-detached Gemma-4 to read off), and its YES/NO must align with the judge's separation under the chosen polarity. Questions that fail these tests, e.g., ones that ask about UI elements specific to a single page or that the base model cannot reliably answer, are demoted to neutral or dropped by the polarity-veto rule of Eq. 5.

OM2W transfer setting. The main text reports the compact bank-transfer table. Here we keep the more detailed accounting for how the OM2W bank was produced and why it is a strict transfer setting. No reported policy is trained on OM2W; only the question bank is filtered, rewritten, expanded, and recalibrated.

Transferred question taxonomy. The filter/rewrite stage keeps general web-interaction predicates where possible and removes benchmark-specific app names or URL patterns when they do not transfer. The expansion stage adds questions for live-web failure modes observed on OM2W, including bot-block pages, timeout or wandering behaviour, hallucinated extraction, partial-filter application,

Table 8: **OM2W transfer performance details.** K is the rollout budget and every row emits one selected trajectory. Canonical judge is WebJudge [Xue et al., 2025] + o4-mini at threshold 4; Sonnet is a cross-check.

System	K	Canonical SR	Gain	Sonnet SR
Gemma-4 base	1	40.0	–	46.8
Gemma-4 + ours	2	45.0	+5.0	partial positive
GPT-5.5 low base	1	43.7	–	47.3
GPT-5.5 high base	1	49.7	–	52.0
GPT-5.5 high + ours	2	57.3	+7.7	55.7
GPT-5.5 high + ours	4	61.0	+11.3	61.7

search-vs-detail-page confusion, and wrong-final-answer stops. The resulting 132-question bank uses the same schema as the training-time bank, so CTS loads it without code changes.

Recalibration. Each question is scored on a stratified pass/fail rollout sample using the base Gemma-4 verifier with logprob-extracted YES probabilities. Lift is computed as the pass/fail separation of the YES probability. Questions whose absolute lift is below the discrimination floor remain in the bank with zero weight; they can be re-certified later on a different rollout sample. Among the 100 certified OM2W questions, 75 have positive lift and 25 have negative lift, so the selector receives both progress evidence and failure-mode evidence.

Multiple-testing control. Certification decides each question separately, so a bank of many questions could admit some by chance. We therefore check the transferred bank against Benjamini–Hochberg (BH) control. Each question is scored on 60 passing and 60 failing trajectories, and YES-probabilities lie in $[0, 1]$, so the two-sided Hoeffding inequality gives a conservative two-sided p-value $p_q = \min\{1,\, 2\exp(-2\,\mathrm{lift}_q^2/(1/n_{\mathrm{pass}} + 1/n_{\mathrm{fail}}))\}$ without distributional assumptions beyond independent trajectories. We run the check on a second build of the transferred bank made with the same procedure (181 questions, 144 certified; Table 7 describes the 132-question build behind the main results): BH at false-discovery rate 0.10 retains 102 certified questions (98 at 0.05), and no uncertified question passes; the stricter family-wise Holm correction at 0.05 retains 83, again with no uncertified question. Every BH discovery has $|\mathrm{lift}| \geq 0.25$. Because the Hoeffding bound is conservative, these counts are lower bounds on the questions that survive simultaneous control.

Sensitivity to the CEGAR proposer. Because uncertified questions carry zero weight, the proposer can change how many questions earn trust but not how much an uncertified one counts. We test this directly with our CEGAR refinement prompt. Three proposers receive the same eight prompts, each built from judge-scored VWA training trajectories and their judge rationales (training steps 20–60). The proposers are a frontier model (Claude Opus 4.6) and two cheaper open models, Gemma-4-31B and Qwen3.5-4B. Every proposal is then scored identically by the base verifier on 300 held-out trajectories from training steps 80–120, and certified by $|\mathrm{lift}| > 0.02$ with BH at 0.10. The three proposers certify 32.5% (13/40), 30.3% (10/33) and 29.2% (7/24) of their proposals, with mean certified $|\mathrm{lift}|$ of 0.37, 0.46 and 0.42. A cheaper proposer therefore yields fewer usable questions, mainly because the smallest model fails to return well-formed proposals in three of eight rounds, but the questions it does propose certify at the same rate and carry comparable evidence.

A.7 OM2W Transfer Performance Details

Table 8 reports the detailed cross-judge performance numbers behind the compact main-text OM2W figure. It is intentionally separate from Table 7: the bank table explains how the question set was transferred and recalibrated, while this table measures how much realised success that transferred evidence adds after CTS selects one trajectory to emit.

The canonical column is the number used in the main paper, since it matches the OM2W judging protocol. The Sonnet column is included only as an external consistency check. It follows the same ordering at the important operating points: the transferred selector improves the Gemma-4 backbone, and the GPT-5.5 high-effort setting gains further as the rollout budget increases from $K{=}2$ to $K{=}4$. Rows with $K{=}1$ are ordinary single rollouts; rows with larger K are CTS-selected outputs, not oracle unions over multiple attempts.

Calibration sample and held-out tasks. The recalibration sample consists of 120 judge-labelled rollouts (60 successful, 60 failed) drawn from the base GPT-5.5 high-effort run on the 300 evaluation tasks, so those 120 tasks contribute labels to the bank's trust weights; no policy is trained on OM2W. Table 9 therefore also reports every arm on the 180 tasks outside the calibration sample, scored with the same canonical judge. The gain is larger there than on the full set, and it is smallest on the calibration subset, where the base rate is highest and the selector has the least headroom. The $K{=}4$ rows are a cumulative chain that adjudicates the $K{=}3$ selection against a further round of rollouts, so not every task receives a fourth candidate.

Table 9: **OM2W success rates on the calibration and held-out task subsets.** Canonical judge (WebJudge + o4-mini, threshold 4). W/L counts held-out tasks that CTS turns from failure to success and vice versa; p is the exact paired McNemar test on the 180 held-out tasks.

System	K	All (300)	Calib. (120)	Held-out (180)	W/L	p
Gemma-4 base	1	40.0	46.7	35.6	–	–
Gemma-4 + ours	2	45.0	47.5	43.3	20/6	$9.4{\times}10^{-3}$
GPT-5.5 high base	1	49.7	55.8	45.6	–	–
GPT-5.5 high + ours	2	57.3	60.8	55.0	20/3	$4.9{\times}10^{-4}$
GPT-5.5 high + ours	4	61.0	62.5	60.0	30/4	$6.2{\times}10^{-6}$

A.8 Selection Controls on Identical Candidates

A selector can only be credited for choosing well if the comparison holds the candidates fixed. Table 10 does so on VWA with $K{=}4$ completed GPT-5.5 rollouts per task (two greedy and two diverse); every row picks one of the same four trajectories. Classifieds and Reddit use the candidate sets behind Table 4; Shopping uses a separate 362-task subset with its own candidates. *Anchor* keeps the first greedy rollout; *random* is the expected success of a uniform pick; *vote, no bank* asks GPT-5.5 to compare the same candidates with the same three-vote majority and anchor default, but without bank evidence; *oracle* succeeds if any candidate does.

Random selection gains nothing over the anchor (45.2 vs. 45.4), so the CTS gain comes from informed choice, not from extra sampling; tested directly against the exact distribution of a uniform pick over the same candidates, CTS is 3.3 standard deviations above it (one-sided $p = 5.5{\times}10^{-4}$). Pooled over 806 tasks, CTS improves on the anchor by 2.2 points (26 tasks gained, 8 lost; exact McNemar $p = 0.003$) and recovers about a fifth of the anchor-to-oracle headroom. Against the bank-free GPT-5.5 vote on the 444 tasks where it was run, no difference is detectable (56.5 vs. 55.4; paired difference $+1.1$ points, 95% CI $[-1.2, +3.5]$, McNemar $p = 0.46$); two one-sided tests bound the difference within ± 3.1 points. CTS reaches this using an open self-verifier rather than a frontier API model.

Table 10: **Selection on identical VWA candidate sets ($K{=}4$).** Success rates in percent; pooled rates carry Wilson 95% intervals. W/L and p compare CTS with the anchor on the same tasks (exact McNemar).

Site	n	Anchor	Random	Vote, no bank	CTS	Oracle	W/L	p
Classifieds	234	60.7	59.7	60.7	62.4	70.5	8/4	0.39
Reddit	210	47.1	46.8	49.5	50.0	58.6	8/2	0.11
Shopping	362	34.5	34.8	–	36.7	45.3	10/2	0.039
Pooled	806	45.4 [42.0, 48.9]	45.2	–	47.6 [44.2, 51.1]	56.1 [52.6, 59.5]	26/8	0.003

Test-time cost and vote agreement. Table 11 measures what one CTS decision costs. We run the selector, configured as in the main experiments, on 200 WAI candidate pairs: bank evidence from at most 8 questions per URL state, then $m{=}3$ comparative votes at temperature 0.7. The verifier is base Gemma-4-31B served with vLLM at tensor parallelism 2 on H200 GPUs, and the bank is the transferred bank of Appendix A.6. The pairs come from two independent runs of an open browser-use policy on the same tasks. Cost depends on trajectory length and bank size rather than on which policy produced the trajectories. Bank evidence is many short calls with one-token answers and takes about half a second; the three votes dominate latency. All of it runs on the hardware that already serves

the policy and makes no external API call. The votes are unanimous in 90.5% of decisions, and the first vote alone already gives the three-vote decision in 97% of them, so $m{=}3$ changes about 3% of decisions: inexpensive insurance against verifier variance rather than the source of the gain.

Table 11: **Cost of one CTS decision** (200 WAI candidate pairs, two trajectories each). Median and 90th percentile.

Component	Median	p90
Bank evidence: verifier calls	48	144
Bank evidence: prompt tokens	16.3k	48.4k
Votes ($m{=}3$): prompt / completion tokens	8.6k / 0.8k	13.9k / 1.0k
Wall-clock: evidence / votes / total (s)	0.5 / 9.3 / 10.1	1.8 / 13.7 / 14.8

A.9 Case Studies

Case A.1: CTS save on WAI Gmail. The selector replaces a long failing rollout with a shorter successful one on the same task.

```
Task: Change profile name to "Alex J. Johnson"; create label "Priority"
with color #EA4335; apply it to all contacts in the Family label.
Greedy trajectory: 87 browser steps; final verifier label = fail.
Diverse trajectory: 25 browser steps; changed profile name, created the
Priority label, applied it to Laura Johnson-Martinez, Leo Martinez, Margaret
Johnson, and Richard Johnson; final verifier label = pass.
CTS decision: swap=True, greedy score=-0.5, diverse score=+0.5,
chosen_success=True.
```

Case A.2: CTS save on a long WAI Handshake chain. The selected rollout completes a multi-hop message, event, and job chain that the greedy rollout leaves unfinished.

```
Task: Read the latest unread message, RSVP to that sender's upcoming event,
and save all active jobs from that sender.
Greedy trajectory: failed to complete the full message/event/job chain.
Diverse trajectory: navigated through messages, identified the sender's
event, RSVP'd, and saved the relevant active jobs.
CTS decision: swap=True, greedy score=-0.5, diverse score=+0.5,
chosen_success=True.
```

Case A.3: No-regression behaviour on WAI Gmail. CTS can also decline a replacement when the greedy rollout is already the better candidate.

```
Task: Disable two-factor authentication.
Greedy trajectory: completed the account-security change successfully.
Diverse trajectory: failed the task.
CTS decision: swap=False; the selector kept the greedy rollout rather than
replacing a working trajectory with a worse sample.
```

Case A.4: Live-web recovery on OM2W Indeed. On a live site, the successful trajectory recovers from blank results and an additional-verification page by returning to the native search flow.

```
Task: Find support-services jobs in Bentonville, Arkansas.
Trajectory snippet: navigate to Indeed; enter "support services" and
"Bentonville, AR"; direct results URL first renders blank; alternate URL
hits an additional-verification page; agent returns home, uses the native
search form, waits for results, and stops with visible listings.
Final answer: includes Healthcare Driver in Bentonville and other
support-services listings; WebJudge label = success.
```

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and §1 state the method contribution and three empirical claims: WAI training, VWA transfer to a closed backbone, and OM2W zero-shot transfer. The corresponding results are reported in §4.

2. **Limitations**

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

   Answer: [Yes]

   Justification: §6 discusses the need for repeated rollouts from the same state, which live websites with irreversible actions may not allow, reliance on one trained open backbone and a frontier judge, limits of a single transferred bank, and misuse risk. The assumptions behind the CTS bound are stated in Appendix A.4.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The only formal result is the CTS conservativeness bound (Eq. 10). Its assumptions are stated in §2.3, and Proposition 1 with a complete proof is in Appendix A.4. The certification rule (Eqs. 3–4) is a definition, not a claimed guarantee.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: §3 gives the benchmark roles, task synthesis summary, and evaluation protocols; §2 specifies reward and selection mechanics; and Appendix A.2 gives CLIFT training details.

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

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

   Answer: [No]

   Justification: Company policy does not permit releasing the code, synthesized task pools, question banks, or trained adapters. To support reproduction, the appendix documents the agent and judge prompts (Appendix A.1), the training setup and task-synthesis pipeline (Appendix A.2), all hyperparameters (Appendix A.3), and bank construction with representative questions (Appendix A.6). All evaluation benchmarks are public.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Method details are in §2; benchmark protocols are in §3; training configuration and prompt templates are summarized in Appendix A.1 and Appendix A.2.

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Results are success rates on fixed benchmark task sets, with task counts and weighted averages reported in the tables. For CTS, §4.4 reports per-app gains, ties/regressions, average solve steps, and swap rate, and Appendices A.7 and A.8 report Wilson 95% intervals and exact paired McNemar tests for the test-time selection results.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix A.2 specifies the Gemma-4-31B + LoRA setup, rollout group size, sharding scheme, peak 8-node 8×H200 training allocation, closed GPT-5.5 API inference usage, judge usage, and verifier usage.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: Training uses synthesized tasks and sandboxed or benchmark environments where possible. The live-web transfer setting is evaluated through benchmark tasks, and limitations and misuse risks are discussed in §6.

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: §5 and §6 note both improved web-agent reliability and misuse risks from stronger autonomous web agents.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse?

    Answer: [N/A]

    Justification: No data, code, or models are released. §6 still recommends rate limits and domain-specific safeguards for deploying stronger web agents.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The benchmarks are cited where used (WebArena-Infinity, VisualWebArena, Online-Mind2Web, and the WebArena family). Open software and models are named where used and were used under their licenses: vLLM, TRL, Gemma-4-31B-it, and Qwen3.5-4B (Apache-2.0); browser-use (MIT); WebArena (Apache-2.0); VisualWebArena and WebArena-Infinity (MIT); Online-Mind2Web (code MIT, data CC BY 4.0). Closed models (GPT-5.5, Claude Opus 4.6, o4-mini) were accessed through their providers' APIs under the providers' terms.

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets are released. The internal task pools and question banks are described in Appendices A.2 and A.6.

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

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

    Answer: [N/A]

    Justification: No crowdsourcing or human-subject study is used.

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

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

    Answer: [N/A]

    Justification: No human-subject research is conducted.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

    Answer: [Yes]

    Justification: §2 describes the training-time comparative judge, base-model verification questions, CEGAR question generation, and the test-time CTS selector; the appendix includes rollout and judge prompt templates.