WebEvolver: Enhancing Web Agent Self-Improvement with Co-evolving World Model

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu,
Haitao Mi, Dong Yu
Tencent AI Lab
tianqfang@tencent.com

Abstract

Agent self-improvement, where agents autonomously train their underlying Large Language Model (LLM) on self-sampled trajectories, shows promising results but often stagnates in web environments due to limited exploration and under-utilization of pretrained web knowledge. To improve the performance of self-improvement, we propose a novel framework that introduces a co-evolving World Model LLM. This world model predicts the next observation based on the current observation and action within the web environment. The World Model serves dual roles: (1) as a virtual web server generating self-instructed training data to continuously refine the agent's policy, and (2) as an imagination engine during inference, enabling look-ahead simulation to guide action selection for the agent LLM. Experiments in real-world web environments (Mind2Web-Live, WebVoyager, and GAIA-web) show a 10% performance gain over existing self-evolving agents, demonstrating the efficacy and generalizability of our approach, without using any distillation from more powerful close-sourced models[1].

1 Introduction

Autonomous agents, especially Web agents operating in online environments, play a crucial role in automating complex tasks, advancing progress towards artificial general intelligence (OpenAI, 2025; Monica.Im, 2025; Qin et al., 2025; Liang et al., 2025). The capabilities of these agents stem from two key components, the design of the system, which facilitates accessing and processing abundant information from the web, and the agent foundation language model itself, which is typically a (Multimodal) Large Language Model (LLM) that generates actions based on the provide context.

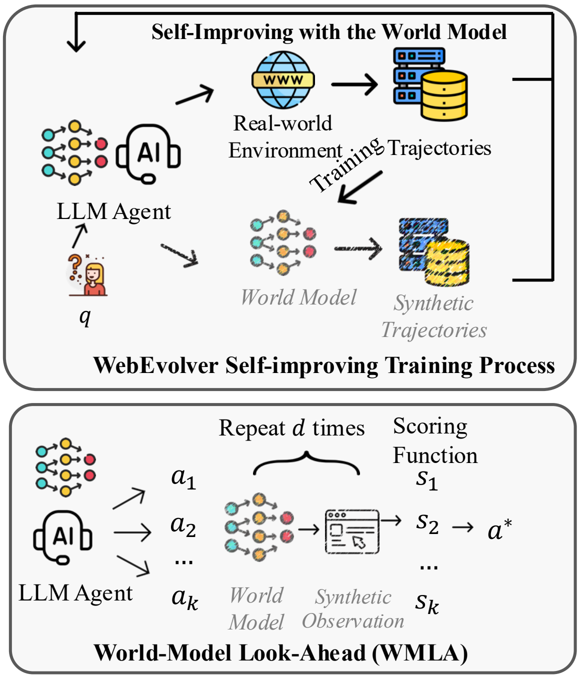


Figure 1: Overview of WebEvolver – A Self-Improving Framework with World-Model Look-Ahead. Our framework co-trains a world model with the agent to predict next-step observations based on current states and actions. The world model serves as a virtual web engine, which generates synthetic trajectories for policy training and enables look-ahead planning to select optimal actions during inference.

Recent work in **agent self-improvement** refines LLM-based agents through iterative cycles of autonomous interaction: agents generate actions, collect behavioral trajectories, and are fine-tuned on this self-collected data after rejection sampling (Yin et al., 2024; Murty et al., 2024; Patel et al., 2024; Aksitov et al., 2023; He et al., 2024b; Xi et al., 2024). While this bootstrapping reduces reliance on human-labeled data, performance eventually plateaus (Zeng et al., 2024).

This stagnation arises from two main bottlenecks. First, exploration diversity declines as the agent overfits to familiar trajectories, limiting discovery

[1] Code is available at https://github.com/Tencent/SelfEvolvingAgent

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 8959–8975
November 4-9, 2025 ©2025 Association for Computational Linguistics

of novel states (He et al., 2024b). Second, although inference-time exploration methods (Koh et al., 2024b; Zhang et al., 2024b; Zhou et al., 2024a; Putta et al., 2024; Yu et al., 2024) have the potential to provide diverse trajectories, they require costly real-world interactions for marginal gains. On the other hand, simulation or imagination-based approaches (Gu et al., 2024; Qiao et al., 2024) typically offer only one/two-step look-ahead, lacking coherent multi-step rollouts.

To address these limitations, we propose integrating a **Co-evolving World Model** into the self-improvement loop to enable better multi-step trajectory synthesis and look-ahead. Our world model is a language model trained to predict the next observation (web page) given the current state and an attempted action. Our key insight is that LLMs, pretrained on vast web content (e.g., Llama-3; Dubey et al., 2024), inherently encode a structured understanding of website dynamics, user intents, and task workflows. We fine-tune it on trajectories collected during agent-environment interactions, allowing it to evolve alongside the agent to provide better simulation results.

As a *virtual web server*, The World Model serves two roles : (1) it generates diverse, self-instructed training trajectories by simulating interactions with unseen web environments, mitigating exploration bottlenecks by exposing the agent to a wider range of scenarios than real interactions alone. While the World Model may produce some hallucinated (i.e., non-realistic) web states, this is not critical during training, as the agent's goal is to learn flexible action prediction, even under noisy circumstances. (2) during inference, the World Model performs multi-step look-ahead simulations (Zhang et al., 2025a), allowing the agent to evaluate possible actions without costly real-world trials. This dual mechanism grounds self-improvement in both real and model-based interactions, ensuring sustained adaptability while reducing reliance on expensive environment interactions.

We validate our framework on real-world, open-domain web environments, including Mind2Web-Live (Pan et al., 2024), WebVoyager (He et al., 2024a), GAIA-web (Mialon et al., 2024), and SimpleQA (Wei et al., 2024)[2]. Experiments show a 10% performance improvement over the self-evolving baseline OpenWebVoyager (He et al., 2024b), with notable gains on complex and unseen tasks.

Our main contributions are:

1. Introducing the co-evolving world model for self-improving web agents, enabling diverse training data generation and low-cost multi-step action search.

2. Providing empirical evidence that world-model-guided self-improvement enhances agent performance and adaptability in open-domain settings, with minimal human supervision and no distillation from stronger LLMs.

This work highlights the importance of integrating dynamic world models into agent frameworks to overcome the limitations of purely data-driven self-training.

2 Related Work

Web Agent Recent advances in web agents leverage (multimodal) large language models as their backbone (Dubey et al., 2024; Jia et al., 2024; OpenAI, 2023; Anthropic, 2025), enabling reasoning through frameworks like ReAct (Yao et al., 2023), MCP (Anthropic, 2024), and cognitive kernel (Zhang et al., 2024a). These agents are evaluated on benchmarks such as WebShop (Yao et al., 2022), Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2024b), VisualWebArena (Koh et al., 2024a), WebVoyager (He et al., 2024a), WebWalker (Wu et al., 2025b), and MMInA (Zhang et al., 2024c). Besides applying off-the-shelf LLMs, there are data scaling efforts like Explorer (Pahuja et al., 2025), NNetNav (Murty et al., 2025), and InSTA (Trabucco et al., 2025) enhance the training of LLMs. Inference-time optimization techniques, including AgentTreeSearch (Koh et al., 2024b), Monte-Carlo Tree Search (Putta et al., 2024; Yu et al., 2024; Zhou et al., 2024a; Zhang et al., 2024b), and Reflexion (Shinn et al., 2023), further improve decision-making. More recently, the development of web agents focus on multi-step Deep Research agent and the training of agent foundation models, such as WebThinker (Li et al., 2025b), WebDancer (Wu et al., 2025a), WebSailor (Li et al., 2025a), WebShaper (Tao et al., 2025), Cognitive Kernel-Pro (Fang et al., 2025), MiroFlow (MiroMindAI, 2025), and so on.

Agent Self-Improvement Beyond using off-the-shelf LLMs as policy models or fine-tuning

[2] We adapt this dataset to search queries on the internet

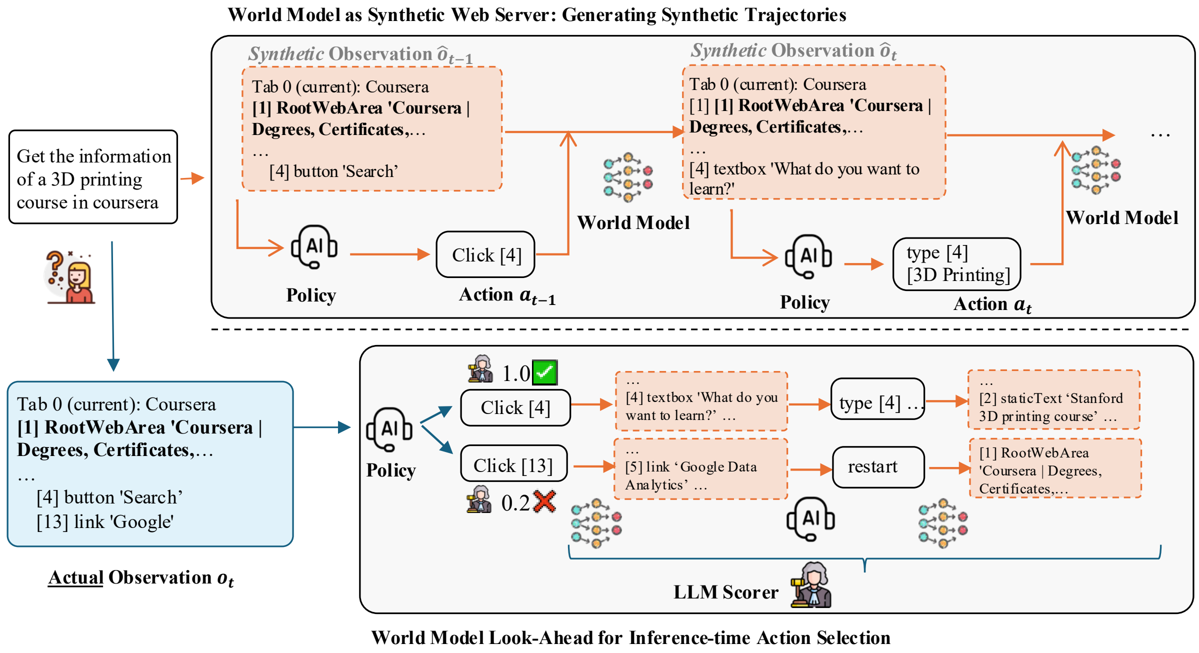

World Model as Synthetic Web Server: Generating Synthetic Trajectories
Synthetic Observation $\hat{o}_{t-1}$
Tab 0 (current): Coursera
[1] RootWebArea 'Coursera | Degrees, Certificates,...
...
[4] button 'Search'
Synthetic Observation $\hat{o}_t$
Tab 0 (current): Coursera
[1] [1] RootWebArea 'Coursera | Degrees, Certificates,...
...
[4] textbox 'What do you want to learn?'
...
Get the information of a 3D printing course in coursera
Policy
Click [4]
Action a_{t-1}
World Model
Policy
type [4] [3D Printing]
Action a_t
World Model
Tab 0 (current): Coursera
[1] RootWebArea 'Coursera | Degrees, Certificates,...
...
[4] button 'Search'
[13] link 'Google'
Actual Observation o_t
Policy
1.0 ✓
Click [4]
Click [13]
0.2 ✗
... [4] textbox 'What do you want to learn?' ...
type [4] ..
... [2] staticText 'Stanford 3D printing course' ...
... [5] link 'Google Data Analytics' ...
restart
[1] RootWebArea 'Coursera | Degrees, Certificates,...
LLM Scorer
World Model Look-Ahead for Inference-time Action Selection

Figure 2: An illustration of the World Model trajectory synthesizing process and World Model Look-ahead for inference-time action selection.

via imitation learning from powerful LLM trajectories, recent work explores bootstrapping agent LLMs with open-source models (Aksitov et al., 2023; Patel et al., 2024), building on advances in self-improving LLM reasoning (Wang et al., 2023; Zelikman et al., 2022; Zeng et al., 2024). BAGEL (Murty et al., 2024), OpenWebVoyager (He et al., 2024b), and Self-Improved Agents (Patel et al., 2024) explored iterative exploration-feedback-optimization cycles, where agents refine their policies by learning from high-quality trajectories in real-world or simulated web environments. To enhance self-improvement, G"odel Agent (Yin et al., 2024) enables agents to dynamically modify their logic and accumulate skills across diverse tasks. WebCoT (Hu et al., 2025) study cognitive behavior injection to the Chain-of-thought of Agent Foundation Model to improve the reasoning ability of the agents. (Zhang et al., 2025b) explores bootstrapping the ability of backtracking in web agent tasks. AgentQ (Putta et al., 2024) and ReST+ReAct (Aksitov et al., 2023) combine reinforcement learning and preference optimization, enabling agents to learn from both successes and failures and improving robustness in multi-step reasoning. While reinforcement learning is promising for self-improvement, real-world, evolving websites pose challenges: environmental uncertainty can lead to inconsistent evaluations of the same action, making it difficult for agents to reliably assess and improve their performance. In addition, while analogous to reinforcement learning (RL), our iterative Supervised Fine-Tuning (SFT) paradigm (with binary trajectory selection) differs from end-to-end RL as it optimizes LLM loss rather than reward functions directly. We collect the agent SFT data first and then use standard LLM SFT framework to optimize the LLM, which is why similar paradigms were termed as agent self-improvement rather than RL.

World Models World models have evolved from their reinforcement learning origins (Ha and Schmidhuber, 2018) to become powerful tools for agent reasoning (Valevski et al., 2024; Alonso et al., 2024; Smith and Wellman, 2023). Recent approaches leverage large language models (LLMs) as implicit world models, enabling agents to simulate and plan through complex tasks. For general reasoning, RAP (Hao et al., 2023) demonstrates how LLMs can serve dual roles as both world models and reasoning agents, using Monte Carlo Tree Search to explore future states. Similarly, WKM (Qiao et al., 2024) shows that structured world knowledge can be distilled from trajectories to guide agent planning. In web environments, methods like WebDreamer (Gu et al., 2024) and WMA (Chae et al., 2025) adapt this paradigm by using LLMs to predict action outcomes through natural language simulations. However, these approaches remain limited by their reliance on off-the-shelf LLMs. In detail, both WebDreamer and WMA works at inference time. Their approaches essentially create sophisticated chain-of-thought pipelines where the world model serves as intermediate information for static GPT-4o. Despite that WMA uses a finetuned world model instead of GPT-4o, eventually the inference-time algorithm is still a prompting pipeline. Neither method actually uses the world model to improve the agent itself through training. Besides, despite WMA also has multi-step simulation (depth up to 3), it is only used for inference-time action selection. Instead, our approach generate trajectories using world model with depth up to 7, and the full trajectories are used for training instead of only for prompting to select best actions.

Our work advances beyond these limitations by co-learning a dedicated world model during agent self-improvement. This enables genuine multi-step trajectory synthesis and look-ahead planning, providing a more robust foundation for interactive decision-making than current prompt-based approaches.

3 Method

In this section, we introduce the WebEvolver, a co-learning framework of World Model and Agent Policy model (Figure 2).

3.1 Problem Formulation

The web agent task is formulated as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$, where the agent receives a natural language query q requiring multi-step web interaction under the environment. The state space $\mathcal{S}$ represents the complete web environment, while the observation space $\mathcal{O}$ is limited to visible elements. At each time step t: $o_t = \Omega(s_t)$, where Ω is a function extracting visible contents like (URL, Web Elements) from the current state s_t. $\mathcal{A}$ represents the whole action space, which, in our case we include `click`, `type`, `goback`, `scroll down/up`, and `stop`, as the atomic web operations. $\mathcal{T}$ represents the deterministic transition function that executes browser operations to advance the state. The agent's policy $\pi(o_t, q) \rightarrow a_t$ generates actions that produce trajectories $\tau = \{(o_1, a_1), \dots, (o_t, a_t)\}$, with final rewards computed through self-assessment $\hat{r}(\tau, q) \in [0, 1]$.

Given a task query q and target website w, we initialize the web environment and get the first observation $o_1 \in \mathcal{O}$. We follow the settings in Cognitive Kernel (Zhang et al., 2024a) and use accessibility tree to represent the elements in o_t. Using an LLM as agent policy model parameterized by θ, we generate chain-of-thoughts h_t and actions a_t at time step t:

$$(h_t, a_t) \sim \pi_\theta(\cdot | I, q, o_{1:t}, h_{1:t-1}, a_{1:t-1}) \quad (1)$$

where I contains system instructions. The transition function $\mathcal{T}$ executes actions on the environment:

$$s_{t+1} = \mathcal{T}(s_t, a_t),\ o_{t+1} = \Omega(s_{t+1}) \quad (2)$$

The complete trajectory is $\tau = (o_1, h_1, a_1, \dots, o_T, h_T, a_T)$, where T denotes the total number of navigation steps.

3.2 Agent Self-Improvement

In this subsection, we introduce the self-improvement of a backbone agent foundation model, denoted as $\mathcal{M}$, and the corresponding policy function is denoted as $\pi_{\mathcal{M}}$.

Trajectories Collection We employ $\mathcal{M}$ to sample actions based on an input query q, which are then used to collect web navigation trajectories. We use $\mathcal{M}$ as the agent foundation model to power Cognitive Kernel, which interacts with web environments. The agent observes the last k steps, represented as webpage accessibility trees, to inform its actions.

For each query $q \in \mathcal{Q}$, a trajectory τ_i is sampled from the policy $\pi_{\theta_M}(\tau \mid I, q)$. To prevent performance degradation from too long contexts, we clip the trajectory history c_t when $t - 1 > k$ by keeping only the latest observations. The thoughts and actions are kept as they contain some compressed information about the history.

$$\begin{aligned} c_t^{\text{clip}} =& (h_1, a_1, h_2, a_2, \dots, h_{t-k}, a_{t-k}, \\ & o_{t-k+1}, h_{t-k+1}, a_{t-k+1}, \dots, o_{t-1}), \end{aligned} \quad (3)$$

such that the new actions are generated with the following function:

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

Notably, we retain the **thought** and **action** at each step to preserve the full reasoning chain while avoiding context overload. Then, rejection sampling is conducted to keep those trajectories that are successfully finished, using an automatic evaluation method $\hat{r}(\tau, q)$.

Iterative Optimization At the i-th iteration of the self-improvement, we denote the collected trajectories after rejection sampling as D_i. We aim to maximize the following objective function:

$$\mathcal{J}(\theta) = \mathbb{E}_{(q,\tau)\sim D_{\text{i}}} \sum_{t=1}^{T} \Big[\log \pi_\theta(a_t | q, c_t^{\text{clip}'}, h_t) + \log \pi_\theta(h_t | q, c_t^{\text{clip}'}) \Big], \tag{5}$$

After acquiring the new policy model $\mathcal{M}_i$, it is used to sample trajectories from the query set $\mathcal{Q}$ again. The newly successful trajectories are then appended to D_i to form a new training dataset D_{i+1} to perform the next round of optimization.

3.3 WebEvolver

In this subsection we introduce the co-learning/co-training world model, and how to use it for trajectory synthesizing and inference-time look-ahead. An illustration figure is presented in Figure 2.

Co-learning World Model The world model is a language model that simulates the next observation $\hat{o}_{t+1}$ conditioned on both the current webpage's accessibility tree (o_t) and a formatted action string (a_{t-1}), thereby predicting state transitions. We learn a world model LLM $\mathcal{M}_w$ using the collected trajectory during self-improvement.

From the a collected trajectory $\tau = \{(o_0, a_0), \ldots, (o_t, a_t)\}$, we can convert it to a world modeling trajectory $\tau_w = \{o_0, (a_0, o_1), \ldots, (a_{t-1}, o_t)\}$, such that the objective of world model is to predict the next observation o_t conditioned on the scheduled action a_{t-1} and previous observations. Similar with the trajectories in agent policy model, we truncate the history observations to avoid performance degrade on long contexts. Here, we simply use the latest observation as history. Besides, we distill some rationales using the original base LLM $\mathcal{M}$ about the logic of the transition function $\mathcal{T}$ to help the generation of the next webpage. Such chain-of-thoughts at step t is denoted as h_t^w. We do not omit the action and thoughts to make the world model aware of some of the previous information and the depth of the trajectory.

$$c_t^w = (a_1, h_1^w, \ldots, a_{t-2}, h_{t-2}^w, o_{t-1}, a_{t-1}), \tag{6}$$

Such that the next webpage observation o_t is generated with the following function, where θ_w is the parameters of $\mathcal{M}_w$.

$$o_t \sim \pi_{\theta_w}(\cdot | I_w, c_t^w) \tag{7}$$

The world model is then optimized using the latest iteration of collected trajectories.

$$\mathcal{J}(\theta_w) = \mathbb{E}_{\tau_w \sim D_{\text{i}}} \sum_{t=1}^{T} \Big[\log \pi_{\theta_w}(o_t | c_t^w, h_t^w) + \log \pi_{\theta_w}(h_t^w | c_t^w) \Big], \tag{8}$$

Trajectory Synthesis We can use an agent policy model M_i and a world model M_w to perform synthetic trajectory generation, enabling us to scale up the training data without interacting with the real web server, which can be very costly. Here, we directly replace the transition function $\mathcal{T}$ with the world model M_w. Specifically, the next synthetic observation is generated with:

$$\hat{o}^t \sim \pi_{\theta_w}(\cdot | I_w, c_t^w) \tag{9}$$

Then, in the next step, the policy model generates next action conditioned on the synthetic observation:

$$(\hat{h}_t, \hat{a}_t) \sim \pi_{\theta_M}(\cdot \mid I, q, \hat{c}_t^{\text{clip}}). \tag{10}$$

Those collected trajectory is thus $\hat{\tau} = \{(o_0, a_0), (\hat{o}_1, \hat{a}_1), \ldots, (\hat{o}_t, \hat{a}_t)\}$, which ultimately forms a trajectory dataset D_w after rejection sampling. By combining D_i from self-improvement and D_w, we can get an augmented new training dataset to train a new policy model, WebEvolver.

Inference-time Look-ahead To enhance the planning ability during inference, we propose a look-ahead mechanism that simulates d-step trajectories using both the agent policy model M_i and the world model M_w. We call this method **W**orld **M**odel **L**ook-**A**head (WMLA). For each candidate action a_t at step t, we first simulate trajectories by generating d-step rollouts $\hat{\tau}_w$ through iterative application of:

$$\begin{aligned} \hat{o}_{t+j} &\sim \pi_{\theta_w}(\cdot | I_w, c_{t+j}^w), \\ (\hat{h}_{t+j}, \hat{a}_{t+j}) &\sim \pi_{\theta_M}(\cdot | I, q, \hat{c}_{t+j}^{\text{clip}}), \end{aligned} \tag{11}$$

where $j \in \{1, \ldots, d\}$, c^{w}_{t+j} and $\hat{c}^{\text{clip}}_{t+j}$ are truncated histories from the world model and policy model, respectively.

Next, we evaluate trajectories by employing an LLM-based evaluator to score each rollout $\hat{\tau}_w$. Following Koh et al. (2024b); Gu et al. (2024), the evaluator assigns a scalar from $\{0, 0.5, 1.0\}$ (incorrect, on track, or complete) based on the trajectory's alignment with task completion. Finally, we select the optimal action $a_t^* = \arg\max_{a_t} \text{Score}(a_t)$ that maximizes expected progress.

4 Experiments

4.1 Setup

We use the Cognitive Kernel (Zhang et al., 2024a) as the foundation agent framework, specifically its Web Agent Module for autonomous Web interaction. Here, the state space $\mathcal{S}$ is the whole Internet, powered by Playwright[3] in the Web docker in Cognitive Kernel. The action space include `type`, `click`, `scroll`, `goback`, `stop`, and `restart`. At each time step t, the observation o_t is the accessibility tree of the visible components in the virtual browser, simulating what humans can perceive when browsing online. The transition function $\mathcal{T}$ executes atomic browser actions based on the current webpage state, updates the webpage, and thus the observation accordingly, and handles execution errors by feeding them back to the reasoning system until task completion or step limit is reached. Regarding the evaluation protocol $\mathcal{R}$, we address potential false negatives in human-annotated stepwise comparisons (Pan et al., 2024) by employing GPT-4o for end-to-end task completion assessment, following the methodology of He et al. (2024a). This method accommodates the existence of multiple distinct trajectories that can each successfully accomplish the same task objective, other than the human-annotated ones. GPT-4o will be provided the full trajectory of the task and asked to evaluate whether the original query q is completed or not, yielding a binary score of 0 or 1.

Regarding self-improvement, the backbone agent foundation model $\mathcal{M}$ we use is `Llama-3.3-70b`, and subsequently the self-improving experiments are also based on `Llama-3.3-70b`. During rejection sampling, `Llama-3.3-70b` instead of GPT-4o is used to evaluate whether the task has successfully completed or not. More details regarding the agent system, including definitions of the atomic operations, system prompts, are detailed in Appendix A.

We select two live web navigation benchmarks for experiments, WebVoyager (He et al., 2024a) and Mind2Web-Live (Pan et al., 2024). Here, the web agent is expected to interact with the real-world web environment to complete the task. Since some websites are not accessible in our experimental web environment, either due to geographical locations or IP blocks, we filter out some websites for our experiments[4]. To ensure robustness, we conduct our experiments roughly at the same time window twice and report the average results.

4.2 Self-Improvement

We use `Llama3.3-70B` as the backbone LLM $\mathcal{M}$ for sampling and self-improving. For the training query, we follow OpenWebVoyager (He et al., 2024b)[5] to use the training set of Mind2web and self-instructed queries from both the websites in WebVoyager and Mind2web, in total 1,516 queries. We first use `Llama3.3-70B` as the backbone agent policy model for sampling queries, and conduct a round of rejection sampling using `Llama3.3-70B` itself as the backbone for evaluation function $\hat{r}$[6], using the evaluation prompt in Appendix A. The trajectories are then used to fine-tune `Llama3.3-70B` to acquire the model named *self-improve (iter 1)*. Then, we use the improved model to conduct another round of trajectory sampling, where the newly sampled finished trajectories are added to the training data in the first round, to train a new model named *self-improve (iter 2)*. In the meantime, we convert the trajectories to the form of training a world model, meaning predicting the next observation o_t based on the scheduled observation a_{t-1} and the histories of the observations.

World Model We adopt a `Llama3.3-70B` to fine-tune the world model, alongside the self-improving of policy model, to get *world model (iter 1)* and *world model (iter 2)*. For synthetic trajectory generation, we use the world model M_w (at iteration 2) and policy model M_1 (at iteration 1, which has a better performance). For each query q, beginning with an initial observation-action pair (o_0, a_0), we

[3] A Javascript version `https://playwright.dev`

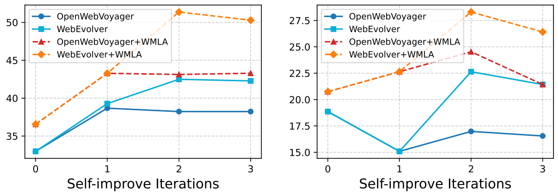
[4] Details about the websites are presented in Appendix B

[5] `https://github.com/MinorJerry/OpenWebVoyager/tree/main/WebVoyager/data_for_training/IL`

[6] In the original OpenWebVoyager paper, GPT-4o serves as the backbone for the scoring function. In this work, to ensure a purely self-improving process, we only employ `Llama3-70B` within the self-improvement loop.

	AllRe-cipes	Apple	ArXiv	BBC	Cam Dict	Cour-sera	ESPN	Git Hub	Google Map	HF	Wolfram Alpha	**WV All**	**M2W Live**
GPT-4o-mini	44.44	39.53	23.26	21.43	30.23	35.71	27.27	31.71	41.46	25.58	36.96	32.55	16.98
GPT-4o	31.11	41.86	27.91	32.56	41.86	47.62	27.27	36.59	36.58	46.51	56.52	38.83	20.75
Self-Improving													
Llama-3.3 70B	35.56	39.53	9.30	28.57	37.21	38.10	50.00	24.39	34.15	23.26	41.30	32.98	18.86
self-improve (1)	55.56	39.53	27.91	45.24	20.93	61.90	34.09	39.02	39.02	23.26	39.13	38.68	15.09
self-improve (2)	40.00	30.23	27.91	30.95	32.56	59.52	29.55	43.90	46.34	41.46	39.13	38.23	16.98
self-improve (3)	44.44	30.23	32.25	33.33	32.56	47.62	31.81	43.90	48.78	34.89	45.65	38.65	16.98
Synthetic Traj.	55.56	41.86	32.25	35.71	34.89	46.51	31.81	34.14	36.59	34.89	43.47	38.98	18.86
WebEvolver	62.22	30.23	37.21	47.62	53.49	59.52	34.09	26.83	46.34	23.26	45.65	42.49	22.64
Inference-time Look-ahead													
+ WebDreamer	64.44	41.86	44.19	**57.14**	30.23	59.52	20.45	41.46	46.34	41.86	43.48	44.61	22.64
+ **WMLA** (d=1)	**66.67**	**46.51**	39.53	42.86	32.56	**69.05**	22.73	43.90	**68.29**	37.21	41.46	46.24	**28.30**
+ **WMLA** (d=2)	64.44	41.86	**46.51**	42.86	**62.79**	66.67	**40.91**	**46.34**	43.90	**53.49**	**54.34**	**51.37**	24.53

Table 1: Task success rate on Text-only WebVoyager test set (WV; 473 queries) and Mind2Web-Live-filtered test set (M2W Live; 53 queries). **WebEvolver** and **WMLA** are our approaches. For *Inference-time Look-ahead*, the backbone policy model we use is WebEvolver. We leave more inference-time look-ahead results on different policy models in Figure 3. Underline indicates the best among self-improving, and **bold** indicates the best performance when inference-look ahead is applied.



WebVoyager **Mind2web-Live**

Figure 3: Visual illustration of overall success rate evolving on WebVoyager and Mind2Web-Live.

alternate between world model prediction and policy decisions: at each timestep t, the world model generates the next synthetic observation $\hat{o}_t$ according to Equation (9), which the policy model then uses to produce the subsequent action $\hat{a}_t$ via Equation (10). This interaction forms complete synthetic multi-step trajectories $\hat{\tau}$ of length $T = 7$ steps, with early termination if the world model generates a terminal state. An example if presented in Figure 4. To have a more diverse training set, we only use the queries that are not successfully executed in self-improving iterations to acquire synthetic trajectories. We apply another round of rejection sampling using the evaluation protocol $\mathcal{R}$, while using zero-shot `Llama3.3-70B` as the backbone language model to follow the setting of self-improving. In the end, the world-model-synthesized data are combined with the SFT data in self-improvement, to train `Llama3.3-70B` to acquire the final model of WebEvolver.

4.3 Inference-time World Model Look-ahead (WMLA)

To perform WMLA, we use the policy model $\mathcal{M}$ to sample up to 3 actions. At time step t, with observation o_t, we use the original policy model with temperature equal to 0 to generate the first action, $a_t^{(1)}$. Since the fine-tuned policy model will have a sharp output distribution, making it hard to directly sample different actions during decoding, besides setting the decoding temperature to 0.7, we add a sentence of additional prompt to guide the policy model to generate the k-th action: *Please generate actions different from* $\{a_t^{(j)}, j \in \{1, \ldots, k-1\}\}$. Then, we use the final world model *world model (iter 2)* and the policy agent model to iteratively sample future look-ahead trajectories based on Equation (11), with a look-ahead depth of 1, 2, and 3. Then, following WebDreamer, we use GPT-4o as the scoring function to rate each action based on the look-ahead results and choose the action with the highest score for execution.

4.4 Results and Analysis

In this subsection, we provide results of self-improvements, the effect of WMLA, the intrinsic evaluation of world models, and additional experiments on GAIA.

WebEvolver and WMLA Main Results Our key findings are presented in Table 1, with the progression of self-improvement across iterations visualized in Figure 3. The first two rows of the ta-

Model	All			Depth=1			Depth=2			Depth=3			Depth≥4		
	STR	Sim.	O/A	STR	Sim.	O/A	STR	Sim.	O/A	STR	Sim.	O/A	STR	Sim.	O/A
gpt-4o	40.62	33.26	37.85	41.24	35.73	40.21	38.20	32.58	36.70	36.99	31.96	37.44	42.41	32.91	37.45
Llama-3.3-70b	39.04	32.25	38.77	43.64	39.51	34.83	39.33	34.83	41.95	39.73	33.33	41.55	36.85	27.99	35.16
iter-1	49.23	37.83	43.15	55.44	44.91	50.52	53.03	39.77	46.59	**53.70**	**40.28**	**46.30**	43.76	33.33	37.73
iter-2	**56.79**	**44.77**	**51.82**	**75.96**	**63.56**	**72.86**	**57.80**	**45.14**	**52.32**	51.24	35.82	45.27	**50.54**	**39.94**	**45.31**

Table 2: Performance of intrinsic evaluation of world modeling. **Structural correctness (STR)** measures syntactic validity of the generated accessibility tree, **Similarity (Sim.)** assesses alignment with ground-truth webpage content, and **Overall assessment (O/A)** evaluates functional and semantic coherence. All values are percentages (range 0-100). Details of the evaluation metrics ae presented in Section 4.4.

ble establish reference performance using GPT-4o and GPT-4o-mini as foundation models. In terms of self-improvement, the initial self-improvement iteration yields a 6% success rate increase over the zero-shot baseline on WebVoyager, due to enhanced format compliance and task familiarity. Performance plateaus at iteration 2, suggesting limited gains from additional similar trajectories. However, incorporating world-model-synthesized data with iteration 1's supervised fine-tuning (SFT) data produces a further 4% improvement. This has better improvement compared to the baseline approach adapted from Patel et al. (2024) that generates synthetic trajectories without world modeling.

For inference-time action selection with WebEvolver, we benchmark against WebDreamer using GPT-4o for both outcome prediction and action evaluation. Our World Model-based Look-ahead (WMLA) demonstrates optimal performance at depth $d = 2$, balancing prediction accuracy against computational overhead. Notably, increasing to $d = 3$ provides diminishing returns, consistent with our world model's performance characteristics (see Table 2).

World Model Intrinsic Evaluation We evaluate our world model's ability to generate plausible next webpages through three metrics: **Structural correctness (STR)** measuring syntactic validity of the generated accessibility tree, **Similarity (Sim.)** assessing alignment with ground-truth webpage content, and **Overall assessment (O/A)** evaluating functional and semantic coherence. While real-time information (e.g., from BBC or Hugging Face) inevitably causes hallucinations during generation, we do not directly evaluate the degree of hallucination. Hallucinations are implicitly captured through Sim. and O/A scores, yet they pose minimal risk in our framework. In fact, they may enhance diversity and knowledge in synthesized trajectories, with benefits empirically validated by downstream performance gains. We use GPT-4o to perform an automatic evaluation of all three metrics and normalize the scores to 0$\sim$ 1. The prompt we used is presented in Appendix A. The results are presented in Table 2. We can see that the performance degrades sharply (scores < 0.50) for generation depths > 2, which is in line with the experiments in WMLA that the performance gain diminishes when WMLA depths ≥ 3.

Model	GAIA Level 1	GAIA Level 2	SimpleQA
Llama 3.3-70b	19.2	10.9	36
iter 1	26.9	15.6	44
iter 2	26.9	12.5	45
WebEvolver	30.7	**17.2**	48
+ WMLA	**34.6**	**17.2**	**58**

Table 3: GAIA-web and SimpleQA performance.

Out-of-domain Generalization We evaluate our improved agent foundation model on GAIA (Mialon et al., 2024), focusing on the web-dependent query subset (GAIA-web)[7], and also SimpleQA (Wei et al., 2024), where we use web agent to explore the answers. Since GAIA typically require multi-step web navigation combined with arithmetic/logical reasoning. and the self-improved agent LLM focuses solely on action generation, we adopt a hybrid approach: we use GPT-4o to decompose queries into sub-tasks that web agents can address, and also leverage GPT-4o for result generation and calculation. The web agent component is based on Llama-based models including WebEvolver. We use bing.com instead of Google due to CAPTCHA challenges, which can also demonstrating our method's out-of-domain generalization since the training data does not contain trajectories in bing.com. Results on Table 4 show consistent improvement on Level 1 and SimpleQA

[7] https://github.com/MinorJerry/WebVoyager/blob/main/data/GAIA_web.jsonl

queries through self-improvement and world model augmentation, mirroring trends observed in WebVoyager and Mind2web-live. However, Level 2 queries, which demand deeper reasoning and extended multi-step interactions, show limited gains, as these capabilities lie beyond our current training scope. This limitation highlights an important direction for future work in developing agents for complex, real-world web tasks.

k	WebVoyager
2	48.62
3	**51.37**
5	50.73

Table 4: Ablations on the branching factor k in WMLA.

Ablations on the Branching Factor k We conducted explicit ablation on the branching factor k (number of sampled candidate actions) in WMLA. Performance plateaus around k=3-5 because: possible action spaces rarely yield >5 substantially different options per state. k=3 as in the paper is still the best choice.

Analysis of World-Model Synthesized Trajectories We provide two cases on the world-model synthesized trajectories, indicating that LLM itself contains useful knowledge about the common structures of the web and has the potential to provide diverse trajectories. It is provided in Figure 4. This case demonstrates an operation involving a click on the 'sort by' menu in the GitHub search console. Although the world model has not been further fine-tuned on trajectories that include clicking the 'sort by' button, it is still able to accurately generate the menu items for GitHub Search, such as sorting by best match, most stars, and so on. This capability arises from the commonsense knowledge inherently encoded in the LLM. We find that this feature is highly beneficial for improving the diversity of interactions with previously unseen websites.

5 Conclusion

In this paper, we present WebEvolver, a framework for agent foundation model self-improvement through co-learning with a world model, which enhances the effectiveness of the self-improvement cycle. The co-learned world model can also be utilized for inference-time look-ahead, aiding in the selection among different sampled actions. Experiments on WebVoyager, Mind2Web-Live, and GAIA-web demonstrate the effectiveness of boosting the performance of self-improving agent.

Limitations

First, the agent system we use includes only an action generation module, whereas recent studies have shown that incorporating a standalone planning module can further enhance agent performance. However, planning is orthogonal to our research focus. Second, because we focus on open-domain, real-world web environments, websites may change over time, making it difficult for future work to exactly replicate the same web conditions. To ensure fair comparisons in our experiments, we complete all tasks within approximately the same time frame. Additionally, we include GAIA-web and SimpleQA as two supplementary evaluation datasets, as they primarily focus on factual questions and are less susceptible to significant changes over time.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix X. Yu, and Sanjiv Kumar. 2023. Rest meets react: Self-improvement for multi-step reasoning LLM agent. *CoRR*, abs/2312.10003.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J. Storkey, Tim Pearce, and François Fleuret. 2024. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Anthropic. 2024. Model context protocol. Open-source protocol.

Anthropic. 2025. Claude 3.7 sonnet: Hybrid reasoning model. `https://www.anthropic.com/news/claude-3-7-sonnet`. Accessed: 2025-04-18.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2025. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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

Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, and 1 others. 2025. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *arXiv preprint arXiv:2508.00414*.

Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2024. Is your LLM secretly a world model of the internet? model-based planning for web agents. *CoRR*, abs/2411.06559.

David Ha and Jürgen Schmidhuber. 2018. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 8154–8173. Association for Computational Linguistics.

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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 6864–6890. 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. *CoRR*, abs/2410.19609.

Minda Hu, Tianqing Fang, Jianshu Zhang, Junyu Ma, Zhisong Zhang, Jingyan Zhou, Hongming Zhang, Haitao Mi, Dong Yu, and Irwin King. 2025. Webcot: Enhancing web agent reasoning by reconstructing chain-of-thought in reflection, branching, and rollback. *arXiv preprint arXiv:2505.20013*.

Mengzhao Jia, Wenhao Yu, Kaixin Ma, Tianqing Fang, Zhihan Zhang, Siru Ouyang, Hongming Zhang, Meng Jiang, and Dong Yu. 2024. Leopard: A vision language model for text-rich multi-image tasks. *CoRR*, abs/2410.01744.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024a. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 881–905. Association for Computational Linguistics.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024b. Tree search for language model agents. *CoRR*, abs/2407.01476.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, and 1 others. 2025a. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025b. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*.

Xinbin Liang, Jinyu Xiang, Zhaoyang Yu, Jiayi Zhang, and Sirui Hong. 2025. Openmanus: An open-source framework for building general ai agents. `https://github.com/mannaandpoem/OpenManus`.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

MiroMindAI. 2025. Miroflow: A consistent agent framework with reproducible performance. `https://github.com/MiroMindAI/MiroFlow`. Accessed: [Insert date of access here].

Monica.Im. 2025. Manus ai. Technical report, Monica.Im.

Shikhar Murty, Christopher D. Manning, Peter Shaw, Mandar Joshi, and Kenton Lee. 2024. BAGEL: bootstrapping agents by guiding exploration with language. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. 2025. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *CoRR*.

OpenAI. 2023. Gpt-4 technical report. Technical Report. A large multimodal model capable of processing image and text inputs and producing text outputs. Achieves human-level performance on various professional benchmarks including passing a simulated bar exam in the top 10

OpenAI. 2025. Introducing deep research. Technical report, OpenAI.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Awadallah. 2025. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. *CoRR*, abs/2502.11357.

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

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

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent Q: advanced reasoning and learning for autonomous AI agents. *CoRR*, abs/2408.07199.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Agent planning with world knowledge model. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Max Olan Smith and Michael P. Wellman. 2023. Co-learning empirical games and world models. *CoRR*, abs/2305.14223.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, and 1 others. 2025. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*.

Brandon Trabucco, Gunnar A. Sigurdsson, Robinson Piramuthu, and Ruslan Salakhutdinov. 2025. Towards internet-scale training for agents. *CoRR*, abs/2502.06776.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. 2024. Diffusion models are real-time game engines. *CoRR*, abs/2408.14837.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025a. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025b. Webwalker: Benchmarking llms in web traversal. *CoRR*, abs/2501.07572.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *CoRR*, abs/2406.04151.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. 2024. Gödel agent: A self-referential agent framework for recursive self-improvement. *CoRR*, abs/2410.04444.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Weihao Zeng, Yuzhen Huang, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. 2024. B-star: Monitoring and balancing exploration and exploitation in self-taught reasoners. *CoRR*, abs/2412.17256.

Hongming Zhang, Ruixin Hong, and Dong Yu. 2025a. Streaming looking ahead with token-level self-reward. *CoRR*, abs/2503.00029.

Hongming Zhang, Xiaoman Pan, Hongwei Wang, Kaixin Ma, Wenhao Yu, and Dong Yu. 2024a. Cognitive kernel: An open-source agent system towards generalist autopilots. *CoRR*, abs/2409.10277.

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

Zhisong Zhang, Tianqing Fang, Kaixin Ma, Wenhao Yu, Hongming Zhang, Haitao Mi, and Dong Yu. 2025b. Enhancing web agents with explicit rollback mechanisms. *Preprint*, arXiv:2504.11788.

Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. 2024c. Mmina: Benchmarking multihop multimodal internet agents. *CoRR*, abs/2404.09992.

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, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

A Details of Agent Implementation

In this section, we present additional details of the prompt we used for the web agent.

The system prompt for web agent action generation:

AGENT SYSTEM PROMPT

You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks. These tasks will be accomplished through the use of specific actions you can issue.

Here’s the information you’ll have:

- The user’s objective: This is the task you’re trying to complete.
- The current observation (web page’s accessibility tree): This is a simplified representation of the webpage, providing key information. Optionally, you may be provided with a screenshot of the webpage. You should pay close attention to the screesnhot to make decisions.
- The open tabs: These are the tabs you have open.
- The previous actions: You can refer to the conversation history with the user to see the actions you have taken. It may be helpful to track your progress.

The actions you can perform are the following:

- ‘click [id]‘: This action clicks on an element with a specific id on the webpage.
- ‘type [id] [content] [press_enter_after=0|1]‘: Use this to type the content into the field with id. By default, the Ënterkey is pressed after typing unless press_enter_after is set to 0.
- ‘wait‘: Wait for the page to load, with a duration of 5 seconds.
- ‘goback‘: Navigate to the previously viewed page.
- ‘restart‘: Navigate to the Google search homepage. When you can’t find information in some websites, try starting over from Google search.
- ‘stop [answer]‘: Issue this action when you believe the task is complete. If the objective is to find a text-based answer, provide the answer in the bracket. If you believe the task is impossible to complete, provide the answer as "N/A" in the bracket.

To be successful, it is very important to follow the following rules:

1. You should only issue an action that is valid given the current observation. For example, you should NOT type into buttons or click on statictext.

2. You should only issue one action at a time.

3. STRICTLY Avoid repeating the same action if the webpage remains unchanged. You may have selected the wrong web element or numerical label. Continuous use of the Wait is also NOT allowed.

4. Issue stop action when you think you have achieved the objective. Don’t generate anything after stop.

Your reply should strictly follow the format: Thought: {{Your brief thoughts (briefly summarize the info that will help complete the task)}} Action: “‘{{the next action you choose to take}}“‘

The system prompt for using world model as a web server, by generating the next observation based on current observation and the scheduled action. We present two variation of world model objectives, the first one is to only predict an abstract short description of what the next observation is (denoted as **Abstract Description**), and the second one is to predict the structured accessibility tree of the next observation (denoted as **Accessibility Tree**).

WORLD MODEL LOOK-AHEAD (ABSTRACT DESCRIPTION)

You are a web server. You are given the current observed accessibility tree of the web page, and an action to perform.

The expected output is a short description on what the next observation is, in the form of free text.

The definitions of the actions are as follows: The actions you can perform are the following:

- ‘click [id]‘: This action clicks on an element with a specific id on the webpage.
- ‘type [id] [content] [press_enter_after=0|1]‘: Use this to type the content into the field with id. By default, the Ënterkey is pressed after typing unless press_enter_after is set to 0.
- ‘scroll [direction=down|up]‘: Scroll the page up or down.
- ‘goback‘: Navigate to the previously viewed page.
- ‘restart‘: Navigate to the original home page and restart the action.

World Model Look-ahead (Accessibility Tree)

You are an intelligent assistant designed to interact with web pages through an accessibility tree. Your task is to predict the accessibility tree of the next web page based on the given starting accessibility tree and a specified action. The format of accessibility tree:

Tab 0 (current): Google \n \n[1] RootWebArea ’Google’ focused: true\n[̃2] link ’Gmail ’\n[̃3] link ’Search Image ’\n[̃4] button ’Google Apps’ expanded: false\n[̃5] link ’Log in’\n[̃6] image ’2024’\n[̃7] combobox ’Search’ focused: true autocomplete: both hasPopup: listbox required: false expanded: false\n[̃8] button ’Share’

The format of action:

type [7] [JQuery selector for elements with specific class] [1]

which indicates typing "JQuery selector for elements with specific class" into the field with id 7, corresponding to the combobox (search box) on the Google homepage.

The definitions of the actions are as follows: The actions you can perform are the following:

- ‘click [id]‘: This action clicks on an element with a specific id on the webpage.
- ‘type [id] [content] [press_enter_after=0|1]‘: Use this to type the content into the field with id. By default, the Ënterkey is pressed after typing unless press_enter_after is set to 0.
- ‘scroll [direction=down|up]‘: Scroll the page up or down.
- ‘goback‘: Navigate to the previously viewed page.
- ‘restart‘: Navigate to the Google search homepage. When you can’t find information in some websites, try starting over from Google search.

The system prompt for automatic evaluation of a web agent task.

Automatic Evaluation

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

1. Web Task Instruction: This is a clear and specific directive provided in natural language, detailing the online activity to be carried out. 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).

2. Result Webpage Accessibility Tree: This is a representation of the web page showing the result or intermediate state of performing a web task. It serves as proof of the actions taken in response to the instruction.

3. Result Response: This is a textual response obtained after the execution of the web task. It serves as textual result in response to the instruction.

- 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 webpage when comparing it to the instructions.

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

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

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

- Note the difference: 1) Result response may contradict the screenshot, then the content of the screenshot prevails, 2) The content in the Result response is not mentioned on the screenshot, choose to believe the content.

You should elaborate on how you arrived at your final evaluation and then provide a definitive verdict on whether the task has been successfully accomplished, either as 'SUCCESS' or 'NOT SUCCESS'.

The system prompt for automatic evaluation of world modeling.

World Model Intrinsic Evaluation

You are tasked with evaluating the accuracy of ntnerated accessibility tree against a ground truth accessibility tree obtained from an actual web server. Your evaluation should focus on three main criteria: structure correctness, element correctness, and similarity. Follow the instructions below to perform a detailed comparison:

Criteria for Evaluation:
1. **Structure Correctness**:

- Ensure that the basic hierarchy and relationships between elements in the generated tree match the ground truth.

- Ensure that interactive elements (like buttons, links, forms) are correctly represented and maintain their intended functionality.

2. **Similarity (GPT-score)**:

- Assess how similar the generated content is compared to the ground truth.

- Provide a similarity score based on the overall content and structure comparison.

3. **Overall Functionality Assessment**:

- Compare the functional coherence of the generated tree to the ground truth tree, focusing on the representation and functionality of interactive elements.

- Evaluate the semantic coherence of the generated tree, ensuring that it conveys the same meaning and purpose as the ground truth.

For example, if if the webpage is on Allrecipe, as long as the generated tree contain necessary recipe, no matter hallucination, it can be considered as success. For example, if the webpage is on google, in searching for some information, then only consider whether the generated tree contain roughly necessary information without the need to check the factuality.

1. **Input Trees**:

- You will be provided with two accessibility trees: one generated by a language model simulating a web browser, and one obtained from an actual web server.

2. **Output Format**:
- Provide rationale of your findings, including:

- Structural discrepancies

```
• Similarity score with an explanation

• Scores should be selected from [0, 1, 2, 3]. 3 means exactly the same and 0 means a total
  failure of generation.

### Example Output
Structure Correctness: [THOUGHT]\n Score: [score]\n
Similarity: [THOUGHT]\n Score: [score]\n
Overall Functionality Assessment: [THOUGHT]\nScore: [score]\n
```

B Additional Details on Mind2web-live and WebVoyager Dataset

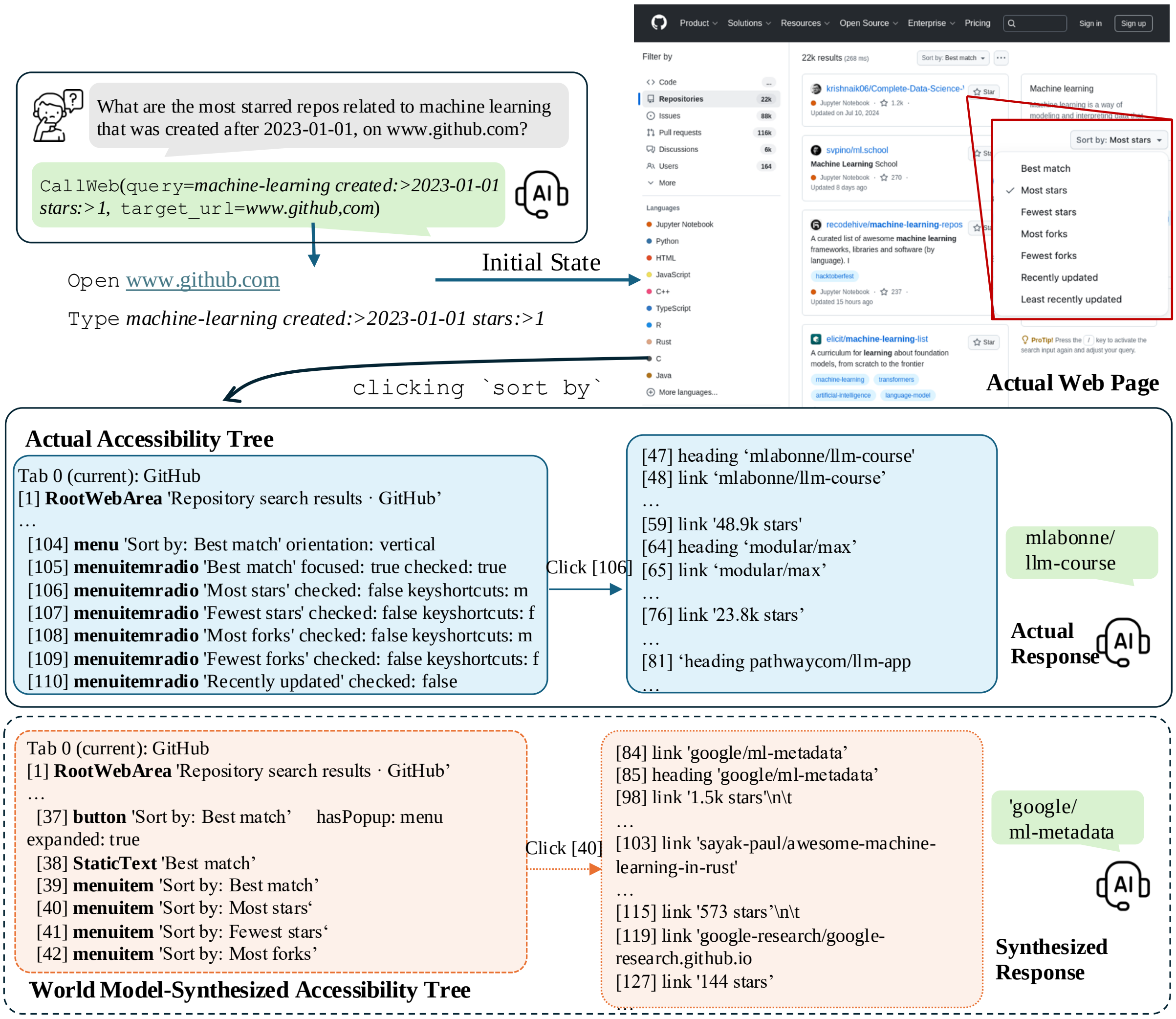
We conduct our evaluations using a subset of the testing portion of Mind2Web-Live[8] and WebVoyager[9]. Here is a list of the websites that are excluded:

Excluded Websites

```
EXCLUDED_WEBSITES_MIND2WEB = { ’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’, }
EXCLUDED_WEBSITES_WEBVOYAGER = { ’booking’, ’espn’, ’amazon’, ’google’, ’googleflight’ }
```

[8] https://huggingface.co/datasets/iMeanAI/Mind2Web-Live/blob/main/mind2web-live_test_20241024.json

[9] https://github.com/MinorJerry/WebVoyager/blob/main/data/WebVoyager_data.jsonl



What are the most starred repos related to machine learning that was created after 2023-01-01, on www.github.com?

CallWeb(query=*machine-learning created:>2023-01-01 stars:>1*, target_url=*www.github,com*)

Open www.github.com

Type *machine-learning created:>2023-01-01 stars:>1*

clicking `sort by`

```
Tab 0 (current): GitHub
[1] RootWebArea 'Repository search results · GitHub'
…
  [104] menu 'Sort by: Best match' orientation: vertical
  [105] menuitemradio 'Best match' focused: true checked: true
  [106] menuitemradio 'Most stars' checked: false keyshortcuts: m
  [107] menuitemradio 'Fewest stars' checked: false keyshortcuts: f
  [108] menuitemradio 'Most forks' checked: false keyshortcuts: m
  [109] menuitemradio 'Fewest forks' checked: false keyshortcuts: f
  [110] menuitemradio 'Recently updated' checked: false
```

Click [106]

```
[47] heading 'mlabonne/llm-course'
[48] link 'mlabonne/llm-course'
…
[59] link '48.9k stars'
[64] heading 'modular/max'
[65] link 'modular/max'
…
[76] link '23.8k stars'
…
[81] 'heading pathwaycom/llm-app
…
```

mlabonne/ llm-course

```
Tab 0 (current): GitHub
[1] RootWebArea 'Repository search results · GitHub'
…
  [37] button 'Sort by: Best match'     hasPopup: menu
expanded: true
  [38] StaticText 'Best match'
  [39] menuitem 'Sort by: Best match'
  [40] menuitem 'Sort by: Most stars'
  [41] menuitem 'Sort by: Fewest stars'
  [42] menuitem 'Sort by: Most forks'
```

Click [40]

```
[84] link 'google/ml-metadata'
[85] heading 'google/ml-metadata'
[98] link '1.5k stars'\n\t
…
[103] link 'sayak-paul/awesome-machine-learning-in-rust'
…
[115] link '573 stars'\n\t
[119] link 'google-research/google-research.github.io
[127] link '144 stars'
…
```

'google/ ml-metadata

Figure 4: An example of world model-synthesized trajectory.