R-WOM: RETRIEVAL-AUGMENTED WORLD MODEL FOR COMPUTER-USE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) can serve as world models to enhance agent decision-making in digital environments by simulating future states and predicting action outcomes, potentially eliminating costly trial-and-error exploration. However, this capability is fundamentally limited by LLM's tendency to hallucination and their reliance on static training knowledge, which could lead to compounding errors that inhibit long-horizon simulations. To systematically investigate whether LLMs are appropriate for world modeling, we probe two core capabilities of world models – *future state prediction* and *reward estimation* – through three tasks: next-state identification, full-procedure planning alignment, and milestone transition recognition. Our analysis shows that while LLMs effectively capture immediate next states and identify meaningful state transitions, their performance rapidly degrades in full-procedure planning. This highlights LLMs' limitations in reliably modeling environment dynamics over long horizons. To address these limitations, we propose the Retrieval-augmented World Model (R-WoM), which grounds LLM simulations by incorporating factual, up-to-date knowledge retrieved from external tutorials. Experiments show that R-WoM achieves substantial improvements of up to 25.3% (OSWorld) and 18.1% (WebArena) compared to baselines, with particular advantage in longer-horizon simulations.

1 INTRODUCTION

World models have evolved from early symbolic planning systems to sophisticated neural architectures that learn latent representations of environment dynamics. Model-based reinforcement learning (MBRL) approaches, such as Dreamer v1-3 (Hafner et al., 2019; 2020; 2023) and MuZero (Schrittwieser et al., 2020), learn latent world models to "imagine" trajectories before selecting actions. More recently, Large Language Model (LLM)-based world models (Hao et al., 2023; Wang et al., 2024; Zhang et al., 2024) have emerged as a new paradigm, leveraging large-scale pretraining to reason about action consequences in realistic digital environments. They show particular promise for long-horizon planning for browser and computer-use agents, where mentally simulating future states can mitigate irreversibility and reduce costly trial-and-error.

However, due to their inherent tendency toward hallucination and reliance on static parametric knowledge, LLMs perform world modeling in a fundamentally ungrounded manner. In complex, multi-step tasks, this detachment from the environment's real-time state can trigger cascading errors: the imagined trajectory gradually diverges from actual dynamics, producing simulations that appear coherent but are ultimately unexecutable. This limitation becomes particularly evident in realistic computer-use environments, as illustrated in Figure 1.

To systematically investigate whether LLMs can serve as effective world models, we probe two core capabilities: **future state prediction** and **reward estimation**. We design three evaluation tasks: next-state prediction and full-procedure planning alignment to assess LLMs' future state prediction capability; and milestone transition recognition to assess LLMs' reward estimation capability. Our analysis reveals that while LLMs demonstrate strong short-term dynamics understanding – such as identifying state changes and recognizing transition outcomes – they fail to maintain accuracy in full-procedure planning. This performance degradation over longer-horizon simulations highlights fundamental limitations of LLM-based world modeling.

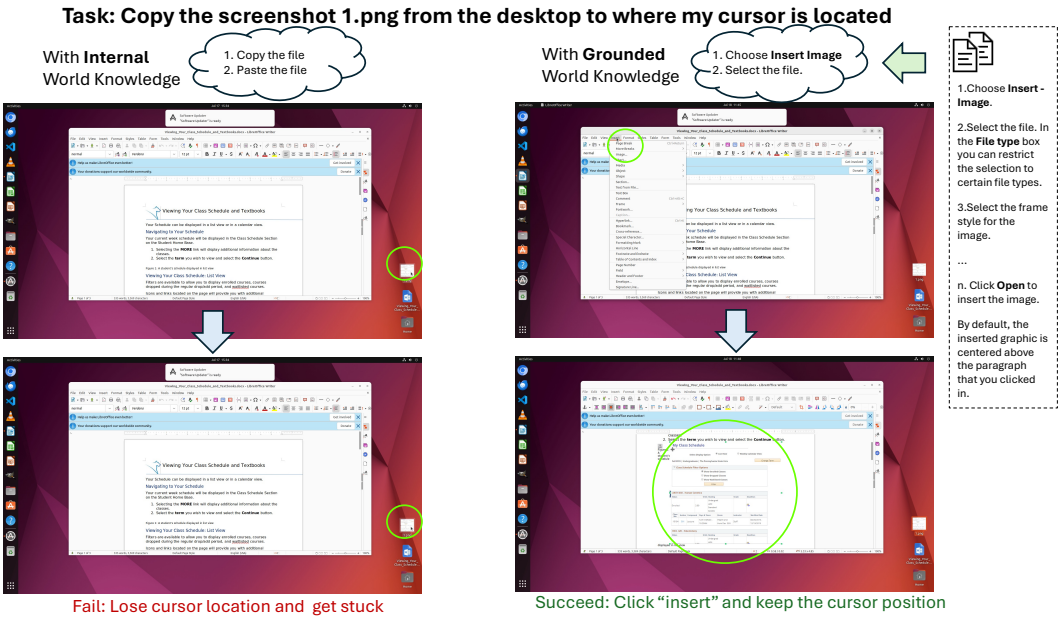


Figure 1: Example task: "Copy the screenshot 1.png from the desktop to where my cursor is located." (**Left:**) Using only internal world knowledge, the agent loses cursor location and gets stuck. (**Right:**) With grounded world knowledge from tutorials, the agent uses the correct "Insert Image" operation while maintaining cursor position. This illustrates how grounding with external knowledge enables more reliable decision-making in realistic environments.

Motivated by these findings, we propose the **Retrieval-augmented World Model (R-WoM)** framework, which enhances LLM-based simulations by grounding them in external knowledge drawn from environment-specific tutorials. The core insight behind R-WoM is that while LLMs possess broad world knowledge from pretraining, they lack the specific, up-to-date procedural knowledge required for accurate simulation in dynamic digital environments. Recent work suggests that tutorials can function as high-level abstractions of environment dynamics (Xu et al., 2024; Zhang et al., 2025a; Su et al., 2025). However, standard retrieval pipelines often surface noisy or tangential information, which undermines the alignment between retrieved tutorials and the world-modeling process. For instance, a query about "fork chatgpt" might retrieve general Git forking tutorials rather than specific procedures for the current application context. To mitigate this, R-WoM incorporates a *reasoning-based* RAG pipeline that combines query rewriting with LLM-based reranking to improve the relevance of retrieved tutorials. In contrast to prior approaches that rely on computationally expensive iterative rollouts between policy and world models (Gu et al., 2024; Fang et al., 2025), R-WoM leverages the more lightweight yet effective chain-of-thought (CoT) reasoning mechanism for multi-step simulation. Moreover, we observe that the use of absolute reward estimation in existing works (Chae et al., 2024; Gu et al., 2024; Fang et al., 2025) could introduce biases and lead to unstable action scoring. To address this limitation, we employ a listwise reward estimation strategy that ranks simulation rollouts relative to each other rather than assigning absolute scores, leading to more robust and consistent action selection. Our key contributions are as follows:

- **Systematic probing of LLMs as world models.** We conduct comprehensive evaluation revealing that while LLMs excel at understanding immediate state changes and local transitions, they critically fail in producing procedures aligned to the environments over long horizons.
- **Retrieval-augmented world modeling framework.** We propose R-WoM, a retrieval-augmented framework that grounds LLM-based world models with external tutorials, enabling environment-specific adaptation through retrieval-augmented simulation and listwise reward estimation.
- **Empirical validation on realistic benchmarks.** We demonstrate R-WoM's effectiveness on two challenging computer-use benchmarks, WebArena (Zhou et al., 2023) and OSWorld (Xie et al., 2024), achieving consistent and substantial improvements (i.e., 7.2% to 25.3%) over competitive baselines, with particular advantages in longer-horizon scenarios.

2 BACKGROUND

2.1 PROBLEM FORMALIZATION

Given an initial task goal g, a computer-use agent interacts with the environment by iteratively receiving observations and executing actions to accomplish the task. Following the notation of prior work (Qin et al., 2025; Fang et al., 2025), we also introduce an intermediate reasoning component thought t, to capture thinking process. The resulting interaction trajectory can be expressed as

$$(g, (o_1, t_1, a_1), (o_2, t_2, a_2), \ldots, (o_n, t_n, a_n)), \tag{1}$$

where o_i is the observation at step i, t_i is the reasoning thought generated before action selection, and a_i is the executed action. At each step i, the LLM-based policy model produces a thought–action pair conditioned on the task goal, the current observation, and the prior interaction history:

$$(t_i, a_i) \sim \pi_p\big(\cdot \mid g, o_i, \{(o_j, t_j, a_j)\}_{j=v}^{i-1}\big), \quad v \in [1,\, i-1] \tag{2}$$

2.2 WORLD MODEL ROLLOUT

In realistic environments, many actions are irreversible or costly to undo, which makes naive trial-and-error exploration infeasible. To address this challenge, researchers explore using a world model (Hafner et al., 2019; 2020; 2023) that can simulate possible futures to be aware of the action outcomes before executing. Formally, at each decision step i, given the set of candidate actions along with their thoughts $\mathcal{A}_c = \{(t_i^{(1)}, a_i^{(1)}), (t_i^{(2)}, a_i^{(2)}), \ldots, (t_i^{(m)}, a_i^{(m)})\}$ proposed by policy model p in Equation 2, the world model performs k-step lookahead rollouts to estimate the potential outcomes of each action candidate $j \in \{1, 2, \ldots, m\}$:

$$\begin{aligned} o_{i+1}^{(j)} &\sim \pi_w(\cdot|g, o_i, t_i^{(j)}, a_i^{(j)}) \\ (t_{i+1}^{(j)}, a_{i+1}^{(j)}) &\sim \pi_w(\cdot|g, o_{i+1}, t_i^{(j)}, a_i^{(j)}) \\ &\vdots \\ o_{i+k}^{(j)} &\sim \pi_w(\cdot|g, o_{i+k-1}^{(j)}, t_{i+k-1}^{(j)}, a_{i+k-1}^{(j)}) \end{aligned} \tag{3}$$

For each k-step rollout trajectory $\hat{\tau}_i^{(j)} = (o_i^{(j)}, t_i^{(j)}, a_i^{(j)}, o_{i+1}^{(j)}, t_{i+1}^{(j)}, a_{i+1}^{(j)}, \ldots, o_{i+k}^{(j)})$, the corresponding rewards are estimated using a model-based/program-based reward function:

$$r(a^j) = R(\hat{\tau}_i^{(j)}, g) \tag{4}$$

The optimal action is then selected from $\mathcal{A}_c$ based on the highest estimated reward.

$$a_i^* = \arg\max_{a_i \in \mathcal{A}_c} r(a_i) \tag{5}$$

3 PRELIMINARY ANALYSIS

We focus on two fundamental capabilities of world models that are critical for computer-use tasks: **future state prediction**, which supports anticipating environment dynamics, and **reward estimation**, which underpins evaluating the outcomes of actions (Hafner et al., 2019; 2020; 2023). Recent work such as WMA (Chae et al., 2024) explores these aspects mainly through next-state identification and immediate reward estimation. However, such analyses do not fully account for the importance of reasoning across extended horizons. To address this, we design probing tasks tailored to these two capabilities by considering longer planning horizon. Specifically, for future state prediction, we design next-state identification and full-procedure planning alignment, which together capture both short and long horizon dynamics; For reward estimation, we design milestone transition recognition, which assesses models' ability to anticipate the outcomes of intermediate transitions. We apply these probes to three state-of-the-art LLMs, Qwen-2.5-VL-72B (Bai et al.,

2025), Claude-3.5-Sonnet[1], and Claude-3.7-Sonnet[2] by sampling trajectories on two challenging browser/computer-use benchmarks: WebArena (Zhou et al., 2023) and OSWorld (Xie et al., 2024). In the following, we introduce these tasks and present the probing analysis, while more details with illustrative examples are provided in Appendix A.1.

3.1 Next-State Identification

To assess the most basic requirement of future state prediction, we follow WMA (Chae et al., 2024) to design this task where models are asked to predict the correct subsequent observation given a current state and action. Given current observation o_i and action a_i, the model predicts the correct subsequent observation from two candidates:

$$\hat{o}_{i+1} = \arg \max_{o \in \{o_{i+1}^{\text{true}}, o_{i+1}^{\text{false}}\}} P(o | o_i, t_i, a_i) \tag{6}$$

Setup: Given the a n-step trajectory, we extract intermediate steps from successful and failed trajectories where $i \in [2, n-2]$ to avoid trivial predictions from initial or terminal states. For each (o_i, a_i, o_{i+1}) triplet, we create a negative sample by selecting the most lexically similar observation from the same trajectory. The lexical analysis is conducted using difflib[3], a Python's built-in library. This requires LLMs to distinguish the true next observation o_{i+1}^{true} from a distractor o_{i+1}^{false}.

Results: As shown in Table 1, models achieve relatively strong accuracy overall, i.e., exceeding 75%, indicating they can capture short-term state changes under various lexical similarity levels.

3.2 Full-Procedure Planning Alignment

While next-state identification evaluates whether an LLM can capture immediate state transitions, effective world models must also reason over longer horizons. To probe this ability, we design a plan alignment task, where models are asked to generate execution plans and these plans are evaluated for consistency with realistic environment dynamics. Formally, given a task goal g and an initial observation o_1, the model produces an execution plan $\hat{P} = (a_1, a_2, \ldots, a_T)$. Then a binary alignment score will be given by the LLM judge as below.

$$B = \Phi\Big(\langle g, o_1 \rangle, \hat{P}, P^{\star}\Big) \tag{7}$$

where P^* denotes the reference procedure derived from environment tutorials. The judgement is based on element attributes (e.g., location, text description, visibility) and operation logic (e.g., feasibility, ordering) with respect to P^*.

Setup: We sample tasks from WebArena and OSWorld benchmarks. For each task, we manually annotate a reference document chunk that is directly relevant to accomplishing the task under the corresponding environment (e.g., a website or software). More annotation details are in Appendix A.2. Models are then prompted to generate execution plans without access to tutorials, and the generated plans are evaluated by an LLM judge (Claude-3.7-Sonnet by default) for alignment against the reference procedures. Details of the evaluation prompt are provided in Appendix A.1.

Results: Table 1 shows that alignment remains moderate across all models, rarely exceeding 65%. This reveals a clear limitation: while LLMs can list plausible actions, they often fail to maintain procedural coherence or respect environment-specific constraints.

3.3 Milestone Transition Recognition

Aside from probing LLM's capability of capturing future states, we also probe whether models can recognize task-relevant progress, an essential skill for reward estimation in world models. The task evaluates whether models can distinguish promising transition sequences from unproductive ones:

$$\hat{S} = \arg \max_{S \in \{S^{\text{true}}, S^{\text{false}}\}} P(\text{success} \mid S, g) \tag{8}$$

[1] https://www.anthropic.com/news/claude-3-5-sonnet

[2] https://www.anthropic.com/news/claude-3-7-sonnet

[3] https://docs.python.org/3/library/difflib.html

Table 1: Probing results across three tasks: next-state identification, full-procedure planning alignment, and milestone transition recognition. All values are percentages.

Model	Next-state identification (by lexical similarity)				Full-procedure planning alignment	Milestone transition recognition
	[0, 0.8]	[0.8, 0.9]	[0.9, 1)	Overall	Accuracy	Accuracy
Qwen-2.5-VL-72B	61.1	84.8	77.6	77.0	50.0	83.7
Claude-3.5-Sonnet	72.2	84.8	81.6	81.0	55.0	85.7
Claude-3.7-Sonnet	88.9	87.9	83.7	86.0	65.0	86.7

where $S = \{o_i, o_{i+h}, o_{i+2h}, \ldots, o_{i+(l-1)h}\}$ denotes a subsequence of length l sampled at interval h from the full trajectory.

Setup: We sample sequences of $l = 3$ consecutive transitions with interval $h = 2$ from both successful and failed trajectories, where the intervals are used to avoid repeated states. Same as next state identification, we also sample steps from steps within $[2, n-2]$ to avoid trivial predictions. For each objective g, we annotate pairs where S^{true} represents a more promising subsequence drawn from a successful trajectory, and S^{false} represents a less effective subsequence from a failed trajectory. More task details can be found in Appendix A.1.

Results: Table 1 shows that all models perform strongly. Claude-3.7-Sonnet achieves the highest accuracy (86.7%), followed by Claude-3.5-Sonnet (85.7%) and Qwen-2.5-VL-72B (83.7%). The consistently high performance across models suggests that LLMs possess reasonable ability to evaluate which transitions are conducive to task progress.

3.4 Discussion

Overall, our probing analysis reveals that modern LLMs demonstrate relatively good short-term predictive and local evaluative capabilities: they can reliably identify next states and recognize task-relevant transitions. However, these strengths do not extend to long-horizon planning, where performance deteriorates sharply in aligning its knowledge to specific environments. This suggests that LLMs might inherently lack robust generalization for world modeling across dynamic environments, thus may require external guidance to sustain accurate simulations over extended horizons.

4 R-WoM Framework

From the probing analysis in Section 3, we identify grounding as a key mechanism for improving the alignment of LLMs to specific environments, which motivates the design of our R-WoM framework.

4.1 Overview

As illustrated in Figure 2, the R-WoM framework employs the retrieval-augmented way to ground world modeling during simulation. Given the task objective and current observation, relevant documentation and tutorials are retrieved and reranked to form the grounding evidence set. This evidence is used to condition the world model during both state transition prediction and reward estimation. Algorithm 1 summarizes the complete R-WoM pipeline, which iteratively applies this process until task completion or termination.

4.2 Design Details

RAG design. We adopt a reasoning-based retrieval design to enhance relevance of retrieved document chunks to the given query. Given the task goal g, we construct a query $q = f_{\text{enc}}(g)$ and retrieve top-k tutorial chunks $\mathcal{C}_k$ based on cosine similarity. An LLM-based reranker (i.e., policy model p here) then conducts a list-wise reranking of candidates based on contextual relevance:

$$\mathcal{E} = f_p^{\text{rank}}(\mathcal{C}, q) \tag{9}$$

yielding the final evidence set $\mathcal{E}$. The world model conditions on $\mathcal{E}$ for grounded future state prediction and reward estimation.

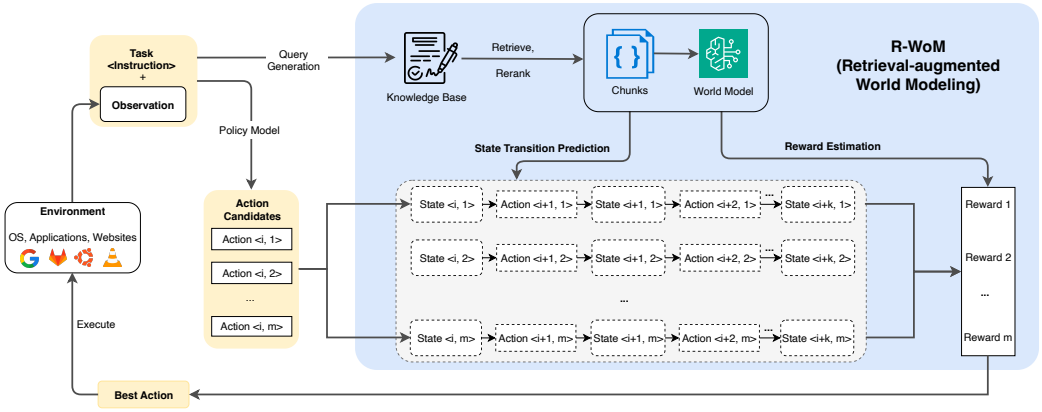


Figure 2: Overview of the R-WoM pipeline. At each time step i, the policy model generates m candidate actions. For each candidate, the world model grounded by retrieved tutorials performs k-step rollouts to simulate a possible future trajectory. The rewards of rollout trajectories are finally estimated by world models to select the best action.

Algorithm 1 The Pipeline of R-WoM

Require: Task objective g, initial observation o_1

$\mathcal{E} \leftarrow$ Retrieve and rerank tutorials relevant to the objective g
$i \leftarrow 1$
while task not completed **do**
  $\mathcal{A}_c \leftarrow \{(t_i^{(1)}, a_i^{(1)}), (t_i^{(2)}, a_i^{(2)}), \ldots, (t_i^{(m)}, a_i^{(m)})\} \sim \pi_p(\cdot|g, o_i)$
  for each $(t_i^{(j)}, a_i^{(j)}) \in \mathcal{A}_c$ **do**
    Generate rollout trajectory $\hat{\tau}_i^{(j)} = \pi_w^{\mathrm{CoT}}(o_i, t_i^{(j)}, a_i^{(j)}; \mathcal{E})$
  end for
  $(t_i^\star, a_i^\star) = \arg\max_{(t_i^{(j)}, a_i^{(j)}) \in \mathcal{A}_c} \left[f_w\big(\{R(\hat{\tau}_i^{(j)}, g, \mathcal{E})\}\big) \right]$
  Execute $a_i^\star$, observe o_{i+1}
  $i \leftarrow i + 1$
end while

R-WoM design. At step i, with tutorial evidence $\mathcal{E}$, for each candidate action $a_i^{(j)} \in \mathcal{A}_i$, the world model performs a chain-of-thought (CoT) (Wei et al., 2022) reasoning process that unfolds a multi-step imagination trajectory in a single forward reasoning sequence, rather than iterative policy–then-world model interaction. The rollout produces a predicted trajectory of k steps:

$$\hat{\tau}_i^{(j)} = \pi_w^{\mathrm{CoT}}(o_i, t_i^{(j)}, a_i^{(j)}; \mathcal{E}) \tag{10}$$

We observe that absolute sparse reward used in previous works (Chae et al., 2024; Gu et al., 2024; Fang et al., 2025) might not effectively distinguish more meaningful rollouts. Therefore, inspired by recent advances in relative reward design (Liu et al., 2024; Choi et al., 2024; Guo et al., 2025), we employ a list-wise ranking mechanism to evaluate simulated trajectories in a relative way.

$$(t_i^\star, a_i^\star) = \arg \max_{(t_i^{(j)}, a_i^{(j)}) \in \mathcal{A}_c} \left[f_w\big(\{R(\hat{\tau}_i^{(j)}, g, \mathcal{E})\}\big) \right] \tag{11}$$

As is shown in Equation 11, each rollout trajectory is scored relatively in the comparative context of all candidates. In this way, we aim to reduce potential bias from absolute reward signals and stablize the selection of most promising action candidate.

5 Experiment

To rigorously evaluate the effectiveness of R-WoM, we formulate the following research questions:

Table 2: End-to-end performance on OSWorld and WebArena across three runs. Best in **bold**; second-best underlined. R-WoM cells include relative improvement over the second-best.

Model	Method	OSWorld (Xie et al., 2024)	WebArena (Zhou et al., 2023)
Qwen-2.5-VL-72B	Vanilla	26.36 ± 2.32	21.84 ± 0.42
	RAG	30.84 ± 1.07	22.42 ± 0.42
	WebDreamer	28.37 ± 2.01	24.50 ± 0.84
	R-WoM	**38.05** $\pm$ **2.29** (+23.4%)	**28.92** $\pm$ **0.43** (+18.1%)
Claude-3.5-Sonnet	Vanilla	22.43 ± 2.25	27.74 ± 0.43
	RAG	22.19 ± 0.92	30.70 ± 0.41
	WebDreamer	23.48 ± 2.14	29.82 ± 0.41
	R-WoM	**26.41** $\pm$ **0.44** (+12.5%)	**33.65** $\pm$ **0.01** (+9.6%)
Claude-3.7-Sonnet	Vanilla	28.47 ± 2.27	28.92 ± 0.41
	RAG	27.76 ± 0.75	32.75 ± 0.72
	WebDreamer	31.24 ± 2.88	31.86 ± 0.01
	R-WoM	**39.13** $\pm$ **1.92** (+25.3%)	**35.11** $\pm$ **1.10** (+7.2%)

- **RQ1**: Does R-WoM improve the performance of computer-use agents compared to established baselines in realistic environments such as browsers and operating systems?
- **RQ2**: How do external tutorials contribute to grounding world models, and to what extent do agents benefit from incorporating this information from tutorials?
- **RQ3**: Can tutorial-grounded world models support longer imagination horizons more effectively than ungrounded counterparts over multi-step rollouts?

5.1 Setup

We evaluate R-WoM against three baselines:

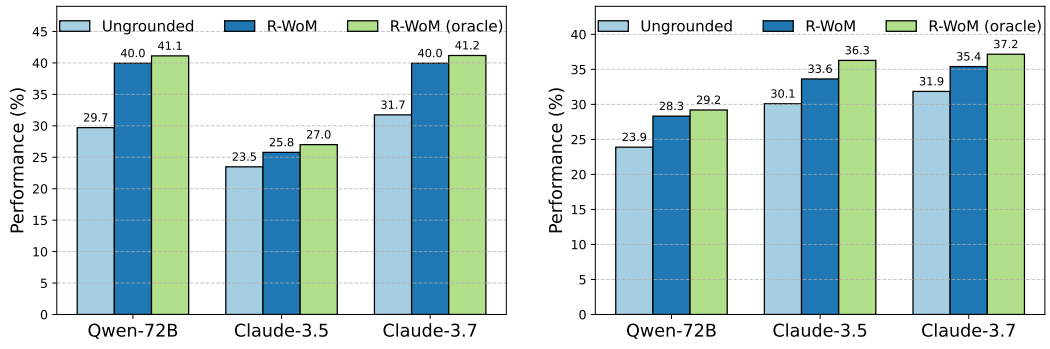
- **Vanilla**: The vanilla approach is adapted from the official implementations: the screenshot-based version for OSWorld provided by GTA-1 (Yang et al., 2025), and the screenshot+axtree version for WebArena provided by WMA (Chae et al., 2024). This approach relies solely on the task objective, current observation (represented as screenshots and axtrees) and prior interaction history.
- **RAG**: A retrieval-augmented generation pipeline that retrieves relevant documentation and augments the LLM before action prediction, which is built upon the vanilla approach.
- **WebDreamer** (Gu et al., 2024): An iterative world-model-based approach that imagines future states assign sparse rewards for imagined trajectories (also built upon the vanilla approach).

We conduct experiments on two comprehensive benchmarks designed for multi-round interactions in realistic computer-use environments: **WebArena** (Zhou et al., 2023), which spans web-based tasks across domains such as e-commerce, social forums, and collaborative platforms; and **OSWorld** (Xie et al., 2024), which covers diverse desktop tasks including file management, terminal commands, and productivity applications. Specifically, we sample a subset from these two benchmarks for our experiments where tutorials available and for retrieval purpose and we collect tutorials from both online websites. The details of the subsets and tutorial collection can be found in Appendix A.2. We test three popular LLM backbones: **Qwen-2.5-VL-72B (Instruct version)** (Bai et al., 2025), **Claude-3.5-Sonnet**, and **Claude-3.7-Sonnet**, serving as both the policy and world model. For methods requiring retrieval, we build the RAG pipeline with Langchain[4], FAISS (Douze et al., 2024) as the vector store, and Qwen-3-Embedding-8B (Zhang et al., 2025b) as the embedding model. More implementation details can be found in Appendix A.3.

5.2 RQ1: End-to-End Performance

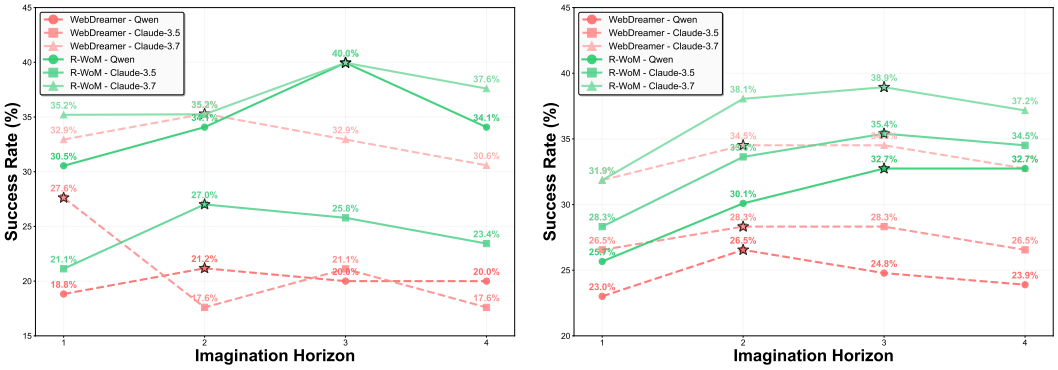
Table 2 reports the overall end-to-end performance. It shows that R-WoM consistently outperforms all alternatives, with improvements of +23.4% on OSWorld and +18.1% on WebArena for Qwen-2.5, +12.5% and +9.6% for Claude-3.5, and +25.3% and +7.2% for Claude-3.7 over the strongest

[4] https://github.com/langchain-ai/langchain



(a) OSWorld (b) WebArena

Figure 3: Performance under different grounding settings, where we compare ungrounded world model: WebDreamer, world model grounded with retrieved tutorials: R-WoM, and world model grounded with oracle tutorials: R-WoM (oracle).



(a) OSWorld (b) WebArena

Figure 4: Success rates (%) across imagination horizons on OSWorld (a) and WebArena (b). R-WoM (green, solid) consistently outperforms WebDreamer (red, dashed) and reaches its peak at larger imagination horizon (at horizon around 3), indicating that grounding benefits world models in simulations over longer horizons.

non-R-WoM baselines. These results reveal that the improvements remain stable across different backbones, highlighting that grounding mitigates error accumulation and continues to enhance both capability-limited and already capable models. Overall, R-WoM provides more consistent, scalable benefits compared with retrieval alone or ungrounded world modeling.

5.3 RQ2: The Role of Tutorials in Grounding World Models

To assess the role of tutorials, we compare three settings: no grounding (WebDreamer), grounding with R-WoM using retrieved tutorials, and grounding with R-WoM using oracle tutorials. Similarly as the full-procedure alignment task in Section 3.2, we also manually annotate document chunks that are relevant to the task from human's perspective. More annotation details and the performance of retrieval can be found in Appendix A.2 and A.4, respectively. As shown in Figure 3, performance consistently improves with the grounding levels, from no grounding to grounding with retrieved tutorials, then to grounding with oracle tutorials. It indicates that access to external procedural knowledge helps models in world modeling. These findings underscore that R-WoM's effectiveness is tightly coupled with tutorial fidelity, and the advances in retrieval and resource curation represent critical levers for future progress.

5.4 RQ3: Ablation studies of imagination horizon

To examine the effect of imagination horizon on end-to-end performance, we vary the horizon from 1 to 4 for both ungrounded (WebDreamer) and grounded (R-WoM) world models, as shown in Figure 4. WebDreamer, the world model without grounding during rollouts, shows modest initial gains but quickly plateaus and even declines beyond 2 steps, reflecting its susceptibility to compounding prediction errors. In contrast, R-WoM maintains consistently higher success across horizons on both OSWorld and WebArena, with improvements lasting up to horizon three before slightly tapering off. These results suggest that tutorial grounding not only mitigates error accumulation but also stabilizes rollouts over longer horizon simulations.

6 Related Works

6.1 Computer-use Agent

One line of works focuses on exploring how to improve agent's understanding of computer-use actions, such as building end-to-end agent frameworks (Agashe et al., 2024; 2025; Song et al., 2025), and training native agent models (Qin et al., 2025; Wang et al., 2025; Lai et al., 2025) or specific action grounding models (Wu et al., 2024; Xie et al., 2025; Yang et al., 2025). Another line of works explores treating LLMs as world models to simulate the computer-use environments. WebDreamer (Gu et al., 2024) pioneers this direction by using LLMs to simulate the outcome of candidate actions, and evaluate these imagined states with discrete reward given by LLM judge (Gu et al., 2024). Subsequent works such as WMA (Chae et al., 2024) adapt this idea to improve planning by abstracting state transitions into natural language summaries. WKM (Qiao et al., 2024) and WebEvolver (Fang et al., 2025) develop co-evolving world models and policies to progressively refine both simulation and planning, moving beyond one-horizon imagination.

6.2 Tutorial-use

Parallel developments leverage tutorials or indirect knowledge to train digital agents. Synatra (Ou et al., 2024) converts human-oriented tutorials into 100k synthetic demonstrations to fine-tune a 7B CodeLLaMA model. Other frameworks generate trajectories guided by tutorial completion or replay (e.g., AgentTrek (Xu et al., 2024), TongUI (Zhang et al., 2025a)) to teach GUI navigation and tool use from multimodal resources. Learn-by-interact (Su et al., 2025) synthesizes trajectories by leveraging tutorials and interaction with the environments. These approaches focus on offline trajectory generation by referring to tutorials while our approach focuses on tutorial-guided grounding of LLMs as world models at inference time.

7 Conclusion and Future Work

We presented a systematic study of LLM-based world models for computer-use tasks, revealing that while they can model state transitions and recognize task-relevant progress, they fail to reliably adapt to unfamiliar environments without grounding. To address this, we proposed the Retrieval-augmented World Model (R-WoM), which incorporates environment-specific tutorial knowledge during the imagination rollouts and reward prediction procedures to reduce hallucinations and stale knowledge. Evaluations on WebArena and OSWorld show that R-WoM consistently outperforms competitive baselines, demonstrating the efficacy of retrieval-augmented grounding for LLM agents in dynamic browser-use and computer-use scenarios. While R-WoM shows promises in improving LLM as world models, some bottlenecks still remain. First, the grounding stage requires availability of online tutorials for the target environment, which has limits in tutorial-scarce domains, or when documentation is outdated or access-restricted. Synthesizing tutorials from tutorial-scarce environments is one of the future directions we aim to explore. Second, despite the efficiency optimizations in R-WoM's rollout simulation and reward estimation, the computational cost is still non-trivial. Conducting world modeling in an agentic way to further reduce costs can be our future work.

8 Ethics and Reproducibility Statement.

Ethics. Our work uses only publicly available benchmarks (OSWorld and WebArena). While retrieval-augmented methods may inherit biases from external sources, our study remains confined to controlled environments.

Reproducibility. Our work is reproducible. We provide the algorithm process of our method, Retrieval-augmented World Model (R-WoM), in Algorithm 1. The experimental setup, are described in Section 5.1 and the implementation details are provided in Appendix A.3.

References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *arXiv preprint arXiv:2410.13232*, 2024.

Heewoong Choi, Sangwon Jung, Hongjoon Ahn, and Taesup Moon. Listwise reward estimation for offline preference-based reinforcement learning. *arXiv preprint arXiv:2408.04190*, 2024.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. Webevolver: Enhancing web agent self-improvement with coevolving world model. *arXiv preprint arXiv:2504.21024*, 2025.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025.

Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, et al. Lipo: Listwise preference optimization through learning-to-rank. *arXiv preprint arXiv:2402.01878*, 2024.

Tianyue Ou, Frank F Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. *Advances in Neural Information Processing Systems*, 37:91618–91652, 2024.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024.

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Silvio Savarese, Zeyuan Chen, Jieyu Zhao, et al. Coact-1: Computer-using agents with coding as actions. *arXiv preprint arXiv:2508.03923*, 2025.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö Arık. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *arXiv preprint arXiv:2501.10893*, 2025.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

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*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *arXiv preprint arXiv:2412.09605*, 2024.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, et al. Gta1: Gui test-time scaling agent. *arXiv preprint arXiv:2507.05791*, 2025.

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.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. Tongui: Building generalized gui agents by learning from multimodal web tutorials. *arXiv preprint arXiv:2504.12679*, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

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

A APPENDIX

Roadmap: Section A.1 introduces the design details of our probing task. Section A.2 introduces the tutorial collection, annotation and retrieval approach for our experiments. Section A.3 presents the implementation details of R-WoM, including action space definition and prompt design. Section A.4 presents additional experimental results.

A.1 DETAILS OF PROBING TASK

PROMPT FOR NEXT STATE IDENTIFICATION

Given the previous state of the web page: {previous_state} and the current action: {current_action}, please reason about the next state. The next state can be one of the following: {state_a}, {state_b}. Please reason about the next state and return the rationale and the choice. The choice should be one of the following: A, B. Output the choice in the following JSON format:

```
{
    "rationale": "...",
    "choice": "..."
}
```

Task 1: Next-state identification. To assess whether the world model can predict the immediate outcome of an action given the current state, the model is asked to discriminate between the true next observation and a lexically similar distractor, as illustrated in Figure 5. In this way, we aim to probe LLM's sensitivity to environment changes. We construct 100 samples drawn from trajectories in WebArena for this task.

Task 2: Full-procedure planning alignment. Moving beyond identifying next state, we would like to probe whether LLM can reason about longer steps of future states. As shown in A.1, given a task objective, the model is asked to generate a multi-step plan, which is then validated against tutorials describing environment dynamics. The evaluation measures whether the model's procedure aligns with realistic element locations, operation sequences, and interaction methods. To assess this capability, we construct 40 samples from trajectories in both OSWorld and WebArena.

PROMPT FOR FULL-PROCEDURE PLANNING ALIGNMENT

You are a grounding validation assistant that verifies whether tutorial-referenced operations in a plan are accurately grounded in the provided documentation.

Evaluation criteria

1. Element Text Accuracy: Exact text matches between plan and tutorial for referenced elements.
2. Location Consistency: Location indicators (position, context) align with tutorial descriptions.
3. Operation Sequence: Prerequisites and dependencies match tutorial methodology.
4. Interaction Method: Specified actions (click, input, select) align with tutorial instructions.
5. Attribute Precision: Element types, properties, and characteristics match tutorial specifications.

Evaluation principle

1. Accept: Plan steps that extend beyond tutorial scope (additional operations are allowed).
2. Reject: Any tutorial-referenced operation with misaligned text, location, or method.

Output Format
Output your response in the following JSON format:

```
{
   "rationale": "Your rationale of your evaluation",
   "answer": "yes/no"
}
```

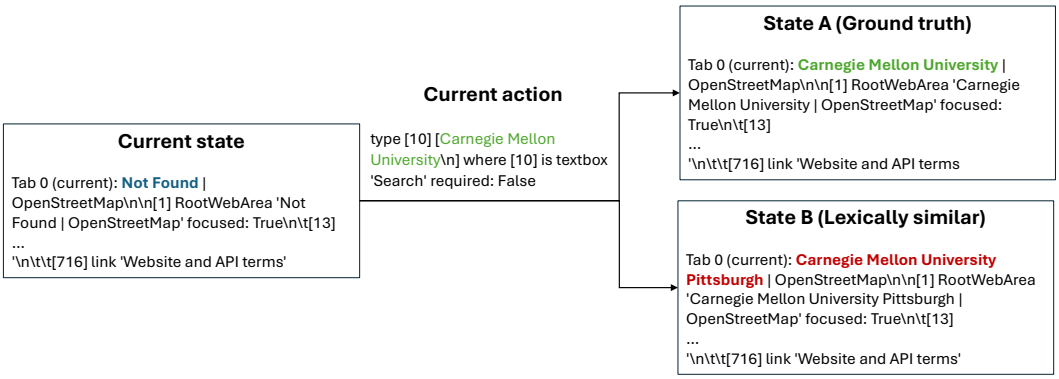


Figure 5: Illustration of the **next-state identification** probing task. Given a current state and an action, the model must choose between two candidate next states: (A) the ground-truth state, and (B) a lexically similar distractor. This task evaluates whether the world model can correctly predict the true next observation rather than being misled by textual similarity.

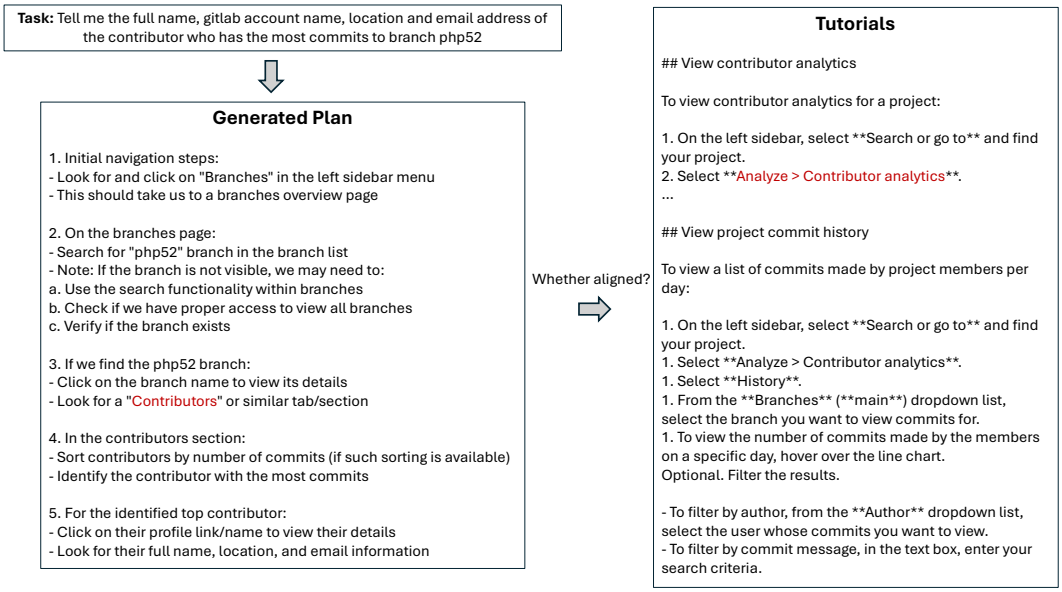


Figure 6: Illustration of the **full-procedure planning alignment** probing task. Given a task objective (top), the model generates a multi-step plan (left), which is then compared against environment-specific tutorials (right). The evaluation checks whether the generated procedure aligns with the tutorials in terms of navigation logic, element selection, and operation feasibility. This task assesses the world model's ability to sustain long-horizon procedural reasoning in realistic environments.

PROMPT FOR MILESTONE TRANSITION RECOGNITION

You are evaluating web automation trajectories to identify which one is more likely to succeed in completing the given task.
The following two trajectories show segments from different agent attempts at the same task. Both agents were following the same initial steps, but diverged when they chose different actions at a critical decision point. Your task is to determine which trajectory segment demonstrates better progress toward completing the task objective. You need to output in the following JSON format as:

```
{
    "answer": "A/B",
    "rationale": "xxx"
}
```

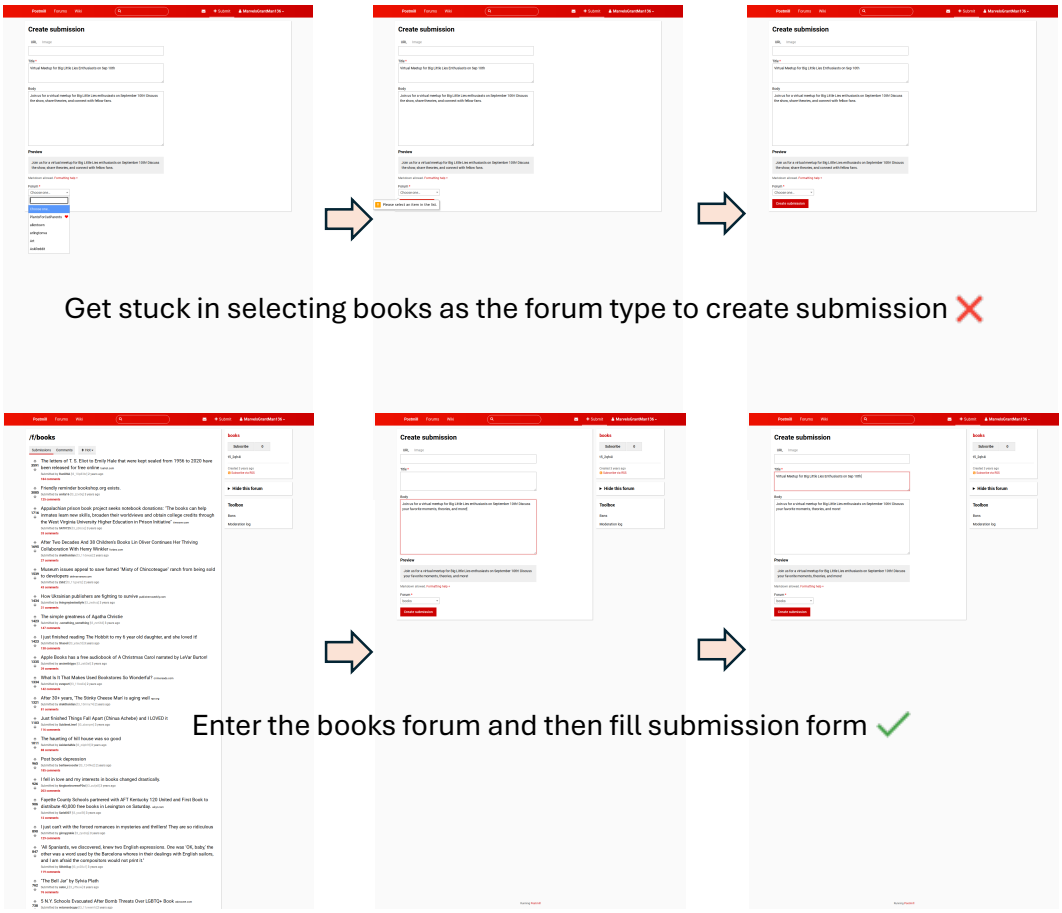


Figure 7: Illustration of the **milestone transition recognition** probing task. Given a sequence of transitions, the model must identify whether they reflect meaningful progress toward the goal. In this example, the top path shows an unproductive transition where the agent gets stuck trying to directly select "books" as a forum type, failing to proceed. The bottom path shows a more promising milestone transition: the agent first enters the books forum and then successfully fills out the submission form. The task evaluates whether the world model can distinguish between effective and ineffective procedural progress.

Task 3: Milestone transition recognition. To probe reward estimation capability of LLMs, we design this task to assess whether LLMs have the capability to capture meaning state transitions. As shown in Figure 7, the LLM is presented with pairs of trajectory segments that diverge at a decision point, one representing a promising milestone transition and the other an unproductive path. The LLM needs to identify which trajectory is more conducive to task success. This setting is evaluated on 98 samples drawn from both successful and failed trajectories in WebArena.

A.2 Tutorial Processing

Our framework relies on tutorials as external grounding for browser- and computer-use tasks. To construct a comprehensive knowledge base, we gather tutorials from both general-purpose and environment-specific resources. For cross-domain instructional guidance, we include WikiHow, which provides structured, step-by-step content spanning a broad range of tasks. For environment-specific domains, we incorporate official documentation from the corresponding software or websites. The complete list of tutorial sources is as follows:

- WikiHow: `https://www.wikihow.com/Main-Page`
- Google Chrome Help: `https://support.google.com/chrome`

- GIMP 3.0 User Manual: https://docs.gimp.org/3.0/en/
- Visual Studio Code Documentation: https://code.visualstudio.com/docs
- Ubuntu Help: https://help.ubuntu.com/22.04/ubuntu-help/
- Mozilla Thunderbird Support: https://support.mozilla.org/en-US/products/thunderbird/learn-basics-get-started
- VLC Media Player User Guide: https://docs.videolan.me/vlc-user/desktop/3.0/en/
- LibreOffice Help: https://help.libreoffice.org/latest/en-US/
- GitLab Documentation: https://docs.gitlab.com/
- Adobe Commerce Admin User Guides: https://experienceleague.adobe.com/en/docs/commerce-admin/user-guides/home

From these sources, we construct a knowledge base of over 30k chunked tutorial documents that collectively support tasks across diverse software and website environments. Since our framework requires tutorial availability to provide concrete grounding, we sample task subsets from OSWorld and WebArena that can be partially mapped to tutorial examples. Specifically, we select 85 tasks from OSWorld, covering domains such as Chrome, GIMP, VSCode, VLC, Thunderbird, and Ubuntu OS, and 113 tasks from WebArena, covering CMS and GitLab domains and we annotate one or two document chunks that are most relevant to each task from human's perspective.

To retrieve useful tutorials at inference time, we adopt a reasoning-based retrieval strategy. This involves query rewriting to anonymize and generalize task queries, followed by LLM-based reranking to reduce false negatives that may arise when relying solely on cosine similarity. The detailed prompts used for query rewriting and reranking are provided below, and the results comparing retrieval strategies are reported in Appendix A.4.

PROMPT FOR QUERY REWRITING

You are an AI assistant that rewrite original query into comprehensive, searchable queries that are easier to retrieve answers from documents. You must follow these rules:

1. Organize the original query to be well-structured and clear with details: Try to make the query detailed and clear. For example, instead of a title like "Fork ChatGPT", a good rewritten query would be, "How could I fork the ChatGPT repository in the gitlab?"
2. Generalize Personal Details: Replace all specific, personal information (like user names, file names, file location) with general descriptions (like "a user", "a xxx format file", "at desktop").

PROMPT FOR RERANKING

Your task is to re-rank a list of documents based on their relevance to a given task. Carefully analyze the task and each numbered document. Your goal is to identify which documents are helpful for completing the task and order them accordingly.
Your output must be a single JSON object with one key: "reranked_indexes". The value for this key must be a list of the original document indexes, sorted from most relevant to least relevant.
Example format:

```
{
    "reranked_indexes": [0, 2, 1]
}
```

A.3 IMPLEMENTATION DETAILS OF R-WOM

To enable automation in browser and computer-use environments, we adopt the official action space definitions provided by WebArena[5] and OSWorld[6], as summarized in Table 3. In practice, we find that direct action coordinate mapping in OSWorld poses challenges for models such as the Qwen

[5] https://github.com/web-arena-x/webarena
[6] https://github.com/xlang-ai/OSWorld

Table 3: Action space for WebArena and OSWorld.

Environment	Action	Definition
WebArena	`click`	Clicks a webpage element identified by its id.
	`type`	Types text into a webpage element; may submit if appropriate.
	`hover`	Moves the cursor over a webpage element.
	`press`	Presses a key or key combination.
	`scroll`	Scrolls the page up or down.
	`new_tab`	Opens a new browser tab.
	`tab_focus`	Focuses a specific browser tab.
	`close_tab`	Closes the active browser tab.
	`goto`	Navigates the current tab to a URL.
	`go_back`	Navigates to the previous page.
	`go_forward`	Navigates to the next page.
	`stop`	Terminates the task and returns an answer (use `N/A` if unknown).
OSWorld	`click`	Clicks a described UI element in the desktop environment.
	`drag_and_drop`	Drags from one described UI location to another.
	`highlight_text_span`	Highlights text between two provided phrases.
	`hold_and_press`	Holds keys and presses a sequence of keys.
	`hotkey`	Presses a hotkey combination.
	`open`	Opens an application or file by name.
	`scroll`	Scrolls within a described element.
	`set_cell_values`	Sets specified cells in a spreadsheet.
	`switch_applications`	Switches focus to another open application.
	`type`	Types text into a described element.
	`wait`	Pauses execution for a short duration.
	`done`	Ends the task successfully and returns the final answer if any.
	`fail`	Ends the task with failure and stop.

series and Claude-3.5-Sonnet. To address this and enable the policy model to generate more effective actions during world model rollouts, we employ GTA-1-7B (Yang et al., 2025) as an auxiliary action grounding model to assist in action generation when evaluating on OSWorld. For retrieval-related approach (i.e., RAG and R-WoM), we use top-5 retrieved document chunks by default to put them into the LLM's context.

PROMPT FOR GENERATING ACTION CANDIDATES

You are a reasoner that analyzes the current state, previous actions, and task progress to determine the next required action.

Available actions
Action space definition
Rules for success

1. When pressing keys, ensure held/pressed keys are within {`KEYBOARD_KEYS`}.
2. Output a single action at each step; do not bundle multiple intents into one step.
3. Only issue actions that are valid for the current observation (e.g., do not type into buttons or click static text).
4. Strictly avoid repeating the same action if the interface state is unchanged.

Response JSON schema

```
{
  "observation": "Description of current state and any changes
     observed",
  "action_candidates": [
    {
      "thought_and_action": "Why this action is appropriate given
         the observation",
      "action_code": {
        "action_type": "action_type",
```

```
        "parameters": {
          "param1": "value1",
          "param2": "value2"
        }
      }
    }
  ]
}
```

Output requirements

- *observation*: provide a detailed description of the current computer state based on the full screenshot, noting any state changes.
- *action_candidates*: include {`branching_factor`} candidates, ordered by confidence (most confident first). For each candidate, include:
  - *thought_and_action*: rationale for the proposed action.
  - *action_code*: the concrete action with its required parameters.

PROMPT FOR RETRIEVAL-AUGMENTED FUTURE STATE ROLLOUTS

You are a world-model assistant with extensive knowledge of desktop and web UIs. Given the previous observations, the task objective, and a candidate action, you must "simulate the future" and describe the plausible future states.

Available actions
Action space definition
Tutorial usage guideline

1. Use tutorials to identify efficient workflow patterns that should be predicted as likely outcomes.
2. Provide a reference to the tutorial if the current situation matches the standard operations in the tutorials. If the current situation does not align with tutorials, rely on internal world knowledge instead.

Environment awareness checklist

- Visible UI elements: text, icons, menus, modals, tooltips
- Element states: enabled/disabled, focused/hovered, loading progress
- Hidden or off-screen affordances revealed by scrolling or clicking
- Cursor position, caret position, selection highlights
- Global context: file system changes, network requests, OS dialogs

Output Format
Produce an ordered chain from **STATE 0** (current) up to **STATE n** ($1 \leq n \leq \{k\}$); you may stop early if no further prediction is useful.

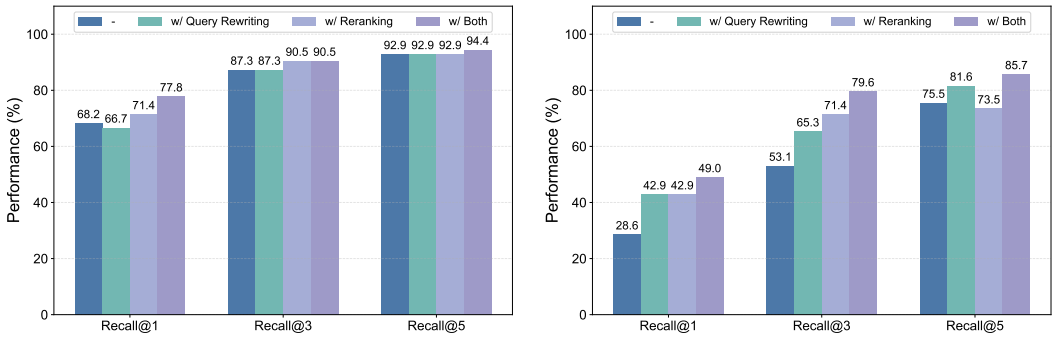
PROMPT FOR RETRIEVAL-AUGMENTED REWARD ESTIMATION

You are an agent that evaluates actions by considering previous observations and the potential outcomes of these actions.
Tutorial Grounding Guidance
Priorize action sequences that follow the standard operations in the tutorials and have captured the milestones and conditions to make more meaningful progress to achieve the task objective.
Output Format
Output your response in the following JSON format:

```
{
  "ranking": [x, x, x] # "indexes of the action candidates, most
      promising first",
  "thought": "your rationale for the ranking result"
}
```

Table 4: Domain-level performance. Best in **bold**; second-best underlined.

Benchmark	Domain	Model	Vanilla	RAG	WebDreamer	R-WoM
OSWorld	chrome (17)	Qwen-2.5-VL-72B	9.95 ± 0.02	8.93 ± 0.02	6.29 ± 0.47	**9.95 ± 0.02**
		Claude-3.5-Sonnet	5.28 ± 0.45	5.27 ± 0.46	4.95 ± 0.02	**5.92 ± 0.00**
		Claude-3.7-Sonnet	5.00 ± 0.00	6.97 ± 0.04	7.31 ± 0.49	**8.95 ± 0.02**
	gimp (22)	Qwen-2.5-VL-72B	7.33 ± 0.47	7.33 ± 0.47	9.33 ± 0.47	**11.33 ± 0.47**
		Claude-3.5-Sonnet	5.33 ± 0.47	5.33 ± 0.47	3.33 ± 0.47	**6.00 ± 0.00**
		Claude-3.7-Sonnet	5.00 ± 0.82	5.33 ± 0.47	9.67 ± 0.47	**10.67 ± 0.47**
	thunderbird (11)	Qwen-2.5-VL-72B	1.33 ± 0.47	2.67 ± 0.47	2.33 ± 0.47	**3.67 ± 0.47**
		Claude-3.5-Sonnet	2.00 ± 0.00	**2.33 ± 0.47**	**2.33 ± 0.47**	2.00 ± 0.00
		Claude-3.7-Sonnet	2.67 ± 0.47	2.33 ± 0.47	2.00 ± 0.00	**4.00 ± 0.00**
	vlc (5)	Qwen-2.5-VL-72B	0.33 ± 0.47	**1.33 ± 0.47**	0.33 ± 0.47	1.00 ± 0.00
		Claude-3.5-Sonnet	1.33 ± 0.47	1.33 ± 0.47	1.67 ± 0.47	**2.00 ± 0.00**
		Claude-3.7-Sonnet	**1.67 ± 0.47**	1.00 ± 0.00	0.33 ± 0.47	0.33 ± 0.47
	os (15)	Qwen-2.5-VL-72B	3.33 ± 0.47	1.33 ± 0.47	**4.33 ± 0.47**	**4.33 ± 0.47**
		Claude-3.5-Sonnet	2.33 ± 0.47	2.00 ± 0.00	**4.67 ± 0.47**	3.00 ± 0.00
		Claude-3.7-Sonnet	5.33 ± 0.47	3.33 ± 0.47	6.33 ± 0.47	**6.67 ± 0.47**
	vs_code (15)	Qwen-2.5-VL-72B	2.33 ± 0.47	**5.33 ± 0.47**	3.67 ± 0.47	4.33 ± 0.47
		Claude-3.5-Sonnet	2.67 ± 0.47	**3.33 ± 0.47**	2.33 ± 0.47	3.00 ± 0.00
		Claude-3.7-Sonnet	4.67 ± 0.47	**5.33 ± 0.47**	4.33 ± 0.47	**5.33 ± 0.47**
WebArena	shopping_admin (57)	Qwen-2.5-VL-72B	11.33 ± 0.47	12.33 ± 0.47	12.33 ± 0.47	**15.33 ± 0.47**
		Claude-3.5-Sonnet	14.33 ± 0.47	15.00 ± 0.00	14.67 ± 0.47	**17.67 ± 0.47**
		Claude-3.7-Sonnet	15.33 ± 0.47	17.33 ± 0.47	18.33 ± 0.47	**19.00 ± 0.82**
	gitlab (56)	Qwen-2.5-VL-72B	13.33 ± 0.47	13.00 ± 0.00	15.33 ± 0.47	**17.33 ± 0.47**
		Claude-3.5-Sonnet	17.00 ± 0.82	19.67 ± 0.47	19.00 ± 0.00	**20.33 ± 0.47**
		Claude-3.7-Sonnet	17.67 ± 0.47	19.67 ± 0.47	17.67 ± 0.47	**20.67 ± 0.47**



(a) OSWorld (b) WebArena

Figure 8: Retrieval performance under different retrieving strategies.

A.4 Additional Experimental Results

Breakdown of end-to-end performance. Table 4 provides a domain-level view of performance. R-WoM consistently achieves the best results across most of the domains, but the relative magnitude of improvement varies. In domains such as `chrome` and `gimp`, where tasks involve longer dependencies and compounding errors, R-WoM exhibits the largest margins over WebDreamer. By contrast, in lighter workloads such as `vlc` or `thunderbird`, the absolute gains are smaller and sometimes comparable to RAG, suggesting that grounding might bring limited additional benefit when task horizons are short. These results imply that grounding is most critical in environments requiring extended planning.

Ablation studies of retrieval performance. Figure 8 shows that retrieval performance improves most when query rewriting and reranking are combined, indicating their complementary effects. Query rewriting is more beneficial in diverse environments like WebArena, while reranking offers steadier gains across settings by filtering irrelevant matches. The overall trend suggests that single strategies yield uneven improvements depending on domain structure, but their integration consistently delivers more robust retrieval.

Table 5: Cost statistics of running different methods across benchmarks.

Benchmark	Model	Method	Avg Turns Per Task ↓	Total # LLM Calls ↓	Total Time
OSWorld	Qwen-2.5-VL-72B	Greedy	23.80	2,028	∼2.1h
	Qwen-2.5-VL-72B	RAG	22.30	1,984	∼2.1h
	Qwen-2.5-VL-72B	WebDreamer	25.70	41,658	∼43.6h
	Qwen-2.5-VL-72B	R-WoM	27.00	11,515	∼12.0h
	Claude-3.5-Sonnet	Greedy	22.30	1,984	∼0.8h
	Claude-3.5-Sonnet	RAG	18.40	1,683	∼0.7h
	Claude-3.5-Sonnet	WebDreamer	24.60	39,747	∼15.9h
	Claude-3.5-Sonnet	R-WoM	22.80	9,778	∼3.9h
	Claude-3.7-Sonnet	Greedy	21.20	1,889	∼0.8h
	Claude-3.7-Sonnet	RAG	21.00	1,947	∼0.8h
	Claude-3.7-Sonnet	WebDreamer	23.60	38,162	∼15.3h
	Claude-3.7-Sonnet	R-WoM	22.00	9,460	∼3.8h
WebArena	Qwen-2.5-VL-72B	Greedy	12.57	1,544	∼1.6h
	Qwen-2.5-VL-72B	RAG	12.11	1,596	∼1.7h
	Qwen-2.5-VL-72B	WebDreamer	12.53	26,948	∼28.2h
	Qwen-2.5-VL-72B	R-WoM	12.99	7,459	∼7.8h
	Claude-3.5-Sonnet	Greedy	13.37	1,624	∼0.7h
	Claude-3.5-Sonnet	RAG	13.11	1,642	∼0.7h
	Claude-3.5-Sonnet	WebDreamer	11.73	25,213	∼10.1h
	Claude-3.5-Sonnet	R-WoM	12.26	7,049	∼2.8h
	Claude-3.7-Sonnet	Greedy	14.49	1,754	∼0.7h
	Claude-3.7-Sonnet	RAG	14.64	1,668	∼0.7h
	Claude-3.7-Sonnet	WebDreamer	16.87	36,176	∼14.5h
	Claude-3.7-Sonnet	R-WoM	16.17	9,186	∼3.7h

Cost comparison. Table 5 shows that WebDreamer is the most expensive, requiring up to 19 calls per step and tens of thousands of total calls, leading to runtimes exceeding 40h on OSWorld and 10–15h on WebArena with Claude models. Our method (R-WoM) reduces this cost by about 2.5×, yielding roughly 75% fewer calls than WebDreamer and cutting runtime to 8–12h on Qwen and 3–4h on Claude while still outperforming lighter baselines. Although R-WoM is costlier than Greedy or RAG, it strikes a better trade-off between efficiency and stability. Looking ahead, agentic calling strategies could further reduce redundant calls and improve cost efficiency.

A.5 Use of Large Language Models

We utilized Large Language Models (LLMs), such as Claude, exclusively for ancillary support in two main areas: (i) language editing and polishing of the manuscript, and (ii) coding assistance for minor boilerplate tasks, such as generating plotting scripts and small utilities. All model-generated outputs were thoroughly reviewed, modified, and rigorously tested by the authors to ensure their accuracy and appropriateness.