
Multimodal Auto Validation For Self-Refinement in Web Agents

Ruhana Azam
University of Illinois
razam2@illinois.edu

Tamer Abuelsaad
Emergence.ai
tea@emergence.ai

Aditya Vempaty
Emergence.ai
aditya@emergence.ai

Ashish Jagmohan
Emergence.ai
ashish@emergence.ai

Abstract

As our world digitizes, web agents that can automate complex and monotonous tasks are becoming essential in streamlining workflows. This paper introduces an approach to improving web agent performance through multi-modal validation and self-refinement. We present a comprehensive study of different modalities (text, vision) and the effect of hierarchy for the automatic validation of web agents, building upon the state-of-the-art Agent-E web automation framework. We also introduce a self-refinement mechanism for web automation, using the developed auto-validator, that enables web agents to detect and self-correct workflow failures. Our results show significant gains on Agent-E’s (a SOTA web agent) prior state-of-art performance, boosting task-completion rates from 76.2% to 81.24% on the subset of the WebVoyager benchmark. The approach presented in this paper paves the way for more reliable digital assistants in complex, real-world scenarios.

1 Introduction

Recent studies indicate that an estimated 92% of jobs now require digital skills [Bergson-Shilcock and Taylor, 2023], driving companies to focus on workflow automation. Generative AI and other automation tools are particularly promising, with the potential to handle 60-70% of an employee’s tasks and contribute between \$2.6 trillion and \$4.4 trillion to global GDP [Chui et al., 2023]. In this landscape, web agents can play an essential role in alleviating workloads. Acting as personal assistants, they can manage schedules, automate routine activities, and efficiently complete simple tasks. By reducing cognitive load, saving time, and optimizing workflows, these tools help individuals and businesses operate more efficiently.

Earlier web agents were primarily rule-based, which limited their adaptability across diverse scenarios. However, advances in Large Language Models (LLMs), known for achieving human-like performance in various tasks [Liang et al., 2022], have sparked renewed interest in leveraging LLMs for web automation. Initial successes have been reported, with LLM-based approaches achieving success rates of 75% in WebArena benchmarks [Pan et al., 2024] and 73.1% in WebVoyager [Abuelsaad et al., 2024]. While these results are promising, they highlight the need for further improvements to meet the reliability standards required for widespread adoption.

We propose using a self-refinement method [Madaan et al., 2023] to enhance web navigation agents. This method employs a validator to provide feedback and self-correct workflow failures. Like many real-life planning settings, web- navigation, in non-synthetic settings, does not have an inherent reward model. Drawing inspiration from LLM-as-a-judge [Zheng et al., 2024], we aim to build a

task-agnostic validation system for web tasks without human supervision. Our validator achieves over 84% agreement with human annotators on WebVoyager tasks, and we demonstrate that the generated labels can improve the performance of web navigation planners. By utilizing feedback from the validator for unsuccessful workflows, our self-refinement mechanism improves agent performance on a subset of WebVoyager tasks from 76.2% to 81.24%.

Our main contributions are:

- Providing a comprehensive understanding of how different modalities can be utilized to build a reliable task-agnostic validator for web workflows. This includes understanding the effect of using a hierarchy of agents to distill information that is fed into the validator.
- Showing that self-refinement can reach SOTA performance for web agents in the Web Voyager tasks without any additional supervision from humans. This proves that agents can self-refine for web navigation and the benefits can be seen with self-validation.
- Highlighting some practical issues that may be faced when implementing a web agent. These findings underscore the complexity of real-world web environments and provide valuable insights for future research.

Moreover, unlike prior work [Putta et al., 2024, Lutz et al., 2024, Koh et al., 2024, Zhou et al., 2023], which primarily evaluates synthetic settings, our approach focuses on real-world applications using the Web Voyager benchmark [He et al., 2024]. This shift toward real-world evaluation not only showcases the adaptability of our approach but also underscores the practical challenges of deploying web agents in live environments.

2 Background

Self-Improving Agents Recent research has focused on enhancing the capabilities of LLMs during training or inference without additional human supervision [Wei et al., 2022, Chen et al., 2023, Wang et al., 2023, Kojima et al., 2023]. Techniques like chain-of-thought prompting and self-consistency, as used in Huang et al. [2022], aim to generate higher-quality outputs. Other methods, such as Self-refine [Madaan et al., 2023], Reflexion [Shinn et al., 2023], and REFINER [Paul et al., 2024], focus on iterative refinement of outputs using actor-critic pipelines or task decomposition. These approaches have been successfully applied to web agents, improving the performance of LLMs in web automation tasks [Putta et al., 2024, Pan et al., 2024, Lutz et al., 2024].

Web Agents The reasoning and planning capabilities in large language models (LLMs) have sparked interest in developing autonomous web agents capable of navigating complex online environments. Several frameworks have emerged to enhance LLM performance in agentic settings, such as chain-of-thought prompting [Wei et al., 2022] and the ReAct paradigm [Yao et al., 2022b], which combine reasoning and action steps. Leveraging such paradigms has shown promising results in prompting web agents [He et al., 2024]. Moreover, using these paradigms with multimodal observations has been shown to boost performance [Koh et al., 2024] further.

Researchers have also explored the use of more traditional search procedures in conjunction with self-critique mechanisms. Putta et al. [2024] has used self-critique as an intermediate reward signal for Monte Carlo Tree Search (MCTS) [Putta et al., 2024]. A similar approach by Lutz et al. [2024] includes a planner that backtracks using self-critique feedback. These labeled trajectories are then used in context to plan future tasks. Other work, inspired by Reflexion [Shinn et al., 2023], has focused on providing reward signals based on visual information at the end of each workflow [Pan et al., 2024].

Websites are textually represented with Document Object Models (DOMS). While most works have focused on the use of raw DOM and/or screenshot signals for planning [He et al., 2024, Pan et al., 2024, Putta et al., 2024, Lutz et al., 2024], a different line of work has focused on the effect of making DOMs more interpretable for LLM-based planners. Abuelsaad et al. [2024] utilizes a hierarchy of agents to create a more interpretable understanding of the DOM and sequence of actions for the planner agent.

Web Navigation Benchmarks As the capabilities of digital web agents have improved, benchmarking methods have become increasingly sophisticated. While early research relied on simplified

website simulations for controlled studies [Shi et al., 2017, Liu et al., 2018], recent efforts have developed more realistic environments that emulate or locally host dynamic websites [Deng et al., 2023, Zhou et al., 2023, Yao et al., 2022a]. However, these environments still lack certain unpredictable elements found in real-world websites. Addressing this limitation, WebVoyager provides a benchmark comprising 643 tasks across 15 real-world websites [He et al., 2024]. Current state-of-the-art performance on the WebVoyager dataset stands at 73.1% [Abuelsead et al., 2024]. The development and evaluation of reliable web agents remain active areas for improvement.

3 Self-Refinement For Web-Navigation

In this paper, we propose a self-refinement mechanism for web navigation tasks. A web navigation agent plans and executes a workflow for a given task, after which a validator assesses whether the task was successfully completed and provides feedback. If the task is incomplete, the agent revises its plan based on the feedback and reattempts the task. This mechanism is illustrated in Figure 1.

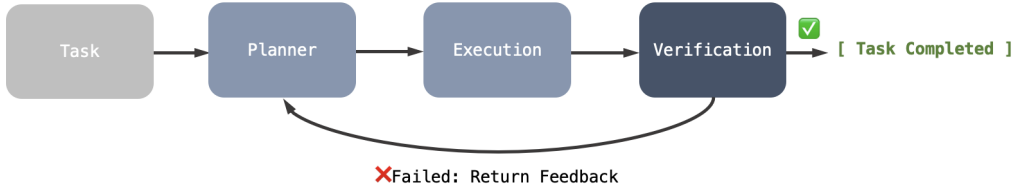


Figure 1: Self-refinement workflow.

In Section 3.1, we specify our problem setting and explain how the task of web navigation can be viewed as a planning problem. Before implementing the self-refinement method, we need to build a validator agent. In Section 3.2, we introduce our methods for building the validator. Finally, in Section 3.3, we discuss our implementation of the self-refinement method.

3.1 Problem Setting

The web can be formalized as a Markov Decision Process (MDP) defined by the tuple (S, A, P, R, γ) , where S represents web pages, A denotes user actions (e.g., link clicks), P comprises mostly deterministic state transitions, R quantifies reward function, and γ models exploration depth. This formulation frames web navigation as a planning problem: given $M = (S, A, R, P, \gamma)$, find the policy π^* that maximizes expected cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$. Although the state and action space is known in such a setting, the transition probability and the reward model are unknown. To this end, in Section 3.2, we attempt to build a validation model that can act as a reward signal in any web navigation environment.

3.2 Validation Methods

In this section, we describe our method for building a domain-agnostic web validator, which can act as a reward signal. Building on the concepts of LLM-as-a-judge [Zheng et al., 2024] and self-critique mechanisms, we utilize (V)LLMs for validation. Moreover, given that the (V)LLMs are highly sensitive to their context windows, we investigate the use of text, vision, multimodality, and hierarchical task summarization for an LLM-based validator.

Prior work has suggested that providing multimodal observations leads to the best performance in LLM-based planners [Koh et al., 2024, He et al., 2024]. Other research has pointed out that DOM observations are difficult for an LLM-based planner to interpret due to their size and verbosity, often containing irrelevant information for the task at hand. While different methods of prompting planner agents have been tested, the best practice for validation remains unclear. To address this, we investigate the use of text, vision, multimodality, and hierarchical task summarization for an LLM-based validator.

We utilize Agent-E [Abuelsead et al., 2024], a multi-agent system implemented using the Auto-Gen [Wu et al., 2024] framework. Using a hierarchy, Agent-E breaks down fine-grained navigation

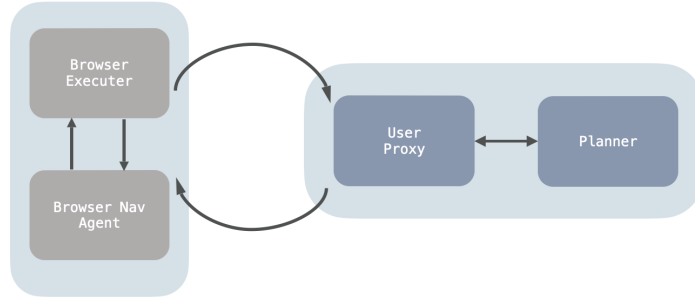


Figure 2: Multi-Agent Architecture in Agent-E.

tasks into high-level steps and simplifies complex DOMs. The hierarchical system consists of two chats. First, a chat between the planner agent and user proxy, which plans and observes the workflow at a high level. Second, a chat between the browser agent and browser nav executor, which handles the low-level observations and browser interactions. For example, the planner agent may suggest a single-step "Search for white shoes", and the browser-agent chat would break down the actions: click the search bar, type "white shoes", and click enter. Examples of the chat logs can be seen in Figure 3. This hierarchical system has been shown to have state-of-the-art performance on WebVoyager task [Abuelsead et al., 2024].

We construct our validator using three distinct modalities:

1. **Task Log (Text):** This method utilizes only the chat log between the planner agent and user proxy, containing the high-level actions and observations.
2. **Screenshots (Vision):** This method employs a sequence of screenshots taken throughout the workflow execution.
3. **Screenshots (Vision) + Final Response (Multimodal):** This method combines a sequence of screenshots with the final text response provided by the planner agent.

In the next section, we will leverage these validation methods to improve web navigation with no additional human supervision. The implementation can be found <https://anonymous.4open.science/r/Agent-E-7E43>.

Prompt: Execute the user task "Choose a dessert recipe on Allrecipes with a prep time of less than 30 minutes, has chocolate as an ingredient, and has a user rating of 4 stars or higher. Provide the name of the recipe, ingredients list, and step-by-step instructions."

Current Page: <https://www.allrecipes.com>

Main Chat

User Proxy: "Execute the user task 'Choose a dessert recipe on Allrecipes with a prep time of less than 30 minutes, has chocolate as an ingredient, and has a user rating of 4 stars or higher. Provide the name of the recipe, ingredients list, and step-by-step instructions.'" Current Page: <https://www.allrecipes.com>

Planner: "plan": 1. Use the search feature on Allrecipes to find dessert recipes with chocolate. 2. Filter the search results by prep time less than 30 minutes. 3. Sort or filter the results by user rating to show recipes with 4 stars or higher. 4. Select the first recipe that meets all criteria. 5. Extract the name of the recipe, ingredients list, and step-by-step instructions from the recipe page. 6. Use the search feature on Allrecipes to enter text 'chocolate dessert' and press enter to submit the query. 7. Terminate."

User Proxy: "I entered the text 'chocolate dessert' into the search field and submitted the query by pressing Enter on the Allrecipes website. Current Page: <https://www.allrecipes.com/search?q=chocolate-dessert>, Title: [chocolate dessert] Results from Allrecipes"

Planner: "n": next_step: "List any available filters or sorting options on the page that can be used to filter results by prep time and sort by user rating." n: terminate: "no"

Nested Chats

Browser Nav Agent: "Use the search feature on Allrecipes to enter text 'chocolate dessert' and press enter to submit the query. Current Page: <https://www.allrecipes.com/>, Title: Allrecipes | Recipes, How-Tos, Videos and More"

Tool Call: get_dom_with_content_type

Tool Response: [{"role": "WebArea", "name": "Allrecipes | Recipes, How-Tos, Videos and More", "children": [{"name": "q", "required": true, "mmid": "251", "tag": "input", "type": "text", "placeholder": "Find a recipe or ingredient", "name": "Click to search", "mmid": "352", "tag": "button", "name": "Magazine", "mmid": "285", "tag": "button", "role": "button", "name": "Newsletters", "mmid": "300", "tag": "a", "name": "q", "required": true, "mmid": "106", "tag": "input", "type": "text", "placeholder": "Search here...", "name": "Click to search", "mmid": "1057", "tag": "button", "role": "button", "name": "NEWSLETTERS", "mmid": "168", "tag": "a"}]}]

Screenshots

Figure 3: Text and vision modalities from Agent-E.

3.3 Self-Refine

Utilizing the self-refine method [Madaan et al., 2023], our approach introduces a self-correcting mechanism for workflow failures. Our system comprises a web navigation agent that plans and executes workflows, complemented by a validator that assesses task completion and provides critical feedback. In cases where a task remains incomplete, the agent leverages the validator’s feedback to revise its strategy and reattempt the task. We implement this iterative refinement process within Agent-E by integrating a verifier agent into the existing chat structure, which already includes the planner and user agent. Figure 4 provides a visual representation of the multiagent system at play.

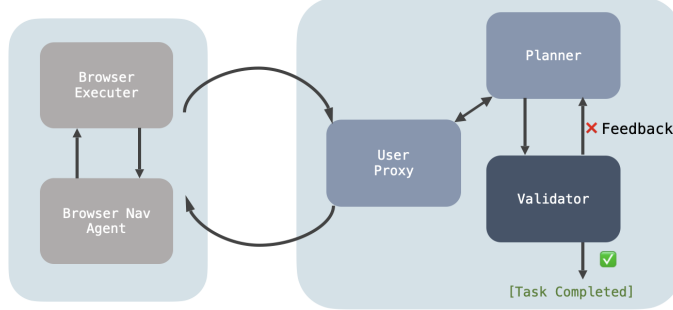


Figure 4: Multi-Agent Architecture in Agent-E with Self-Verifier.

4 Experiments & Results

In this section, we show how self-refinement can enhance web navigation without requiring additional human supervision. First, in Section 4.1, we demonstrate the effectiveness of our validation model. Our results highlight the characteristics of various modalities and state representations in prompting (V)LLMs-based validator models. While the validator is not perfect, we show in Section 4.2 that it still leads to significant performance improvements in web navigation tasks when combined with a self-refinement mechanism. Finally, in Section 4.3, we discuss key insights from our experiments to serve as valuable guidance for future practitioners.

Dataset Our experiments are benchmarked using WebVoyager. This dataset comprises 643 tasks that require agents to interact with live websites. These tasks span 15 diverse websites covering various aspects of daily life, including shopping, finding news articles, and booking flights.

Evaluation Method Since most tasks in the WebVoyager benchmark are open-ended and occur on live websites, there may be more than one correct workflow. A human annotator is employed to observe the agents’ workflows and labels each task as ‘complete’ or ‘incomplete.’ A task is considered complete only if the agent successfully finishes all parts of the instructed task and remains on the designated website. The **Validator Accuracy** measures the percentage of times the validator’s label and human annotator’s labels match.

Models We use Agent-E as our base web-navigation agent. To remain consistent with prior work benchmarks on WebVoyager [He et al., 2024, Abuelsaad et al., 2024], we utilize GPT-4-Turbo (gpt-4-turbo-preview) as a planner and browser navigator in our Agent-E implementation. For the validator, we use GPT4-Omini, GPT-4-Turbo (for text), and GPT-4V (for vision). Our exact implementation of Agent-E and validator is available at <https://anonymous.4open.science/r/Agent-E-7E43>.

4.1 Validator Results

This section demonstrates the effectiveness of different validation models that receive different modalities of the workflow. For this experiment, we run Agent-E over each WebVoyager task. We

utilize GPT4-Turbo for modalities with text only and GPT4-V for modes with vision. Then the validator and human annotators label if the task was completed by the agent.

Table 1 summarizes the performance of each modality of the validator. The Task Log (text) validator demonstrated the best performance, of 84.24%, with the multimodal validator performing similarly at 83%. The vision validator performed notably worse than the multimodal validator, indicating the importance of the agent’s final answer in some tasks. However, between 17.68 – 19.67% of tasks were labeled True Negatives. This indicates there are cases where LLMs cannot correctly plan a web navigation task, but can determine that a workflow is not completed successfully.

	True Positive	True Negative	False Positive	False Negative	Validator Accuracy
Task Log (text)	66.56%	17.68%	7.40%	8.36%	84.24%
Screenshots (vision)	52.03%	18.02%	3.60%	26.13%	70.04%
Screenshot + Final Response (multimodal)	63.33%	19.67%	5.00%	12.00%	83.00%

Table 1: Confusion matrix and accuracies of validators.

The task-specific performance of the validators can be seen in Table 2. Although overall the text validator outperforms the vision validator, this section indicates there are tasks where the vision validator performs better. The Booking.com site has a notably difficult DOM, making it consistently one of the most challenging tasks for web navigation. For these tasks, the vision validator significantly outperforms the Task Log (text) validator. Additionally, the vision validator also performed notably better for Google Flight tasks. This website is highly dynamic and requires navigating widgets which are difficult to represent in the DOM. On the other hand, highly text-based tasks perform significantly better with some text modality (e.g., Google, Huggingface, Wolfram Alpha). This difference in task-specific performance highlights the benefit of having task-specific validators.

	Amazon	Allrecipes	Apple	Arxiv	BBC News	Booking.com	Coursera	Cambridge Dictionary
Task Log (text)	75.61%	82.22%	79.07%	88.37%	92.86%	69.77%	95.24%	93.02%
Screenshots (vision)	77.50%	80.00%	80.49%	88.37%	90.48%	85.00%	92.86%	95.24%
Screenshot + Final Response (multimodal)	65.85%	84.44%	81.40%	90.70%	88.10%	83.72%	92.86%	95.35%
	ESPN	Github	Google	Google Maps	Google Flights	Huggingface	Wolfram Alpha	Overall
Task Log (text)	93.18%	90.24%	93.02%	90.24%	90.48%	70.00%	91.30%	84.24%
Screenshots (vision)	95.35%	94.74%	65.38%	66.67%	92.11%	56.25%	33.33%	70.04%
Screenshot + Final Response (multimodal)	95.35%	87.80%	92.31%	90.48%	95.12%	75.00%	90.48%	83.00%

Table 2: Validator accuracy by website.

4.2 Self-Refinement Results

In Section 4.1, we demonstrate that our validators can, in some cases, detect failed workflows. In this section, we leverage these validators using a self-refinement approach. In this experiment, Agent-E completes and revises its workflows based on feedback from our validator. The experiment is conducted on 322 tasks from the WebVoyager dataset, with the selected tasks evenly distributed across the websites — specifically, the task_id with an even number in the WebVoyager dataset.

Table 3 compares the performance of Agent-E with and without Self-Validation across text and vision tasks, using GPT-4 Omni as the validation model. The greatest performance improvement is observed in vision tasks, where Self-Validation increases accuracy from 76.2% to 81.24%. This boost in performance highlights the utility of our validator in enhancing web navigation.

	Success Rate
Self-Refine (Text)	79.81%
Self-Refine (Vision)	81.24%
Self-Refine (Vision + Final Text Response)	79.35%
Agent-E (text)	76.2%

Table 3: The success rate of web-navigation methods on 322 WebVoyager tasks.

The performance of each method, by website, can be seen in Table 4. Although the performance of different modalities is similar — with performance ranging from 79.25% – 81.24% — there are notable differences depending on the task. Tasks that are primarily text-based and performed on simple websites tend to perform best with the text validator (e.g., Google Search, Arxiv, Hugging Face, WolframAlpha). In contrast, websites that are highly dynamic with complex DOMs perform better with the vision validator (e.g., Google Flights and Booking.com). Notably, Booking.com shows performance gains of over 13% using vision over text. Using a hierarchy of agents makes DOMs more interpretable for an LLM, the vision modality proves to be more effective when the site is sufficiently complex. Additionally, we found that the multimodal validator does not outperform the text-only or vision-only validators, suggesting the potential need for specialized, task-specific validators. It is worth noting that our multimodal validator only contains text from the final planner response.

	Amazon	Allrecipes	Apple	Arxiv	BBC News	Booking.com	Coursera	Cambridge Dictionary
Task Log (text)	89.47%	78.26%	100.00%	90.48%	95.00%	23.81%	80.95%	86.36%
Screenshots (vision)	73.68%	91.30%	86.36%	66.67%	80.95%	41.18%	95.24%	95.45%
Screenshot + Final Response (multimodal)	80.00%	91.30%	90.91%	57.14%	80.95%	22.73%	90.48%	95.45%
	ESPN	Github	Google	Google Maps	Google Flights	Huggingface	Wolfram Alpha	Overall
Task Log (text)	72.73%	85.00%	95.24%	76.19%	61.90%	70.00%	91.30%	79.81%
Screenshots (vision)	100.00%	100.00%	76.19%	76.19%	85.71%	63.18%	77.27%	81.24%
Screenshot + Final Response (multimodal)	95.45%	100.00%	95.24%	80.95%	61.90%	85.71%	60.00%	79.35%

Table 4: The success rate of web-navigation methods on 322 WebVoyager tasks by website.

4.3 Technical Challenges and Observations

During our experimentation, we encountered several technical challenges and made noteworthy observations:

Screenshot Acquisition Due to variability in website loading times and browser performance, screenshot capture for sequence information was often unsuccessful. To mitigate this issue, we implemented multiple screenshot attempts. Yet, in some cases, all attempts failed. When presenting results for the vision validator, we omitted tasks where no screenshots were captured throughout the workflow.

JSON Formatting Our validator is prompted to output two parts: 1) the task evaluation and 2) the feedback. To ensure reliable parsing of these two parts, we required the agent to output them in JSON format. In instances where the JSON formatting failed, we had the validator label the workflow as incomplete and proceed with the refinement process.

Modality Specific Performance The combined vision and text approach demonstrated comparable performance to text-only methods and surpassed vision-only approaches in most tasks. We made two notable observations in cases which led to such performance:

- For question-answering tasks, the model can more easily verify answers when explicitly stated in text rather than inferred from a series of screenshots. This explains the large jump in performance between the screenshot and multimodal validator for tasks like Google.
- The vision-based models exhibited superior performance on complex websites such as booking.com and flights.com. Oftentimes, the navigator would unsuccessfully utilize filters and widgets on such websites, yet the planner agent would assume that the task was completed successfully. In such cases, the task log verifier was seen to be overly optimistic about the task completion. An example can be seen in Figure 5.

The variability in performance across different modalities and websites underscores the importance of implementing robust, multi-modal validation strategies. Future implementations can consider adaptive approaches that can dynamically select the most appropriate validation method based on the task and website characteristics. Additionally, practitioners should be prepared to handle technical issues such as inconsistent screenshot capture and JSON parsing failures, potentially by implementing fallback mechanisms or alternative data collection methods.

Task: Reserve a hotel in downtown Chicago with a rating of 9 or higher for a stay from September 20-27, 2024, which offers free cancellation and includes a fitness center.

Validator Reason: The workflow was successfully completed as all the criteria were met: a hotel in downtown Chicago with a rating of 9 or higher was found, it offers free cancellation, includes a fitness center, and the stay is from September 20-27, 2024.

Figure 5: Task log validator failure case.

5 Conclusion

In this study, we developed an effective validator and integrated it into a self-refinement mechanism. This approach allowed web agents to detect and self-correct workflow errors without additional human supervision. Our experiments on the WebVoyager benchmark demonstrated significant improvements, boosting task completion rates from 76.20% to 81.24%, surpassing prior state-of-the-art performance. While the overall performance across modalities was comparable, with success rates ranging from 79.25% to 81.24%, we observed significant task-specific variations in effectiveness. This highlights that the best modality for validation is task-dependent, suggesting the need for specialized validators.

While we encountered technical challenges, particularly in screenshot acquisition, our findings underscore the complexity of real-world web environments and provide valuable insights for future research. This work contributes to the development of more reliable and adaptable web agents. It brings us closer to the goal of creating autonomous digital assistants capable of navigating the intricacies of the modern web.

References

- Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-E : From Autonomous Web Navigation to Foundational Design Principles in Agentic Systems. pages 1–18, 2024.
- Amanda Bergson-Shilcock and Roderick Taylor. Closing the digital" skill" divide: The payoff for workers, business, and the economy. *National Skills Coalition*, 2023.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.
- Michael Chui, Eric Hazan, Roger Roberts, Alex Singla, and Kate Smaje. The economic potential of generative ai. 2023.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.
- Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024. URL <https://arxiv.org/abs/2401.13919>.
- Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

- Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. 2024. URL <http://arxiv.org/abs/2401.13649>.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL <https://arxiv.org/abs/2205.11916>.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.
- Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL <https://arxiv.org/abs/1802.08802>.
- Michael Lutz, Arth Bohra, Manvel Saroyan, Artem Harutyunyan, and Giovanni Campagna. Wilbur: Adaptive in-context learning for robust and accurate web agents. *arXiv preprint arXiv:2404.05902*, 2024.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. pages 1–54, 2023. URL <http://arxiv.org/abs/2303.17651>.
- Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*, 2024.
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian’s, Malta, March 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.eacl-long.67>.
- Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents. pages 1–22, 2024. URL <http://arxiv.org/abs/2408.07199>.
- Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL <https://arxiv.org/abs/2203.11171>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. In *COLM*, 2024.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL <https://webarena.dev>.