VisualWebArena: EVALUATING MULTIMODAL AGENTS ON REALISTIC VISUAL WEB TASKS

Jing Yu Koh **Robert Lo*** **Lawrence Jang*** **Vikram Duvvur*** **Ming Chong Lim***
Po-Yu Huang* **Graham Neubig** **Shuyan Zhou** **Ruslan Salakhutdinov** **Daniel Fried**
Carnegie Mellon University
{jingyuk,rsalakhu,dfried}@cs.cmu.edu

ABSTRACT

Autonomous agents capable of planning, reasoning, and executing actions on the web offer a promising avenue for automating computer tasks. However, the majority of existing benchmarks primarily focus on text-based agents, neglecting many natural tasks that require visual information to effectively solve. Given that most computer interfaces cater to human perception, visual information often augments textual data in ways that text-only models struggle to harness effectively. To bridge this gap, we introduce VisualWebArena, a benchmark designed to assess the performance of multimodal web agents on realistic *visually grounded tasks*. VisualWebArena comprises of a set of diverse and complex web-based tasks that evaluate various capabilities of autonomous multimodal agents. To perform on this benchmark, agents need to accurately process image-text inputs, interpret natural language instructions, and execute actions on websites to accomplish user-defined objectives. We conduct an extensive evaluation of state-of-the-art LLM-based autonomous agents, including several multimodal models. Through extensive quantitative and qualitative analysis, we identify several limitations of text-only LLM agents, and reveal gaps in the capabilities of state-of-the-art multimodal language agents. VisualWebArena provides a framework for evaluating multimodal autonomous language agents, and offers insights towards building stronger autonomous agents for the web. Our code, baseline models, and data are publicly available at https://jykoh.com/vwa.

1 INTRODUCTION

Automating routine computer tasks with autonomous agents is a long standing goal of artificial intelligence research (Franklin & Graesser, 1996; Jennings et al., 1998). To achieve this, we need agents that can navigate computers effectively, process visual and textual inputs, handle high-level natural language instructions, and execute actions to achieve desired goals. As digital interfaces today are primarily built for human eyes, effective visual understanding is necessary for many routine computer tasks. For example, humans frequently perform tasks on the web which involve visual references, such as "Help me order a green polo shirt from Amazon", or rely on pictures rather than text to communicate. Productive work done on the computer is also often explicitly visual, e.g., working with Microsoft Excel sheets, creating presentations in Google Slides, or performing creative work in Adobe Photoshop. However, many agent benchmarks today focus on text-based tasks, neglecting the evaluation (and consequently the development) of multimodal autonomous agents.

To address this gap, we propose VisualWebArena (Fig. 1), a benchmark suite designed to rigorously assess and advance the visual and textual capabilities of autonomous agents. VisualWebArena builds off the WebArena (Zhou et al., 2024) framework, leveraging reproducible self-hosted environments and execution-based evaluations. VisualWebArena introduces a set of unique visual tasks, and emphasizes integrating visual understanding with language processing, closely simulating human interaction with modern computing interfaces. Our contributions are summarized as follows:

*Equal contribution.

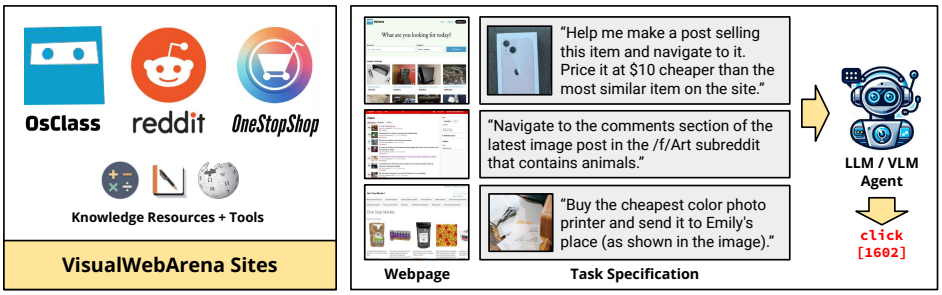

Figure 1: `VisualWebArena` is a benchmark suite of 910 realistic, visually grounded tasks on self-hosted web environments that involve web navigation and visual understanding.

- We introduce `VisualWebArena`, a set of 910 realistic tasks over three diverse web environments: Classifieds, Shopping, and Reddit. The Classifieds environment is a new contribution with real world data, while the Shopping and Reddit environments are inherited from WebArena. All tasks we introduce are *visually grounded*, and require visual understanding of webpage content to effectively solve (while WebArena does not). 25.2% of the tasks also include images as input (Fig. 1), and require understanding interleaved image-text inputs.
- We extensively benchmark the autonomous capabilities of state-of-the-art (SOTA) large language models (LLM) and vision-language models (VLMs), demonstrating that strong VLMs outperform text-based LLMs. The best VLM agents achieve a success rate of 16.4% on `VisualWebArena`, which is still significantly below human performance of 88.7%.
- We propose a new VLM agent inspired by Set-of-Marks prompting Yang et al. (2023a), simplifying the action space of the model. We show that this model substantially outperforms other baseline LLM agents, especially on sites that are more visually complex.

2 Related Work

Language-Guided Web Agent Benchmarks The development of reproducible environments for autonomous agents has seen considerable progress in recent years. Earlier efforts introduced reinforcement learning environments (Brockman et al., 2016), but extended into web domains (Shi et al., 2017; Liu et al., 2018). Recent web agent benchmarks introduced tasks involving actions on static internet pages (Deng et al., 2023) as well as interaction in simulated web environments (Yao et al., 2022; Zhou et al., 2024). AgentBench (Liu et al., 2023c) extends the scope of agents for computer interaction beyond the web, exploring database management and operating system functionalities.

LLM Agents There has been significant recent interest in using Large Language Models (LLMs) for developing autonomous agents (Xi et al., 2023; Wang et al., 2023a). State-of-the-art LLMs (Google, 2023; OpenAI, 2023; Chowdhery et al., 2023; Rae et al., 2021; Zhang et al., 2022; Touvron et al., 2023a;b; Jiang et al., 2023; 2024) based on the Transformer (Vaswani et al., 2017) architecture have demonstrated impressive abilities in learning from in-context examples (Brown et al., 2020; Chan et al., 2022), reasoning (Wei et al., 2022; Yao et al., 2023; Wang et al., 2023c; Besta et al., 2023), following instructions (Chung et al., 2022; Longpre et al., 2023; Ouyang et al., 2022), and operating over long-context sequences (Tay et al., 2021; Bertsch et al., 2023; Tworkowski et al., 2023). Several recent works leverage these abilities for building autonomous web agents: Kim et al. (2023) propose a recursive prompting method to improve GPT-4 performance on MiniWoB++ (Liu et al., 2018). Liu et al. (2023d) propose a method of orchestrating multiple LLM agents to improve performance on the WebShop (Yao et al., 2022) environment. Zeng et al. (2023) fine-tunes the LLaMA-2 models on interaction trajectories with instructions, improving over baseline agents.

Vision-Language Models Finally, our work builds off advances in vision-language models (VLMs), used for many multimodal tasks such as image captioning (Vinyals et al., 2015), visual question answering (Antol et al., 2015), and other benchmarks (Mialon et al., 2023; Yue et al., 2023; Tong et al., 2024). Frozen (Tsimpoukelli et al., 2021) was one of the first approaches to demonstrate the effectiveness of finetuning a visual encoder to map images into the embedding space of a LLM, introducing compelling few-shot multimodal abilities. Alayrac et al. (2022) introduced cross-attention layers and scaled up model sizes and training data. Wang et al. (2023b) introduced trainable

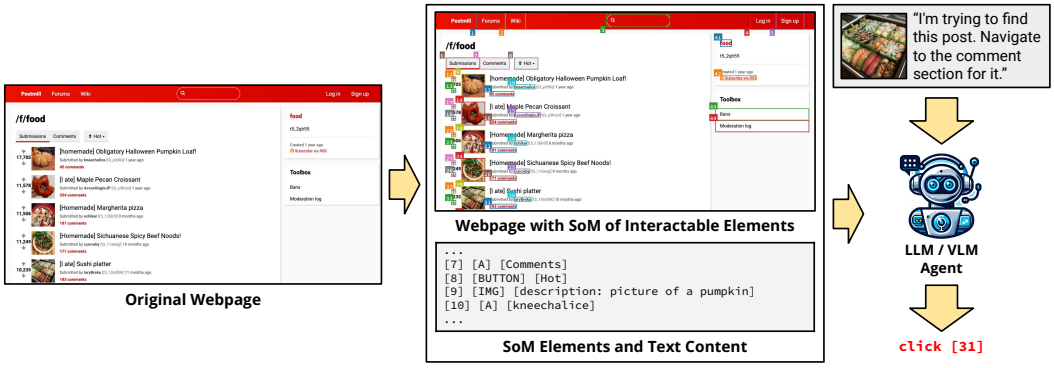

Figure 2: Set-of-Marks (Yang et al., 2023a) augmented webpage screenshot. Every interactable element is highlighted with a bounding box and a unique ID.

visual expert modules to improve vision-language fusion. Liu et al. (2023b) proposed fine-tuning on images paired with instructions to improve text generation performance on several multimodal tasks. GPT-4V (OpenAI, 2023) introduces visual processing to the GPT-4 family of models. Gemini (Google, 2023) is multimodal from the beginning (in contrast to post-hoc fine-tuned models), and can handle text interleaved with visual and audio inputs. Several recent work have also explored using VLMs to build agents for mobile platforms (Zhan & Zhang, 2023; Chu et al., 2023; Yang et al., 2023b) and the web (Gur et al., 2023; Hong et al., 2023). Zheng et al. (2024) is contemporaneous work which performs action grounding to identify appropriate HTML elements for enabling agents to execute actions. In contrast, our proposed SoM agent uses JavaScript to produce a Set-of-Marks (Yang et al., 2023a) for the VLM to directly use as an observation and action space.

3 VisualWebArena Environment

In order to ensure reproducibility, realism, and determinism, all websites in the VisualWebArena framework are provided as standalone self-hosted web applications. The textual and visual content are acquired from real world counterparts, while the code is based off open-source infrastructure commonly used in real websites. We formally define the environment, observation space, and action space below, but encourage readers to refer to WebArena (Zhou et al., 2024) for more details.

The environment and agent can be modeled as a partially observable Markov decision process (POMDP): $\mathcal{E} = (S, A, \Omega, T)$, where S represents the set of states, A represents the set of actions (Sec. 3.2), and Ω represents the set of observations (Sec. 3.1). The transition function is defined as $T : S \times A \rightarrow S$, with deterministic transitions between states conditioned on actions. At each time step t, the environment is in some state s_t (e.g., a particular page), with a partial observation $o_t \in \Omega$. An agent issues an action $a_t \in A$ conditioned on o_t, which results in a new state $s_{t+1} \in S$ and a new partial observation $o_{t+1} \in \Omega$ of the resulting page. The action a_t may be an action to be executed on the webpage (Tab. 1), or it may simply be a string output for information seeking tasks (Sec. 3.3).

Finally, we define the reward function $R : S \times A \rightarrow \{0, 1\}$ (Sec. 3.3) to measure the success of a task execution. In VisualWebArena, the reward function returns 1 at the final step if the state transitions align with the expectations of the task objective (i.e., the goal is achieved), and 0 otherwise.

3.1 Observation Space

The observation space Ω is modeled after a realistic web browsing experience. Observations include the webpage URLs, opened tabs (possibly multiple tabs of different websites), and the webpage content of the focused tab. In 25.2% of tasks, the intent also involves one or more input images (e.g., the first and third tasks in Fig. 1). The webpage content can be represented in several ways:

1. Raw web page HTML as a Document Object Model (DOM) tree, commonly used in previous work on autonomous web agents (Shi et al., 2017; Liu et al., 2018; Deng et al., 2023).

2. The accessibility tree,[1] which provides a structured and simplified representation of the webpage content that is optimized for assistive technologies. This is the primary representation that WebArena (Zhou et al., 2024) uses for its baseline LLM agents.
3. Web page screenshots, represented as RGB arrays, which has demonstrated efficacy in prior work on visual agents (Gur et al., 2023; Hong et al., 2023; Yan et al., 2023).
4. We introduce a new visual representation inspired by Set-of-Marks (SoM) prompting (Yang et al., 2023a). For every interactable element on the webpage, we label it with a bounding box and an ID (Fig. 2), producing a screenshot that allows a visual agent to reference elements on the page by their unique ID. We provide more details and analysis in Sec. 5.3.

3.2 Action Space

The full set of actions A is summarized in Tab. 1. The arguments for action a_t is the unique element ID from the current observation o_t. An advantage of this representation (over predicting (x, y) coordinates) is that it allows us to focus on high level reasoning rather than low-level control, as many SOTA VLMs and LLMs were not explicitly trained for referencing elements at such fine granularity. For the agents with accessibility tree representations, the argument is the element ID in the tree. For the SoM representation, we use the unique IDs assigned in the current page (see Fig. 2).

Action Type a	**Description**
`click [elem]`	Click on element `elem`.
`hover [elem]`	Hover on element `elem`.
`type [elem] [text]`	Type `text` on element `elem`.
`press [key_comb]`	Press a key combination.
`new_tab`	Open a new tab.
`tab_focus [index]`	Focus on the i-th tab.
`tab_close`	Close current tab.
`goto [url]`	Open `url`.
`go_back`	Click the back button.
`go_forward`	Click the forward button.
`scroll [up|down]`	Scroll up or down the page.
`stop [answer]`	End the task with an optional output.

Table 1: Set of possible actions A.

3.3 Evaluation

In order to evaluate performance on `VisualWebArena`, we introduce new visually grounded evaluation metrics to the functional evaluation paradigm of WebArena. These allow us to comprehensively evaluate the correctness of execution traces on open ended visually grounded tasks. The rewards for each task are hand designed functions using the primitives described below.

Information Seeking Tasks Information seeking tasks (e.g., the first task in Tab. 2) expect a string output $\hat{a}$ from the model. We mostly adopt similar reward functions introduced in WebArena for measuring text correctness against a groundtruth output a^*:

- `exact_match`: This can be defined as $\mathbf{1}_{\{\hat{a}=a^*\}}$ (where $\mathbf{1}$ represents the indicator function). Only outputs that are exactly equal to the groundtruth are given a score of 1. This is used in tasks where an exact response (e.g., a numerical answer) is expected.
- `must_include`: This reward function gives a score of 1 if all elements in a^* are contained in $\hat{a}$ and 0 otherwise. For example, if $\hat{a}$ = "\$1.99, \$2.50, \$10.00" and $a^* = \{$"1.99", "2.50", "10.00"$\}$, the task is awarded a score of 1 as all expected elements are present in the output. This is primarily used in tasks where we expect an unordered list of outputs, or we expect text output to contain a particular keyword.
- `must_exclude`: This is a new function we introduce, which is the converse of `must_include`. A reward of 0 is assigned if any element from a specified set a^* is found in $\hat{a}$ (and 1 otherwise). For instance, if $\hat{a}$ = "\$1.99, \$2.50, \$10.00" and $a^* = \{$"1.50", "2.00"$\}$, the reward is 1 as none of the prohibited elements are in the output.
- `fuzzy_match`: This function queries a LLM (in our implementation, `gpt-4-1106-preview`, to evaluate whether a^* and $\hat{a}$ are semantically equivalent. The LLM is prompted to output either "correct", "incorrect", or "partially correct", and we assign a reward of 1 if the output is "correct" and 0 otherwise.[2]

In addition, we also introduce several new visual functions for measuring open ended tasks:

[1] `https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree`

[2] We do not consider non-binary rewards in this work, but it would be a useful direction to explore in the future towards more continuous scales of performance.

Webpage / Input Image(s)	Example Intent	Reward Function $r(s, a)$ Implementation

	Add something like what the man is wearing to my wish list.	`url="/wishlist"`
		`locator(".wishlist .product-image-photo")`
		`eval_vqa(`s`, "Is this a polo shirt? (yes/no)", "yes")`
		`eval_vqa(`s`, "Is this shirt green? (yes/no)", "yes")`

	Create a post for each of the following images in the most related forums.	`eval_fuzzy_image_match(`s`, `a^*`)`

	Navigate to my listing of the white car and change the price to $25000. Update the price in the description as well.	`url="/index.php?page=item&id=84144"`
		`must_include(`$\hat{a}$`, "$25000 |OR| $25,000")`
		`must_exclude(`$\hat{a}$`, "$30000 |OR| $30,000")`

Table 2: Various evaluation metrics to assign reward $r(s, a) \in R : S \times A \rightarrow \{0, 1\}$. Our execution-based reward primitives allow us to benchmark many diverse, realistic, and open-ended tasks.

- `eval_vqa`: Similar to `fuzzy_match`, this function queries a VLM capable of performing visual question answering (VQA) (Antol et al., 2015). We use BLIP-2-T5XL (Li et al., 2023) in our implementation. We query the VLM with an image and a question. If the output of the VLM contains the groundtruth answer a^*, a reward of 1 is assigned.
- `eval_fuzzy_image_match`: This function checks whether a query image is similar to a groundtruth image according to the structural similarity index measure (SSIM) (Wang et al., 2004). If the SSIM between a query image and the groundtruth image is higher than a prespecified threshold $t \in [0, 1]$, a reward of 1 is assigned, and 0 otherwise.

Navigation and Actions Many tasks in `VisualWebArena` require navigating through multiple webpages, and executing actions to change the underlying state s of the environment. To accurately evaluate certain objectives, we require reward functions that examine the final webpage state to determine whether the task was successfully accomplished.

Each evaluator consists of a `locator` as well as a URL. The URL can be a specific page, or a function (e.g., the last page that the agent navigated to). The `locator` describes the object on the page that should be examined (e.g., all `img` elements, or all elements with the `.product-image-photo` class). During evaluation, we use the locator to retrieve the corresponding image or text content, and reuse the functions from the information seeking tasks to check for correctness.

4 Curating Visually Grounded Tasks

4.1 Web Environments

`VisualWebArena` is designed around three realistic web environments that involve visually rich content. Several tasks also require referencing information from a self-hosted Wikipedia knowledge base, and others involve interaction across more than one of these websites (Fig. 3).

Classifieds We introduce a new Classifieds website in `VisualWebArena`, inspired by real world marketplaces such as Craigslist and Facebook Marketplace. This new site provides a distinct environment compared to existing ones in WebArena, introducing visually grounded tasks centered around user interactions typical in online classifieds websites (posting, searching, commenting). The site's infrastructure uses OSClass, a robust open-source Content Management System (CMS) designed for classifieds websites, used in multiple real world sites. OSClass enables us to simulate functions such as search, posting new listings, commenting, and leaving reviews and ratings. The site contains 65,955 listings, each consisting of a title, a description, and a product image.

Shopping The Shopping site follows the e-commerce environment from WebArena (Zhou et al., 2024), with product information and content scraped from Amazon and released in WebShop (Yao et al., 2022). Visual understanding of product images is required for successfully navigating and completing tasks on e-commerce platforms, making this a natural choice for `VisualWebArena`.

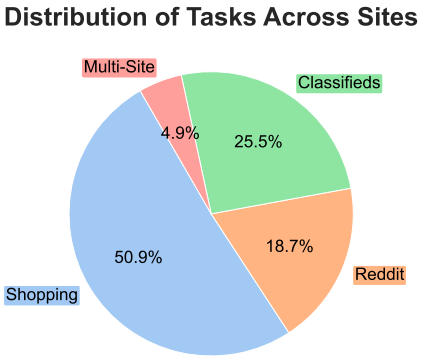

Figure 3: Tasks proportion by sites.

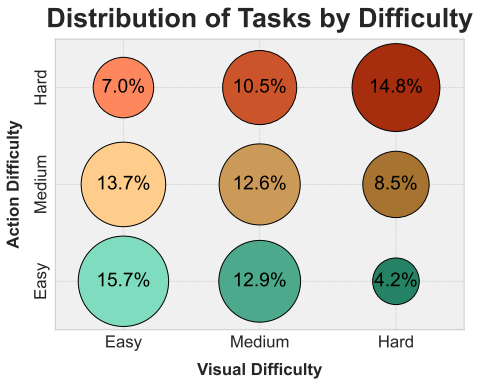

Figure 4: Tasks proportion by difficulty.

Reddit The Reddit site also follows the same environment from WebArena, and represents a social forum platform. The site contains 31,464 posts containing a diverse set of images across different subreddits and forums, such as natural images, memes, consumer electronics, and charts.

4.2 TASKS

Task Creation We introduce a set of 910 new tasks, split across the three sites detailed earlier (Fig. 3). We focus on curating realistic visually grounded tasks, following a similar process as task creation in WebArena. We start by curating *intent templates* (e.g., "Find me the {{`attribute`}} {{`item`}}. It should be between {{`range`}}."), which can be manually expanded by the annotator with different arguments to form multiple tasks (e.g., "Find me the cheapest red Toyota. It should be between $3000 to $6000."). The tasks were curated by 6 graduate students (co-authors of this paper). We encouraged annotators to be creative, and make use of the visual layouts of the websites, input images, and cross-site functionalities to develop creative and realistic tasks. When tasks involved input images, these were sourced from royalty-free, attribution-free sources, and MS-COCO (Lin et al., 2014). Annotators also wrote the reward functions using the primitives described in Sec. 3.3. We collected a total of 314 unique templates (average of 2.9 tasks per template).

While the majority of tasks can be solved, we also included a small subset (46 tasks, or 5.1%) which are unachievable. This subset tests the ability of agents to terminate early in the event where a task cannot be solved, which is essential in many real world scenarios.

Visually Grounded Tasks A key aspect of `VisualWebArena` is the inherent visual grounding of all tasks. Each task demands visual understanding, requiring agents to process and interpret visual information rather than relying solely on textual or HTML-based cues. This aligns closely with modern human-computer interfaces, where visual information (e.g., icons, colors) is often critical. For instance, a typical task might involve selecting a visually specific item, such as a "green polo shirt" where the color is visually discernible but not explicitly mentioned in text.

Task Complexity We classify each task into three difficulty levels: **easy**, **medium**, and **hard**. This classification is particularly useful for assessing performance across a spectrum of agents, ranging from smaller models to state-of-the-art LLMs and VLMs. We find in our analysis (Sec. 5) that many open-source models (e.g., LLaMA-2-70B, IDEFICS-80B) achieve a success rate of close to 0 on medium or hard tasks, but non-zero performance on easy tasks. This suggests that running open-source models on the easy subset would provide useful signal during development as well as faster iteration cycles (assuming performance between weaker and stronger agents are correlated).

We annotate both the *action* and *visual* difficulty of each task (breakdown in Fig. 4). The action difficulty is determined by the estimated number of actions that a human would need to complete the task. **Easy** tasks are defined as those that require three or fewer actions, **medium** tasks involve four to nine actions, and **hard** tasks demand ten or more. Visual difficulty is similarly segmented: **Easy** tasks involve basic visual identification of colors, shapes, and high-level object detection (e.g., recognizing the presence of a cat). **Medium** tasks require discerning patterns, semantic understanding, or OCR on large text of shorter lengths. **Hard** tasks involve multiple image inputs, OCR on small or lengthy text, or detection of fine details.

4.3 HUMAN PERFORMANCE

We measure the success rate of 7 college students on `VisualWebArena` tasks. Several of these students also assisted with task creation, and to avoid data leakage, we ensured that they were not assigned to the same tasks that they initially created. We sample one task per template, collecting a representative set of 230 tasks. We find that humans do well at this task, achieving an overall success rate of 88.7% (Tab. 3). The mistakes made in the remaining tasks are usually minor, such as not reading the task correctly or missing a part of the objective. For example, one task asked to add a particular item to the wishlist, but the human added it to the shopping cart instead. Another common failure mode was for tasks that required exhaustive search (e.g., "Find and navigate to the comments of this exact image."). Users were often unable to find the appropriate post after searching for 5-10 mins and gave up, assuming that the task was unachievable. In many shopping tasks, humans also did not look through all possible candidate pages to identify the cheapest or most highly reviewed product. We found these failure modes interesting, as they represent issues that strong agents would be well poised to handle, potentially achieving above human performance (and speed).

5 BASELINES

We run several baselines to benchmark the performance of state-of-the-art LLM and VLM agents. All models are prompt-based and provided with 3 in-context examples (one from each environment), which share no overlap with the benchmark tasks. The prompts we use are provided in the appendix. We summarize the results in Tab. 3 and describe the baselines in detail in the following sections.

5.1 TEXT-BASED LLM AGENTS

Several prior works developed strong autonomous agents by prompting text-only LLMs (Zhou et al., 2024; Kim et al., 2023; Liu et al., 2023d). We run several text-based LLM agents with Chain-of-Thought prompting Wei et al. (2022) over the accessibility tree representations of the websites as input. We leave more advanced prompting strategies for future work. Our baselines include API-based LLMs, including GPT-4 Turbo (`gpt-4-1106-preview`), GPT-3.5 Turbo (`gpt-3.5-turbo-1106`), and Gemini-Pro, as well as open sourced LLMs (LLaMA-2-70B and Mixtral-8x7B).

5.2 IMAGE CAPTION AUGMENTED LLM AGENTS

`VisualWebArena` is a visually grounded benchmark, and we expect that leveraging complementary visual information would improve performance. Hence, we run pretrained image captioning models on every `img` element on the HTML page, and augment the accessibility tree with this information as the image alt-text before passing this as input to the LLM agents. If a task contains input images, we also caption them and include the captions as part of the prompt. We run experiments on GPT-3.5 with two recent image captioning models, BLIP-2-T5XL (Li et al., 2023) and LLaVA-v1.5-7B (Liu et al., 2023a). Our results with GPT-3.5 as the LLM backbone ("Caption-augmented" section of Tab. 3) suggest that the LLaVA and BLIP-2 captioning models achieve comparable performance.

5.3 MULTIMODAL AGENTS

Finally, we benchmark strong API-based and open-source VLMs as agents. We evaluate several models capable of processing multiple interleaved image-and-text inputs: GPT-4V (OpenAI, 2023), Gemini-Pro (Google, 2023), IDEFICS-80B-Instruct (a reimplementation of Flamingo (Alayrac et al., 2022)), and CogVLM (Wang et al., 2023b). We experiment with two settings:

Image Screenshot + Captions + Accessibility Tree: This approach provides the accessibility tree representation augmented with image captions as accessibility tree alt-text from BLIP-2-T5XL (similar to the caption-augmented agent), as well as the screenshot of the current webpage as inputs.

Image Screenshot + Captions + SoM: Inspired by Set-of-Marks prompting Yang et al. (2023a), we perform an initial preprocessing step by using JavaScript to automatically annotate every interactable element on the webpage with a bounding box and a unique ID. The annotated screenshot containing bounding boxes and IDs, are provided as input to the multimodal model along with a text

Model Type	LLM Backbone	Visual Backbone	Inputs	Success Rate (↑)			
				Classifieds	Reddit	Shopping	Overall
Text-only	LLaMA-2-70B	-	Acc. Tree	0.43%	1.43%	1.29%	1.10%
	Mixtral-8x7B			1.71%	2.86%	1.29%	1.76%
	Gemini-Pro			0.85%	0.95%	3.43%	2.20%
	GPT-3.5			0.43%	0.95%	3.65%	2.20%
	GPT-4			5.56%	4.76%	9.23%	7.25%
Caption-augmented	LLaMA-2-70B	BLIP-2-T5XL	Acc. Tree + Caps	0.00%	0.95%	0.86%	0.66%
	Mixtral-8x7B	BLIP-2-T5XL		1.28%	0.48%	2.79%	1.87%
	GPT-3.5	LLaVA-7B		1.28%	1.43%	4.08%	2.75%
	GPT-3.5	BLIP-2-T5XL		0.85%	1.43%	4.72%	2.97%
	Gemini-Pro	BLIP-2-T5XL		1.71%	1.43%	6.01%	3.85%
	GPT-4	BLIP-2-T5XL		8.55%	8.57%	16.74%	12.75%
Multimodal	IDEFICS-80B-Instruct		Image + Caps + Acc. Tree	0.43%	0.95%	0.86%	0.77%
	CogVLM			0.00%	0.48%	0.43%	0.33%
	Gemini-Pro			3.42%	4.29%	8.15%	6.04%
	GPT-4V			8.12%	12.38%	**19.74%**	15.05%
Multimodal (SoM)	IDEFICS-80B-Instruct		Image + Caps + SoM	0.85%	0.95%	1.07%	0.99%
	CogVLM			0.00%	0.48%	0.43%	0.33%
	Gemini-Pro			3.42%	3.81%	7.73%	5.71%
	GPT-4V			**9.83%**	**17.14%**	19.31%	**16.37%**
Human Performance	-	-	Webpage	91.07%	87.10%	88.39%	88.70%

Table 3: Success rates of baseline LLM and VLM agents on `VisualWebArena`.

representation of the SoM (see Fig. 2). Similar to the baselines above, we also provide the captions from BLIP-2-T5XL for all `img` elements on the page. There have been several projects[3] that propose similar representations. Most have been proof-of-concept demos, and to the best of our knowledge, we are the first to systematically benchmark this on a realistic and interactive web environment.

6 Results and Analysis

Our main baseline results are summarized in Tab. 3. All existing models significantly underperform compared to humans, which indicate significant headroom in `VisualWebArena` for future work. We discuss some main findings below, with further analysis in the appendix.

Text-based LLMs Perform Poorly State-of-the-art text-only LLMs generally achieve poor results, with the best model (GPT-4) achieving an overall success rate of 7.25%. When we augment the LLMs with captions, this considerably improves success rate (7.25% to 12.75% for GPT-4).

Multimodality Helps Using multimodal agents significantly improves the success rate: GPT-4V (`gpt-4-1106-vision-preview`) achieves an overall success rate of 15.05%, substantially improving over the text-only agents. emini-Pro also experiences a significant uplift in success rate, from 3.85% (caption-augmented) to 6.04% (multimodal). Text-based agents may be limited in their ability to process complex images (e.g., those that require OCR or recognition of non-salient objects).

SoM Improves Navigability We observe that the SoM representation (Sec. 5.3) further improves the performance of GPT-4V over using the accessibility tree, boosting overall success rate ($15.05\% \rightarrow 16.37\%$). We observe particularly substantial improvements on Classifieds and Reddit, from $12.38\% \rightarrow 17.14\%$ and $8.12\% \rightarrow 9.83\%$ respectively. We attribute this to the Classifieds and Reddit websites containing denser visual content. These websites often contain multiple smaller sized images that are arranged very closely (Fig. 2). In many of these pages, the accessibility tree does not provide sufficient information to disentangle elements that are spatially close. We hypothesize that the SoM representation is superior for strong VLM agents, which can more accurately disentangle and click on the desired elements. For the other VLMs, SoM does not significantly improve success rates, which we attribute to the finding from Yang et al. (2023a) that only GPT-4V demonstrates this emergent grounding ability (perhaps due to scale or training data).

One GPT-4V + SoM execution trajectory that we found particularly compelling was Reddit task #139, which requires exact image matching to find a post and block a user (Fig. 5). The model initially attempts to search for the correct forum, and when this fails it navigates to the list of forums.

[3]For example, GPT-4V-ACT and vimGPT propose similar interfaces.

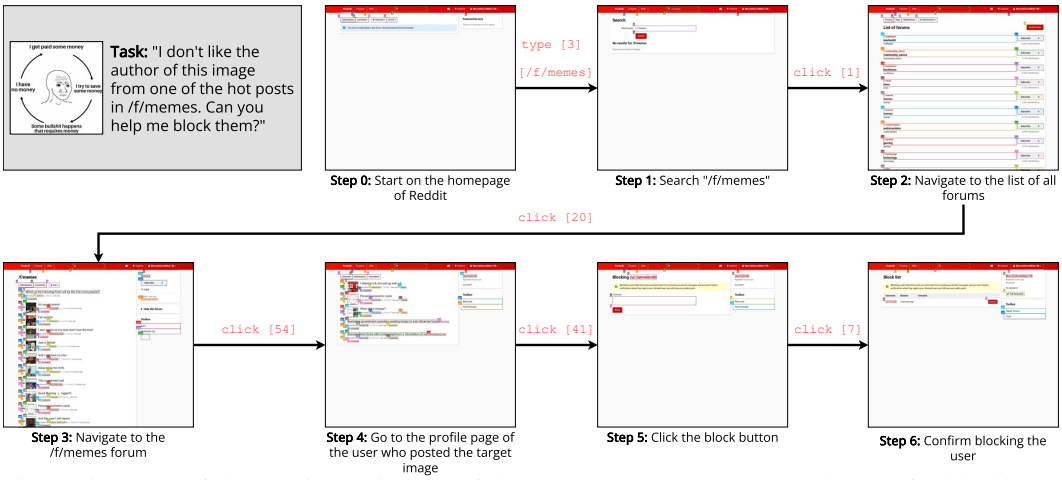

Figure 5: Successful execution trajectory of the GPT-4V + SoM agent on the task for blocking a user that posted a certain picture. The text in red represents the actions output by the agent.

After navigating correctly to `/f/memes`, it identifies the offending image out of the many images on the page (Step 3 in Fig. 5) and blocks the author efficiently without any unnecessary actions.

6.1 Performance by Task Type

We analyze the success rate of the best VLM agent baseline (GPT-4V + SoM) across several additional subsets of tasks (Tab. 4). We include further analysis for other models in the appendix.

OCR Tasks 17.1% of `VisualWebArena` require optical character recognition (OCR), such as reading text from product images, or extracting text from an input image. We find that GPT-4V + SoM generally performs worse on tasks that require OCR (13.4%) compared to tasks which do not (16.9%), suggesting that OCR may be a bottleneck for current agents.

Task Subset	% of Total	SR (↑)
OCR required	17.1%	13.4%
No OCR required	82.9%	**16.9%**
Exact image match	8.7%	**18.9%**
No exact image match	91.3%	16.2%
Image inputs	25.2%	**19.0%**
No image inputs	74.8%	14.9%

Table 4: Success rate (SR) of GPT-4V (SoM) across different types of tasks.

Exact Image Match 8.7% of tasks require exact image matching, which requires agents to identify precise visual matches. GPT-4V + SoM achieves a slightly higher success rate on this subset (18.9%) compared to other tasks (16.2%), suggesting that exact image matching is not a primary bottleneck.

Image Input Tasks 25.2% of `VisualWebArena` include one or more input images as part of the objective. These tasks generally appear more tractable for the GPT-4V + SoM agent, and it achieves a higher success rate (19.0%) compared to tasks without image inputs (14.9%).

7 Conclusion

In this work, we introduced `VisualWebArena`, a benchmark of realistic tasks designed to rigorously evaluate and advance the capabilities of autonomous multimodal web agents. `VisualWebArena` represents a significant step towards addressing the gap in the evaluation of multimodal agents on visually grounded tasks. We also introduce a visual agent inspired by Set-of-Marks prompting, and demonstrate the potential of this approach for simplifying action spaces and improving performance on visually complex websites. Our extensive evaluation of state-of-the-art LLM and VLM agents demonstrate that while VLMs show promise, there remains a considerable performance gap compared to humans, who achieve very high success rates on `VisualWebArena`. Our quantitative and qualitative analysis also highlights several common failure modes of existing LLM and VLM agents. We expect future work on improving the reasoning, visual understanding, and planning abilities of agents to be particularly exciting and promising areas.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *NeurIPS*, 2022.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R Gormley. Unlimiformer: Long-range transformers with unlimited length input. *NeurIPS*, 2023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

Stephanie CY Chan, Ishita Dasgupta, Junkyung Kim, Dharshan Kumaran, Andrew K Lampinen, and Felix Hill. Transformers generalize differently from information stored in context vs in weights. *NeurIPS MemARI Workshop*, 2022.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *JMLR*, 2023.

Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *NeurIPS*, 2023.

Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International workshop on agent theories, architectures, and languages*, pp. 21–35. Springer, 1996.

Gemini Team Google. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *ICLR*, 2020.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*, 2023.

Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1:7–38, 1998.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *NeurIPS*, 2023.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *ICML*, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *ICLR*, 2018.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023b.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023c.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960*, 2023d.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *ICML*, 2023.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *ICML*, 2017.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *ICLR*, 2021.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. *arXiv preprint arXiv:2401.06209*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *NeurIPS*, 2021.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *NeurIPS*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023a.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ICLR*, 2023c.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

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

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023a.

Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023b.

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

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

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv preprint arXiv:2311.16502*, 2023.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*, 2023.

Zhuosheng Zhan and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *ICLR*, 2024.

APPENDIX

We provide further analysis on model failure modes for Gemini and GPT-4 (Sec. A), more details on the new Classifieds environment (Sec. B), and on the task collection process (Sec. C).

A FURTHER ANALYSIS

A.1 FEW-SHOT PROMPTING

# Examples	Success Rate (↑)			
	Classifieds	**Reddit**	**Shopping**	**Overall**
0	4.29%	2.38%	0.43%	2.86%
1	5.36%	1.43%	2.14%	3.63%
3	**8.15%**	**4.29%**	**3.42%**	**6.04%**

Table 5: Performance with different number of in-context examples.

In most of our main experimental results, we prompt the model with 3 in-context examples. We perform an analysis of the success rate against the number of in-context examples provided (Tab. 5). For 1-shot experiments, we provide the model with the single in-context example from its corresponding environment. All experiments are run with the multimodal Gemini-Pro model (as GPT-4V is prohibitively expensive) with the Image + Caption + Acc. Tree as the observation space.

We observe that overall success rate tends to increase with the number of examples provided, with a significant jump from 1 to 3 in-context examples. The improved results with a greater number of examples suggest that the performance of the VLM agents may improve significantly if we fine-tune the models on web trajectories, which will be an exciting direction for future work.

A.2 PERFORMANCE BY TASK DIFFICULTY

Visual Difficulty (v)

Action Difficulty (a)

a \v	Easy	Medium	Hard	Overall
Easy	18.9%	11.1%	10.5%	14.8%
Medium	1.6%	6.1%	7.8%	4.7%
hard	1.6%	4.2%	1.5%	2.4%
Overall	9.0%	7.3%	4.8%	7.3%

(a) Success rate of GPT-4 Text-only

a \v	Easy	Medium	Hard	Overall
Easy	23.1%	18.8%	13.2%	20.1%
Medium	14.4%	9.6%	5.2%	10.4%
Hard	7.8%	7.3%	8.1%	7.8%
Overall	16.9%	12.2%	8.0%	12.7%

(b) Success rate of GPT-4 + Captions

a \v	Easy	Medium	Hard	Overall
Easy	30.1%	20.5%	26.3%	25.8%
Medium	15.2%	11.3%	11.7%	12.9%
hard	14.1%	10.4%	8.9%	10.5%
Overall	21.4%	14.3%	12.4%	16.4%

(c) Success rate of GPT-4V + SoM

a \v	Easy	Medium	Hard	Overall
Easy	6.0	7.7	6.1	6.9
Medium	10.4	10.6	7.2	10.0
Hard	14.1	9.2	12.5	12.1
Overall	9.5	9.4	10.2	9.6

(d) Trajectory length of GPT-4V + SoM

Figure 6: Success rates (a, b, c) and trajectory lengths (d) across different difficulty levels.

We conduct an analysis of the GPT-4 models across different action and visual difficulty levels (Fig. 6). We observe that success rate generally decreases as action/vision difficulty increases, which makes intuitive sense based on the difficulty taxonomy described in Sec. 4.2. The findings also show that multimodal models perform better especially on hard visual tasks. On this subset, GPT-4V + SoM achieves an average success rate of 12.4%, which is significantly higher than that of the caption-augmented (8.0%) and the text-only agents (4.8%). In addition to success rates, the GPT-4V trajectory lengths also increased with action difficulty, with harder tasks requiring more steps.

A.3 TASK SUBSET ANALYSIS

In this section, we provide more fine-grained analysis across different task subsets, similar to the one in Sec. 6.1 of the main paper. We examine both the GPT-4 text and multimodal agents, as well as the Gemini-Pro agents. This analysis may provide useful insights towards capabilities that future VLM models should have to perform well on web navigation tasks (specifically, OCR, exact image matching, and handling multiple interleaved image and text inputs).

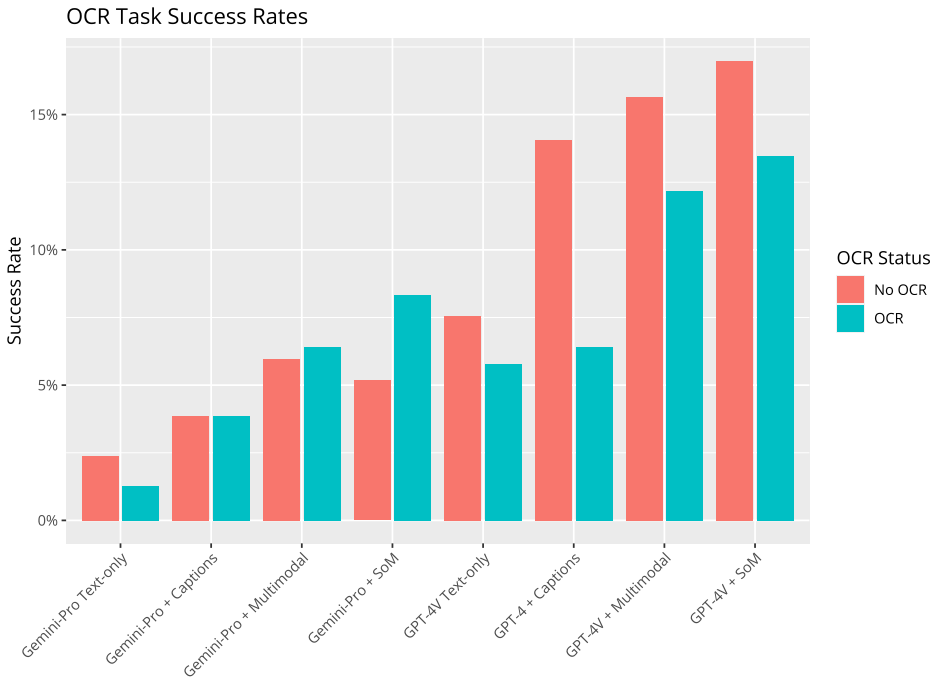

Figure 7: Success rate of GPT-4 and Gemini agents on tasks that do not require OCR vs. tasks that do.

OCR Tasks On OCR tasks, which take up 17.1% of the benchmark, we observe that the GPT-4 family of models achieve a lower success rate on tasks that require OCR compared to tasks that do not (Fig. 7). This is consistent with the findings for GPT-4V + SoM reported in Sec. 6.1 of the main paper. We also observe that introducing multimodality (over just captions) substantially improves performance on OCR tasks (from 6.4% to 12.2%), showcasing the importance of having multimodal models for text recognition capabilities, as captioning models generally do not capture such finegrained information.

For Gemini-Pro agents, we also observe similar trends, with the multimodal and SoM models achieving a higher than proportionate gain on the OCR subset (compared to the non-OCR subset). Interestingly, the multimodal Gemini-Pro agents achieve a higher success rate on tasks that require OCR compared to tasks that do not. These results may suggest that it has strong inherent OCR capabilities, which we believe will be useful to explore in future work (especially on the stronger Gemini-Ultra model once it is generally available).

Exact Image Match Of the tasks in `VisualWebArena`, 8.7% require exact image matching, which tests the ability of agents to identify images that have the exact same content (in contrast to those that are just semantically similar). From Fig. 8, we observe that the GPT-4V SoM model achieves a higher succeess rate on tasks that expect exact image match, while the other GPT-4 agents achieve a relatively lower success rate on the exact match subset. This suggests that the SoM representation may be more optimal for exact image match, due to its visual-centric observation and action space.

For the Gemini models, we observe that success rates on exact match tasks are substantially lower than success rates on non-exact match tasks. Interestingly, we also observe a similar trend as the GPT-4 agents, where introducing multimodality improves success rates on exact match tasks, which is further bolstered with the SoM representation.

Image Input Tasks 25.2% (229 tasks) in `VisualWebArena` are specified with image inputs (e.g., the task in Fig. 5, and the first and third tasks in Fig. 1). The results of the Gemini-Pro and GPT-4 agents are summarized in Fig. 9.

We observe that for the GPT-4 agent, success rates are generally higher on tasks that involve image inputs, with the exception of the text-only agent. This aligns with intuition, as agents that do not

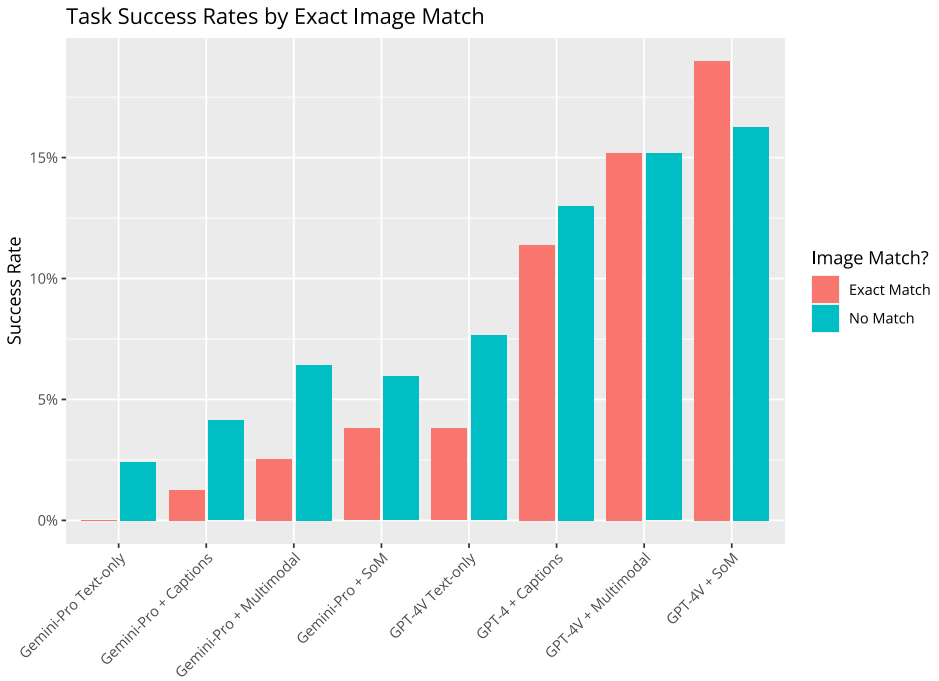

Figure 8: Success rates of agents on tasks that require exact image match vs. those that do not.

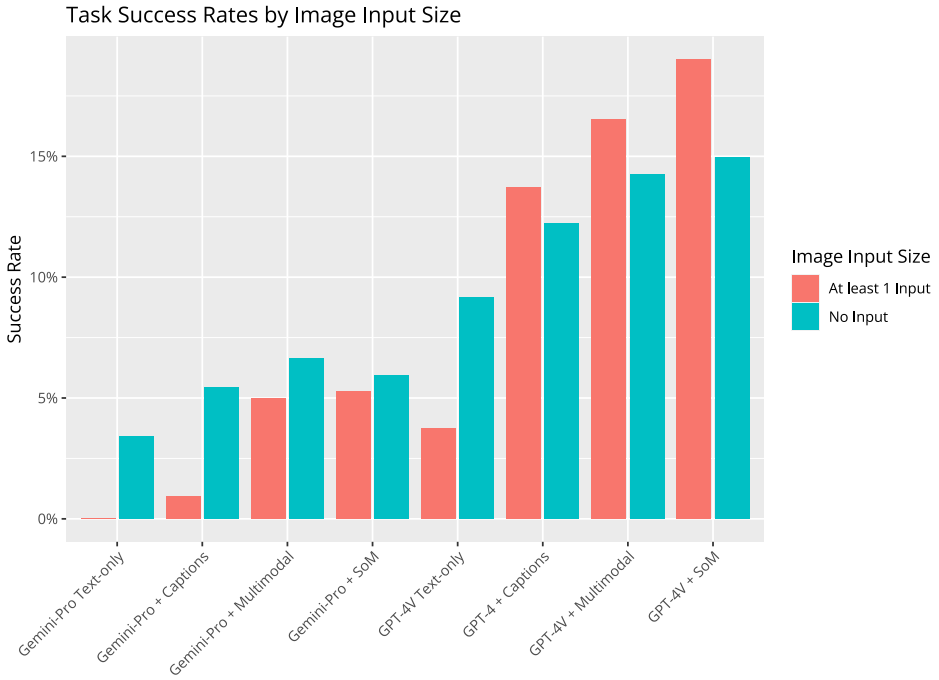

Figure 9: Success rates of agents on tasks that include input images as part of the specification vs. tasks that are specified with just written text.

have access to visual information would not be able to understand the task correctly, and would perform worse at successfully accomplishing it. For the captioning, multimodal, and SoM GPT-4 agents, success rates are higher on the tasks involving image input, which we attribute to these tasks being more tractable once the visual content is correctly understood.

Interestingly, we see a contrast with the Gemini-Pro agents, where success rate is generally lower on tasks that involve input images. This may imply that the model may not be able to process multiple

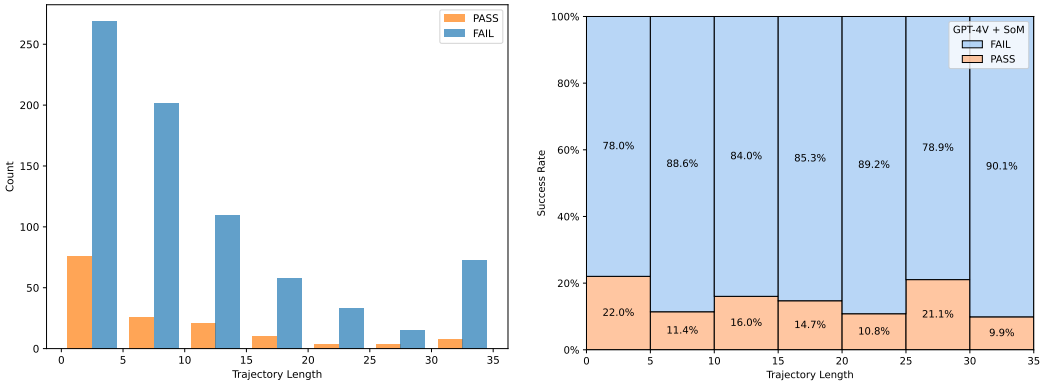

Figure 10: Performance of the GPT-4V + SoM agent across different trajectory lengths.

interleaved image-text inputs as well. This may be useful to revisit in the future with the stronger Gemini-Ultra model once it is released, or with stronger open sourced VLMs. We believe that being able to handle interleaved multimodal inputs will be a core requirement for strong web agents, and more comprehensive error analysis with stronger models may yield useful insights.

Trajectory Lengths vs. Success Rates Hard reasoning tasks, on average, require more steps to be successfully solved. We plot the trajectory length of the GPT-4V + SoM model in Fig. 10. The findings suggest that the model assumes a significant portion of tasks can be completed in a few steps, as it terminates a majority of tasks after less than 10 steps. However, this assumption doesn't imply that the model successfully solves the majority of tasks: the error rate remains relatively uniform across longer trajectory lengths.

A.4 Failure Modes

In this section, we describe other common issues we observed with our baseline agent models.

Failure Over Longer Horizons We observed that in several examples, the agents would correctly perform a task but undo it, leading to failure. The GPT-4 captioning-only model on shopping task 54 ("Add the one [poster] with waves to my wish list.") made an assumption that the product image with a caption about a lighthouse was the correct one, and added it to the wishlist. However, after going to the wish list page the agent removes the poster because "there is no explicit mention of waves in the current items listed on the Wish List page." This issue is not unique to the text input agents; even the GPT-4 SoM agent faced a similar problem in shopping task 397 ("Buy the item on the page with a banana theme."). The agent initially added the correct item to the shopping cart and proceeded to check out, but stopped in the middle stating in the reasoning trace output that it does not think the item fit the criteria (despite having added it to the cart just a few steps ago).

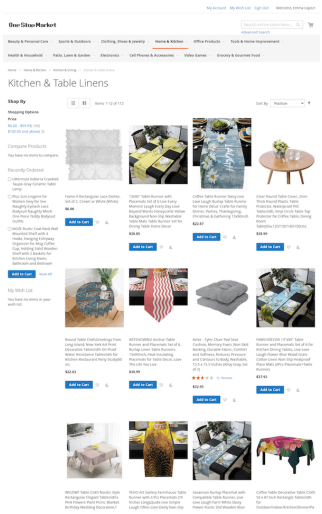

Figure 11: The starting page for the task "Add the red one in the second row of this page to my shopping cart."

Failures on Easy Tasks We observed surprisingly poor performance on many tasks with easy action and easy visual difficulty levels, such as in shopping task 46, which tasks the agent to add the red product in the second row to the cart (starting on the page shown in Fig. 11). The multimodal and SoM GPT-4V agents clicked on a blue tablecloth in the first row and gave up when they couldn't find an option to order it in red. Despite appearing to be a simple task (the correct product is the red cloth in the second row), none of the agents we benchmarked were able to successfully complete it.

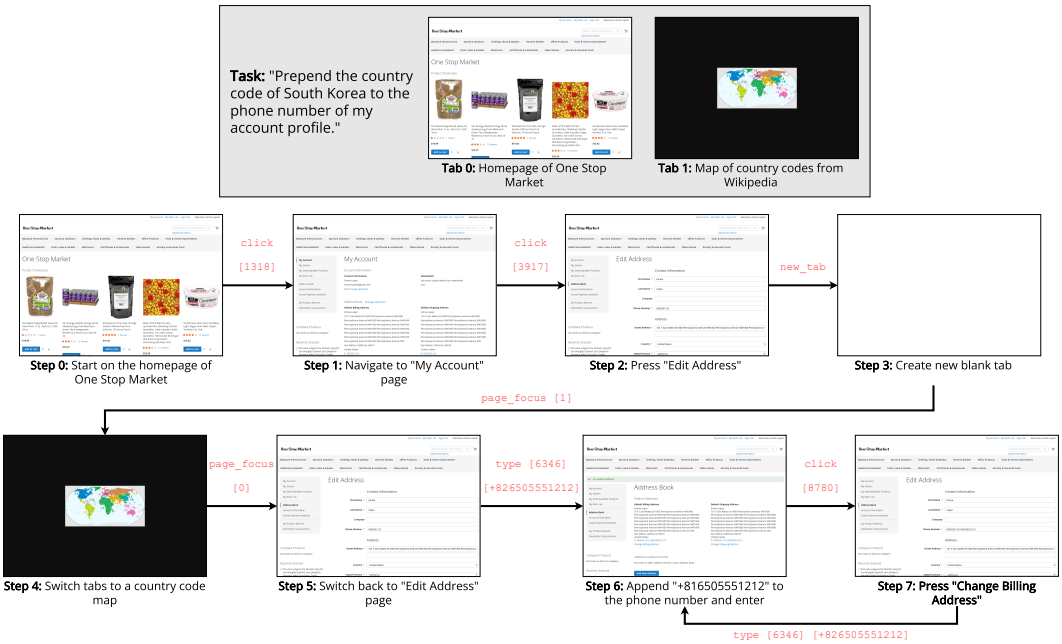

Figure 12: Unsuccessful execution trajectory of the GPT-4V multimodal agent on the task for adding the a country code to the user's phone number. The text in red represents the commands output by the agent.

Giving Up Too Early Another frequent issue we observed that occurred across all the agents was giving up too early. For example, GPT-4V + SoM fails on shopping task 248 ("Order a 6 pack of the green chocolate bars. If the shipping is more than 7% of the total price, leave a 3 star review mentioning it, otherwise 5."). This tasks involves several steps which the model is able to correctly plan out, but the very first action needed is to slightly scroll down so the "add to cart" button is visible. However, even after identifying the correct product the model gives up on the first step instead of scrolling, because it does not immediately see the button. There are other instances of this occurring, such as in shopping task 175, where an agent will use the search bar to search for something, and then immediately give up because it does not see the target product instead of trying new ways to find the product.

Getting Stuck in Loops Another issue we observed was oscillating or looping between pages, where the agent would look something up or navigate to a page, unsuccessfully attempt to perform the next action (such as adding it to the cart), and on failure it goes back and repeats from the beginning. An example of this is in classifieds task 205 where the model is tasked to compare two makeup palettes in two tabs, and the GPT-4V agent spends most of the time switching between the tabs. We believe that these issues will likely be alleviated by introducing more sophisticated tracking of past states and execution history, which is a promising direction for future work.

Failure Example: Changing User Phone Number Shopping task #345, a multi-site task that also involves the Wikipedia site, demonstrated several interesting points of failure that we saw throughout the execution traces for many other tasks. Fig. 12 contains the execution trace of the GPT-4V multimodal agent for the task "Prepend the country code of South Korea to the phone number of my account profile." There are three major mistakes made by the agent in this execution trace:

- **Useless actions:** In step 3 of the trajectory, the agent creates a new blank tab and does not interact with it for the rest of the trajectory. While this does not impact the correctness of the final task, it does show that the agents sometimes take unnecessary steps.
- **Appending text instead of replacing:** Many agents added text to input fields without deleting the previous text, which would often result in long, repeating search queries or addresses. An example of this occurs in step 7 of Fig. 12.

- **Repeating actions:** Another frequent issue we saw across agents was repeating actions, like how the agent kept jumping between step 6 and step 7 of Fig. 12 until it hit the maximum trajectory length. In this case, we believe this looping effect stems from the issue mentioned above and each time the agent tries to correct the phone number, it keeps appending the correct number instead of replacing the incorrect number with the correct number.

A.5 Comparison Between Agents

In this section, we describe some qualitative differences we observed between the different agents on various tasks in `VisualWebArena`.

Text-only vs. Caption-augmented Agents For GPT-4, the text model unsurprisingly performs much worse than even the captioning model, failing to even do the most basic tasks. For example, Reddit task #101 is the relatively simple task to "Navigate to the comments section of a post that contains a picture of a keyboard." Out of all of the GPT-4 baseline agents, the text-only agent is the only one to fail this task, as it's unable to identify the appropriate post from just the title. Interestingly, it still manages to make an educated guess and navigate to the hottest post on /f/MechanicalKeyboards (which unfortunately, did not include a keyboard in its image).

Caption-augmented vs. SoM Agents We observed in many examples that the GPT-4V SoM and multimodal agents outperformed the caption-augmented baselines in terms of navigation capabilities and visual understanding. The multimodal models were generally better at understanding visual information on webpages, as relevant information in many images is lost when they are translated into captions. One pertinent example is Reddit task #40, where a picture of the skyline of Pittsburgh is provided, and the task is "I'd like to find the subreddit for the city this photo was taken in. Can you navigate to it?". The GPT-4V + SoM agent correctly identifies the location of the photo, with the first line of its reasoning output as *"The photo shows a city skyline with prominent buildings labeled with logos for UPMC and PNC, which suggests that the photo was taken in Pittsburgh, Pennsylvania."*. Using this information, the agent is able to successfully navigate to the appropriate subreddit, /f/pittsburgh. In contrast, the captioning agent labels the image as "city skyline with many tall buildings" (as this is the output from the BLIP-2 model), which prevents it from identifying the appropriate subreddit. This highlights a fundamental issue with captioning models: they frequently highlight salient visual information, which hinders success on many tasks that require fine-grained visual information not typically captured by captions.

Multimodal vs. SoM Agents The SoM representation generally performs better on tasks that require more navigation steps, due to its simplified observation and action space. One example is classifieds task #31, "Find the latest listing of a white Google Pixel phone and post a comment offering $10 less than their asking price." (Fig. 13). While the multimodal model was unable to search for the correct terms, the SoM model was able to leverage the simplified action space to traverse more efficiently throughout the environment. It succeeded at this task by filtering for cell phones after the initial search for more relevant results, and managed to fill out the necessary comment form fields. From our observations, the SoM representation is generally more efficient compared to the multimodal representation (which only has access to the page screenshot and accessibility tree). With a strong VLM capable of SoM, the agent does not have to implicitly perform visual co-referencing to match elements from the accessibility tree to the visual buttons and inputs that it wants to interact with.

B The Classifieds Environment

The Classifieds environment contains 65,955 listings, each with a title, text description, and a product image of the item being sold. To populate the site with realistic content, we scraped data across a variety of categories on Craigslist over 3 weeks, focusing on the Northeastern States of the US (similar to the geographic region in the Reddit site). This approach ensured a diverse and rich dataset, representative of real-world classifieds posts. We utilized the `scrubadub` Python package for redacting Personally Identifiable Information (PII), including addresses, phone numbers, and emails. We use generated placeholders for names (e.g., "Bill Smith"), emails with fictitious addresses (e.g., `bill_smith@example.com`), and phone numbers with the fictional 555-prefix numbers.

Fig. 14 and 15 show two pages within the Classifieds site, the homepage and the detail page of a particular listing. Users can also use the search function, or filter posts by category or location to find items.

C Task Collection Process

Our main task collection process is described in Sec. 4.2. We collected the set of 910 tasks by recruiting 6 computer science graduate students (co-authors of this paper), who were all familiar with commercial versions of the Classifieds, Shopping, and Reddit sites, and have used them in their personal lives.

Annotators were first instructed to spend some time exploring the `VisualWebArena` websites, to familiarize themselves with their functionality and content (as this may differ slightly from real world implementations). During task creation, we encouraged annotators to be creative, and make use of the visual layouts of the websites, input images, and cross-site functionalities to develop creative and realistic tasks. We ensured that there were no repeated tasks, and that there were not too many tasks of the same type (by first producing templates, followed by instantiating them with different arguments to create multiple tasks, as described in Sec. 4.2).

D Baseline Agent Settings

For all baseline agents we report in the paper, we use a webpage viewport size of 1280×2048, and truncate text observations to 3840 tokens (or 15360 characters for Gemini). For models with shorter context windows (e.g., LLaMA, IDEFICS, CogVLM), we instead use a viewport size of 1280×720 and truncate text observations to 640 tokens. For GPT-3.5 and GPT-4 models, we follow Zhou et al. (2024) in using a temperature of 1.0 and a top-p of 0.9. For Gemini models we use the suggested default temperature of 0.9 and top-p of 1.0. For the remaining models, we find that they benefit from sampling from lower temperatures, and use a temperature of 0.6 and top-p of 0.95. Nucleus sampling (Holtzman et al., 2020) is used in all experiments.

The system message and the prompt with in-context examples for the baseline SoM agents are shown in Fig. 16 and Fig. 17 respectively. We prompt the model with 3 in-context examples for all baselines. For multimodal and SoM models, we include the screenshot of each in-context example as well as the screenshot of the current page. For text-only and caption augmented models, the examples consist of just the text and captions.

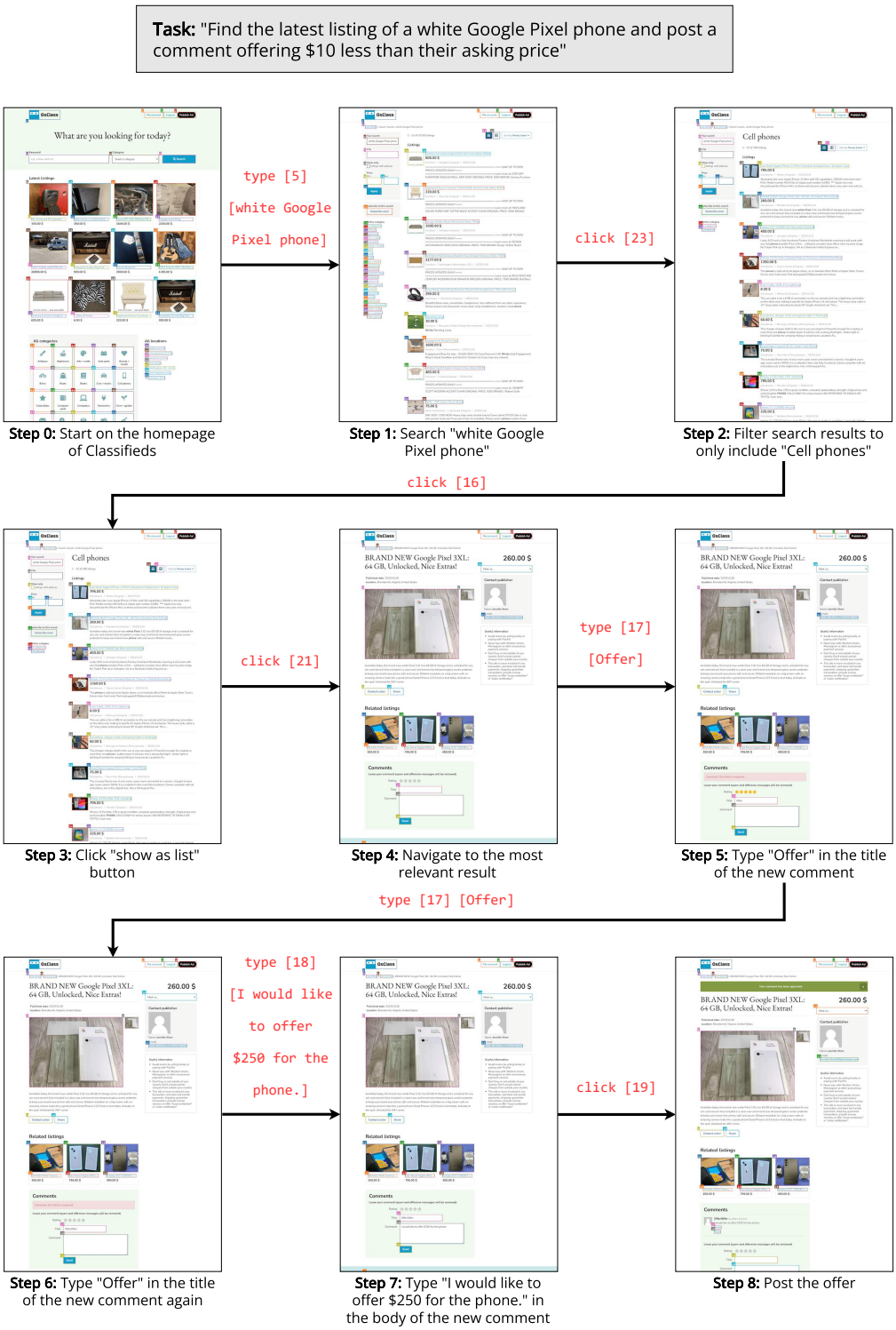

Figure 13: GPT-4V SoM agent on Classifieds task #31, "Find the latest listing of a white Google Pixel phone and post a comment offering $10 less than their asking price.". It succeeds at the task by leveraging the more efficient navigation space.

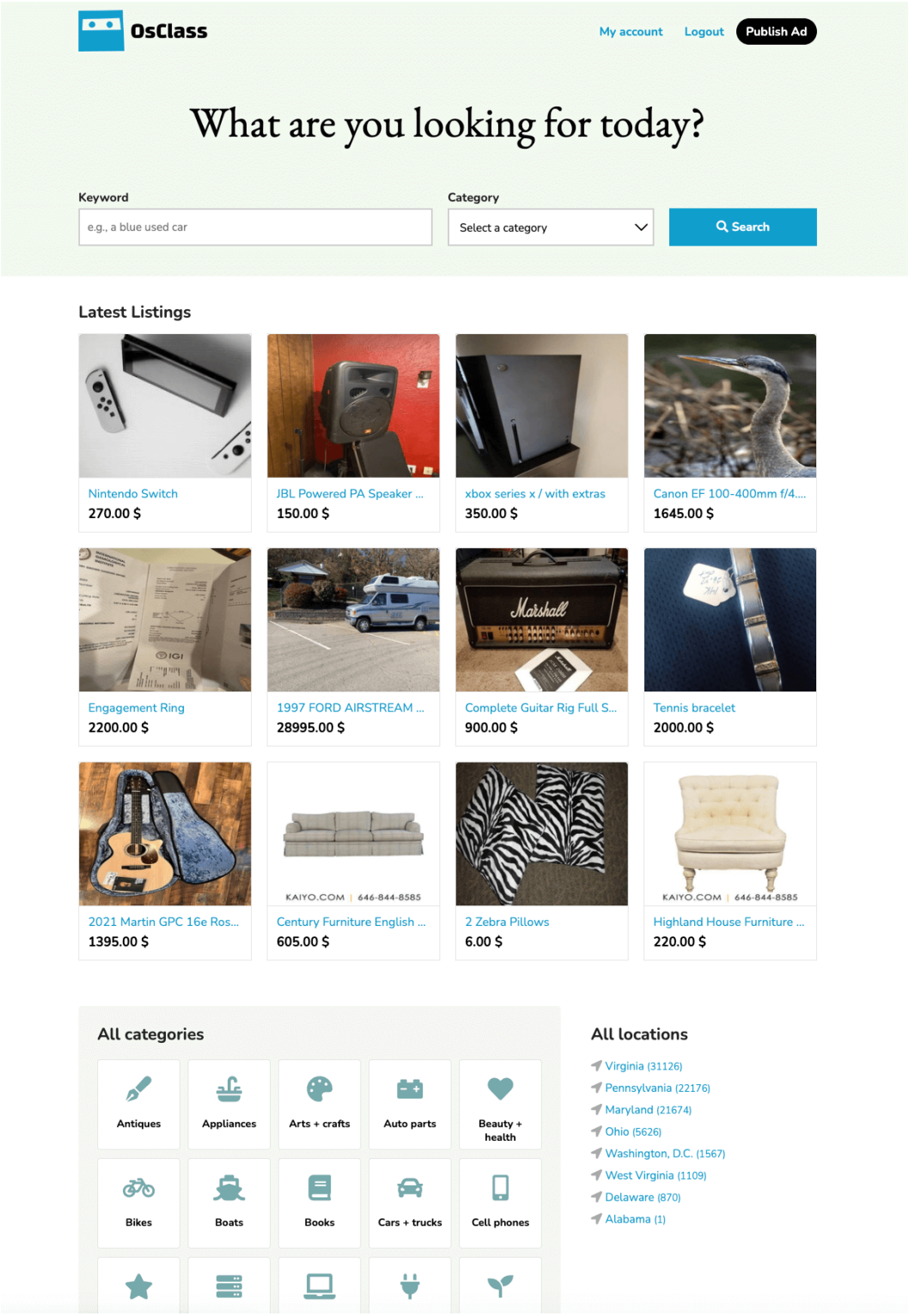

Figure 14: Homepage of the Classifieds site. Users can search for keywords, filter by category, or post location.

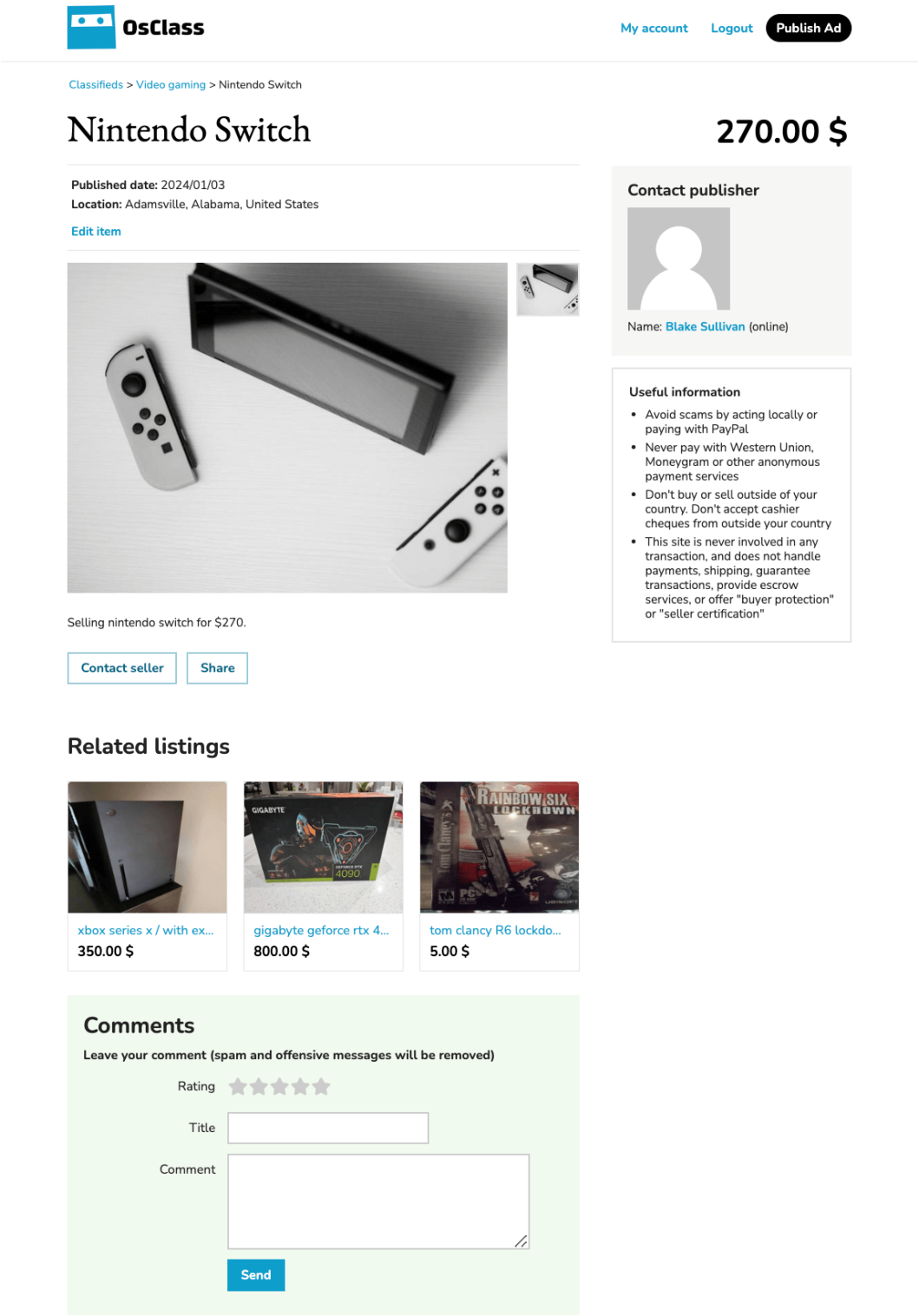

Figure 15: Example post in the Classifieds website. Users can add comments and reviews to individual listings.

You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks. These tasks will be accomplished through the use of specific actions you can issue.

Here's the information you'll have:
The user's objective: This is the task you're trying to complete.
The current web page screenshot: This is a screenshot of the webpage, with each interactable element assigned a unique numerical id. Each bounding box and its respective id shares the same color.
The observation, which lists the IDs of all interactable elements on the current web page with their text content if any, in the format [id] [tagType] [text content]. tagType is the type of the element, such as button, link, or textbox. text content is the text content of the element. For example, [1234] [button] ['Add to Cart'] means that there is a button with id 1234 and text content 'Add to Cart' on the current web page. [] [StaticText] [text] means that the element is of some text that is not interactable.
The current web page's URL: This is the page you're currently navigating.
The open tabs: These are the tabs you have open.
The previous action: This is the action you just performed. It may be helpful to track your progress.

The actions you can perform fall into several categories:

Page Operation Actions:
```click [id]```: This action clicks on an element with a specific id on the webpage.
```type [id] [content]```: Use this to type the content into the field with id. By default, the "Enter" key is pressed after typing unless press_enter_after is set to 0, i.e., ```type [id] [content] [0]```.
```hover [id]```: Hover over an element with id.
```press [key_comb]```: Simulates the pressing of a key combination on the keyboard (e.g., Ctrl+v).
```scroll [down]``` or ```scroll [up]```: Scroll the page up or down.

Tab Management Actions:
```new_tab```: Open a new, empty browser tab.
```tab_focus [tab_index]```: Switch the browser's focus to a specific tab using its index.
```close_tab```: Close the currently active tab.

URL Navigation Actions:
```goto [url]```: Navigate to a specific URL.
```go_back```: Navigate to the previously viewed page.
```go_forward```: Navigate to the next page (if a previous 'go_back' action was performed).

Completion Action:
```stop [answer]```: Issue this action when you believe the task is complete. If the objective is to find a text-based answer, provide the answer in the bracket.

Homepage:
If you want to visit other websites, check out the homepage at http://homepage.com. It has a list of websites you can visit.
http://homepage.com/password.html lists all the account name and password for the websites. You can use them to log in to the websites.

To be successful, it is very important to follow the following rules:
1. You should only issue an action that is valid given the current observation
2. You should only issue one action at a time.
3. You should follow the examples to reason step by step and then issue the next action.
4. Generate the action in the correct format. Start with a "In summary, the next action I will perform is" phrase, followed by action inside ``````. For example, "In summary, the next action I will perform is ```click [1234]```".
5. Issue stop action when you think you have achieved the objective. Don't generate anything after stop.

Figure 16: System message used in our SoM agent.

example_user:
IMAGES: (1) current page screenshot
OBSERVATION:
[31] [IMG] [Image, description: hp fx-7010dn fax machine, url: B08GKZ3ZKD.0.jpg]
[32] [A] [HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)]
[ ] [StaticText] [$279.49]
[33] [BUTTON] [Add to Cart]
[34] [A] [Add to Wish List]
[35] [A] [Add to Compare]
URL: http://onestopmarket.com/office-products/office-electronics.html
OBJECTIVE: What is the price of HP Inkjet Fax Machine?
PREVIOUS ACTION: None
example_assistant: Let's think step-by-step. This page list the information of HP Inkjet Fax Machine, which is the product identified in the objective. Its price is $279.49. I think I have achieved the objective. I will issue the stop action with the answer. In summary, the next action I will perform is ´´´stop [$279.49]´´´

example_user:
IMAGES: (1) current page screenshot
OBSERVATION:
[ ] [StaticText] [/f/food]
[ ] [StaticText] [[homemade] Obligatory Halloween Pumpkin Loaf! Submitted by kneechalice t3_yid9lu 1 year ago]
[9] [IMG] [ ]
[ ] [StaticText] [Submitted by kneechalice t3_yid9lu 1 year ago]
[10] [A] [kneechalice]
[11] [A] [45 comments]
[ ] [StaticText] [[I ate] Maple Pecan Croissant Submitted by AccordingtoJP t3_y3hrpn 1 year ago]
[14] [IMG] [ ]
[ ] [StaticText] [Submitted by AccordingtoJP t3_y3hrpn 1 year ago]
[15] [A] [AccordingtoJP]
[16] [A] [204 comments]
URL: http://reddit.com
OBJECTIVE: Tell me what the top comment on the croissant post says.
PREVIOUS ACTION: None
example_assistant: Let's think step-by-step. This page has a post titled '[I ate] Maple Pecan Croissant', which is the post mentioned in the objective. In order to find the top comment, I will navigate into the comments section of the post. In summary, the next action I will perform is ´´´click [11]´´´

example_user:
IMAGES: (1) current page screenshot
OBSERVATION:
[ ] [StaticText] [What are you looking for today?]
[5] [INPUT] [ ]
[6] [SELECT] [Select a category]
[7] [BUTTON] [Search]
[ ] [StaticText] [Latest Listings]
[ ] [StaticText] [Atlas Powered Audio System w/ Tripod 150.00 $ Music instruments Borough of Red Lion (Pennsylvania) 2023/11/16]
[8] [IMG] [Atlas Powered Audio System w/ Tripod]
[9] [A] [Atlas Powered Audio System w/ Tripod]
[ ] [StaticText] [150.00 $]
[ ] [StaticText] [Neptune Gaming Console 350.00 $ Video gaming Pennwyn (Pennsylvania) 2023/11/16]
[10] [IMG] [Neptune Gaming Console]
[11] [A] [Neptune Gaming Console]
[ ] [StaticText] [350.00 $]
URL: http://classifieds.com
OBJECTIVE: Help me find the cheapest dark colored guitar.
PREVIOUS ACTION: None
example_assistant: Let's think step-by-step. The objective is to find the cheapest dark colored guitar on the site. The site has a search box whose ID is [5]. I can search for guitars by entering "guitar". I can submit this by pressing the Enter afterwards. In summary, the next action I will perform is ´´´type [5] [guitar] [1]´´´

Figure 17: In-context examples and prompt used for our SoM agents. The webpage screenshots with SoM are added to the end of each round of the example_user dialogue.