WebArbiter: A Principle-Guided Reasoning Process Reward Model for Web Agents

Anonymous submission

Abstract

Web agents hold great potential for automating complex computer tasks, yet their interactions involve long horizons, multi-step decisions, and actions that can be irreversible. In such settings, outcome-based supervision is sparse and delayed, often rewarding incorrect trajectories and failing to support inference-time scaling. This motivates the use of Process Reward Models (WebPRMs) for web navigation, but existing approaches remain limited: scalar WebPRMs collapse progress into coarse, weakly grounded signals, while checklist-based WebPRMs rely on brittle template matching that fails under layout or semantic changes and often mislabels superficially correct actions as successful, providing little insight or interpretability. To address these challenges, we introduce **WebArbiter**, a reasoning-first, principle-inducing WebPRM that formulates reward modeling as text generation, producing structured justifications that conclude with a preference verdict and identify the action most conducive to task completion under the current context. Training follows a two-stage pipeline: reasoning distillation equips the model with coherent principle-guided reasoning, and reinforcement learning corrects teacher biases by directly aligning verdicts with correctness, enabling stronger generalization. To support systematic evaluation, we release WEBPRMBENCH, a comprehensive benchmark spanning four diverse web environments with rich tasks and high-quality preference annotations. On WEBPRMBENCH, WebArbiter-7B outperforms the strongest baseline, Gemini Flash, by 10.9%. In reward-guided trajectory search on WebArena-Lite, it surpasses the best prior WebPRM by up to 7.2%, underscoring its robustness and practical value in real-world complex web tasks.

1 Introduction

Large Language Models (LLMs) (Achiam et al. 2023; Guo et al. 2025a) have demonstrated impressive capabilities in planning (Huang et al. 2024; Zhang et al. 2025a), decision-making (Li et al. 2024), and complex task execution (Xi et al. 2024; Zhang et al. 2025b). Extending these abilities with browser access enables LLM agents to perform complex web tasks similar to humans (OpenAI 2025b; Anthropic 2024; Adept 2022). However, web interactions involve long horizons, multi-step decisions, and actions that can be irreversible. For example, submitting an incorrect form may not be recoverable. This requires agents to make reliable decisions throughout the interaction process, rather than relying solely on final outcomes. Traditional Outcome Reward Models (ORMs) are ill-suited: they provide only sparse and delayed feedback, may misclassify incorrect trajectories as successes, and cannot guide inference-time strategies, such as reward-guided search.

Recent studies on web agents (Zhang et al. 2025b; Koh et al. 2025) have introduced step-level rewards using LLM-as-judge. While such supervision can be useful, LLM-as-judge suffers from high cost, limited scalability, and susceptibility to hallucination, often rewarding fluent but incorrect actions. This motivates the development of dedicated Process Reward Models (WebPRMs) for web tasks. Existing WebPRMs largely fall into two categories: scalar WebPRM (Miao et al. 2025), which collapse progress into coarse scores with little interpretability or weak grounding; and generative WebPRM (Chae et al. 2025), which rely on checklists that are brittle under dynamic layouts and shifting semantics. Moreover, lacking explicit reasoning, generative WebPRMs remain vulnerable to surface correlations and sensitive to page changes. These limitations highlight the need for a reasoning-first WebPRM that can verify progress, resist superficial biases, and provide interpretable chains for diagnosing errors.

To this end, we propose **WebArbiter**, a reasoning-first, principle-inducing WebPRM. It formulates process reward modeling as text generation: given task context and candidate actions with their reasoning traces, the model produces a structured justification that concludes with a preference verdict, identifying the action most conducive to task completion. Unlike scalar scores or checklist-based methods tied to fixed templates, WebArbiter dynamically derives principles from user intent and the current state, incorporates them into reasoning chains that verify whether an action advances task completion. Training follows a two-stage pipeline: reasoning distillation equips the model with coherent principle-guided reasoning, and reinforcement learning corrects teacher biases and aligns verdicts with correctness. This design transforms reward signals from shallow correlations into auditable analyses, making judgments robust to environment and page variations, resistant to spurious cues, and accurate in credit assignment.

To advance the evaluation of WebPRMs, we introduce WEBPRMBENCH, the first comprehensive evaluation benchmark spanning diverse environments dedicated to WebPRMs. It provides 1,287 step-level preference instances, each

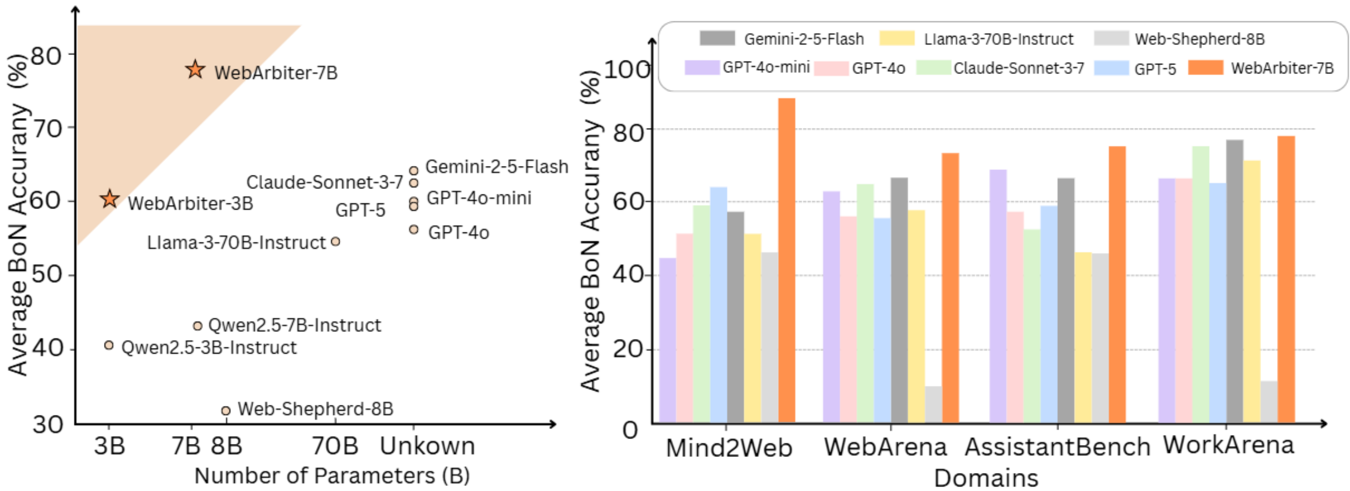


Figure 1: Performance comparison on WEBPRMBENCH. **Left:** *Average Best-of-N Acc* vs. model size, showing superior efficiency despite smaller scale. **Right:** Domain-wise *Avg BoN Acc*, where WEBARBITER achieves the best results across all environments, confirming robustness and scalability.

consisting of one correct action and four rejected alternatives, collected across 4 web environments: AssistantBench (Yoran et al. 2024), Mind2Web (Deng et al. 2023), WorkArena (Drouin et al. 2024; Boisvert et al. 2025), and WebArena (Zhou et al. 2023). The tasks span everyday activities such as online shopping and forum posting, as well as enterprise scenarios like updating schedules in IT management platforms. By combining scale, diversity, and fine-grained supervision, WEBPRMBENCH establishes a unified standard for systematic evaluation of WebPRMs, with *Pairwise* and *Best-of-N (BoN) Accuracy* as the primary metrics.

Extensive experiments on WEBPRMBENCH show that WebArbiter achieves SOTA *Avg. BoN Acc*, consistently surpassing both proprietary LLMs and previous SOTA WebPRM, WebShepherd, across all environments, and outperforming the strongest LLM baseline, Gemini Flash, by +10.9%. Beyond static evaluation, WebArbiter also proves effective in practice: in reward-guided trajectory search on WebArena-Lite (Liu et al. 2024b), it delivers substantial gains, surpassing WebShepherd by up to 7.2%, further demonstrating robustness in realistic interaction settings.

The key contributions of this work are:

1. We propose WebArbiter, a reasoning-first, principle-inducing PRM trained with reasoning distillation and reinforcement learning, providing auditable reasoning chains and correctness-aligned signals.
2. We release WEBPRMBENCH, the first comprehensive evaluation benchmark to provide systematic WebPRM evaluation across 4 web environments, using *Pairwise* and *Best-of-N (BoN) Accuracy* as standard metrics.
3. We show that WebArbiter achieves SOTA performance on WEBPRMBENCH, surpassing both proprietary LLMs and the previous SOTA WebPRM. WebArbiter delivers up to +7.2% gains in reward-guided trajectory search on WebArena-Lite.
4. We analyze the training dynamics of WebArbiter, revealing how different strategies influence performance.

2 Related Work

2.1 LLM-Based Autonomous Web Agents

LLM advances have enabled browser-operating agents (Kim, Baldi, and McAleer 2024; Sun et al. 2024; Prasad et al. 2023; Fu et al. 2024; Ma et al. 2023; Zheng et al. 2023b; Tao et al. 2023). One line distills environment-specific state–action pairs from demonstrations, strong on seen states yet brittle on novel ones, with SteP as a leading example on WebArena (Sodhi et al. 2024; Zhou et al. 2023). A second line pursues open-ended exploration via reflexive evaluation and search (Pan et al. 2024; Shinn et al. 2024; Koh et al. 2024; Zhang et al. 2025b). A third direction applies reinforcement learning (Qi et al. 2025; Wei et al. 2025), yet real sites provide sparse and delayed signals, which makes value learning unstable without dense step feedback. Therefore, WebAgents require a process-level judge that assesses progress step by step and supplies auditable signals for search and planning.

2.2 Reward Models in Reasoning and Web Tasks

RMs fall into two families. Scalar RMs attach a single numeric score to a response with a linear scorer and use either absolute or discriminative schemes for evaluation (Uesato et al. 2022; Ouyang et al. 2022; Liu et al. 2024a, 2025; Park et al. 2024; Wang et al. 2024a, 2023b, 2024b). Generative RMs instead produce natural–language feedback from which rewards are extracted, aligning with LLM-as-Judge and supporting both single-instance evaluation and multi-response comparison; they show promising scalability but raise reliability concerns due to bias and hallucination (Lightman et al. 2023; Wang et al. 2023a; Zhang et al. 2025c; Wu et al. 2024; Ye et al. 2025; Zhang et al. 2024; Zheng et al. 2023a).

Building on these, Reasoning RMs cast judging as a deliberate process: they first generate an explicit, context-grounded chain of principle and analysis, then issue a single preference verdict, yielding adaptive test-time compute, stronger grounding, and interpretable feedback (Chen et al. 2025; Guo et al. 2025b; Mahan et al. 2024). In web agents, action rewards have been derived by the following methods: LLM-as-Judge (Zhang et al. 2025b; Koh et al. 2025), slow and unstable during search; scalar scoring (Miao et al. 2025), which collapses progress into coarse values with little interpretability and weak grounding; and checklist-driven generative feedback (Chae et al. 2025), whose external templates are brittle under layout and semantic drift and prone to surface correlations. These limitations motivate a reasoning-first approach that turns rewards from shallow correlations into auditable analyses. WebArbiter produces structured justifications with a single preference verdict, induces principles from the current instruction and state, and is trained by reasoning distillation followed by reinforcement learning, so that judgments remain robust to environment variations, resist spurious cues, and provide accurate credit assignment while supporting inference-time scaling.

3 Methodology

In this section, we present the design of WebArbiter. We begin by framing web navigation as a Partially Observable Markov Decision Process (POMDP) in §3.1, then describe how we construct a pairwise-preference dataset for training in §3.2. We introduce the training pipeline of WebArbiter model in §3.3. For clarity, we summarize all notations in Appendix A.

3.1 Background

We formalize web navigation as a POMDP. The environment $\mathcal{E}$ is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, and an observation space $\mathcal{O}$. $T : \mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$ denotes the state transition function. At step p, the agent receives a partial observation $o_p\in\mathcal{O}$, executes $a_p\in\mathcal{A}$, and transitions to $s_{p+1} = T(s_p, a_p)$ with a new observation o_{p+1}. Following WebArena (Zhou et al. 2024), we represent observations using accessibility trees, i.e., text-only encodings of visible interactive elements and their attributes. Given a task instruction $\mathcal{I}$ and the initial state $s_0 \in \mathcal{S}$, the agent aims to generate a trajectory $\tau = (a_1, \ldots, a_P)$ that completes the task. Here P is the trajectory length and $a_p \in \mathcal{A}$ denotes the action at step p. The task evaluator determines whether the task is completed based on the final state.

3.2 Training Dataset Construction

We build on the WEBPRM COLLECTION (Chae et al. 2025) for training WebArbiter. Each instance consists of an instruction $\mathcal{I}$, a sequence of observations $O = (o_1, \ldots, o_P)$, and expert-annotated trajectories. Specifically, the dataset provides a set of positive actions $A^+ = (a^+_1, \ldots, a^+_P)$ taken from expert demonstrations and negative actions $A^- = (a^-_1, \ldots, a^-_P)$ obtained from rejected trajectories. We convert these into pairwise preference samples where each candidate action is paired with its reasoning trace, yielding the preference dataset $\mathcal{D}_{\text{Train}}$ used for WebArbiter training.

3.3 WebArbiter: a Principle-Inducing Reasoning Process Reward Model

WebArbiter is built on a Transformer-decoder backbone and formulates process reward modeling as a text generation task. At each state, it evaluates candidate actions $\{(a^q_p, c^q_p)\}^Q_{q=1}$, where each action a^q_p is paired with a reasoning trace c^q_p explaining why the agent generated this action. Given task instruction $\mathcal{I}$, observation o_p, and history $(a_{<p}, c_{<p})$, the model autoregressively generates a structured justification $j = (j_1, \ldots, j_L)$ of length L that concludes with a preference verdict $\hat{y}$ selecting the most appropriate action among the candidates. The historical traces are $c_{<p} = \{c_1, \ldots, c_{p-1}\}$, i.e., the per-action reasoning traces for previously executed actions. A concrete training example is provided in Appendix B. While our experiments instantiate this framework in the standard pairwise preference setting, the design is general and extends naturally to multi-candidate.

Unlike the scalar WebPRM (Miao et al. 2025) that collapses progress into opaque scores or the checklist-based WebPRM (Chae et al. 2025), WebArbiter is a reasoning-first, principle-inducing WebPRM: it dynamically derives principles from user intent and the current state, integrates them into reasoning chains that explicitly assess whether each candidate action truly advances task completion. This design moves reward signals beyond shallow correlations toward auditable analyses, yielding judgments that are robust to environment changes, resistant to spurious cues, and precise in credit assignment.

Formally, the preference dataset is defined as

$$\mathcal{D}_{\text{Train}} = \{(\mathcal{I}^{(i)}, o^{(i)}_p, a^{(i)}_{<p}, c^{(i)}_{<p}, (a^{1(i)}_p, c^{1(i)}_p), (a^{2(i)}_p, c^{2(i)}_p), y^{(i)})\}^M_{i=1}, \quad (1)$$

where $y \in \{a^1_p, a^2_p\}$ denotes the preferred action. For notational simplicity, let

$$x = (\mathcal{I},\ o_p,\ a_{<p},\ c_{<p},\ (a^1_p, c^1_p),\ (a^2_p, c^2_p)). \quad (2)$$

WebArbiter π_θ is parameterized by θ and models the justification j autoregressively:

$$\pi_\theta(j \mid x) = \prod_{l=1}^{L} \pi_\theta(j_l \mid x, j_{<l}). \quad (3)$$

Training Overview. The overall training objective is to maximize the likelihood that the predicted preference matches the ground truth:

$$\max_{\pi_\theta}\ \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{Train}},\ \hat{y}\sim\pi_\theta(j|x)} \left[\mathbb{1}(\hat{y} = y)\right]. \quad (4)$$

Training proceeds in two stages. The first stage, described in §3.3, is reasoning distillation, which equips the model with the ability to generate coherent principle-guided justifications. This stage encourages judgments to be grounded in explicit reasoning rather than surface correlations, as we later validate through ablation studies in §5.1.

Concretely, we sample K examples from $\mathcal{D}_{\text{Train}}$ to form $\mathcal{D}_{\text{SFT}}$ for supervised distillation, while the remaining data is used as $\mathcal{D}_{\text{RL}}$ for reinforcement learning. The second stage, detailed in §3.3, is reinforcement learning, which aligns the

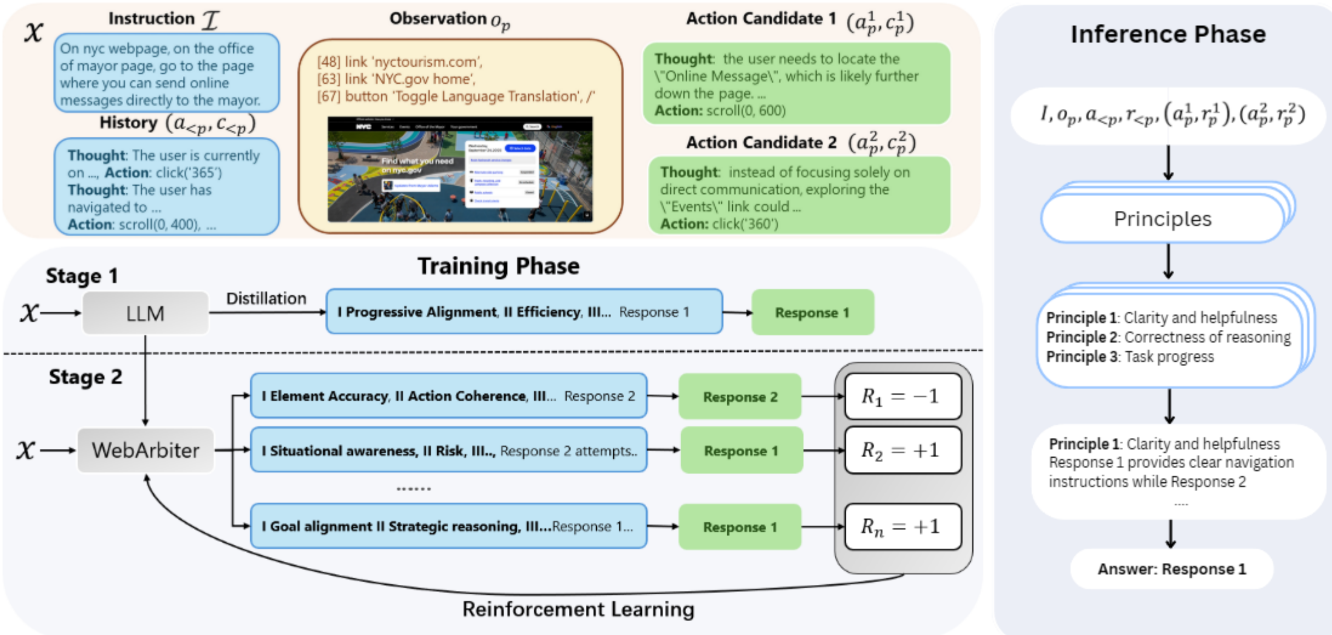


Figure 2: Overview of WebArbiter. Given an instruction $\mathcal{I}$, current observation o_p, and history $(a_{<p}, c_{<p})$, the model compares candidate actions (a_p^1, c_p^1) and (a_p^2, c_p^2). In **Stage 1**, principle-guided reasoning traces are distilled from a stronger teacher LLM. In **Stage 2**, WEBARBITER is trained with reinforcement learning using verifiable rewards $R \in \{-1, +1\}$, producing structured justifications and a final verdict. During inference, the model induces principles (e.g., clarity, correctness, progress) from $(\mathcal{I}, o_p, a_{<p}, c_{<p}, (a_p^1, r_p^1), (a_p^2, r_p^2))$, applies them to candidate actions, and outputs an auditable judgment identifying the action that best advances task completion.

verdicts with correctness signals and produces interpretable step-level rewards for long-horizon tasks. Together, these stages enable WebArbiter to deliver robust, interpretable, and scalable supervision for web agents.

Stage 1: Reasoning Distillation. Directly prompting an instruction-tuned LLM as a reward model often yields superficial, inconsistent chains that do not justify why an action advances the task. We therefore distill principle-guided reasoning from a stronger teacher. Concretely, o3 synthesizes structured justifications that first derive task-specific principles from the instruction and state, then ground these principles in the page, compare candidate actions against them, and finally output the preferred action. This equips WebArbiter with principles rather than surface heuristics. Ablations in §5.1 show that removing explicit principles and relying solely on reasoning-based justifications notably degrades performance, highlighting the role of principle induction in stabilizing step-level judgments. Given $(x^{(i)}, y^{(i)}) \in \mathcal{D}_{\text{SFT}}$, the teacher generates a justification $\hat{j}^{(i)} = (\hat{j}_1^{(i)}, \ldots, \hat{j}_{L_i}^{(i)})$. The distillation dataset is then: $\mathcal{D}_{\text{SFT}} = \{x^{(i)}, \hat{j}^{(i)})\}_{i=1}^{K}$.

Objective. Reasoning distillation adjusts θ to maximize the likelihood of generating the teacher justification $\hat{j}$ that concludes with the preferred action y given x. We minimize the standard negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{K} \sum_{i=1}^{K} \sum_{l=1}^{L_i} \log \pi_\theta\left(\hat{j}_l^{(i)} \mid x^{(i)}, \hat{j}_{<l}^{(i)}\right). \quad (5)$$

Stage 2: Reinforcement Learning. While distillation provides initial reasoning ability, it inherits teacher biases and may overfit to superficial patterns, limiting generalization to unseen environments. To further enhance judgment accuracy, stability, and generalization, we introduce a reinforcement learning stage. WebArbiter π_θ is treated as a policy that outputs a justification j that concludes with a final verdict $\hat{y}$. During rollout, π_θ generates the full justification and verdict, after which a correctness reward $R(x, \hat{y}) \in \{-1, 1\}$ is assigned solely based on whether $\hat{y}$ matches the ground-truth preference y. The distilled model from §3.3 serves as the reference policy π_{ref}, ensuring stable optimization.

Objective. Reinforcement learning adjusts θ to maximize the expected reward while stabilizing reasoning style via KL regularization. The optimization objective is defined as:

$$\begin{aligned}\mathcal{L}_{\text{RL}}(\theta) = \max_{\pi_\theta} \; & \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{RL}},\, \hat{y}\sim\pi_\theta(j|x)}\left[R(x, \hat{y})\right] \\ & - \beta\, \mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}).\end{aligned} \quad (6)$$

In practice, we adopt Group Relative Policy Optimization (GRPO) (Shao et al. 2024) to optimize this objective, which enables stable updates under binary verifiable rewards. Through this reinforcement learning stage, WebArbiter directly aligns its verdicts with correctness signals and converts structured justifications into reliable, interpretable step-level reward signals.

4 WEBPRMBENCH

In this section, we introduce WEBPRMBENCH, the first comprehensive evaluation benchmark spanning diverse environments for WebPRMs. Details of dataset construction and the evaluation protocol are provided below.

4.1 Benchmark Construction

WEBPRMBENCH is constructed from sucessful trajectories in AGENTREWARDBENCH (Lù et al. 2025), expanding beyond WEBREWARDBENCH (Chae et al. 2025), which only provides Mind2Web (Deng et al. 2023) and limited WebArena data (Zhou et al. 2023). We enrich WebArena with additional trajectories and incorporate AssistantBench (Yoran et al. 2024) and WorkArena (Drouin et al. 2024; Boisvert et al. 2025), resulting in broader coverage of real-world tasks across four domains: Mind2Web, WebArena, AssistantBench, and WorkArena. Mind2Web emphasizes cross-task generalization across heterogeneous websites. WebArena provides controlled environments such as shopping, CMS, forums, and GitLab. AssistantBench introduces open-world tasks on real websites. WorkArena focuses on enterprise workflows, including IT and HR. This diversity enables systematic evaluation across both consumer-facing and enterprise scenarios, while covering different levels of control, openness, and task complexity.

For each state, the action from the successful trajectory is retained as the positive label, and four rejected alternatives with associated reasoning traces are synthesized to form preference pairs. To ensure data quality, we sample negatives from diverse policy models to broaden coverage, apply rule-based filters to remove invalid or mismatched actions, discard inconsistent cases, and conduct expert verification to further ensure reliability. We also conduct targeted auditing to eliminate potential false negatives. Reasoning traces are truncated to a fixed length to minimize formatting noise. The resulting benchmark spans 1,287 preference pairs across four environments, as shown in Tab. 1.

4.2 Evaluation Protocol

Evaluating WebPRMs requires metrics that capture both local preference fidelity and global decision reliability under realistic multi-candidate settings. Inspired by RMB (Zhou et al. 2025), we adopt two complementary metrics: *Pairwise Accuracy*, which measures correctness on individual preference pairs, and *Best-of-N (BoN) Accuracy*, which evaluates robustness when ranking among multiple distractors. Compared with *Pairwise Acc*, *BoN Acc* applies a stricter criterion by requiring the correct action to outrank all distractors simultaneously, providing stronger discriminative power and better alignment with downstream agent performance.

Pairwise Acc. Given a preference pair (a^+, a^-), where a^+ is the correct action and a^- a rejected one, the WebPRM is correct if it assigns higher preference to a^+. Formally:

$$Acc_{Pairwise} = \frac{1}{|\mathcal{D}_{\text{Bench}}|} \sum_{(a^+, a^-) \in \mathcal{D}_{\text{Bench}}} \mathbb{1}\big[\pi_\theta(a^+) \succ \pi_\theta(a^-)\big]. \tag{7}$$

BoN Acc. For each instance $(a^+, a^{-_1}, \ldots, a^{-_Q}) \in \mathcal{D}_{\text{Bench}}$, the WebPRM is considered correct only when a^+ is consistently ranked above all Q distractors, with $Q = 4$ in our benchmark. BoN Acc is:

$$Acc_{BoN} = \frac{1}{|\mathcal{D}_{\text{Bench}}|} \sum_{i=1}^{|\mathcal{D}_{\text{Bench}}|} \prod_{q=1}^{Q} \mathbb{1}[\pi_\theta(a_i^+) \succ \pi_\theta(a_i^{-_q})]. \tag{8}$$

5 Experiments

We conduct comprehensive experiments to evaluate WebArbiter on the reward modeling benchmark WEBPRMBENCH in § 5.1 and on practical applications in § 5.2.

5.1 WEBPRMBENCH

Experimental Setup

Baselines. We compare WebArbiter against three categories of baselines. (1) Proprietary LLM-as-judge models, including GPT-4o-mini (OpenAI 2024b), GPT-4o (OpenAI 2024a), GPT-5 (OpenAI 2025a), Claude-3.7-Sonnet (Anthropic 2025), and Gemini-2.5-Flash (Sundar Pichai and Demis Hassabis 2025), which are prompted to act as judges by selecting the preferred action given task context. (2) Open-source LLM-as-judge models, represented by Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Qwen et al. 2025), and Llama-3-70B-Instruct (Meta 2024), providing accessible yet competitive instruction-tuned baselines. (3) WebPRMs, where we include WebShepherd (Chae et al. 2025).

Implementation Details. We train WebArbiter on WEBPRM Collection (Chae et al. 2025), which comprises 30k step-level preference pairs drawn from the Mind2Web environment. We use 10k pairs for stage-1 reasoning distillation and the remainder for stage-2 reinforcement learning. Models are initialized from Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Qwen et al. 2025) and fine-tuned with LoRA (Hu et al. 2022). Further implementation details are provided in the Appendix C.

Evaluation Metrics. We report results using two complementary metrics: *Pairwise Accuracy*, which measures correctness on individual preference pairs, and *Best-of-N (BoN) Accuracy*, which evaluates robustness under multi-candidate settings. Detailed definitions are provided in § 4.2.

Main Results

WebArbiter Significantly Outperforms Baselines. As shown in Tab. 2, WebArbiter consistently surpasses both proprietary and open-source LLMs across all environments with *BoN Acc*. While LLM-as-judge methods often maintain moderate *Pairwise Acc*, their performance drops sharply on *BoN Acc*, revealing poor robustness to hard negatives. In contrast, WebArbiter sustains strong results on both metrics, establishing its reliability under realistic multi-candidate settings.

Advantage over the SOTA WebPRM. WebShepherd (Chae et al. 2025) represents the previous SOTA WebPRMs. Trained on the same WEBPRM Collection, which was drawn from the Mind2Web environment, WebArbiter-7B achieves an *Avg. BoN Acc* of 77.78%,

Models	Mind2Web			WebArena	AssistantBench	WorkArena	Avg.
	Cross-Task	Cross-Website	Cross-Domain				
Count	142	148	417	371	29	180	1287

Table 1: Data distribution of WEBPRMBENCH, the first comprehensive evaluation benchmark spanning diverse environments for WebPRMs.

Models	Mind2Web		WebArena		AssistantBench		WorkArena		Avg.	
	Pairwise	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*
LLM-as-judge, Proprietary Language Models										
GPT-4o-mini	80.28	45.69	81.40	61.46	**88.79**	<u>68.97</u>	84.86	66.67	83.83	60.70
GPT-4o	80.60	52.84	79.78	56.06	83.62	58.62	85.69	66.67	82.42	58.55
GPT-5	82.06	62.26	76.89	55.26	76.72	58.62	80.83	65.56	79.12	60.42
Claude Sonnet	81.19	59.60	81.40	62.53	76.92	53.85	<u>88.06</u>	72.22	81.89	62.05
Gemini Flash	81.94	57.86	<u>83.89</u>	<u>67.39</u>	<u>84.62</u>	65.38	**92.36**	<u>77.08</u>	<u>85.70</u>	<u>66.93</u>
LLM-as-judge, Open-source Language Models										
Qwen2.5-3B-Instruct	76.89	37.66	69.54	35.04	80.17	48.28	70.56	42.22	74.29	40.80
Qwen2.5-7B-Instruct	79.02	41.00	72.44	40.16	76.72	37.93	78.06	51.67	76.56	42.69
Llama-3-70B-Instruct	79.43	50.96	77.90	56.60	78.85	46.15	87.50	70.14	80.92	55.96
WebPRMs (3B)										
WebShepherd-3B	37.41	21.22	20.33	9.47	36.54	17.24	10.49	2.44	26.19	14.84
★ WebArbiter-3B	<u>93.28</u>	<u>78.75</u>	83.29	56.87	76.72	44.83	84.03	60.56	84.33	60.25
WebPRMs (7B+)										
WebShepherd-8B	72.35	46.68	33.16	12.37	55.77	44.83	35.85	12.68	49.28	29.14
★ WebArbiter-7B	**96.47**	**90.07**	**84.30**	**71.43**	80.17	**72.41**	87.36	**77.22**	**87.08**	**77.78**

Table 2: Results on WEBPRMBENCH with *Pairwise* and *BoN Acc*. ★ denotes our models. Bold numbers indicate the best results, while <u>underlined</u> numbers denote the second best. Our WebArbiter-7B achieves the highest *BoN Acc* across all environments, with an *Avg. BoN Acc* of 77.78%, outperforming the second-best baseline, i.e., Gemini Flash, by 10.85%.

surpassing WebShepherd-8B by an absolute gain of 48%. Unlike WebShepherd, which relies on fragile checklists, WebArbiter employs principle-guided reasoning, yielding judgments robust to environment and page variations. Case studies illustrating these differences are provided in Appendix E.

Robust Generalization Across Environments. WebArbiter not only excels in-domain, achieving 96.47% *Pairwise Acc* and 90.07% *BoN Acc* on Mind2Web, but also generalizes across diverse benchmarks. On WebArena, it outperforms the second-best baseline by nearly 4% in *BoN Acc*, gains about 3% on AssistantBench, and still edges out strong baselines on WorkArena with 77.22%. These results confirm that principle-guided reasoning supports both strong in-domain learning and robustness across heterogeneous, noisy, and enterprise-level environments.

Ablation Study. We compare four training variants to disentangle the effects of reinforcement learning, principle guidance, and justification style. *Instruct (Original)* denotes a purely instruction-tuned model without additional optimization. *Cold Start RL* directly applies RL on top of the instruction model. *Cold Start RL + Principles* augments RL with principle prompting during training, enabling explicit principle induction before judgment. $SFT_{w/o\ Principles}$ + *RL* performs reasoning distillation without principles, followed by RL, thereby testing whether narrative-style justifications alone are sufficient. As shown in Tab. 3, WebArbiter achieves the best performance. Explicit principles anchor judgments to progress, producing stable supervision under multi-candidate web settings.

RL Alone is Unstable Across Web Environments. *Cold Start RL* performs well on in-domain Mind2Web but collapses on out-of-domain benchmarks. This highlights that reward optimization without reasoning distillation struggles in noisy and complex environments.

Principles Enable Cross-Environment Generalization. Augmenting RL with principles boosts *Avg. BoN Acc*, especially in structurally diverse environments such as WebArena and AssistantBench. Principles provide transferable facets for reasoning, reducing reliance on brittle layout cues and improving robustness to web variability.

Method	Mind2Web		WebArena		AssistantBench		WorkArena		Avg.	
	Pairwise	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*	*Pairwise*	*BoN*
Instruct (Original)	79.02	41.00	72.44	40.16	76.72	37.93	78.06	51.67	76.56	42.69
Instruct + Cold Start RL	**97.63**	**91.38**	67.59	43.40	71.55	34.48	73.33	55.00	77.53	56.07
Instruct + Cold Start RL + Principles	96.42	87.88	<u>84.10</u>	<u>60.65</u>	<u>79.31</u>	<u>55.17</u>	<u>83.19</u>	<u>55.56</u>	<u>85.75</u>	<u>64.81</u>
Instruct + $SFT_{\text{w/o Principles}}$ + RL	94.26	82.39	75.34	49.87	68.97	41.38	78.61	54.44	79.30	57.02
★ WebArbiter	<u>96.47</u>	<u>90.07</u>	**84.30**	**71.43**	**80.17**	**72.41**	**87.36**	**77.22**	**87.08**	**77.78**

Table 3: Ablation results on WEBPRMBENCH with Qwen2.5-7B-Instruct as backbone. Best results are in bold and the second best underlined. WEBARBITER, combining principle-guided reasoning distillation with RL, achieves the highest overall performance.

Reasoning Without Principles is Insufficient. $SFT_{\textit{w/o Principles}}$ + *RL*, i.e., narrative-style justifications alone, improves fluency but lags behind principle-aware settings. This confirms that narrating reasoning chains without principles cannot ensure alignment with true task progress in complex, long-horizon real-world web navigation.

5.2 Reward-Guided Trajectory Search

Experimental Setup and Implementations. Reward-guided trajectory search represents one of the most practical applications of PRMs, as it directly leverages fine-grained step-level supervision to improve decision quality during agent execution. To evaluate WebArbiter in this setting, we conduct experiments on WebArena-Lite[1] (Liu et al. 2024b), which contains diverse, long-horizon tasks such as online shopping and content management, closely reflecting real-world web activities. Performance is measured with Success Rate. Following WebShepherd (Chae et al. 2025), we adopt a Best-of-N sampling strategy: the policy model generates $N = 5$ candidate actions for each step, and WebArbiter selects the most promising one through a Knockout Tournament mechanism (Guo et al. 2025b). We evaluate two policies, GPT-4o-mini (OpenAI 2024b) and GPT-4o (OpenAI 2024a).

Analysis. As shown in Tab. 4, WebArbiter achieves substantial average improvements under both policy models, far surpassing baselines. Its advantages arise from three main factors. First, reasoning mitigates spurious correlations that often mislead WebPRMs in domains such as Shopping and Reddit. Gains in Shopping are particularly striking, as tasks require dense semantic retrieval and inference; stronger policies can roll out more promising candidate actions, and WebArbiter's structured reward modeling further amplifies these benefits. Second, in GitLab, tasks frequently allow multiple equivalent paths. WebShepherd is brittle under such variability, whereas WebArbiter leverages reasoning over historical trajectories and current states to evaluate action validity, enabling stronger generalization in dynamic workflows. By contrast, CMS exhibits a more template-driven structure, where actions closely follow standardized patterns. In such cases, checklist-based supervision remains comparatively effective, which narrows the relative performance margin. Overall, WebArbiter's reasoning-first design consistently provides robust, interpretable, and scalable supervision across diverse domains.

[1]We did not have access to the MAP domain during this work and therefore excluded it from our experiments.

Policy	WebPRM	Shopping	CMS	Reddit	GitLab	Avg.	Δ
GPT-4o-mini	w/o Trajectory Search*	21.74	22.86	19.05	34.38	24.51	–
	GPT-4o-mini	24.44	22.86	26.32	33.33	26.74	+2.23
	WebShepherd-8B*	26.09	**45.71**	23.81	40.62	34.06	+9.55
	★ WebArbiter-7B	**37.78**	42.86	**36.84**	**46.67**	**41.04**	**+16.53**
GPT-4o	w/o Trajectory Search*	23.91	31.43	28.57	56.25	35.04	–
	GPT-4o-mini	26.67	37.14	42.11	40.00	36.48	+1.44
	WebShepherd-8B*	30.43	**42.86**	47.62	46.88	41.95	+6.91
	★ WebArbiter-7B	**44.44**	**42.86**	**52.63**	**56.67**	**49.15**	**+14.11**

Table 4: Success Rates (%) of trajectory search with GPT-4o-mini and GPT-4o as policy on WebArena-lite. * Results reported from the WebShepherd (Chae et al. 2025). Δ is relative to the *w/o Trajectory Search* baseline. Our WebArbiter consistently achieves the highest gains across both policy models.

6 Conclusion

We presented WEBARBITER, a reasoning-first, principle-inducing process reward model that frames reward modeling as structured text generation and produces auditable step-level judgments with rationales. Through reasoning distillation and reinforcement learning, WebArbiter transforms superficial correlations into robust signals that verify genuine task progress, enforce trajectory consistency, and generalize across dynamic websites. To support systematic evaluation, we released WEBPRMBENCH, the first comprehensive evaluation benchmark spanning diverse environments for WebPRMs in web navigation, covering four domains with diverse tasks and fine-grained step-level supervision. Extensive experiments demonstrate SOTA performance on WEBPRMBENCH and substantial improvements in reward-guided trajectory search on WebArena-Lite, establishing principle-guided reasoning WebPRMs as a robust and interpretable foundation for scalable web agents.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Adept. 2022. Act-1: Transformer for actions. https://www.adept.ai/blog/act-1/.

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

Anthropic. 2025. Claude 3.7 Sonnet and Claude Code. https://www.anthropic.com/news/claude-3-7-sonnet.

Boisvert, L.; Thakkar, M.; Gasse, M.; Caccia, M.; Chezelles, T. L. S. D.; Cappart, Q.; Chapados, N.; Lacoste, A.; and Drouin, A. 2025. WorkArena++: Towards Compositional Planning and Reasoning-based Common Knowledge Work Tasks. arXiv:2407.05291.

Chae, H.; Kim, S.; Cho, J.; Kim, S.; Moon, S.; Hwangbo, G.; Lim, D.; Kim, M.; Hwang, Y.; Gwak, M.; Choi, D.; Kang, M.; Im, G.; Cho, B.; Kim, H.; Han, J. H.; Kwon, T.; Kim, M.; woo Kwak, B.; Kang, D.; and Yeo, J. 2025. Web-Shepherd: Advancing PRMs for Reinforcing Web Agents. arXiv:2505.15277.

Chen, X.; Li, G.; Wang, Z.; Jin, B.; Qian, C.; Wang, Y.; Wang, H.; Zhang, Y.; Zhang, D.; Zhang, T.; Tong, H.; and Ji, H. 2025. RM-R1: Reward Modeling as Reasoning. arXiv:2505.02387.

Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; and Su, Y. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36: 28091–28114.

Drouin, A.; Gasse, M.; Caccia, M.; Laradji, I. H.; Verme, M. D.; Marty, T.; Boisvert, L.; Thakkar, M.; Cappart, Q.; Vazquez, D.; Chapados, N.; and Lacoste, A. 2024. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? arXiv:2403.07718.

Fu, Y.; Kim, D.-K.; Kim, J.; Sohn, S.; Logeswaran, L.; Bae, K.; and Lee, H. 2024. Autoguide: Automated generation and selection of state-aware guidelines for large language model agents. *arXiv preprint arXiv:2403.08978*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guo, J.; Chi, Z.; Dong, L.; Dong, Q.; Wu, X.; Huang, S.; and Wei, F. 2025b. Reward Reasoning Model. arXiv:2505.14674.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; and Chen, E. 2024. Understanding the planning of LLM agents: A survey. arXiv:2402.02716.

Kim, G.; Baldi, P.; and McAleer, S. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.

Koh, J. Y.; McAleer, S.; Fried, D.; and Salakhutdinov, R. 2024. Tree Search for Language Model Agents. *arXiv preprint arXiv:2407.01476*.

Koh, J. Y.; McAleer, S.; Fried, D.; and Salakhutdinov, R. 2025. Tree Search for Language Model Agents. arXiv:2407.01476.

Li, M.; Zhao, S.; Wang, Q.; Wang, K.; Zhou, Y.; Srivastava, S.; Gokmen, C.; Lee, T.; Li, E. L.; Zhang, R.; et al. 2024. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37: 100428–100534.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Liu, C. Y.; Zeng, L.; Liu, J.; Yan, R.; He, J.; Wang, C.; Yan, S.; Liu, Y.; and Zhou, Y. 2024a. Skywork-Reward: Bag of Tricks for Reward Modeling in LLMs. arXiv:2410.18451.

Liu, X.; Zhang, T.; Gu, Y.; Iong, I. L.; Xu, Y.; Song, X.; Zhang, S.; Lai, H.; Liu, X.; Zhao, H.; Sun, J.; Yang, X.; Yang, Y.; Qi, Z.; Yao, S.; Sun, X.; Cheng, S.; Zheng, Q.; Yu, H.; Zhang, H.; Hong, W.; Ding, M.; Pan, L.; Gu, X.; Zeng, A.; Du, Z.; Song, C. H.; Su, Y.; Dong, Y.; and Tang, J. 2024b. VisualAgentBench: Towards Large Multimodal Models as Visual Foundation Agents. arXiv:2408.06327.

Liu, Y.; Yao, Z.; Min, R.; Cao, Y.; Hou, L.; and Li, J. 2025. PairJudge RM: Perform Best-of-N Sampling with Knockout Tournament. arXiv:2501.13007.

Lù, X. H.; Kazemnejad, A.; Meade, N.; Patel, A.; Shin, D.; Zambrano, A.; Stańczak, K.; Shaw, P.; Pal, C. J.; and Reddy, S. 2025. AgentRewardBench: Evaluating Automatic Evaluations of Web Agent Trajectories. arXiv:2504.08942.

Ma, K.; Zhang, H.; Wang, H.; Pan, X.; and Yu, D. 2023. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*.

Mahan, D.; Phung, D. V.; Rafailov, R.; Blagden, C.; Lile, N.; Castricato, L.; Fränken, J.-P.; Finn, C.; and Albalak, A. 2024. Generative Reward Models. arXiv:2410.12832.

Meta. 2024. Introducing Meta Llama 3: The most capable openly available LLM to date. https://ai.meta.com/blog/meta-llama-3/.

Miao, B.; Wu, Y.; Gao, M.; Yu, Q.; Bu, W.; Zhang, W.; Li, Y.; Tang, S.; Chua, T.-S.; and Li, J. 2025. Boosting Virtual Agent Learning and Reasoning: A Step-Wise, Multi-Dimensional, and Generalist Reward Model with Benchmark. arXiv:2503.18665.

OpenAI. 2024a. GPT-4o. https://platform.openai.com/docs/models/gpt-4o.

OpenAI. 2024b. GPT-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. 2025a. GPT-5 is here. https://openai.com/gpt-5/.

OpenAI. 2025b. Introducing Operator. https://openai.com/index/introducing-operator/.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155.

Pan, J.; Zhang, Y.; Tomlin, N.; Zhou, Y.; Levine, S.; and Suhr, A. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Park, J.; Jwa, S.; Ren, M.; Kim, D.; and Choi, S. 2024. OffsetBias: Leveraging Debiased Data for Tuning Evaluators. arXiv:2407.06551.

Prasad, A.; Koller, A.; Hartmann, M.; Clark, P.; Sabharwal, A.; Bansal, M.; and Khot, T. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

Qi, Z.; Liu, X.; Iong, I. L.; Lai, H.; Sun, X.; Zhao, W.; Yang, Y.; Yang, X.; Sun, J.; Yao, S.; Zhang, T.; Xu, W.; Tang, J.; and Dong, Y. 2025. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. arXiv:2411.02337.

Qwen; :; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Tang, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2025. Qwen2.5 Technical Report. arXiv:2412.15115.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2024. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv: 2409.19256*.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Sodhi, P.; Branavan, S. R. K.; Artzi, Y.; and McDonald, R. 2024. SteP: Stacked LLM Policies for Web Actions. arXiv:2310.03720.

Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; and Zhang, C. 2024. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 36.

Sundar Pichai; and Demis Hassabis. 2025. Gemini 2.5 Flash. https://deepmind.google/models/gemini/flash/.

Tao, H.; TV, S.; Shlapentokh-Rothman, M.; Hoiem, D.; and Ji, H. 2023. Webwise: Web interface control and sequential exploration with large language models. *arXiv preprint arXiv:2310.16042*.

Uesato, J.; Kushman, N.; Kumar, R.; Song, F.; Siegel, N.; Wang, L.; Creswell, A.; Irving, G.; and Higgins, I. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Wang, P.; Li, L.; Shao, Z.; Xu, R.; Dai, D.; Li, Y.; Chen, D.; Wu, Y.; and Sui, Z. 2023a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Wang, T.; Kulikov, I.; Golovneva, O.; Yu, P.; Yuan, W.; Dwivedi-Yu, J.; Pang, R. Y.; Fazel-Zarandi, M.; Weston, J.; and Li, X. 2024a. Self-Taught Evaluators. arXiv:2408.02666.

Wang, Z.; Dong, Y.; Delalleau, O.; Zeng, J.; Shen, G.; Egert, D.; Zhang, J. J.; Sreedhar, M. N.; and Kuchaiev, O. 2024b. HelpSteer2: Open-source dataset for training top-performing reward models. arXiv:2406.08673.

Wang, Z.; Dong, Y.; Zeng, J.; Adams, V.; Sreedhar, M. N.; Egert, D.; Delalleau, O.; Scowcroft, J. P.; Kant, N.; Swope, A.; and Kuchaiev, O. 2023b. HelpSteer: Multi-attribute Helpfulness Dataset for SteerLM. arXiv:2311.09528.

Wei, Z.; Yao, W.; Liu, Y.; Zhang, W.; Lu, Q.; Qiu, L.; Yu, C.; Xu, P.; Zhang, C.; Yin, B.; Yun, H.; and Li, L. 2025. WebAgent-R1: Training Web Agents via End-to-End Multi-Turn Reinforcement Learning. arXiv:2505.16421.

Wu, T.; Yuan, W.; Golovneva, O.; Xu, J.; Tian, Y.; Jiao, J.; Weston, J.; and Sukhbaatar, S. 2024. Meta-Rewarding Language Models: Self-Improving Alignment with LLM-as-a-Meta-Judge. arXiv:2407.19594.

Xi, Z.; Ding, Y.; Chen, W.; Hong, B.; Guo, H.; Wang, J.; Yang, D.; Liao, C.; Guo, X.; He, W.; et al. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*.

Ye, Z.; Li, X.; Li, Q.; Ai, Q.; Zhou, Y.; Shen, W.; Yan, D.; and Liu, Y. 2025. Learning LLM-as-a-judge for preference alignment. In *The Thirteenth International Conference on Learning Representations*.

Yoran, O.; Amouyal, S. J.; Malaviya, C.; Bogin, B.; Press, O.; and Berant, J. 2024. AssistantBench: Can Web Agents Solve Realistic and Time-Consuming Tasks? arXiv:2407.15711.

Zhang, L.; Hosseini, A.; Bansal, H.; Kazemi, M.; Kumar, A.; and Agarwal, R. 2024. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.

Zhang, Y.; Lin, C.; Tang, S.; Chen, H.; Zhou, S.; Ma, Y.; and Tresp, V. 2025a. SwarmAgentic: Towards Fully Automated Agentic System Generation via Swarm Intelligence. *arXiv preprint arXiv:2506.15672*.

Zhang, Y.; Ma, Z.; Ma, Y.; Han, Z.; Wu, Y.; and Tresp, V. 2025b. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 23378–23386.

Zhang, Z.; Zheng, C.; Wu, Y.; Zhang, B.; Lin, R.; Yu, B.; Liu, D.; Zhou, J.; and Lin, J. 2025c. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623.

Zheng, L.; Wang, R.; Wang, X.; and An, B. 2023b. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*.

Zheng, Y.; Zhang, R.; Zhang, J.; Ye, Y.; Luo, Z.; Feng, Z.; and Ma, Y. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*

Linguistics (Volume 3: System Demonstrations). Bangkok, Thailand: Association for Computational Linguistics.

Zhou, E.; Zheng, G.; Wang, B.; Xi, Z.; Dou, S.; Bao, R.; Shen, W.; Xiong, L.; Fan, J.; Mou, Y.; Zheng, R.; Gui, T.; Zhang, Q.; and Huang, X. 2025. RMB: Comprehensively Benchmarking Reward Models in LLM Alignment. arXiv:2410.09893.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; Alon, U.; and Neubig, G. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. arXiv:2307.13854.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Contents

A Notation Summary

For clarity, we summarize the main notations used throughout this paper:

- $\mathcal{E}$: web environment, defined by state space $\mathcal{S}$, action space $\mathcal{A}$, and observation space $\mathcal{O}$.
- T: state transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.
- $\mathcal{I}$: task instruction.
- s_p, o_p, a_p: state, observation, and action at step p.
- c_p: reasoning trace associated with action a_p.
- $c_{<p}$: reasoning traces of all previously executed actions.
- $\tau = (a_1, \ldots, a_P)$: trajectory of length P.
- $j = (j_1, \ldots, j_L)$: structured justification of length L, consisting of explicit reasoning and a final verdict.
- π_θ: WebArbiter model parameterized by θ.
- $\hat{y}$: predicted preference verdict.
- $\mathcal{D}_{\text{Train}}, \mathcal{D}_{\text{SFT}}, \mathcal{D}_{\text{RL}}$: training datasets for supervised distillation and reinforcement learning.
- $\mathcal{D}_{\text{Bench}}$: evaluation dataset for WEBARBITER-BENCH.
- a^+: action from a successful trajectory (positive label).
- a^- or a^{-q}: rejected action ($q = 1, \ldots, Q$).
- Q: number of distractors in WEBARBITER-BENCH (set to 4).
- M: dataset size (training or evaluation).
- $R(x, \hat{y})$: verifiable reward signal.

B Example of Preference Dataset

To illustrate the construction of a preference pair, consider a task instruction from WEBARENA:

- **Instruction** $\mathcal{I}$: “Find the 2026 conference submission page on the ICLR website.‘’
- **Current observation** o_p: [356] banner ’header’, role=’banner’ [359] link ’Home’ [380] button ’Select Year (2026)’ [386] button ’Dates’ [391] button ’Calls’ [396] button ’Guides’ [401] button ’Organization’ [403] heading ’ICLR 2026’
- **History** $(a_{<p}, c_{<p})$:
  - a_1: Search "ICLR", c_1: “I need to find the official ICLR website first. Let me search for ’ICLR’ to locate it.‘’
  - a_2: Click link "ICLR homepage", c_2: “I can see the ICLR official website link in the search results. Clicking on it will take me to the ICLR homepage where I can find the conference submission information.‘’
- **Candidate actions and reasoning traces**:
  - (a_p^1, c_p^1): Click link "Call for Papers" ; c_p^1: “I can see a ’Call for Papers’ link on the ICLR homepage. This link would likely lead to the submission details page, which should contain information about the 2026 conference submission process that I’m looking for.‘’
  - (a_p^2, c_p^2): Click "About" link; c_p^2: “I can see an ’About’ link on the ICLR homepage. Since I need to find the 2026 conference submission page, the ’About’ section might contain conference overview information including links to submission details or important dates for the 2026 conference.‘’
- **Label** y: a_p^1 is preferred.

This example is represented in the dataset as:

$$(\mathcal{I},\, o_p,\, a_{<p},\, c_{<p},\, (a_p^1, c_p^1),\, (a_p^2, c_p^2),\, y = a_p^1).$$

C Training Details

All training is conducted on 8 NVIDIA A100-80GB GPUs with fixed random seeds. Our training framework is bead on LLama-Factory (Zheng et al. 2024) and VERL (Sheng et al. 2024)

Distillation Stage. We train the model for 5 epochs with a learning rate of 8e-4, using LoRA with a rank of 128. We apply a cosine learning rate scheduler with a warmup ratio of 0.1. We set the batch size to 256 and the maximum sequence length to 8,192 tokens.

RLVR Stage. We employ the VERL framework for GRPO training. The learning rate is set to 7.0×10^{-6}. The training uses a fixed batch size of 1,024 with mini-batch size of 128, and adopts Fully Sharded Data Parallel (FSDP) for enhanced memory efficiency. For rollout generation, we deploy vLLM with tensor parallelism of 4 and GPU memory utilization limited to 0.4. Response sampling uses standard parameters (temperature=1.0, top-p=1.0), generating 7 candidate responses per prompt. We apply KL regularization with a coefficient of 1.0×10^{-3} and clip ratio of 0.2. The maximum input sequence length is 8,192 tokens, and the maximum response length is 4,096 tokens.

D Prompt Repository

WebArbiter

```
You are a skilled expert at evaluating assistant responses. You should evaluate given
    responses based on the given judging criteria.\n Given the context of the
    conversation and two responses from the Assistant, you need to determine the better
    response. Provide an overall comprehensive comparison upon them.
#### Intent ####
{intent}
#### AXTREE ####
Note: [bid] is the unique alpha-numeric identifier at the beginning of lines for each
    element in the AXTree. Always use bid to refer to elements in your actions.
{observation}
#### Trajectory ####
Note: The trajectory contains the sequence of previous actions and their corresponding
    thoughts. Each entry reflects the agent’s internal reasoning (‘thought‘) and the
    concrete operation it performed (‘action‘).
{trajectory}
#### start url ####
{start_url}
#### current url ####
The URL provides clues about the user’s position in the application flow. Use both the
    path and query parameters to infer page type (e.g., homepage, search results,
    product detail, cart, checkout).
{current_url}
#### Assistant Responses ####
[The Begin of Response 1]
THOUGHT:
{thought1}
ACTION:
{action1}
[The End of Response 1]
[The Begin of Response 2]
THOUGHT:
{thought2}
ACTION:
{action2}
[The End of Response 2]
### Output Instructions ###
Format your output strictly using the following XML-style tags:
<State>Summarize the current state based on the URL, AXTree, and previous actions.
    Include what page the user is currently on, and what relevant UI elements or
    information are visible.</State>
<Criteria>Other potential criteria specific to the query and the context, and the
    weights of each criteria.</Criteria>
<Analysis>Compare Response 1 and Response 2 in detail according to the <State> and
    <Criteria>.</Analysis>
<Answer>Response 1 or Response 2</Answer>
Rules for <Answer>:
- If Response 1 is better, output exactly: <Answer>Response 1</Answer>
- If Response 2 is better, output exactly: <Answer>Response 2</Answer>
Important Notes:
- Be objective and base your evaluation strictly on the content of the responses.
- Do not let the response order, length bias your judgment.
```

E Case Study: WebArbiter vs. WebShepherd

To further illustrate the differences between WebArbiter and WebShepherd, we present a representative example from WEBARENA. The task instruction is:

> *"What is the rating of* Ugreen lightning to 3.5mm cable*? Round to the nearest whole number."*

At the current step, the agent observes a search-results page listing the target product. The snippet already shows a "65%" rating in the result list, but the product detail page has not been opened yet. Two candidate actions are considered:

- a_p^1: `send_msg_to_user` with "65%".
  c_p^1: "The list view already shows a 65% rating, so answer directly."
- a_p^2: Click the target product entry to open its detail page, then extract and (if needed) round the rating.
  c_p^2: "Verify the rating on the product page (correct product, correct field) before responding."

WebShepherd. WebShepherd evaluates candidates using *checklist*-style templates that are *precompiled* before the next observation. These checklists typically include predicates such as "verify on the product page"; hence, even when a rating is already visible in an earlier search-results snippet, the checklist *still requires verification*, and thus tends to favor a_p^2. When the actual page deviates from the ex-ante forecast (e.g., an early results page surfaces), the precompiled predicates become stale: they enforce a verification path that may be unnecessary or even brittle under interstitials or layout drift.

WebArbiter. WebArbiter derives principles such as *"answer directly when the objective's required field is already unambiguously satisfied by the current observation," "ensure correct rounding,"* and *"avoid redundant navigation when the answer is already grounded."* It performs *dynamic expectation alignment*: (i) it forms expectation about what evidence is needed, (ii) compares the actual page with that expectation, and (iii) revises principle weights and candidate scoring accordingly. Concretely, upon seeing a clear "65%" rating in the snippet, it downweights "must verify on product page" and upweights "answer directly with proper rounding," issuing a preference verdict for a_p^1 and correctly completing the task with minimal steps.

Discussion. This case illustrates a key limitation of precompiled, open-loop checklists: they conflate *procedural requirements* ("must navigate to detail page") with *goal satisfaction* and thus underperform when early observations already satisfy the objective. In contrast, WebArbiter grounds decisions in explicit, principle-guided reasoning *and* closed-loop, dynamic expectation alignment (predict $\rightarrow$ observe $\rightarrow$ compare $\rightarrow$ revise), enabling it to act on already-sufficient evidence and remain robust to goal–observation mismatches.

$\mathcal{I}$ Instruction

What is the rating of Ugreen lightning to 3.5mm cable? Round to the nearest whole number.

$(a_{<p}, r_{<p})$ Trajectory

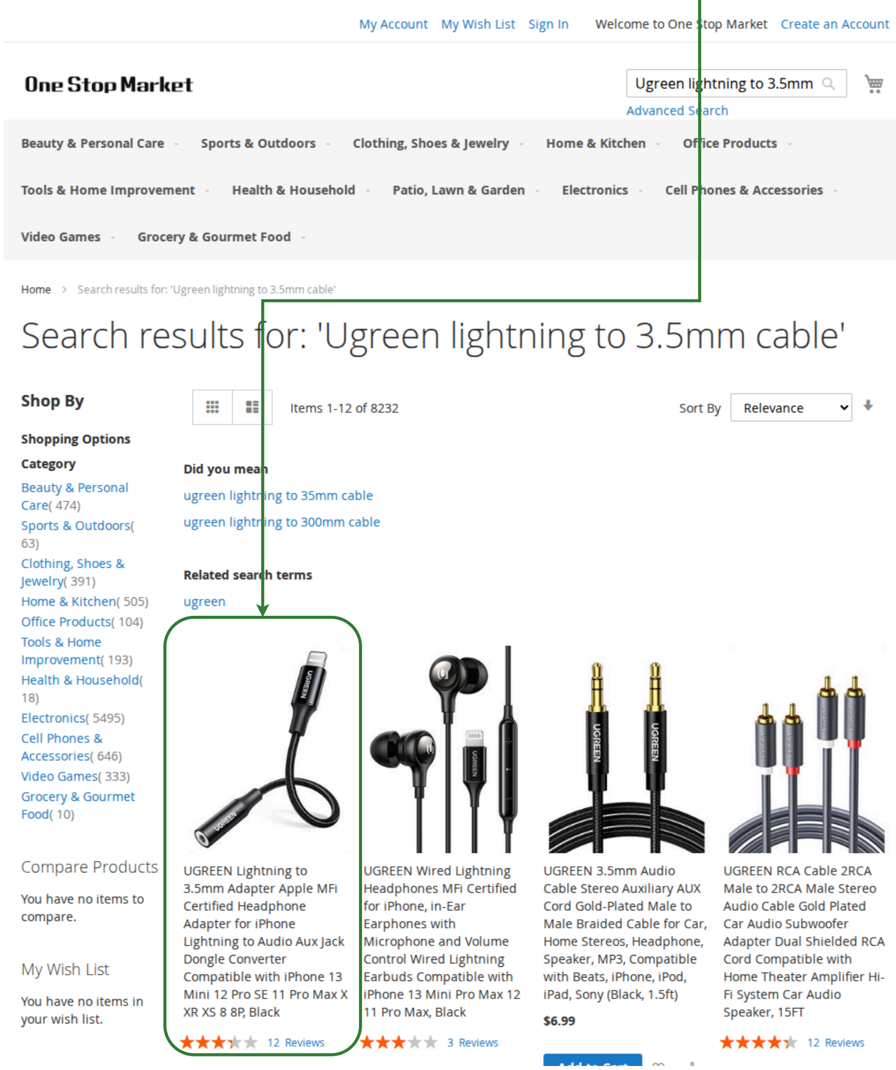
Thought: On One Stop Market's homepage, the search ...
Action: *type [2826] "Ugreen lightning to 3.5mm cable"*
Thought: The current Page is on ....
...

(a_p^1, r_p^1) Action Candidate 1

Thought: The page lists a 65% rating on the search results snippet.
Action: `send_msg_to_user ["65%"]`

(a_p^2, r_p^2) Action Candidate 2

Thought: Open the product page to verify the rating, then round to nearest whole.
Action: `click [6368]`



Websheperd Analysis

✗ **Action Candidate 1**

Checklist 1: Search for Product (Yes)
Results for 'Ugreen lightning to 3.5mm cable' are visible; the list view already shows a 65% rating, so I can answer directly.

Checklist 2: Extract Rating (No)
Outputs '65%' from the results snippet without locating a rating field or confirming formatting on the product page.

Checklist 3: Verify Page (No)
Does not open the product detail page; no on-page verification or source evidence is performed.

✓ **Action Candidate 2**

Checklist 1: Search for Product (Yes)
Results are visible, but the correct next step is to open the product page to verify the rating.

Checklist 2: Extract Rating (In Progress)
Plan to locate the rating on the product page and then round, the action moves toward the rating field, but the value has not yet been located or parsed at this step.

Checklist 3: Verify Page (No)
Does not open the product detail page; no on-page verification or source evidence is performed.

WebAribiter Analysis

Principle

1. Correctness of the rating value (60%)
2. Correct rounding to the nearest whole number (20%)
3. Reference to the right product (not a different product) (10%)
4. Action appropriateness (should not click a wrong element) (10%)

Comparing against the principle:

1. Correct rating – both fine.
2. Rounding – both fine.
3. Correct product
   - Action Candidate1 implicitly correct
   - Action Candidate 2 may be off.
4. Action appropriateness
   - Action Candidate 1 minimal

Conclusion:

✓ **Action Candidate 1** — more accurate and progressive (Chosen)

✗ **Action Candidate 2** — may click the wrong element