OpenWebVoyager: Building Multimodal Web Agents via Iterative Real-World Exploration, Feedback and Optimization

Hongliang He[1,3*], **Wenlin Yao**[2†], **Kaixin Ma**[2], **Wenhao Yu**[2], **Hongming Zhang**[2], **Tianqing Fang**[2], **Zhenzhong Lan**[3], **Dong Yu**[2]
[1]Zhejiang University, [2]Tencent AI Lab (Seattle), [3]Westlake University

Abstract

The rapid development of large language and multimodal models has sparked significant interest in using proprietary models, such as GPT-4o, to develop autonomous agents capable of handling real-world scenarios like web navigation. Although recent open-source efforts have tried to equip agents with the ability to explore environments and continuously improve over time, they are building text-only agents in synthetic environments where the reward signals are clearly defined. Such agents struggle to generalize to realistic settings that require multimodal perception abilities and lack ground-truth signals. In this paper, we introduce an open-source framework designed to facilitate the development of multimodal web agent that can autonomously conduct real-world exploration and improve itself. We first train the base model with imitation learning to gain the basic abilities. We then let the agent explore the open web and collect feedback on its trajectories. After that, it further improves its policy by learning from well-performing trajectories judged by another general-purpose model. This exploration-feedback-optimization cycle can continue for several iterations. Experimental results show that our web agent successfully improves itself after each iteration, demonstrating strong performance across multiple test sets.[1]

1 Introduction

Developing autonomous agents that can complete complex tasks such as web navigation has been a significant challenge for the AI community (Zhou et al., 2023; Gur et al., 2023; Deng et al., 2024; Koh et al., 2024). Recent advancements of large language and multimodal models such as Claude (Anthropic, 2024) and GPT-4o (OpenAI, 2024) have made it possible to build such agents via prompt engineering (He et al., 2024; Zheng et al., 2024b; Ma et al., 2023). However, these agents struggle to improve further due to their reliance on closed-source models. Another line of work has explored alternative ways to build agents by starting off with weaker open-source models and gradually improving model performance by iteratively exploring the environment, collecting feedback signals, and updating the policy model (Xi et al., 2024; Putta et al., 2024; Patel et al., 2024). However, existing studies have only focused on building text-only agents in synthetic environments (Song et al., 2024; Murty et al., 2024). The synthetic environments provide the benefit of well-defined reward signals, allowing the agents to effectively differentiate the quality of the trajectories and learn accordingly. However, synthetic environments fail to capture the complexity of **real-world** scenarios, leading to potential generalization issues when applied to real-world tasks. Moreover, real-world environments often lack built-in reward signals, while web elements are becoming increasingly diverse, trajectory sampling more time-consuming and prone to obsolescence, all of which pose other challenges in agent's learning and improvement process (He et al., 2024; Pan et al., 2024). Additionally, real-world webpages are designed based on human visual preference, ignoring the visual inputs can cause significant information loss that impacts the agent's performance.

To address above limitations and explore open-source models in real-world settings, we propose OpenWebVoyager, an open-source framework for building multimodal web agents via iterative real-world exploration, feedback and optimization. We show that OpenWebVoyager can learn to perform real-world web navigation tasks through an initial *imitation learning* (IL) phase followed by multiple exploration-feedback-optimization cycles. To do so, we start by compiling a diverse set of web task queries and collecting corresponding agent trajecto-

*Work done during the internship at Tencent AI Lab.
†Work done while at Tencent AI Lab.
[1]Code and data will be released at https://github.com/MinorJerry/OpenWebVoyager. Contact: hehongliang@westlake.edu.cn

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27545–27564
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

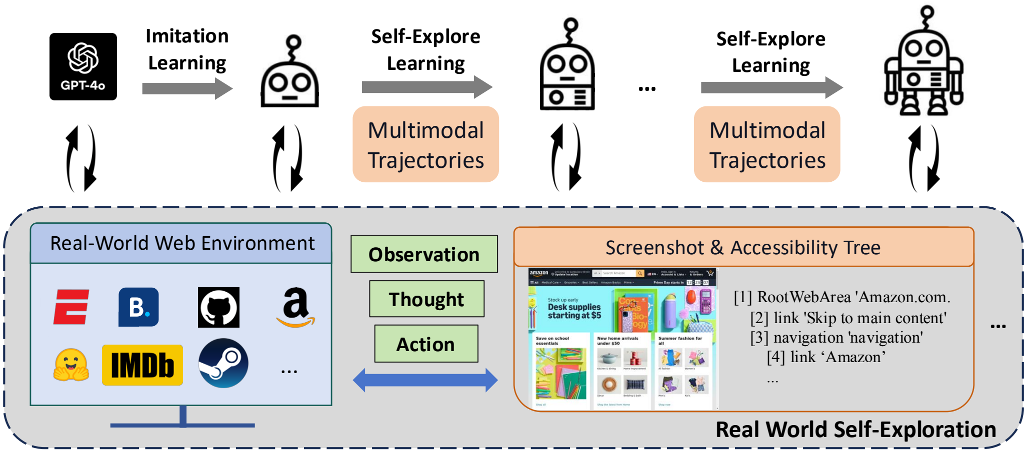


Figure 1: The overall process of OpenWebVoyager, including the Imitation Learning phase and the exploration-feedback-optimization cycles. The agent learns basic multimodal web navigation skills through Imitation Learning and continues to explore real-world web environments. GPT-4o provides feedback on explored multimodal trajectories, leaving successful trajectories for the agent to improve.

ries using a state-of-the-art multimodal agent WebVoyager (He et al., 2024) based on GPT-4o, which we refer to as **WebVoyager-4o**. During the imitation learning phase, we train OpenWebVoyager on trajectories where WebVoyager-4o successfully completes the task to teach the agent basic skills to perform web navigation. Subsequently, within the exploration-feedback-optimization cycle, we continue to synthesize new web tasks, allowing our agent to explore and gather more trajectories. During this stage, we follow He et al. (2024) and leverage GPT-4o to automatically evaluate the correctness of the trajectories produced by OpenWebVoyager. After gathering feedbacks, we retain successful trajectories and merge them with the data from IL phase to conduct the next round of training to improve OpenWebVoyager. The improved agent is then used to sample new trajectories in the next iteration. This streamlined and effective design frees us from the limitations and obsolescence of manually collected trajectories, relying more on GPT-4o's supervision, thus bringing the feasibility of continuous optimization.

In our experiments, we employ idefics2-8b-instruct (Laurençon et al., 2024) as our backbone model and select 48 common websites from the WebVoyager and Mind2Web datasets (Deng et al., 2024) to gather trajectories. The overall process includes one imitation learning phase and three exploration-feedback-optimization cycles. For each phase, we leverage self-instruct (Wang et al., 2022) to generate new web queries. We assess the agent's performance using the Task Success Rate on the WebVoyager and Mind2Web test sets. Results indicate a gradual increase in task success rate across the four phases on the WebVoyager test set from 19.9% to 25.8% and on the Mind2Web cross task set from 6.3% to 19.6%, demonstrating the potential for iterative optimization in multimodal web agents. Additionally, a slight improvement is observed on the Mind2Web cross-web (unseen web) set from 6.6% to 10.4%, suggesting that the exploration-feedback-optimization cycle can, to some extent, generalize to unseen websites.

2 Related Work

2.1 Multimodal Web Agents

Recently, there has been a growing interest in building multimodal web agents, particularly those that combine visual and textual understanding capabilities. Unlike traditional HTML-dependent LLM-based agents (Lutz et al., 2024; Zhou et al., 2023; Gur et al., 2023; Nakano et al., 2021; Ma et al., 2023), Large Multimodal Model (LMM)-based agents can perform a wider variety of web tasks and adapt to more complex web environments. The main difference lies in the observation space. To acquire multimodal input signals, SeeAct (Zheng et al., 2024a) focuses on annotating images of web pages using bounding boxes and index labels of candidate web elements. WebVoyager (He et al., 2024) and VisualWebArena (Koh et al., 2024) both use a JavaScript tool to extract web elements and annotate them on screenshots in a Set-of-Mark (Yang et al., 2023) format. DUAL-VCR (Kil et al., 2024) contextualizes each web element with its neighbors in the screenshot. SCAFFOLD (Lei et al., 2024) introduces dot matrices and coordinates on images to enhance visual grounding. Most of the aforementioned multimodal web agents rely on prompting closed-source multimodal models such as GPT-4V (OpenAI, 2023), Claude (Anthropic, 2024), and Gemini (Team et al., 2023). These models' strong visual grounding and understanding capabilities enable them to correctly interpret webpage screenshots and engage in proper planning using paradigms like ReAct (Yao et al., 2022) or Chain-of-Thought (Wei et al., 2022). While some previous works attempted to leverage open-source vision-language models to build web agents (Zheng et al., 2024a; Koh et al., 2024), they found that models such as BLIP-2-T5 (Jian et al., 2024), LLaVA (Liu et al., 2024), and Idefics (Laurençon et al., 2023) can hardly achieve satisfactory performance. The main reason is that the pretraining of those open-source vision-language models mostly focuses on aligning image-text features and visual question answering instead of image-text interleaved agent trajectories. In this work, we propose an agent built upon an open-source model that can automatically collect trajectories to continuously improve itself, leading to salient gains in performance. And unlike prior methods that focus on specialized agent frameworks (Lutz et al., 2024; Abuelsaad et al., 2024), our approach is model-agnostic and adaptable to diverse frameworks.

2.2 Self-Improving Web Agents

Researchers also have attempted to boost agents and adapt them to complex environments through exploration and self-improvement (Yu et al., 2024; Zhang et al., 2024b). AgentGYM (Xi et al., 2024) proposes a framework that unifies a wide range of environments for real-time exploration and evolution of LLM-based agents. AgentQ (Putta et al., 2024) integrates Monte Carlo Tree Search (MCTS) and Direct Preference Optimization (DPO; Rafailov et al., 2024) algorithms to iteratively update the policy of LLM-based web agents based on successful and failed web trajectories. Patel et al. (2024) suggests improvement by utilizing web agents to collect and filter in-domain trajectories, plus out-of-domain tasks along with hypothetical solution trajectories. However, there is still a lack of exploration on how to leverage multimodal web signals to achieve self-improvement. We aim to enable multimodal web agents to adapt to complex and dynamic online environments, enhancing their generality and ability to operate across numerous online websites.

3 Method

In this section, we introduce OpenWebVoyager, an innovative web agent that outlines a path of iterative optimization for LMM-based Web Agents to handle intricate online web tasks. Firstly, we enable the agent to learn web navigation trajectories of WebVoyager-4o in the first stage to gain basic web knowledge and navigation skills, namely Imitation Learning (IL). Subsequently, the agent iteratively explores and improves with the feedback from GPT-4o.

3.1 Task Formulation

In the web browsing environment $\mathcal{E}$, consider the web navigation process as a Partially Observable Markov Decision Process (POMDP). The setup is defined by the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, R)$, where $\mathcal{S}$ denotes the state space, $\mathcal{O}$ represents the observation space, and $\mathcal{A}$ is the action space. $\mathcal{T}$ is the deterministic transition function that performs web operations in the browser to promote the process. The reward R in this environment is typically a sparse signal indicating success or failure, with values of 1 or 0, respectively.

Given a task query q and its corresponding website w, we can initialize the web environment $\mathcal{E}$ by setting the state s_1 to this web page, and obtain the first step observation $o_1 \in \mathcal{O}$. In this work, we adopt the vision-language setting that the observation in each step will include an accessibility tree and a screenshot, i.e., $o_1 = (o_1^a, o_1^s)$. Let θ represents the parameters of the Large Multimodal Models (LMMs). Following the ReAct paradigm, we derive thoughts and actions using LMMs: $(h_1, a_1) \sim \pi_\theta(\cdot|I, q, o_1) = \pi_\theta(\cdot|I, q, o_1^a, o_1^s)$, where I denotes the system prompt, including answer formats, the introduction of web operations and some guidelines. The transition function $\mathcal{T}$ is then applied to parse the action and execute it on the web page, obtaining the next state s_2. Therefore, at time step t, we have:

$$(h_t, a_t) \sim \pi_\theta(\cdot|I, q, o_1^a, o_1^s, h_1, a_1, ..., o_t^a, o_t^s) \quad (1)$$

$$s_{t+1} = \mathcal{T}(s_t, a_t; \mathcal{E}). \quad (2)$$

The full trajectory can be represented as $\tau = (o_1^a, o_1^s, h_1, a_1, ..., o_T^a, o_T^s, h_T, a_T)$, where T is the number of iterations in web navigation, i.e., the length of the trajectory.

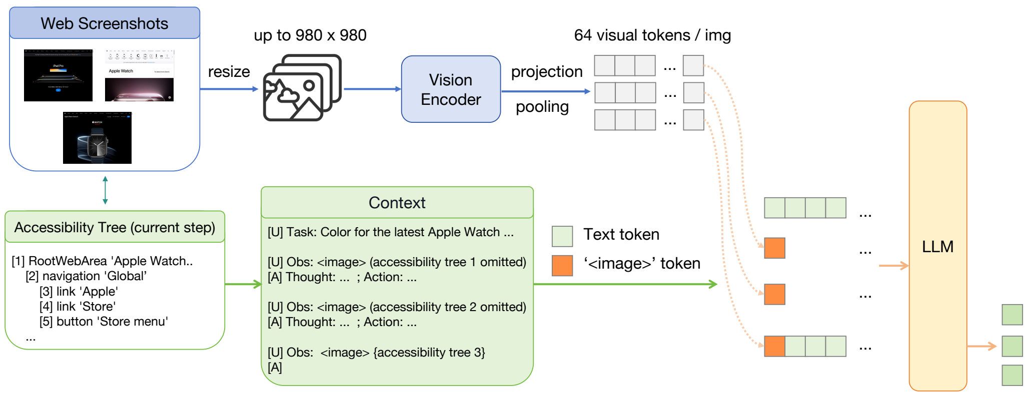


Figure 2: The model architecture of our multimodal web agent. We use the most recent 3 web screenshots to demonstrate the page changes after performing web actions and label the web elements in the accessibility tree to facilitate the agent in selection and response. Considering the limitation of sequence length and to avoid confusion, we only retain the most recent accessibility tree.

3.2 OpenWebVoyager Overview

Environment We adopt the Selenium-based online web navigation environment provided by WebVoyager (He et al., 2024). In contrast to WebVoyager, we do not employ the Set-of-Mark approach to mark elements on screenshots because open-source LMMs face significant visual grounding issues in identifying numerical tags on screenshots. We modify the observation of the web page to include the accessibility tree and its corresponding unmarked screenshot. Figure 4 in Appendix A shows a specific example of the observation space.

Model and Learning We adopt Idefics2 (Laurençon et al., 2024) as the backbone LMM for building OpenWebVoyager. Idefics2 is well-suited for our task as it incorporates interleaved image-text documents during training, boosting the model's multi-image reasoning and long-context comprehension capabilities. Additionally, Idefics2 supports encoding high-resolution images up to 980x980 pixels, which is necessary for preserving the fine-grained visual details on the webpage screenshots. In Figure 2, we elaborate on how we adapt the Idefics2 architecture to build OpenWebVoyager. Similar to the messages fed into GPT-4o, we embed the <image> token at the corresponding position in the context, aligning it with the accessibility tree. The Idefics2-based agent will make a decision based on the observation containing multimodal information. Figure 1 illustrates the full process of IL and exploration-feedback-optimization cycle: collecting trajectories for Imitation Learning via WebVoyager-4o, training the base agent, and then continuously exploring new trajectories. Based on feedback from GPT-4o, successful trajectories are leveraged for optimization.

3.3 Web Task Queries Collection

Queries for the Imitation Learning The IL phase is crucial as it forms the foundation for subsequent improvements. We aim to gather a diverse set of web tasks of varying difficulty, enabling GPT-4o to collect diverse trajectories. We choose 48 popular websites, then select and synthesize queries $\mathcal{Q}_{\text{IL}}$ from multiple perspectives before IL. The details of $\mathcal{Q}_{\text{IL}}$ are shown in Appendix D.

Queries for Real-World Exploration We continue to use the self-instruct (Wang et al., 2022) approach to generate new queries that are similar but not duplicated based on existing queries. In each exploration-feedback-optimization cycle, we automatically generate 480 queries for 48 websites, with 10 queries for each website. The agent then conducts web exploration based on these tasks.

3.4 Imitation Learning

Trajectories Collection We utilize GPT-4o along with the WebVoyager paradigm (He et al., 2024) to generate web navigation trajectories corresponding to the above queries. The agent is named WebVoyager-4o and configured to receive observations consisting of the latest k steps, including accessibility trees and screenshots. i.e., for each $q_i \in \mathcal{Q}_{\text{IL}}$, $\tau_i \sim \pi_{\theta_g}(\tau|I, q_i)$, we clip long context c_t to avoid performance degeneration when $t > k$:

$$c_t^{\text{clip}} = (h_1, a_1, h_2, a_2, ..., h_{t-k}, a_{t-k}, o_{t-k+1}, h_{t-k+1}, a_{t-k+1}, ..., o_t), \quad (3)$$

$$(h_t, a_t) \sim \pi_{\theta_g}(\cdot|I, q, c_t^{\text{clip}}). \quad (4)$$

It is worth noting that we preserve the thought and action of each step to maintain the full reasoning process without occupying excessive context. The collected trajectories fall into three pre-defined categories: **unfinished** (exceeding the maximum iteration of Navigation), **finished & unsuccessful**, and **finished & successful**. In this stage, to better distill knowledge from GPT-4o, we filter out unfinished trajectories, retaining only the other ones for training in Imitation Learning. Meanwhile, we resample the unfinished tasks once to improve the utilization of queries and reduce the problem of navigation failure due to sampling randomness. Throughout the process, all trajectories are **collected through online interactions**, with no offline data being utilized.

Learning We adopt Idefics2 (Laurençon et al., 2024) to learn trajectories collected through WebVoyager-4o. In Idefics2, screenshots are encoded as 64 visual tokens. However, the length of each accessibility tree is much longer than 64 tokens. Considering the sequence length issue, we further truncate the context and the number of images, retaining the latest k images while keeping only one accessibility tree of the current page. That is, we remove $k-1$ accessibility trees in Eq. 3:

$$\begin{aligned} c_t^{\text{clip}'} = (&h_1, a_1, ..., h_{t-k}, a_{t-k}, \\ &o^s_{t-k+1}, h_{t-k+1}, a_{t-k+1}, ..., o^s_t, o^a_t). \end{aligned} \quad (5)$$

Let D_{IL} represents the collected trajectories, and θ denote the parameters of the Idefics2 model. We aim to maximize the following objective function:

$$\begin{aligned} \mathcal{J}_{\text{IL}}(\theta) = \mathbb{E}_{(q,\tau)\sim D_{\text{IL}}} \sum_{t=1}^{T} \Big[&\log \pi_\theta(a_t|q, c_t^{\text{clip}'}, h_t) \\ &+ \log \pi_\theta(h_t|q, c_t^{\text{clip}'}) \Big], \end{aligned} \quad (6)$$

where the system prompt I is no longer provided because of its considerable length. Through Imitation Learning, the agent has already learned the basic operation logic and response format, so there is no need for the system prompt.

3.5 Iterative Optimization

After the Imitation Learning phase, the trained agent π_{θ_b} will proceed to explore websites and undergo multiple cycles of exploration-feedback-optimization. We continue to generate more queries using self-instruct. Instead of relying on WebVoyager-4o to collect trajectories, the agent collects trajectories on its own. At each exploration-feedback-optimization cycle, we employ trajectory-level rejection sampling via GPT-4o. An LMM capable of evaluating multimodal web trajectories is indispensable to ensure the quality of the trajectories and automate the process. **GPT-4o is a natural choice for this task**, as its capabilities are comparable to those of humans, and its stability is even slightly higher than that of humans. We discuss more reasons for choosing GPT-4o to provide feedback in Appendix C. Let Q^j_{SI} be the query set for j-th optimization, for every $q \in Q^j_{\text{SI}}$, we sample several trajectories from the model $\pi_{\theta_{j-1}}$, with GPT-4o acting as the Auto Evaluator, accepting only trajectories that GPT-4o deems as successfully navigated. We consider this auto evaluation method reliable because assessing the correctness of a trajectory is much easier than obtaining a correct trajectory. He et al. (2024) also demonstrates a high level of evaluation consistency between GPT-4o and humans.

Let D^j_{SI} represent the set of trajectories collected after rejection sampling in the j-th optimization. We mix the collected trajectory sets with D_{IL} and continue fine-tuning $\pi_{\theta_{j-1}}$ by maximizing the following objective:

$$\begin{aligned} \mathcal{J}^j_{\text{SI}}(\theta) = \mathbb{E}_{(q,\tau)\sim D_{\text{SI}}} \sum_{t=1}^{T} \Big[&\log \pi_\theta(\\ a_t|q, c_t^{\text{clip}'}, h_t) &+ \log \pi_\theta(h_t|q, c_t^{\text{clip}'}) \Big], \end{aligned} \quad (7)$$

where $j = 1, ..., m$ denotes the times of optimization, $D_{\text{SI}} = D_{\text{IL}} \cup D^j_{ev}$ denotes the mixed trajectory set and π_{θ_0} is set to π_{θ_b}. The complete procedure is shown in Algorithm 1 in Appendix B.

4 Experiment

4.1 Dataset and Metric

Training Dataset In §3.4, we have outlined the composition of the query set Q_{IL} during the Imitation Learning stage, which includes 48 websites mentioned in Mind2Web (Deng et al., 2024) and WebVoyager (He et al., 2024), along with 1516 relevant task queries collected. Notably, in the real-world and online setting, some websites in Mind2Web have become **inaccessible** ("shut down" or blocked by CAPTCHAs). As a result, we have to exclude these websites and retained

	Allrecipes	Amazon	Apple	ArXiv	GitHub	Booking	ESPN	Coursera
$\text{OpenWebVoyager}_{\text{IL}}$	17.8%	12.2%	20.9%	14.0%	14.6%	9.1%	9.1%	31.0%
$\text{OpenWebVoyager}_{\text{iter-1}}$	35.2%	26.8%	11.6%	18.6%	24.4%	6.8%	2.3%	28.6%
$\text{OpenWebVoyager}_{\text{iter-2}}$	22.2%	36.6%	27.9%	20.9%	19.5%	6.8%	6.8%	33.3%
$\text{OpenWebVoyager}_{\text{iter-3}}$	24.4%	24.4%	20.9%	18.6%	31.7%	18.2%	11.4%	42.9%
$\text{OpenWebVoyager}_{\text{iter-3-dgs}}$	20.0%	31.7%	18.6%	23.3%	24.4%	13.6%	25.0%	42.9%
$\text{OpenWebVoyager}_{\text{iter-3-dgs-g}}$	22.2%	29.3%	32.6%	20.9%	26.8%	11.4%	11.4%	42.9%

	Cambridge Dictionary	BBC News	Google Flights	Google Map	Google Search	Huggingface	Wolfram Alpha	Overall
$\text{OpenWebVoyager}_{\text{IL}}$	37.2%	9.5%	9.5%	22.0%	44.2%	20.9%	26.1%	19.9%
$\text{OpenWebVoyager}_{\text{iter-1}}$	25.6%	9.5%	19.0%	26.8%	44.2%	25.6%	32.6%	22.6%
$\text{OpenWebVoyager}_{\text{iter-2}}$	23.3%	14.3%	19.0%	22.0%	41.9%	11.6%	34.8%	22.7%
$\text{OpenWebVoyager}_{\text{iter-3}}$	37.2%	11.9%	11.9%	26.8%	39.5%	30.2%	37.0%	25.8%
$\text{OpenWebVoyager}_{\text{iter-3-dgs}}$	30.2%	11.9%	21.4%	22.0%	39.5%	23.3%	34.8%	25.5%
$\text{OpenWebVoyager}_{\text{iter-3-dgs-g}}$	34.9%	14.3%	21.4%	29.3%	44.2%	32.6%	37.0%	27.4%

Table 1: Task success rate on WebVoyager test set (643 queries). All websites are seen during training. 'IL', 'iter-1', 'iter-2', and 'iter-3' represent agents after IL, 1st, 2nd, and 3rd optimization, respectively. 'dgs' and 'dgs-g' denote difficulty-guided sampling, i.e., sample more trajectories for webs with low sampling accuracy, the former by adding trajectories sampled by the agent itself and the latter by adding trajectories sampled by GPT-4o.

Agents	Mind2Web cross-task (unseen task)				Mind2Web cross-web (unseen web)			
	Entertainment	Shopping	Travel	Overall	Entertainment	Shopping	Travel	Overall
$\text{OpenWebVoyager}_{\text{IL}}$	8.2%	5.9%	4.3%	6.3%	3.0%	13.3%	4.7%	6.6%
$\text{OpenWebVoyager}_{\text{iter-1}}$	12.2%	0.0%	4.3%	7.1%	6.1%	6.7%	9.3%	7.5%
$\text{OpenWebVoyager}_{\text{iter-2}}$	24.5%	5.9%	6.5%	14.3%	15.2%	10.0%	7.0%	10.4%
$\text{OpenWebVoyager}_{\text{iter-3}}$	26.5%	23.5%	10.9%	19.6%	6.1%	20%	7.0%	10.4%
$\text{OpenWebVoyager}_{\text{iter-3-dgs}}$	18.4%	23.5%	10.9%	16.1%	9.1%	16.7%	25.6%	17.9%
$\text{OpenWebVoyager}_{\text{iter-3-dgs-g}}$	22.4%	29.4%	15.2%	20.5%	3.0%	20.0%	23.3%	16.0%

Table 2: Task success rate on Mind2Web cross-task and cross-web test set. In cross-task set, the queries from the same websites are seen during training. In cross-website set, the websites are not seen during training but still belong to the Entertainment, Shopping, and Travel Domain.

as many tasks as possible. We use WebVoyager-4o to gather corresponding trajectories for these queries, with each query having a maximum of 2 trajectories. Then we retain 1165 **finished** (including both successful and unsuccessful) trajectories, with a total of 7253 interaction turns. During the j-th exploration-feedback-optimization cycle, we expend 480 queries for 48 selected websites. The trajectories are sampled via $\pi_{\theta_{j-1}}$ and the maximum resampling count is set to 5.

Evaluation Dataset To evaluate the performance of our agent, we use the following datasets: 1) WebVoyager (He et al., 2024) test set, comprising 15 websites seen during training and 643 queries; 2) Mind2Web (Deng et al., 2024) cross-task test set, which includes 33 websites seen during training and a total of 112 queries. 3) Mind2Web cross-website test set, we select 2 websites each from "Entertainment", "Shopping", and "Travel" categories; the websites are **unseen during training**, amounting to a total of 106 queries.

Metric Following WebVoyager, we adopt Task Success Rate automatically evaluated by GPT-4o as the primary metric. To view the exploration efficiency in the exploration-feedback-optimization cycle, we define Success@K (S@K) as the ratio of tasks that get success within K samples. Additionally, we pay attention to the finish rate (F@1), where a task is considered finished as long as the agent selects 'ANSWER' within the maximum navigation steps. Table 3 shows the details of the query set and collected trajectories in exploration-feedback-optimization cycles.

4.2 Experimental Details

To collect data for imitation learning phase, we adopt the state-of-the-art model GPT-4o with WebVoyager framework (WebVoyager-4o) to sample

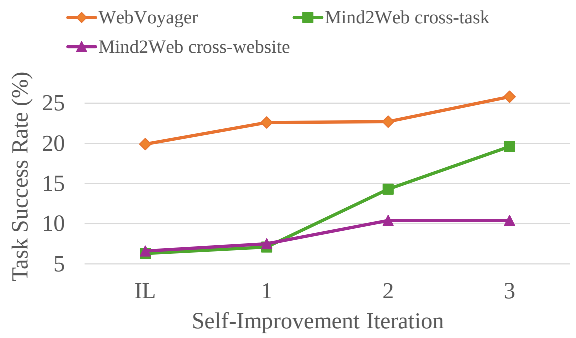


Figure 3: Performance growth of OpenWebVoyager on WebVoyager and Mind2Web test set from Imitation Learning phase to 3rd exploration-feedback-optimization cycle.

web navigation trajectories. We set $k = 3$, i.e., the context contains at most 3 screenshots and corresponding accessibility trees but retains the thoughts and actions generated by GPT-4o in each step. Our agent builds upon Idefics2-8b-instruct with outstanding vision-language capabilities to complete the imitation learning and exploration-feedback-optimization cycles. During fine-tuning, the max sequence length is set to 8192. We no longer use system prompts and further clip the context to accept a maximum of 3 screenshots and 1 accessibility tree. The original resolution of the screenshots is 1024*768 and the screenshots are resized such that the longer length is no larger than 980, before feeding into Idefics2. We set the batch size to 64 and train for 300 iterations in each phase, approximately 2 - 3 epochs. In the exploration-feedback-optimization phase, we iteratively train our agent with a total of $m = 3$ iterations. When the agent performs exploration, we set the temperature to 1.2 to improve the randomness. The agent samples up to 5 trajectories for each given task query. We still select GPT-4o as the feedback model and trajectories with positive feedback are gathered for further optimizations.

4.3 Main Results

Throughout the entire process of Imitation Learning and exploration-feedback-optimization cycles, we trained four models: $\text{OpenWebVoyager}_{\text{IL}}$, $\text{OpenWebVoyager}_{\text{iter-1}}$, $\text{OpenWebVoyager}_{\text{iter-2}}$, and $\text{OpenWebVoyager}_{\text{iter-3}}$. Table 1 shows the performance of these models on the WebVoyager test set. Table 2 presents the results of these models on the Mind2Web cross-task and cross-website test set. We show the performance changes of our agent on these datasets from imitation learning phase to the third optimization cycle in Figure 3.

From the results in Table 1 and Table 2, we observe a general improvement in task success rates in both WebVoyager and Mind2Web cross-task test set as optimization progressed. This indicates the effectiveness of our method when the webs in the test set have been trained on or explored during the training phase. In the Mind2Web cross-web test set, the optimization cycle also provides some enhancement in agent's performance, although not as prominently as in the cross-task set. Also, the improvement is unstable on these unexplored websites, agent suffers from sampling randomness and is more likely to get stuck during web navigation.

Table 3 shows the results of GPT-4o's feedback on the trajectories sampled by the agent during the exploration phase. We find that despite having 5 chances for resampling, The agent still performs poorly on many websites. Therefore, we consider increasing the number of trajectories specifically for these "difficult" websites during exploration-feedback-optimization phase. To investigate the effectiveness of this **difficulty-guided sampling (DGS)** strategy, we train $\text{OpenWebVoyager}_{\text{iter-3-dgs-g}}$ and $\text{OpenWebVoyager}_{\text{iter-3-dgs}}$. The former involves adding some trajectories sampled by WebVoyager-4o for webs with S@5 below 40% during the third iteration, while the latter adds some trajectories sampled by the agent itself. Compared to $\text{OpenWebVoyager}_{\text{iter-3}}$, adding exploration trajectories to the "difficult" websites can improve performance for certain websites like Google Flights. However, influenced by the sampling randomness, the optimization is not stable, as seen in Booking, GitHub, and others. We believe the potential reason why DGS failed to consistently enhance performance lies in the fact that, although DGS increases trajectory sampling for these websites, only a small fraction of simple tasks succeeded, **which does not provide more diversity**. So, incorporating WebVoyager-4o's sampling of more diverse trajectories during exploration has led to some overall performance improvements.

4.4 Discussion

The average length of trajectories. During inference, we record the length of trajectories when they are finished (the agent provides answers) and successful. The variation of the average length of web navigation trajectories is shown in Table 4.

In our experiments, we observe that as iterative

Improvement Iteration	Query	Traj. From	Success Traj.	Turns	F@1	S@1	S@2	S@3	S@4	S@5
iter-1	480	π_{θ_b}	152	943	74.6%	10.4%	19.6%	24.4%	27.5%	31.7%
iter-2	480	π_{θ_1}	205	1324	87.1%	16.0%	24.0%	30.2%	36.9%	42.7%
iter-3	480	π_{θ_2}	207	1333	91.5%	18.8%	27.9%	35.2%	41.0%	43.1%

Table 3: Details of query set and trajectory set during the exploration-feedback-optimization cycle. The feedback on task success or not is provided by GPT-4o. F@1 indicates the finish rate of the first exploration. S@K represents the task success rate within K explorations. Each task will sample the trajectory up to 5 times until it succeeds or fails all 5 times, successful trajectories will be retained to improve our agent.

Agent	WebVoyager		Mind2Web cross-task		Mind2Web cross-website	
	Finish	Success	Finish	Success	Finish	Success
OpenWebVoyager$_{\text{IL}}$	6.47	5.26	8.77	7.00	9.28	9.29
OpenWebVoyager$_{\text{iter-1}}$	6.17	5.02	7.58	5.00	7.98	9.63
OpenWebVoyager$_{\text{iter-2}}$	5.89	5.04	7.33	6.31	7.13	7.45
OpenWebVoyager$_{\text{iter-3}}$	5.47	5.07	7.67	7.59	6.16	6.91

Table 4: The average length of trajectories across different optimization cycles on various test sets. 'Finish' and 'Success' indicates that we calculate the average length for finished or successful trajectories, respectively.

Agent	WebVoyager (643 tasks)				
	R	RS	S	RS / R	RS / S
OpenWebVoyager$_{\text{IL}}$	61	8	128	13.1%	6.3%
OpenWebVoyager$_{\text{iter-1}}$	75	16	145	21.3%	11.0%
OpenWebVoyager$_{\text{iter-2}}$	88	22	146	25.0%	15.1%
OpenWebVoyager$_{\text{iter-3}}$	142	40	166	28.2%	24.1%

Table 5: The frequency of the agent using the restart action: Let R denote the number of trajectories with restart, RS the number of successful trajectories with restart, and S the total number of successful trajectories.

optimization progresses, agents tend to complete tasks in fewer interaction steps and navigate more quickly on familiar websites. This phenomenon creates a cycle where trajectories obtained during the exploration-feedback phase become shorter, leading the model to increase its focus on learning from shorter trajectories during optimization.

Hallucination limits performance. We find that agents often directly hallucinate answers that do not appear during the navigation. The decrease in trajectory length might have increased the frequency of this issue. The agent tends to terminate navigation directly instead of continuing the search after a certain length of the trajectory. As shown in Table 3, we can also observe that the results for F@1 are high, but S@1 are relatively low. This indicates that agent believes it has finished the task but is actually unsuccessful. While the finish rate and success rate in GPT-4o-sampled trajectories are close. This insight suggests that in future exploration, we can increase the diversity of sampling by varying the task difficulty and trajectory length.

Training Trajectories	Result
$D_{\text{IL}} \cup D_{\text{iter-1}} \cup D_{\text{iter-2}}$	20.8%
$D_{\text{IL}} \cup D_{\text{iter-2}}$	23.3%

Table 6: Study on whether to use a mixture of data from previous phases in exploration-feedback-optimization cycle (OpenWebVoyager$_{\text{iter-1}}$ $\rightarrow$ OpenWebVoyager$_{\text{iter-2}}$).

Restart to the search engine and solve tasks. In WebVoyager's paradigm, an important web action is to restart navigation from the search engine when encountering difficulties. In this paper, the 'Restart' action is also provided in the data for training during the Imitation Learning phase. We observe the frequency of our agent using restart action, calculate their success rates, and the ratio of successful tasks using restart to the total successful tasks, as shown in Table 5. We can infer from the results in the WebVoyager test set that as agents undergo iterative optimization, they increasingly prefer to use the search engine. The proportion of successful trajectories achieved by using the search engine is rising among all successful trajectories, addressing some of the navigation failure issues.

Why using GPT-4o for Feedback during Exploration The way GPT-4o is used in the exploration phase differs from its use in the imitation learning phase. During imitation learning, the agent distills GPT-4o's web navigation capabilities. However, in the exploration phase, the agent samples its own trajectories, and GPT-4o only provides a reward signal to ensure trajectory quality. The number of GPT-4o calls is significantly lower than in the imitation learning phase. Therefore, from a cost-efficient perspective, it is undesirable to extend the imitation learning phase for too long. After the agent has learned a certain level of web navigation skills, transitioning to exploration-feedback-optimization becomes a better choice.

In addition, we select GPT-4o for the following reasons: (1) Currently, there is no available open-source reward model capable of providing feedback to LMM-based web agents, especially for trajectories that include multiple consecutive screenshots. (2) For judging the success of multi-modal web navigation trajectories, GPT-4o exhibits high consistency with human judgments (kappa = 0.72). This ensures the accuracy of feedback and the quality of explored trajectories. (3) Automation of the entire process is necessary. Therefore, a tool that can provide feedback for explored trajectories is essential. GPT-4o naturally fits this role, and other models with high agreement with human evaluations could also be utilized.

Other settings and parameters. Trajectory collection is time-consuming, especially in the exploration phase where each query requires up to 5 resampled trajectories to tackle relatively difficult navigation tasks. So we primarily adjust hyper-parameters such as learning rate and global batch size during the IL phase. However, we ultimately find that this has little significance, as the error is much smaller compared to the challenges posed by webpage navigation and sampling randomness.

In optimization cycles, we also try to mix all trajectories that considered success through GPT-4o's feedback, for example, using $D_{\text{IL}} \cup D_{\text{iter-1}} \cup D_{\text{iter-2}}$ to improve OpenWebVoyager$_{\text{iter-1}}$. We select 120 WebVoyager queries and compare task success rate in Table 6.

Other discussions are shown in Appendix C.

5 Conclusion

In this paper, we explore how to construct a multimodal web agent via iterative exploration, feedback and optimization. We adopt idefics2-8b-instruct as the backbone LMM model and collect web task queries from numerous websites. Initially, our agent learns the web operation logic of GPT-4o through Imitation Learning. Then it enters the exploration-feedback-optimization cycles, exploring and collecting trajectories based on new web tasks, retaining the trajectories that GPT-4o considers correct for further learning, updating, and optimization. We focus on building an LMM-based iterative optimization web agent with multi-image understanding capabilities, enabling it to adapt to complex and dynamic online web environments. The entire process primarily involves the agent's self-exploration and GPT-4o's supervision, reducing human intervention and allowing continuous expansion to ensure the agent's generality.

Limitations

First, we only consider the most common executable web actions in the simulated environment, including clicking, typing, and scrolling, without more advanced actions such as dragging and zooming. Additionally, our approach is based on a relatively small LMM Idefics2 with 8B parameters, which may limit the agent's ability to effectively navigate websites of unseen domains and respond to complex user queries. The low performance on complex websites might further affect exploration efficiency, leading to minimal improvement and time-consuming during the exploration-feedback-optimization process. Last, our model still primarily relies on accessibility trees, we hope to improve the visual grounding and multi-image reasoning capabilities so that it can directly use web screenshots for planning like GPT-4o.

Ethics Statement

In light of the potential risks associated with online web navigation, all our experiments adhere strictly to ethical guidelines. Our approach includes human supervision as well as GPT-4's monitoring for content violations. Throughout the sampling of all web task trajectories, no violations by the agent are detected. A small portion of tasks are filtered due to the sensitivity of advertisements or content on news websites. None of the tasks involve private information such as personal names, account passwords, etc. The tasks typically include information-seeking activities and do not include actual bookings or payment transactions. In our work, the web agent's sampled trajectories are intended solely for research purposes. The agent operates in a simulated human-like manner, with a slow sampling frequency, ensuring no pressure is placed on the explored websites.

Acknowledgments

This work is supported by Tencent AI Lab. And Zhenzhong Lan is supported by the Scientific Research Project of Westlake University (Grant No. WU2024B003).

References

Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. 2024. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *Preprint*, arXiv:2407.13032.

AI Anthropic. 2024. Introducing the next generation of claude.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

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

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Yiren Jian, Chongyang Gao, and Soroush Vosoughi. 2024. Bootstrapping vision-language learning with decoupled language pre-training. *Advances in Neural Information Processing Systems*, 36.

Jihyung Kil, Chan Hee Song, Boyuan Zheng, Xiang Deng, Yu Su, and Wei-Lun Chao. 2024. Dual-view visual contextualization for web navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14445–14454.

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

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*.

Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M. Rush, Douwe Kiela, Matthieu Cord, and Victor Sanh. 2023. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *Preprint*, arXiv:2306.16527.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *Preprint*, arXiv:2405.02246.

Xuanyu Lei, Zonghan Yang, Xinrui Chen, Peng Li, and Yang Liu. 2024. Scaffolding coordinates to promote vision-language coordination in large multi-modal models. *arXiv preprint arXiv:2402.12058*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024. Llava-onevision: Easy visual task transfer. *Preprint*, arXiv:2408.03326.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Michael Lutz, Arth Bohra, Manvel Saroyan, Artem Harutyunyan, and Giovanni Campagna. 2024. Wilbur: Adaptive in-context learning for robust and accurate web agents. *arXiv preprint arXiv:2404.05902*.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation. *Preprint*, arXiv:2309.08172.

Shikhar Murty, Christopher Manning, Peter Shaw, Mandar Joshi, and Kenton Lee. 2024. Bagel: Bootstrapping agents by guiding exploration with language. *arXiv preprint arXiv:2403.08140*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. Hello gpt-4o.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. 2024. Large language models can self-improve at web agent tasks. *arXiv preprint arXiv:2405.20309*.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *Preprint*, arXiv:2403.02502.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *Preprint*, arXiv:2101.10382.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning. *arXiv preprint arXiv:2410.02052*.

Hongming Zhang, Xiaoman Pan, Hongwei Wang, Kaixin Ma, Wenhao Yu, and Dong Yu. 2024a. Cognitive kernel: An open-source agent system towards generalist autopilots.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2024b. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *arXiv preprint arXiv:2408.15978*.

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

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Environment and Prompts

We adopt the framework of WebVoyager for online real-world web navigation. The web actions used are the most basic clicks, inputs, and scroll operations as shown in Table 7. Unlike WebVoyager, we do not use the Set-of-Mark approach to label screenshots. Instead, we combine screenshots and the accessibility tree as observations for the agent to make decisions. Figure 4 illustrates an example of observation.

Based on the changes in observations, we slightly modify the system prompt of WebVoyager (He et al., 2024) during the Imitation Learning phase to accommodate the paradigm of accessibility tree + screenshot. In terms of web operation implementation, each element in the accessibility tree has pre-saved attribute information, where 'union_bound' labels the position information of the element.We use Selenium to locate the element that appears in this position and then access it.

In the WebVoyager framework, in addition to the system prompt, the author has designed error reflection to ensure effectiveness. When a certain action fails, there will be a prompt saying: "`The action you have chosen cannot be executed. Please double-check if you have selected the correct element or used the correct action format. Then provide the revised Thought and Action.`" This prompt serves to remind the agent to correct errors. While training our own Agent, although we no longer use the system prompt, we still retain the error reflection mechanism.

B Algorithm

In Algorithm 1, we present the complete algorithm of OpenWebVoyager. It mainly consists of an Imitation Learning (IL) phase and multiple exploration-feedback-optimization cycles. In the IL phase, GPT-4o (π_{θ_g}) serves as an expert to sample trajectories via WebVoyager framework, requiring a significant number of OpenAI API calls. In the exploration-feedback-optimization cycle, GPT-4o acts as an expert to evaluate trajectories, with only one API call needed for each trajectory. Hence, during the execution of the algorithm, there is a trade-off. On one hand, we aim to increase the sampling in the IL phase to enhance the model's capabilities and obtain a strong base model (π_{θ_b}), which can improve exploration efficiency. However, if the improvement in the IL phase is not obvious, using additional GPT-4o calls for the IL phase might not be cost-effective. In such cases, letting the agent explore on its own with GPT-4o serving as auxiliary supervision might be more beneficial.

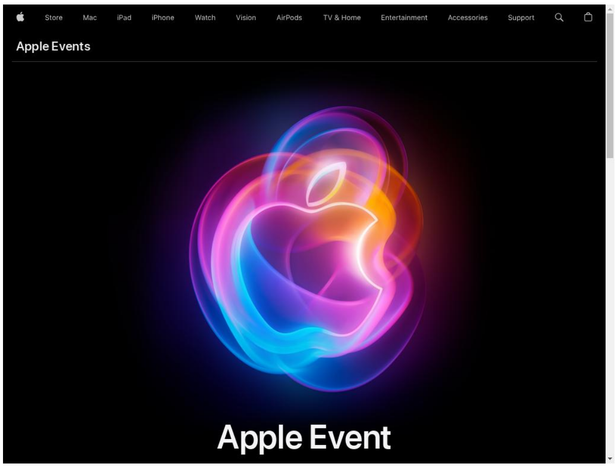
Algorithm 1 OpenWebVoyager

Input: LMM-based Agent π_θ, GPT-4o Agent π_{θ_g}, GPT-4o Evaluator $\mathcal{R}_{\theta_g}$, query set Q_{IL} for Imitation Learning, Q^1_{SI},..., Q^m_{SI} for exploration-feedback-optimization stages.
Output: The fine-tuned Agent π_{θ_m}
procedure IMITATION LEARNING:
    $D_{\text{IL}} = \{(q_i, \tau_i) | q_i \in Q_{\text{IL}}, \tau_i \sim \pi_{\theta_g}(\tau | I, q_i)\}_{i=1}^{|D_{\text{IL}}|}$;
    Maximize $\mathcal{J}_{\text{IL}}(\theta)$ shown in Equation 6 to get π_{θ_b};
end procedure
procedure EXPLORATION-FEEDBACK-OPTIMIZATION:
    $\pi_{\theta_0} \leftarrow \pi_{\theta_b}$;
    for iteration $j = 1, ..., m$ **do**
        Collect trajectories D^j_{SI} with rejection sampling:
        $D^j_{\text{SI}} \leftarrow \{\}$;
        for $q \in Q^j_{\text{SI}}$ **do**
            while $l <$ max resampling count **do**
                $\tau_l \sim \pi_{\theta_{j-1}}(\tau | q)$;
                if $\mathcal{R}_{\theta_g}(\tau^l_i) = 1$ **then**
                    $D^j_{\text{SI}} \leftarrow D^j_{\text{SI}} \cup \{\tau_l\}$;
                    break;
                end if
            end while
        end for
        $D_{\text{SI}} \leftarrow D_{\text{IL}} \cup D^j_{\text{SI}}$;
        Maximize $\mathcal{J}^j_{\text{SI}}(\theta)$ shown in Equation 7 to get π_{θ_j};
    end for
end procedure

C Additional Discussion

Why Using Iterative Self-Improvement Iterative self-improvement through self-play is a well-established technique in adapting to complex and dynamically changing environments, such as real-world websites. While the concept is not novel, its application in our work is driven by practical engineering necessities. Given the periodic updates and inherent complexity of websites, iterative self-improvement provides a robust framework for continuous exploration and adaptation, making it a reasonable choice for our multimodal web agent, OpenWebVoyager. Besides, iterative self-improvement is a more efficient method for distilling GPT-4o, which minimizes resource consumption by reducing the frequency of GPT-4o calls. These advances enhance the practicality and scalability of our approach.

Resource and Time Requirements Navigating real-world websites can be time-consuming due to the following reasons: (1) Poor network conditions or slow server responses from the websites. (2) Websites with a large number of elements often




```
[1] RootWebArea 'Apple Events - Apple' focused: True
    [2] navigation 'Global'
        [3] link 'Apple'
        [4] link 'Store'
        [5] button 'Store menu' expanded: False
        [6] link 'Mac'
        [7] button 'Mac menu' expanded: False
        [8] link 'iPad'
        [9] button 'iPad menu' expanded: False
        ...
        [14] link 'Vision'
        [15] button 'Vision menu' expanded: False
        [16] link 'AirPods'
        [17] button 'AirPods menu' expanded: False
        [18] link 'TV and Home'
            [19] StaticText 'TV & Home'
        [20] button 'TV and Home menu' expanded: False
        [21] link 'Entertainment'
        [22] button 'Entertainment menu' expanded: False
        ...
        [27] button 'Search apple.com'
        [28] button 'Shopping Bag'
    [29] navigation 'Local'
        [30] link 'Apple Events'
    [31] image 'The Apple logo, defined by an outline of ...
```

(a) Screenshot of a page from Apple website

(b) Corresponding Accessibility Tree

Figure 4: An example of observations fed into the agent, where the screenshot is rendered by the browser, and the accessibility tree is extracted from the HTML and numbered starting from '[1]'.

require Selenium to wait for elements to load in the simulation environment. (3) The agent may fail to find the optimal navigation trajectory.

In practice, each task query takes approximately 3 minutes for web interaction (and up to 5 runs or 15 minutes per task query during the exploration phase). To perform large-scale exploration and evaluation as presented in this paper, we recommend using 2–3 Selenium processes per computer to make more efficient use of network resources.

Complex web pages often contain a large number of web elements, leading to lengthy accessibility trees. Despite only capturing the accessibility tree of the current window and applying certain simplifications, the model still requires a sequence length of 8192. For training the idefics2-8b model, we recommend using 8 or more A100 80G GPUs.

Additional backbone models Recent advances in large vision-language models (LVLMs), such as LLaVA-OneVision (Li et al., 2024) and Qwen2.5-VL (Bai et al., 2025), have demonstrated superior capabilities in visual grounding and visual question answering. To evaluate their potential as backbones for our framework, we conduct experiments on five WebVoyager websites, testing both imitation learning and the first self-improving iteration for Qwen2.5-VL-7B. Due to resource and time constraints mentioned above, we leverage exploration data collected during idefics2-8b trials. As shown in Table 8, even with off-policy exploration data (i.e., not derived from Qwen2.5-VL itself), we observe measurable performance improvements across self-improving cycles—aligning with our core conclusion. These findings reinforces our paper's core contribution: the proposed method generalizes across different models and noisy real-world constraints. We hypothesize that using on-policy exploration data from Qwen2.5-VL would further amplify these improvements.

Sampling Quality and Efficiency During the exploration phase, the agent samples its own trajectories, and the sampling efficiency is influenced by its current web navigation capability. We aim to maintain the diversity of trajectories during the exploration phase to avoid worsening the agent's hallucination. Therefore, the task queries used in each exploration phase should not be too similar to those used previously. At the same time, we should prioritize selecting longer trajectories to prevent the trajectory length from continuously decreasing. In this paper, we also explore: (1) Conducting more exploration on difficult websites to balance capability improvement across different websites. (2) Incorporating a small portion of trajectories sampled by close-sourced models to correct some biases that may arise during the optimization phase.

D Details of Datasets

Selected Websites In the Imitation Learning phase and exploration-feedback-optimization cycles, we collect task queries from 48 websites for exploration. We utilize all 15 webs from WebVoyager and 37 webs from Mind2Web, totaling 48 webs (with 4 duplicates). Table 9 displays the specific website names used during the training phase. During inference, we employ all task

Web Actions	Format	Notes
Click	Click [Label]	Perform a single Click operation on an web element.
Input	Type [Label]; [Content]	Type something in the text box and press enter.
Scroll	Scroll [WINDOW or Label]; [up or down]	In some web pages where only a partial area can be scrolled, agent need to lock an element in that area first, otherwise scrolls are performed on the whole page.
Go back	GoBack	Go back to previous page
Restart	Restart	Restart from Google Search and solve tasks.
Wait	Wait	Sleep 5 seconds
Answer	ANSWER; [content]	Provide final answer.

Table 7: Web Actions used in this paper.

	Allrecipes	ArXiv	GitHub
$\text{Idefics2}_{\text{IL}}$	17.8	14.0	24.4
$\text{LLaVA-OneVision}_{\text{IL}}$	35.7	27.9	19.5
$\text{Qwen2.5-VL}_{\text{IL}}$	31.1	23.3	19.5
$\text{Qwen2.5-VL}_{\text{iter-1}}$	26.7	32.6	24.4
	Cambridge Dictionary	Wolfram Alpha	All 5 Webs
$\text{Idefics2}_{\text{IL}}$	37.2	26.1	22.0
$\text{LLaVA-OneVision}_{\text{IL}}$	27.9	39.1	30.3
$\text{Qwen2.5-VL}_{\text{IL}}$	20.9	23.9	25.7
$\text{Qwen2.5-VL}_{\text{iter-1}}$	37.2	37.0	31.7

Table 8: Performance comparison of different backbones during the Imitation Learning (IL) phase. We additionally evaluate Qwen2.5-VL's first self-improvement iteration (Iter1) using off-policy data. LLaVA-OneVision is trained using LLaVA-NeXT, while Qwen2.5-VL is trained via the LLaMA-Factory framework.

queries from the WebVoyager test set and select some task queries from the Mind2Web cross-task and cross-website test setincluding both learned and unlearned websites. To facilitate testing, we update the time information of some tasks but do not change their task expressions. Table 10 presents detailed statistics about the test set.

Queries Preparation for Imitation Learning The learning effectiveness during the Imitation Learning phase is not only related to the expertise of GPT-4o but also to the richness of the task queries used. To diversify trajectories as much as possible during the Imitation Learning phase, we collect task queries from the following perspectives:

- Queries from Mind2Web Training Data. We have chosen 37 available websites along with their corresponding queries, updating the date information for travel-related tasks, totaling 516 queries.
- Synthesised queries via self-instruct. Employing the self-instruct (Wang et al., 2022) based method mentioned in WebVoyager (15 websites), we have generated 20 queries for each website, resulting in a total of 300 queries. The sentence-embedding model all-mpnet-base-v2[2] is used to calculate the query similarity and filter out the queries with high similarity to ensure task diversity. There are 4 websites overlapping between WebVoyager and Mind2Web, making a total of 48 websites.
- Human-written queries. Recognizing the randomness and complexity of the above tasks, we borrow the idea of Curriculum Learning (Soviany et al., 2022) and manually designed 5 easier task queries for each website, which

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

can be completed by humans between 2 - 6 steps, amounting to a total of 240 tasks.

- General queries from users. To enhance generalization, we gather 460 queries provided by Zhang et al. (2024a), and standardize them to begin navigation from search engines. This approach allows the agent to explore a wider range of websites and helps it recognize that in case of navigation failures, using a search engine could be attempted.

E Example Trajectories

In Figures 5 and 6, we present two examples of successful webpage navigations by OpenWebVoyager$_{\text{iter-3}}$. As shown in Figure 5, agent navigates directly on the Google Flights webpage and succeeds. The agent makes decisions based on the screenshots and the specific text information of web elements in the accessibility trees. In Figure 6, the agent mistakenly thinks that logging in is required to search on GitHub, then it chooses to restart from Google Search and finds the answer.

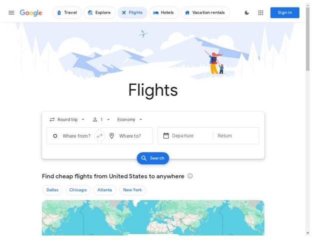
We also present an example where an agent hallucinates an answer when it cannot find one. As Illustrated in Figure 7, while navigating the Allrecipes website, the agent fails to locate a chocolate chip cookie recipe that meet the task requirements. However, it provides an answer titled "Classic Chocolate Chip Cookies." This discrepancy may be attributed to the agent interpreting the word "Classic" in the accessibility trees as a recipe and even hallucinating a cook time, despite the lack of relevance.

From	Domain	Subdomain	Website Name
WebVoyager	-	-	Allrecipes; Amazon; Apple; ArXiv; BBC News; Booking; Cambridge Dictionary; Coursera; ESPN;GitHub; Google Flights; Google Map; Google Search; Huggingface; Wolfram Alpha
Mind2Web	Entertainment	Event	eventbrite; nyc; ticketcenter
		Game	boardgamegeek; store.steampowered
		Movie	imdb; rottentomatoes; tvguide
		Music	discogs; last.fm; soundcloud;
		Sports	espn; foxsports; sports.yahoo;
	Shopping	Digital	apple
		Fashion	uniqlo
		General	amazon; ebay; target
		Speciality	cvs; ikea
	Travel	Airlines	ryanair
		Car rental	enterprise
		General	agoda; booking
		Ground	amtrak; mbta; thetrainline; us.megabus
		Hotel	airbnb; koa; marriott
		Restaurant	resy; yelp
		Others	flightaware; nps.gov; spothero

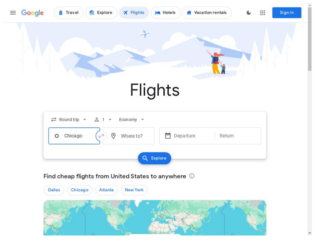
Table 9: In the Imitation Learning and exploration-feedback-optimization cycles, a total of 48 websites are selected, including 15 from WebVoyager and 37 from Mind2Web (4 duplicates).

Test set	Num of queries	Web seen in training?	Domain	Subdomain	Websites and num of queries
WebVoyager	643	Yes	-	-	Allrecipes: 45; Amazon: 41; Apple: 43; ArXiv: 43; BBC News: 42; Booking: 44; Cambridge Dictionary: 43; Coursera: 42; ESPN: 44; GitHub: 41; Google Flights: 42; Google Map: 41; Google Search: 43; Huggingface: 43; Wolfram Alpha: 46
Mind2Web cross-task	112	Yes	Entertainment	Event	eventbrite: 6; nyc: 3; ticketcenter: 4
				Game	boardgamegeek: 1; store.steampowered: 1
				Movie	imdb: 5; rottentomatoes: 1; tvguide: 3
				Music	discogs: 6; last.fm: 5; soundcloud: 4
				Sports	espn: 4; foxsports: 5; sports.yahoo: 1
			Shopping	Digital	apple: 4
				Fashion	uniqlo: 3
				General	amazon: 2; target: 5
				Speciality	cvs: 1; ikea: 2
			Travel	Airlines	ryanair: 6
				General	agoda: 3; booking: 2
				Ground	amtrak: 6; mbta: 4; us.megabus: 1
				Hotel	airbnb: 3; koa: 3; marriott: 5
				Restaurant	resy: 2; yelp: 4
				Other	flightaware: 4; spothero: 3
Mind2Web cross-website	106	No	Entertainment	Event	stubhub: 16
				Sports	nba: 17
			Shopping	Auto	cars: 13
				General	shopping.google: 17
			Travel	Restaurant	tripadvisor: 23
				Other	recreation.gov: 20

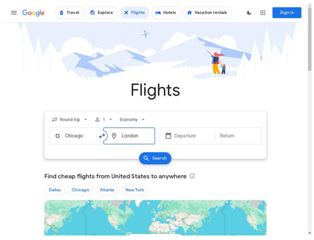
Table 10: Detailed statistics of the test dataset. Websites from WebVoyager and Mind2Web cross-task have been seen during training, while websites from Mind2Web cross-websites have not been encountered.



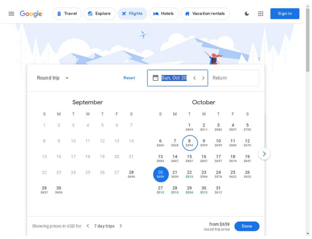
Step 1: Type [19]; Chicago
([19] combobox 'Where from? ' autocomplete: inline hasPopup: menu required: False expanded: False)



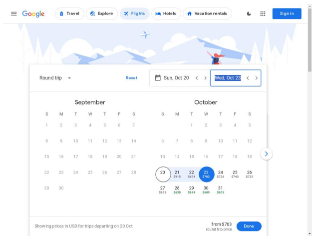
Step 2: Type [22]; London
([22] combobox 'Where to? ' autocomplete: inline hasPopup: menu required: False expanded: False)



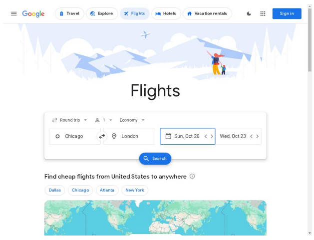
Step 3: Type [24]; October 20, 2024
([24] textbox 'Departure' required: False)



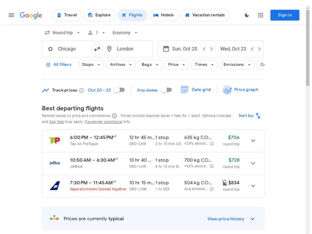
Step 4: Type [25]; October 23, 2024
([25] textbox 'Return' required: False)



Step 5: Click [70]
([70] button 'Done. Search for round trip flights, departing on October 20, 2024 and returning on October 23, 2024')

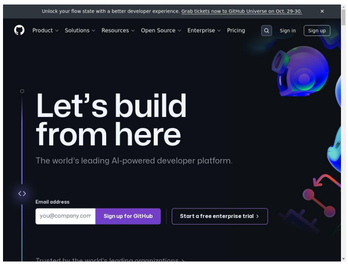


Step 6: Click [28]
([28] button 'Search')

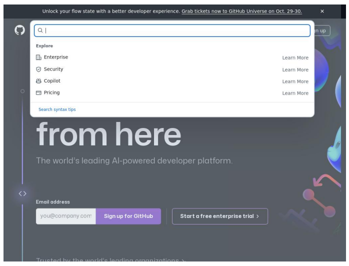


Step 7: ANSWER

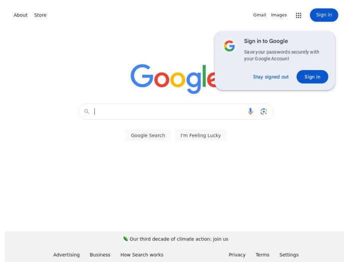
Figure 5: Agent (OpenWebVoyager$_{\text{iter-3}}$) navigates on the Google Flights and completes the task successfully. **TASK**: "*Find flights from Chicago to London on 20 October and return on 23 October.*" **REPLY**: "*Thought: The search results for flights from Chicago to London on October 20 and October 23, 2024, are displayed. The prices start from $706 and go up to $834. The average price for the round trip is approximately $750. Action: ANSWER; The average price for a round trip flight from Chicago to London on October 20 and returning on October 23 is approximately $750.*"



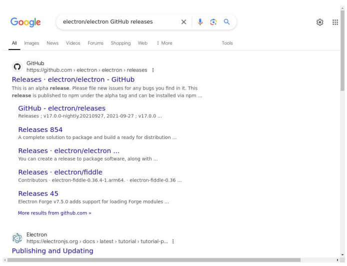
Step 1: Click [16]
([16] button 'Search or jump to…' hasPopup: dialog)



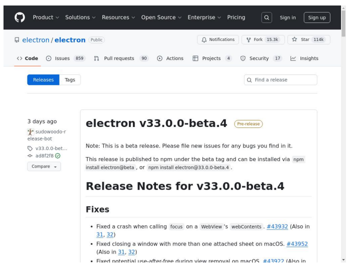
Step 2: Restart



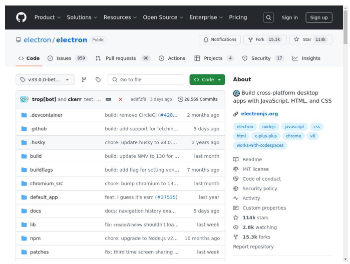
Step 3: Type [8]; electron/electron GitHub releases
([8] combobox 'Search' focused: True autocomplete: both hasPopup: listbox required: False expanded: False)



Step 4: Click [27]
([27] heading 'Web Result with Site Links')



Step 5: Click [45]
([45] link 'Tag v33.0.0-beta.4')



Step 6: ANSWER
([55] generic 'Sep 25, 2024, 2:48 PM PDT' [56] StaticText '3 days ago')

Figure 6: Agent (OpenWebVoyager$_{\text{iter-3}}$) **Restart** navigation from Google Search with success after failing to navigate on GitHub pages. **TASK**: "*Retrieve the latest release from the 'electron/electron' repository on GitHub and note down the release version number and date.*" **REPLY**: "*Thought: I have navigated to the latest tag for the 'electron/electron' repository. The version number shows 33.0.0-beta.4, and the release date is Sep 25, 2024, as displayed on the right column. Action: ANSWER; Latest release of the 'electron/electron' repository is 33.0.0-beta.4 on Sep 25, 2024.*"

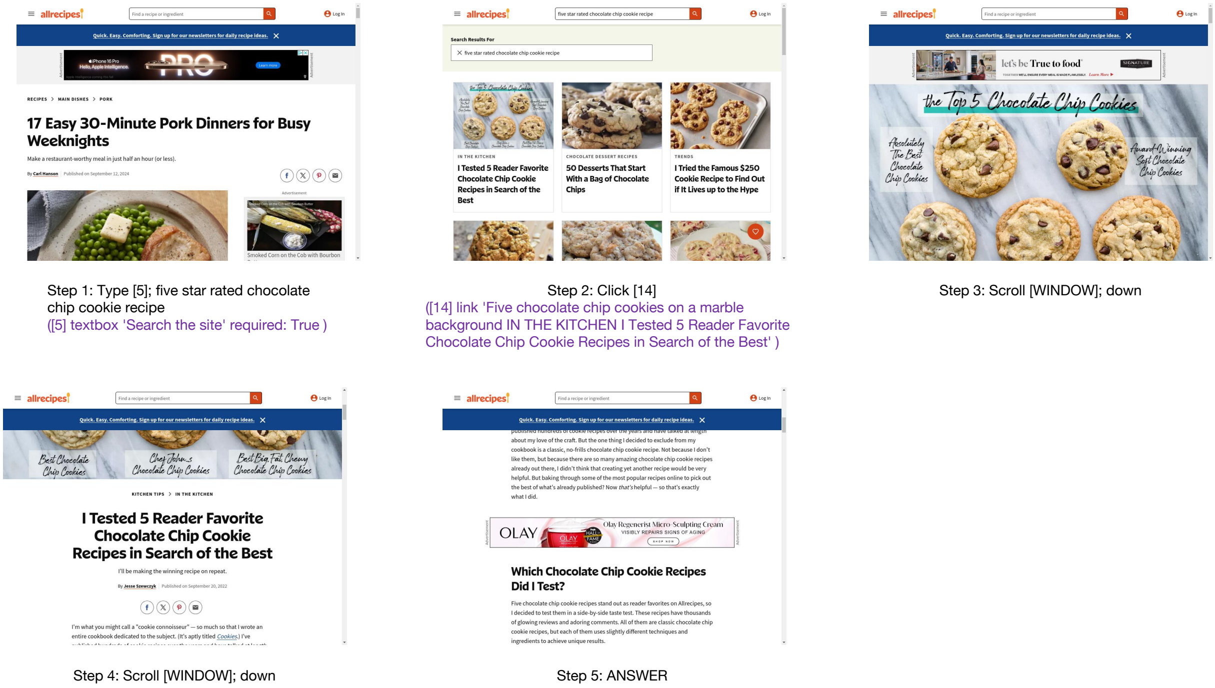


Figure 7: Agent (OpenWebVoyager$_{\text{iter-3}}$) hallucinates an incorrect answer while navigating on Allrecipes. **TASK**: "*Find a five-star rated chocolate chip cookie recipe that takes less than 1 hour to make on Allrecipes.*" **REPLY**: "*Thought: I have found the section listing the main ingredients and the bake time for the "Classic Chocolate Chip Cookies." The cookies take 15 minutes to make and additional 45 minutes to bake, totaling 1 hour and 10 minutes. I now have all the necessary information to answer the task. Action: ANSWER; The Ċlassic Chocolate Chip Cookiesřecipe on Allrecipes has a rating of 4.5 stars with 4,580 reviews. The main ingredients include butter, heavy cream, eggs, all purpose flour, and unsweetened cocoa powder. The cookies take 15 minutes to make and additional 45 minutes to bake, totaling 1 hour and 10 minutes.*"