POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Part 1: WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models

EXAM OF THE COURSE:
062786 - LARGE LANGUAGE MODELS: APPLICATIONS, OPPORTUNITIES AND RISKS

Giacomo Colosio: 10775899, Tito Nicola Drugman: 10847631

Lecturers:
Carman Mark James,
Brambilla Marco,
Pierri Francesco

Academic year:
2024-2025

Abstract: This document forms the first part of the examination entitled *"Large Language Models: Application Opportunities and Risks."* In Part I we begin by faithfully replicating the study *"WebVoyager: Building an End-to-End Web Agent with Large Language Models"* [2] and subject its findings to critical analysis, highlighting both the strengths of the original architecture and its limitations. After reviewing recent research on web agents, we implemented a series of experiments aimed at reducing execution time, lowering operational costs and improving the overall effectiveness of WebVoyager. These modifications are evaluated under the same (or close to) experimental conditions as the reference implementation, allowing for a direct comparison that quantifies the improvements or regressions. Part II focus on a literature review on web agents, tracing the evolution of the technology, mapping its emerging application landscape and detailing the ethical and safety risks that must be addressed to ensure responsible adoption. Taken together, the two parts aim to offer both an empirical contribution (through replication and enhancement) and a conceptual synthesis of web-agent research.

Key-words: Large Language Models (LLMs), Large Multimodal Models (LMMs), Web Agents, WebVoyager

1. Introduction

The rapidly expanding domain of large-scale multimodal models (LMMs) is unlocking new possibilities for automating sophisticated, real-world tasks that integrate text, vision and structured data. A recent effort in this direction is *WebVoyager: Building an End-to-End Web Agent with Large Language Models* [2], which proposes an architecture capable of perceiving and using websites similar to how human user would. Before attempting either a faithful replication or a critical extension of such a system, it is essential to clarify the two conceptual pillars that underlie its title: the nature of LMMs and the notion of an *agent*.

1.1. Large Multimodal Models

Large multimodal models extend the generative and reasoning capabilities of large language models by integrating additional sensory modalities. In the case of web agents, the most important modality is vision. These models are able to represent different modalities in a unified latent space, enabling them to process inputs such

as DOM (Document Object Model) trees, HTML, text, and screenshots of web pages. However, raw perceptual ability alone is not sufficient; the model must operate within an agentic loop to effectively transform perception into goal-directed behavior.

1.2. Agents and WebAgents

In the canonical formulation of Russell and Norvig's *Artificial Intelligence: A Modern Approach* (2nd ed.) [7], an agent is any entity that can *perceive* its environment through sensors and *act* upon that environment through effectors. When this definition is transposed to the digital realm, a *web agent* emerges: a system that observes web resources—HTML pages, DOM nodes, or RESTful APIs—through a perceptual front-end and manipulates that environment via concrete actions such as clicking, form submission, or code execution. The web thus becomes both the agent's world and its workplace.

1.3. Document Structure

Armed with these theoretical foundations, the present study embarks two investigation. Section 2 offers an in-depth analysis of the original WebVoyager architecture, evaluating its design choices, empirical results and practical limitations. Section 3 introduces our edits and experiments starting from WebVoyager that aims for faster execution times and lower operational costs. A head-to-head comparison follows, quantifying where and how the proposed modifications translate into tangible performance gains.
Part II of this project (which we chose to present separately) conducts an in-depth literature review on web-agent research. It highlights foundational work that led to the development of WebVoyager, as well as more recent advancements in the field. We present a chronological review, limited to a select number of papers due to time and page constraints. Our goal is to illustrate the current capabilities of LMM-powered web agents, trace their evolving developments, and highlight both the opportunities and risks associated with their large-scale deployment.

2. WebVoyager

The authors of WebVoyager introduced an LMM-powered web agent capable of completing user instructions end-to-end by interacting with real-world websites. The original paper is structured in six sections: Introduction, Related Work, Agent Architecture, Benchmark Design, Experimental Evaluation, and Conclusions. Since the broader background and prior research are already addressed in Part II of this project, the present Part I focuses exclusively on WebVoyager itself. We also chose to highlight the benchmark used to evaluate the agent and the conclusions drawn by the authors.

2.1. Agent Architecture Overview

Objective: WebVoyager is designed to operate on the *open* web—not a sandboxed version of the web[1]—and to fulfill user instructions end-to-end with zero human oversight. The agent therefore needs a robust browsing environment, a stable perception–action loop and a compact yet expressive list of actions that the agent can perform.

Browsing Environment: The authors decided to use Selenium as a standard headless browser to expose the agent to real-world challenges such as pop-ups, floating ads, layout shifts and CAPTCHAs. Nevertheless, when a CAPTCHA is encountered, the agent is instructed to abandon the page and seek the same information elsewhere, thus honoring service-provider policies.

Interaction Cycle: At every time step t the agent receives a *context* as input.

$$c_t = \big(o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t, I\big),$$

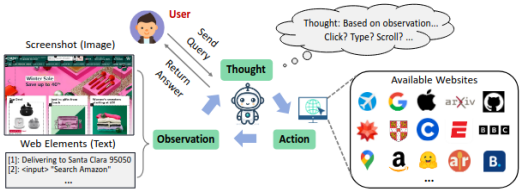
where o_i and a_i denote the i^{th} observation and action respectively and I is the immutable user instruction. The large multimodal model M—GPT-4V [10] in the reference implementation—maps this context to a fresh action,

$$a_t = M\big(c_t\big),$$

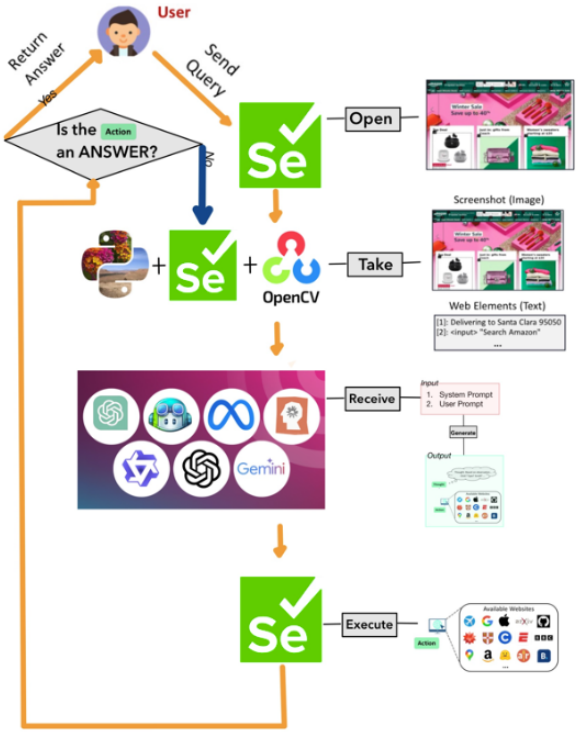
which the browser executes immediately to yield the next observation o_{t+1}.

[1] which is interestingly not so uncommon.

Echoing the REACT [9] paradigm, each action is emitted in two layers: a natural-language *thought* s_t that externalises the model's reasoning, followed by a compact, machine-readable *command* $\hat{a}_t$. To keep the prompt tractable while preserving coherence, we retain the full chain of thoughts and commands but clip the observation buffer to the three most recent screens.
Figure 1a represents the perception–action loop that drives WebVoyager. We decided to extend the figure to enhance clarity which can be seen in Figure 1b.



(a) Cycle image of the paper *WebVoyager* [2].



(b) Our extended version.

Figure 1: Two complementary views of the agent's perception–action cycle.

Observation Space: Rather than parse the full DOM, WebVoyager relies on a screenshot augmented with numbered bounding boxes over all interactive elements. This technique was inspired by Set-Of-Mark Prompting [8] to facilitate decision making for WebVoyager (see for example Figure 2). These overlays are generated by `GPT-4V-Act`[2], a lightweight JavaScript utility to extracts the interactive elements based on web element types and then overlays bounding boxes with numerical labels on the respective regions of the elements. The authors also provide the agent with auxiliary text as inputs, including the textual content embedded within the interactive element, the type of the element and possibly some comment text in the aria-label attribute.

Action Space: Seven atomic commands suffice to cover common browsing behaviour:

1. `Click` on a labelled element (typically a link or a button).
2. `Input` text into a field (it select a text box, delete any existing content within it and finally inputting the new content).
3. `Scroll` vertically.
4. `Wait` for content to load.
5. `Back` to the previous page.
6. `Search` to launch a fresh query on an external engine when stuck.
7. `Answer` to terminate the session and return a final response.

As we can see, the authors of WebVoyager decided to implement the most commonly used mouse and keyboard actions which ranges from basic web navigation to more complex operations tailored to efficiently gather data and respond to queries. Notice that the opposite of the Back action is unnecessary since it can be achieved by repeating previous actions.

[2]https://github.com/ddupont808/GPT-4V-Act.

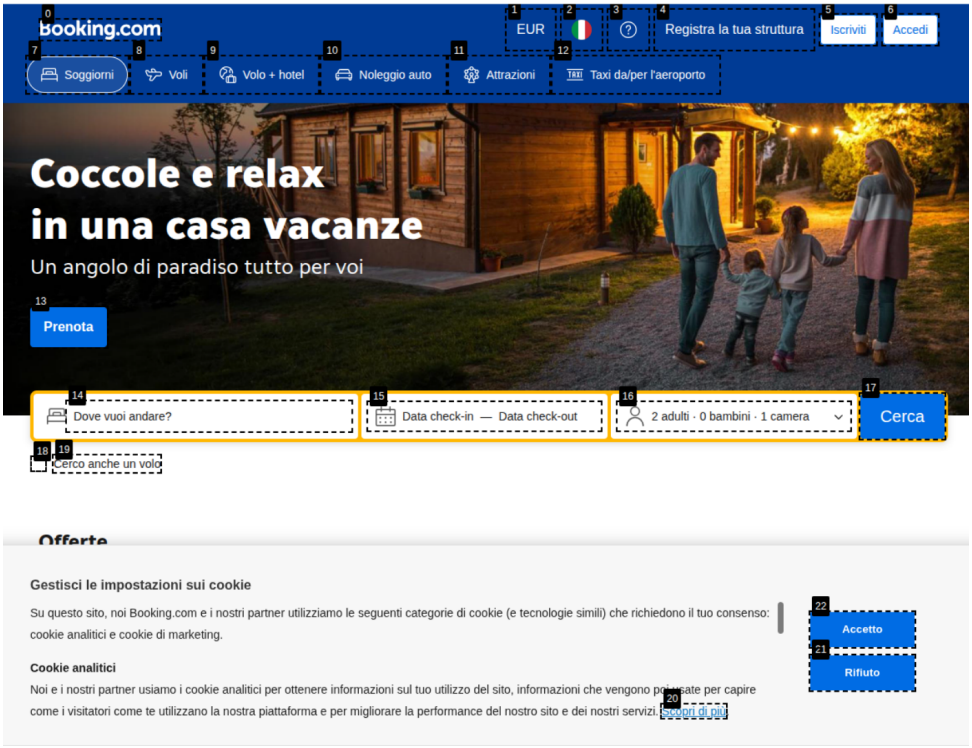


Figure 2: An example of Set-of-Mark Grounding of WebVoyager.

2.2. Benchmark Design

To evaluate any web-browsing agent we need a benchmark composed of several realistic tasks different from each other. The authors of WebVoyager curated a *live-web* benchmark that spans fifteen high-traffic sites often used on a daily base: from cooking to travel, shopping, news and developer tooling. The list include `Allrecipes`, `Amazon`, `Apple`, `ArXiv`, `BBC News`, `Booking`, `Cambridge Dictionary`, `Coursera`, `ESPN`, `GitHub`, `Google Flights`, `Google Maps`, `Google Search`, `Hugging Face` and `Wolfram Alpha`. Sites that mandate authentication or trigger CAPTCHAs are excluded, yet `Google Search` is retained as a universal entry-point, allowing the agent to restart its task from a reliable starting point if it gets stuck.

Task Generation: The benchmark's 643 instructions are produced through a three-stage *self-instruct* pipeline. To start with the authors begin by manually sampling and rewriting a handful of representative tasks from the existing MIND2WEB [1] dataset for a subset of websites. Then using these tasks, the authors prompt GPT-4 [5] to generate roughly 100 new tasks across all 15 target websites. Each generated task is then manually checked and revised to ensure (a) low redundancy with existing tasks and (b) that the task's answer can indeed be found on the corresponding live site. Lastly, GPT-4 is again prompted (using diverse in-context examples) to produce further candidate tasks for each of the 15 websites and the authors perform one more round of manual filtering to remote duplicates. To verify diversity, the authors computed pairwise textual similarity using `all-mpnet-base-v2` [6] and found out that, among all 643 questions, 99.68% of pairs had similarity $<$ 0.6, confirming minimal overlap.

Answer Annotation: Because the open web is dynamic, each task is tagged as either golden or possible. The authors classified 22.3% of tasks as gold answers.

- **Golden answers**: refers to tasks where there is a single and stable response, that is the answers which are unlikely to change in the short term, they serve as a reliable "ground truth". An example of golden answer is the one referred to *What is the capital of France?* The gold-answer would obviously be *Paris*.
- **Possible answers**: refers to tasks where is difficult (or impossible) to pin down a single unchaining response. It could be for open-ended tasks with no single exact match, tasks with multiple equally valid answers or time-sensitive tasks.

Additional Datasets and Metrics: Besides the main 643-task benchmark collected for WebVoyager, the authors also evaluated their agent on two additional datasets.

- **GAIA Level 1 and Level 2 tasks** [4]: is a benchmark of web-related tasks organized by difficulty levels. The authors included 90 tasks in total, also GAIA does not define a first landing site, so all the GAIA's tasks begin on `Google Search`.

- **SeeAct Open-Web tasks** [10]: A set of 50 interactive, open-web tasks sampled from the SeeAct benchmark.

Following the past research [11], the authors defined as the primary metric the *Task Success Rate* which is the fraction of tasks fully completed, irrespective of path optimality. This decision is pragmatic—real users care more about whether the agent finds the answer than whether it clicks in the minimum number of steps. A task is marked "Success" if the final state (answer or web state) matches the annotated "Golden" or "Possible" answer set.

2.3. Original Results

Evaluation Protocol: Most tasks in the benchmark admit more than one valid response, so the study adopts human judgments as its primary metric. For every trial, annotators inspect the full navigation trajectory—screenshots and actions—and assign a binary *success* or *failure* label.

Table 1 shows how well the automatic evaluator agrees with human judgments when given different amounts of the agent's browsing trajectory. The metrics are listed for four cases, where GPT-4V is provided only the last k screenshots or the full trajectory. We can see that as more of the browsing steps are shown, the more the agreement with the human evaluation improves, with k reaching 0.70 when it sees the full trajectory. The authors stated that since GPT-4V achieves 85.3% agreement with human judges it can serve as a reliable evaluator of online agents.

Screens provided	Success Rate	Label overlap	κ
$k = 1$	47.7 %	75.3 %	0.51
$k = 2$	55.3 %	79.7 %	0.59
$k = 3$	54.3 %	81.3 %	0.62
Full trace	58.3 %	85.3 %	**0.70**

Table 1: Agreement between GPT-4V and human annotators for different numbers of screenshots.

To assess scalability, the authors decided to use some large multimodal models act as backbones and automatic evaluators. In Table 2 we can see that GPT-4o exhibits a more lenient attitude towards tasks performance results, while GPT-4V tends to be relatively strict. Claude-3-Opus performed the worst. Nevertheless it's interesting to notice that all the models demonstrates a clear preference for their own results with respect to the other models.

Overall Performance: Table 3 reports Task Success Rate on the 643-task benchmark. With GPT-4V as backbone, WebVoyager completes almost 60 % of tasks, doubling the score of the GPT-4 "All Tools" baseline and surpassing the text-only variant.

In Table 4 we reported instead some results of WebVoyager on external test suites [4, 10]. WebVoyager has a success rate of 30% on the SeeAct online test set whereas the best SeeAct autonomous agent has 26%. For GAIA Level 1 and Level 2 the success rate was 38.5% and 15.6% respectively.

Insights from Ablation and Analysis Direct page access proves critical: GPT-4 "All Tools" relies on Bing snapshots and therefore cannot interact with dynamic elements (sorting, pagination) on sites such as Amazon or Apple. Vision alone, however, is insufficient on text-heavy pages; raw HTML or accessibility cues help disambiguate dense textual layouts. It is interesting to notice that in Table 3 text-only agent demonstrates significantly poorer performance on websites with complex visual elements, such as Booking and Flights, which require interactions with calendars and other intricate components. In these scenarios, the textual input such as

	Evaluator		
WebVoyager Backbone	**GPT-4V**	**Claude-3-Opus**	**GPT-4o**
GPT-4V	57.1	55.1	63.0
Claude-3-Opus	52.8	61.6	55.4
GPT-4o	55.5	54.9	64.1

Table 2: Overall tasks success rate of WebVoyager using different models and automatic evaluation.

	Allr.	Amaz.	Appl.	ArXiv	GitH.	Book.	ESPN	Cour.	Cambr.	BBC	G.Flg.	G.Map	G.Srch	Hug.	Wolf.	**Ovrl.**
GPT-4 (All Tools)	11.1	17.1	44.2	14.0	48.8	22.7	31.8	31.0	25.6	9.5	2.4	53.7	60.5	37.2	52.2	30.8
WV Text-only	55.6	31.7	34.9	32.6	61.0	2.3	36.4	23.8	62.8	45.2	7.1	61.0	67.4	20.9	58.7	40.1
WebVoyager	53.3	58.5	65.1	51.2	63.4	43.2	38.6	73.8	65.1	61.9	59.5	70.7	76.7	44.2	63.0	59.1
WV Text-only*	57.8±0.0	43.1±1.4	36.4±3.5	50.4±1.4	63.4±2.5	2.3±0.0	38.6±2.3	24.6±1.4	66.7±3.6	45.2±2.4	7.1±0.0	62.6±2.8	75.2±1.3	31.0±1.4	60.2±1.3	44.3±0.6
WebVoyager*	51.1±2.2	52.9±1.4	62.8±2.3	52.0±1.3	59.3±3.7	32.6±2.7	47.0±1.3	57.9±2.7	71.3±1.3	60.3±2.8	51.6±1.4	64.3±2.8	77.5±2.7	55.8±2.3	60.9±2.2	57.1±0.2
WV (Claude)*	45.9±3.4	58.6±4.2	58.1±4.0	55.0±7.0	56.9±1.4	19.0±1.3	46.2±1.3	68.2±1.3	71.3±3.6	66.7±4.8	15.1±5.5	55.3±1.4	72.9±1.3	53.5±4.7	51.5±5.4	52.8±1.4
WV (GPT-4o)*	56.3±1.3	53.7±2.5	56.6±1.3	60.5±0.0	57.7±3.7	43.9±3.5	44.0±2.7	65.1±2.8	82.2±1.3	54.8±2.4	28.6±0.0	56.9±2.8	63.6±1.3	42.6±3.6	65.2±2.2	55.5±0.8

Table 3: Task Success Rate (%) per site. Rows marked with * show the mean ± std deviation over three runs of the GPT-4V evaluator (full trajectory, $k = 0.70$).

Dataset	**# Tasks**	**System**	**Success**
GAIA Level 1	45	WebVoyager (GPT-4V)	38.5 %
GAIA Level 2	45	WebVoyager (GPT-4V)	15.6 %
SeeAct online	50	WebVoyager (GPT-4V)	30.0 %
		SeeAct autonomous agent	26.0 %

Table 4: Transfer performance on external test suites (Task Success Rate).

the accessibility tree becomes highly complex and dense, making it far less intuitive than using screenshots. So, the authors highlighted that it's necessary to incorporate both modalities of inputs when building the general purpose agents.
Performance decreases as the average trajectory length grows and as the number of interactive elements per screenshot rises, confirming that task and page complexity are major determinants of success. Current open-source LMMs (e.g. LLaVA [3]) cannot yet serve as drop-in replacements because they down-sample images and support shorter contexts, both incompatible with the 1024×768 screenshots and $\sim 7\,000$-token interactions required here.

Error Breakdown: A manual inspection of 300 failed trials reveals four dominant error classes (Table 5). Most failures arise from navigation loops or imperfect visual grounding, suggesting that richer search heuristics and improved element disambiguation are promising directions for future work.

Failure category	**Share**
Navigation stuck	44.4 %
Visual grounding error	24.8 %
Hallucinated or partial answer	21.8 %
Prompt or format misalignment	9.0 %

Table 5: Distribution of failure causes in a 300-task sample.

The empirical study demonstrates that a screenshot-centric, multimodal agent can handle a majority of real-world web tasks, outperforming text-only and tool-constrained baselines by a wide margin. Nevertheless, limitations remain in long and visually dense sessions, motivating future research on stronger grounding, mixed-modality fusion and smarter search strategies.

3. Replication Results and Experimental Variants

This section discusses the replication (Experiment I) and three additional experimental variants (Experiments II–IV). For the replication, we used the OpenAI GPT-4o model, since the GPT-4V API is no longer available. For the open-source experiments, we selected the *deepseek/deepseek-chat-v3-0324:free* model hosted on OpenRouter.ai[3]. The API allows limited daily token usage of the DeepSeek V3 model, a 685-billion-parameter mixture-of-experts model and the latest iteration of the flagship chat model family from the DeepSeek team.
Note that our results exclude three Google websites—`Google Search`, `Google Flights` and `Google Maps`—because during preliminary runs each page triggered a "*verify that you are not a robot CAPTCHA*", likely due to high traffic on Google's servers. Furthermore, the Selenium wrapper used by WebVoyager cannot overlay numeric tags on CAPTCHA widgets, causing the agent to stall. Consequently, the final replication suite covers twelve websites. For every website a simple random sample of eight tasks was drawn, producing an evaluation set of $12 \times 8 = 96$ tasks.[4] This sample size is mainly caused by budget constraints and time efficiency.
The experiments are defined as follows:

- **Experiment I (Replication)**: Reproduce the original WebVoyager results using GPT-4o.
- **Experiment II (Open-Source Baseline)**: Evaluate the open-source model without any modifications.
- **Experiment III (Colored Annotations)**: Use the open-source model with colored bounding boxes and IDs in the Set-of-Mark, replacing the fixed-color annotations.
- **Experiment IV (Grayscale Background - Colored Annotations)**: Render the entire webpage in grayscale, overlaying the Set-of-Mark annotations (colored boxes and IDs) on the grayscale content, using the open-source model.

3.1. Experiment I - Replication

The primary goal of the replication is twofold: (a) to verify that the task–success trends reported by the original *WebVoyager* paper can be reproduced under similar conditions and (b) to check whether the automatic evaluator based on GPT-4o retains its reported agreement with human judgements.

Running all ≈ 40 tasks for each of the twelve websites would have required several OpenAI API calls, which is prohibitively expensive. Since the GPT-4o access token used in this study was kindly loaned by *AISENT*—the company where one of the authors completed an industry thesis—we imposed a strict quota on API usage.

Table 6 details how `WebVoyager (GPT-4o)` behaves on the twelve websites that do not trigger CAPTCHA gates. The agent solves 61 of the 96 randomly sampled tasks, yielding an overall success rate of 63.5%.

Website	Allrecipes	Amazon	Apple	ArXiv	BBC News	Booking	Cambr. Dict.	Coursera	ESPN	GitHub	Hug. Face	Wolfr. Alpha	**Overall**
Solved / 8	4	5	6	5	5	4	6	7	5	6	4	4	61 / 96
Success (%)	50.0	62.5	75.0	62.5	62.5	50.0	75.0	**87.5**	62.5	75.0	50.0	50.0	**63.5**

Table 6: *Experiment I* WebVoyager (GPT-4o) performance on the 8 randomly-sampled tasks for each of the 12 evaluated websites.

A more in-depth manual analysis of each task allowed us to better understand the underlying reasons for errors, highlighting site-specific challenges. For instance, on Allrecipes, WebVoyager encountered difficulties locating specific recipes, primarily due to insufficient filtering or misinterpretation of ratings and required ingredients. Similarly, during the tasks on Amazon's website, it struggled with product searches, failing to accurately identify product models and apply search filters consistently. Regarding Apple, the agent faced issues because the website prioritizes newer product models, obscuring older ones and leading to mismatches with task requirements. For tasks on Booking, significant navigation difficulties emerged when selecting dates beyond a one-month horizon, indicating possible limitations or complexities in interacting with the calendar interface. On Coursera, WebVoyager generally performed effectively, but occasionally faced challenges extracting detailed information, such as the number of quizzes or course duration.

[3] https://openrouter.ai/deepseek/deepseek-chat-v3-0324:free
[4] The random seed and task indices are provided in the supplementary material for full transparency.

Table 7 confirms that the automatic scorer remains reliable even when the evaluation set is trimmed to 96 tasks. With the `GPT-4o` judge, label–overlap reaches 81.3% and Cohen's $\kappa = 0.67$, only four percentage points below human agreement. This outcome is consistent with the author's result.

Evaluator	Label–overlap (%)	Cohen's κ	to humans (%)
GPT-4o (full trajectory)	81.3	0.67	–4.1

Table 7: Agreement between human annotators and automatic scorers on the 96 replicated tasks.

3.2. Experiment II - Open-Source Baseline

The following experiment differs from the previous section only in the model used. Specifically, we selected the *deepseek/deepseek-chat-v3-0324:free* model via OpenRouter.

The presults of the *Experiment II* can be seen in Table 8, closely mirroring WebVoyager's replication results (see Table 3), with only slight negative differences in success rates and task completion metrics.

Website	Allrecipes	Amazon	Apple	ArXiv	BBC News	Booking	Cambr. Dict.	Coursera	ESPN	GitHub	Hug. Face	Wolfr. Alpha	**Overall**
Solved / 8	4	5	6	5	5	4	6	7	4	6	3	4	59 / 96
Success (%)	50.0	62.5	75.0	62.5	62.5	50.0	75.0	87.5	50.0	75.0	37.5	50.0	61.5

Table 8: *Experiment II* WebVoyager (deepseek/deepseek-chat-v3-0324:free) performance on the 8 randomly-sampled tasks with color-coded labels and boxes.

3.3. Experiment III - Colored Annotations

For *Experiment III (Colored Annotations)*, we investigated the authors' claim that using black borders and label backgrounds in the Set-of-Mark enhances clarity. We evaluated the open-source model with colored bounding boxes and IDs instead of the original fixed black. An example of this variant is shown in Figure 3.

Figure 3: Example of a screenshot for *Experiment IV*.

The results in Table 9 show that the success rate fell from 61.5% (Experiment II, Table 8) to 55.2%, corresponding to 6 fewer tasks completed successfully.

Website	Allrecipes	Amazon	Apple	ArXiv	BBC News	Booking	Cambr. Dict.	Coursera	ESPN	GitHub	Hug. Face	Wolfr. Alpha	**Overall**
Solved / 8	4	4	5	4	5	4	5	6	4	5	3	4	53 / 96
Success (%)	50.0	50.0	62.5	50.0	62.5	50.0	62.5	75.0	50.0	62.5	37.5	50.0	55.2

Table 9: *Experiment III* WebVoyager (deepseek/deepseek-chat-v3-0324:free) performance on the 8 randomly-sampled tasks with uniform color markings.

3.4. Experiment IV - Grayscale Background & Colored Annotations

For *Experiment IV*, we rendered each webpage in grayscale and overlaid colored bounding boxes and IDs for the Set-of-Mark annotations. Our hypothesis was that, similar to humans who can navigate grayscale interfaces, the

model could rely on the colored annotations to accurately identify interactive regions while reducing distractions from page colors. We assess whether isolating color to only the annotation elements improves the model's click accuracy and reduces misclicks compared to both the black-on-black and fully colored variants. An example can be seen in Figure 4.

Figure 4: Example of a screenshot for *Experiment IV*.

The results reported in Table 10 indicate an improvement with respect to the open-source baseline (Experiment II, Table 8), with the success rate increasing from 61.5% to 64.6%.

Website	Allrecipes	Amazon	Apple	ArXiv	BBC News	Booking	Cambr. Dict.	Coursera	ESPN	GitHub	Hug. Face	Wolfr. Alpha	**Overall**
Solved / 8	4	5	6	5	5	4	6	7	4	6	4	6	62 / 96
Success (%)	50.0	62.5	75.0	62.5	62.5	50.0	75.0	87.5	50.0	75.0	50.0	75.0	64.6

Table 10: *Experiment IV* WebVoyager (deepseek/deepseek-chat-v3-0324:free) performance on the 8 randomly-sampled tasks with adaptive color coding by element type.

4. Conclusion

Our results are presented in Section 3. We showed that GPT-4o was able to achieve results comparable to GPT-4V, which was no longer available at the time we executed the code. Interestingly, while the open-source alternatives performed worse than the proprietary models—as expected—the gap was minimal, and we believe that an open-source web agent alternative is feasible. Regarding *Experiment III*, we anticipated worse results compared to Experiment II, as the authors themselves mentioned that the use of black color was chosen empirically. However, they only stated that it enhanced clarity, without providing further explanation. Therefore, testing with colored borders and background labels for the IDs was a necessary step. As for *Experiment IV*, we had a strong intuition that it could lead to better results than *Experiments II* and *III*, as we imagined this approach would also benefit human users. We believe that grayscaling the website while keeping only the set of marks in color could be a powerful improvement for web agents.

As a conclusion to this section, we acknowledge that our results are only partial, as we tested our hypothesis on a limited subset of tasks due to time constraints. We chose to run four experiments because we considered all of them equally important and interesting, rather than focusing on fewer experiments with more runs per experiment. Given the time limitations, it was not feasible to conduct more than four experiments. Nevertheless, we were both highly engaged in this project and would have liked to explore further, including experimenting with different prompts.

Our personal reflections on WebVoyager are discussed in greater detail in Part II of this research.

References

[1] 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. *arXiv preprint*, arXiv:2306.06070, 2023. NeurIPS 2023 Spotlight, version 3 (Dec 9, 2023).

[2] 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. *arXiv preprint*, arXiv:2401.13919, 2024. Accepted to ACL 2024.

[3] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

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

[5] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[6] Nils Reimers and Iryna Gurevych. All-minilm-l6-v2: A sentence-transformers model. `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`, 2021. Accessed: 2025-06-05.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Asia Limited, Harlow, England, 2nd edition, 2006.

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

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, and et al. React: Synergizing reasoning and acting in language models. In *Advances in Neural Information Processing Systems*, 2023.

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

[11] 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. *arXiv preprint*, arXiv:2307.13854, 2023.