World Models for Web Agents

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large Language Models (LLMs) have recently advanced to power autonomous web agents. However, they still struggle with long-horizon tasks, often making mistakes such as repeating unnecessary actions. An LLM-based agent might fail to recognize that an item has already been added to a shopping cart and attempt to click the 'add' button again. In contrast, humans easily identify when an item has been added, as they maintain an awareness of the task progression when interacting with the web interface, rarely repeating such actions. This distinction arises from the presence of a *world model* in humans (*i.e.*, an internal representation that simulates interactions with the environment) and its absence in current LLM-based agents [LeCun, 2022]. Realizing this absence, we propose World-Model-Augmented (WMA) Web Agents, which integrate world models to enhance the decision-making capabilities of LLM-based agents. We introduce a novel mechanism allowing agents to focus on state transition information for making informed action choices. Evaluations on WebArena prove that WMA Web Agent outperforms existing baselines, such as the Tree Search Agent [Koh et al., 2024b], by improving action-selection accuracy and reducing errors in web navigation tasks. We present the first successful integration of world models in LLM-based web agents, suggesting a guidance for effective automation in dynamic web environments.

1 Introduction

Advancements in Large language models (LLMs) have made them increasingly attractive for automating complex tasks, such as web navigation [Shi et al., 2017, Kim et al., 2024]. These models, when used as digital agents, can generate action sequences (*e.g.*, `click [33]`) to accomplish user-defined goals. Despite their success in simple, short-term tasks [Yao et al., 2022], LLM-based agents face significant challenges in more complex, long-horizon environments like WebArena [Zhou et al., 2023]. For example, while humans excel at web navigation tasks, web agents score only 14.3% accuracy compared to 84.2% for humans [Zhou et al., 2023]. This stark performance gap raises a critical question: *Why do LLM-based agents, despite their advancements, still struggle to match human-level proficiency in web navigation?*

A key reason for this shortfall lies in how machines and humans approach complex tasks differently. Human gather background knowledge about how the world works through observation and comparably interactions in a task-independent, unsupervised manner [LeCun, 2022]. This provides a foundation for *world models* in humans — internal representations of how actions affect the environment [Craik, 1944, Jonassen and Henning, 1996, Ha and Schmidhuber, 2018]. These world models allow humans to predict the outcomes of their actions, enabling better decision-making in dynamic environments. Consider a task of booking a non-refundable flight ticket. Humans intuitively understand the binding nature of such transaction and make careful decisions to avoid mistakes. In contrast, existing LLM-based agents tend to operate in a reactive manner, relying heavily on trial-and-error. This approach introduces significant risks in real-world scenarios, such as making

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

irreversible decisions (*e.g.*, purchasing non-refundable flight tickets). Koh et al. [2024b] attempts to address this issue with an inference-time tree search algorithm. While this approach improves decision-making during inference time through multi-step planning, it still relies on trial-and-error which makes it prone to irreversible and destructive actions.

Recent research [Levine, 2021, LeCun, 2022] suggests that the absence of world models in machine intelligence is a fundamental limitation that hinders their performance as autonomous agents for long-horizon tasks. Acknowledging such absence, fields like robotics and deep reinforcement learning (RL) in game environment readily adopted world models. In robotics, systems like UniPi [Du et al., 2023] and UniSim [Yang et al., 2024] leverage world models to enhance decision-making and generalization through text-to-video decision-making and dynamic interaction simulations. In game environment, the Dreamer series [Hafner et al., 2020a, 2022, 2024] use world models to predict future states and optimize policy using imagined rollouts in a compact latent space, therefore enabling fast learning in real-world environments. Both fields require a deep understanding of *environment dynamics*, where actions taken by the agent continually reshape the environment. These examples underscore the transformative potential of world models in bridging the performance gap between humans and autonomous agents. We recognize such potential of world models, and hypothesize that expanding its application to the web environment will help LLM-based web agents to select proper actions and reduce the risk of destructive outcomes that often occur in traditional trial-and-error approaches.

To this end, we introduce World-Model-Augmented (WMA) Web Agent, a LLM-based web agent with world model that compensates for the limited awareness of environment dynamics in vanilla LLMs during long-horizon tasks. Instead of providing naive information about a single static webpage, we present a novel abstraction scheme of the state observation for training our world model. This scheme specifically captures the state difference incurred by transition. We also present how the world model can be used to update action-selection policy without further training. Taking full advantage of our framework, WMA Web Agent chooses the optimal action for the best outcome.

Experiments on WebArena [Zhou et al., 2023] show that our WMA Web Agent is significantly more accurate in their action-selection policy compared to baseline agents. We confirm that the world model trained within our framework can accurately simulate action execution, outperforming baselines such as naively prompted LLMs. Results of our experiments underscore the promising potential of world models in web navigation tasks. As the first work to introduce world models into web agents, we expect to open the doors for a more reliable and safer web navigation experience to the users with satisfying performance.

The key contributions of our study are as follows:

- We introduce WMA Web Agent, *the first* to incorporate a world model into web agents, enabling policy adaptation through simulated environment feedback.
- We present a novel observation abstraction scheme focused on *state differences from transitions*, designed to increase information density for LLMs.
- Through extensive experiments, we validate that our world model significantly improves the agent's action-selecting policy. We also demonstrate that access to the predicted next state is crucial for accurately estimating the reward of each sampled action.

2 Related Work

Web Agent Benchmarks. Many benchmarks have been introduced to evaluate LLM-based agents' ability in web navigation [Kim et al., 2024]. MiniWoB [Shi et al., 2017] and MiniWoB++ [Liu et al., 2018] are among the first widely adopted benchmarks. More recently, WebShop [Yao et al., 2022] simulates e-commerce environments where agents are tested to execute tasks on the web based on given text instructions. These early benchmarks lay the groundwork for evaluating web agents. However, they are limited to specific and constrained environments. For more generalizable evaluations, Mind2Web [Deng et al., 2024] curates web tasks across various domains, and WebArena [Zhou et al., 2023] further emphasizes functional correctness and more realistic scenarios such as posting AI-related articles on Reddit.[1] Since WebArena closely resembles the complexity of real-world web interactions, we adopt it for our evaluation.

[1] `https://www.reddit.com/`

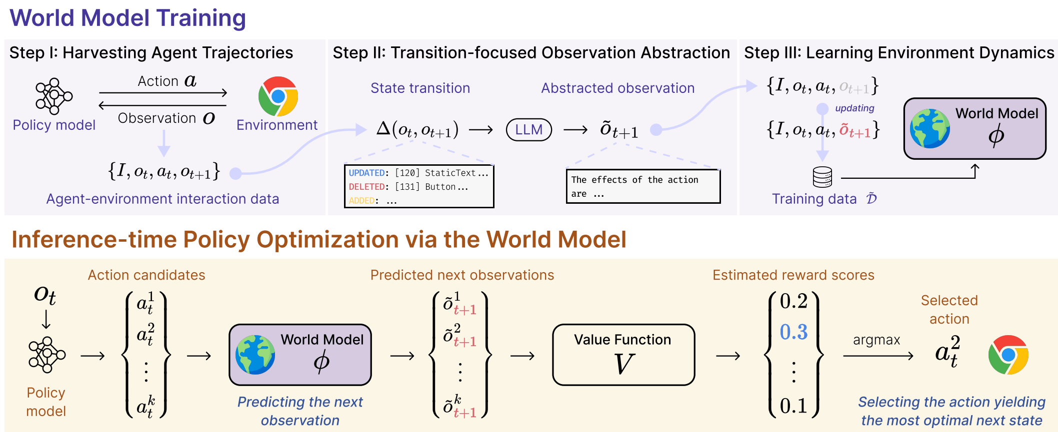


Figure 1: Overview of our framework. We start by collecting a dataset for training the world model (Top). For effective learning and prediction on environment dynamics, we abstract the accessibility tree to free-form description with a specific focus on state transition. Then, we perform inference-time policy optimization by choosing the optimal action leading to the optimal next state (Bottom).

LLM-based Web Agents. In recent years, LLM-based agents have become popular in the web navigation domain. However, since many powerful proprietary LLMs do not provide access to model parameters, many studies of web navigation have been focusing on training-free methods where LLMs directly learn from user inputs (*i.e.*, prompts) without task-specific training [Sodhi et al., 2023, Zheng et al., 2023]. For instance, Wilbur [Lutz et al., 2024] and Agent Workflow Memory [Wang et al., 2024b] leverage a verification model [Pan et al., 2024b] with prompt-based methods to collect successful trajectory data for guiding the agent's policy at inference time. AutoEval [Pan et al., 2024b] and Tree Search Agent [Koh et al., 2024b] increase the number of trials and reasoning paths, further improving system performance. However, due to their trial-and-error nature, these approaches can not only be computationally inefficient in gathering trajectories as tasks become more complex but also are more prone to undesired results (*e.g.*, booking a non-refundable ticket). Our WMA Web Agent reduce such risks via a *world model*, which predicts future observations and their rewards before actually making an action.

World Model in Building Autonomous Agents. *World models* refer to systems that generate internal representations of the world, predicting the effects of their actions on environments [LeCun, 2022]. In RL, simulating observations and environmental feedback using world models allow the policy model to learn [Sutton, 1990] or plan [Ha and Schmidhuber, 2018, Hafner et al., 2019b] without actually interacting with the environment. While some world models are trained with raw observations [Oh et al., 2015, Chiappa et al., 2017], others are built on latent representations [Hafner et al., 2019a, 2020b]. For instance, in the image domain Hafner et al. [2020b] train a world model by training it to first compute a posterior stochastic state based on the current image and then a prior stochastic state that tries to predict the posterior without access to the image. Within the field of LLMs, Zhang et al. [2024] converts visual observations into natural language and employs an LLM-based world model for text-based games, and Wang et al. [2024a] further converts observations into a structural format (*e.g.*, JSON), improving LLMs' reasoning over state transition functions. In web navigation, environments are built upon not only natural language but also more complex text modalities such as HTML and DOM trees. We address this by transforming them to a novel free-from description, highlighting the state difference between each time step.

3 World-Model-Augmented Web Agents

The key motivation of our work is to teach web agents to produce actions with an increased awareness of environment dynamics (*i.e.*, cause-and-effect relationships between actions and the web environment) and thereby improve their ability to navigate complex environments. We introduce World-Model-Augmented (WMA) Web Agent, which integrates the concept of a world model aligned to our motivation. First, we build a world model by collecting data from interactions between the agent and the environment. Then, we train the model on the collected dataset. During inference time, our WMA Web Agent improves its action-selection policy by using the world model, with enhanced understanding of the environment dynamics.

Problem Description. As agents in most real-world scenarios frequently deal with information that is limited, unclear, or incomplete, we consider a Partially Observable Markov Decision Process (POMDP) environment $\mathcal{E}$ with a hidden state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, and transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. Action space $\mathcal{A}$ is defined as language-guided web actions, such as `CLICK`, `TYPE`, and `HOVER` with action description. Observation space $\mathcal{O}$ is an accessibility tree of the webpage, a simplified version of DOM tree [Zhou et al., 2023]. The agent is asked to produce a sequence of actions to reach the goal state by interacting with the environment $\mathcal{E}$. In a POMDP, the agent receives partial observations o_{t+1} from $\mathcal{E}$ after the action a_t has taken in place. Such state transition from s_t to s_{t+1} is managed by the transition function $\mathcal{T}$ of the environment.

3.1 Training a World Model

3.1.1 Step I: Harvesting Agent-Environment Interaction Data

Our goal in this step is to construct a training dataset $\mathcal{D} = \{I, o_t, a_t, o_{t+1}\}$ for world model ϕ. The ground-truth next state data is collected from the browser environment $\mathcal{E}$. Generated by the interaction between the the agent θ and $\mathcal{E}$, we construct $\tilde{\mathcal{D}}$ from trajectory $\tau = \{o_0, a_1, o_1, ..., a_n\}$ based on synthetic user instructions I.

To illustrate the details of how our dataset $\tilde{\mathcal{D}}$ is constructed, we explain the process of augmenting WebArena dataset [Zhou et al., 2023]. We base our augmentation strategy on existing remedies used when no annotated user instruction exists for a particular website. Because the original Webarena dataset lacks diversity in user instructions I for it to be fully robust, we augment it by synthetically generating I using an LLM. Our strategy also includes manually inspecting the quality of synthetic I to verify whether they are feasible in the given web environment. After creating a diverse set of I, we collect trajectories τ from interactions between θ and $\mathcal{E}$ by using prompting methods performing each synthetic I. To ensure the diversity of trajectories, we sample k number of trajectories for each I.

3.1.2 Step II: Transition-focused Observation Abstraction

Accessibility tree, a compact list of elements annotated with element id [Zhou et al., 2023], is the most common format for representing observation o in web environments due to its relative simplicity compared to the raw HTML format [Drouin et al., 2024, Koh et al., 2024a]. However, we still deem this format as suboptimal for training language models to learn the dynamics of the web environment for two reasons. First, although recent LLMs have advanced to process extremely long context lengths [Gu and Dao, 2023], the accessibility format results in observations quite burdensome, with about 4000 tokens long on average (Figure 2). Second, accessibility format only contains static information about a single page, with little or no information on state transition.

In RL settings with world models such as in robotics and game environments, estimated latent vector often replaces the full observation of visual input to avoid excessive memory footprint and promote effective learning [Doerr et al., 2018, Hafner et al., 2019c]. Motivated by such simplified replacement of the original observation, we take a similar approach. In our framework, the original representation o (*i.e.*, accessibility tree) is abstracted into a compact yet more informative format for LLMs' comprehension.

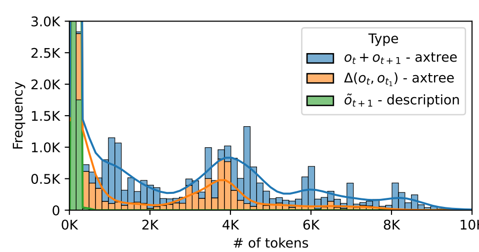


Figure 2: Distribution of sequence length for each representation type of observation. Accessibility tree format (axtree) requires an extremely long input context length.

We use free-form description for abstracting the state in a more flexible and compact manner with more information gain compared to a naive accessibility tree or HTML representation. Previous research naively summarizes [Sridhar et al., 2023] and retrieves [Deng et al., 2024] state observations, focusing only on reducing the input length. This causes the generated summary to be repetitive and uninformative sentences about the current static webpage. Therefore,

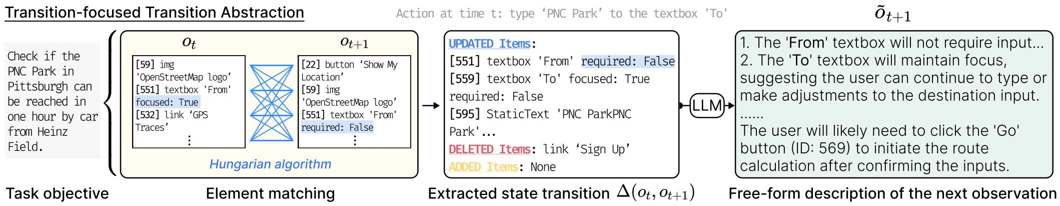


Figure 3: The overview of transition-focused observation abstraction. Through this process, we increase the information density to allow LLMs to understand and learn dynamics effectively.

traditional state representation approaches are severely limited in capturing the critical differences between the dynamic transitions between consecutive states.

Instead of summarizing o_{t+1}, we provide a mechanism that formulates a free-form description that focuses on the state differences incurred from the transition between o_t and o_{t+1} (*i.e.*, $\Delta(o_t, o_{t+1})$) for generating $\tilde{o}_{t+1}$. To obtain $\Delta(o_t, o_{t+1})$, we use the Hungarian's algorithm that calculates a cost matrix for matching elements between o_t and o_{t+1}. Details for the full algorithm is provided in Algorithm 1. Then, the mapped results of the algorithm are used to construct a sequence that shows either updated, deleted, and added elements respectively denoted by the identifiers `UPDATED`, `DELETED`, and `ADDED`. Finally, we ask an LLM to generate a free-form description focusing on the effect of a_t on $\mathcal{E}$ by using $\Delta(o_t, o_{t+1})$. Overview of observation abstraction is shown in Figure 3.

3.1.3 Step III: Learning Environment Dynamics

Using the dataset $\tilde{\mathcal{D}}$ constructed from the previous steps, we train world model ϕ to learn environment dynamics. The primary function of ϕ is to predict the abstracted observation $\tilde{o}$ of the next state s_{t+1}, given three inputs: the user instruction I, the current observation o_t, and the current action a_t. ϕ is trained to optimize the following objective function:

$$\mathcal{L}_\phi = -\log \sum_{(\tilde{o},o,a,I)\in\tilde{\mathcal{D}}} p(\tilde{o}_{t+1}|o_t, a_t, I) \tag{1}$$

Through this training process, the world model ϕ learns to model the environment dynamics in response to actions taken by the agent θ. In essence, it learns to approximate the transition function T that governs how the environment evolves in response to actions.

3.2 Inference-time Policy Optimization with World Model

The learned dynamics from the trained world model ϕ is incorporated by the agent θ during inference time. Our goal is to find an optimal policy a_t for the current timestep t while considering its effect on the environment. By simulating the transition $\mathcal{T}(s_t, a_t)$ using our world model ϕ from §3.1, we estimate the results of a_t on the environment. Overview of our inference pipeline is depicted in Figure 1 (Bottom).

We begin by sampling k distinct action candidates $\{a_t^1, a_t^2, ..., a_t^k\}$ from the agent's policy distribution π_θ using top-p decoding algorithm [Holtzman et al., 2019], allowing exploration of diverse next states s_{t+1} [Wang et al., 2022]. Then, with the world model ϕ, we simulate the execution of a_t to access next state information of s_{t+1} without altering the actual environment. We obtain k number of observations $\tilde{o}_{t+1}$ of the future timestep $t+1$ for each sampled action candidates:

$$\{\tilde{o}^i_{t+1}\}^k_{i=1} = \{\phi(o_t, a^i_t, I)\}^k_{i=1} \tag{2}$$

Lastly, we choose an action that leads to the most successful future state s_{t+1}, *i.e.*, that yields the highest reward score. For evaluating the states, we adopt an off-the-shelf LLM used as a value function $V(\cdot)$ in Koh et al. [2024b] to evaluate the simulated next observations based on its accomplishment in its progression towards the user-defined goal I. This value function outputs a scalar reward score $r \in [0, 1]$. Specifically, we select the best action $\hat{a}_t$ directly affecting $\mathcal{E}$ using:

$$\hat{a}_t = \underset{a_t\in\{a^1_t,\ldots,a^k_t\}}{\operatorname{argmax}} V(I, o_t, a_t, \phi(o_t, a_t, I)) \tag{3}$$

Table 1: Trajectory-wise evaluation results on WebArena [Zhou et al., 2023].

Agent LLM	Method	Max Actions	Success Rate		Δ
			Vanilla	+Method	
GPT-4	AutoEval [Pan et al., 2024b]	30	15.6%	20.2%	-
	BrowserGym (GPT-4) [Drouin et al., 2024]		14.9%	23.5%	-
	SteP [Sodhi et al., 2023]		14.9%	**35.8%**	-
GPT-4o	Tree Search Agent [Koh et al., 2024b]	5	15.0%	19.2%	+28.0%
	WMA (ours)	5	11.7%	15.5%	+32.5%
GPT-4o-mini	**WMA (ours)**	5	7.1%	13.7%	+93.0%

Table 2: Success rates and relative change (Δ) of the WMA agent on WA websites.

Website	Vanilla CoT	WMA	Δ
CMS	8.2%	9.3%	+13%
Map	0.9%	22.3%	+2378%
Shopping	18.8%	19.3%	+3%
Reddit	0.0%	5.3%	-
Gitlab	3.1%	8.7%	+181%
Overall	7.1%	12.7%	+79%

This formulation allows the agent to make reasoned decisions from current state and each potential future state pair resulting from each potential action candidates. We highlight that our approach can be adapted to many versions of web agents, including both prompting-based web agents [Pan et al., 2024a, Wang et al., 2024b] or fine-tuned web agents [Gur et al., 2023, Lai et al., 2024].

4 Experiments

4.1 Experimental Setup

Evaluation and Benchmarks. We use two evaluation setups: **(1) end-to-end evaluation**, for evaluating the pass rate of the end-to-end task completion of user instruction, and [Zhou et al., 2023, Lai et al., 2024] **(2) step-wise evaluation**, for calculating the accuracy of selecting the gold action in each step. The end-to-end evaluation test set is provided by the official WebArena benchmark [Zhou et al., 2023]. WebArena is designed to evaluate agents within the provided environment by interacting with it. It covers 812 real-life tasks across five different websites, spanning four key domains – e-commerce, social forums, collaborative software development, and content management.

Agent LLMs. Following Koh et al. [2024b], GPT-4o (`gpt-4o-0513`) is used as our backbone agent tested for WebArena experiments. Additionally, we test with GPT-4o-mini (`gpt-4o-mini-0718`) to explore more resource-efficient configurations for general use.

Baselines. For baseline agents, we incorporate prompting-based LLMs, leaving incorporation of domain-specific (*e.g.*, shopping domain) techniques [Sodhi et al., 2023, Wang et al., 2024b] for future work. AutoEval [Pan et al., 2024b] leverages the critic from VLM evaluator in applying Reflexion [Shinn et al., 2024]. The most competitive baseline is Tree Search Agent [Koh et al., 2024b], which explores multiple trajectories and selects an optimal path using a search algorithm during inference time. The major difference between our WMA Web Agent and the Tree Search Agent is that WMA Web Agent only takes a peek at the future states via simulation and does not actually explore diverse states during inference time.

4.2 Implementation Details

Data Collection. We employ GPT-4o-mini as the agent to gather 14K instances from the WebArena envrionment. To ensure the uniqueness and quality of the collected data, heuristic filtering is applied

Table 3: Head-to-head comparison with Tree Search Agent [Koh et al., 2024b] on the performance, inference time and API cost .

Method	Shop.	Shop. Admin	Reddit	Gitlab	Map	API Cost	Inf. time (sec.)
Tree Search Agent	28.1	16.5	10.5	13.3	25.8	$2.7	678
WMA (ours)	20.8	14.3	10.5	13.3	18.6	$0.4	140

to remove identical instances. This process improves the overall diversity and relevance of the dataset, which is crucial for subsequent analysis. Detailed insights into the size and characteristics of the resulting dataset are discussed in §4.5.

World Model. We use Llama-3.1-8B-Instruct [Dubey et al., 2024] as our backbone LLM for building our world model[2]. When training, we employ QLoRA [Dettmers et al., 2024] and liger kernel [Hsu et al., 2024] to reduce computational cost.

Value Model. We explore two implementation setups for our value model: (1) prompted LLMs to predict the reward score, and (2) fine-tuned LLMs from the Mind2Web [Deng et al., 2024] data. In the latter setting, the reward score is calculated step-by-step based on its progress toward the goal, *i.e.*, $(t+1)/(len(\tau))$ assuming the human-annotated trajectory is the optimal path. Details of the implementation are in Appendix B.4.

4.3 Main Results

As shown in Table 1, our WMA Web Agent significantly improves vanilla agents by far for both GPT-4o-mini (13.7%) and GPT-4o (15.5%) on WebArena benchmark. Our WMA Web Agent outperforms the Tree Search Agent [Koh et al., 2024b], although the latter utilizes oracle observation of future states unlike ours. We provide a more detailed analysis comparing WMA Web Agent and Tree Search Agent in Table 3 and subsection 4.4, proving our method's efficiency.

We also look at the success rates and the relative performance improvements in each domain of the WebArena benchmark, with and without our WMA Web Agent 2. Our method shows 79% increase in performance overall, proving its effectiveness in web navigation in general. It shows significant improvement in domains that are deemed particularly challenging, such as the map domain, followed by Gitlab and Reddit. Also, our solution is also comparably easily integrated with other prompting baselines (*e.g.*, AutoEval [Pan et al., 2024b]).

4.4 Time and Cost Effectiveness of WMA Agents Compared to Tree Search Agent.

We compare our WMA Agent with Tree Search Agent regarding time and API cost efficiency Koh et al. [2024b]. We show the results are shown in Table 3. Tree Search Agent takes about 678 seconds on average for conducting inference on a single instance since it explore diverse future states by interacting with the actual environment. However, WMA Agent takes only 140 seconds per instance by leveraging the simulated environment via the world model. While WMA Agent provide time- and cost-efficient exploration, it show comparable performance to Tree Search Agent in Reddit, Gitlab, and Shopping Admin domains.

4.5 Ablations

Observation Abstraction. We evaluate the effectiveness of transition-focused observation abstraction format (described in §3.1.2) for training and predicting with our world model. Our approach is compared to a world model trained on full accessibility tree. The results of this comparison are presented in Table 4. Results prove that attempting to predict the full accessibility tree impairs the world model's comprehension of the state, compared to our novel abstraction method.

Table 4: Ablation results of observation abstraction.

Method	SR
Vanilla CoT	7.1
w/o observation abstraction	6.4
WMA (ours)	**12.7**

[2]https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct

Table 5: Ablation on the training world model using step-wise evaluation.

World Model	Training	MRR	Hit@1
GPT-4o-mini	✗	55.90	44.0
GPT-4o	✗	68.94	57.0
WMA (ours)	✓	**75.82**	**67.0**

Table 6: Ablation on the training value model.

Value Model	Training	SR
GPT-4o-mini	✗	12.2
Llama-3.1-8B	✓	12.7

Prompted World Model vs. Fine-tuned World Model. Results shown in Table 5, compared to the world model trained on our collected agent trajectories (§3.1) proves that the prompted LLMs do not have enough knowledge on environment dynamics for successful task completion, consistent with our hypothesis.

Choice of Implementation for the Value Model. We explore which implementation of the value model works best for WMA Web Agent. We compare two setups: (1) the prompted value model and (2) the fine-tuned value model. The results are shown in Table 6. Interestingly, we find that training our value model on Mind2Web shows a slightly better performance compared to GPT-4o-mini, which provides a cost-effective option for implementing the value model in WMA Agents.

Access to Next State in Value Score Calculation. To assess the impact of incorporating the next state when calculating the value score, we compare our reward calculation method to a Q-value function approach. Unlike WMA Web Agent, the Q-value function directly predicts the reward score based on the current observation-action pair (o_t, a_t) without the future state. We also compare WMA Web Agent with a setting that uses ground-truth observation of the next state, similar to Koh et al. [2024b]. The results in Table 7 show that the access to the next state plays critical role in accurate prediction on the reward.

Table 7: Step-wise evaluation results that show the importance of the access to the next state.

Method	MRR	Hit@1
w/o o_{t+1}	62.04	45.1
WMA (ours)	**75.82**	**67.0**

5 Discussion and Future Work

Self-refining with the Simulated Environmental Feedback. Currently, we incorporate our world model only for selecting optimal policy at inference time. However, leveraging the simulated feedback from our world model for refining the policy [Wang et al., 2022] might a direction that future work can explore to improve performance.

Improving the Value Models. In this work, we utilize an off-the-shelf value model, as there is no available value model that is known to work well on various websites nor the feedback data for training the model. A promising direction to improve the current value model would be collecting and leveraging a massive dataset using pairwise feedback across diverse web interactions and learning a value model with the data.

6 Conclusion

We present the first framework to incorporate world models into LLM-based web agents, addressing the challenges associated with complex web navigation tasks. Experiments demonstrate that the World-Model-Augmented (WMA) Web Agent significantly improves action-selection policies by enhancing the agent's awareness of environment dynamics. Our results on WebArena show that this approach substantially outperforms baseline LLM-based agents, reducing the need for trial-and-error and mitigating the risk of destructive actions.

The introduction of world models in web agents marks a promising direction for future research in automating complex tasks. By enabling agents to predict the outcomes of their actions, we bridge the gap between human-like decision-making and machine autonomy. Our findings pave the way for developing more robust and safe digital agents capable of performing intricate tasks across dynamic web environments.

References

Silvia Chiappa, Sébastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

Kenneth Craik. The nature of explanation. 1944. URL `https://api.semanticscholar.org/CorpusID:41364251`.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Andreas Doerr, Christian Daniel, Martin Schiegg, Duy Nguyen-Tuong, Stefan Schaal, Marc Toussaint, and Sebastian Trimpe. Probabilistic recurrent state-space models. In *International Conference on Machine Learning*, 2018. URL `https://api.semanticscholar.org/CorpusID:45425492`.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation, 2023. URL `https://arxiv.org/abs/2302.00111`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019b.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2019c. URL `https://api.semanticscholar.org/CorpusID:208547755`.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020a. URL `https://arxiv.org/abs/1912.01603`.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL `https://arxiv.org/abs/2010.02193`.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL `https://arxiv.org/abs/2301.04104`.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, and Siyu Zhu. Liger-kernel: Efficient triton kernels for llm training, 2024. URL `https://github.com/linkedin/Liger-Kernel`.

David H. Jonassen and Philip Henning. Mental models: Knowledge in the head and knowledge in the world. *Educational Technology archive*, 39:37–42, 1996. URL `https://api.semanticscholar.org/CorpusID:140355958`.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024a.

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

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. Autowebglm: Bootstrap and reinforce a large language model-based web navigating agent. *arXiv preprint arXiv:2404.03648*, 2024.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Sergey Levine. Understanding the world through action, 2021. URL `https://arxiv.org/abs/2110.12543`.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*, 2018.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

Michael Lutz, Arth Bohra, Manvel Saroyan, Artem Harutyunyan, and Giovanni Campagna. Wilbur: Adaptive in-context learning for robust and accurate web agents. *arXiv preprint arXiv:2404.05902*, 2024.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024a.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024b.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Paloma Sodhi, SRK Branavan, and Ryan McDonald. Heap: Hierarchical policies for web actions using llms. *arXiv preprint arXiv:2310.03720*, 2023.

Abishek Sridhar, Robert Lo, Frank F Xu, Hao Zhu, and Shuyan Zhou. Hierarchical prompting assists large language model on web navigation. *arXiv preprint arXiv:2305.14257*, 2023.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2:160–163, 1990. URL `https://api.semanticscholar.org/CorpusID:207162288`.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. Can language models serve as text-based world simulators? *arXiv preprint arXiv:2406.06485*, 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators, 2024. URL `https://arxiv.org/abs/2310.06114`.

Shunyu Yao, Howard Yang Chen, John, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. 2022. doi: 10.48550/arXiv.2207.01206.

Alex Zhang, Khanh Nguyen, Jens Tuyls, Albert Lin, and Karthik Narasimhan. Language-guided world models: A model-based approach to ai control. *arXiv preprint arXiv:2402.01695*, 2024.

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

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2023.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Appendix

A Limitation

Modality. As an initial step toward world models for web agents, we mainly focus on building text-based world models. In web navigation, however, visual information also plays a critical role in accurate perception of the environment [Liu et al., 2024, Zheng et al., 2024]. Future work might incorporate visual information in addition to textual information for improving the learning of dynamics in the environment [Koh et al., 2024b].

Multi-step Planning. Our current approach demonstrates that simulating action execution via our world model significantly aids web agents in selecting actions with awareness of environmental dynamics within a single time step. However, the potential of the idea of using world models for web agents extends beyond this single-step prediction. Our model is trained to predict the abstracted next state $\tilde{o}_{t+1}$ from the previous observation o_t and the current action a_t. This world model can be extended for multi-step planning that generates a sequence of actions without interaction with the environment by recursively feeding the predicted state o_{t+1} back into the agent θ as the new observation, along with current a_t action from the agent, we can generate predictions for multiple steps into the future. This capability opens up exciting possibilities for more sophisticated planning strategies with reduced negative impact of repetitive trial-and-error. Future work could explore leveraging this multi-step prediction capability to enable web agents to reason about longer-term consequences of their actions, evaluate complex action sequences, and make more informed decisions in scenarios requiring extended foresight. Additionally, incorporating techniques such as Monte Carlo Tree Search [Koh et al., 2024b] or other planning algorithms could further enhance the agent's ability to navigate complex, multi-step tasks in web environments.

B Implementation Details

B.1 World Model

B.1.1 Dataset Construction

We leverage WebArena environment to collect agent trajectories. In total we obtain 14,200 instances using GPT-4o-mini with CoT prompt provided in Zhou et al. [2023].

Transition-focused Observation Abastraction. For implementing Hungarian algorithm we use munkres python package[3]. We describe the algorithm used for transition-focused observation abstraction in Algorithm 1.

Algorithm 1: Observation Tree State Matching for $\Delta(o_t, o_{t+1})$

Input : States $o_t = [e_0^t, \ldots, e_{n-1}^t], o_{t+1} = [e_0^{t+1}, \ldots, e_{m-1}^{t+1}]$. Each e_i has name n_i, role r_i, location l_i. Weights $\omega_n, \omega_r, \omega_l$.
Output : S_{t+1}^{TaO}

$U \leftarrow \emptyset$
if *len*$(o_{t+1}) \leq \tau \cdot$ *len*(o_t) **then**
  # Construct cost matrix for Hungarian matching
  $C_{i,j} \leftarrow \omega_n \cdot \mathbf{1}_{n_i^t = n_j^{t+1}} + \omega_r \cdot \mathbf{1}_{r_i^t = r_j^{t+1}} + \omega_l \cdot |l_i^t - l_j^{t+1}|$
  # Apply Hungarian algorithm to find optimal matching
  $M^* \leftarrow \underset{M}{\text{argmin}} \sum_{i,j} C_{i,j} \cdot M_{i,j}$
  # Identify unmatched elements
  $U \leftarrow \{j | M_{i,j}^* = 0, \forall i \in \{0, \ldots, n-1\}\}$
end
if *len*$(U) \geq m - n$ *or* $U = \emptyset$ **then**
  $S_{t+1}^{\text{TaO}} \leftarrow o_{t+1}$
else
  # Construct TaO state based on unmatched and nearby elements
  $S_{t+1}^{\text{TaO}} \leftarrow [e_j^{t+1} | j \in U$ or (len$(U) \leq x$ and $\min_{u \in U} |u - j| \leq y)]$
end

B.2 Training

We use hyperparameters of 2 epochs, 10^{-5} learning rate to train our world model and the value model. For efficient training, we use QLoRA and liger kernel. The models are trained using 8 RTX 4090 GPUs and the training took around 3 GPU hours.

B.3 Inference

We use top-p decoding with $p = 0.7$ for sampling 20 actions from the model.

B.4 WebArena Environment

To ensure fair comparison and reproducibility, we conducted our experiments using the WebArena environment. Specifically, we utilized an Amazon Web Services (AWS) EC2 instance pre-configured with the Docker environment for WebArena[4] This setup is identical to the experimental configuration employed by Zhou et al. [2023] in their original study. By using this standardized environment, we maintain consistency with previous research and facilitate direct comparisons of our results with those reported in the literature. The WebArena Docker environment encapsulates all necessary

[3] https://pypi.org/project/munkres/

[4] https://github.com/web-arena-x/webarena/blob/main/environment_docker/README.md#pre-installed-amazon-machine-image

dependencies, web interfaces, and evaluation metrics, ensuring that our experiments are conducted under controlled and replicable conditions.

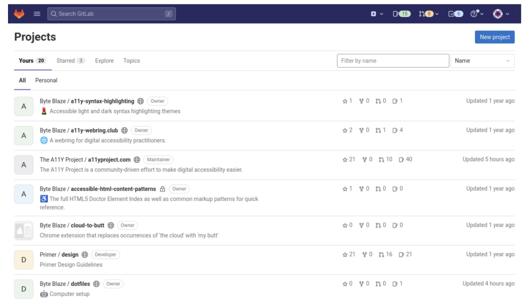
C Qualitative Examples

User intent: Open my latest updated issue that has keyword "dependency" in its title to check if it is closed

Observation

```
Tab 0 (current): Projects Â· Dashboard Â· GitLab

[7] RootWebArea 'Projects Â· Dashboard Â· GitLab' focused: True url:
http://ec2-3-139-65-91.us-east-2.compute.amazonaws.com:8023/
 [21] link 'Skip to content' url: http://ec2-3-139-65-91.us-
east-2.compute.amazonaws.com:8023/#content-body
 [27] link 'Dashboard' url: http://ec2-3-139-65-91.us-
east-2.compute.amazonaws.com:8023/
  [28] image ''
 [1941] button '' hasPopup: menu expanded: False
 [57] image ''
 [59] textbox 'Search GitLab' required: False
 [5] StaticText '/'
 [71] link 'Create new...' url: http://ec2-3-139-65-91.us-
east-2.compute.amazonaws.com:8023/projects/new
  [72] image ''
  [74] image ''
 [88] link 'Issues' url: http://ec2-3-139-65-91.us-
east-2.compute.amazonaws.com:8023/dashboard/issues?
assignee_username=byteblaze
  [89] image ''
  [91] generic '15 assigned issues'
... (omitted)
```



Action Candidate 1 (✅ **Selected**)
Action: click [88]
Next state prediction:
Let's think step by step about the expected changes after performing the current action of clicking on the 'Issues' link.

Key changes in the accessibility tree based on this action would include:
1. The RootWebArea title has changed to 'Issues Â· Dashboard Â· GitLab', indicating a transition from the project dashboard to the issues dashboard.
2. A new button for managing issues labeled 'New issue' has been added to the page, allowing users to create new issues directly.
3. A custom checkbox for filtering issues labeled 'Open only' has been introduced, enabling users to set the visibility of their issues.
4. A new static text element states 'Age sort order', which indicates that users can sort issues based on their creation date.
5. A new button labeled 'Filter by label' has been added, allowing users to apply labels to issues.
6. A new button option 'Link' is introduced for each issue view link, indicating that users can access the corresponding issue details.
7. A new link labeled 'Epoch talks' has been added, providing access to the specific issue.
8. The issues list has been updated to include relevant fields such as 'Title', 'Description', and 'Updated', showcasing the current status of each issue.
9. The 'Nov 12, 2022' and 'Nov 30, 2022' updated time fields provide context on when specific issues were last updated.

Based on these changes, the next state of the webpage will reflect the issues dashboard where users can view, filter, and manage their issues related to their assigned projects.

[Next State] The expected effect is that:
1. The page will now display a list of issues relevant to the user, with fields marked as 'Title', 'Description', and 'Updated' to provide **details about each issue.**
2. Users can filter and sort the issues using the available options, such as the 'Open only' checkbox and the 'Age sort order' dropdown.
3. Users will have the ability to create new issues using the 'New issue' button and will have access to relevant links for each issue.
4. The user can then proceed to find the issue titled 'dependency' from among these issues and check if it is closed.
Value score: 0.5974999999999999

Action Candidate 2
Action: type [59] [dependency] [1]
Next state prediction:
Let's think step by step about the expected changes after performing the action of typing "dependency" into the search box.

Key changes in the accessibility tree based on this action would include:
1. A new StaticText 'dependency' was added, indicating that the search term has been entered successfully.
2. An additional StaticText 'dependency' was inserted, signifying the search is actively filtering the results.
3. A new heading 'Search results for "dependency"' has been created, confirming that the search has been triggered and is returning relevant results.
4. The 'Search GitLab' textbox remains in the same state, but this time it has a new attribute indicating that it is focused, indicating that the user can interact with it.
5. The main content area of the page is now occupied by items specifically related to the search query (dependency).
6. There are new links to individual projects, such as 'dependency-check', 'dependency-parser', and 'dependency-tracker', suggesting that these items match the search term.

[Next State]
The expected effect is that the webpage will display detailed search results related to the keyword "dependency". This will allow the user to view **project listings** such as 'dependency-check', 'dependency-parser', and 'dependency-tracker', which may potentially contain the updated issue with the keyword in the title. To proceed, the user will be able to check each of these project results to determine if the latest issues are closed.
Value score: 0.41574999999999995

Figure 4: An example inference of WMA Agents in Gitlab website (instance #175). Among the sampled actions from the policy model (*i.e.*, GPT-4o), WMA Agent selects the more proper action `click [88]` that directly goes to the issue page by leveraging the learned environment dynamics.