MolmoWeb: Open Visual Web Agent and Open Data for the Open Web

Tanmay Gupta[1⋆], Piper Wolters[1*], Zixian Ma[1,2*], Peter Sushko[1*], Rock Yuren Pang[1,2], Yue Yang[1], Taira Anderson[1], Zhongzheng Ren[1,2,3], Harsh Trivedi[1], Winson Han[1], and Ranjay Krishna[1,2]

[1] Allen Institute for AI
[2] University of Washington
[3] UNC-Chapel Hill

Abstract. Web agents—autonomous systems that navigate and execute tasks on the web on behalf of users—have the potential to transform how people interact with the digital world. However, the most capable web agents today rely on proprietary models with undisclosed training data and recipes, limiting scientific understanding, reproducibility, and community-driven progress. We believe agents for the open web should be built in the open. To this end, we introduce (1) MolmoWebMix, a large and diverse mixture of browser task demonstrations and web-GUI perception data and (2) MolmoWeb a family of fully open multimodal web agents. Specifically, MolmoWebMix combines over 100K synthetic task trajectories from multiple complementary generation pipelines with 30K+ human demonstrations, atomic web-skill trajectories, and GUI perception data, including referring expression grounding and screenshot question answering. MolmoWeb agents operate as instruction-conditioned visual-language action policies: given a task instruction and a webpage screenshot, they predict the next browser action, requiring no access to HTML, accessibility trees, or specialized APIs. Available in 4B and 8B size, on browser-use benchmarks like WebVoyager, Online-Mind2Web, and DeepShop, MolmoWeb agents achieve state-of-the-art results outperforming similar scale open-weight-only models such as Fara-7B, and Holo1-7B. MolmoWeb-8B also surpasses set-of-marks (SoM) agents built on much larger closed frontier models like GPT-4o. We further demonstrate consistent gains through test-time scaling via parallel rollouts with best-of-N selection, achieving 94.7% and 60.5% pass@4 (compared to 78.2% and 35.3% pass@1) on WebVoyager and Online-Mind2Web respectively. We release model checkpoints, training data, code, and a unified evaluation harness to enable reproducibility and accelerate open research on web agents (GitHub).

Keywords: Multimodal Web Agents · Agentic AI · Open Weights and Data · Vision-Language Models

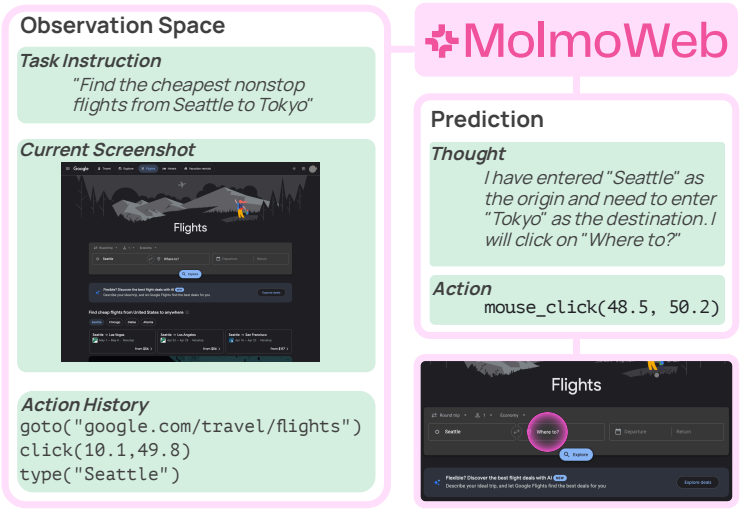

Fig. 1: Overview of MolmoWeb. At each step, the agent receives the task instruction, current screenshot, and action history as input. It then predicts a natural language thought followed by a browser action.

1 Introduction

The World Wide Web is an indispensable part of modern human life, with billions of people relying on it daily for information, communication, commerce, and entertainment [13]. From booking flights and managing finances to navigating government services, comparison shopping, and coordinating logistics, countless everyday tasks require users to interact with complex, multi-step web interfaces that demand sustained attention, domain knowledge, and patience [13, 22]. Web agents—autonomous systems that can navigate and execute such tasks on the web at the behest of users [20]—have the potential to fundamentally reshape how humans interact with this vast digital ecosystem [36, 41]. Effective web agents could broaden digital access for users who face barriers due to disability or inaccessible design [36, 41], as well as for users with limited digital skills or digital literacy [22].

Recent progress in large language and vision-language models has brought web agents closer to practical viability, supported by research benchmarks demonstrating multi-step web interaction (e.g., clicking, form filling, and navigation) under natural-language instructions [5, 17, 31, 46]. In parallel, frontier labs now expose computer- or browser-use capabilities where models perceive GUI state via screenshots and output low-level actions such as click/type/scroll to complete workflows [7, 23, 25]. However, the most capable end-to-end systems are typically offered as hosted, proprietary services with limited disclosure of training data and full recipes [7, 23]. This exacerbates long-standing concerns that

⋆ Equal contribution.

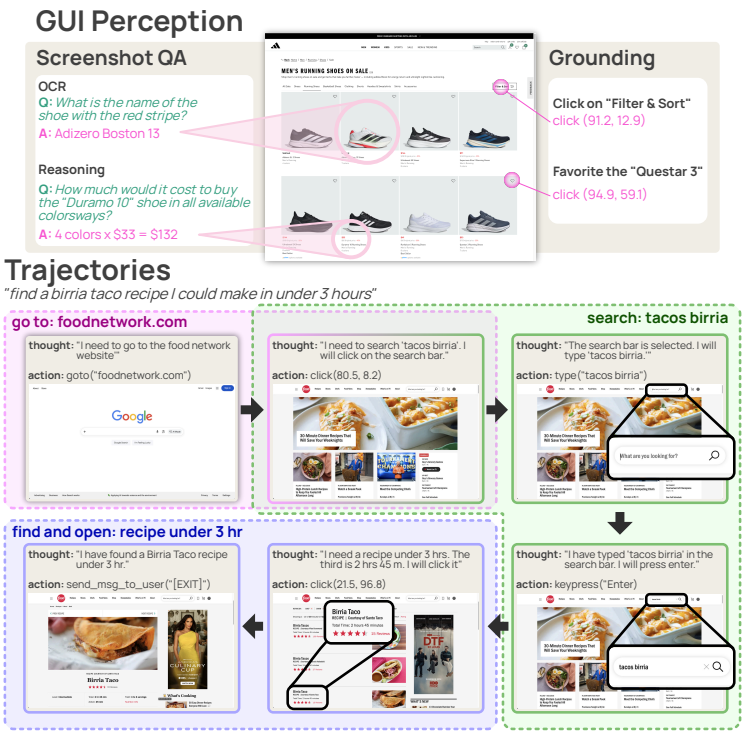

Fig. 2: Overview of MolmoWebMix dataset. To capture the diverse capabilities required for web agents, MolmoWebMix combines GUI perception data involving screenshot QA **(top left)** and referring expression grounding **(top right)**, with synthetic and human task trajectories demonstrating task completion on the web **(bottom)**. The human trajectories are further segmented into atomic skills (Section B in the Appendix shows our skill taxonomy).

insufficient reporting and artifacts hinder reproducibility and scientific understanding of what actually drives performance [9, 27]. The resulting opacity is especially concerning for autonomous agents operating on the open web, where trustworthy deployment depends on transparency, accountability, and auditable behavior [21, 40].

We believe agents for the open web should be built in the open. To this end, we provide the research community with a complete, transparent, and reproducible foundation including: **MolmoWebMix**, a mixture of diverse task demonstrations on the browser and web-GUI perception data (Sec. 2); and **MolmoWeb**, a family of fully open and efficient multimodal web agents (Sec. 3). MolmoWeb agents are vision-language models (VLMs) trained to serve as instruction-conditioned multimodal action policies for the web. Given a task instruction, action history, and a screenshot of the webpage, MolmoWeb agent predicts the next browser action. The action is executed in the browser to yield a new observation and this loop repeats until the task is complete.

Crucially, MolmoWeb agents operate on any website using only the visual interface that human users see, without requiring specialized APIs or access to the underlying HTML or accessibility tree (AxTree). This vision-centric design is a deliberate choice motivated by several considerations. First, it mirrors how people actually use the web, grounding the agent in the same perceptual modality and making its behavior more interpretable.

Second, it avoids the brittleness of Document Object Model (DOM) based representations, which vary significantly across websites, frameworks, and even minor page updates, and which can be incomplete or misleading for dynamically rendered content. Third, it sidesteps the substantial token consumption and compute overhead that structured page representations incur: AxTree inputs can easily consume tens of thousands of tokens per page, whereas a single screenshot provides a compact, information-rich representation of the same content.

The key to training capable MolmoWeb agents is curating a high-quality and diverse training data mixture that simultaneously teaches the model to understand web screenshots as well as task execution behaviors, such as filling forms, multi-hop navigation, and finding relevant information from a webpage. MolmoWebMix contains data from four complementary data sources: (1) synthetic task trajectories generated at scale from multiple complementary pipelines, (2) human demonstrations collected by crowdworkers on real websites, (3) atomic web skill trajectories providing targeted supervision for compositional browser-use skills, and (4) GUI perception data consisting of referring expression grounding and screenshot question-answering examples.

Despite their relatively compact size at 4B or 8B scale, MolmoWeb agents are highly performant. **Without distilling from other visual web agents**, MolmoWeb agents are competitive with or outperform open-weights agents of comparable size, including Fara-7B [2] and Holo1-7B [1]. More notably, they also outperform set-of-marks (SoM) prompting based web agents [44] built on much larger closed frontier models like GPT-4o that take both AxTree and SoM-annotated screenshots as inputs. This result is particularly striking because these closed-model baselines enjoy substantially richer input representations *and* orders-of-magnitude more parameters, yet a well-trained, vision-only MolmoWeb agent achieves stronger task completion rates—suggesting that data quality and targeted training can compensate for raw model scale and additional inputs. Finally, we demonstrate further gains by leveraging increased compute at inference-time via multiple rollouts with best outcome selection using a VLM judge.

2 MolmoWebMix

Our training data consists of 3 broad categories: task trajectories, atomic skill trajectories, and GUI perception data. We outline the trajectory generation pipeline below, which encompasses web browsing task sampling, trajectory execution, and filtering out failed trajectories, as shown in Figure 3. Specific details

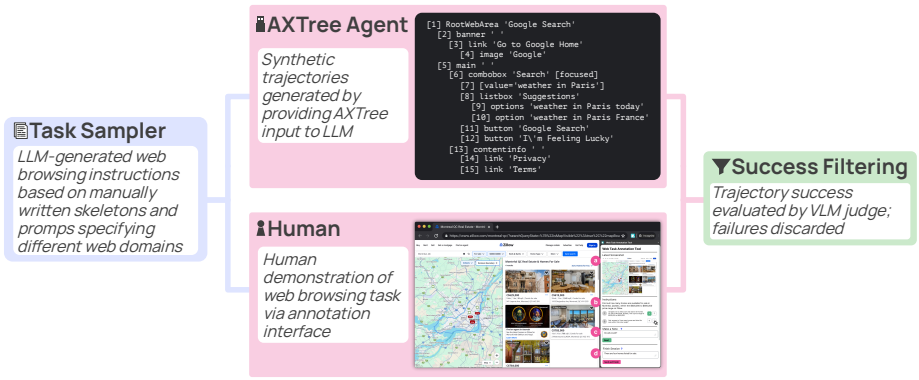

Fig. 3: Data generation pipeline. Our pipeline consists of a task sampler, trajectory generation executed by either humans or AxTree agents, and success filtering.

about the atomic skill trajectories and GUI perception data can be found in Section B of the Appendix.

2.1 Task trajectories

Trajectories from AxTree agents. A bulk of our synthetic trajectories are generated using an LLM agent that operates on the AxTree representation of the webpage. At each step, the agent receives the serialized AxTree (visualized in Figure 3) as its observation along with the instruction and past actions, and predicts the next action by referencing element browser IDs (`bid`) assigned to each interactive node. We used Gemini-3-Flash-Preview as the agent backbone.

Tasks were sourced from a combination of manually authored and LLM-generated instructions, covering popular websites and benchmark evaluation sites (see Section D in the Appendix for details). Although the agent observes only the AxTree, we capture a screenshot at each step so that all trajectories share a unified observation format. Actions predicted in bid-space are post-processed into pixel-space coordinates (clicks, typing, scrolling). See Table 2 for the complete list of actions.

Synthetic trajectories were filtered for task success using the WebVoyager LLM judge [11], retaining only trajectories deemed successfully completed.

Trajectories from multiagent harness. To generate trajectories with potentially a higher quality of thoughts and trajectories, we design a multi-agent system (Figure 4) where the system orchestrates web navigation through three specialized roles working in concert: a `Planner` generates the immediate next subgoal based on the high-level task goal as well as task progress determined from verification feedback; an `Operator` executes one low-level browser action (like clicking, scrolling etc.) at each step to accomplish the current subgoal; and a `Verifier` checks whether the current subgoal has been completed by analyzing the most recent 5 screenshots. The trajectory generation process operates

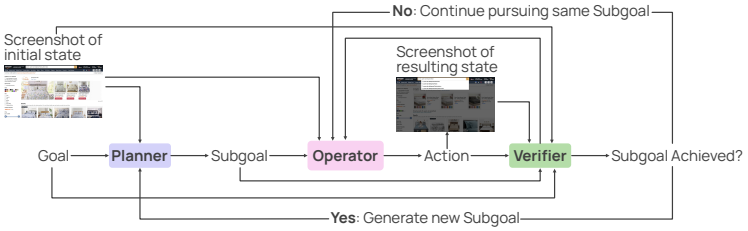

Fig. 4: Multi-agent trajectory generation pipeline. A Planner decomposes the task into sequential subgoals. For each subgoal, an Operator generates an action conditioned on the current state and subgoal. After execution, the resulting state is evaluated by a Verifier using screenshots to determine whether the subgoal is achieved. If not, the system continues attempting the same subgoal; otherwise, the Planner generates the next one. The loop repeats until the overall goal is completed.

iteratively. At each step, the system first runs the `Verifier` to check if the current subgoal is complete. If the subgoal is verified as complete or fails to complete within 5 steps, the `Planner` is called to generate the next subgoal. The system then injects the current subgoal, completed steps, and planner/verifier reasoning into the text prompt and passes this augmented prompt along with the current screenshot to the `Operator`, which returns a concrete browser action. Finally, all historical information is tracked and fed back into subsequent planning and execution cycles. We use Gemini-2.5-Flash for the `Planner`, GPT4-o as the `Verifier`, and Gemini AxTree agent as the Operator. We empirically find that this multi-agent setup achieves higher task completion success rates than using the Gemini AxTree agent alone, scoring 78.5 vs. 74.4 on WebVoyager [11].

Trajectories from humans. In addition to the synthetic trajectories, we collected human demonstrations via a custom Chrome extension that captures browser interaction events (e.g., clicks, scrolls, keystrokes) and corresponding screenshots, and post-processes them into standardized trajectories. Crowdworkers were provided with tasks sourced from a mix of manually authored and LLM-generated instructions (see Section C in the Appendix for details). In general, tasks were chosen to span a wide range of popular websites and to require a variety of browser task-completion skills including search, form-filling, and multi-hop navigation.

To support fine-grained annotation, each task instruction was decomposed into an ordered sequence of subtasks. Workers checked off each subtask upon completion within the annotation tool and submitted a final text response (e.g., an answer to a question or a completion acknowledgment). If a subtask could not be completed due to missing content or an unexpected webpage state, workers recorded their best attempt along with a descriptive note explaining the failure.

Each trajectory is reviewed by a human to verify task completion and ensure that screenshots and actions were correctly captured. Trajectories failing review were revised or re-collected.

Table 1: Statistics of MolmoWebMix. We also report its constituent sources compared to other datasets.

Dataset / Source	Trajectories	Steps	Domains	Avg steps	Mixture Ratio	Open-source
Mind2Web [5]	2.4K	17K	137	7.3	–	Yes
AutoWebWorld [42]	11.6K	255K	29	21.9	–	Yes
Fara [2]	145K	1.01M	–	6.9	–	No
MolmoWebMix-Traj	278.5K	2.2M	2.6K	13.2	0.80	Yes
Synthetic						
AxTree Single-Agent	70K	793K	1.3K	11.4	0.35	Yes
Axtree Multi-Agent	35K	438K	1.1K	12.5	0.18	Yes
Axtree Atomic Skills	5.5K	68.7K	540	12.4	0.02	Yes
Node-Traversal	16K	151K	833	9.5	0.02	Yes
Human						
Human	36K	623K	1.1K	20.8	0.18	Yes
Human Skills	116K	781K	1.1K	6.8	0.05	Yes
MolmoWebMix-Perception	–	10.5M	–	–	0.20	Yes
PixMoPoints + SyntheticGround	–	8.3M	–	–	0.15	Yes
ScreenshotQA	–	2.2M	–	–	0.05	Yes

2.2 GUI perception

Effective web navigation requires the agent to perceive and understand webpage screenshots before taking any action. GUI perception data teaches the model to identify interactive elements on the page (e.g., buttons, links, input fields, and other controls), understand their function and affordances, and extract information from the page in response to natural language queries. This combines skills akin to referring expression grounding, OCR, and reading comprehension, grounded in the visual structure of web interfaces.

Grounding. Grounding data consists of (screenshot, element description) → click point pairs, where the model must predict the pixel coordinate to click for interacting with a specified element. We extract these pairs automatically from AxTree agent trajectories. For each screenshot, we enumerate all clickable elements in the AxTree and for each generate a natural language description using its accessible name and role either with a template or GPT5. The ground-truth click coordinate is sampled randomly within the element's bounding box using a clipped Gaussian prior centered at the element's center, encouraging the model to learn spatially robust clicking rather than always targeting exact centers. In total, we generate more than 7M grounding QA pairs, comprising 3.4M templated queries and 3.8M GPT-5-generated queries. In addition to new grounding data, we also repurpose the PixmoPoints data from Molmo [4] by formatting single-point QA pairs into click actions and include 1.1M examples in our grounding mixture.

Screenshot QA. Screenshot QA data consists of (screenshot, question) → answer pairs that enhance the model's ability to read and reason about webpage content. For each screenshot in a subset of the AxTree agent trajectories, we provide the corresponding AxTree to an LLM and prompt it to generate question–answer pairs. Questions cover three categories: OCR queries about text and

values present on the page (e.g., prices, counts, text content), affordance queries about actions available on the page (e.g., "Where would I find financial news on this page?"), and summarization queries about the overall content or purpose of a page element. To ensure that questions and answers rely solely on visual content, we remove samples that contain references to AxTree-specific information, such as element IDs (e.g., "Click on Bid 32"). In total, the dataset covers 395 websites and 2,237,252 QA pairs, split between 54% OCR, 26% affordance, and 20% summarization questions.

3 MolmoWeb

In this section, we describe the MolmoWeb architecture, its observation and action space, and the training procedure. MolmoWeb is designed to be simple and practical: we build on an existing vision-language model backbone, define a minimal but expressive action space over browser operations, and train end-to-end via supervised fine-tuning on the MolmoWebMix mixture.

Architecture. MolmoWeb is built on the Molmo2 architecture [4], a multimodal language model that processes interleaved sequences of images and text. As shown in Figure 1, the model takes as input the current webpage screenshot together with the task instruction and the history of past actions, and produces a structured output comprising a natural language thought followed by the next action to execute.

3.1 Observation and action space

Observations. At each step t, the model receives a screenshot of the current browser viewport along with the task instruction. To provide temporal context, the history of actions taken in 10 prior steps is appended as context along with the URL and title of the current page.
Actions. The model predicts a JSON object encoding both a *thought* and an *action* specifying the operation to perform. The thought is a brief natural language rationale for the chosen action. The full action space is described in Tab. 2. Mouse actions are parameterized by spatial coordinates normalized to $[0, 100]$, with 2 decimal points, which are denormalized to viewport pixel coordinates at execution time.

Training. We train MolmoWeb end-to-end via supervised fine-tuning (SFT) on all training data described in Sec. 2. All data types, including task trajectories, atomic skill trajectories, and GUI perception data, are mixed in a single training stage. The mixing ratios across data sources are treated as hyperparameters; we ablate different mixtures and select the combination that best balances performance across evaluation benchmarks. We report the datasets' final ratios in the mixture in Table 1.

Table 2: Action space of MolmoWeb. Spatial coordinates are normalized to $[0, 100]$ during training and denormalized to viewport pixels for browser execution. Section E in the Appendix shows the distribution of actions in various subsets of MolmoWebMix.

Action	Description
`goto(url)`	Navigate the browser to a URL
`mouse_click(x, y, ...)`	Click at a viewport coordinate
`mouse_drag_and_drop(...)`	Drag from one coordinate to another
`scroll(delta_x, delta_y)`	Scroll the page by an offset
`scroll_at(x, y, dx, dy)`	Scroll at a specific coordinate
`hover_at(x, y)`	Hover at a specific coordinate
`keyboard_type(text)`	Type a string of text
`keyboard_press(key)`	Press a key or key combination
`go_back()`	Navigate to the previous page
`new_tab()`	Open a new browser tab
`tab_focus(index)`	Switch focus to a browser tab
`noop(wait_ms)`	Wait (e.g., for page load or captcha)
`send_msg_to_user(msg)`	Display a message to the user

We finetune Molmo2 [4] (with Qwen3 language model and SigLIP2 vision encoder) following its best practice and tune the language model, the vision encoder, and the adapter starting from the single-image checkpoint (pretrained on image captioning and finetuned on single-image QA). We train all models with 64 H100 GPUs with a global batch size of 128 for up to 50K steps (approximately 3.2 epochs on average, with slightly different numbers of epochs across datasets based on their final ratio in the mixture).

4 Experiments

Our experiments consist of comparison to prior art, a study on scaling test-time compute through more inference steps per rollout as well as parallel rollouts with best-of-N selection, data ablation studies to understand the impact of scale and role of human vs synthetic data, comparison of token sampling strategies, and grounding evaluation. We first describe our evaluation setup and then present our experimental results.

Evaluation setup. We evaluate MolmoWeb on four popular web browsing benchmarks using live websites: WebVoyager [11], Online-Mind2Web [43], DeepShop [19], and WebTailBench [2]. Because some tasks are time-sensitive, we change dates in outdated requests (*e.g.*, find a flight on August 5, 2025) to be meaningful for the task across all benchmarks. For each benchmark and model, we run 3-5 evaluations up to 100 steps and report the average score

Table 3: Comparison to prior work on browser-use benchmarks. Benchmark columns are abbreviated as WebV (WebVoyager [11]), OM2W (Online-Mind2Web [43]), DShop (DeepShop [19]), and WTB (WebTailBench [2]). For models marked with a *, the numbers reported are from Fara [2]. Italic Online-Mind2Web results are from the Online-Mind2Web auto-eval leaderboard. Since it's unclear what judge was used by [2] for WebTailBench, we used the WebVoyager judge to compute numbers marked with †. We highlight the **best** and second-best performing numbers for each benchmark in the sub-8B, open-weight model category.

Model	# Steps	WebV	OM2W	DShop	WTB
API only					
w/o vision					
Axtree Agent (GPT-5)	30	70.6	41.9	40.7	29.2†
Axtree Agent (Gemini-3-flash)	30	74.4	34.8	45.1	62.1†
Axtree Agent (Gemini-3-flash)	100	85.6	44.8	55.3	63.5†
w/ vision					
SoM Agent (GPT-4o)*	100	65.1	34.6	16.0	30.8
SoM Agent (o3)*	100	79.3	55.4	49.7	52.7
SoM Agent (GPT-5)*	100	90.6	57.7	49.1	60.4
OpenAI computer-use-preview*	100	70.9	*58.3*	24.7	25.7
Gemini computer-use-preview	100	88.6	*57.3*	62.0	63.0†
Yutori Navigator [34]	–	–	*64.7*	–	–
Open weight					
Holo1-7B [1]	30	55.4	–	–	–
UI-TARS-1.5-7B* [30]	100	66.4	31.3	11.6	19.5
GLM-4.1V-9B-Thinking* [12]	100	66.8	33.9	32.0	22.4
Fara-7B* [2]	100	73.5	<u>34.1</u>	26.2	38.4
Ours: Open weight, data, training, and evaluation					
MolmoWeb-4B	100	<u>75.2</u>	31.3	<u>35.6</u>	<u>43.8</u>†
MolmoWeb-8B	100	**78.2**	**35.3**	**42.3**	**49.5**†

across runs. If a model does not complete the task by the maximum number of steps, it is considered a failure. As environment errors occasionally occur, we allow up to 10 retries per trajectory; tasks that do not complete within this budget are also marked as failures. We use the same prompt and LLM-as-a-judge setup corresponding to each published benchmark. Specifically, we use GPT-4o with the official prompt from WebVoyager and DeepShop, and o4-mini with respective judge for Online-Mind2Web. Since, it is unclear what judge was used for WebTailBench, we used the WebVoyager judge. For WebVoyager, we use the task set provided by [2] which removed infeasible tasks. All evaluations are run in a Browserbase environment, allowing many live browsers to run in parallel with some captcha-solving capability.

4.1 Comparison to prior work

In Tab. 3, we compare MolmoWeb to web-agents based on proprietary APIs as well as open-weight models.

MolmoWeb establishes a new state-of-the-art among open-weight models. MolmoWeb-8B shows healthy gains over leading open-weight models like Fara across all benchmarks. Even MolmoWeb-4B outperforms all open-weight models on WebVoyager and DeepShop while being competitive on Online-Mind2Web and WebTailBench. These gains largely demonstrate efficacy of MolmoWebMix in teaching a strong foundation vision-language model like Molmo2 to follow instructions for browser-task completion.

Comparison to closed visual agents. Compared to SoM agents that use set-of-marks annotated visual prompts, MolmoWeb-8B easily outperforms GPT-4o, matches the performance of o3 on WebVoyager, and is only 6 pts behind GPT-5 and o3 on DeepShop. MolmoWeb-8B also outperforms OpenAI computer-use-preview on WebVoyager and DeepShop as per the performance reported in [2].

Comparison to non-visual teacher agent. Our synthetic data generation pipeline is powered by Gemini-3-flash axtree agent. It is natural to ask how does the visual student, MolmoWeb, fare against the non-visual axtree-guided teacher? MolmoWeb-8B agent is 5+ pts behind the teacher on WebVoyager and Online-Mind2Web while being 10+ pts behind on DeepShop and WebTailBench. Besides the observation space, other factors contributing to this gap include: (i) differences in model sizes with Gemini presumably being a much larger model compared to our humble 4B and 8B models); (ii) click action requires pixel-space pointing capability from MolmoWeb but the Axtree agent uses unique numeric identifiers (`bid` [3]) of the target element which is then programmatically mapped to a click location; and (iii) answering questions based on text rendered on the webpage requires MolmoWeb to implicitly perform OCR to parse text from screenshot in addition to reading comprehension required by the Axtree agent.

4.2 Test-time scaling

We explore two ways of utilizing more compute at inference time: (i) performing parallel rollouts with a best-of-N selection with an LLM-judge; and (ii) increasing the number of inference steps.

Parallel rollouts. We study how much performance can be gained by running k independent agents on the same task and selecting the best outcome. To get an unbiased, low variance estimate of pass@k, we collect $m > k$ rollouts per task and compute the estimate:

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{m-c}{k}}{\binom{m}{k}}, \tag{1}$$

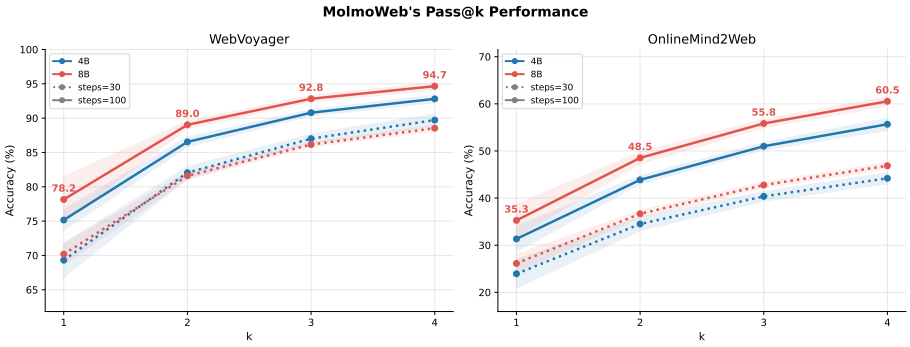

Fig. 5: Pass@k performance on WebVoyager and Online-Mind2Web Accuracy improves as k increases, with MolmoWeb-8B generally outperforming MolmoWeb-4B, and 100 steps outperforming 30 steps by a large margin. The MolmoWeb-8B model with max 100 steps reaches 94.7% and 60.5% at Pass@4.

where c is the number of successful rollouts among the m attempts. We use an LLM judge to evaluate success for each rollout, and we set $m = 5$ to balance accuracy with computational cost. We plot MolmoWeb-4B and MolmoWeb-8B's pass@k performance on the WebVoyager benchmark with max steps of 30 and $k = 1, ..., 4$ in Fig. 5 using the respective judge for best rollout selection. We see that for both 4B and 8B models, parallel rollouts improve success rates substantially with pass@4 leading to 20+ pt gains over single rollout performance. Parallel rollouts help mitigate the compounding error problem where a single incorrect action could throw the agent off-course. These massive gains from parallel rollouts suggest (or, perhaps explain to the unsurprised why) self-distillation from best-of-N rollouts and RL might be effective strategies for further improving single rollout performance.

Increasing inference steps. Figure 5 shows consistent gain from increasing the maximum number of inference steps from 30 to 100 across all benchmarks. However, we note that the gains from scaling via parallel runs (each with 30 steps) and picking the best result using an LLM judge far outperform increasing the number of inference steps (e.g., 8B model achieves 86.2% via 3 parallel runs with max 30 steps resulting in a total of 90 steps compared to 78.2% via increasing the steps to 100 in one run).

4.3 Training data ablations

In Table 4, we evaluate the effect of data scale and contributions from synthetic and human data to final performance. The ablations used a slightly earlier version of MolmoWebMix which had fewer human trajectories as well as fewer synthetic trajectories but largely had a similar composition to the final version of MolmoWebMix.

Table 4: Data ablations. These ablations use an earlier version of MolmoWebMix with both fewer synthetic and human trajectories compared to the final experiments and use 30 inference steps.

(a) Effect of training data scale

Training data	WebV	OM2W
1%	44.5	11.7
10%	63.2	20.4
100%	**68.5**	**21.9**

(b) Humans vs. synthetic data

Training data	#trajs	WebV	OM2W
human only	28K	27.8	13.2
synthetic only	106K	67.8	**22.0**
synthetic + human	134K	**68.5**	21.4

Performance improves with training data scale. Table 4a shows that, unsurprisingly, performance improves with data scale. Interestingly, about 85 to 90% of the performance can be achieved with just 10% of the dataset highlighting the effectiveness of the training mixture.

Limited gains from human data on benchmarks. Table 4b shows that gains from human data are somewhat limited and not consistent across benchmarks. This is likely due to lower volume of human data, differences in task distribution compared to benchmarks, difficulty in learning new actions present only in human data (`scroll_at` and `mouse_drag_and_drop`), difference in style and quality of post-hoc generated thoughts, and human annotation noise (e.g. variable scroll amounts in human annotation as opposed to always scrolling by 100% of viewport height in synthetic data). We hypothesize that synthetic and human data represent distinctly different web task completion policies and the model struggles to learn generalizable behavior across both. For instance, while we observe the model trained only on human data produces both `scroll` and `scroll_at` actions, when trained on human and synthetic trajectories the model almost always produces `scroll` which tries to scroll the page instead of the element.

Synthetic trajectories are more effective for learning than human trajectories for the same tasks. To further investigate the difference in learning from trajectories collected by AxTree agents and humans, we collect 2700 web browsing trajectories with each method, using the same task instructions. We then train the model on each of these datasets, mixed with MolmoWebMix-Perception data. Table 5 shows that the model trained on AxTree-sourced trajectories consistently outperforms the one trained on human trajectories, suggesting that the synthetic data provides a more reliable learning signal. We hypothesize two contributing factors for higher signal-to-noise ratio in synthetic trajectories. First, humans tend to exhibit more exploratory behavior, particularly on unfamiliar websites, resulting in longer and noisier trajectories with detours that may hinder imitation learning. Second, the LLM agent operates on the accessibility tree, which encodes rich structural and semantic information

about page elements that may not be immediately apparent from visual cues alone. This additional context likely enables the LLM to produce more direct and consistent trajectories, making them more effective as training data despite being synthetically generated.

Table 5: Learning from LLM-generated trajectories is more effective than learning from human demonstrations.

Source	#trajs	DeepShop	WebV	OM2W
human	2.7K	19.8	35.4	9.0
synthetic	2.7K	**24.4**	**53.0**	**16.8**

Sampling strategies. Table 6 compares greedy, top-k, and top-p (nucleus) sampling on WebV. We find that sampling strategy significantly affects MolmoWeb's performance, achieving over 5% gains with top-k and top-p sampling compared to greedy decoding. Qualitatively, we find the greedy-decoding could get stuck at specific states (e.g. trying to click at the same location or continuing to scroll even when past attempts at doing so have failed), while the other randomized strategies overcome this behavior. Between top-k and top-p sampling, we see that nucleus sampling with p=0.8 and temperature=0.7 performs the best on Webvoyager and hence is the default sampling strategy used for all experiments. We choose the sampling parameters based on Qwen3's (the base LLM used in Molmo2) recommended parameters on HuggingFace [38].

Table 6: Effect of sampling strategy. Greedy sampling performs significantly worse than random sampling strategies.

Strategy	WebV
greedy (temperature=0.0)	61.4
top-k (temperature=0.7, k=20)	67.4
top-p (temperature=0.7, p=0.8)	**68.5**

5 Related work

Prior work on LLM-driven web agents broadly splits into operating on DOM-derived text representations [5, 10, 46] and exposing web capabilities via APIs to tool-use agents [33, 35], with additional work on fine-tuning [16], multi-agent orchestration [6], and tree search [15]. More recent efforts build multimodal agents that operate directly on screenshots, including proprietary models [8, 24] and open-weights models such as UI-Tars [29, 30, 37], Holo-1 [1], Fara [2], and OpenCUA [39]. We also draw on GUI understanding work for auxiliary supervision [18, 28, 45]. Evaluation has evolved from sandboxed environments [14, 32, 46] toward live websites with VLM-as-a-judge verification [11, 26, 43]. Unlike prior

open-weights work, we release full training data, pipelines, and model weights without distillation from proprietary vision-based agents; see Appendix Section H for further discussion.

6 Capabilities and Limitations

Below, we outline the limitations of MolmoWeb along various dimensions (specific details are in Section G of the Appendix):

- **Instruction following:** MolmoWeb performs best for more specific instructions but performance may degrade as the instructions become more ambiguous or require more exploration.
- **OCR and reading comprehension:** We see instances of failures due to OCR for smaller texts or for answering complex questions requiring understanding of large passages of text.
- **Latency:** MolmoWeb is not optimized for latency. Additionally, browser action execution is much slower for hosted browsers like Browserbase, than for local browsers.
- **Thoughts:** An interesting capability of MolmoWeb is producing thoughts, though sometimes thoughts do not correlate well with actions.
- **Action space:** MolmoWeb uses the action space that humans use when interacting with GUIs, which is not necessarily the most efficient way to tackle web browsing tasks.
- **Error correction and restarts:** While MolmoWeb shows some error correction behavior, the agent may sometimes get stuck in states where it incorrectly keeps predicting the same action without being able to recover or course-correct.

7 Conclusion

We introduced MolmoWebMix and MolmoWeb, a fully open data and model suite for multimodal web agents. Operating purely from screenshots, MolmoWeb agents outperform both comparable open-weight models and set-of-marks agents built on much larger proprietary models, while benefiting from test-time scaling via parallel rollouts. We'll release all checkpoints, data, code, and evaluation tools to support reproducible research and accelerate progress on open web agents.

References

1. Andreux, M., Skuk, B.B., Benchekroun, H., Bir'e, E., Bonnet, A., Bordie, R., Bout, N., Brunel, M., Cedoz, P.L., Chassang, A., Chen, M., Constantinou, A.D., d'Andign'e, A., de la Jonquiere, H., Delfosse, A., Denoyer, L., Deprez, A., Derupti, A., Eickenberg, M., Federico, M., Kantor, C., Koegler, X., Labb'e, Y., Lee, M., de Kergaradec, E.L.J., Mahla, A., Manevich, A., Maret, A., Masson, C., Maurin, R., Mena, A., Modard, P., Moyal, A., Kerbel, A.N., Revelle, J., Richter, M.L., Santos, M., Sifre, L., Theillard, M., Thibault, M., Thiry, L., Tronchon, L., Usunier, N., Wu, T.: Surfer-h meets holo1: Cost-efficient web agent powered by open weights. ArXiv **abs/2506.02865** (2025), https://api.semanticscholar.org/CorpusID:279119644
2. Awadallah, A., Lara, Y., Magazine, R., Mozannar, H., Nambi, A., Pandya, Y., Rajeswaran, A., Rosset, C., Taymanov, A., Vineet, V., Whitehead, S., Zhao, A.: Fara-7b: An efficient agentic model for computer use. arXiv:2511.19663 (2025)
3. de Chezelles, T.L.S., Gasse, M., Lacoste, A., Drouin, A., Caccia, M., Boisvert, L., Thakkar, M., Marty, T., Assouel, R., Shayegan, S.O., Jang, L., Lù, X.H., Yoran, O., Kong, D., Xu, F.F., Reddy, S., Cappart, Q., Neubig, G., Salakhutdinov, R., Chapados, N.: The browsergym ecosystem for web agent research. ArXiv **abs/2412.05467** (2024), https://api.semanticscholar.org/CorpusID:274598279
4. Clark, C., Zhang, J., Ma, Z., Park, J.S., Salehi, M., Tripathi, R., Lee, S., Ren, Z., Kim, C.D., Yang, Y., Shao, V., Yang, Y., Huang, W., Gao, Z., Anderson, T., Zhang, J., Jain, J., Stoica, G., Han, W., Farhadi, A., Krishna, R.: Molmo2: Open weights and data for vision-language models with video understanding and grounding (2026), https://arxiv.org/abs/2601.10611
5. Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., Su, Y.: Mind2web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems **36** (2024)
6. Fourney, A., Bansal, G., Mozannar, H., Tan, C., Salinas, E., Zhu, E., Niedtner, F., Proebsting, G., Bassman, G., Gerrits, J., Alber, J., Chang, P., Loynd, R., West, R., Dibia, V., Awadallah, A., Kamar, E., Hosn, R., Amershi, S.: Magentic-one: A generalist multi-agent system for solving complex tasks (2024), https://arxiv.org/abs/2411.04468
7. Google: Computer use | gemini api | google ai for developers. Documentation, https://ai.google.dev/gemini-api/docs/computer-use, accessed: 2026-03-05
8. Google: Computer use | gemini API documentation (2026), https://ai.google.dev/gemini-api/docs/computer-use, accessed: 2026-03-04
9. Gundersen, O.E., Kjensmo, S.: State of the art: Reproducibility in artificial intelligence. In: AAAI Conference on Artificial Intelligence (AAAI) (2018), https://ojs.aaai.org/index.php/AAAI/article/view/11503
10. Gur, I., Furuta, H., Huang, A., Safdari, M., Matsuo, Y., Eck, D., Faust, A.: A real-world WebAgent with planning, long context understanding, and program synthesis. arXiv preprint arXiv:2307.12856 (2023)
11. He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., Yu, D.: Webvoyager: Building an end-to-end web agent with large multimodal models. In: ACL (2024)
12. Hong, W., Yu, W., Gu, X., Wang, G., Gan, G., Tang, H., Cheng, J., Qi, J., Ji, J., Pan, L., et al.: Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. arXiv preprint arXiv:2507.01006 (2025)

13. International Telecommunication Union (ITU): Facts and figures 2024: Internet use. Web page (2024), https://www.itu.int/itu-d/reports/statistics/2024/11/10/ff24-internet-use/, accessed: 2026-03-05
14. Koh, J.Y., Lo, R., Jang, L., Duvvur, V., Lim, M.C., Huang, P.Y., Neubig, G., Zhou, S., Salakhutdinov, R., Fried, D.: Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. ArXiv **abs/2401.13649** (2024), https://api.semanticscholar.org/CorpusID:267199749
15. Koh, J.Y., McAleer, S., Fried, D., Salakhutdinov, R.: Tree search for language model agents. arXiv preprint arXiv:2407.01476 (2024)
16. Lai, H., Liu, X., Iong, I.L., Yao, S., Chen, Y., Shen, P., Yu, H., Zhang, H., Zhang, X., Dong, Y., Tang, J.: AutoWebGLM: Bootstrap and reinforce a large language model-based web navigating agent (2024), https://arxiv.org/abs/2404.03648
17. Liu, E.Z., Guu, K., Pasupat, P., Shi, T.T., Liang, P.: Reinforcement learning on web interfaces using workflow-guided exploration. In: International Conference on Learning Representations (ICLR) (2018), https://arxiv.org/pdf/1802.08802
18. Lu, Y., Yang, J., Shen, Y., Awadallah, A.: Omniparser for pure vision based gui agent. ArXiv **abs/2408.00203** (2024), https://api.semanticscholar.org/CorpusID:271601072
19. Lyu, Y., Zhang, X., Yan, L., de Rijke, M., Ren, Z., Chen, X.: Deepshop: A benchmark for deep research shopping agents. ArXiv **abs/2506.02839** (2025), https://api.semanticscholar.org/CorpusID:279118560
20. Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al.: Webgpt: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332 (2021)
21. National Institute of Standards and Technology (NIST): Artificial intelligence risk management framework (ai rmf 1.0). NIST Special Publication (2023), https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf, accessed: 2026-03-05
22. OECD: Briefing note: Digital skills and digital inclusion. PDF (2023), https://www.oecd.org/content/dam/oecd/en/about/projects/cfe/oecd-city-network-on-jobs-and-skills/Briefing-note-Digital-skills-and-digital-inclusion.pdf, accessed: 2026-03-05
23. OpenAI: Computer use | openai api. Documentation, https://developers.openai.com/api/docs/guides/tools-computer-use/, accessed: 2026-03-05
24. OpenAI: Computer use | openai api documentation (2025), https://developers.openai.com/api/docs/guides/tools-computer-use/, accessed: 2026-03-04
25. OpenAI: Openai for developers in 2025. Blog post (2025), https://developers.openai.com/blog/openai-for-developers-2025/, accessed: 2026-03-05
26. Pan, Y., Kong, D., Zhou, S., Cui, C., Leng, Y., Jiang, B., Liu, H., Shang, Y., Zhou, S., Wu, T., et al.: WebCanvas: Benchmarking web agents in online environments. arXiv preprint arXiv:2406.12373 (2024)
27. Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivère, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). Journal of Machine Learning Research **22**(164) (2021), https://jmlr.org/papers/v22/20-303.html
28. Qian, R., Yin, X., Deng, C., Peng, Z., Xiong, J., Zhai, W., Dou, D.: Uground: Towards unified visual grounding with unrolled transformers. ArXiv **abs/2510.03853** (2025), https://api.semanticscholar.org/CorpusID:281843677

29. Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., Zhong, W., Li, K., Yang, J., Miao, Y., Lin, W., Liu, L., Jiang, X., Ma, Q., Li, J., Xiao, X., Cai, K., Li, C., Zheng, Y., Jin, C., Li, C., Zhou, X., Wang, M., Chen, H., Li, Z., Yang, H., Liu, H., Lin, F., Peng, T., Liu, X., Shi, G.: Ui-tars: Pioneering automated gui interaction with native agents. ArXiv **abs/2501.12326** (2025), https://api.semanticscholar.org/CorpusID:275788034
30. Seed, B.: Ui-tars-1.5. https://seed-tars.com/1.5 (2025)
31. Shi, T.T., Karpathy, A., Fan, L., Hernandez, J., Liang, P., et al.: World of bits: An open-domain platform for web-based agents. In: Proceedings of the 34th International Conference on Machine Learning (ICML) (2017), https://proceedings.mlr.press/v70/shi17a/shi17a.pdf
32. Shi, T., Karpathy, A., Fan, L.J., Hernández, J.Z., Liang, P.: World of bits: An open-domain platform for web-based agents. In: International Conference on Machine Learning (2017), https://api.semanticscholar.org/CorpusID:34953552
33. Song, Y., Xu, F.F., Zhou, S., Neubig, G.: Beyond browsing: Api-based web agents. In: Annual Meeting of the Association for Computational Linguistics (2024), https://api.semanticscholar.org/CorpusID:273507298
34. Team, T.Y.: Introducing navigator. https://yutori.com/blog/introducing-navigator (2025), yutori Blog
35. Trivedi, H., Khot, T., Hartmann, M., Manku, R.R., Dong, V., Li, E., Gupta, S., Sabharwal, A., Balasubramanian, N.: Appworld: A controllable world of apps and people for benchmarking interactive coding agents. ArXiv **abs/2407.18901** (2024), https://api.semanticscholar.org/CorpusID:271516633
36. W3C Web Accessibility Initiative (WAI): Diverse abilities and barriers. Web page (2024), https://www.w3.org/WAI/people-use-web/abilities-barriers/, accessed: 2026-03-05
37. Wang, H., Zou, H., Song, H., Feng, J., Fang, J., Lu, J., Liu, L., Luo, Q., Liang, S., Huang, S., et al.: UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning. arXiv preprint arXiv:2509.02544 (2025)
38. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191 (2024)
39. Wang, X., Wang, B., Lu, D., Yang, J., Xie, T., Wang, J., Deng, J., Guo, X., Xu, Y., Wu, C.H., Shen, Z., Li, Z., Li, R., Li, X., Chen, J., Zheng, B., Li, P., Lei, F., Cao, R., Fu, Y., Shin, D., Shin, M., Hu, J., Wang, Y., Chen, J., Ye, Y., Zhang, D., Du, D., Hu, H., Chen, H., Zhou, Z., Yao, H., Chen, Z., Gu, Q., Wang, Y., Wang, H., Yang, D., Zhong, V., Sung, F., Yang, Z., Yu, T.: OpenCUA: Open foundations for computer-use agents. ArXiv **abs/2508.09123** (2025), https://api.semanticscholar.org/CorpusID:280635573
40. Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Mellor, J., et al.: Taxonomy of risks posed by language models. In: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT) (2022), https://facctconference.org/static/pdfs_2022/facct22-3533088.pdf
41. World Health Organization (WHO): Disability and health. Web page (2023), https://www.who.int/news-room/fact-sheets/detail/disability-and-health, accessed: 2026-03-05
42. Wu, Y., Peng, Y., Chen, Y., Ruan, J., Zhuang, Z., Yang, C., Zhang, J., Chen, M., Tseng, Y., Yu, Z., Chen, L., Zhai, Y., Liu, B., Wu, C., Luo, Y.: Autowebworld: Synthesizing infinite verifiable web environments via finite state machines (2026), https://arxiv.org/abs/2602.14296

43. Xue, T., Qi, W., Shi, T., Song, C.H., Gou, B., Song, D.X., Sun, H., Su, Y.: An illusion of progress? assessing the current state of web agents. ArXiv **abs/2504.01382** (2025), https://api.semanticscholar.org/CorpusID:277502135
44. Yang, J., Zhang, H., Li, F., Zou, X., Li, C., Gao, J.: Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. arXiv preprint arXiv:2310.11441 (2023)
45. You, K., Zhang, H., Schoop, E., Weers, F., Swearngin, A., Nichols, J., Yang, Y., Gan, Z.: Ferret-ui: Grounded mobile ui understanding with multimodal llms. In: European Conference on Computer Vision (2024), https://api.semanticscholar.org/CorpusID:269005503
46. Zhou, S., Xu, F.F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., Neubig, G.: Webarena: A realistic web environment for building autonomous agents. ArXiv **abs/2307.13854** (2023), https://api.semanticscholar.org/CorpusID:260164780