Enhancing Web Agents with Explicit Rollback Mechanisms

Anonymous ACL submission

Abstract

With recent advancements in large language models, web agents have been greatly improved. However, dealing with complex and dynamic web environments requires more advanced planning and search abilities. Previous studies usually adopt a greedy one-way search strategy, which may struggle to recover from erroneous states. In this work, we enhance web agents with an explicit rollback mechanism, enabling the agent to revert back to a previous state in its navigation trajectory. This mechanism gives the model the flexibility to directly control the search process, leading to an effective and efficient web navigation method. We conduct experiments on two live web navigation benchmarks with zero-shot and fine-tuning settings. The results demonstrate the effectiveness of our proposed approach.

1 Introduction

With the advancement of large language models (LLMs), LLM-powered autonomous agents have demonstrated great potential in solving a wide range of real-world tasks (Wang et al., 2024), among which web navigation is a typical example, as it requires the agent's ability to interact with the dynamic web environment (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024b).

One of the main challenges in developing web agents is dealing with the dynamic and ever-changing nature of real-world web environments. For example, web pages might be removed, or the layouts of a website can be reorganized. Due to these complexities, agents may sometimes be trapped in erroneous states, where well-planned actions may fail to produce expected outcomes. Most web agents adopt a one-way greedy search strategy, which may have difficulties getting out of unexpected or unpromising states. A straightforward solution to alleviate this issue is to explicitly incorporate more effective search algorithms.

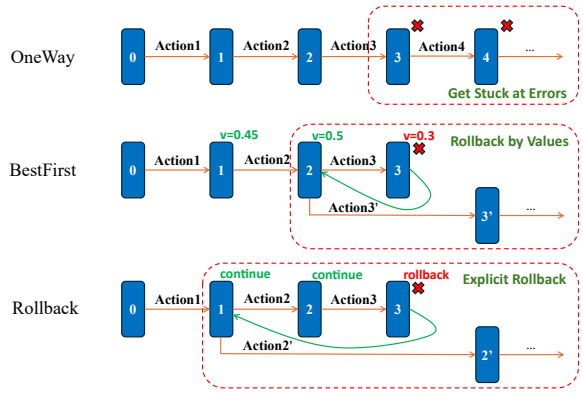

Figure 1: An overview of different search strategies. The OneWay strategy may get stuck in erroneous states, the BestFirst strategy perform rollback based on state values, while our proposed strategy directly let the models to decide *when and where to rollback*.

Search algorithms have been extensively studied in previous research for LLM reasoning. For non-interactive tasks, parallel search has been shown effective, with common techniques such as majority voting (Wang et al., 2023), breadth-first search (Yao et al., 2023), beam search (Xie et al., 2023), and Monte-Carlo tree search (Zhou et al., 2024a; Chen et al., 2024). However, it is difficult to directly apply these methods to web navigation tasks considering the much larger cost of maintaining multiple web environments to support parallel interactions. Therefore, a suitable search algorithm for web navigation tasks should maintain an overall serialized search process for efficiency while keeping the flexibility to switch to alternative paths to escape from errors. To this end, Koh et al. (2024) incorporate a best-first search algorithm, which selects the highest-scored state at each step and allows switching to more promising states. However, this approach often results in frequent state switching, which can still bring considerable overhead. The main reason is that, for an intermediate state that can lead to the final goal, it may not receive a relatively high score, since there is no intermediate evidence of its potential. In this case, the best-first

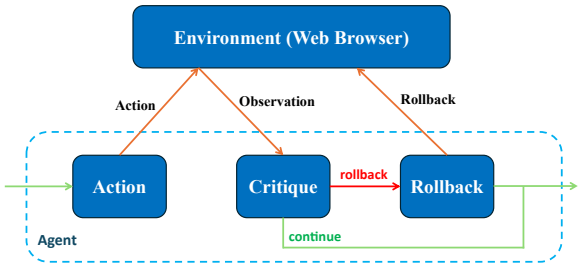

Figure 2: An illustration of the components in our web agent enhanced with explicit rollback mechanisms.

strategy might switch to another state that appears to be more promising.

In this work, we enhance web agents with an explicit rollback mechanism, which allows the agent to revert to a previous state if it judges the current state as erroneous or unpromising. As shown in Figure 1, our approach enhances the one-way search algorithm with a rollback mechanism, enabling quick escape from erroneous states while maintaining greater efficiency than the best-first algorithm with fewer state switches. One important feature of our strategy is the flexibility of the rollback operation: *we let the models to more directly influence the search process by deciding when and where to rollback*. Compared with previous backtracking methods[1] for reasoning and agent tasks (Lutz et al., 2024; Yang et al., 2025), our approach can achieve multi-step rollbacks in a single operation. One contemporary work (Li et al., 2025) similarly investigates a rollback-based approach by identifying the error locations. However, their work primarily addresses agent tasks with clean environments, which are less complex than the real-world web tasks that we explore in this research.

We conduct experiments with two typical settings: zero-shot prompting with strong LLMs and fine-tuning relatively smaller LLMs using demonstration data from larger models. Through evaluations on two real-world web navigation benchmarks, we show the effectiveness of our proposed approach, which can achieve the best overall results while maintaining efficiency.

2 Method

Given a web navigation task, the agent needs to interact with a web browser to solve the task in a step-by-step way. The agent starts with an initial state s_0, where the browser is on the homepage of the target website. At each step i, the agent receives an observation o_i (i.e., the current webpage) from the environment at the current state s_i. Based on this observation, the agent performs an appropriate action a_i. This action transitions the environment to a new state s_{i+1}, which in turn provides a new observation o_{i+1} to the agent for the next decision. These iterations continue until the target task is resolved or the navigation budget is exhausted.

[1]Many web agents (including ours) already include a "go-back" action which supports rolling back a single step.

Algorithm 1 Search with Rollback.

Input: Web Environment env, Maximum Step T.

1: $S \leftarrow []$ ▷ Navigation trajectory
2: **for** t in range(T) **do**
3: $a \leftarrow \text{ACTION}(env.state)$ ▷ Decide next action
4: **if** a == "stop" **then break end if** ▷ Task finished
5: env.step(a) ▷ Execute the action
6: $c \leftarrow \text{CRITIQUE}(a, env.state)$ ▷ When to rollback
7: **if** c == "rollback" **then**
8: $i \leftarrow \text{ROLLBACK}(a, c, S)$ ▷ Where to rollback
9: env.rollback($S[i]$) ▷ Perform rollback
10: $S \leftarrow S[:i+1]$ ▷ Slice trajectory
11: **else**
12: S.append($env.state$) ▷ Continue with current
13: **end if**
14: **end for**

We enhance the search process of web agents with an explicit rollback mechanism by incorporating model-based rollback modules. Figure 2 illustrates the modular design of our system, and Algorithm 1 provides an outline of our strategy. Similar to the greedy OneWay method, our strategy first predicts and executes an action based on the current observation (lines 3-5). Next, a binary decision is made to either continue or roll back (line 6). If a rollback is selected, the rollback operation will be performed by resetting the browser environment and the state trajectory (lines 7-10). In addition to the action module that generates actions based on the current observation, we introduce two additional components to facilitate the rollback mechanism: a critique module and a rollback module.

The critique module first makes judgments on the new state after executing the previous action, inspired by the self-refinement strategies for LLM reasoning (Shinn et al., 2023; Madaan et al., 2023). In addition to the judgments, the module also makes a binary decision: either to proceed with the current state or to revert to a previous one. This approach allows the model to directly determine *when to rollback*, offering a flexible and straightforward method to make such decisions.

Upon a "rollback" decision, an explicit rollback module is used to decide *where to rollback*. We list all preceding states as inputs, from which the module selects a specific state to revert to. This

	Mind2WebLive				Webvoyager			
	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)
Llama3.3-70B-Instruct								
OneWay	$20.92_{\pm 3.61}$	$41.59_{\pm 2.75}$	$14.4_{\pm 0.2}$	0	$38.06_{\pm 1.70}$	$57.55_{\pm 2.20}$	$12.2_{\pm 0.2}$	0
BestFirst	$21.16_{\pm 0.41}$	$\mathbf{46.45}_{\pm 0.41}$	$14.7_{\pm 0.2}$	$8.1_{\pm 0.1}$	$39.82_{\pm 3.16}$	$61.47_{\pm 3.18}$	$12.0_{\pm 0.1}$	$5.3_{\pm 0.1}$
Rollback	$\mathbf{24.07}_{\pm 1.42}$	$45.29_{\pm 0.20}$	$\mathbf{14.1}_{\pm 0.2}$	$5.0_{\pm 0.3}$	$\mathbf{44.30}_{\pm 1.32}$	$\mathbf{63.24}_{\pm 2.23}$	$\mathbf{11.8}_{\pm 0.2}$	$3.2_{\pm 0.2}$
Qwen2.5-72B-Instruct								
OneWay	$24.53_{\pm 0.70}$	$43.49_{\pm 0.85}$	$\mathbf{13.0}_{\pm 0.1}$	0	$49.56_{\pm 1.61}$	$65.21_{\pm 2.54}$	$\mathbf{10.2}_{\pm 0.3}$	0
BestFirst	$23.82_{\pm 1.18}$	$\mathbf{48.84}_{\pm 2.30}$	$13.6_{\pm 0.1}$	$6.3_{\pm 0.1}$	$47.95_{\pm 2.34}$	$64.84_{\pm 3.30}$	$10.8_{\pm 0.1}$	$4.3_{\pm 0.2}$
Rollback	$\mathbf{27.36}_{\pm 0.94}$	$45.57_{\pm 1.42}$	$13.5_{\pm 0.2}$	$4.5_{\pm 0.1}$	$\mathbf{51.90}_{\pm 4.24}$	$\mathbf{69.12}_{\pm 3.79}$	$\mathbf{10.2}_{\pm 0.1}$	$2.1_{\pm 0.2}$

Table 1: Zero-shot results with LLAMA-3.3-70B-INSTRUCT and QWEN2.5-72B-INSTRUCT.

strategy provides a flexible and efficient way to perform multi-step rollbacks, which can help the agent quickly escape erroneous states.

3 Experiments

3.1 Settings

In our agent framework, we use open-source LLMs as the backbone for all the modules. We carefully design prompts for the modules to realize their corresponding functionalities; detailed prompts can be found in Appendix A. We adopt a text-based benchmarking environment that represents web pages with accessibility trees, and the automatic browser is implemented using Playwright.[2] For the rollback mechanism, we record the URL of each step and reset the browser to the corresponding URL if a rollback is triggered. Unless specified otherwise, we set a maximum step budget of 16 and stop the navigation if this budget is used up.

We mainly compare three search strategies:[3] 1) OneWay, which is a greedy-styled search strategy that does not perform any explicit rollbacks except the built-in "go-back" action, 2) BestFirst (Koh et al., 2024), which applies state switches according to the critique's value estimation, and 3) Rollback, which is our strategy with the enhancement of specified rollback modules.

We adopt two live web navigation benchmarks for our evaluation – Mind2Web-Live (Pan et al., 2024) and WebVoyager (He et al., 2024a) – that involve interacting with real-world websites. These benchmarks are effective for assessing web agents' abilities to handle real-world environments, where unexpected problems or errors may occur. To ensure robustness, we conduct our experiments three times and report the average results. We use GPT-4o as our automatic evaluator, allowing more comprehensive assessments of the navigation trajectory. The scorer assigns fine-grained scores between 0.0 and 1.0 for the successful completion of the target task. The evaluation prompt is provided in Appendix B and more details about the experimental settings can be found in Appendix C.

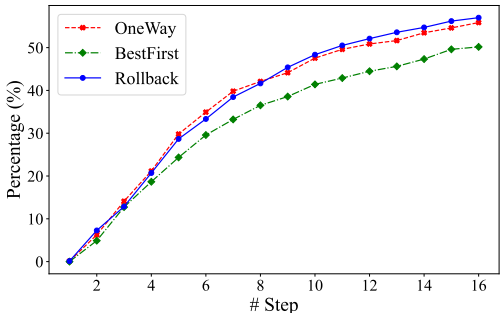

Figure 3: Task-finishing rate analysis. Here, x-axis denotes the number of steps the agent takes, and the y-axis denotes the percentage of the tasks that can be finished within a specific step limit.

3.2 Zero-shot

Setting. We start with zero-shot prompting settings, where instruction-tuned LLMs are directly prompted to perform target tasks. We adopt two widely-adopted open-source LLMs[4] for this setting: LLAMA-3.3-70B-INSTRUCT and QWEN2.5-72B-INSTRUCT, considering their impressive instruction-following abilities.

Result. The zero-shot results are presented in Table 1. Here, "Full" and "Partial" indicate full task success and fine-grained partial scores, respectively. "Step" indicates the average number of navigation steps, while "Switch" denotes the number of state

[2] https://playwright.dev/

[3] In our preliminary experiments, we also tried other methods, such as beam-search, but found them to be much less effective and efficient than the above three strategies.

[4] Appendix D provides additional zero-shot evaluation results with smaller LMs and GPT models.

	Mind2WebLive				Webvoyager			
	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)
Llama3.1-8B-Instruct								
OneWay	$20.75_{\pm 3.77}$	$38.49_{\pm 2.64}$	$14.5_{\pm 0.1}$	0	$33.92_{\pm 1.75}$	$52.97_{\pm 1.68}$	$\mathbf{11.3}_{\pm 0.1}$	0
BestFirst	$19.81_{\pm 0.94}$	$38.96_{\pm 1.42}$	$14.8_{\pm 0.2}$	$6.1_{\pm 0.2}$	$32.31_{\pm 0.44}$	$54.59_{\pm 0.68}$	$11.9_{\pm 0.1}$	$4.6_{\pm 0.1}$
Rollback	$\mathbf{21.70}_{\pm 2.83}$	$\mathbf{41.75}_{\pm 2.50}$	$\mathbf{14.2}_{\pm 0.1}$	$4.6_{\pm 0.2}$	$\mathbf{37.43}_{\pm 1.17}$	$\mathbf{57.29}_{\pm 2.90}$	$\mathbf{11.3}_{\pm 0.1}$	$2.7_{\pm 0.3}$
Qwen2.5-7B-Instruct								
OneWay	$13.38_{\pm 1.57}$	$31.42_{\pm 1.66}$	$\mathbf{13.7}_{\pm 0.2}$	0	$28.78_{\pm 2.26}$	$47.98_{\pm 1.69}$	$\mathbf{10.9}_{\pm 0.1}$	0
BestFirst	$17.91_{\pm 2.05}$	$\mathbf{37.86}_{\pm 1.86}$	$15.0_{\pm 0.2}$	$6.7_{\pm 0.2}$	$30.22_{\pm 1.21}$	$51.92_{\pm 1.41}$	$12.1_{\pm 0.1}$	$4.7_{\pm 0.1}$
Rollback	$\mathbf{20.75}_{\pm 2.67}$	$37.74_{\pm 2.04}$	$14.5_{\pm 0.4}$	$5.4_{\pm 0.1}$	$\mathbf{33.82}_{\pm 0.60}$	$\mathbf{54.57}_{\pm 1.63}$	$11.3_{\pm 0.2}$	$3.1_{\pm 0.2}$

Table 2: Fine-tuning results with LLAMA-3.1-8B-INSTRUCT and QWEN2.5-7B-INSTRUCT.

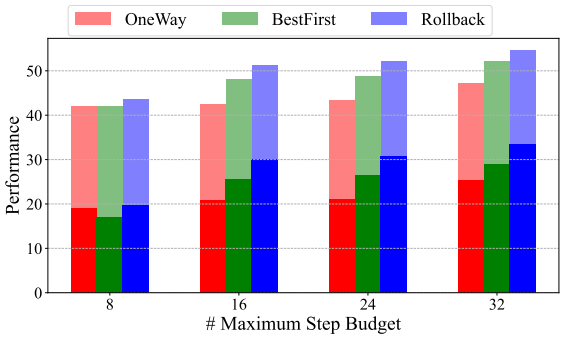

Figure 4: Results with different maximum step budgets. The light bars indicate the Partial% scores, while the darker and shaded parts represent Full% scores.

switches,[5] where rollbacks are performed. The results demonstrate a consistent pattern: Compared with OneWay, our strategy achieves better results; compared to BestFirst, our strategy requires fewer state switches and performs better in complete task success. These illustrate the effectiveness of our proposed rollback mechanisms.

Analysis. We further provide a task-finishing rate analysis with different search methods, and the results are shown in Figure 3. Overall, we can see that our strategy can complete tasks in a similar rate to OneWay. In contrast, BestFirst is less efficient, as it more frequently makes state switches.

3.3 Fine-tuning

Setting. LLMs can achieve impressive results, but usually cost too much; it would be great if we could adopt smaller LMs to complete web navigation tasks. However, when smaller LMs are directly prompted, they often struggle to comprehend complex task instructions, as demonstrated by their poorer performance shown in Table 3 of Appendix D. To address this issue, we seek to enhance smaller LMs' web navigation abilities through knowledge distillation from larger LMs. We use QWEN2.5-72B-INSTRUCT as the teacher model considering its overall better performance, and fine-tune LLAMA-3.1-8B-INSTRUCT and QWEN2.5-7B-INSTRUCT. The tuning datasets are taken from the training split of Mind2Web-Live and the imitation learning training data from OpenWebVoyager (He et al., 2024b).

Result. The fine-tuning results are presented in Table 2. The overall patterns are similar to those in the zero-shot results of the teacher models: our proposed approach can achieve a good balance between effectiveness and efficiency, yielding the overall best results at a reasonable cost.

Analysis. We further provide an analysis on test-time scaling (Snell et al., 2024) by varying the maximum step budget to assess how much the agent can improve with additional search budgets. Due to the higher computational costs of this analysis, we randomly selected 100 instances from the combination of two benchmarks. The results are shown in Figure 4. When the step budget is limited, such as with 8 steps, different methods perform similarly. However, as the number of steps increases, the strategies with rollback mechanisms improve the results more rapidly than the vanilla OneWay approach. In particular, our proposed strategy consistently achieves the best overall performance, demonstrating its effectiveness in utilizing increased step budgets.

4 Conclusion

In this work, we enhance web agents with explicit rollback mechanisms that enable them to return to previous states. Our approach allows the agent to directly decide when and where to rollback, resulting in a flexible and efficient scheme. Evaluations on two web navigation benchmarks show the effectiveness of our approach in both zero-shot and fine-tuning settings.

[5] Note that by design, the OneWay strategy does not perform any rollbacks.

Limitations

This work has some limitations. First, we focus solely on web navigation tasks, while the proposed approach could be extended to a wider range of agent tasks. Moreover, due to computational resource constraints, we only tune LMs with a scale of around 7B, while tuning and improving larger LMs could have the potential to achieve better results. Finally, our rollback mechanism is based on a straightforward URL redirection method. However, in many real-world scenarios, reverting actions can be significantly more challenging or even unfeasible due to irreversible changes.

Ethics Statement

Our research focuses mainly on the advancement of algorithms to improve web agents. Therefore, we have not applied any additional aggressive filtering techniques to the text data. We use open-source language models in their existing form, without making further modifications to improve safety or minimize bias. As a result, the text data and models that we use may have issues related to offensiveness, toxicity, fairness, or bias that we have not specifically addressed, as these are not the primary objectives of our study. Beyond these points, we do not anticipate any other ethical concerns or risks associated with our research.

References

Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. 2024. When is tree search useful for LLM planning? it depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13659–13678, Bangkok, Thailand. Association for Computational Linguistics.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024a. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. 2024b. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. *arXiv preprint arXiv:2410.19609*.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, and 1 others. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. 2025. Generator-assistant stepwise rollback framework for large language model agent. *arXiv preprint arXiv:2503.02519*.

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*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and 1 others. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. 2025. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.

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. 2024b. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

A Web Agent Prompts

ACTION MODULE

You are an assistant helping to browse and operate web pages to solve a specific task.

Available Information
- Target Task: The specific task you need to accomplish.
- Previous Trace Review: A concise review of previous actions and navigation trajectory (before the current state).
- Experience: Summaries of previous reverting points, some of which indicate failures. These can guide the next actions by avoiding unsuccessful attempts.
- Previous Tryings: Previously tried actions from the current state. Avoid repeating the unsuccessful or already-tried operations.
- Accessibility Tree: A simplified representation of the current webpage (web page's accessibility tree), showing key elements in the current window.

Actions
- click [id] link name: Click on an element with a specific ‘id‘ on the webpage. Only click on clickable elements like links and buttons.
- type [id] context: Type the ‘context‘ into the field with ‘id‘ (this action includes pressing enter by default).
- scroll up or scroll down: Scroll the page up or down.
- wait: Wait for the page to load (5 seconds).
- goback: Navigate to the previously viewed page.
- restart: Navigate to the starting URL. Use this if you think you get stuck.
- stop [answer] (summary): Issue this action when you believe the task is complete and provide the ‘answer‘. If the task is impossible to complete, provide the answer as "N/A". Include a short ‘summary‘ of all previous steps (navigation history) leading to this answer in parentheses.

Guidelines
- For complex tasks requiring multiple reasoning steps, proceed in a well-planned, step-by-step manner.
- Only issue actions that are valid based on the current observation (accessibility tree). For example, do NOT type into buttons, do NOT click on StaticText. If there are no suitable elements in the accessibility tree, do NOT fake ones and do NOT use placeholders like ‘[id]‘.
- Issue only one action at a time.
- Avoid repeating the same action if the webpage remains unchanged. Maybe the wrong web element or numerical label has been selected. Continuous use of the ‘wait‘ action is not allowed.
- Issue the ‘stop‘ action when the objective has been achieved.
- If there is a cookie banner on the page, accept it.
- Avoid assuming a specific current date (for example, 2023); use terms like "current" or "latest" if needed. If a specific date is mentioned in the user query, retain that date.
- Consider a decomposition-based method that search for simpler queries if a complex query does not yield helpful results.
- Remember to try scrolling up or down to find more information on the current page since at each step we only show the accessibility tree restricted to the current window.
- Avoid repeated tryings listed in ‘Previous Tryings‘ (if existing) since those paths have been tried before. Also check previous experience listed in ‘Experience‘ to avoid repeating failures.

Target Task
{Target task description}

Previous Trace Review
{Previous trajectories}

Experience
{Experience from previous rollbacks}

Previous Tryings
{Previously tried actions from the current state}

Accessibility Tree
{The accessibility tree for the current web page}

Output
Please generate your response, your reply should strictly follow the format (each item should be put in one line):
- FirstThought: First, provide your brief first thoughts and rationales for your action.
- SecondThought: Next, if applicable, examine previous experience (in ‘Experience‘ section), previous trace (in ‘Previous Trace‘ section) and alternative tryings (in ‘Previous Tryings‘ section) from the current state to see if you are repeating previously tried actions. If so, revise your decision and try other options.
- Thought: Next, provide a clean version of your final thoughts. In this field, do NOT explicitly mention previous experience (in ‘Experience‘ section) and tryings (in ‘Previous Tryings‘ section). This field should be used to train models to make good decisions directly. However, still reflect on the steps listed in ‘Previous Trace‘ to enhance the model’s reflection ability.
- Action: Finally, directly output the next action you choose to take. Remember to only issue one action and strictly follow the required formats.

CRITIQUE MODULE

You are an assistant responsible for evaluating the actions of an intelligent agent navigating a web browser to accomplish specific web-based tasks. Your goal is to assess the agent’s latest action and provide constructive feedback.

Available Information
- Target Task: The specific web-based task the agent aims to complete.
- Previous Trace Review: A summary of the agent’s past actions and decisions.
- Previous Observation: A simplified representation of the previous webpage before the action (previous web page’s accessibility tree).
- Action to Evaluate: The current step of decision to be evaluated, which transforms the previous web page to the current one.
- New Observation: A simplified representation of the new webpage after executing the action (current web page’s accessibility tree).

Evaluator Actions
After evaluation, guide the agent by choosing one of the following actions:
- continue: Proceed with the current state.
- back: Revert to a previous state.

Guidelines
- **Observation and Comparison**:
  - Provide brief descriptions of the current accessibility tree.
  - Compare the differences between previous and new observations to assess the effects of the agent’s actions.
- **Details**: Record important details from the current accessibility tree for future reference.
  - Due to space constraints, the accessibility trees are not stored for later steps. Therefore, key information helpful for solving the task should be noted within these fields to avoid loss.
  - If there is no useful information in the current webpage, simply fill in "No useful information".
- **Critic**: Evaluate the effectiveness of the current action.
  - You are evaluating an intermediate step, which may be partial for the full task. Focus on whether the step and the collected information are helpful towards completing the task, rather than whether the job is finished. The main goal is to correct cases where no or little information is found.
  - Carefully check the history to see if the agent is repeatedly performing bad actions. Avoid repeating actions that have already been tried and found unproductive.
- **Scoring Criteria**: Notice that we are only evaluating the current step, which is allowed to be an intermediate and partial step towards the full task.
  - score = 1.0: You find that the status of the task is stuck or in an erroneous state, and you need to adjust the direction of your planning and action or revert to a previous state.
  - score = 3.0: You find that the current step is reasonable or promising towards completing the target task.
  - score = 5.0: You find that the current step is a very critical and successful intermediate step to complete this task.

- **Action**: Decide whether we should continue with the current state or go back to a previous state.
  - Notice that the ‘back‘ action should be chosen cautiously, which will bring extra costs; it should be used ONLY when you are sure that the current state is not promising for any meaningful progress.
  - At many times, you may encounter a state when there are no immediate positive feedback, but if you think it can still lead to promising future states, you are encouraged to try a few more steps (such as scrolling down multiple times).
  - However, if you still do not obtain reasonable results after several attempts, then consider reverting to a previous state.

Target Task
{Target task description}

Previous Trace Review
{Previous trajectories}

Previous Observation
{The accessibility tree for the previous web page before the action}

Action to Evaluate
{The current step of decision to be evaluated}

New Observation
{The accessibility tree for the new web page after the action}

Output
Please generate your response, your reply should strictly follow the format (each item should be put in one line):
- Observation: First, carefully compare the web pages before and after the action (‘Previous Observation‘ and ‘New Observation‘) and briefly describe what are the changes that the action brings.
- Details: Next, briefly record important information from the current accessibility tree for later reference.
- Critic: Next, provide your thoughts and evaluations for the action and the current state.
- Score: Next, rate a float score ranging from 1.0 to 5.0 for the action, in increments of 0.5.
- Action: Finally, select one action from the options provided: ‘continue‘ or ‘back‘.

ROLLBACK MODULE

You are an assistant helping to browse and operate web pages to solve a specific task. In this step, you find that you are stuck or in a bad state and will need to revert back to a previous step. Please read the relevant information and provide your decisions.

Available Information
- Target Task: The target task you are going to solve.
- Previous Trace: All thoughts, actions and observations in previous steps. You should select one from these steps to revert back.
- Most Recent Action: The most recent action that you just take.
- Critic: Feedback on the latest action, explaining why a revert is necessary.

Guidelines
- Analyze the current state and previous trace to decide which previous step to revert back.
- Provide the index of the previous step to return to, the index of each step can be found in the ‘Previous Trace‘ section (annotated with angle brackets "<>").
- Summarize the experiences and lessons learned from the navigation trace between the current state and the returning point. This summary should be self-contained and informative for future decisions. (There is no need to describe the learning of the reverting mechanism.)

Target Task
{Target task description}

Previous Trace
{A full list of previous trajectories with step indexes}

Most Recent Action
{The most recent action}

Critic
{Feedback on the latest action}

Output
Please generate your response, your reply should strictly follow the format (each item should be put in one line):

- Analysis: First, provide your analysis and thoughts for your decisions.
- BackIdx: Next, Provide the "<index>" of the step to return; you can ONLY go back to the steps annotated with "[Checkpoint Saved]".
- Experience: Finally, provide a brief, self-contained summary of experiences and lessons learned from the navigation trace between the current state and the returning point.

B Evaluation Prompt

EVALUATION

You are an assistant tasked with evaluating a web-agent's navigation trace and response to a user's query.

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

- Task: A clear and specific directive provided in natural language, detailing the online activity to be carried out. These requirements may include conducting searches, verifying information, comparing prices, checking availability, or any other action relevant to the specified web service (such as Amazon, Apple, ArXiv, BBC News, Booking, etc).
- Navigation Trajectory: A series of webpage representations and actions showing the trajectory of performing a web task. It serves as a proof of the actions taken in response to the instruction.
- Predicted Response: The predicted textual answer after the navigation process.
- Gold Information (Optional): Reference information for your evaluation (such as a successful trajectory or reference response). Sometimes the gold information may be unavailable (N/A).

- You DO NOT NEED to interact with web pages or perform actions such as booking flights or conducting searches on websites.
- You SHOULD NOT make assumptions based on information not presented in the accessibility tree when comparing it to the instructions.
- Your primary responsibility is to conduct a thorough assessment of the web task instruction against the outcome depicted in the accessibility tree and in the response, evaluating whether the actions taken align with the given instructions.
- NOTE that the instruction may involve more than one task, for example, locating the garage and summarizing the review. Failing to complete either task, such as not providing a summary, should be considered unsuccessful.
- NOTE that the accessibility tree is authentic, but the response provided by LLM is generated at the end of web browsing, and there may be discrepancies between the response and the accessibility tree.
- Note the difference: 1) Result response may contradict the accessibility tree, then the content of the accessibility tree prevails, 2) The content in the Result response is not mentioned on the accessibility tree, choose to believe the content.

Here is the Task: {orig_query}
Here is the Navigation Trajectory: {str_trajectory}
Here is the Predicted Response: {pred_answer}
Here is the Gold Information: {gold_info}
I'll repeat the target task: "{orig_query}", and the starting URL of the navigation is "{target_url}".

Please evaluate the navigation trace for completing this target task. Your output should strictly follow this format for your evaluation:

- Summary: Give a summary of the navigation trajectory.
- Thought: Provide a brief summary of your thoughts and rationale for the output in one concise line.
- Criteria: Examine the target task and list important evaluation criteria (crucial sub-steps or information to collect towards completing the target task). Also indicate whether each criterion has been met within the navigation trace and the predicted response.
- Score: Assign ONE overall float score between 0.0 and 1.0 for the navigation trace and the predicted response. A score of 1.0 indicates full correctness (task completed), 0.0 indicates total failure (no useful information or meaningful sub-steps), and a score in between reflects partial correctness. The score should be assigned based on the criteria analysis. Note that acknowledging failures should not contribute to an increase in the score.

C Detailed Settings

We conduct our evaluations using a subset of the testing portion of Mind2Web-Live[6] and WebVoyager[7]. Due to network connection problems and anti-scraping techniques employed by target websites, we are unable to establish stable connections to certain target websites in the test sets. Therefore, we first perform a connection testing and filter out the websites to which we have trouble connecting. Here is a list of the

[6] https://huggingface.co/datasets/iMeanAI/Mind2Web-Live/blob/main/mind2web-live_test_20241024.json
[7] https://github.com/MinorJerry/WebVoyager/blob/main/data/WebVoyager_data.jsonl

websites that are excluded:

EXCLUDED WEBSITES

```
EXCLUDED_WEBSITES = { 'exploretock', 'kohls', 'united', 'parking', 'viator', 'delta',
'redbox', 'soundcloud', 'gamestop', 'travelzoo', 'amctheatres', 'ryanair', 'cargurus',
'resy', 'rentalcars', 'kbb', 'cabelas', 'menards', 'yellowpages', 'tripadvisor',
'tiktok.music', 'stubhub', 'thumbtack', 'weather', 'uhaul', 'health.usnews', 'healthgrades',
'theweathernetwork', 'zocdoc', 'usnews.education', 'epicurious', 'osu.edu', 'ups',
'dmv.virginia.gov', 'extraspace', 'finance.yahoo', 'pinterest', 'sixflags', 'spothero',
'justice.gov', 'foxsports', 'ign', 'koa', 'tvguide', 'webmd', 'sports.yahoo', 'babycenter',
'tesla', 'booking', 'dictionary.cambridge.org', 'espn', 'amazon', 'google', 'github',
'allrecipes', }
```

After applying the filtering process, we have 42 and 342 testing instances remaining for Mind2Web-Live and WebVoyager, respectively. For the training datasets for fine-tuning, we adopt the training portion of Mind2Web-Live[8] and the imitation learning training data from OpenWebVoyager.[9] After applying a similar filtering procedure, the final training set contains a total of 455 instances.

For fine-tuning, we adopt a standard full-model supervised fine-tuning (SFT) recipe using OpenRLHF (Hu et al., 2024).[10] A single model is trained in a multi-task way to support different modules by using their corresponding prompts.

D Extra Results

	Mind2WebLive				Webvoyager			
	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)
Llama3.1-8B-Instruct								
OneWay	$0.63_{\pm 0.89}$	$19.35_{\pm 0.87}$	$15.9_{\pm 0.1}$	0	$3.03_{\pm 1.59}$	$24.90_{\pm 0.77}$	$15.9_{\pm 0.1}$	0
BestFirst	$0.63_{\pm 0.89}$	$22.08_{\pm 1.92}$	$16.0_{\pm 0.0}$	$9.1_{\pm 0.5}$	$3.13_{\pm 0.13}$	$29.91_{\pm 1.25}$	$15.9_{\pm 0.1}$	$6.6_{\pm 0.2}$
Rollback	$0.63_{\pm 0.89}$	$22.14_{\pm 0.54}$	$16.0_{\pm 0.0}$	$8.3_{\pm 0.2}$	$3.23_{\pm 0.25}$	$28.66_{\pm 1.55}$	$15.9_{\pm 0.0}$	$5.6_{\pm 0.3}$
Qwen2.5-7B-Instruct								
OneWay	$3.77_{\pm 2.67}$	$11.89_{\pm 2.80}$	$15.5_{\pm 0.4}$	0	$11.79_{\pm 0.60}$	$22.27_{\pm 0.30}$	$14.5_{\pm 0.1}$	0
BestFirst	$1.26_{\pm 1.78}$	$12.23_{\pm 1.19}$	$15.8_{\pm 0.1}$	$12.5_{\pm 0.7}$	$13.16_{\pm 1.49}$	$26.13_{\pm 1.59}$	$14.5_{\pm 0.1}$	$9.4_{\pm 0.2}$
Rollback	$5.03_{\pm 2.35}$	$14.91_{\pm 2.89}$	$15.4_{\pm 0.2}$	$12.0_{\pm 0.2}$	$12.98_{\pm 1.36}$	$26.24_{\pm 1.25}$	$14.4_{\pm 0.2}$	$8.5_{\pm 0.1}$

Table 3: Zero-shot results with LLAMA-3.1-8B-INSTRUCT and QWEN2.5-7B-INSTRUCT. The performance is much worse with these "smaller" LMs if not fine-tuned, when compared to the 70B-level ones.

	Full%(↑)	Partial%(↑)	Step(↓)	Switch(↓)
GPT-4o-mini				
OneWay	$29.50_{\pm 0.50}$	$49.57_{\pm 0.62}$	$13.0_{\pm 0.1}$	0
BestFirst	$28.15_{\pm 1.15}$	$51.19_{\pm 1.44}$	$13.2_{\pm 0.1}$	$7.8_{\pm 0.3}$
Rollback	$34.85_{\pm 3.54}$	$56.82_{\pm 2.32}$	$12.8_{\pm 0.1}$	$6.3_{\pm 0.3}$
GPT-4o				
OneWay	$38.50_{\pm 0.50}$	$55.17_{\pm 1.57}$	$11.5_{\pm 0.1}$	0
BestFirst	$32.83_{\pm 1.52}$	$55.73_{\pm 0.93}$	$12.4_{\pm 0.1}$	$7.0_{\pm 0.2}$
Rollback	$43.50_{\pm 1.50}$	$60.00_{\pm 1.30}$	$11.6_{\pm 0.2}$	$3.5_{\pm 0.3}$

Table 4: Zero-shot results (on a 100 randomly selected subset) with GPT-4O-MINI and GPT-4O. The overall trends are similar to those with open-source models.

[8] https://huggingface.co/datasets/iMeanAI/Mind2Web-Live/blob/main/mind2web-live_train_20240528.json
[9] https://github.com/MinorJerry/OpenWebVoyager/tree/main/WebVoyager/data_for_training/IL
[10] https://github.com/OpenRLHF/OpenRLHF/blob/main/examples/scripts/train_sft_llama.sh