

DynaWeb: Model-Based Reinforcement Learning of Web Agents

Hang Ding^{1,*}, Peidong Liu^{2,*}, Junqiao Wang^{1,*}, Ziwei Ji³, Meng Cao^{4,8},
Rongzhao Zhang⁵, Lynn Ai⁶, Eric Yang⁶, Tianyu Shi^{4,6}, Lei Yu⁷

¹Shanghai Jiao Tong University, ²Sichuan University,

³Hong Kong University of Science and Technology, ⁴McGill University,

⁵Shanghai AI Lab, ⁶Gradient, ⁷University of Toronto,

⁸Mila - Quebec AI Institute

*Equal contribution. Correspondence: jadeleiyu@cs.toronto.edu, tianyu.shi3@mcgill.ca

Abstract

The development of autonomous web agents, powered by Large Language Models (LLMs) and reinforcement learning (RL), represents a significant step towards general-purpose AI assistants. However, training these agents is severely hampered by the challenges of interacting with the live internet, which is inefficient, costly, and fraught with risks. Model-based reinforcement learning (MBRL) offers a promising solution by learning a *world model* of the environment to enable simulated interaction. This paper introduces DynaWeb, a novel MBRL framework that trains web agents through interacting with a web world model trained to predict naturalistic web page representations given agent actions. This model serves as a synthetic web environment where an agent policy can *dream* by generating vast quantities of rollout action trajectories for efficient online reinforcement learning. Beyond free policy rollouts, DynaWeb incorporates real expert trajectories from training data, which are randomly interleaved with on-policy rollouts during training to improve stability and sample efficiency. Experiments conducted on the challenging WebArena and WebVoyager benchmarks demonstrate that DynaWeb consistently and significantly improves the performance of state-of-the-art open-source web agent models. Our findings establish the viability of training web agents through imagination, offering a scalable and efficient way to scale up online agentic RL. Our code can be found.¹

1 Introduction

The paradigm of artificial intelligence is rapidly shifting toward proactive, agentic systems that can autonomously execute complex, long-horizon tasks in open-ended environments. Large Language Models (LLMs) have emerged as a powerful backbone for such agents, enabling rich rea-

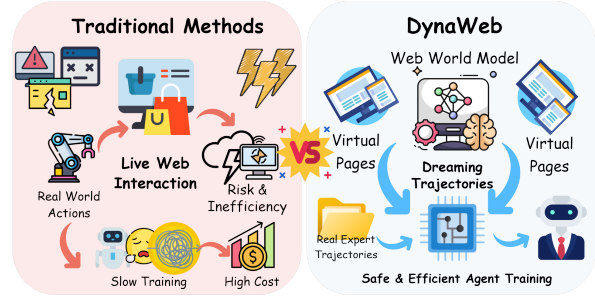


Figure 1: Comparison between traditional web agent training via live web interaction and DynaWeb. By replacing risky and inefficient real-world interaction with a learned web world model, DynaWeb enables imagination-driven training using virtual pages and dreamed trajectories, optionally augmented with real expert data, resulting in safer and more efficient agent optimization.

soning, flexible action generation, and natural language interaction. In the web domain, LLM-based agents have demonstrated strong capabilities in navigating real websites and accomplishing user-specified goals through multi-step interaction, fueled by advances in prompting, structured reasoning, and action abstractions (Yao et al., 2023; Zhou et al., 2024b; He et al., 2024). Beyond imitation learning, recent work has shown that training web agents with online reinforcement learning (RL) can substantially improve robustness, exploration, and long-horizon decision-making, allowing agents to surpass static demonstrations and adapt through trial-and-error interaction (Wei et al., 2025; Qi et al., 2025; Zhuang et al., 2025).

Despite its promise, online RL for web agents remains constrained by the cost and risk of real-environment interaction. Large-scale on-policy data collection requires direct live-web interaction, making training inefficient, expensive, and hard to control. Agents may trigger irreversible actions, including unintended purchases, account

¹<https://github.com/jadeleiyu/MBRL-Agent/tree/rllm>

changes, or data submissions, while facing non-deterministic pages, transient failures, and external interference. These issues make pure online RL costly and hazardous for real-world web environments (Zhou et al., 2024b; Qi et al., 2025). This raises a key question: how can web agents retain the benefits of online RL while sharply reducing direct reliance on the live web? A natural solution is to replace risky real interaction with a learned, controllable surrogate that approximates web dynamics. Recent work therefore explores web *world models* as learned web-environment simulators. However, their role remains mostly auxiliary. Some methods use world models at inference time for short-horizon look-ahead and action evaluation, treating them as decision-time reasoning tools rather than learning substrates (Gu et al., 2024; Wei et al., 2025). Others synthesize offline trajectories for supervised fine-tuning or imitation-style training, decoupling generated experience from on-policy optimization (Fang et al., 2025a; Pahuja et al., 2025). Still missing is an *online* model-based reinforcement learning (MBRL) paradigm for web agents, where imagined rollouts act as first-class experience that directly augments on-policy policy optimization.

In this work, we revisit classical model-based reinforcement learning for modern web agents. Inspired by Dyna (Sutton, 1991) and imagination-based frameworks such as Dreamer (Hafner et al., 2020), we propose **DynaWeb**, an MBRL framework that turns a web world model from a planning or data-generation tool into a core component of online RL. DynaWeb uses the world model as a controllable synthetic web environment to replace or augment costly real interaction. Specifically, an LLM-based web world model acts as a learned web server: given the current page representation and agent action, it predicts the next page state and provides task-level feedback for policy optimization. By mixing real and imagined experience, DynaWeb enables scalable on-policy RL through imagination while retaining the benefits of interaction.

Crucially, DynaWeb uses two complementary experience sources. Besides policy-driven imagined rollouts from the web world model, it incorporates fully real expert trajectories from existing training data. These trajectories are independent of the world model and reflect ground-truth web interactions. During training, real expert trajectories are randomly interleaved with imagined

rollouts and real on-policy interaction, giving the agent high-quality demonstrations while without costly live-web exploration. This simple interleaving strategy preserves the on-policy signal and enables efficient online RL without real-environment interactions.

Our contributions are summarized as follows:

- We train a web world model that predicts naturalistic web page state transitions in the form of structured accessibility tree representations, enabling realistic simulation of web environment dynamics without live web interaction.
- We show that policy-driven imagined rollouts generated by a learned web world model can be directly used as on-policy training experience for reinforcement learning of web agents.
- We propose **DynaWeb**, a model-based reinforcement learning framework that integrates fully real expert trajectories from training data with imagined rollouts from a world model, enabling effective policy optimization without interacting with the live web.
- Through extensive experiments and analysis, we demonstrate consistent performance improvements on WebArena and WebVoyager, and provide empirical insights into the design and use of world-model-based imagination for training web agents.

2 Related work

Web Agent Recent advances in web agents are largely driven by (multimodal) large language models (LLMs) serving as the core decision-making backbone (Dubey et al., 2024; Jia et al., 2024; OpenAI, 2023; Anthropic, 2025). On top of these models, reasoning and interaction frameworks such as ReAct (Yao et al., 2023), MCP (Anthropic, 2024), and Cognitive Kernel (Zhang et al., 2024a) enable structured multi-step web actions. Web agents are commonly evaluated on interactive benchmarks including WebShop (Yao et al., 2022), Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2024b), VisualWebArena (Koh et al., 2024a), WebVoyager (He et al., 2024), WebWalker (Wu et al., 2025b), and MMInA (Zhang et al., 2024c).

Beyond off-the-shelf prompting, a broad line of work improves web agents via data scaling and stronger agent training pipelines. Data-centric efforts such as Explorer (Pahuja et al., 2025), NNetNav (Murty et al., 2025), and InSTA (Trabucco et al., 2025) collect or synthesize high-quality interaction data, while recent agent foundation models and deep research agents push toward more general web interaction at scale, e.g., 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., 2025b), MiroFlow (MiroMindAI, 2025), and multimodal research agents such as WebWatcher (Geng et al., 2025). In parallel, inference-time optimization (e.g., tree search and self-reflection) further improves decision-making without additional training (Koh et al., 2024b; Putta et al., 2024; Yu et al., 2024; Zhou et al., 2024a; Zhang et al., 2024b; Shinn et al., 2023).

World Models World models stem from model-based reinforcement learning (Ha and Schmidhuber, 2018) and have recently re-emerged as a useful abstraction for agent reasoning and planning (Valevski et al., 2024; Alonso et al., 2024; Smith and Wellman, 2023). With LLMs, several works view them as implicit world models for simulating future states. RAP (Hao et al., 2023) combines LLM-based world modeling with Monte Carlo Tree Search for hypothetical trajectory exploration, while WKM (Qiao et al., 2024) distills structured world knowledge from agent trajectories to guide planning.

In web environments, WebDreamer (Gu et al., 2024) and WMA (Chae et al., 2025) use LLMs to predict web-action outcomes through natural-language simulation. Yet these methods mainly use world models at inference time, as planning or prompting modules for action selection. Even with multi-step simulation, imagined roll-outs guide immediate decisions rather than improve the policy through training.

By contrast, DynaWeb makes the web world model part of learning itself. It trains a dedicated world model and uses its multi-step imagined trajectories for policy optimization, moving beyond inference-time reasoning toward genuine model-based RL for web agents. This directly targets the data inefficiency and real-interaction cost of prior RL-based web agents, enabling scalable imagination-driven training.

Reinforcement Learning of Agents RL improves agent policies through interactive feedback, making it well suited to long-horizon web decision making (Wei et al., 2025; Qi et al., 2025; Wu et al., 2025a; Zhuang et al., 2025). WebAgent-R1 (Wei et al., 2025) is a representative online RL method, optimizing multi-turn web policies with outcome rewards and scalable trajectory sampling, such as multi-group GRPO (Shao et al., 2024). WebRL (Qi et al., 2025) instead stresses self-evolving curricula and result-supervised feedback to generate tasks and improve robustness.

Other works modify RL objectives or add supervision to shape reasoning and planning. WebDancer (Wu et al., 2025a) combines QA-style structured supervision with policy optimization, such as DAPO (Yu et al., 2025), to learn actionable reasoning traces. WorkForceAgent-R1 (Zhuang et al., 2025) combines behavior cloning with GRPO-style optimization for single-step reasoning and multi-step planning. Beyond pure online RL, AutoWebGLM (Lai et al., 2024) uses SFT, RL, rejection-sampling fine-tuning, and curriculum learning to first bootstrap navigation skills and then refine them with RL.

Despite strong results, these agents still require substantial real-environment interaction, which is costly, unstable, and risky at scale, motivating our imagination-driven training framework.

3 Method

3.1 Problem Formulation

We formulate the web agent task as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$. Given a natural language query q and a target website w , the agent is required to complete a multi-step web interaction to satisfy the task objective. The state space $\mathcal{S}$ represents the full underlying web environment, which is not directly observable by the agent. Instead, at each time step t , the agent receives a partial observation $o_t \in \mathcal{O}$ obtained via an observation function $o_t = \Omega(s_t)$, where $\Omega(\cdot)$ extracts visible information such as the current URL and web elements from state s_t .

The action space $\mathcal{A}$ consists of atomic browser operations, including click, type, goback, scroll up/down, and stop. Executing an action $a_t \in \mathcal{A}$ deterministically advances the environment according to the transition function $\mathcal{T}$:

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

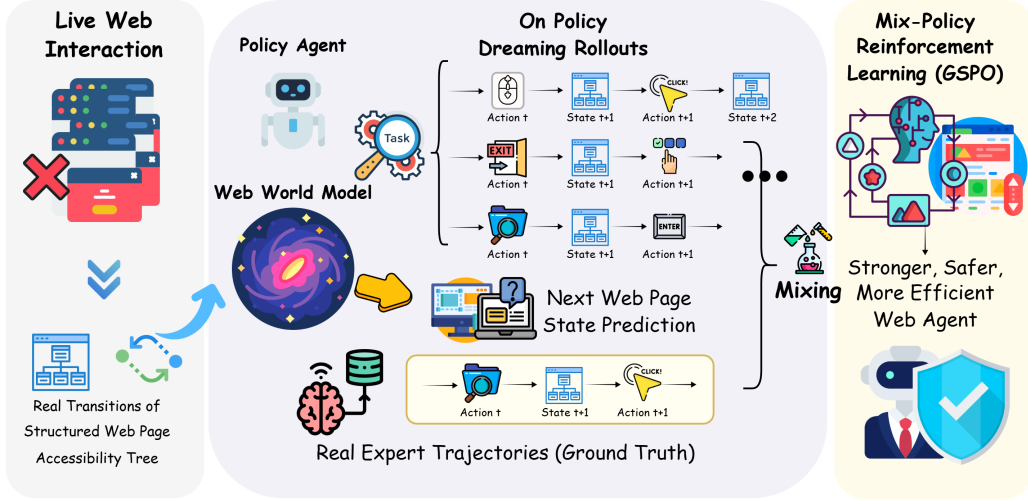


Figure 2: **Overview of DynaWeb.** DynaWeb trains web agents via imagination-driven, model-based reinforcement learning. A learned web world model serves as a synthetic environment, enabling the agent to generate multi-step imagined rollouts without interacting with the live web. These imagined trajectories are mixed with a small fraction of real expert trajectories to stabilize learning. The agent policy is optimized using sequence-level policy optimization, allowing efficient and robust credit assignment for long-horizon web tasks with sparse terminal rewards.

Following prior work (Zhang et al., 2024a), we represent each observation o_t using an accessibility tree that captures the structured layout of visible web elements. The agent is parameterized by a large language model (LLM) policy π_θ , which conditions on the task query q , the interaction history, and system instructions I to generate reasoning traces and actions. At time step t , the agent samples a chain-of-thought h_t together with an action a_t as

$$(h_t, a_t) \sim \pi_\theta(\cdot \mid I, q, o_{1:t}, h_{1:t-1}, a_{1:t-1}). \quad (2)$$

An interaction episode produces a trajectory $\tau = (o_1, h_1, a_1, \dots, o_T, h_T, a_T)$, where T denotes the total number of interaction steps. Upon termination, the trajectory is assigned a scalar reward $\hat{r}(\tau, q) \in [0, 1]$ based on task completion, obtained via self-assessment or environment-specific evaluation.

3.2 Web World Model

We introduce a web world model to approximate web transition dynamics by predicting how visible states change after agent actions. Given a partial observation o_t as an accessibility tree and an action $a_t \in \mathcal{A}$, the world model predicts the next observation:

$$\hat{o}_{t+1} \sim p_\phi(\cdot \mid o_t, a_t, I), \quad (3)$$

where p_ϕ is an LLM parameterized by ϕ . Rather than learning latent dynamics, it operates directly

in observation space and generates web page representations that can be consumed by the policy during imagined rollouts.

Directly predicting the full next textual observation is inefficient, since many web transitions modify only a small part of the current page and thus o_{t+1} is often highly similar to o_t . We therefore decompose next-state prediction into two steps: the model first predicts a free-form natural-language description of the state change $\Delta(o_t, o_{t+1})$ caused by a_t , and then applies this change to o_t to obtain o_{t+1} . Thus, training focuses on the reasoning-intensive subtask of predicting state changes, while inference leverages the LLM’s instruction-following ability to perform state editing.

The world model is trained on real web interaction trajectories from StanfordNLP/NNet-Nav (Murty et al., 2025). We clean the data by removing erroneous or incomplete trajectories, including missing observations, invalid actions, and inconsistent transitions. For each valid transition (I, o_t, a_t, o_{t+1}) , we prompt GPT-5 to generate both a reasoning trace r and the corresponding state-change description $\Delta(o_t, o_{t+1})$. These GPT-5-generated reasoning traces are used as SFT targets to explicitly strengthen the world model’s transition reasoning and improve its generalization beyond surface-level state matching. The world model is trained to predict both the reasoning trace

and subsequent state changes conditioned on the instruction, current accessibility tree, and executed action:

$$\mathcal{L}_\phi = \sum_{(I, o_t, a_t, r, \Delta)} -\log p_\phi(r, \Delta | I, o_t, a_t). \quad (4)$$

The learned world model then serves as a reusable simulator for generating multi-step imagined trajectories, which we use for model-based reinforcement learning in the next section.

3.3 DynaWeb: Model-based RL of Web Agents

DynaWeb trains web agents through model-based reinforcement learning by relying on imagined rollouts generated by a learned web world model, eliminating the need for live web interaction during training. Given a task query q and an initial observation o_1 , the agent policy π_θ interacts with the world model as a synthetic environment to produce multi-step trajectories. At each step t , the agent samples an action

$$a_t \sim \pi_\theta(\cdot | o_{1:t}, h_{1:t-1}, a_{1:t-1}, q), \quad (5)$$

and the world model predicts the next observation

$$\hat{o}_{t+1} \sim p_\phi(\cdot | \hat{o}_t, a_t, q), \quad \hat{o}_1 = o_1. \quad (6)$$

Iterating this process yields an imagined trajectory

$$\hat{\tau} = (\hat{o}_1, h_1, a_1, \dots, \hat{o}_T, h_T, a_T), \quad (7)$$

without interacting with the live web environment.

To enable reinforcement learning on imagined trajectories, a task-level reward signal is obtained via model-based self-assessment conditioned on the task query. After trajectory termination, a task-level completion reward $\hat{r}(\hat{\tau}, q) \in \{0, 1\}$ is obtained via model-based self-assessment with respect to the task query, indicating whether the task objective has been achieved.

We treat the return as $G(\hat{\tau}) = \hat{r}(\hat{\tau}, q)$ and use the resulting imagined rollouts as valid training data for policy gradient optimization, substantially reducing reliance on real-environment interaction while preserving task-level supervision.

Beyond policy-driven imagined rollouts, DynaWeb additionally incorporates fully real expert trajectories sampled from training data to improve training stability and sample efficiency. These expert trajectories correspond to ground-truth web interactions and are entirely independent of the

world model. In practice, real expert trajectories are randomly interleaved with policy-driven imagined rollouts during training, allowing the agent to benefit from high-quality demonstrations while learning under a simulated environment.

To optimize the agent policy from this mixture of real and imagined experiences, we adopt Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025). Concretely, GSPO lifts importance sampling from the token level to the rollout (sequence) level by assigning each trajectory a single sequence-level ratio, which is then applied uniformly to all tokens in the trajectory. Let $\{\hat{\tau}^i\}_{i=1}^G$ denote a group of rollouts sampled from $\pi_{\theta_{\text{old}}}$ (either imagined via the world model or real expert trajectories), and let y^i be the serialized token sequence produced by the policy along $\hat{\tau}^i$ (i.e., concatenating all generated reasoning traces and actions). GSPO optimizes the clipped objective

$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}_{q \sim \mathcal{D}, \{\hat{\tau}^i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \left[\frac{1}{G} \sum_{i=1}^G \min \left(s^i(\theta) \hat{A}^i, \text{clip}(s^i(\theta), 1 - \varepsilon, 1 + \varepsilon) \hat{A}^i \right) \right] \quad (8)$$

where $\hat{A}^i$ denotes the (trajectory-level) advantage computed from the terminal return $G(\hat{\tau}^i)$. The sequence-level ratio $s^i(\theta)$ is defined as the geometric mean of token-wise likelihood ratios:

$$\begin{aligned} s^i(\theta) &= \left(\frac{\pi_\theta(y^i | q, o_1)}{\pi_{\theta_{\text{old}}}(y^i | q, o_1)} \right)^{1/|y^i|} \\ &= \exp \left(\frac{1}{|y^i|} \sum_{k=1}^{|y^i|} \log r_k^i(\theta) \right), \end{aligned} \quad (9)$$

with token-level ratios

$$r_k^i(\theta) = \frac{\pi_\theta(y_k^i | q, o_1, y_{<k}^i)}{\pi_{\theta_{\text{old}}}(y_k^i | q, o_1, y_{<k}^i)}. \quad (10)$$

4 Experiments

4.1 Setups

Benchmarks and metrics We evaluate DynaWeb on two challenging web-navigation benchmarks: WebArena (Zhou et al., 2024b), with 812 real-life tasks in simulated environments across five websites and four domains, and WebVoyager (He et al., 2024), with 643 open-ended tasks from 15 popular live websites. We use Success Rate (SR), the percentage of instructions successfully completed by the agent trajectory, as the

Table 1: Success rate (%) on self-hosted websites in WebArena. Results are reported for five representative website domains (Reddit, GitLab, Maps, CMS, and Shopping). DynaWeb achieves the highest average success rate and consistently outperforms strong supervised, RL, and inference-time lookahead baselines across most domains. Best results are in bold, and second-best are underlined.

<i>Self-hosted Websites (WebArena)</i>						
<i>Method</i>	Reddit	Gitlab	Map	CMS	Shopping	<i>Average SR</i>
Llama3.1-8B-Ins +CoT	0.0	3.3	3.3	2.9	11.1	4.1
GPT-4o	10.5	10.0	20.0	20.0	11.1	14.3
Qwen2.5-32B	10.5	20.0	<u>19.2</u>	19.2	17.8	17.3
NNetNav	16.4	8.7	11.4	16.7	25.9	15.8
Go-Browse	30.7	15.3	17.9	25.3	22.4	22.3
WebRL	<u>35.6</u>	<u>23.5</u>	19.1	<u>30.6</u>	24.7	<u>26.7</u>
ITL	26.5	18.0	12.6	23.6	<u>31.3</u>	22.4
DynaWeb-8B (Ours)	43.8	28.7	17.8	31.5	33.2	31.0

main metric. For WebArena, we use the official AWS EC2 Docker environment for fair and reproducible comparison. For WebVoyager, we remove tasks whose websites are inaccessible in our environment due to geographic or IP restrictions. Following (Fang et al., 2025a), we run experiments twice in roughly the same time window and report average results.

Web Agents We use the open-source NNetNav agent family (Murty et al., 2025), fine-tuned from Llama-3.1-8B, as the DynaWeb backbone. NNetNav provides two agents trained on live and simulated web-navigation trajectories. For each benchmark, we initialize from the corresponding NNetNav checkpoint and train it in the DynaWeb MBRL environment simulated by a WWM. During training, 40% of dreamed trajectories are randomly replaced with gold-standard trajectories from the NNetNav SFT datasets.

World Models For WWMs, we fine-tune GPT-oss-120b on curated next-state prediction data as described in Section 3, obtaining a live WWM for WebVoyager and a simulated WWM for WebArena. Each WWM is SFT-trained for one epoch with LlamaFactory. During GSPO rollouts, we limit synthetic trajectories to at most 10 steps and terminate early if the WWM predicts a terminal state, reducing hallucination effects. To cover navigation states at different stages, rollout initial states are randomly sampled from NNetNav-live or NNetNav-wa SFT trajectories, including both initial and intermediate gold-standard states.

Baselines We evaluated DynaWeb against several representative baselines, described as follows.

1) *Base models*: We include an open-source base agent built on Llama-3.1-8B-Instruct with vanilla chain-of-thought prompting, which serves as a minimal reasoning-enabled baseline without additional training or planning mechanisms. 2) *Proprietary model*: We report results from GPT-4o as a strong closed-source baseline, representing the performance of state-of-the-art proprietary web agents under the same evaluation protocol. *Supervised fine-tuning (SFT)*: We consider supervised fine-tuning on offline successful web navigation trajectories. This includes the NNetNav agents of (Murty et al., 2025), trained on expert-collected demonstrations, as well as Go-Browse (Gandhi and Neubig, 2025), where successful trajectories are automatically collected via structured exploration and used for standard supervised training without reinforcement learning.

4) *Offline reinforcement learning*: The offline-RL baseline refers to Llama-3.1-8B-based agents trained using the WebRL framework by (Qi et al., 2025). In this setting, the agent first interacts with online web environments to collect exploration trajectories, which are then scored by a trained outcome-based reward model and used for offline reinforcement learning. 5) *Inference-time lookahead (ITL)*: The ITL baseline is an Llama-3.1-8B-based agent equipped with a web world model at inference time. The agent performs inference-time policy optimization by simulating multiple candidate future states at each action step and selecting

the action with the highest expected advantage.

4.2 Main Results

As shown in Table 1, **DynaWeb** achieves the best overall performance on WebArena, improving the average success rate from 26.7 (Offline-RL) to **31.0**, a relative gain of **16.1%**. The gains are broad rather than localized: DynaWeb performs best on Reddit, Gitlab, CMS, and Shopping, suggesting that its advantages generalize across diverse interfaces and task types rather than depending on specific site templates. Compared with Offline-RL and ITL, whose gains are often tied to particular exploration settings, DynaWeb more consistently converts partial and imperfect exploration signals into effective learning. This indicates that integrating imagined experience into on-policy optimization provides a stronger training signal for complex multi-step web interaction.

As shown in Table 2, this trend largely transfers to the live, open-world setting of WebVoyager. DynaWeb achieves the best All Rec performance and outperforms competing methods on many websites, including Amazon, Apple, BBC News, Cambridge Dict, Coursera, Google Map, Google Search, HuggingFace, and Wolfram α . These results show that imagination-driven model-based training remains effective even in dynamic, partially observable, and less controlled environments. Meanwhile, the remaining gaps are also informative: DynaWeb underperforms stronger baselines on a few sites such as ArXiv and GitHub, which require longer-horizon planning under highly branching actions and rapidly changing page states. Overall, the results suggest that DynaWeb provides robust and transferable improvements for web agents, while highlighting long-horizon world modeling and dynamic UI handling as key challenges for future work.

5 Analysis

We further analyze DynaWeb to better understand how key design choices in model-based dreaming affect training stability and overall agent performance.

5.1 WM Dream Length Matters

As discussed in Section 3, the length of synthetic trajectories generated through agentworld model interaction (referred to as the *dream length*) plays a central role in determining learning effectiveness. From the perspective of sequential

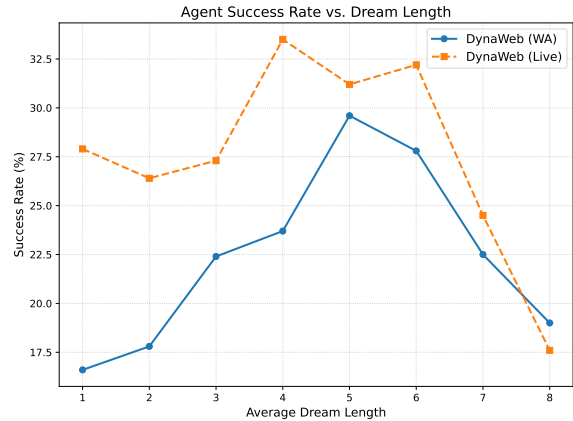


Figure 3: Agent Success Rate vs. Dream Length.

decision making, dream length controls the effective rollout depth, i.e., the number of imagined interaction steps over which the agent can accumulate credit and propagate reward signals. This induces a fundamental trade-off: trajectories that are too short limit the agents ability to learn multi-step action sequences required for non-trivial web tasks, while excessively long rollouts amplify compounding world model errors and hallucinations.

To characterize this trade-off, we perform an ablation study in which DynaWeb agents are trained with varying maximum dream lengths. As shown in Figure 3, we observe a clear performance peak when the average dream length is around 45 steps. Shorter rollouts fail to capture sufficient interaction depth for task completion, whereas longer rollouts suffer from degraded training signals due to accumulated model inaccuracies. These results validate our design choice of restricting dream depth during training, and highlight that effective model-based web RL requires balancing rollout depth against simulation fidelity. Improving the robustness of world models to support longer-horizon, high-fidelity imagined trajectories remains an important direction for future work.

5.2 Effect of Real Interaction Data Percentage

We analyze the role of real-environment interaction data in stabilizing and regularizing model-based web reinforcement learning. While DynaWeb relies heavily on imagined rollouts generated by the web world model, incorporating a portion of ground-truth trajectories from real environments can anchor learning to reliable state transi-

Table 2: Success rate (%) on live websites in WebVoyager. Results are reported across a diverse set of commonly accessed real-world websites. DynaWeb achieves the highest overall success rate and consistently outperforms supervised, RL, and inference-time lookahead baselines on most websites. Best results are in bold, and second-best are underlined.

Method	Live Websites (WebVoyager)												
	All Rec	Ama	Apple	ArXiv	BBC News	Camb Dict	Cour sera	ESPN	Git Hub	G. Map	GSea rch	HF	Wfram α
Llama3.1-8B-Ins+CoT	2.4	2.5	0.0	9.5	7.6	11.8	0.0	0.0	1.2	4.3	0.0	2.1	5.2
GPT-4o	31.1	25.8	29.7	27.9	<u>32.6</u>	41.9	35.3	27.3	26.4	33.5	<u>5.3</u>	19.4	37.1
NNetNav	28.9	24.5	25.2	40.1	32.2	56.8	35.8	15.6	14.1	44.4	0.0	12.9	39.4
Go-Browse	30.4	24.1	26.1	41.8	31.0	54.9	35.9	17.6	16.2	42.1	1.8	18.7	41.2
WebRL	<u>32.6</u>	20.9	27.3	50.4	29.7	<u>57.5</u>	<u>36.6</u>	<u>19.8</u>	<u>18.4</u>	40.6	2.7	<u>20.1</u>	<u>43.7</u>
ITL	25.6	<u>28.7</u>	<u>30.6</u>	43.7	25.1	39.0	35.8	15.4	15.0	<u>46.5</u>	0.0	1.6	18.8
DynaWeb-8B (Ours)	38.7	43.8	34.2	<u>45.7</u>	39.8	61.6	38.3	<u>19.8</u>	11.2	49.5	9.2	22.6	45.4

tions and reward signals.

Figure 4 reports the average success rate of DynaWeb agents under varying proportions of real interaction data. Two clear trends emerge. First, agents trained exclusively on simulated trajectories tend to underperform the SFT baseline, indicating that world model hallucinations and compounding errors can degrade learning when unregularized. Second, introducing approximately 40% real trajectories leads to a substantial performance improvement over SFT, after which additional real data yields diminishing returns.

These results suggest that real interaction data serves as a critical regularizer for imagination-driven training: a moderate amount is sufficient to stabilize learning and correct systematic model biases, while the majority of training signal can still be efficiently obtained from simulated rollouts. This balance highlights the practical advantage of DynaWeb in reducing reliance on costly real-environment interaction without sacrificing performance.

5.3 Essential Role of WM Training

We assess whether explicitly training a web world model is necessary for effective model-based reinforcement learning. To isolate the contribution of world model training, we keep the agent architecture, RL algorithm, and training protocol fixed, and replace the learned DynaWeb world model with a frozen GPT-oss-120b that is prompted to predict next-step observations during imagined rollouts.

As shown in Table 3, agents trained with the supervised world model achieve substantially higher downstream success rates on both WebArena and

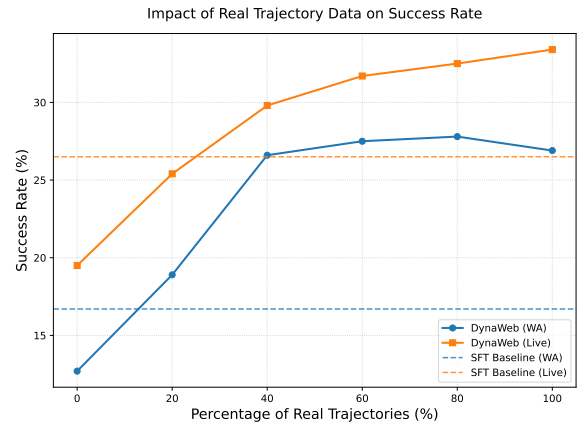


Figure 4: Impact of real trajectory data on success rate.

Table 3: Effect of world model training on downstream agent performance.

Benchmark / World Model	DynaWeb WM	GPT-oss-120b
WebArena (Simulated)	31.0	20.9
WebVoyager (Live)	35.4	28.6

WebVoyager. The large performance gap indicates that strong general-purpose LLM priors alone are insufficient to function as a reliable simulator for imagination-driven RL in web environments without learning environment-specific transition dynamics. Crucially, this result suggests that the effectiveness of DynaWeb does not arise merely from increased model capacity or prompting, but from grounding policy optimization in a world model explicitly trained to capture web interaction dynamics.

6 Conclusion

We introduce **DynaWeb**, a model-based RL framework that trains web agents through imagination rather than live interaction. By learning a web world model and mixing policy-driven imagined rollouts with real expert trajectories, DynaWeb enables on-policy optimization while reducing real-environment cost and risk. Across WebArena and WebVoyager, it consistently improves strong open-source agents.

Our analysis identifies key factors for imagination-driven training: rollout length, real expert data as regularization, and explicitly trained world models. These results suggest that model-based imagination offers a practical and scalable path for training web agents, and point toward world-model-centric learning as a promising direction for long-horizon decision making in complex environments.

Limitations

Our experiments are limited to text-based accessibility-tree environments and relatively small open-source agents. Extending DynaWeb to multimodal web interaction and larger-scale agent training remains important future work. In addition, although DynaWeb reduces reliance on live-web interaction, the quality of imagined rollouts still depends on the fidelity of the learned world model, especially for long-horizon or highly dynamic web transitions.

References

- 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.
- Hyunjoo Chae, Namyoun 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, Samuel 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, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025a. [Webevolver: Enhancing web agent self-improvement with coevolving world model](#). *arXiv preprint arXiv:2504.21024*.
- Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, and 1 others. 2025b. [Cognitive kernel-pro: A framework for deep research agents and agent foundation models training](#). *arXiv preprint arXiv:2508.00414*.
- Apurva Gandhi and Graham Neubig. 2025. [Go-browse: Training web agents with structured exploration](#). *Preprint*, arXiv:2506.03533.
- Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qichen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. [Webwatcher: Breaking new frontier of vision-language deep research agent](#). *Preprint*, arXiv:2508.05748.
- 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.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2020. [Dream to control: Learning behaviors by latent imagination](#). *Preprint*, arXiv:1912.01603.
- 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. 2024. *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.
- 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.
- Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. *Autowebglm: A large language model-based web navigating agent*. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, page 52955306, New York, NY, USA. Association for Computing Machinery.
- 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*.
- MiroMindAI. 2025. *Miroflow: A consistent agent framework with reproducible performance*. <https://github.com/MiroMindAI/MiroFlow>. Accessed: [Insert date of access here].
- 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.
- 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.
- 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.
- Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jidai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. *Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning*. *Preprint*, arXiv:2411.02337.
- 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*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. *Deepseekmath: Pushing the limits of mathematical reasoning in open language models*. *Preprint*, arXiv:2402.03300.
- 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.
- Richard S Sutton. 1991. *Dyna, an integrated architecture for learning, planning, and reacting*. *ACM Sigart Bulletin*, 2(4):160–163.

- 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.
- Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. 2025. [Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning](#). *Preprint*, arXiv:2505.16421.
- 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.
- 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.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. [Dapo: An open-source llm reinforcement learning system at scale](#). *Preprint*, arXiv:2503.14476.
- 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.
- 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.
- Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. 2024c. [Mmina: Benchmarking multihop multimodal internet agents](#). *CoRR*, abs/2404.09992.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. [Group sequence policy optimization](#). *Preprint*, arXiv:2507.18071.
- 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.
- Yuchen Zhuang, Di Jin, Jiaao Chen, Wenqi Shi, Hanrui Wang, and Chao Zhang. 2025. [Workforceagent-r1: Incentivizing reasoning capability in llm-based web agents via reinforcement learning](#). *Preprint*, arXiv:2505.22942.

A Error Analysis of Local vs. Full-Page Transition Prediction

Web state transitions exhibit heterogeneous difficulty. While many actions induce localized updates on the current page, others lead to full page transitions that substantially alter the entire accessibility tree. These two regimes differ significantly in terms of prediction complexity and error characteristics.

To better characterize this distinction, we partition the evaluation set into two subsets: (i) non-full-page-transition actions, where the resulting state can be described as a localized modification of the current page, and (ii) full page transition actions, which involve navigation to a substantially different page state.

For non-full-page-transition actions, we evaluate prediction quality using a sequence-level match rate against the ground-truth state-change description. Under this metric, the world model achieves a match rate of 97%, indicating that it is highly reliable in modeling local updates, which constitute the majority of interaction steps.

We then analyze the more challenging subset of full page transition actions, which account for approximately 36% of the dataset. In this regime, exact matching becomes less informative due to the large structural differences between states. Instead, we adopt an LLM-as-a-judge protocol using GPT-5 to assess prediction quality along multiple dimensions:

- *Action semantics*: correctness of understanding the intended effect of the action;
- *Causal reasoning*: validity and consistency of the inferred transition logic;
- *Resulting page correctness*: alignment of the predicted page with the ground-truth state;
- *Faithfulness*: absence of hallucinated elements or content that should not appear.

Each dimension is scored in $[0, 1]$, and the overall score is obtained by summing across the four criteria, yielding a final score in $[0, 4]$. Under this protocol, the world model achieves an average score of 3.4/4 on full page transition actions.

Further inspection shows that, although discrepancies with the ground-truth accessibility tree occasionally arise, only about 12% of cases are judged as completely incorrect, i.e., predictions that fail on multiple core dimensions and result in substantially wrong page states. The remaining cases typically preserve the high-level structure and semantics of the transition, despite minor inconsistencies at the element level.

Overall, these results indicate that the world model maintains high fidelity on local transitions and remains reasonably robust under full page transitions, despite the increased difficulty of the latter. This gap in difficulty also helps explain the sensitivity to rollout length observed in Section 5.1, as errors in full page transitions are more likely to propagate and compound over multi-step imagination.

B More Details About DynaWeb

B.1 WebArena Training Prompt

We provide the full system prompt used to train our web agent on WebArena. The prompt defines the agents role, available actions, observation format, and task completion criteria, and is used consistently across both real-environment interaction and imagined rollouts generated by the web world model. This prompt is fixed throughout training and evaluation to ensure reproducibility.

B.2 World Model System Prompt

We provide the full system prompt used by the web world model. The prompt specifies the input information available to the model, including the user objective, current webpage state, and executed actions, as well as the required output format for predicting web state changes and the resulting next-step accessibility tree. This prompt is fixed throughout training and inference, and is shared across all experiments to ensure reproducibility.

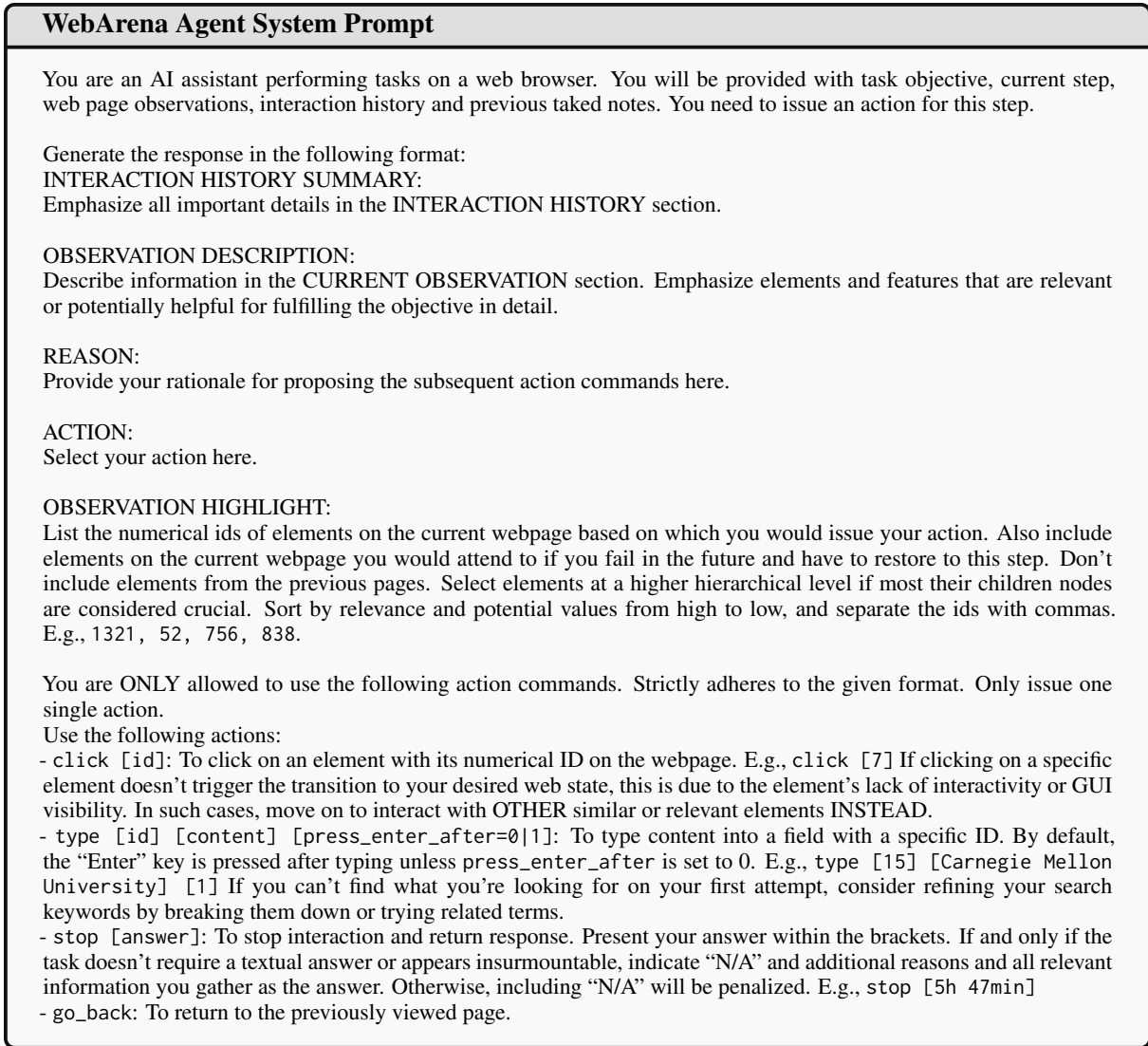


Figure 5: System prompt used for training and evaluation of the WebArena agent.

B.3 Training Recipe and Hyperparameters

This subsection summarizes the training recipe used for DynaWeb agent optimization. Our training follows an online reinforcement learning setup in a synthetic web environment powered by a learned world model. The agent is initialized from a supervised web navigation model and optimized using Group Sequence Policy Optimization (GSPO) with sparse terminal rewards. Imagined rollouts generated by the world model are interleaved with real expert trajectories to stabilize learning.

Web World Model System Prompt

You are an intelligent agent that predicts next state from given current action in a web environment, with your own logical reasoning.

Here's the information you'll have:

The user's objective: This is the task you're trying to complete.

The current web page's accessibility tree: This is a simplified representation of the webpage, providing key information.

The current web page's URL: This is the page you're currently navigating.

The previous action: This is the action you just performed in the previous step. It may be helpful to track your progress.

The current action: This is the current action that you performed to achieve the user's objective in the current web page's accessibility tree.

The format of previous actions can fall into several categories:

Page Operation Actions:

click [id]: This action clicks on an element with a specific id on the webpage.

type [id] [content]: Use this to type the content into the field with id. By default, the 'Enter' key is pressed after typing unless press_enter_after is set to 0, i.e., type [id] [content] [0].

hover [id]: Hover over an element with id.

press [key_comb]: Simulates the pressing of a key combination on the keyboard (e.g., Ctrl+v).

scroll [down] or scroll [up]: Scroll the page up or down.

Tab Management Actions:

new_tab: Open a new, empty browser tab.

tab_focus [tab_index]: Switch the browser's focus to a specific tab using its index.

close_tab: Close the currently active tab.

URL Navigation Actions:

goto [url]: Navigate to a specific URL.

go_back: Navigate to the previously viewed page.

go_forward: Navigate to the next page (if a previous 'go_back' action was performed)

Completion Action:

stop [answer]: Done when you believe the task is complete.

Given the information above, you should first perform reasoning to predict expected changes on the current web page's accessibility tree,

and then generate the resulting next web page's accessibility tree based on your predicted web page changes.

Generate your answer in the following format:

[Web state changes]

changes

[Next page accessibility tree]

next_acc_tree

where changes are the predicted web page changes, and next_acc_tree is your predicted next page accessibility tree. For next_acc_tree, you MUST generate a valid accessibility tree based on your predicted changes, do NOT output summary descriptions of how the next_acc_tree will change.

If the full predicted next_acc_tree is too long, you should output a pruned tree by removing unimportant elements that the web agent will unlikely use in subsequent action steps.

Meanwhile, you should NOT omit key unchanged elements that the web agent might interact with in future time steps.

Note: Even if the web page does not change, you must still output the complete next_acc_tree !

Figure 6: System prompt used to train the web world model for predicting next-step accessibility trees from actions and current observations.

Category	Setting
Backbone policy model	LLaMA-3.1-8B-Instruct
Optimization algorithm	GSPO (sequence-level policy optimization)
Advantage estimator	GRPO-style estimator
Policy learning rate	1×10^{-6}
Entropy coefficient	0
KL regularization	Disabled
Training epochs	10
Train batch size	4
Validation batch size	32
Max prompt length	32,000 tokens
Max response length	16,000 tokens
Max tokens per GPU (PPO)	48,000
Rollout engine	vLLM (asynchronous)
Rollout samples per prompt (n)	8
Rollout temperature	0.7
Rollout top- p	0.9
Validation top- p	0.8
Validation top- k	20
World model max steps	10
Agent max steps	10
World model max tokens	8,192
World model temperature	0.7
World model top- p	0.9
Precision	bfloat16
Gradient checkpointing	Enabled
FSDP parameter offloading	Enabled
Optimizer offloading	Enabled
GPUs	$8 \times$ H100 (single node)

Table 4: Key hyperparameters for DynaWeb agent training.