PROPOSER-AGENT-EVALUATOR (PAE): AUTONOMOUS SKILL DISCOVERY FOR FOUNDATION MODEL INTERNET AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The vision of a broadly capable and goal-directed agent, such as an Internet-browsing agent in the digital world and a household humanoid in the physical world, has rapidly advanced, thanks to the generalization capability of foundation models. Such a generalist agent needs to have a large and diverse skill repertoire, such as finding directions between two travel locations and buying specific items from the Internet. If each skill needs to be specified manually through a fixed set of human-annotated instructions, the agent's skill repertoire will necessarily be limited due to the quantity and diversity of human-annotated instructions. In this work, we address this challenge by proposing Proposer-Agent-Evaluator(PAE), a complete working system that enables foundation model agents to autonomously discover and practice skills in the wild. At the heart of PAE is a context-aware task proposer that autonomously proposes tasks for the agent to practice with context information of the websites such as user demos or even just the name of the website itself. Then, the agent policy attempts those tasks with thoughts and actual web operations in the real world with resulting trajectories evaluated by an autonomous model-based success evaluator. The success evaluation serves as the reward signal for the agent to refine its policies through RL. We validate PAE on challenging vision-based web navigation, using both real-world and self-hosted websites from WebVoyager (He et al., 2024) and WebArena (Zhou et al., 2024a). Our results show that PAE significantly improves the zero-shot generalization capability of VLM Internet agents (more than 30% relative improvement) to both unseen tasks and websites. Our model also achieves an absolute advantage of over 10% (from 22.6% to 33.0%) comparing to other state-of-the-art open source VLM agents including Qwen2VL-72B. To the best of our knowledge, this work represents the first working system to apply autonomous task proposal with RL for agents that generalizes real-world human-annotated benchmarks with sota performances. We plan to release our models and code to facilitate further research.

1 INTRODUCTION

The vision of broadly capable and goal-directed agent, such as an Internet-browsing agent in the digital world and a household humanoid in the physical world, has long captured our imagination. With recent advancements in foundation models (OpenAI, 2024; GeminiTeam, 2024), this vision is no longer a distant dream. These developments have significantly accelerated the progress of generalist agents (Liu et al., 2023b) in real-world decision-making scenarios such as navigating through online websites to make travel plans (He et al., 2024) and solving real-user Github issues (Jimenez et al., 2024), making them a rapidly emerging research frontier. To succeed in these decision-making domains, goal-directed post-training is often needed to elicit long-horizon reward-maximizing behaviors such as information seeking (Hong et al., 2023a) and recovery from mistakes (Bai et al., 2024), instead of only imitating the most probable actions in the pre-training corpus.

A crucial requirement for a successful post-training approach is to endow the generalist agent with a large and diverse goal-directed skill repertoire. This can include finding directions between two travel locations and buying specific items from the Internet, which the agent can then exploit to solve real-world tasks proposed by users. However, manually specifying the skills (Deng et al., 2023) (i.e. through a static set of human-annotated instruction templates such as "Find the driving directions and estimated time to travel from Location A to Location B") will likely result in a limited skill repertoire. First of all, generating high-quality human-annotated task templates can be expensive, making it impractical to scale up. The use of a small set of task templates fails to capture the range of skills an agent needs for the full breadth of the real world, leading to distribution shift problems when deployed at the test time. Furthermore, human-generated instructions have limited diversity due to human creativity (Wang et al., 2023b), failing to capture the long-tail distribution of real-world tasks that the agent needs to solve. With these disadvantages, it naturally raises the research question: *instead of requiring users to manually define tasks for foundation model agents, can these agents* ***automatically discover*** *and practice potentially useful skills on their own?*

In order to discover its own skills and improve autonomously, such an agent would need to be able to propose semantically meaningful tasks and then determine if it was successful in performing them. Such success detections can then serve as reward signals to apply Reinforcement Learning (RL) to optimize the agents. While prior works have explored the use of foundation models to propose skills to the agents to practice and detecting successes in simplified environments such as games (Du et al., 2023; Colas et al., 2023) and robotics with limited number of scenes (Zhang et al., 2023b; Zhou et al., 2024b), little has been understood in terms of whether such diverse skills can generalize to real-world human request such as web agents and what the key design decisions are to improve such generalization.

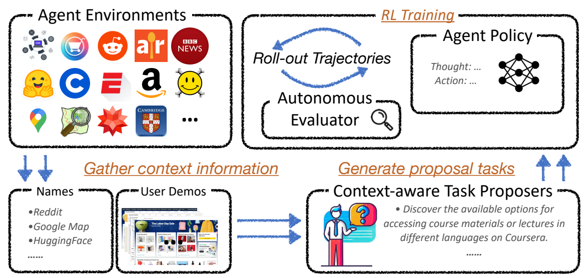


Figure 1: **An overview of our method** showing the main components of our autonomous skill discovery framework, that endows the agent with autonomously discovered skills to prepare for future human requests.

To this end, **our main contribution** is to propose a fully working system, Proposer-Agent-Evaluator(PAE), for foundation model agents (in particular, Internet agents) to autonomously discover new skills without any human supervision and such new skills can be effectively exploited to solve unseen real-world human-annotated tasks in a zero-shot manner. In this way, the training workflow can be easily scaled to make use of diverse self-generated instructions in large quantities to enrich the skill repertoire of the agent. PAE is built with the awareness the asymmetric capabilities of sota VLMs as task proposers/ evaluators and as agents (discussed in Section 6) in some realistic task settings such as web agents and designed to make best use of this asymmetry. Intuitively, VLMs are very good at confirming whether a specific product has been added to the shopping cart (e.g. by looking at the final screenshot to see if the shopping cart contains the product), while less good at actually navigating the web to find the product and add it to the cart. To obtain the most robust reward signal without accessing the hidden state information, we apply an image-based evaluator that only provides sparse 0/1 rewards based on the final outcome. To propose feasible and realistic tasks, PAE employs a class of context-aware task proposers where the context of functions and constraints crucially define what actions are supported by the specific environments (e.g., creating a reddit post) while others may not be supported (e.g., checking the protected information of other users). Such context can be implicitly defined from different sources and are shown to be effective, such as user demos and even website name alone! Finally, we design an additional reasoning step before the agent outputs actual actions, which enables the agents to better reflect on its skills and results in a significant improvement in its generalization capability to unseen human-annotated tasks.

The scope of our experiments covers challenging end-to-end vision-based web navigation, where the observation space simply contains the screenshot of the current web page and the action space contains primitive web operations such as clicking on links and typing into text boxes. We validate the effectiveness of PAE framework with realistic web-navigation benchmarks, including 16 domains both from online websites like Amazon from WebVoyager (He et al., 2024) and self-hosted websites like PostMill from WebArena (Zhou et al., 2024a). In our experiments, we find that PAE with LLaVa-1.6 (Liu et al., 2024) as the agent policy can autonomously discover useful skills through interactions with various websites without any human supervisions. More importantly, our results demonstrate that these skills can zero-shot transfer to unseen test instructions and even unseen test websites. On websites from WebVoyager and WebArena, PAE attains a 30% relative improvement in average success rate, enabling LLaVa-1.6-7B to achieve performance comparable with LLaVa-

1.6-34B fine-tuned with demonstration data despite using 5x fewer test-time compute. Compared to other state-of-the-art open-sourced VLM agents, including Qwen2VL-72B (Yang et al., 2024a), our model achieves an absolute performance gain of over 10% (22.6% to 33.0%). *To the best of our knowledge, this work is the first to develop a working system of autonomous skill discovery for foundation model agents that directly generalizes to real-world human-annotated benchmarks.*

2 Related Works

Foundation model agents. Thanks to the generalization capabilities of Large Language Models (LLMs) (Brown et al., 2020; Llama3Team, 2024; GeminiTeam, 2024) and Vision Language Models (VLMs) (OpenAI, 2024; Liu et al., 2024; Wang et al., 2024b; Liu et al., 2023a), recent works have successfully extended such agents to more general real-world use cases (Bai et al., 2024; Zheng et al., 2024; He et al., 2024; Zhang et al., 2023a; Zhou et al., 2024a; Koh et al., 2024; Gur et al., 2021; Furuta et al., 2024). Besides constructing prompting wrappers around proprietary VLMs (Zhang et al., 2023a; He et al., 2024; Zheng et al., 2024; Xie et al., 2024; Yang et al., 2024b; Wang et al., 2023a) and fine-tuning open-source VLMs with expert demonstrations (Gur et al., 2021; Hong et al., 2023b; Furuta et al., 2024; Zhang & Zhang, 2024; Zeng et al., 2023; Chen et al., 2023), a recent trend has emerged involving the interactive improvement of LLM/VLM, in particular web/GUI agents, through autonomous evaluator feedback (Pan et al., 2024; Bai et al., 2024; Putta et al., 2024), where evaluator LLMs/VLMs are prompted to evaluate the success of the agents to serve as the reward signal. This approach aims to elicit goal-oriented and reward-optimizing behaviors from foundation models with minimal human supervision. However, these methods still depend on a static set of human-curated task templates, which can constrain their potential and scalability. Our work introduces a novel framework where agents can *discover* and practice the skills they find useful, thereby eliminating the reliance on predefined and human-curated task templates. This approach opens up new possibilities for scalability and adaptability in training autonomous LLM/VLM agents.

Self-generated instructions. Self-generated instructions for improving LLMs have been shown to be effective in single-turn LLM alignment (Wang et al., 2023b; Yuan et al., 2024; Wu et al., 2024; Wang et al., 2024a) and reasoning (Pang et al., 2024) domains without interactions with an external environment. AgentGen (Hu et al., 2024) employs a similar methodology to fine-tune LLM agents with ground-truth trajectories in self-generated environments and tasks. However, its feasibility in the self-play agent setting with RL and autonomous evaluators has not been understood. The closest works to ours employ autonomous RL and foundation model task proposers to simplified environments such as games (Zhang et al., 2024a; Faldor et al., 2024; Colas et al., 2023; 2020) and robotics settings with limited number of scenes (Du et al., 2023; Zhang et al., 2023b; Zhou et al., 2024b). While they have shown that the use of autonomous RL combined with foundation model task proposers can help the agent learn diverse skills, this work takes an important step forward to study when those skills can generalize to human requests in realistic benchmarks in the context of web agents and what the best design choices are for such generalization.

Unsupervised skill discovery in deep RL. Unsupervised skill discovery has been an important research direction in the field of traditional deep RL literatrue (Achiam et al., 2018; Eysenbach et al., 2018) where various algorithms have been developed to discover new robotic skills such as humanoid walking without the need of explicitly defined reward functions. Common algorithms in this field aim to discover *every possible* skill (either meaningful skills like walking or less meaningful ones like random twisting) through either maximizing the mutual information between different states and skill latent vectors (Campos et al., 2020; Laskin et al., 2022; Sharma et al., 2020), or maximizing the divergence of each skill as measured in a metric space (Park et al., 2022; 2023; 2024). In contrast, our work *only discovers meaningful skills* as specified through natural language instructions with the help of pre-trained foundation models, significantly reducing the search space of skills in LLM/VLM agent applications with complex state spaces.

3 Proposer-Agent-Evaluator (PAE): Autonomous Skill Discovery System For Foundation Model Agents

Next, we will explain the technical contributions of this paper. In this section, we will define the general system of PAE including a task proposer, an agent policy, and an autonomous evaluator. We will begin by formalizing the learning goal of this system and detailing the roles of each key component in the system. Then we will walk through our practical algorithm in the system. In the section to follow, we will provide the example of applying PAE to VLM Internet agents.

Problem setup We begin by formalizing the problem setup of autonomous skill discovery for real-world agents. The learning goal of PAE is to find a reward-maximizing policy π parameterized by θ in a contextual Markov Decision Process (MDP) environment defined by $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, H, \mathcal{C}\}$, where $\mathcal{S}, \mathcal{A}$ are the state space and action space respectively, and H is the horizon within which the agent must complete the task. We assume that the agent has access to the environment and can collect online roll-out trajectories through accessing the dynamics model $\mathcal{T}$ as a function of determining the next states given the current states and actions. *Crucially, we assume that the ground-truth task distribution $\mathcal{C}$ and the reward function $\mathcal{R}$ are hidden during training and we have to use a proxy task distribution $\hat{\mathcal{C}}$ and reward function $\hat{\mathcal{R}}$ instead.* Consider the setting of training a real-world Internet agent. The dynamics model $\mathcal{T}$ would be a simulated browser environment that the Internet agent can interact with. The ground-truth task distribution $\mathcal{C}$ might be the distribution of tasks that would be asked by the real users when the Internet agent is deployed and a possible choice for the reward function $\mathcal{R}$ might be whether the agent has satisfactorily completed the tasks for the real users. In such a real-world setting, although the agent can freely access resources from the Internet through a simulated browser environment during training, assuming knowledge of the ground-truth task distribution and reward function is impractical. Therefore, we employ VLM-based task proposers $\hat{\mathcal{C}}$ and reward model $\hat{\mathcal{R}}$ as proxies. The desired outcome is that improving the policy π_θ with $\hat{\mathcal{C}}$ and $\hat{\mathcal{R}}$ can lead to an improved policy that can successfully generalize to the ground-truth task distribution and reward functions only used as evaluations.

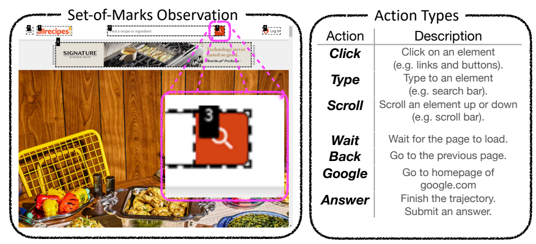


Figure 2: **An illustration of the observation space and action space of our vision-based web navigation environment.** The observation space is augmented with set-of-marks that label each interactable element with a unique number. At each step, the web agent first chooses an element to interact with by referring to its number and then choose the action type to perform on this element (e.g., click, type, and etc.).

Key components Figure 1 shows the interplay between the key components in our framework, including a context-aware task proposer, an agent policy, and an autonomous evaluator. The role of the **task proposer** $\hat{\mathcal{C}}$ is to serve as a proxy to improve on the ground-truth task distribution $\mathcal{C}$ during the learning process. However, it might be unrealistic to expect the task proposer to generate feasible tasks without knowledge of the environment. To provide more context of the functions and constraints of the environment, we assume access to some key information of the environment $z_{\mathcal{M}}$ based on which the tasks $\hat{\mathcal{C}}(z_{\mathcal{M}})$ are proposed. In the Internet agent example, this key information can be screenshots of the websites from user demos, or even just the name of the website itself if it is a well-known website such as Amazon.com. Similarly, the **autonomous evaluator** $\hat{\mathcal{R}}$ serves as a proxy of the ground-truth reward function $\mathcal{R}$. The input to the autonomous evaluator is the current state, the current action from the agent policy, and current task that the agent is attempting. In principle, any RL algorithm can be used to update the **agent policy** π using a dataset $\mathcal{D}$ that stores all the autonomous interaction data. In practice, we instantiate VLM-based task proposers and autonomous evaluators by prompting foundation models and they are kept unchanged throughout our practical algorithm.

4 Proposer-Agent-Evaluator for VLM Internet Agents

With the general framework set up, we are now ready to discuss the concrete instantiation of PAE in the setting of VLM Internet agents. We start by introducing the environment of vision-based web navigation and then explain how we implement the key components from the PAE in this setting.

4.1 Vision-Based Web Browsing Environment

We consider the general vision-based web browsing environment (He et al., 2024; Koh et al., 2024). The goal for VLM agents in this environment is to navigate through realistic web pages to complete some user tasks c_t such as "Investigate in the Hugging Face documentation how to utilize the 'Trainer' API for training a model on a custom dataset, and note the configurable parameters of the Trainer class". As illustrated in Figure 2, each **observation** s_t from the observation space contains only the screenshot of the last web page just like how humans interact with the Internet. To provide better action grounding, we follow the practice from prior works (Zheng et al., 2024; He et al., 2024) to augment the observation space with number marks on top of each interactive element such as web links and text boxes. To execute a web browsing action, the Internet agent can directly output the number of the element to interact with and the corresponding action such as clicking and typing, without the need of locating the coordinates of each web element. Therefore each web **action** a_t contains the type of the action to perform and the number of the element to interact with. Each episode finishes either when the agent chooses to finish through the "Answer" action or when a maximum number of 10 steps have been reached. In our experiments, we use ground-truth success detectors (based on either human annotations or functional verifiers) and human annotated tasks from WebArena (Zhou et al., 2024a) and WebVoyager (He et al., 2024) to evaluate the performance of different policies. Crucially, both the ground-truth success detector and the distribution of human tasks are kept hidden, which challenges the generalization capability of the learnt skills to generalize to a hidden reward function and task distributions.

4.2 Context-Aware Task Proposer

In order to generate a diverse set of feasible tasks, we frame task proposing $\hat{\mathcal{C}}$ as a conditional auto-regressive generation based on the context information of the websites. Thanks to the vast pre-training knowledge of relevant context for popular websites like Amazon.com, we find it suffice to use only **website name** as $z_{\mathcal{M}}$. However, for less common or access restricted websites such as self-hosted websites in WebArena, it is necessary to supply the task proposer with richer context. In the cases of **user demos** being available, we consider an alternative to sample some additional screenshots from the user demos to serve as the context information. In our experiments, we consider both using proprietary models such as Claude-3-Sonnet (Anthropic, 2024) and open-source models such as Qwen2VL-7B Yang et al. (2024a) for the task proposers, with promptsd in Appendix B.

4.3 Image-Based Outcome Evaluator

To take full advantage of the asymmetric capability of SOTA VLMs as agents and as evaluators (experiment results presented in Section 6, we empirically find it reliable for the autonomous evaluators to complete the easiest evaluation: evaluating the success of the final outcome (Bai et al., 2024; He et al., 2024) based on the final three screenshots and the agents' final answers to provide only 0/1 response in the end. Other alternatives such as code-based (Zhang et al., 2024a) or step-based evaluations (Pan et al., 2024) are either impractical without access to hidden state information or too noisy because of the hallucination issues present even in SOTA VLMs. In our experiments, we also consider both using proprietary models such as Claude-3-Sonnet (Anthropic, 2024) and open-source models such as Qwen2VL-7B Yang et al. (2024a), with prompts presented in Appendix B.

4.4 Chain-of-Thought Agent Policy

Crucially, as the ultimate goal for the agent policy is to complete human requests, the agent should not only learn diverse skills on the proposed tasks but also reflect on the skills learnt so that they can be helpful for unseen human requests. Therefore, we incorporate an additional reasoning step to outputs the agent's chain-of-thought before the actual web operation. This reasoning step is optimized with the RL algorithm just like the actual web operation. Because of the 0/1 reward structure and infrastructure complexity of thousands of distributed fully-functioning web browsers, we employ the most simple online policy optimization algorithm Filtered Behavior Cloning (Filtered BC), that simply imiatates all thoughts and actions in successful trajectories with the negative log-liklihood loss. We find that this simple policy optimization objective can already lead to an superior generalization capability of the learnt agent. In our experiments, our agent policy is initialized from LLaVa-1.6-Mistral-7B and LLaVa-1.6-Yi-34B (Liu et al., 2024).

5 Experiments

The goal of our experiments is to understand the effectiveness of PAE to complete real-world visual web tasks. Specifically, we design experiments to answer the following questions: **(1)** Can our autonomous skill discovery framework successfully discover skills useful for zero-shot transfer to tasks from an evaluation task distribution unseen to the task proposer? **(2)** How does the models trained with PAE compare with other open-source VLM agents? **(3)** How does the effectiveness of PAE scale with the size and performance of the base model? **(4)** How does the use of different contexts (e.g. website names and user demos) affect the performance?

5.1 Environments

WebVoyager (He et al., 2024) contains a set of 643 tasks spanning 15 websites in the real world such as ESPN and Arxiv. As tasks in Google Flights and Booking domain are no longer feasible due to website updates, we use the subset of 557 tasks spanning the other 13 websites. Human annotations are carried out for evaluating the success of each trajectory as the ground-truth performance measure.

		Allrecipes	Amazon	Apple	ArXiv	GitHub	ESPN	Coursera
Proprietary	Claude 3.5 Sonnet	50.0	68.3	60.4	46.5	58.5	27.3	78.6
	Claude 3 Sonnet	15.9	46.3	51.1	39.5	41.4	11.3	45.2
Open-source	Qwen2VL-7B	0.0	0.0	2.3	2.3	0.0	0.0	2.3
	Qwen2VL-72B	0.0	29.0	28.1	18.2	9.7	5.9	**48.5**
	InternVL2.5-8B	0	0	0	0	0	0	0
	LLaVa-7B	0	0	0	0	0	0	0
	LLaVa-34B	0	0	2.3	0	2.4	0	0
Ours	LLaVa-7B SFT	4.5	39.0	18.6	16.3	4.9	5.1	16.7
	LLaVa-7B PAE	14.3	37.5	17.5	19.0	**14.6**	0.0	33.3
	LLaVa-34B SFT	6.8	26.8	23.3	16.3	4.9	8.6	26.8
	LLaVa-34B PAE	**22.7**	**53.7**	**38.5**	**25.6**	**14.6**	**13.6**	42.9
		Cambridge Dictionary	BBC News	Google Map	Google Search	HuggingFace	Wolfram Alpha	Average
Proprietary	Claude 3.5 Sonnet	86.0	36.6	58.5	30.2	44.2	66.7	50.5
	Claude 3 Sonnet	79.1	40.5	41.5	41.9	37.2	61.9	42.4
Open-source	Qwen2VL-7B	2.3	0.0	0.0	4.7	0.0	4.8	1.4
	Qwen2VL-72B	60.6	12.5	16.1	21.2	9.1	36.4	22.6
	InternVL2.5-8B	0	0	0	0	2.3	0	0.2
	LLaVa-7B	0	0	0	0	0	0	0
	LLaVa-34B	0	2.3	0	2.3	2.3	0	0.9
Ours	LLaVa-7B SFT	41.9	7.1	19.5	9.3	0	11.9	14.9
	LLaVa-7B PAE	52.4	18.6	22.5	**23.3**	19.0	24.4	22.3
	LLaVa-34B SFT	67.4	16.7	12.2	**23.3**	20.9	38.1	22.2
	LLaVa-34B PAE	**74.4**	**39.0**	**22.0**	18.6	**25.6**	**42.9**	**33.0**

Table 1: **Success rate comparisons on WebVoyager.** The results are automatically annotated by Claude Sonnet 3 and human alignment is reported in Figure 4. For PAE , a running average of the evaluation results at each iteration is reported. The final column is a weighted average by the number of tasks on different websites. The results may be different from reported in other papers due to the dynamic nature of online websites.

WebArena (Zhou et al., 2024a) is a sand-boxed environment that kept an archived version of 5 popular websites from different domains, including OpenStreetMap, GitLab, PostMill, a store content management system (CMS), and an E-commerce website (OneStopMarket). It includes in total 812 hand-written tasks with functional verifications as the ground-truth reward function. Since GitLab and CMU do not support multi-thread data collection necessary for RL fine-tuning, our experiments are carried out using the task subsets on OpenStreetMap, PostMill, and OneStopMarket. As open-source VLM agents fail to achieve non-trivial performances on PostMill and OneStopMarket (Zhou et al., 2024a), we hand rewrote tasks in those two websites and supplement them with verification functions. Due to these practical constraints, the resulting **WebArena Easy** contains 108 original tasks on OpenStreetMap and 50 rewritten tasks on PostMill and OneStopMarket each.

5.2 Baseline Comparisons

We validate the effectiveness of PAE by comparing it with **(1)** proprietary VLMs, **(2)** state-of-the-art open-source VLMs, and **(3)** an alterative supervised fine-tuning (SFT) approach. We consider **Claude 3 Sonnet** and **Claude 3.5 Sonnet** (Anthropic, 2024) for proprietary VLMs, and **Qwen2VL-7B**, **Qwen2VL-72B** (Yang et al., 2024a), **InternVL-2.5-XComposer-7B** (Zhang et al., 2024b), and **LLaVa-Next-7B/34B** (Liu et al., 2024) for SOTA open-source VLMs. All models are prompted similar to He et al. (2024) using set-of-marks augmented screenshot observations and including chain-of-thought in the action outputs. The prompts are included in Appendix B. As SOTA open-source models struggle to achieve non-trivial performance in the challenging web navigation benchmarks except the largest Qwen2VL-72B, we include another baseline **LLaVa-SFT** that fine-tunes LLaVa with Claude 3 Sonnet (Anthropic, 2024) agent trajectories on self-generated tasks on 85 real-world websites not included in WebVoyager and WebArena. More details in the data generation for SFT can be found in Appendix D. To study the effects of different contexts for our task proposer, we compare the performance of two variants from PAE as discussed in Section 4.2. **LLaVa-34B PAE** and **LLaVa-7B PAE** uses only the name of the website as the context, while **LLaVa-7B-PAE (User Demos)** uses 10 additional screenshots per website from human collected user demos.

		OpenStreetMap	PostMill	OneStopMarket	Average
Proprietary	Claude 3.5 Sonnet	38.3	70.0	53.0	50.1
	Claude 3 Sonnet	24.3	55.8	41.7	36.0
Open-source	Qwen2VL-7B	0.7	10.2	20.2	7.5
	Qwen2VL-72B	16.0	32.8	32.7	23.9
	InternVL2.5-8B	1.8	0.5	6.0	2.5
	LLaVa-7B	0.0	0.0	0.0	0.0
	LLaVa-34B	0.9	0.0	0.0	0.5
Ours	LLaVa-7B SFT	15.2	16.8	25.4	18.0
	LLaVa-7B PAE	19.5	21.1	**42.3**	24.6
	LLaVa-7B PAE (User Demos)	**21.7**	21.5	42.1	**25.7**

Table 2: **Success rate comparisons on WebArena Easy.** Success and failure are detected with ground-truth verification functions. For PAE , a running average of the evaluation results at each iteration is reported. The final "Average" column is a weighted average by the number of tasks on different websites.

5.3 Main results

We present our main baseline comparisons of PAE with other baselines in Table 1, 2, and 3. Overall, comparing to the SFT checkpoint using demonstration data, LLaVa-7B PAE can achieve an average of 7.4% and 10.8% absolute improvement in terms of success rates on WebVoyager and WebArena Easy respectively. A similar improvement of 10.4% on WebVoyager is observed for LLaVa-34B PAE as well, indicating a favorable scaling performance of PAE. As a result, our resulting model LLaVa-34B PAE achieves an absolute success rate of 10.4% on WebVoyaer over the prior state-of-the-art open-source VLM agents. Similarly, LLaVa-7B PAE also establishes a new state-of-the-art performance on WebArena Easy, surpassing the prior best performing model Qwen2VL-72B with 10× more parameters. More importantly, our analysis shows that PAE can enable Internet agents to learn general web browsing capabilities that zero-shot transfer to unseen websites.

How does existing open-source and proprietary models perform in vision-based web navigation? First, we note the difficulty and significance of real-world vision-based web navigation, even for state-of-the-art medium-size open-source VLM agents such as Qwen2VL-7B and InternVL2.5-8B with set-of-marks augmented observations and chain-of-thought prompting. In particular, on the WebVoyager benchmark, among open-source VLM agents, only the largest Qwen2VL-72B can achieve a non-trivial average success rate of 22.6% on WebVoyager, while all other open-source agents completely fail on this benchmark with average success rate under 2%. On the other hand, closed-source proprietary models start to show promise in becoming a generalist Internet agent with Claude 3.5 Sonnet achieving an average success rate at 50.5% and 50.1% on WebVoyager and WebArena Easy. Comparing LLaVa-7B SFT and LLaVa-7B, we find that supervised fine-tuning on demonstration data can significantly improve the general web browsing capabilities of open-source VLM agents. Even if the SFT demonstration data is collected on out-of-distribution online websites, the general web browsing capabilities can zero-shot transfer to WebVoyager websites, resulting in a performance improvement from 0% to 14.9%.

Model	Seen Websites	Unseen Websites
Claude 3 Sonnet	42.4	25.0
Qwen2VL-7B	1.4	1.4
Qwen2VL-72B	22.6	8.3
LLaVa-7B SFT	14.9	9.1
LLaVa-7B PAE	22.3	16.3
LLaVa-34B SFT	22.2	16.1
LLaVa-34B PAE	**33.0**	**21.4**

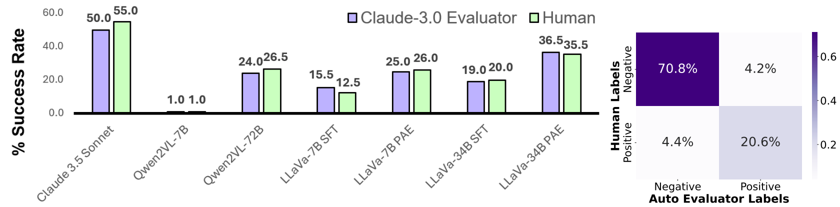
Table 3: **Task success rate comparisons on unseen websites** that PAE never interacts with. We select 85 unseen real-world online websites and generate 500 synthetic tasks similar to the procedure in WebVoyager (He et al., 2024). Seen websites are 13 online websites in WebVoyager. Results show that PAE can discover general web browsing skills useful for unseen websites.

Is PAE able to autonomously discover and practice skills useful for unseen evaluation instructions? On top of the performance gain from downstream fine-tuning, LLaVa-7B PAE additionally improves the success rate by more than 30% relatively (14.9% to 22.3% on WebVoyager and 18.0% to 24.6% on WebArena Easy). In particular, LLaVa-7B PAE beats LLaVa-7B SFT across the board with substantial improvements on 10 out of 13 websites from WebVoyager and all 3 websites from WebArena Easy, showing the robustness of the PAE framework. In fact, LLaVa-7B PAE even beats the LLaVa-34B SFT (22.3% compared to 22.2%), a model more than 5x larger (7B and 34B), resulting a better performing model with 5x less test-time compute. The release of our models marks a significant advancement of screenshot-based web browsing capabilities of open-source VLM agents from the prior SOTA of 22.6% to 33.0% on WebVoyager. It also enables medium-size VLMs such as LLaVa-7B to beat the prior SOTA Qwen2VL-72B with 10× more parameters on WebArena Easy. Notably, all of the improvements from PAE are achieved in a self-play setting without any human intervention, only knowing the names of the websites!

Does PAE scale well with larger and more capable base models? To test the scaling performance of PAE , we repeat our experiments on WebVoyager with a larger and more capable VLM base model LLaVa-1.6-34B (Liu et al., 2024). With a better base model, we still find a similar performance gain of PAE despite the model size change from 7B to 34B (7.4% compared to 10.8% absolute success rate improvement). Again, LLaVa-34B PAE beats LLaVa-7B PAE on 12 out of 13 websites from WebVoyager. Our scaling experiments suggest that PAE a favorable scaling property that can similarly improve better and larger base VLM agents as they become available.

Do the skills learnt by PAE generalize to unseen environments? To understand the generalization of LLaVa-7B PAE to the websites that it has never interacted with, we apply the workflow from He et al. (2024) to generate 500 tasks using Claude 3 Sonnet on 85 unseen online websites and test the checkpoints from WebVoyager experiments. Results are presented in Table 3 and a list of the websites is included in Appendix D. We observe that PAE for both LLaVa-7B and LLava-34B enable the agents to learn general web-browsing skills that can be zero-shot transferred to unseen websites, with 7.2% and 5.3% improvement in absolute success rate respectively.

6 DISCUSSIONS



(a) Correlation between Human and Autonomous Evaluator. (b) Confusion Matrix

Figure 4: **Correlation and confusion matrix** analysis of different models in Webvoyager. (a) Correlation between human evaluations and our autonomous evaluator across various models at the system level. (b) Confusion matrix of the overall correlation between human evaluations and our autonomous evaluator at the instance level. Both results show strong correlation between our autonomous evaluator and human evaluations.

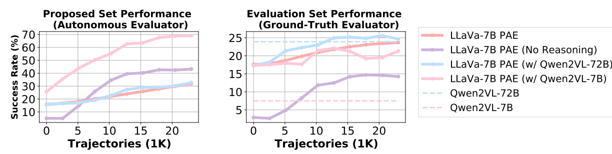


Figure 3: **Ablation experiments on WebArena Easy.** The left figure measures the performance on the set of proposed tasks by different models with autonomous evaluators while the right figure measures the performance on WebArena Easy with the ground-truth evaluator.

Open-source VLMs as task proposers and autonomous evaluators In Figure 3, we present the results of using open-source VLMs (Qwen2VL-7B and Qwen2VL-72B) as task proposers and autonomous evaluators, thus eliminating the dependence of PAE on proprietary models. We found that LLava-7B PAE using Qwen2VL-72B as the task proposer and evaluator achieved a similar performance as using Claude 3 Sonnet as the task proposer and evaluator, despite their significant difference in agent performances (23.9% compared to 26.0% average success rate). As a result of this improvement, LLava-7B PAE using Qwen2VL-72B as the task proposer and evaluator achieved a better performance compared to Qwen2VL-72B itself. Perhaps more surprisingly, even Qwen2VL-7B with much inferior agent performance compared to LLaVa-7B SFT (7.5% compared to 18.0%) can be used to make significant improvements (from 18.0% to 23.1%). These results demonstrate that the improvements from PAE root in the asymmetric capabilities of state-of-the-art VLMs as agents and as task proposers/evaluators, instead of imitating a stronger VLM.

The effect of additional reasoning step. We also perform an additional ablation on the effect of the PAE design choice of asking the VLMs to output their thoughts first prior to the actual web operations. We consider an additional baseline of directly outputting the web operations without thoughts, and carry out the similar SFT and Filtered BC experiments using the same setup described in Section 5.2. As reported in Figure 3, although PAE without reasoning can also achieve improvements in the proposed set, the lack of additional reasoning step results in a significant inferior performance in its generalization to the unseen human-written evaluation set.

Alignment with human judgements. We demonstrate the effectiveness of our autonomous evaluator with a user study. We randomly select 200 trajectories for each method and present all screenshots in the trajectories, the corresponding actions at each step, and the task descriptions to the human annotator to decide if the task has actually been completed or not. As shown in Figure 4(a), there is a high correlation between our evaluator and human assessments across different models with an average misalignment of 1.7% at the system level and 8.9% at the instance level. The effectiveness of PAE as judged by human annotators is consistent with what is reported in Table 1.

Error analysis. To understand where the improvement of PAE comes from, we conducted a user study to analyze different error types across various models. With a high-level evaluation of model capacities, we classified the error types into the following categories: **(1) Low-level skills missing error** refer to the cases where the agent has a reasonable plan to solve the problem but fails to execute precise actions on the website, such as not knowing which button to click to navigate to the desired page. **(2) High-level planning or reasoning errors** refer to the cases where the agent fails to generate a plan in its thoughts to solve the given task or cannot arrive at the correct answer through reasoning with the website's screenshots. **(3) Visual hallucinations** refer to the cases where the agent generates responses with made-up information that are not supported by the screenshot. For example, the agent may claim that it has found a product that the task asked for while remaining at the homepage of google search, or the agent may produce a wrong answer while being on the right

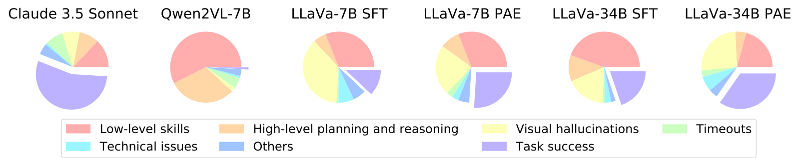


Figure 5: **Failure mode** analysis for different models in WebVoyager environment.

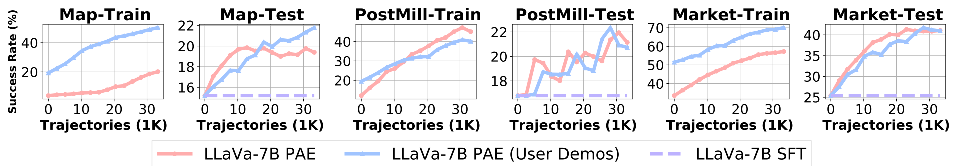


Figure 6: **Online sample complexity comparisons on different websites in WebArena Easy between PAE using different contexts.** Note that PAE with different contexts for task proposers uses different training tasks. Learning curves are smoothed with exponential running averages.

page. **(4) Timeouts** refer to the cases where the agent is on the right track to solving the tasks, but couldn't complete the task within maximum number of steps. **(5) Technical Issues** are not the fault of the agent but caused by environment problems such as websites out of service and connection issues. **(6) Others** include other less often error types such as the task itself is impossible.

We present the results of error analysis for different models in WebVoyager in Figure 5, with a more detailed analysis and full trajectories in Appendix H. Comparing LLaVa-7B SFT with LLaVa-34B SFT, we observe that the predominant failure mode for LLaVa-7B SFT is visual hallucinations while that for LLaVa-34B is low-level skill missing errors. This is because the reasoning capability for LLaVa-7B base model is limited so it tends to imitate the demonstration data to produce answers that look similar without being aware of the correctness of the answers. While LLaVa-34B SFT is more aware of the correctness of answers (evidenced by a reduced visual hallucination rate), it does not have the low-level web navigation skills so often fall short of low-level operations. PAE can effectively improve on the major failure mode for both 7B and 34B models. In particular, for LLaVa-7B SFT, PAE can reduce the visual hallucination rate (from 37% to 23%), making the agent more aware of the goal of actually completing the tasks instead of imitating the demonstrations. For LLaVa-34B SFT, PAE can effectively enrich the skill repertoire with low-level web navigation skills, thereby reducing the low-level skill missing error (from 45% to 21%). Comparing our models with other VLM agents, we find that other open-source VLM agents such as Qwen2VL-7B and Qwen2VL-72B mostly struggle with low-level web navigation skills while the error types for more advanced proprietary models such as Claude 3.5 Sonnet are more spread out.

Comparison of different choices of contexts. We present our study on the effects of using different contexts on WebArena in Table 2 and Figure 6. By comparing the success rate between LLaVa-7B PAE and LLaVa-7B PAE (User Demos), we find additional information significantly improves the performance in the original WebArena task set Map (19.5% to 21.7%) but does not make a big difference in the rewritten easier task sets on PostMill and OneStopMarket. By manually inspecting the tasks proposed with and without user demos, we find that many tasks proposed with website names alone are too hard or even impossible given the supported features of OpenStreetMap. For example, a task like "Locate the closest movie theater to the address 456 Oak Street, Chicago, Illinois, and provide the theater's name, address, and current movie showtimes." is impossible to be completed on OpenStreetMap as it does not contain information related to the current movie showtimes. As shown in the learning curve in Figure 6, indeed the agent achieves a significantly lower performance on the training tasks of PAE compared to that of PAE (User Demos). On the contrary, this gap in terms of training set performances is much reduced on PostMill and OneStopMarket. We hypothesize that this is because our simplified tasks on PostMill and OneStopMarket only examine the basic usages of the websites such as "Go to a forum related to relationship advice" and "Browse the Patio and Garden shopping category" and such tasks can be easily proposed with rudimentary understanding of the websites inferred from the names of the websites alone. As the tasks get harder and involve more complicated interactions with elements on different websites, we expect the use of context information to play a more important role.

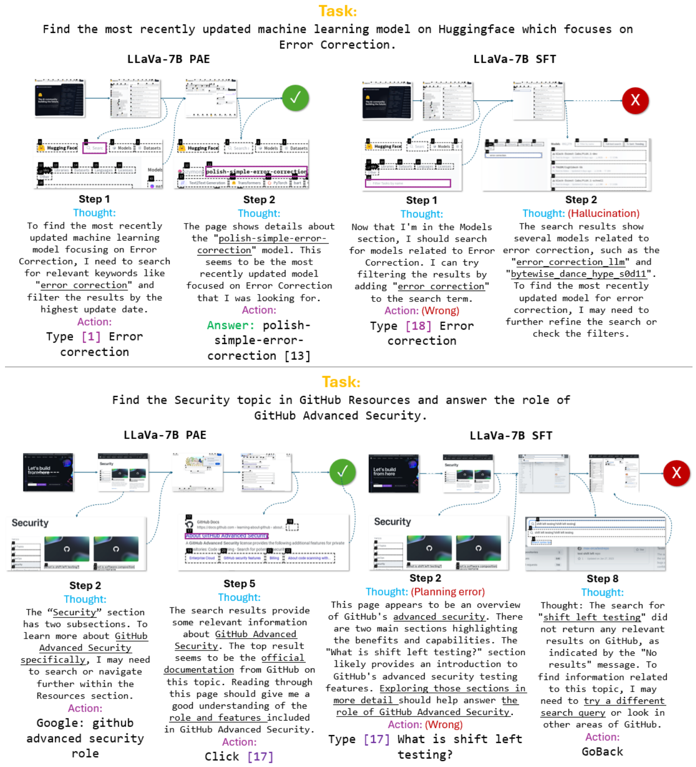


Figure 7: **Qualitative comparison between LLaVa-7B PAE and LLaVa-7B SFT** on the same tasks. LLaVa-7B PAE model successfully completed two tasks using learned skills from the RL training.

Qualitative comparisons. To qualitatively understand the benefits of PAE , we present snippets of example trajectories in Figure 7 from evaluations on WebVoyager where LLaVa-7B PAE and LLaVa-7B SFT attempt the same tasks. Full trajectories are included in Appendix H. In the first example, we find that while LLaVa-7B knows SFT that it should use the search bar to find models related to error correction, it fails to choose the correct search bar (should be [18] instead of [1]). However, LLaVa-7B PAE learns the skill of using the search bar through typing into the correct index [1] and executes its plan to complete the task. In the second example, the agent needs to navigate to the Advanced Security page of Github. While both models are able to navigate to the Security page of Github first, there turns out to be no direct links from the Security page to the Advanced Security page. As a result, LLaVa-7B SFT ends up wandering in Github without finding the Advanced Security page. In contrast, LLaVa-7B PAE learns the skill of using Google Search in the absence of a direct link and it successfully navigates to the right page with its help. In both cases, we observe qualitative evidence of PAE teaching the agent a diverse repertoire that can effectively help the agent to complete unseen tasks.

7 Conclusions and Future Work

In this paper, we introduced a working system, PAE, for autonomous skill discovery with foundation model agents, addressing the limitations of using a static set of human-annotated instructions for fine-tuning agents. Instead of manually specifying what the agents should learn, our system enables the agents to explore, practice, and refine new skills autonomously through open-ended interactions with various environments. The framework's key components—task proposer, action policy, and autonomous evaluator—work together to generate, attempt, and evaluate tasks without any human intervention, leading to more than 10% improvement over prior state-of-the-art performance across benchmarks like WebArena Easy and WebVoyager among open-source VLM agents (22.6% to 33%). This work paves the way for more capable open-source foundation model agents, with future research focused on extending this approach to other domains and integrating it with better approaches to make use of the context information.

Reproducibility Statement

To facilitate reproducibility of our work, we plan to open-source the model checkpoint and code. To provide more details about our practical algorithm, we have included the algorithm pseudo-code in Algorithm 1. We have also included all the prompts that we have used for the task proposer, the agent policy, and the autonomous evaluator in Appendix B. More details for gathering and processing the SFT dataset have been included in Appendix D. An discussion of the hyperparameter tuning of our method has been included in Appendix G.

Ethics Statement

This work aims to enhance autonomous Internet agents through open-ended interactions with the web. However, the irresponsible or unrestricted use of such agents may pose risks, including personal data leaks or vulnerabilities to malicious attacks. To mitigate these risks, it is crucial to implement robust precautionary measures. In our experiments involving open-ended web navigation, we ensure that the agent is restricted from accessing personal accounts and employ appropriate firewalls to block DNS requests to suspicious websites. These safeguards help prevent unintended consequences and protect sensitive information.

References

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms, 2018. URL https://arxiv.org/abs/1807.10299.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning, 2024. URL https://arxiv.org/abs/2406.11896.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills, 2020. URL https://arxiv.org/abs/2002.03647.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity-driven exploration, 2020. URL https://arxiv.org/abs/2002.09253.

Cédric Colas, Laetitia Teodorescu, Pierre-Yves Oudeyer, Xingdi Yuan, and Marc-Alexandre Côté. Augmenting autotelic agents with large language models, 2023. URL https://arxiv.org/abs/2305.12487.

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, 2023. URL https://arxiv.org/abs/2306.06070.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models, 2023. URL https://arxiv.org/abs/2302.06692.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018. URL https://arxiv.org/abs/1802.06070.

Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code, 2024. URL https://arxiv.org/abs/2405.15568.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models, 2024. URL https://arxiv.org/abs/2305.11854.

GeminiTeam. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis, 2021.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024. URL https://arxiv.org/abs/2401.13919.

Joey Hong, Sergey Levine, and Anca Dragan. Zero-shot goal-directed dialogue via rl on imagined conversations, 2023a. URL https://arxiv.org/abs/2311.05584.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2023b. URL https://arxiv.org/abs/2312.08914.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, Saravan Rajmohan, and Dongmei Zhang. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation, 2024. URL https://arxiv.org/abs/2408.00764.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024. URL https://arxiv.org/abs/2401.13649.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 34478–34491. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/debf482a7dbdc401f9052dbe15702837-Paper-Conference.pdf.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023a. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

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, 2023b. URL https://arxiv.org/abs/2308.03688.

Llama3Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents, 2024. URL https://arxiv.org/abs/2404.06474.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization, 2024. URL https://arxiv.org/abs/2404.19733.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery, 2022. URL https://arxiv.org/abs/2202.00914.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery, 2023. URL https://arxiv.org/abs/2302.05103.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2024. URL https://arxiv.org/abs/2310.08887.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents, 2024. URL https://arxiv.org/abs/2408.07199.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills, 2020. URL https://arxiv.org/abs/1907.01657.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL https://arxiv.org/abs/2305.16291.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. Self-taught evaluators, 2024a. URL https://arxiv.org/abs/2408.02666.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. Cogvlm: Visual expert for pretrained language models, 2024b. URL https://arxiv.org/abs/2311.03079.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023b. URL https://arxiv.org/abs/2212.10560.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge, 2024. URL https://arxiv.org/abs/2407.19594.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024a. URL `https://arxiv.org/abs/2407.10671`.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024b. URL `https://arxiv.org/abs/2405.15793`.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2024. URL `https://arxiv.org/abs/2401.10020`.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2023. URL `https://arxiv.org/abs/2310.12823`.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users, 2023a. URL `https://arxiv.org/abs/2312.13771`.

Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. Omni: Open-endedness via models of human notions of interestingness, 2024a. URL `https://arxiv.org/abs/2306.01711`.

Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance, 2023b. URL `https://arxiv.org/abs/2310.10021`.

Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, Songyang Zhang, Wenwei Zhang, Yining Li, Yang Gao, Peng Sun, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Hang Yan, Conghui He, Xingcheng Zhang, Kai Chen, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output, 2024b. URL `https://arxiv.org/abs/2407.03320`.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents, 2024. URL `https://arxiv.org/abs/2309.11436`.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024. URL `https://arxiv.org/abs/2401.01614`.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024a. URL `https://arxiv.org/abs/2307.13854`.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models, 2024b. URL `https://arxiv.org/abs/2407.20635`.

Appendices

A Algorithm

In Algorithm 1, we include a formal definitions of our practical algorithm of PAE as presented in Section 3.

Algorithm 1 Proposer-Agent-Evaluator: Practical Algorithm

Require: Context information $z_{\mathcal{M}}$, task proposer $\hat{\mathcal{C}}$, autonomous evaluator $\hat{\mathcal{R}}$.

Initialize policy π from a pre-trained checkpoint.
Initialize replay buffer $\mathcal{D} \leftarrow \{\}$.
Propose tasks based on the context information.
Obtain proposal task distribution $\hat{\mathcal{C}}(z_{\mathcal{M}})$.
for each global iteration **do**
    for each trajectory to be collected **do**
        Sample a task from the task proposer $c \sim \hat{\mathcal{C}}(z_{\mathcal{M}})$.
        Reset the environment to obtain the initial observation s_0
        for each environment step t **do**
            Sample $a_t \sim \pi(\cdot|s_t, c)$, $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t, c)$.
            if done **then**
                ## Autonomously evaluate the outcome of the agent rollout.
                $r_t \leftarrow \hat{\mathcal{R}}(s_t, a_t, c)$.
            else
                $r_t \leftarrow 0$.
            end if
            $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1}, c)\}$.
        end for
    end for
    ## Update the agent policy with any RL algorithm.
    $\pi \leftarrow$ RL_update$(\pi, \mathcal{D})$
end for

B All prompts in the experiments

For completeness, we include examples of the prompts that we have used in this section. In particular, in Figure 8, we have provided the prompt that we used for the Claude-Sonnet-3 autonomous evaluator to evaluate the success for the task completion for all tasks in WebArena. A similar is used for all tasks in WebVoyager. In Figure 9, 10, 11 we have included the prompts that we used for generating the proposal tasks for each domain. We used the same prompts with 3 additional website screenshots appended to the messages for PAE + User Demos. It is worth noting that our task proposers are domain-general and have little domain customizations. In particular, for all 13 real-world websites from WebVoyager, we use the same prompt to generate tasks except with the placeholder of "web_name". This shows that our PAE framework can easily scale to multiple websites without the need for domain-specific knowledge. The prompt for zero-shot VLM agents are included in Figure 12, 13, and 14.

C Prompts For Zero-Shot VLM Agents

We also append the prompts (Figure 12, 13, and 14) that we used for the zero-shot baselines including Claude-Sonnet-3, Claude-Sonnet-3.5, Qwen2VL, InternVL2b5, LLaVa-1.6-7B, and LLaVa-1.6-34B. The prompt for WebVoyager tasks largely follow from that used in the prior literature (He et al., 2024). We include additional necessary domain knowledge of the WebArena tasks and evaluation protocols in the prompt that we used for WebArena.

Autonomous Evaluator Prompt

You are an expert in evaluating the performance of a web navigation agent. The agent is designed to help a human user navigate a website to complete a task. Your goal is to decide whether the agent's execution is successful or not.

As an evaluator, you will be presented with three primary components to assist you in your role:

1. Web Task Instruction: This is a clear and specific directive provided in natural language, detailing the online activity to be carried out.
2. Result Response: This is a textual response obtained after the execution of the web task. It serves as textual result in response to the instruction.
3. Result Screenshots: This is a visual representation of the screen showing the result or intermediate state of performing a web task. It serves as visual proof of the actions taken in response to the instruction.

– You SHOULD NOT make assumptions based on information not presented in the screenshot when comparing it to the instructions.

– Your primary responsibility is to conduct a thorough assessment of the web task instruction against the outcome depicted in the screenshot and in the response, evaluating whether the actions taken align with the given instructions.

– NOTE that the instruction may involve more than one task, for example, locating the garage and summarizing the review. Failing to complete either task, such as not providing a summary, should be considered unsuccessful.

– NOTE that the screenshot is authentic, but the response provided by LLM is generated at the end of web browsing, and there may be discrepancies between the text and the screenshots.

– NOTE that if the content in the Result response is not mentioned on or different from the screenshot, mark it as not success.

You should explicilt consider the following criterions:

- Whether the claims in the response can be verified by the screenshot. E.g. if the response claims the distance between two places, the screenshot should show the direction. YOU SHOULD EXPECT THAT THERE IS A HIGH CHANCE THAT THE AGENT WILL MAKE UP AN ANSWER NOT VERIFIED BY THE SCREENSHOT.
- Whether the agent completes EXACTLY what the task asks for. E.g. if the task asks to find a specific place, the agent should not find a similar place.

In your responses: You should first provide thoughts EXPLICITLY VERIFY ALL THREE CRITERIONS and then provide a definitive verdict on whether the task has been successfully accomplished, either as 'SUCCESS' or 'NOT SUCCESS'.

A task is 'SUCCESS' only when all of the criteria are met. If any of the criteria are not met, the task should be considered 'NOT SUCCESS'.

Figure 8: The prompt used by the autonomous evaluator for Claude-Sonnet-3. Same prompt is used to evaluate tasks from WebArena websites. The evaluator takes as inputs the task description, the response from the agent's ANSWER action, and last three screenshots in the trajectory. The evaluation result is a binary verdict of 'SUCCESS' or 'NOT SUCCESS'.

Task Proposer Prompt for WebVoyager
{"web_name": "Apple", "id": "Apple–40", "ques": "Find the pricing and specifications for the latest Mac Studio model, including the available CPU and GPU options.", "web": "https://www.apple.com/"}
We are training a model to navigate the web. We need your help to generate instructions. With the examples provided above, please give 25 more example tasks for the model to learn from in the domain of {web_name}. You should imagine tasks that are likely proposed by a most likely user of this website. A few demos of users navigating through the web are provided above.
YOU SHOULD MAKE USE OF THE DEMOS PROVIDES TO GENERATE TASKS, SO THAT YOUR TASKS ARE REALISTIC AND RELEVANT TO THE WEBSITE.
Please follow the corresponding guidelines:
1)First output your thoughts first on how you should come up with diverse tasks that examine various capabilities on the particular website, and how these tasks reflect the need of the potential user. Then you should say 'Output:' and then followed by the outputs STRUCTURED IN JSONL FORMAT. You should not say anything else in the response.
2)PLEASE MAKE SURE TO HAVE 25 examples in the response!!!
3)Your proposed tasks should be DIVERSE AND COVER A WIDE RANGE OF DIFFERENT POSSIBILITIES AND DIFFICULTY in the domain of {web_name}. Remember, your job is to propose tasks that will help the model learn to navigate the web to deal with various real world requests.
4)Your task should be objective and unambiguous. The carry-out of the task should NOT BE DEPENDENT on the user's personal information such as the CURRENT TIME OR LOCATION.
5)You should express your tasks in as diverse expressions as possible to help the model learn to understand different ways of expressing the same task.
6)Your tasks should be able to be evaluated OBJECTIVELY. That is, by looking at the last three screenshots and the answer provided by an agent, it should be possible to tell without ambiguity whether the task was completed successfully or not.
7)Your tasks should require a minimum completion steps from 3 to 7 steps, your tasks should have a diverse coverage in difficulty as measured by the minimum completion step. I.E. You should propose not only tasks that may take more than 4 steps to complete but also tasks that can be completed within 3 steps.
8)Humans should have a 100% success rate in completing the task.
9)Your tasks should be able to be completed without having to sign in to the website.

Figure 9: Prompts used by Claude-Sonnet-3 for proposing tasks in WebVoyager experiments. For PAE + User Demos, we use the same prompt with additional user demos appended to the message.

D Details for SFT

SFT for WebVoyager. As shown in Table 1, unlike proprietary VLMs, none of the open-source VLM agent is able to follow the instructions and achieve non-trivial performances in real-world web navigation tasks in the zero-shot manner. Such models can rarely get success rewards in the process of RL, thus leading to very slow convergence. To "warm-up" the open-source VLM agent to achieve a non-trivial performance at the start of RL training, we turn to enhancing the performances with SFT before RL. Note that the SFT process may not be needed if the base VLM agent model can already achieve non-trivial performances such as Claude 3 Sonnet. To prevent data contamination, we gather 85 out-of-distribution real-world websites (listed in Figure 15 and 16), and collect 11220 trajectories in total using Claude 3 Sonnet with the prompt specified in Figure 12. The average trajectory success rate is 25% as measured by our Claude 3 Sonnet evaluator. Each action in the

Task Proposer Prompt for WebArena Map

{"web_name": "map", "id": "map–2", "ques": "Tell me the full address of all international airports that are within a driving distance of 50 km to University of California, Berkeley"}
{"web_name": "map", "id": "map–10", "ques": "I will arrive San Francisco Airport soon. Provide the name of a Hilton hotel in the vicinity, if available. Then, tell me the the shortest walking distance to a supermarket from the hotel."}
{"web_name": "map", "id": "map–17", "ques": "Check if the ikea in pittsburgh can be reached in one hour by car from hobart street"}

We are training a model to navigate the web. We need your help to generate instructions. With the examples provided above, please give 25 more example tasks for the model to learn from in the domain of OpenStreetMap. You should imagine who is the most likely user for the website and propose tasks that are likely to be proposed by this user. Please follow the corresponding guidelines:
1)First output your thoughts first on how you should come up with diverse tasks that examine various capabilities on the particular website, and how these tasks reflect the need of the potential user. Then you should say 'Output:' and then followed by the outputs STRUCTURED IN JSONL FORMAT. You should not say anything else in the response.
2)PLEASE MAKE SURE TO HAVE 25 examples in the response!!!
3)Your proposed tasks should be DIVERSE AND COVER A WIDE RANGE OF DIFFERENT POSSIBILITIES AND DIFFICULTY in the domain of OpenStreetMap. Remember, your job is to propose tasks that will help the model learn to navigate the web to deal with various real world requests. 4)Your task should be objective and unambiguous. The carry-out of the task should NOT BE DEPENDENT on the user's personal information such as the CURRENT TIME OR LOCATION.
5)You should express your tasks in as diverse expressions as possible to help the model learn to understand different ways of expressing the same task.
6)Your tasks should be able to be evaluated OBJECTIVELY. That is, by looking at the last three screenshots and the answer provided by an agent, it should be possible to tell without ambiguity whether the task was completed successfully or not.
7)Your tasks should require a minimum completion steps from 3 to 7 steps, your tasks should have a diverse coverage in difficulty as measured by the minimum completion step. I.E. You should propose not only tasks that may take more than 4 steps to complete but also tasks that can be completed within 3 steps.
8)Humans should have a 100% success rate in completing the task.
9)Your tasks should be able to be completed without having to sign in to the website.

Figure 10: Prompts used by Claude-Sonnet-3 for proposing WebArena Tasks for Map. For PAE + User Demos, we use the same prompt with additional user demos appended to the message. For this domain only, we provide three hand-written in-domain examples to set the right difficulty for the task proposer. Such in-domain examples are not needed for all other domains including Reddit and OneStopMarket, and other real-world WebVoyager websites.

trajectories contains both thoughts and actual web actions shown in Figure 2. All 11220 trajectories are used for SFT. RL training is carried out on top of the SFT checkpoint.

SFT for WebArena. In our preliminary experiments, we found that the SFT checkpoint trained on real-world websites do not generalize well to self-hosted websites on WebArena. This is potentially because of the distribution shift between real-world commercial websites and self-hosted websites. For example, most real-world map websites such as Google Maps and Apple Maps support advanced fuzzy search capabilities such as "pittsburgh to new york" while OpenStreetMap from WebArena will not return any results with such queries. Therefore, we collect 3000 Claude 3 Sonnet generated trajectories each from OpenStreetMap, Reddit, and OneStopMarket websites from WebArena. We

Task Proposer Prompt for WebArena Reddit and OneStopMarket

{"web_name": "Apple", "id": "Apple–40", "ques": "Find the pricing and specifications for the latest Mac Studio model, including the available CPU and GPU options.", "web": "https://www.apple.com/"}

We are training a model to navigate the web. We need your help to generate instructions. With the examples provided above, please give 25 more example tasks for the model to learn from in the domain of {web_name}.

You should provide tasks in the DOMAIN OF {web_name}.

Please follow the corresponding guidelines: 1)First answer how many screenshots are provided and describe in detail the functions of the website that you see from each of the screenshot. Then output your thoughts first. Then you should say 'Output:' and then followed by the outputs STRUCTURED IN JSONL FORMAT. You should not say anything else in the response.

2)PLEASE MAKE SURE TO HAVE 25 examples in the response!!!

4)Your task should start from the home page of the website instead of the shown screenshots.

5)Your task does not need to be the same as real users would do, but it should examine diverse capabilities of the agent to do web navigartion.

6)Your tasks should examine the VERY BASIC functions of the website and should not require complicated web page operations. They can be completed within 5 steps.

7)THIS DOMAIN IS A SELF-HOSTED STATIC DOMAIN AND DIFFERENT FROM POPULAR WEBSITES, DO NOT ASSUME ANY INFORMATION NOT PROVIDED IN THE SCREENSHOTS.

8)Your tasks should examine the capability of the web agent to find some information on the website, navigating to some specific web pages. Do not propose tasks that involve making actual modifications to the websites.

9)Your tasks should result in the agent landing in a single groundtruth web page or finding a single grounth truth answer. The landed webpage can be some specific categories, a drafted post, some search results, or even the homepage of the website. When the task is to to find some information, specify exactly what information the agent should find such as the price, the number of comments, the title, etc. It can also be information about the current account.

Figure 11: Prompts used by Claude-Sonnet-3 for proposing WebArena Tasks for Reddit and OneStopMarket. For PAE + User Demos, we use the same prompt with additional user demos appended to the message.

use the prompts from Figure 13 and 14 for the Claude agent. The average trajectory success rate is 27% as measured by our Claude 3 Sonnet evaluator. The SFT checkpoint for WebArena is fine-tuned from the SFT checkpoint for WebVoyager.

E Additional Results on WebArena

For completeness, we have also provided additional experiment results of different models from Table 2 in the original task split of WebArena (Zhou et al., 2024a). As shwon in the comparison results presented in Table 4, even SOTA proprietary VLM agents like Claude 3 Sonnet struggle with the tasks in WebArena with a success rate of only 14.6% with set-of-marks observations and chain-of-thought prompting. After performing SFT using the demonstrations generated by Claude 3 Sonnet, LLaVa-7B SFT can only achieve 1.4% and 5.8% success rate on PostMill and OneStopMarket. By manually inspecting the roll-out trajectories generated by LLaVa-SFT, we found that around half of the successful trajectories on those two websites are false positives from the WebArena evaluator. In these trajectories, the agent simply guessed the answer to be "no" or "N/A" where the ground truth happens to be that the task is not executable. As a result, the actual success rate on those two

Zero-Shot VLM Agent Prompt for WebVoyager (1/2)
Imagine you are a robot browsing the web, just like humans. Now you need to complete a task. In each iteration, you will receive an Observation that includes a screenshot of a webpage and some texts. This screenshot will feature Numerical Labels placed in the TOP LEFT corner of each Web Element. Carefully analyze the visual information to identify the Numerical Label corresponding to the Web Element that requires interaction, then follow the guidelines and choose one of the following actions:
1. Click a Web Element.
2. Delete existing content in a textbox and then type content.
3. Scroll up or down. Multiple scrolls are allowed to browse the webpage. Pay attention!! The default scroll is the whole window. If the scroll widget is located in a certain area of the webpage, then you have to specify a Web Element in that area. I would hover the mouse there and then scroll.
4. Wait. Typically used to wait for unfinished webpage processes, with a duration of 5 seconds.
5. Go back, returning to the previous webpage.
6. Google, directly jump to the Google search page. When you can't find information in some websites, try starting over with Google.
7. Answer. This action should only be chosen when all questions in the task have been solved.

Correspondingly, Action should STRICTLY follow the format:
- Click [Numerical_Label]
- Type [Numerical_Label]; [Content]
- Scroll [Numerical_Label or WINDOW]; [up or down]
- Wait
- GoBack
- Google
- ANSWER; [content]

Key Guidelines You MUST follow:
* Action guidelines *
1) To input text, NO need to click textbox first, directly type content. After typing, the system automatically hits 'ENTER' key. Sometimes you should click the search button to apply search filters. Try to use simple language when searching.
2) You must Distinguish between textbox and search button, don't type content into the button! If no textbox is found, you may need to click the search button first before the textbox is displayed.
3) Execute only one action per iteration.
4) STRICTLY Avoid repeating the same action if the webpage remains unchanged. You may have selected the wrong web element or numerical label. Continuous use of the Wait is also NOT allowed.
5) When a complex Task involves multiple questions or steps, select "ANSWER" only at the very end, after addressing all of these questions (steps). Flexibly combine your own abilities with the information in the web page. Double check the formatting requirements in the task when ANSWER.
* Web Browsing Guidelines *
1) Don't interact with useless web elements like Login, Sign-in, donation that appear in Webpages. Pay attention to Key Web Elements like search textbox and menu.
2) Vsit video websites like YouTube is allowed BUT you can't play videos. Clicking to download PDF is allowed and will be analyzed by the Assistant API.
3) Focus on the numerical labels in the TOP LEFT corner of each rectangle (element). Ensure you don't mix them up with other numbers (e.g. Calendar) on the page.
4) Focus on the date in task, you must look for results that match the date. It may be necessary to find the correct year, month and day at calendar.
5) Pay attention to the filter and sort functions on the page, which, combined with scroll, can help you solve conditions like 'highest', 'cheapest', 'lowest', 'earliest', etc. Try your best to find the answer that best fits the task.

Your reply should strictly follow the format:
Thought: {Your brief thoughts (briefly summarize the info that will help ANSWER)}
Action: {One Action format you choose}

Then the User will provide:
Observation: {A labeled screenshot Given by User}

Figure 12: The prompt used for all zero-shot VLM agents for WebVoyager websites, including Claude-Sonnet-3, Claude-Sonnet-3.4, Qwen2-VL, InternVL-2.5-XComposer, LLaVa-1.6-7B, and LLaVa-1.6-34B.

Zero-Shot VLM Agent Prompt for WebArena (2/2)
Imagine you are a robot browsing the web, just like humans. Now you need to complete a task. In each iteration, you will receive an Observation that includes a screenshot of a webpage, some texts and the accessibility tree of the webpage. This screenshot will feature Numerical Labels placed in the TOP LEFT corner of each Web Element. The accessbility tree contains information about the web elements and their properties. The numrical labels in the screenshot correspond to the web elements in the accessibility tree.
Carefully analyze the visual information to identify the Numerical Label corresponding to the Web Element that requires interaction, then follow the guidelines and choose one of the following actions:
1. Click a Web Element.
2. Delete existing content in a textbox and then type content.
3. Scroll up or down. Multiple scrolls are allowed to browse the webpage. Pay attention!! The default scroll is the whole window. If the scroll widget is located in a certain area of the webpage, then you have to specify a Web Element in that area. I would hover the mouse there and then scroll.
4. Wait. Typically used to wait for unfinished webpage processes, with a duration of 5 seconds.
5. Go back, returning to the previous webpage.
6. Answer. This action should only be chosen when all questions in the task have been solved.

Correspondingly, Action should STRICTLY follow the format:
- Click [Numerical_Label]
- Type [Numerical_Label]; [Content]
- Scroll [Numerical_Label or WINDOW]; [up or down]
- Wait
- GoBack
- ANSWER; [content]

Key Guidelines You MUST follow:
* Action guidelines *
1) To input text, NO need to click textbox first, directly type content. After typing, the system automatically hits 'ENTER' key. Sometimes you should click the search button to apply search filters. Try to use simple language when searching.
2) You must Distinguish between textbox and search button, don't type content into the button! If no textbox is found, you may need to click the search button first before the textbox is displayed.
3) Execute only one action per iteration.
4) STRICTLY Avoid repeating the same action if the webpage remains unchanged. You may have selected the wrong web element or numerical label. Continuous use of the Wait is also NOT allowed.
5) When a complex Task involves multiple questions or steps, select "ANSWER" only at the very end, after addressing all of these questions (steps). Flexibly combine your own abilities with the information in the web page. Double check the formatting requirements in the task when ANSWER.
6) If you can't find the answer using the given website because there is no such information on the website after some attempts, you should report "N/A" as the answer to represent that the task is impossible to solve with the given website. You may have 15 steps to try to solve the task.
7) Only provide answer based on the information from the image, make sure the answer is consistent with the image, don't hallucinate any information that is not based on image.

* Web Browsing Guidelines *
1) Focus on the numerical labels in the TOP LEFT corner of each rectangle (element). Ensure you don't mix them up with other numbers (e.g. Calendar) on the page.
2) Pay attention to the filter and sort functions on the page, which, combined with scroll, can help you solve conditions like 'highest', 'cheapest', 'lowest', 'earliest', etc. Try your best to find the answer that best fits the task.

Figure 13: The prompt used for all zero-shot VLM agents for WebArena websites, including Claude-Sonnet-3, Claude-Sonnet-3.4, Qwen2-VL, InternVL-2.5-XComposer, LLaVa-1.6-7B, and LLaVa-1.6-34B. To be continued in Figure 14.

Zero-Shot VLM Agent Prompt for WebArena

* OpenStreetMap Usage Guidelines *
1) When you need to search the address of a location, you can just type the location in the 'search' bar. You don't need to use the directions button to get the address. The directions button is only used when you need to find the distance/walk/drive time between two locations.
2) When you are trying to search for a location, you may get no results. This is because the OpenStreetMap does not support approximate search. You may try to search some alternative keywords or try to find the location by yourself. Note that openstreet map does not support search phrase like 'Cafe near CMU", you should try to find it by yourself.
3) When you need to find the distance/walk/drive time between two locations, you should FIRST CLICK ON THE DIRECTIONS BUTTON (drawn as two arrows), to the right of the 'Go' Button and usually labeled as [10] or [11]. AND ONLY INPUTTING THE TWO LOCATIONS AFTER CLICKING ON THE DIRECTIONS BUTTON WHEN THE DIRECTIONS SEARCH BARS ARE SHOWN.
4) When you are trying to type some locations in the directions search bar, sometimes you may receive an alert of 'couldn't locate' followed by the location you typed. This means the location you typed is not found in the map. Do not immeadiately try something else. You need to quit the direction and find the precise name of this location by searching it in the map first.
5) When you search the walk/drive/bike time, make sure that you are USING THE RIGHT MODE OF TRANSPORTATION. The default mode is usually set to 'Drive'.
6) When you need to get the DD of some location, you need to click the location shown in the search result in the left part of the screen. The DD will be shown then starting with 'Location:'.
7) When you need to answer the zip code of some location, you should directly answer the 5-digit zip code. The answer shoule be "15232" instead of "The zip code of the location is 15232". Note that the zipcode will be displayed in the search result, you don't need to click the location to the information page to find the zip code.
8) When you need to answer the phone of some location, please omit the part of the country code. The answer should be "4122683259" instead of "+1 412 268 3259".

* Reddit Usage Guidelines *
1) You are already in the reddit website, though you may not see the 'reddit' in any part of the screenshot. You do not need to further navigate to the reddit website.
2) When you want to find a subreddit, you need to first navigate to Forums to see the list of subreddits. Under forums, you will see only a subset of subreddits. To get the full list of subreddits, you need to navigate to the Alphabetical option. To know you can see the full list of subreddits, you will see 'All Forums' in the observation. Often you will not find a focused subreddit that exactly matches your query. In that case, go ahead with the closest relevant subreddit. To know that you have reached a subreddit successfully, you will see '/f/subreddit_name' in the observation.
3) When you want to post forum in reddit, remember to fill up all the content, then click the button 'Create forum'. The button maybe located below out of the screenshot, you need to scroll down to find it.
4) When you want to ask or post something in a subreddit, you need to first find that subreddit and then finish the work. 5) forums and subreddits are the same thing.

Your reply should strictly follow the format:
Thought: Your brief thoughts (briefly summarize the info that will help ANSWER)
Action: One Action format you choose

Then the User will provide:
Observation: A labeled screenshot Given by User
Remember only execute one action in each step. For example, 'Action: Type [8]; CMU, Type [9] Pittsburgh' is not allowed. You should execute the action 'Type [8]; CMU' first, then 'Type [9] Pittsburgh' in the next step.

Remember to always make your answer simple and clear. For example, if you want to report the zip code of some location, always say "ANSWER; 06516" instead of "The zip code of the location is 06516".

Figure 14: The prompt used for all zero-shot VLM agents for WebArena websites, including Claude-Sonnet-3, Claude-Sonnet-3.4, Qwen2-VL, InternVL-2.5-XComposer, LLaVa-1.6-7B, and LLaVa-1.6-34B. Continued from Figure 13.

websites is lower than 2%, leaving very sparse reward signals for RL to make meaningful improvements. We therefore rewrote the tasks on PostMill and OneStopMarket to be easier and report the performances of PAE in Table 2.

		OpenStreetMap	PostMill	OneStopMarket	Average
Proprietary	Claude 3 Sonnet	24.3	10.6	11.2	14.6
Open-source	Qwen2VL-7B	0.7	0.0	1.3	0.7
	InternVL2.5-8B	2.6	0.2	3.3	2.3
	LLaVa-7B	0.0	0.0	0.0	0.0
Ours	LLaVa-7B SFT	15.2	1.4	5.8	7.2

Table 4: Success rate comparisons across different domains from WebArena. Success and failure are detected with ground-truth verification functions. All tasks from OpenStreetMap are kept unchanged from WebArena task splits.

F LIMITATIONS

Despite the progress of PAE for open-source VLM agents, there are still some limitations due to practical constraints. First of all, due to the limitations in fundamental capabilities of open-source base VLM models, our models trained with PAE are still inferior to state-of-the-art proprietary models in realistic web navigations, where advanced reasoning and planning capabilities are required. Moreover, because of the hallucination issues of open-source VLMs, we found them unreliable to serve as the autonomous evaluators and had to rely on advanced proprietary VLMs for judging the success and providing rewards. Finally, because of the dynamic nature of the real websites that we are using, some of our results may not be produced exactly, although a significant improvement from PAE should still be observed.

G HYPERPARAMETERS

We include the hyperparameters that we have used in Table 5. As shown in the table, the only hyperparameters that PAE have on top of standard supervised fine-tuning are number of trajectories to collect in each global iteration in Algorithm 1, number of proposed tasks from the task proposer before RL training, and the number of seen screenshots for the evaluator. In our experiments, we found that PAE is relatively not sensitive to the choices of these hyperparameters, showing the robustness of PAE .

H MORE QUALITATIVE EXAMPLES

In this section, we present additional qualitative examples of agent trajectories while performing tasks to further demonstrate the effectiveness of our PAE . We will also release the full dataset for further analysis.

Full trajectories of examples in Section 6. Here, we provide the complete trajectories for the examples discussed in the qualitative comparisons in Section 6, as shown in Figures 17–20. We detail the agent's thoughts and actions at each time step throughout the entire trajectory.

Some representative successful trajectories. We also showcase representative successful trajectories generated by the LLaVa-7B PAE model to highlight the strengths of our method. In Figure 21, the task is ”Show the most played games on Steam, and tell me the number of players currently in-game.” In Figure 22, the task is ”Find out the starting price for the most recent model of the iMac on the Apple website.” In Figure 23, the task is “Look up the use of modal verbs in the grammar section for expressing possibility (e.g., ’might’, ’could’, ’may’) and find examples of their usage in sentences on the Cambridge Dictionary.” Finally, in Figure 24, the task is ”Search for plumbers available now but not open 24 hours in Orlando, FL.”

Table 5: Hyperparameters for All Experiments

Environment	Hyperparameter	Considered	Chosen
WebVoyager	learning rate	{2e-5, 5e-5, 2e-4}	2e-5
	rollout trajectories	{512, 1024, 2048, 4096}	4096
	rollout temperature	{0.4, 1.0, 2.0}	1.0
	maximum gradient norm	{0.01}	0.01
	actor updates epochs per iteration	{1, 2, 4, 8, 20}	4
	batch size	{8}	8
	gradient accumulation size	{16, 32}	32
	number of proposed tasks	{10000, 50000, 100000}	100000
	number of seen screenshots for evaluator	{1, 3}	3
WebArena Easy	learning rate	{2e-5, 5e-5, 2e-4}	2e-5
	rollout trajectories	{512, 1024, 2048, 4096}	2048
	rollout temperature	{0.4, 1.0, 2.0}	1.0
	maximum gradient norm	{0.01}	0.01
	actor updates epochs per iteration	{1, 2, 4, 8, 20}	2
	batch size	{8}	8
	gradient accumulation size	{16, 32}	32
	number of proposed tasks	{10000, 30000, 100000}	30000
	number of seen screenshots for evaluator	{1, 3}	3

Table 6: Hyperparameters for PAE for WebVoyager and WebArena Easy experiments.

Detailed explanations of the error type definitions. To clarify the precise definition of the different error categories used in Section 6, we provide more comprehensive explanations with example trajectories:

(1) Low-level skill missing errors refer to cases where the agent has a reasonable plan to solve the problem but fails to execute precise actions on the website, such as not knowing which button to click to reach the desired page. We classify trajectories where the agent seems to follow a reasonable plan but struggles with specific operations into this category. For example, in Figure 25, the task is "Find the Easy Vegetarian Spinach Lasagna recipe on Allrecipes and tell me what the latest review says." The agent attempts to search for the desired item but fails to click the correct button to reach the detailed page in the search results.

(2) High-level planning or reasoning errors occur when the agent fails to generate a complete plan or cannot reason correctly with the website's screenshots to solve the task. Trajectories where the agent cannot devise a plan for complex tasks or misinterprets the screenshot's content are categorized as such. For instance, in Figure 26, the task is "Give 12 lbs of 4-cyanoindole, converted to molar and indicate the percentage of C, H, N." The agent should first search on Google about the chemical definition of 4-cyanoindole, then use WolframAlpha to calculate the result. However, the agent fails to get the precise definition of 4-cyanoindole, and doesn't know how to solve the task.

(3) Visual hallucinations refer to instances where the agent generates fabricated responses not supported by the screenshot. The agent might, for example, claim to have found a requested product while still on the Google homepage or provide an incorrect answer even when on the correct page. In Figure 27, the task is "Find out the trade-in value for an iPhone 13 Pro Max in good condition on the Apple website". The agent claims with a very detailed answer but actually it never access any page related to the trade-in on the website.

(4) Timeouts occur when the agent is on the right track to solving the task but cannot complete it within the maximum number of steps. This error indicates that the agent did nothing wrong but was constrained by the environment's step limits. For example, in Figure 28, the task is "Go to the Plus section of Cambridge Dictionary, find Image quizzes, and complete an easy quiz about Animals. Tell me your final score." The agent reaches the maximum time step limit (10) while attempting to finish the quiz.

(5) Technical issues are not caused by the agent but by environmental problems, such as websites being down or connection failures. In Figure 29, the ChromeDriver crashes after a valid operation.

(6) Others include less frequent error types, such as when the task itself is impossible to complete.

Out-of-distribution Websites of WebVoyager for SFT (1/2)
Allrecipes:
Simply Recipes: `https://www.simplyrecipes.com`
Food Network: `https://www.foodnetwork.com`
Taste of Home: `https://www.tasteofhome.com`
Yummly: `https://www.yummly.com`
Food.com: `https://www.food.com`

Amazon:
eBay: `https://www.ebay.com`
Walmart: `https://www.walmart.com`
Target: `https://www.target.com`
Best Buy: `https://www.bestbuy.com`
Alibaba: `https://www.alibaba.com`

Apple:
Samsung: `https://www.samsung.com`
Microsoft: `https://www.microsoft.com`
Sony: `https://www.sony.com`
Google Store: `https://store.google.com`
Dell: `https://www.dell.com`

ArXiv:
SSRN: `https://www.ssrn.com`
ResearchGate: `https://www.researchgate.net`
bioRxiv: `https://www.biorxiv.org`
IEEE Xplore: `https://ieeexplore.ieee.org`
PubMed: `https://pubmed.ncbi.nlm.nih.gov`

GitHub:
GitLab: `https://about.gitlab.com`
Bitbucket: `https://bitbucket.org`
SourceForge: `https://sourceforge.net`
Codebase: `https://www.codebasehq.com`
Gitea: `https://gitea.io`

ESPN:
CBS Sports: `https://www.cbssports.com`
Fox Sports: `https://www.foxsports.com`
NBC Sports: `https://www.nbcsports.com`
Bleacher Report: `https://www.bleacherreport.com`
Sky Sports: `https://www.skysports.com`

Coursera:
edX: `https://www.edx.org`
Udacity: `https://www.udacity.com`
Udemy: `https://www.udemy.com`
FutureLearn: `https://www.futurelearn.com`
Khan Academy: `https://www.khanacademy.org`

Figure 15: A list of 85 websites that we used to collect demonstration trajectories with Claude 3 Sonnet. In total 11220 trajectories were collected with different tasks. These websites were also used for testing the zeroshot generalization of PAE to out-of-distribution websites in Section 5. List continued in Figure 16.

Out-of-distribution Websites of WebVoyager for SFT (2/2)
Cambridge Dictionary:
Merriam-Webster: `https://www.merriam-webster.com`
Dictionary.com: `https://www.dictionary.com`
Oxford Learner's Dictionaries: `https://www.oxfordlearnersdictionaries.com`
Collins English Dictionary: `https://www.collinsdictionary.com`
YourDictionary: `https://www.yourdictionary.com`

BBC News:
CNN: `https://www.cnn.com`
Al Jazeera: `https://www.aljazeera.com`
Reuters: `https://www.reuters.com`
The Guardian: `https://www.theguardian.com`
NBC News: `https://www.nbcnews.com`

Google Maps:
Apple Maps: `https://maps.apple.com`
Bing Maps: `https://www.bing.com/maps`
MapQuest: `https://www.mapquest.com`
Waze: `https://www.waze.com`
Here WeGo: `https://wego.here.com`

Google Search:
Bing: `https://www.bing.com`
Yahoo Search: `https://search.yahoo.com`
DuckDuckGo: `https://duckduckgo.com`
Baidu: `https://www.baidu.com`
Yandex: `https://yandex.com`

Hugging Face:
OpenAI: `https://openai.com`
TensorFlow: `https://www.tensorflow.org`
PyTorch: `https://pytorch.org`
Kaggle: `https://www.kaggle.com`
SpaCy: `https://spacy.io`

Wolfram Alpha:
Google Scholar: `https://scholar.google.com`
Mathway: `https://www.mathway.com`
Symbolab: `https://www.symbolab.com`
Microsoft Math Solver: `https://mathsolver.microsoft.com`
Desmos: `https://www.desmos.com`

Figure 16: A list of 85 websites that we used to collect demonstration trajectories with Claude 3 Sonnet. In total 11220 trajectories were collected with different tasks. These websites were also used for testing the zeroshot generalization of PAE to out-of-distribution websites in Section **??**. List continued from Figure 16.

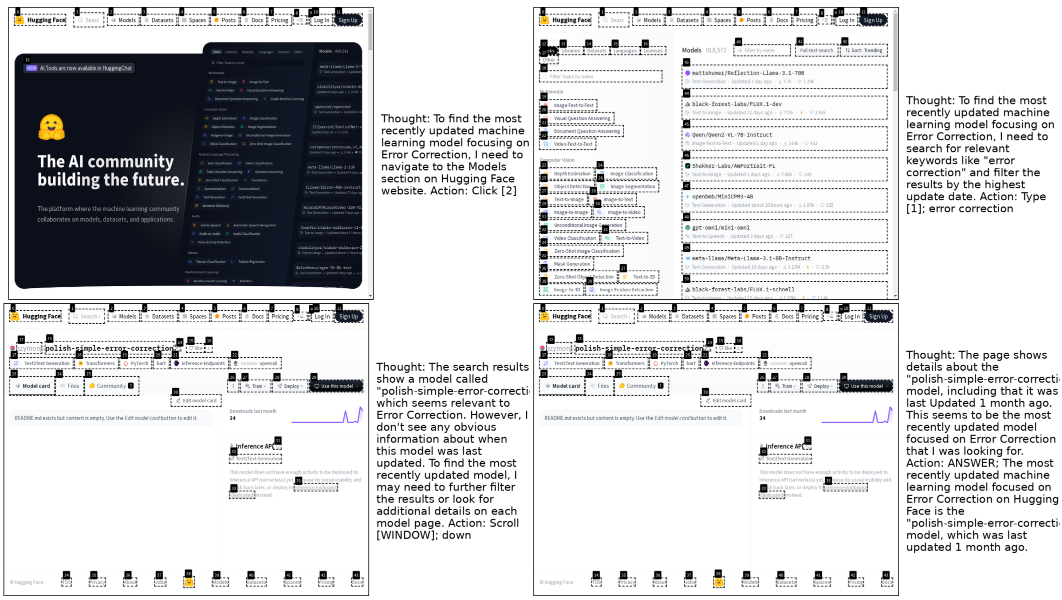


Figure 17: Full trajectories of success trajectory 1 in Figure 7 with task 'Find the most recently updated machine learning model on Huggingface which focuses on Error Correction' executed by model LLaVa-7B PAE.

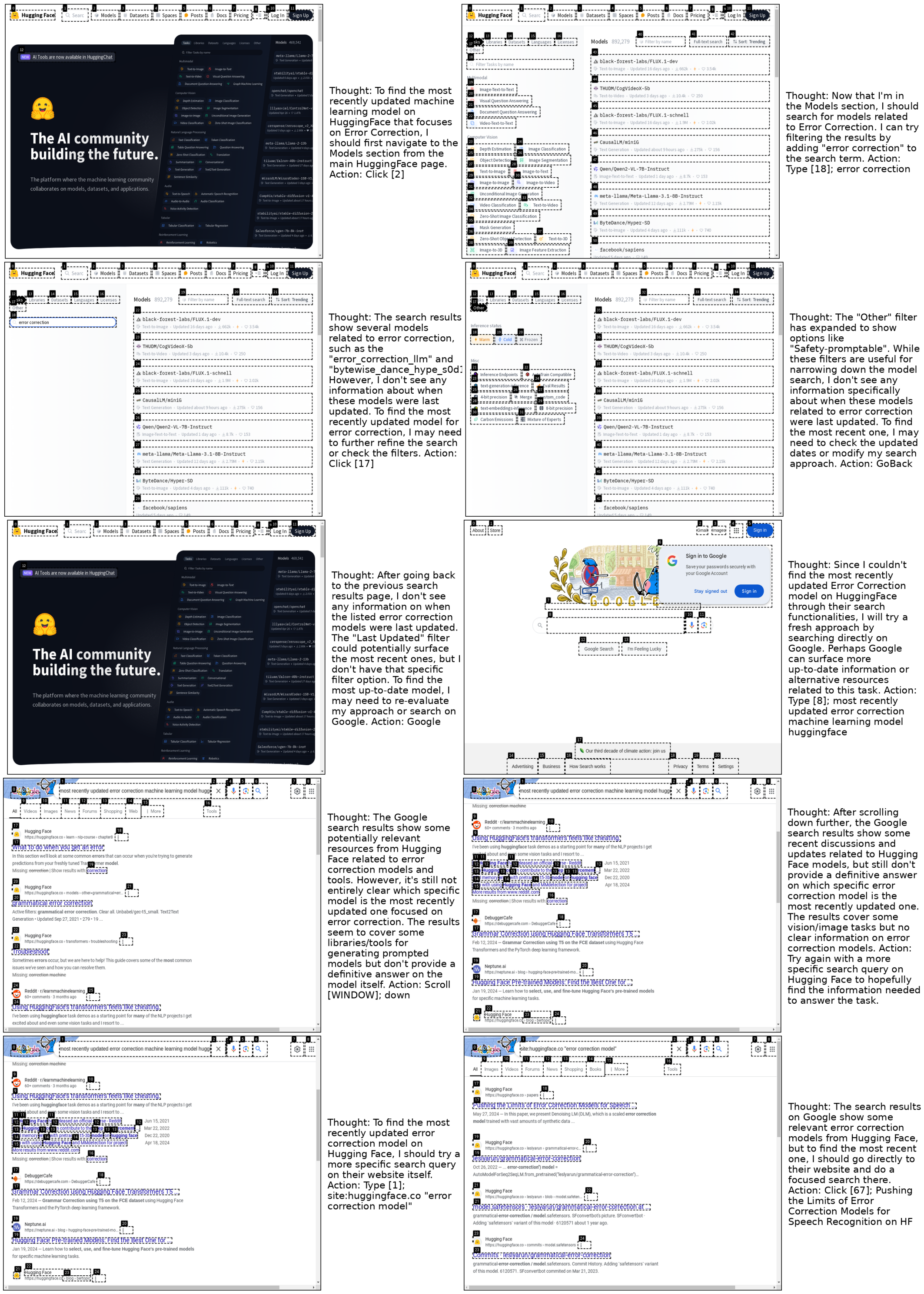


Figure 18: Full trajectories of fail trajectory 1 in Figure 7 with task 'Find the most recently updated machine learning model on Huggingface which focuses on Error Correction' executed by model LLaVa-7B SFT.

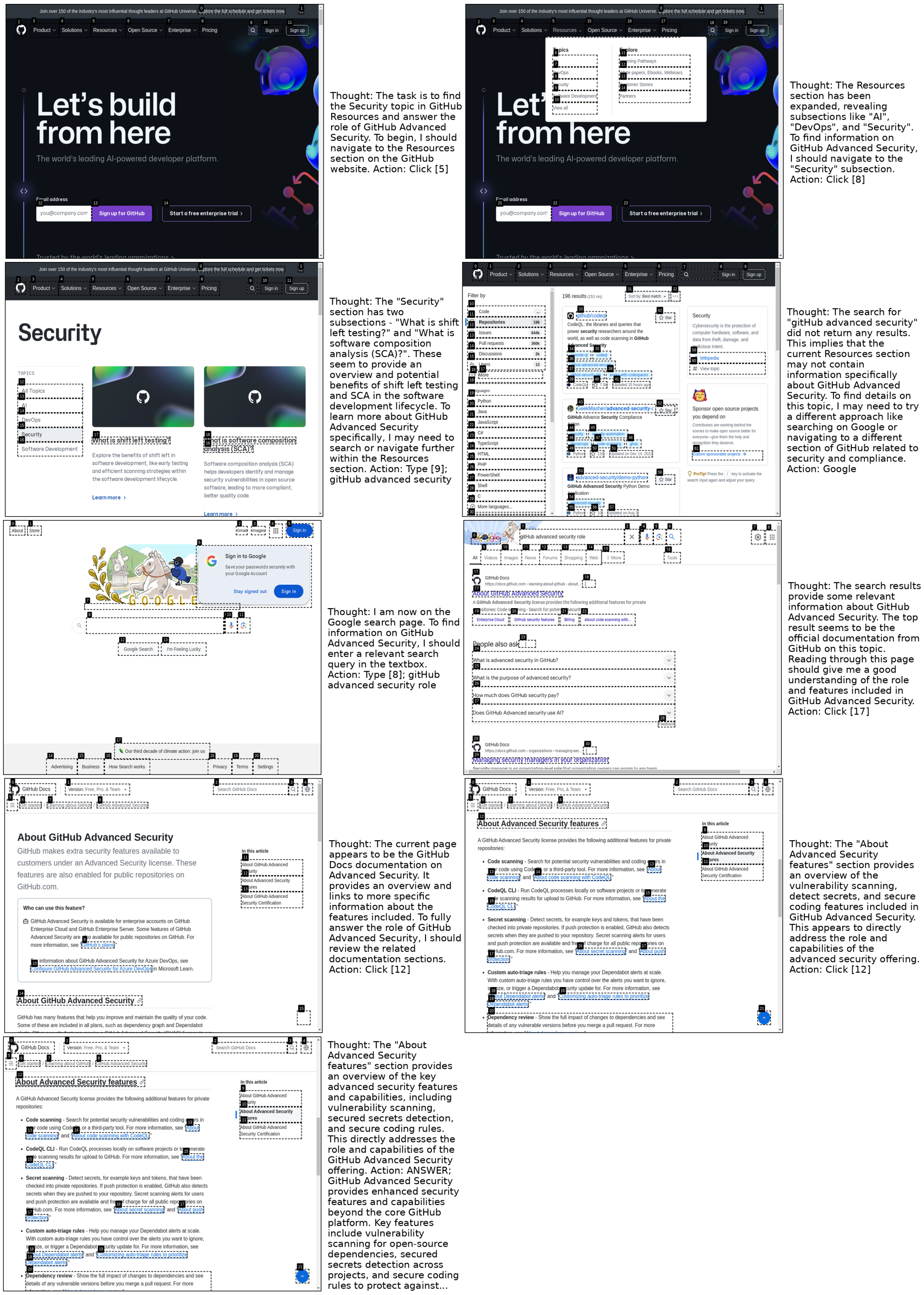


Figure 19: Full trajectories of success trajectory 2 in Figure 7 with task 'Find the Security topic in GitHub Resources and answer the role of GitHub Advanced Security' executed by model LLaVa-7B PAE.

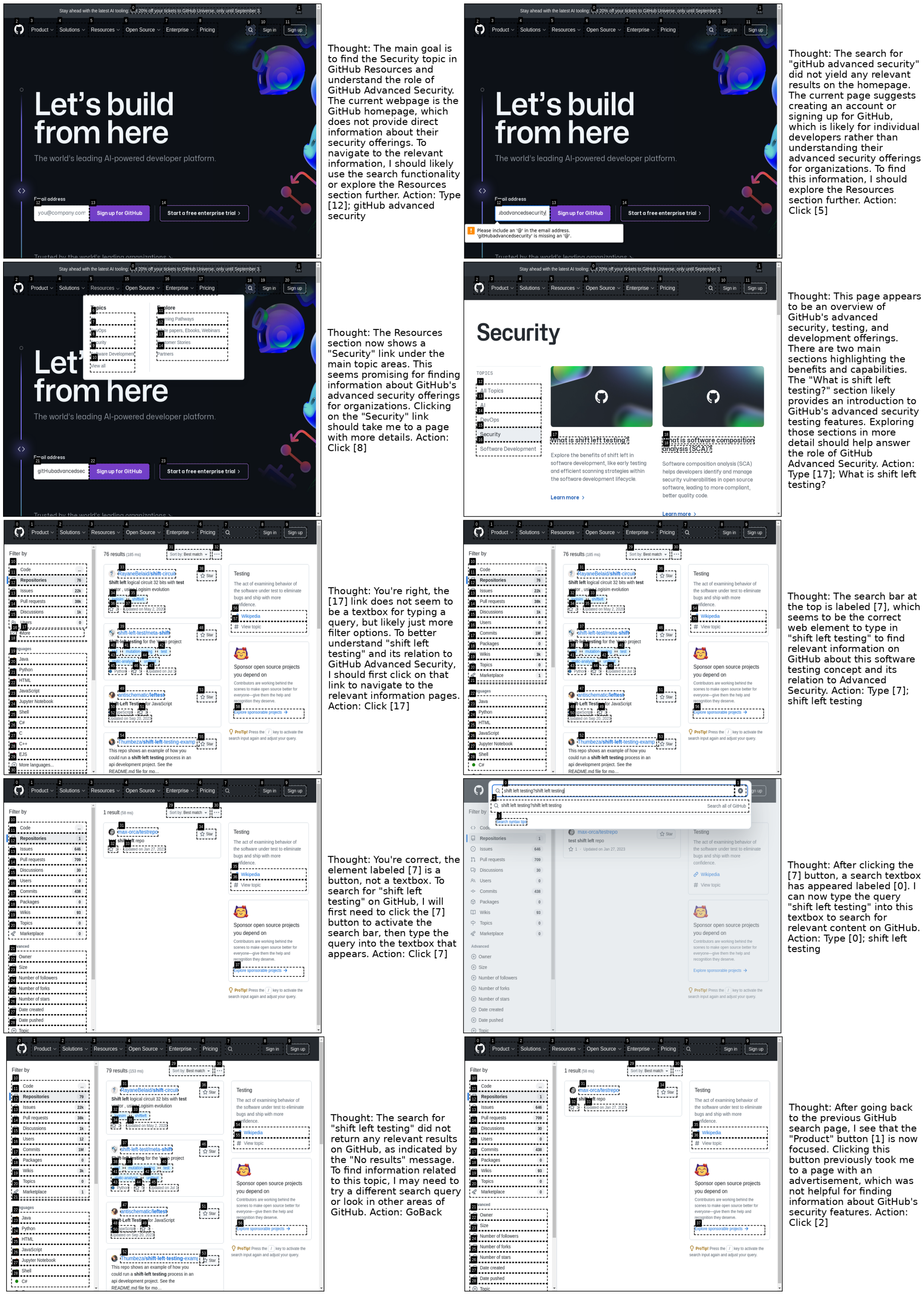


Figure 20: Full trajectories of fail trajectory 2 in Figure 7 with task 'Find the Security topic in GitHub Resources and answer the role of GitHub Advanced Security' executed by model LLaVa-7B SFT.

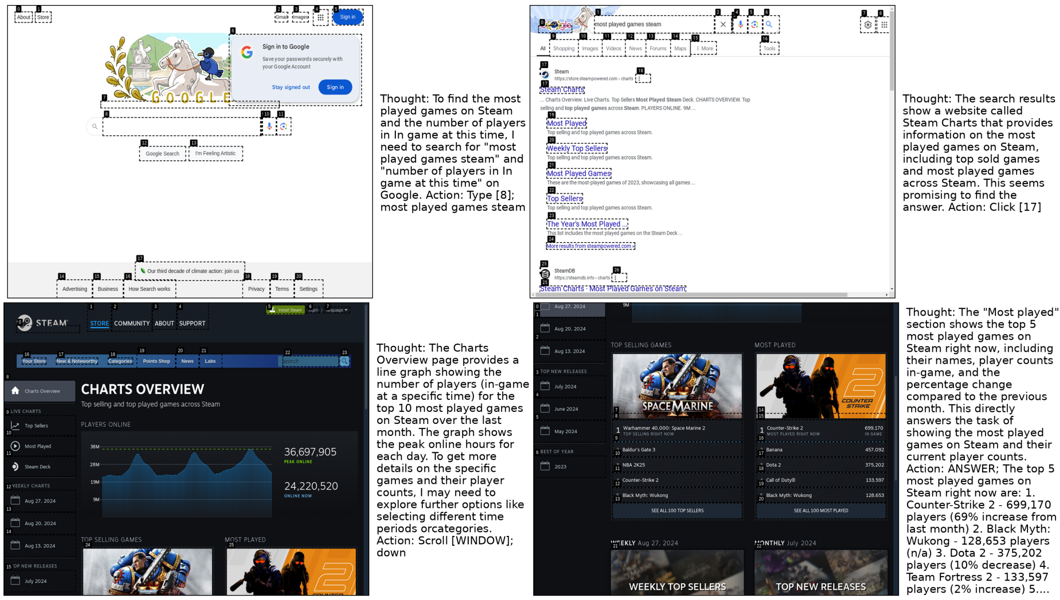


Figure 21: Extra full trajectories of successful trajectory 1 with task 'Show most played games in Steam. And tell me the number of players in In game at this time' executed by model LLaVa-7B PAE.

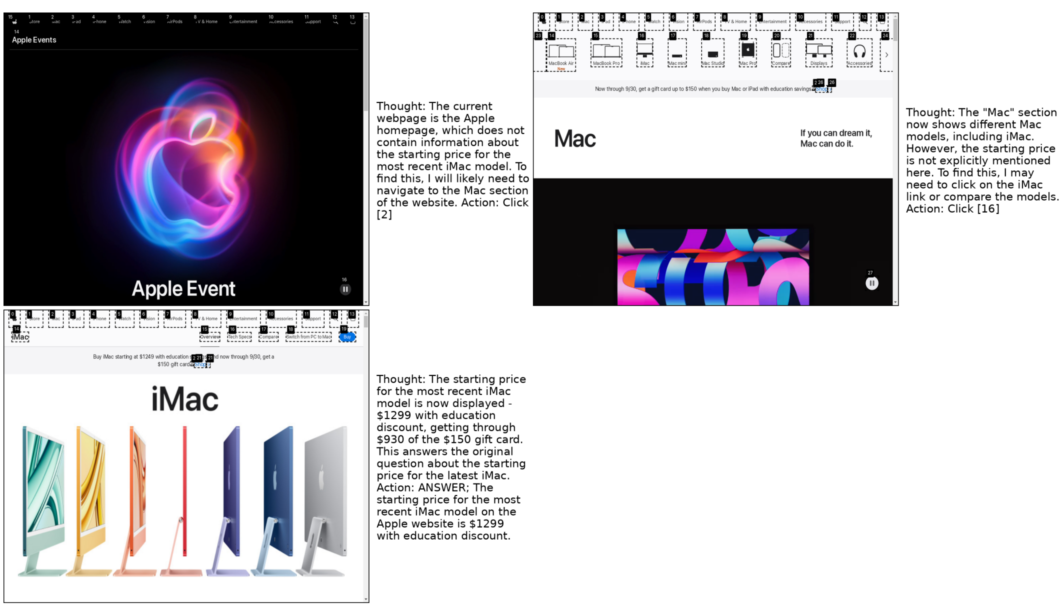
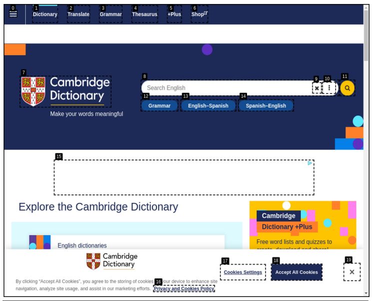
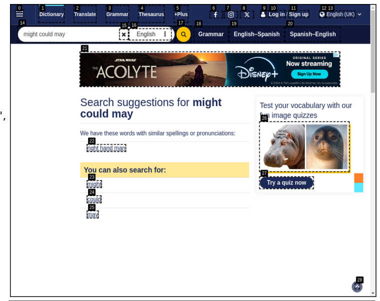


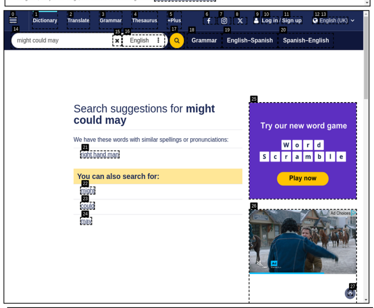
Figure 22: Extra full trajectories of successful trajectory 2 with task 'Find out the starting price for the most recent model of the iMac on the Apple website' executed by model LLaVa-7B PAE.



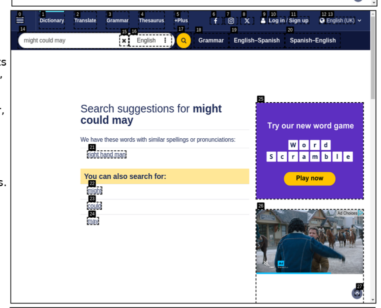
Thought: To find examples of modal verbs like "might", "could", and "may" expressing possibility, I need to search for those words in the grammar section of the Cambridge Dictionary website. Action: Type [8]; might could may



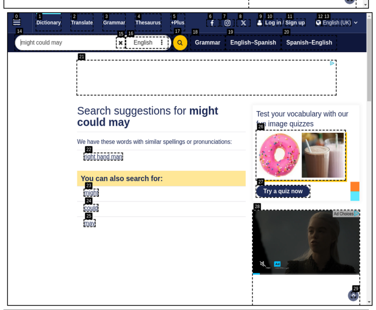
Thought: The search box has been populated with "might could may" which are the modal verbs I wanted to search for expressing possibility. I should now click the search button to get the results. Action: Click [17]



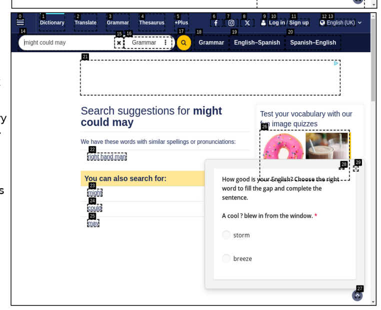
Thought: The search results show that "might", "could", and "may" expressing possibility have been entered correctly. However, it seems there are no results found for these words, which is quite surprising given their common usage and importance as modal verbs. Action: Try a new search with similar spelling or pronunciations, perhaps I missed some letters or the pronunciation might be helpful.



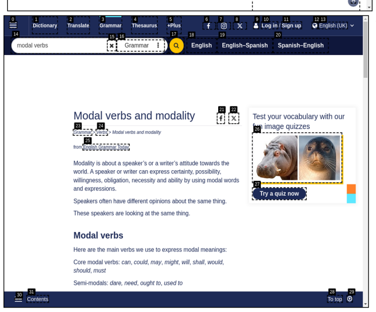
Thought: After multiple unsuccessful searches, I realize I may be missing some key information or steps necessary to find examples and explanations for modal verbs like "might", "could", and "may" on this Cambridge Dictionary website. Action: GoBack



Thought: After going back to the search page, I see there is a "Grammar" option to set the dictionary search to grammar terms. This along with searching directly for "modal verbs" or related terms like "might", "could", "may" may provide the examples and explanations I need. Action: Click [18]; Set dictionary search to Grammar

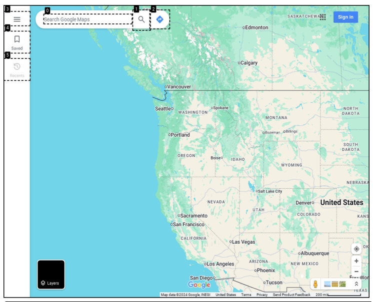


Thought: With the search set to the Grammar section, I now search directly for "modal verbs" which should provide examples and explanations for the usage of modal verbs like "might", "could", "may". Action: Type [14]; modal verbs

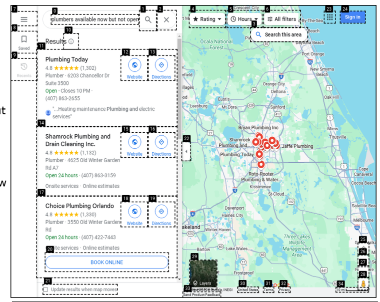


Thought: The search results for "modal verbs" provide a useful definition and examples for the usage of modal verbs like "might", "could", "may" to express possibility, willingness, obligation, necessity and ability. The first result seems to cover all the key points I need about the usage of modal verbs, including: * They express the speaker's attitude towards what they are saying. * Possibility, willingness, obligation, necessity and ability are different options that can be expressed with modal verbs. The example sentences shown illustrate these different uses of modal verbs. Action:...

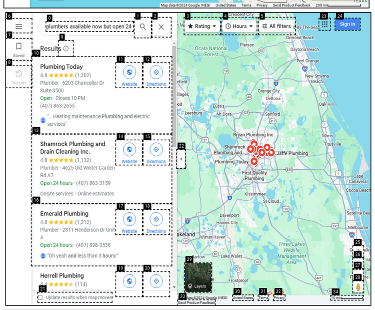
Figure 23: Extra full trajectories of successful trajectory 3 with task ‘Look up the use of modal verbs in grammar section for expressing possibility (e.g., ‘might’, ‘could’, ‘may’) and find examples of their usage in sentences on the Cambridge Dictionary’ executed by model LLaVa-7B PAE.



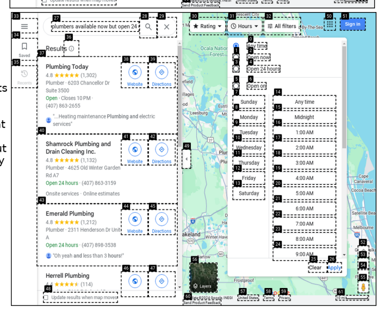
Thought: To search for plumbers available now but not open 24 hours in Orlando, FL on Google Maps, I need to enter the relevant keywords in the search box. Action: Type [0]; plumbers available now but not open 24 hours orlando, fl



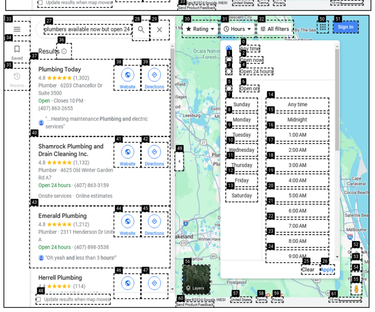
Thought: The search results show some plumbing companies that are open 24 hours, which is not what we are looking for. To find plumbers available now but not open 24 hours, we may need to further refine the search keywords or filters. Action: Type [0]; plumbers available now but open 24 hours orlando, fl



Thought: The search results still show some plumbing companies that are open 24 hours, which is not what we are looking for. To find plumbers available now but not open 24 hours, we may need to further refine the search keywords or try a different approach like looking at the business hours listed. Action: Click [4]



Thought: By unchecking the "Open 24 hours" filter, we are now seeing results for plumbing companies that are open during certain hours but not necessarily 24 hours a day. This should help us find options available now but not open around the clock. Action: Type [0]; plumbers available now but not open 24 hours orlando, fl



Thought: The search results are now showing plumbing companies that are available now but not open 24 hours a day in the Orlando area. The results are filtered to show opening hours from Monday to Saturday. Action: ANSWER; The search results show plumbing companies that are available now but not open 24 hours a day in the Orlando, FL area. The results are filtered to show their opening hours from Monday to Saturday, indicating when they are open and closed each day. For example, one result opens at 8 AM and closes at 4 PM from Monday to Friday, then reopens at 2...

Figure 24: Extra full trajectories of successful trajectory 4 with task 'Search for plumbers available now but not open 24 hours in Orlando, FL' executed by model LLaVa-7B PAE.

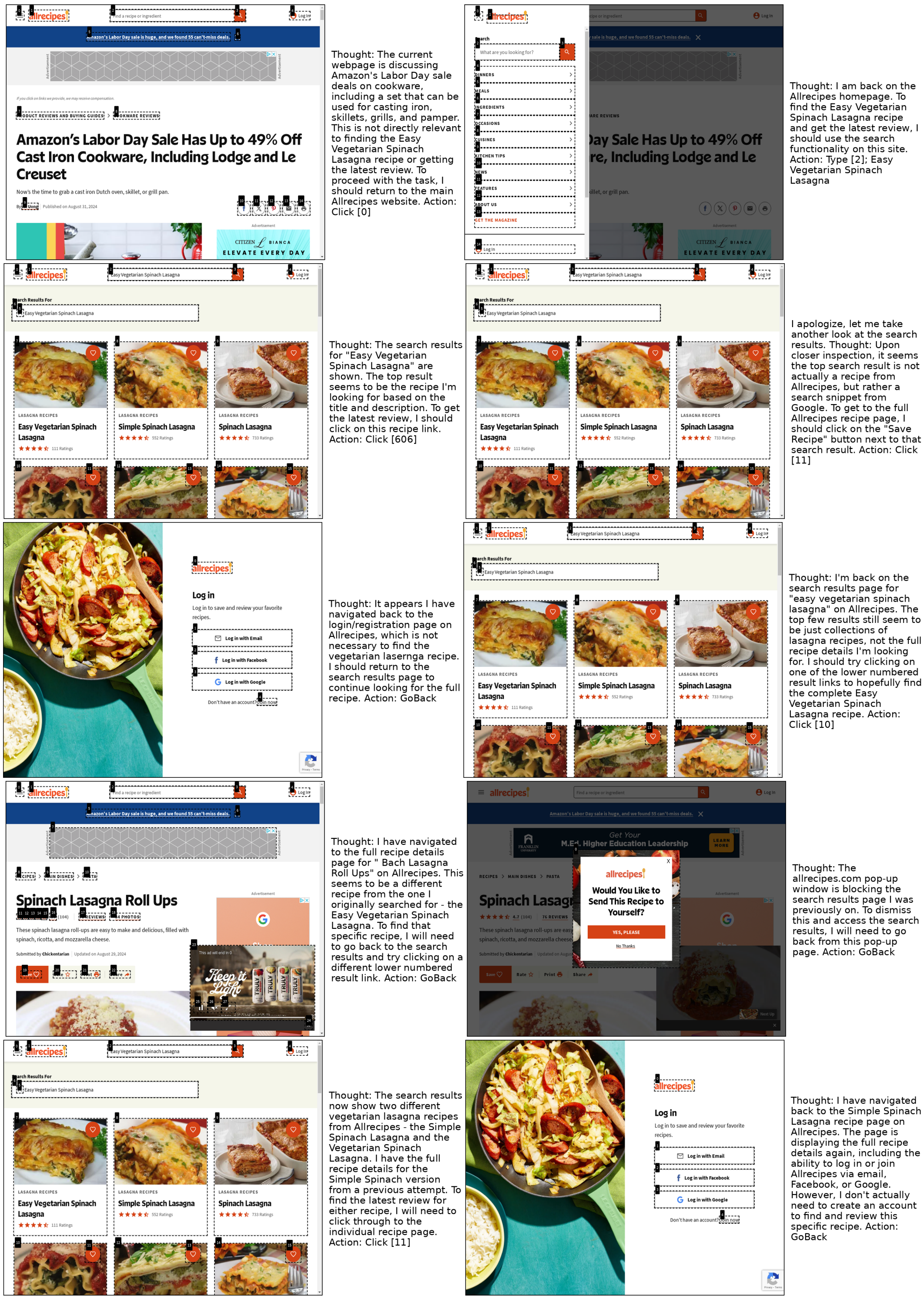


Figure 25: Extra full trajectories of fail trajectory 1, with error type **Low-level Operational error**, executed by model LLaVa-7B SFT. The task is 'Find the Easy Vegetarian Spinach Lasagna recipe on Allrecipes and tell me what the latest review says'.

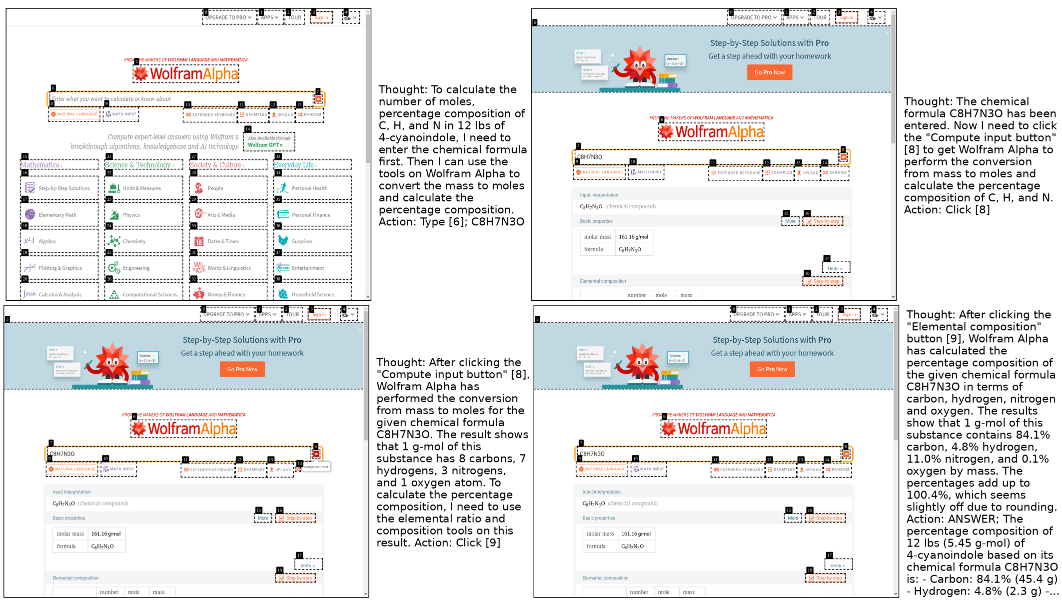
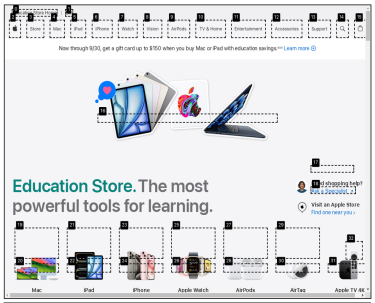
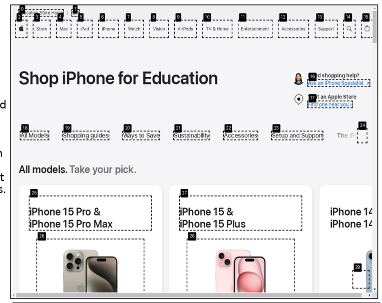


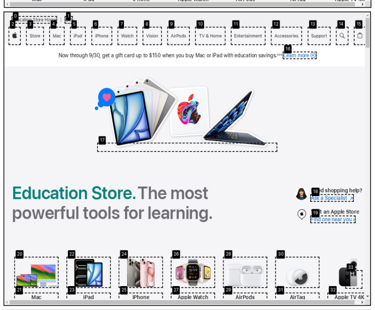
Figure 26: Extra full trajectories of fail trajectory 2, with error type **Planning or Reasoning error**, executed by model LLaVa-7B PAE. The task is 'Give 12 lbs of 4-cyanoindole, converted to molar and indicate the percentage of C, H, N'.



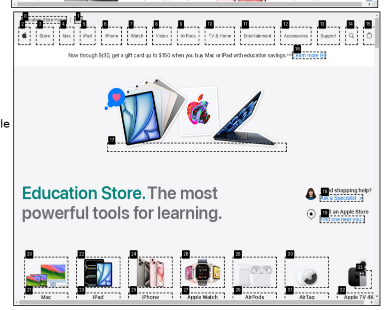
Thought: The task is to find the trade-in value for an iPhone 13 Pro Max on the Apple website. Based on the screenshot, there is an iPhone section that likely contains information about trade-in values for iPhones. Action: Click [24]



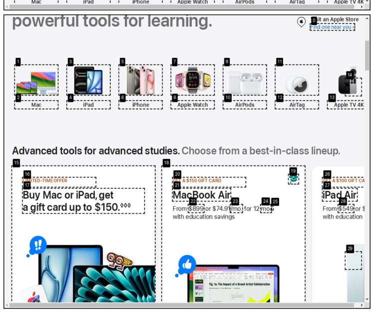
Thought: The previous iPhone section has been filtered to only show iPad models. To find the trade-in value for an iPhone 13 Pro Max, I need to navigate back to the main iPhone section. Action: GoBack



Thought: The current screenshot shows the iPhone section on the Apple Store website. There is an iPhone 13 Pro Max model listed under the product filters. Clicking on that model should allow me to check the trade-in value details for it. Action: Click [26161]



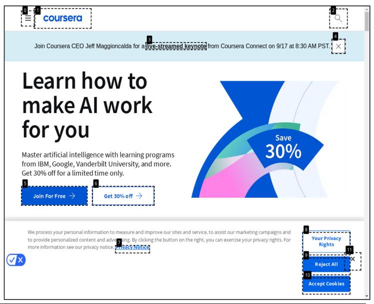
You're correct, my previous action was incorrect. Let me re-evaluate the situation. Thought: After filtering for the iPhone 13 Pro Max model, I should now see the specific product details on this page, including any trade-in offer information. I don't need to click on any additional links or elements. Action: Scroll [WINDOW]; down



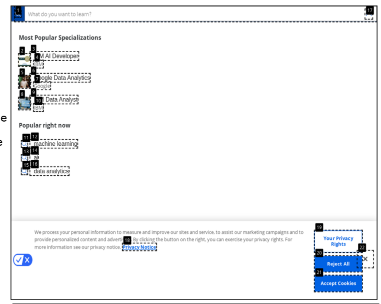
Thought: The page is now showing the detailed product information for the iPhone 13 Pro Max model. I can see the pricing, options to get a gift card with education pricing, and trade-in values are listed under the "Limited-time offer" section. Action: ANSWER; The trade-in values listed for the iPhone 13 Pro Max in good condition on this Apple website are: - Trade in an eligible iPhone X or later for $450 off the $1,099 price. - Or trade in any iPhone model to get $375 off the $1,099 price. With education savings, you can get this top iPhone 13 Pro Max for just $1,049, or $26.12 per month for 24...

Figure 27: Extra full trajectories of fail trajectory 3, with error type **Visual Hallucination**, executed by model LLaVa-7B SFT. The task is 'Find out the trade-in value for an iPhone 13 Pro Max in good condition on the Apple website'.

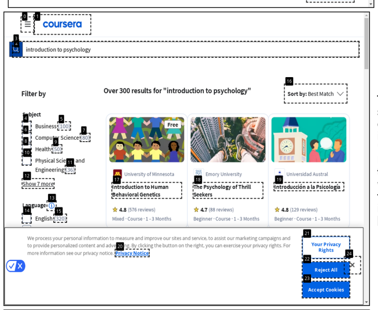
Figure 28: Extra full trajectories of fail trajectory 4, with error type **Timeouts**, executed by model Claude 3.5 Sonnet. The task is 'Go to the Plus section of Cambridge Dictionary, find Image quizzes and do an easy quiz about Animals and tell me your final score'.



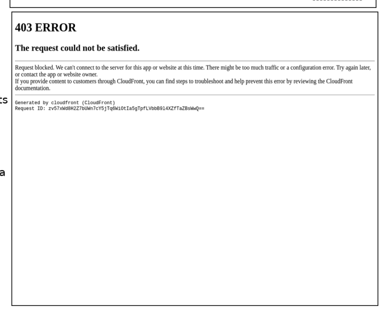
Thought: The current page is the homepage of Coursera. To find a course on introduction to psychology, I need to use the search functionality. Action: Click [2]



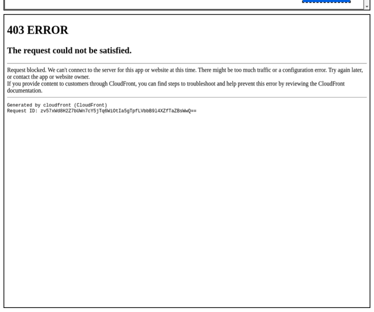
Thought: The search bar is now visible to search for courses related to introduction to psychology. Action: Type [1]; introduction to psychology



Thought: The search results show some relevant courses on psychology, including one titled "Introduction to Human Behavioral Genetics" from the University of Minnesota which seems promising to find the requested information. Action: Click [17]



Thought: The course page provides details about the "Introduction to Human Behavioral Genetics" course, including that it is from the University of Minnesota and requires 3 minutes of effort to complete the troubleshooting steps before reviewing it. However, it does not directly provide the instructor's name or approval time. Action: Click [11]



You're correct, my previous action was incorrect. Let me re-evaluate the information provided. Thought: The "CloudFront" text is likely a part of the website that cannot be interacted with directly. To find the instructor's name and approval time, I should look for more relevant details on this page. Action: ANSWER; The instructor's name for the "Introduction to Human Behavioral Genetics" course is not provided, along with the approval time. The course details indicate it requires 3 minutes of effort to complete the troubleshooting steps before reviewing it.

Figure 29: Extra full trajectories of fail trajectory 5, with error type **Technical issues**, executed by model LLaVa-7B PAE. The task is ‘Identify a course on Coursera that provides an introduction to Psychology, list the instructor’s name, the institution offering it, and how many hours it will approximately take to complete’.