Reinforcement World Model Learning for LLM-based Agents

Xiao Yu[1] **Baolin Peng**[2*] **Ruize Xu**[3] **Yelong Shen**[2] **Pengcheng He**[2]
Suman Nath[2] **Nikhil Singh**[3] **Jianfeng Gao**[2†] **Zhou Yu**[1†]

[1]Columbia University, New York [2]Microsoft Research, Redmond [3]Dartmouth College, Hanover

Abstract

Large language models (LLMs) have achieved strong performance in language-centric tasks. However, in agentic settings, LLMs often struggle to anticipate action consequences and adapt to environment dynamics, highlighting the need for world-modeling capabilities in LLM-based agents. We propose Reinforcement World Model Learning (RWML), a self-supervised method that learns action-conditioned world models for LLM-based agents on textual states using sim-to-real gap rewards. Our method aligns simulated next states produced by the model with realized next states observed from the environment, encouraging consistency between internal world simulations and actual environment dynamics in a pre-trained embedding space. Unlike next-state token prediction, which prioritizes token-level fidelity (i.e., reproducing exact wording) over semantic equivalence and can lead to model collapse, our method provides a more robust training signal and is empirically less susceptible to reward hacking than LLM-as-a-judge. We evaluate our method on ALFWorld and τ^2 Bench and observe significant gains over the base model, despite using no expert data, task-success rewards, or strong-LLM annotations—only an off-the-shelf embedding model as a distance metric. When combined with task-success rewards, our method outperforms direct task-success reward RL by 6.9 and 5.7 points on ALFWorld and τ^2 Bench respectively; it further outperforms expert-data training on ALFWorld and is comparable to it on τ^2 Bench.

1 Introduction

Large language models (LLMs) have achieved remarkable success in a wide range of language-centric tasks, including question answering, code generation, and multi-step reasoning [7, 72, 34, 55, 17, 49]. These advances have motivated growing interests in using LLMs as autonomous agents to interact with realistic environments and complete long-horizon tasks [78, 18]. Despite strong linguistic and reasoning abilities, LLM-based agents struggle in many agentic settings that require anticipating action consequences and adapting to environment dynamics [40]. This discrepancy highlights the distinction between *language competence* from pretraining and *agentic intelligence* required for LLM-based agents.

A key reason for this limitation is the misalignment between standard pretraining objectives and agentic use cases. Standard pretraining objectives such as next-token prediction over static text corpora emphasize language understanding and generation. In contrast, modern LLM-based agents operate in complex, long-horizon environments, where successful task completion requires reasoning about both the current state and how the environment might evolve in response to actions [35, 27, 26]. The ability to model potential future outcomes of one's actions is central to biological intelligence. Research

*Project lead. †Equal advising.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

in neuroscience and psychology shows that humans, animals, and intelligent systems use internal world models to reason, plan, explore, and learn efficiently from very few trials [14, 67, 16, 15, 6]. We believe this capacity for world modeling is likewise essential for effective reasoning and planning in LLM-based agents.

Recent work has explored equipping LLM-based agents with world-modeling capabilities, training LLMs to predict next-states using next-token prediction (i.e., SFT). Examples include Zhang et al. [84], Yu et al. [81] which teaches LLMs to model environment transitions using trajectories provided by expert policies or high-quality synthetic data generated with stronger language models. While effective in some settings, these methods face scalability challenges: (1) they rely heavily on high-quality data from experts/strong LLMs; and (2) they are based on SFT, which prioritizes token-level fidelity (i.e., reproducing exact wording) over semantic equivalence and can lead to model collapse.

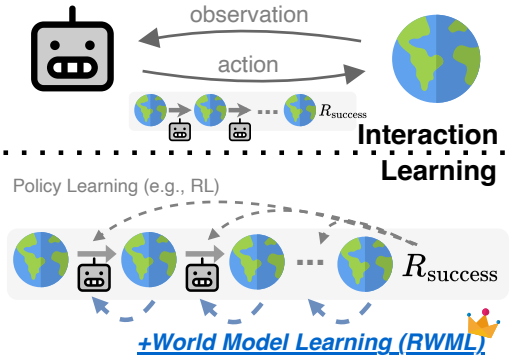


Figure 1: We propose RWML as a scalable, self-supervised method to improve the world modeling ability of LLM-based agent by learning from next-states, prior to downstream policy RL which learns from task-success reward.

In this paper, we propose Reinforcement World Model Learning (RWML), a self-supervised RL method for learning action-conditioned world models in LLM-based agents. The key intuition is that interaction data provides transition supervision: each action is followed by an observed next state, which can be used to improve the model's environment understanding. Concretely, RWML trains the LLM to minimize the discrepancy between simulated and realized next states, measured primarily in a pre-trained embedding space, encouraging semantic consistency with environment dynamics rather than surface-level token matching. This learned world knowledge can then transfer to better decision-making and downstream policy RL. RWML combines two design choices that distinguish it from prior work: unlike separate-world-model methods [65, 8, 25, 73], it trains a *single* LLM for both world modeling and policy learning; and unlike SFT-based single-model methods [84, 81], it uses RL to optimize semantic sim-to-real alignment rather than token-level fidelity. We evaluate our method on two long-horizon agent benchmarks (ALFWorld and τ^2 Bench), and find that RWML significantly improves the base model performance by 19.6 and 6.9 points without using any expert data, generative strong-LLM annotations, or task-success reward signals, but only an off-the-shelf embedding model used as a distance metric. When RWML is used as a mid-training stage before policy RL with task-success rewards, the resulting agents outperform direct policy RL by 6.9 and 5.7 points on ALFWorld and τ^2 Bench, respectively. They also outperform expert-data training on ALFWorld and are comparable on τ^2 Bench.

In summary, our contributions are: (1) We propose RWML as a scalable, self-supervised training method for LLM-based agents that learns action-conditioned world models from sim-to-real gap rewards. (2) We evaluate our method on two long-horizon benchmarks (ALFWorld and τ^2 Bench) and find that RWML significantly improves base model performance. When combined with task-success rewards, our models outperform standard RL, and outperform or match expert-data training on ALFWorld and τ^2 Bench, respectively. (3) We conduct comprehensive analyses — including model forgetting, qualitative analysis, and more — to highlight the benefits of RL and world model learning for LLM-based agents.[1]

2 Method

2.1 Notation

Completing tasks in complex, long-horizon environments is typically formulated as a Markov Decision Process of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. In the generic setting of multi-step tasks, an LLM-powered agent π_θ receives a task instruction and an observation[2] from the environment $s_t \sim \mathcal{S}$, generates an

[1] Code is available at `https://github.com/jasonyux/RWML`.

[2] Technically, any input to the agent from our environments is an observation (as in POMDP). However, to simplify notation we use s to generally denote input data received from the environment.

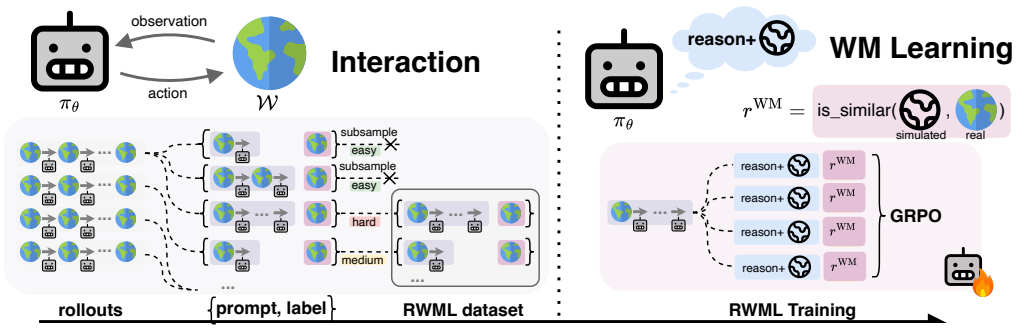


Figure 2: Overview of RWML. Given a target model π_θ, we first collect training data for RWML by using π_θ to gather rollouts $(s_0, a_0, s_1, a_1, ...s_T)$ with the environment, and then convert these rollouts into $\langle s_{\leq t}, a_t, s_{t+1} \rangle$ triplets for all t, after subsampling "too easy" samples defined in Equation (1). We then train π_θ to reason as a world model via GRPO, using lightweight reward functions (e.g., embedding-based cosine similarity) to compare the predicted $\hat{s}_{t+1}$ with the real s_{t+1}.

action $a_t \sim \pi(\cdot|s_t)$, and receives a new observation $s_{t+1} \sim \mathcal{S}$. During action generation, the model is often given up to H turns of interaction history $\langle s_{t-H}, a_{t-H}, ..., s_t \rangle$, and is allowed to think/reason before generating the next action a_t. This interaction process is repeated until the task completion or reaching a maximum number of steps, upon which a terminal reward $r_T \sim \mathcal{R}(a_T, s_T)$ is returned based on whether the task is failed/completed successfully. The discounting factor $\gamma \in (0, 1]$ is used to discount and propagate future rewards during RL training. Note that since this work trains LLMs as world models, *we denote generated states as* $\hat{s}_t$ to distinguish them from real states s_t.

For example, in environments such as ALFWorld [60], an action a_t may be "go to sidetable 1", and the resulting state s_{t+1} describes the outcome of that interaction, such as the objects currently available to agent (e.g., "You arrive at sidetable 1. On the sidetable you see a mug, a pepper shaker, and a tomato."). In more complex environments such as τ^2 Bench [5], an action a_t could be a tool-call or a response to the user, and the next state s_{t+1} returns either a tool response (often in json format), or a natural language response generated by the user simulator (powered by an LLM). For more details on each environment, please see Sections D and E, respectively.

2.2 Reinforcement World Model Learning

A key challenge in scaling agentic post-training methods such as RL is their reliance on accurate task-success rewards provided at the end of an episode. While effective, these rewards are sparse and require careful design by domain experts [5, 74, 53, 88]. As tasks and environments become more complex, this reliance introduces scaling challenges.

We introduce Reinforcement World Model Learning (RWML), a scalable, self-supervised *mid-training* method where the agent learns accurate world model knowledge from the environment dynamics $\mathcal{T}$, before further post-training with task-success rewards (e.g., RFT or policy RL). Intuitively, RWML trains an LLM policy π_θ to *also be able to reason about the consequences* $\hat{s}_{t+1}$ given an action a_t and a history H:

$$(\text{reasoning}, \hat{s}_{t+1}) \sim \pi_\theta(\cdot|s_{\leq t}, a_t); s_{\leq t} \equiv \langle s_{t-H}, a_{t-H}, ..., s_t \rangle$$

where "reasoning" denotes reasoning tokens generated by the model before generating the final prediction of the next state $\hat{s}_{t+1}$. To evaluate the quality of the prediction, we use a simple reward function, typically binarized for robustness[3] that compares the distance between $\hat{s}_{t+1}$ and the ground truth s_{t+1}:

$$r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1}) = \begin{cases} 1.0, & \text{if } d(\hat{s}_{t+1}, s_{t+1}) < \tau_d, \\ 0.0, & \text{otherwise.} \end{cases}$$

where τ_d is a hyperparameter, and d is implemented mainly using an off-the-shelf embedding model $E(\cdot)$ such as Qwen3-Embedding-8B [85] with cosine similarity [32, 85]: $d(\hat{s}_{t+1}, s_{t+1}) = 1 - \cos(E(\hat{s}_{t+1}), E(s_{t+1}))$. To optimize this reward, we use standard GRPO [57, 17]:

$$\mathbb{E}_{(x,y)\sim\pi_{\theta_{\text{old}}}} \left[\min\left(\rho_\theta A, \text{clip}(\rho_\theta, 1-\epsilon, 1+\epsilon)A\right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\theta_{\text{ref}}}) \right],$$

[3] Empirically, we find that binarized rewards are more robust and less susceptible to hacking (see Section 3.4).

where $\rho_\theta = \pi_\theta(y|x)/\pi_{\theta_{\text{old}}}(y|x)$ is the importance sampling ratio, β is the KL regularization coefficient, and $A = [r^{\text{WM}} - \text{mean}(r^{\text{WM}})]/\text{std}(r^{\text{WM}})$ is the group-relative advantage using our reward function. We note that the entire process does not require any expert data, stronger LLMs, or task-success reward signals. We present example prompts and outputs generated by the model during RWML training in Tables A5 and A6.

To collect training data for RWML, we directly use the target model π_θ to gather rollouts $(s_0, a_0, s_1, a_1, ...s_T)$ with the environment, and then convert the rollouts into triplets of $\langle s_{\leq t}, a_t, s_{t+1}\rangle$ for all t. To improve coverage and diversity, we perform $N > 1$ rollouts per training task. To help the model focus on learning non-trivial world model knowledge during RL, we follow intuitions from [62, 64] and subsample the portion of the dataset that are "too easy" to learn. Specifically, we first use SFT to train a separate LLM π'_θ capable of predicting $\hat{s}_{t+1}$ using 10% of the full dataset. Then, we use this π'_θ to generate $\hat{s}_{t+1}$ on the other 90% of the dataset (i.e., the training split), subsampling training samples that consistently achieve high reward through $K = 10$ attempts:

$$\frac{1}{K}\sum_{K} r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1}) > \tau_{\text{easy}}, \tag{1}$$

where τ_{easy} is a hyperparameter. For "easy" samples above the threshold, we only include them in the final training split with probability $p = 0.1$, prioritizing harder samples while preserving diversity. This resulting dataset is used for GRPO training in RWML, as described in the previous section. An overview of the entire process is shown in Figure 2.

3 Experiments

We evaluate RWML on two widely used long-horizon environments that require accurate world and tool understanding for effective planning and task completion.

3.1 Experiment Setup

Benchmarks We conduct experiments on two popular agent benchmarks, ALFWorld [60] and τ^2 Bench [5]. ALFWorld is a text-based embodied environment where the agent needs to locate and interact with objects to complete household tasks using natural language instructions. τ^2 Bench is an interleaved tool-use environment where the model acts as a customer service agent and uses tool-calls to resolve issues while conversing to a simulated user who raised the issue. We use the official training and test splits provided by each benchmark for training and evaluation.

Baselines We compare with other policy and world model related training methods from three categories: (1) learning from task-success reward; (2) learning from interaction/transition function $\mathcal{T}$, similar to our method; and (3) learning from expert annotations/stronger LLMs.

1. **Learning from task-success reward**: we consider Reinforced Finetuning (RFT) using rejection sampling and standard RL with task-success reward (Policy RL). RFT first uses the target model to rollout N trajectories per training task and then performs SFT training only on the trajectories that correctly solved the task [68, 82]. Policy RL directly uses GRPO to train the base model π_θ to optimize for task-success reward using online rollouts [22, 79].
2. **Learning from interaction/transition function**: we consider World Model SFT (WM SFT) which uses identical training data as RWML, but trains the model to directly predict s_{t+1} using SFT. Note that no reasoning is involved in WM SFT as only s_{t+1} is available.
3. **Learning from expert/strong LLMs**: we compare with recent world-model learning methods that use expert trajectories or strong-LLM annotations, including Implicit World Modeling (IWM) and Self-Reflection (SR) [84, 81]. Using expert rollouts $(s_0, a^*_0, s_1, a^*_1, ...)$, these methods first augment them with alternative, non-optimal action-state pairs (s_t, a'_t) generated by the target model π_θ. Then, either these data are converted to next-state prediction triplets $\langle s_{\leq t}, a_t, s_{t+1}\rangle$ for WM learning, or a strong LLM is used to synthesize reasoning data (contrasting expert actions with alternative non-optimal actions) for reflection learning. Finally, these data are combined with the expert policy data (i.e., predict a^*_{t+1}), and SFT is used to train on the combined dataset. Since these two methods heavily rely on expert rollouts, we also consider a simpler baseline that directly learns the expert policy

Table 1: Performance on ALFWorld and τ^2 Bench. All results are averaged over 3 runs, with a maximum step of 30. Our methods are highlighted in gray. *We use Qwen3-235B-A22B-instruct for ALFWorld, and Qwen3-235B-A22B-thinking for τ^2 Bench.

Method	ALFWorld			τ^2 Bench			
	ID	OOD	AVG	Retail	Telecom	Airline	AVG
REACT(Qwen2.5-7B)	16.2$_{\pm 1.0}$	6.8$_{\pm 2.0}$	13.0$_{\pm 1.3}$	15.0$_{\pm 2.0}$	27.5$_{\pm 0.0}$	18.3$_{\pm 2.4}$	20.7$_{\pm 0.9}$
REACT(Qwen3-8B)	40.9$_{\pm 1.5}$	31.3$_{\pm 2.6}$	37.7$_{\pm 1.8}$	37.7$_{\pm 4.9}$	31.2$_{\pm 4.2}$	21.6$_{\pm 5.2}$	31.9$_{\pm 2.9}$
REACT(Qwen3-235B*)	38.0$_{\pm 0.4}$	32.3$_{\pm 2.7}$	36.1$_{\pm 0.7}$	50.6$_{\pm 2.7}$	48.8$_{\pm 3.8}$	51.3$_{\pm 5.5}$	50.0$_{\pm 3.4}$
REACT(GPT-4.1)	42.5$_{\pm 1.0}$	47.4$_{\pm 0.7}$	44.1$_{\pm 0.5}$	55.8$_{\pm 2.4}$	41.7$_{\pm 4.3}$	48.3$_{\pm 2.4}$	48.7$_{\pm 4.5}$
REACT(GPT-5)	51.6$_{\pm 1.3}$	44.8$_{\pm 0.7}$	49.3$_{\pm 0.9}$	55.8$_{\pm 7.2}$	65.0$_{\pm 5.4}$	55.0$_{\pm 4.1}$	59.3$_{\pm 0.5}$
Learning from task success reward							
RFT	34.4$_{\pm 3.8}$	34.4$_{\pm 3.4}$	34.4$_{\pm 2.6}$	43.3$_{\pm 3.1}$	33.3$_{\pm 3.1}$	13.3$_{\pm 2.4}$	33.3$_{\pm 1.7}$
Policy RL	82.1$_{\pm 3.6}$	79.2$_{\pm 2.0}$	81.0$_{\pm 1.6}$	40.8$_{\pm 1.2}$	39.2$_{\pm 1.2}$	30.0$_{\pm 8.2}$	38.0$_{\pm 1.6}$
Self-Supervised							
WM SFT	3.1$_{\pm 0.0}$	2.1$_{\pm 0.7}$	2.8$_{\pm 0.3}$	32.3$_{\pm 5.3}$	24.1$_{\pm 6.1}$	26.9$_{\pm 6.6}$	27.9$_{\pm 3.1}$
RWML (ours)	34.4$_{\pm 0.6}$	29.2$_{\pm 7.5}$	32.6$_{\pm 2.1}$	40.8$_{\pm 4.0}$	40.5$_{\pm 4.9}$	31.3$_{\pm 6.7}$	38.8$_{\pm 2.5}$
Self-Supervised + Policy RL							
WM SFT + Policy RL	76.2$_{\pm 3.4}$	82.3$_{\pm 0.7}$	80.4$_{\pm 1.5}$	40.8$_{\pm 4.2}$	45.0$_{\pm 5.4}$	30.0$_{\pm 7.1}$	40.3$_{\pm 3.9}$
RWML + Policy RL (ours)	**86.7**$_{\pm 2.8}$	**90.1**$_{\pm 0.7}$	**87.9**$_{\pm 1.6}$	**44.2**$_{\pm 2.1}$	**45.8**$_{\pm 2.4}$	**38.3**$_{\pm 2.4}$	**43.7**$_{\pm 2.1}$

Table 2: Comparing ours against training methods that uses expert data/strong LLMs. Imitation Learning, IWM, and SR are reproduced following Zhang et al. [84], which reports 78.1, 82.8, 82.0 for ID and 64.1, 70.3, 71.1 for OOD on ALFWorld, respectively. Highest score is in **bold**, second highest score is in *underline*.

Method	ALFWorld			τ^2 Bench			
	ID	OOD	AVG	Retail	Telecom	Airline	AVG
Learning from experts/strong LLMs							
Imitation Learning	84.9$_{\pm 1.9}$	77.6$_{\pm 3.2}$	82.5$_{\pm 2.3}$	**48.3**$_{\pm 1.2}$	41.7$_{\pm 3.1}$	38.3$_{\pm 2.4}$	*43.7*$_{\pm 1.3}$
IWM	*85.6*$_{\pm 1.6}$	78.1$_{\pm 1.6}$	83.1$_{\pm 1.0}$	40.8$_{\pm 4.3}$	44.2$_{\pm 5.1}$	**46.7**$_{\pm 2.4}$	43.3$_{\pm 2.6}$
SR	83.9$_{\pm 1.0}$	*82.3*$_{\pm 0.7}$	*83.3*$_{\pm 0.4}$	*45.0*$_{\pm 3.5}$	**45.8**$_{\pm 8.3}$	*43.3*$_{\pm 2.4}$	**45.0**$_{\pm 3.6}$
Self-Supervised + Policy RL							
WM SFT + Policy RL	76.2$_{\pm 3.4}$	*82.3*$_{\pm 0.7}$	80.4$_{\pm 1.5}$	40.8$_{\pm 4.2}$	*45.0*$_{\pm 5.4}$	30.0$_{\pm 7.1}$	40.3$_{\pm 3.9}$
RWML + Policy RL (ours)	**86.7**$_{\pm 2.8}$	**90.1**$_{\pm 0.7}$	**87.9**$_{\pm 1.6}$	44.2$_{\pm 2.1}$	**45.8**$_{\pm 2.4}$	38.3$_{\pm 2.4}$	*43.7*$_{\pm 2.1}$

using SFT (denoted as "Imitation Learning"). For more implementation details, please refer to Sections D.3 and E.4.

In addition to these training-based methods, we also evaluate REACT-style prompting [78] on closed-source LLMs such as GPT-5 [61] as additional references. For a more high-level comparison between these methods and our approach, please see Table A1.

Models and Training Data Following prior work [22, 79, 84], we train from Qwen2.5-7B-Instruct [52] on ALFWorld for all methods. On τ^2 Bench, we train from Qwen3-8B [76] for all methods, due to the difficulty of the benchmark and the enhanced tool-use capabilities from Qwen3 models.

For RWML, we collect interaction data using π_θ to rollout N trajectories per training task with temperature $\tau = 1.0$, with $N = 3$ for ALFWorld and $N = 6$ for τ^2 Bench. Then, we split all turns into triplets of $\langle s_{\leq t}, a_t, s_{t+1} \rangle$ for all t, using 90% of the triplets for training and 10% for validation. Finally, we subsample "simple" training samples using a τ_{easy} that corresponds to $\sim$30% of the training data for both benchmarks. In contrast to our baselines, we note that the entire process **does not require any expert annotation/stronger LLMs nor require task success/failure signals**. Only triplets of $\langle s_{\leq t}, a_t, s_{t+1} \rangle$ are required.

Finally, for Policy RL training, we use GRPO to let the model learn to solve the tasks using task-success rewards with $\gamma = 1.0$. For ALFWorld, we follow prior work [79] and allow a maximum of 30 steps per task. For τ^2 Bench, due to cost concerns we train and evaluate using Qwen3-235B-

A22B-Instruct [76] as the user simulator, and allow a maximum step of 30 per task. For evaluation results using the official setting (GPT-4.1 as user simulator), please refer to Section E.3. All trainings are performed with B200 GPUs. For more training and hyperparameter details, please see Section D and Section E for ALFWorld and τ^2 Bench, respectively.

3.2 Main Results

In Table 1 we demonstrate the effectiveness of RWML as a self-supervised method, trained solely from interaction data. Without using any expert data, generative strong-LLM annotations, or task-success reward signals (but only an off-the-shelf embedding model as the distance metric), RWML significantly improved agentic capability compared to the base model, advancing 19.6 and 6.9 points on ALFWorld and τ^2 Bench, respectively. When combined with task-success reward (i.e., Policy RL), we find our models outperform all other training-based baselines. Notably, in Table 2 we find (1) on ALFWorld, our models even outperform approaches that use expert annotations/strong LLMs; and (2) on τ^2 Bench, our models achieve the second best overall score (43.7 vs. 45.0), comparable to the best within error bars, despite not accessing any expert data/strong LLMs. This demonstrates the effectiveness of RWML, whose scalable, self-supervised design represents a promising direction for "mid-training" algorithms that can complement post-training methods such as Policy RL to further improve performance. For a qualitative and quantitative analysis of how RWML improves model decision-making, please see Section 4.1.

For additional results such as world modeling accuracy before and after RWML and a preliminary extension to even more complex environments such as TerminalBench, please see Section F.

3.3 RWML Forgets Less

Beyond improving performance on the target environment, a key advantage of RL-based methods such as RWML is that they often better preserve prior knowledge and capabilities than SFT-based methods. In Table 3, we evaluate this by comparing the susceptibility of RWML and WM SFT to catastrophic forgetting [33, 43]. We evaluate our models trained on ALFWorld and τ^2 Bench on (1) general knowledge benchmarks such as MMLU-Redux [23] and IFEval [87]; (2) math and STEM problems such as MATH-500 [38], GSM8k [13], and GPQA-Diamond [54]; and (3) coding tasks such as LiveCodeBench [29]. In Table 3, we find RWML leads to less model forgetting compared to WM SFT on nearly all benchmarks. We believe this is consistent with findings from prior work [58, 9], that online RL preserves prior knowledge and capabilities significantly better than SFT due to its on-policy nature. For more analysis on model parameter updates, please see Section 4.2.

3.4 Ablation Studies

In Table 4, we ablate key components of RWML: replacing the embedding-based reward with LLM-as-a-judge [86], removing data subsampling ("w/o subsample"), and removing RWML training entirely ("w/o training"). For LLM-as-a-judge, we consider two variants: prompting the LLM to compare the generated $\hat{s}_{t+1}$ with the ground truth s_{t+1} and return a *real-valued reward* $r \in [0, 1]$, allowing for partial credits. We denote this as "w/ LLM-as-a-judge". Alternatively, we prompt the LLM to return a *binary reward* of either 0.0 or 1.0. We denote this as "w/ bin(LLM-as-a-judge)". In both cases, we use Qwen-3-235B-A22B-Instruct [76] as the judge model as it is a fast, strong, open-source LLM that can be hosted locally.

Results in Table 4 show that each component contributes to RWML's performance. First, we find weaker models such as Qwen2.5-7B are more sensitive to noisy rollout data: on average, 43.2% of its actions are invalid given the current observation. Without subsampling, these invalid transitions enter training and can dominate the reward signal, causing model collapse[4]. Second, LLM-as-a-judge can be unreliable and sometimes rewards hacked responses (see Section G), while subsampling "easy" training samples further improves performance. Finally, to isolate the effect of the next-state prediction objective from mere environment exposure or better formatting, we train a control with the same data and budget but a format-only reward: it improves only marginally over no training (15.1 vs. 13.0 on ALFWorld; 32.2 vs. 31.9 on τ^2 Bench), confirming that the gains come from world

[4]To verify this, we re-run the ablation with scripted rollouts (0.0% invalid actions) and find that removing subsampling no longer causes a large drop, confirming that the collapse is driven by data quality rather than the training method (Table A11).

Table 3: Measuring forgetting after training on ALFWorld and τ^2 Bench. For LiveCodeBench, we use questions between 2025-01-01 and 2025-04-30. Evaluation is done with temperature of 1.0 and max response length of 16k using EvalScope [45]. Largest performance degradation (Δ) is highlighted in dark red. Best viewed in color.

		ALFWorld			τ^2 Bench		
		Qwen2.5-7B	+WM SFT	+RWML	Qwen3-8B	+WM SFT	+RWML
General	MMLU-Redux	77.26	67.16(Δ-10.10)	74.88(Δ-2.38)	87.75	87.02(Δ-0.73)	87.42(Δ-0.33)
	IFEval	71.34	68.39(Δ-2.95)	69.32(Δ-2.02)	84.46	82.07(Δ-2.39)	83.36(Δ-1.10)
Math & STEM	MATH-500	75.40	71.60(Δ-3.80)	75.40(Δ0.00)	92.80	92.80(Δ0.00)	92.80(Δ0.00)
	GSM8k	91.66	90.45(Δ-1.21)	91.28(Δ-0.38)	96.13	95.53(Δ-0.60)	95.68(Δ-0.45)
	GPQA-Diamond	32.83	25.25(Δ-7.58)	28.79(Δ-4.05)	59.09	57.07(Δ-2.02)	58.08(Δ-1.01)
Coding	LiveCodeBench	19.23	15.38(Δ-3.85)	16.48(Δ-2.75)	43.41	41.21(Δ-2.20)	43.41(Δ0.00)

Table 4: Ablation studies on RWML. We use Qwen2.5-7B-Instruct on ALFWorld and Qwen3-8B on τ^2 Bench. We find that stronger base models (e.g., Qwen3-8B on τ^2 Bench) is less susceptible to data quality/reward hacking, and that subsampling "too easy" training samples is beneficial to further improve performance.

Method	ALFWorld			τ^2 Bench			
	ID	OOD	AVG	Retail	Telecom	Airline	AVG
RWML (ours)	**34.4**$\pm$0.6	**29.2**$\pm$7.5	**32.6**$\pm$2.1	**40.8**$\pm$4.0	**40.5**$\pm$4.9	**31.3**$\pm$6.7	**38.8**$\pm$2.5
- w/ bin(LLM-as-a-judge)	21.9$\pm$2.4	9.9$\pm$1.9	14.5$\pm$1.3	30.0$\pm$5.4	34.2$\pm$1.2	28.3$\pm$4.7	31.3$\pm$2.9
- w/ LLM-as-a-judge	3.9$\pm$1.0	3.0$\pm$1.2	3.6$\pm$1.3	36.1$\pm$2.5	34.2$\pm$4.3	21.7$\pm$4.7	33.7$\pm$3.9
- w/o subsample	3.1$\pm$1.3	2.6$\pm$1.5	2.9$\pm$1.0	39.2$\pm$6.2	40.0$\pm$2.0	28.3$\pm$6.2	36.3$\pm$2.5
- w/o training	16.2$\pm$1.0	6.8$\pm$2.0	13.0$\pm$1.3	37.7$\pm$4.9	31.2$\pm$4.2	21.6$\pm$5.2	31.9$\pm$2.9

model learning (Table A12). For more results such as using different embedding models, please see Section F.3. Overall, these results highlight the importance of reliable reward metrics and data-quality control in RWML.

4 Discussion

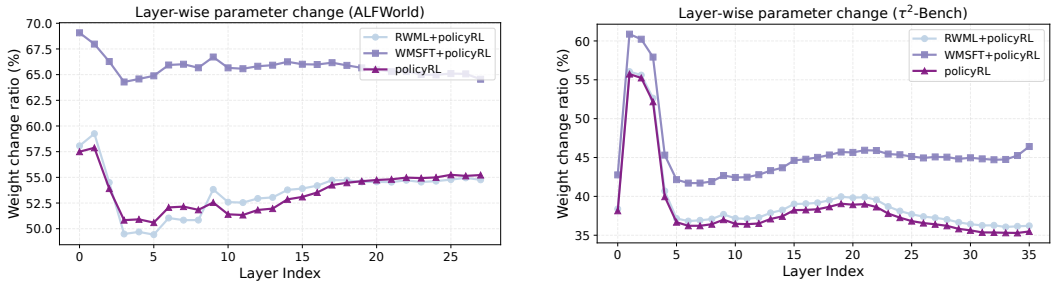
4.1 Impact of RWML on Decision-Making

To better understand how RWML improves task performance (Section 3.2), we analyze the decision-making behavior of RWML-trained models through qualitative and quantitative analysis. Qualitatively, in Figure 5 we find RWML-trained models produce more accurate and efficient decisions, utilizing its improved knowledge about the environment. For example, in ALFWorld, our model correctly predicts that a "knife" is most likely on "countertop" rather than other locations, completing the task within 5 steps. In τ^2 Bench, it correctly considers the possibility that the airplane mode is on — a case omitted by the base model.

Quantitatively, we find RWML effectively mitigates generating invalid/ineffective actions on both benchmarks, despite not being explicitly trained to do so. On ALFWorld, the proportion of invalid (e.g., formatting errors) or inefficient actions (e.g., "look" and "examine" actions) drops from 59.30% to 39.45% after RWML. Similarly, on τ^2 Bench, the proportion of invalid tool calls (e.g., made-up tool names or incorrect arguments) decreases from 24.90% to 8.84% per tool-call made. Overall, our qualitative and quantitative results demonstrate that RWML meaningfully improves the decision-making ability of an LLM in agentic environments.

4.2 Weight Change Analysis

The behavioral differences above motivate a lower-level analysis of how RWML and WM SFT modify the pretrained model parameters. Following Zhu et al. [89], we examine parameter-wise weight changes relative to the untrained base model, adopting the same definition and threshold



(a) ALFWorld (b) τ^2 Bench

Figure 3: Comparing parameter change ratios per layer across models trained with different algorithms. We find WM SFT-trained models shows significantly more parameter change compare to RWML and Policy RL, potentially contributing to model forgetting in Section 3.3.

$\eta = 10^{-3}$ to identify major point-wise updates:

$$|\hat{w}_i - w_i| > \eta \cdot \max(|w_i|, |\hat{w}_i|),$$

where $w_i, \hat{w}_i \in \mathbb{R}$ are finite, non-zero scalars of models' weight points before and after tuning.

For each layer, we compute the ratio of parameters that undergo major updates. Figure 3 shows the effect after downstream policy learning, with full intermediate-checkpoint results in Figure A1a. Across the full results, RWML induces fewer parameter changes than WM SFT, suggesting smaller, more targeted updates that avoid widespread modification of the pretrained representation space.

Importantly, this compact update behavior also helps explain why RWML integrates well with subsequent policy learning, as shown in Figure 3. When followed by Policy RL, its weight-change ratios remain remarkably close to those of Policy RL applied directly to the base model. In contrast, WM SFT exhibit substantially higher change ratios after policy optimization, reflecting stronger parametric interference. These observations suggest that RWML maintains a parameter landscape more compatible with policy learning, reducing conflict and redundancy during post-training.

Overall, we find this parameter update behavior of RWML is consistent across both benchmarks and largely invariant to different transformer components, including attention (Q/K/V/O) and MLP projection layers (see Section H). The consistently lower change ratio of RWML-trained models compared to that of WM SFT also aligns with our findings in Section 3.3, which shows that RWML better mitigates catastrophic forgetting. These results provide a perspective distinct from the conventional SFT-then-RL paradigm: applying RL in both "mid-training" and post-training stages appears to produce more stable, consistent parameter updates, and may help explain the improved performance.

4.3 Impact of Base Model Capability

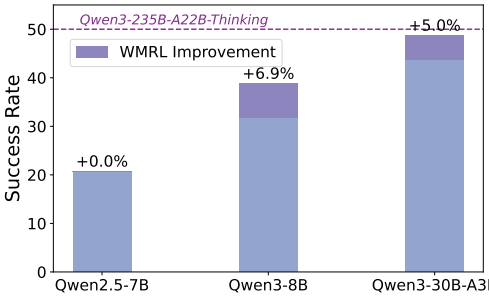


Figure 4: RWML training with different base models on τ^2 Bench.

Finally, we study how RWML scales with base model capability. While RWML provides self-supervised world model training, we find transferring this learned world knowledge to downstream decision-making may require sufficient reasoning and agentic capabilities from the base model. In Figure 4, we perform RWML training with three different base models: Qwen2.5-7B, Qwen3-8B, and Qwen3-30B-A3B[5]. We find that weaker models like Qwen2.5-7B struggle to transfer world knowledge to decision-making on the challenging τ^2 Bench, while stronger models (Qwen3-8B and Qwen3-30B-A3B) show substantial gains, approaching the performance of Qwen3-235B-A22B-Thinking-2507. This suggests that RWML is most effective for (sufficiently) strong base models. We leave improving transfer abilities for weaker models to future work.

[5]We use Qwen3-30B-A3B-Thinking-2507, an enhanced version of Qwen3-30B-A3B post-trained with additional reasoning and agent data, leaving less room for further improvement.

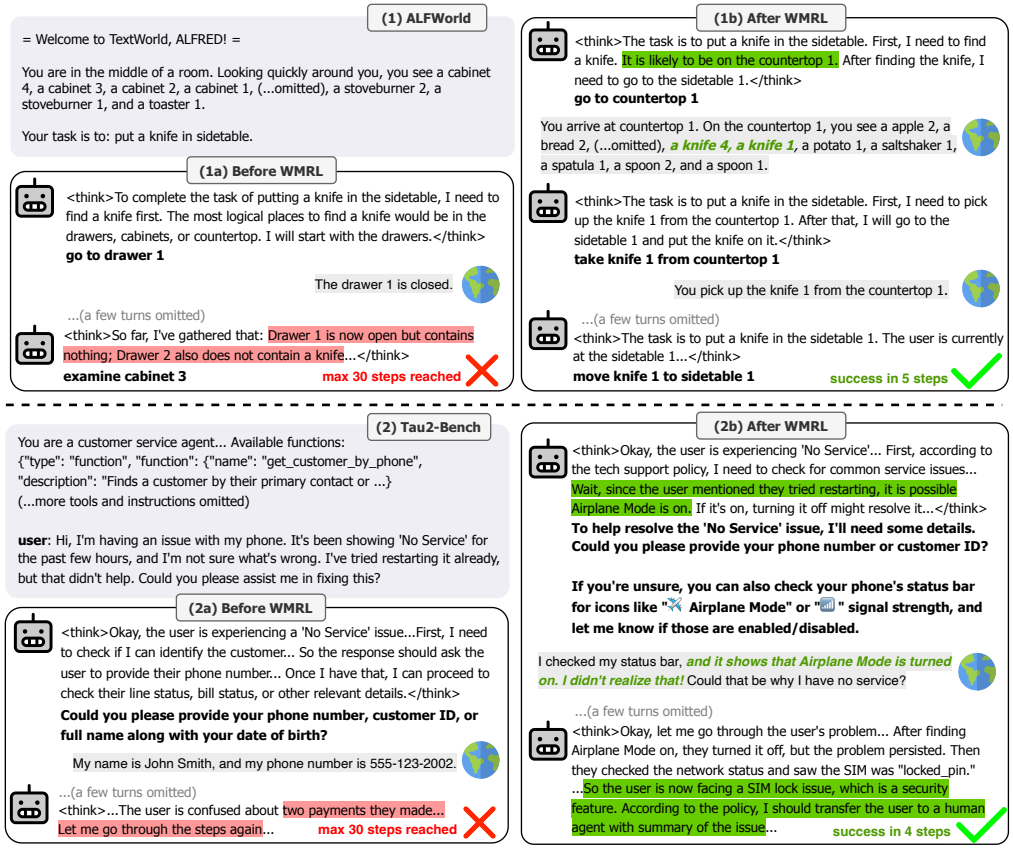


Figure 5: After RWML, models produce more accurate and efficient decisions by leveraging its improved knowledge of the environment.

5 Related Work

Training Decision-Making Agents LLM-based agents [78, 59] are increasingly used in many real-life scenarios such as interactive gaming [69, 22], software engineering [30, 77], and computer/phone/browser-use [74, 53, 88, 80]. Early training methods primarily rely on imitation learning (SFT), using demonstrations from human experts [18, 10, 70] or stronger-LLM trajectories often paired with manually designed workflows/heuristics [83, 11, 63, 75]. While high-quality SFT data offers dense supervision signals, it is difficult to scale due to the high cost of collecting such demonstrations. Recent RL methods instead learn from terminal rewards (i.e., task success) through trial and error [20, 66, 41, 31, 71], often with PPO [56] or GRPO [57]. However, designing task-success rewards in complex environments still requires substantial human expertise [12, 74, 24], limiting scalability. Together, these works motivate the need for more scalable training methods to bridge the gap between next-token-prediction pretrained models and their downstream applications in long-horizon agentic environments.

Training World Models Beyond task-success rewards, interaction data contains rich information for decision-making. Dyna algorithms [65] provide an early example, training a separate world model to combine model-based and model-free learning. Recent LLM-agent methods either train a *separate* world model for inference-time planning such as MCTS [26, 73, 8, 25, 37], or jointly learn world models and policies within a single model to improve generalization [19, 84, 79, 81, 21, 36, 51]. These approaches, however, either require expensive multi-model training/inference or rely on expert/strong-LLM annotations for world model learning. Closest to our work is RLVR-World [73], which also trains world models with RL but differs from RWML in both scope and method. In scope, RLVR-World trains *separate* world models, used for example in predictive control or for evaluating policies, whereas RWML improves the *policy model itself* through a scalable mid-training step, so

that the same model later acts in the environment. In method, for text environments RLVR-World relies on task-specific rewards based on engineered rules (e.g., accuracy over state properties) and performs SFT before RL, whereas RWML uses a generic embedding-based sim-to-real gap reward and performs RL only. Motivated by findings that jointly learning world modeling and acting can outperform separately trained world models [81, 84], RWML focuses on improving a *single* model in a scalable, *self-supervised* manner.

6 Conclusion

We propose RWML, a scalable, self-supervised mid-training method that improves the environment understanding and decision-making ability of LLM-based agents, complementing downstream post-training with task-success rewards such as RFT or policy RL. Without expert data, generative strong-LLM annotations, or task-success signals, RWML trains the LLM as an action-conditioned world model by aligning simulated and observed next states through a sim-to-real gap metric, instantiated primarily with an off-the-shelf embedding model in our experiments. On ALFWorld and τ^2 Bench, RWML achieves significant gains using only interaction data and, when combined with policy RL, outperforms direct policy RL, and further outperforms expert-data training on ALFWorld while remaining comparable to it on τ^2 Bench. We hope RWML motivates scalable, self-supervised training methods for LLM-based agents in the era of agentic RL.

References

[1] AgentInfra Team. All-in-one agent sandbox environment, 2025. URL https://sandbox.agent-infra.com/.

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

[3] Yauhen Babakhin, Radek Osmulski, Ronay Ak, Gabriel Moreira, Mengyao Xu, Benedikt Schifferer, Bo Liu, and Even Oldridge. Llama-embed-nemotron-8b: A universal text embedding model for multilingual and cross-lingual tasks, 2025. URL https://arxiv.org/abs/2511.07025.

[4] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL https://arxiv.org/abs/2404.08471.

[5] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

[6] M.S. Bennett. *A Brief History of Intelligence: Evolution, AI, and the Five Breakthroughs That Made Our Brains*. HarperCollins, 2023. ISBN 9780063286368. URL https://books.google.com/books?id=tymCEAAAQBAJ.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[8] 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, 2025. URL https://arxiv.org/abs/2410.13232.

[9] Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025. URL https://arxiv.org/abs/2510.18874.

[10] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Guicourse: From general vision language models to versatile gui agents, 2025. URL https://arxiv.org/abs/2406.11317.

[11] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models, 2024. URL https://arxiv.org/abs/2403.12881.

[12] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

[13] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[14] K. J. W. Craik. The nature of explanation. *Philosophy*, 19(73):173–174, 1944.

[15] Nathaniel D Daw and Peter Dayan. The algorithmic anatomy of model-based evaluation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130478, 2014.

[16] Nathaniel D. Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711, 2005. URL https://api.semanticscholar.org/CorpusID:16385268.

[17] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[18] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL https://arxiv.org/abs/2306.06070.

[19] FAIR CodeGen team, Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, David Zhang, Kunhao Zheng, Jordi Armengol-Estapé, Pedram Bashiri, Maximilian Beck, Pierre Chambon, Abhishek Charnalia, Chris Cummins, Juliette Decugis, Zacharias V. Fisches, François Fleuret, Fabian Gloeckle, Alex Gu, Michael Hassid, Daniel Haziza, Badr Youbi Idrissi, Christian

Keller, Rahul Kindi, Hugh Leather, Gallil Maimon, Aram Markosyan, Francisco Massa, Pierre-Emmanuel Mazaré, Vegard Mella, Naila Murray, Keyur Muzumdar, Peter O'Hearn, Matteo Pagliardini, Dmitrii Pedchenko, Tal Remez, Volker Seeker, Marco Selvi, Oren Sultan, Sida Wang, Luca Wehrstedt, Ori Yoran, Lingming Zhang, Taco Cohen, Yossi Adi, and Gabriel Synnaeve. Cwm: An open-weights llm for research on code generation with world models, 2025. URL https://arxiv.org/abs/2510.02387.

[20] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025. URL https://arxiv.org/abs/2504.11536.

[21] Jichen Feng, Yifan Zhang, Chenggong Zhang, Yifu Lu, Shilong Liu, and Mengdi Wang. Web world models, 2025. URL https://arxiv.org/abs/2512.23676.

[22] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

[23] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu?, 2025. URL https://arxiv.org/abs/2406.04127.

[24] Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jiménez Gutiérrez, Yiheng Shu, Chan Hee Song, Jiaman Wu, Shijie Chen, Hanane Nour Moussa, Tianshu Zhang, Jian Xie, Yifei Li, Tianci Xue, Zeyi Liao, Kai Zhang, Boyuan Zheng, Zhaowei Cai, Viktor Rozgic, Morteza Ziyadi, Huan Sun, and Yu Su. Mind2web 2: Evaluating agentic search with agent-as-a-judge, 2025. URL https://arxiv.org/abs/2506.21506.

[25] Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your llm secretly a world model of the internet? model-based planning for web agents, 2025. URL https://arxiv.org/abs/2411.06559.

[26] 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, 2023. URL https://arxiv.org/abs/2305.14992.

[27] Zhiting Hu and Tianmin Shu. Language models, agent models, and world models: The law for machine reasoning and planning, 2023. URL https://arxiv.org/abs/2312.05230.

[28] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

[29] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

[30] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

[31] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[32] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020. URL https://arxiv.org/abs/2004.04906.

[33] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

[34] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019. URL https://arxiv.org/abs/1901.07291.

[35] Yann LeCun. A path towards autonomous machine intelligence version, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf.

[36] Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Pony Ma, Guanhua Chen, Heng Ji, and Mengdi Wang. From word to world: Can large language models be implicit text-based world models?, 2025. URL https://arxiv.org/abs/2512.18832.

[37] Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Pony Ma, Guanhua Chen, and Heng Ji. From word to world: Can large language models be implicit text-based world models?, 2026. URL https://arxiv.org/abs/2512.18832.

[38] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[39] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

[40] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025. URL https://arxiv.org/abs/2308.03688.

[41] Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025.

[42] Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. Agrail: A lifelong agent guardrail with effective and adaptive safety detection, 2025. URL https://arxiv.org/abs/2502.11448.

[43] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025. URL https://arxiv.org/abs/2308.08747.

[44] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

[45] ModelScope. EvalScope: Evaluation framework for large models, 2024. URL https://github.com/modelscope/evalscope.

[46] Moonshot AI. Kimi code cli. https://github.com/MoonshotAI/kimi-cli, 2026. GitHub repository.

[47] Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning, 2026. URL https://arxiv.org/abs/2603.14482.

[48] OpenAI. New and improved content moderation tooling. https://openai.com/index/new-and-improved-content-moderation-tooling/, 2022. Accessed: 2025-05-13.

[49] OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[50] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, 2025. URL https://arxiv.org/abs/2412.21139. arXiv:2412.21139, accepted at ICML 2025.

[51] Cheng Qian, Emre Can Acikgoz, Bingxuan Li, Xiusi Chen, Yuji Zhang, Bingxiang He, Qinyu Luo, Dilek Hakkani-Tür, Gokhan Tur, Yunzhu Li, et al. Current agents fail to leverage world model as tool for foresight. *arXiv preprint arXiv:2601.03905*, 2026.

[52] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[53] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. URL https://arxiv.org/abs/2405.14573.

[54] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

[55] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

[56] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[57] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[58] Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

[59] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[60] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL https://arxiv.org/abs/2010.03768.

[61] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, and et al. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

[62] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[63] Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan O. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments, 2025. URL https://arxiv.org/abs/2501.10893.

[64] Yiyou Sun, Yuhan Cao, Pohao Huang, Haoyue Bai, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. Rl grokking recipe: How does rl unlock and transfer new algorithms in llms?, 2025. URL https://arxiv.org/abs/2509.21016.

[65] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

[66] Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. rllm: A framework for post-training language agents. https://pretty-radio-b75.notion.site/rLLM-A-Framework-for-Post-Training-Language-Agents-21b81902c146819db63cd98a54ba5f31, 2025. Notion Blog.

[67] Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189, 1948.

[68] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[69] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

[70] Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Haotian Yao, Ziwei Chen, Qizheng Gu, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua:

Open foundations for computer-use agents, 2025. URL https://arxiv.org/abs/2508.09123.

[71] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

[72] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

[73] Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. Rlvr-world: Training world models with reinforcement learning, 2025. URL https://arxiv.org/abs/2505.13934.

[74] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

[75] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction, 2025. URL https://arxiv.org/abs/2412.04454.

[76] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[77] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

[78] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[79] Xiao Yu, Baolin Peng, Michel Galley, Hao Cheng, Qianhui Wu, Janardhan Kulkarni, Suman Nath, Zhou Yu, and Jianfeng Gao. Dyna-mind: Learning to simulate from experience for better ai agents, 2025. URL https://arxiv.org/abs/2510.09577.

[80] Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning, 2025. URL https://arxiv.org/abs/2410.02052.

[81] Xiao Yu, Baolin Peng, Ruize Xu, Michel Galley, Hao Cheng, Suman Nath, Jianfeng Gao, and Zhou Yu. Dyna-think: Synergizing reasoning, acting, and world model simulation in ai agents, 2025. URL https://arxiv.org/abs/2506.00320.

[82] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

[83] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2023. URL https://arxiv.org/abs/2310.12823.

[84] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience, 2025. URL https://arxiv.org/abs/2510.08558.

[85] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

[86] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

[87] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

[88] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

[89] Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Jinwon Lee, David Z. Pan, Zhangyang Wang, Yuandong Tian, and Kai Sheng Tai. The path not taken: RLVR provably learns off the principals. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=N75EWQQnb3.

A Impact Statements

This paper presents work that aims to advance the agentic capabilities of LLM-based agents through a scalable, self-supervised method. While most LLM-based agent methods are not designed for unethical use, their applications and data collection processes may still pose risks of misuse. In this work, we propose RWML, which improves world modeling in LLM-based agents using interaction data without expert annotations or stronger LLMs, and is trained exclusively on established, isolated benchmarks without real-world impact. We believe that developing guardrails, such as safety filters [48, 28, 42], and using isolated environments like sandboxes [1, 50], is essential for safe AI agent research. We do not condone the use of RWML or its constituent methods for any unlawful or morally unjust purposes.

B Limitations

RWML for more complex/multimodal environments In this work, we mainly consider *challenging, long-horizon text-based environments* such as ALFWorld and τ^2 Bench. However, we believe extending RWML to more complex and multimodal environments is an important future direction. We believe that how to best represent next states in complex environments such as OS, web, and phone-use remains an open question, as also reflected in the JEPA line of work on representation learning [2, 4, 47]. In Section F.4, we present a preliminary extension of RWML to TerminalBench [44] — a benchmark with highly complex coding environments. We leave investigating how to represent and learn next-state representations for complex, multimodal environments to future work.

Distance metric beyond pre-trained embeddings In this work, we mainly consider using off-the-shelf embedding models to compute the distance between the simulated and observed next states. Such a choice is simple and effective, but may not be optimal for all environments and may not capture all the necessary information for effective world modeling. For example, in Section E.1, we

find that for structured tool responses, a generic lexical-overlap metric (ROUGE-L) is more effective than the embedding-based distance, likely because the latter is less sensitive to output structure such as missing keys/values; the embedding-only reward nonetheless remains effective both standalone and after Policy RL (Table A3). While we find *RWML is effective with both metrics*, how to choose and design better distance metrics for different environments and tasks remains an important future direction. We leave this to future work.

Rollout coverage In this work, RWML uses rollouts collected from the target model itself, following common practice in large-scale SFT and synthetic data collection where data is generated from existing models for scalability and ease of use. While this makes data collection simple and *self-supervised*, the resulting rollout distribution may under-cover niche states, rare failure modes, or exploratory actions that the initial model seldom visits. Future work could explore more targeted data collection strategies to improve coverage of such underrepresented transitions.

C LLM Usage

This work used LLMs as general-purpose writing assistants to improve the grammar and clarity of the paper. We *did not* use LLMs to generate any research ideas, automate experiments, or analyze results.

Table A1: Comparing RWML to related methods. *Expert Actions* indicates if the method requires expert rollouts; *LLM Synthetic Data* indicates if the method uses LLM-generated data; *Task Success Reward* indicates if the method requires accessing task-success reward signals for training/data collection.

	Expert Actions	**LLM Synthetic Data**	**Task-Success Reward**
IWM [84, 81]	Required	✗	✗
SR [84, 81]	Required	Required	✗
Imitation Learning	Required	✗	✗
RFT	✗	✗	Required
RWML (ours)	✗	✗	✗

D More Details on ALFWorld

ALFWorld [60] is a text-based, long-horizon agent environment designed to align with the embodied ALFRED benchmark [60]. An ALFWorld task can involve over 50 locations and require more than 50 steps for an expert policy, challenging agents to plan, track subgoals, and explore these locations efficiently. In particular, a key challenge in ALFWorld is identifying likely locations of household items (e.g., desklamps are likely on desks, shelves, or dressers). This makes ALFWorld well suited for evaluating both pretrained commonsense and learned world knowledge of an LLM-based agent.

D.1 RWML Training Setup

To collect training data for RWML (and WM SFT), we use the target model (Qwen2.5-7B-Instruct) to rollout $N = 3$ trajectories per training task with temperature $\tau = 1.0$. We use 2048 tasks from the ALFWorld original training set with a maximum step of 30. We then convert these rollouts to triplets of $\langle s_{\leq t}, a_t, s_{t+1} \rangle$. After minor postprocessing (e.g., removing some samples that contains invalid actions), we obtain 21,011 triplets for training and 2,288 for validation. Then, we finetune a filtering model with SFT on the validation split, and subsampled "too easy" training samples with $\tau_d = 0.1, \tau_{\text{easy}} = 0.0$ which corresponds to $\sim$30% of the original training data. We set $p = 0.1$ to subsample these "too easy" training samples, so that the final dataset retains mostly medium-to-hard examples while preserving sufficient dataset diversity. We note that this value is chosen heuristically without tuning and is also fixed for τ^2 Bench (see Section D.1). An overview of hyperparameter heuristics/intuitions is shown in Table A2. This results in a final training set of 15,813 triplets. For a fair comparison, both RWML and WM SFT are then trained on this final training set.

For r^{WM} during training, we use Qwen3-Embedding-8B [85] with $\tau_d = 0.2$. For RWML, we prompt the model to generate reasoning before making a final prediction of the next state. For WM SFT,

Table A2: Hyperparameters introduced in RWML. *Value* represents value used in ALFWorld, τ^2 Bench, respectively.

Stage	Name	Value	Heuristics	Intuition
RWML Data	τ_d	0.1, 0.15	Set such that "too easy" samples correspond to $\sim$30% of the original dataset	Spending more training on medium-to-hard samples better incentivizes the model to learn non-trivial world-model knowledge.
RWML Data	τ_{easy}	0.0, 0.0	Fixed	Spending more training on medium-to-hard samples better incentivizes the model to learn non-trivial world-model knowledge.
RWML Data	p	0.1, 0.1	Fixed	The final dataset retains mostly medium-to-hard examples while preserving sufficient dataset diversity.
RWML Training	τ_d	0.2, 0.4	Add 0.15 from τ_d used in data collection, then rounded to nearest 0.2	Slightly higher than that of data construction since it uses a fine-tuned model. Values that are too high (e.g., >0.5) make the world-modeling task overly easy, encouraging generic next-state descriptions. Values that are too low (e.g., <0.1) approximate exact matching and become overly strict in many settings.

we directly train the model to predict the next state with empty reasoning tokens. Since there is no reasoning data available for the triplets, we find this training method for WM SFT can better enable generalization/reasoning during the second stage Policy RL training. We present the prompts used for RWML and WM SFT in Table A13 and Table A14, respectively. For RWML, we train over 2 epochs using a learning rate of 1e-6, batch size of 32, group size of 8 with 2xB200 GPUs. For WM SFT, we train over 2 epochs using a learning rate of 2e-6, effective batch size of 32 over 4xB200 GPUs.

D.2 Policy RL Training Setup

For Policy RL, we mainly follow setups and prompts from Feng et al. [22], Yu et al. [79]. We use the official training split from ALFWorld during training, prompting the model to generate reasoning tokens (i.e., `<think>...</think>`) before generating an action. We use a maximum step of 15 during training, using $\gamma = 1.0$ to propagate terminal task success rewards to every turn in the trajectory. We note that this setup is identical to all Policy RL experiments (e.g., RWML+Policy RL and WM SFT+Policy RL). The only differences is the starting model checkpoint. All Policy RL runs are trained with GRPO over 300 steps with a group size of 8 on 2xB200 GPUs. Average training time is 28 hours.

D.3 Other Training Setup

For Imitation Learning, IWM, and SR, we use expert data from the official ALFWorld dataset, following Zhang et al. [84]. For reflection data in SR, we follow Zhang et al. [84] and use a branching factor of 3 per expert action. Then, we use the prompt provided in Zhang et al. [84] to generate reflection data with the target model (i.e., Qwen2.5-7B-Instruct). However, we find that the target model cannot consistently generate coherent reflections, likely due to its limited long-context instruction-following capability. Therefore, following Yu et al. [81], we use a stronger LLM (i.e., GPT-4o) to generate reflections. After data collection, we follow Zhang et al. [84] and performed SFT training by combining the world model data/critic data with the original expert actions dataset. All training is done on 4xB200 GPUs, similar to our WM SFT.

E More Details on τ^2 Bench

τ^2 Bench [5] is a text-based, long-horizon agent environment designed to evaluate the customer service ability of an LLM-based agent in a dual-control environment, where both the agent and user can make use of tool calls to act in a shared, dynamic environment. A τ^2 Bench task, in practice, requires the agent to communicate with the user to gather information, make tool-calls, and adapt to the evolving environment state (e.g., users making tool-calls). Tasks spans across domains such as telecom, retail, and airline support, creating diverse multi-step interactions that challenge agents to plan, adapt, and coordinate with the user. In particular, solving τ^2 Bench tasks requires reasoning about which tool to use (e.g., what information or actions each tool provides) as well as modeling user intent and behavior. This makes τ^2 Bench suitable for evaluating the tool/user understanding and modeling ability of an LLM-based agent.

E.1 RWML Training Setup

To collect training data for RWML (and WM SFT), we use the target model (Qwen3-8B) to rollout $N_{\text{total}} = 6$ trajectories per training task. Specifically, since τ^2 Bench has limited training samples (178 tasks in the training split), we performed rollout with $N = 3$ using GPT-4.1 as the user simulator and $N = 3$ using Qwen3-235B-A22B-Instruct as the user simulator to promote diversity. We then converted all rollouts to triplets of $\langle s_{\leq t}, a_t, s_{t+1} \rangle$. To prevent the model from memorizing database values in the tool responses (e.g.,{"customer_id": "abc123", "full_name": "John Doe"}), we masked these values by converting them to the corresponding OpenAPI schema (e.g., {"type": "object", "properties": {"customer_id": {"type": "string"}, "full_name": {"type": "string"}}}). Similarly, to prevent memorization of user details, we also provide basic user information available to the user simulator (e.g., "I am John Doe. My phone number is 123-456-7890.") in the prompt. We do not provide information regarding the user's intent in the world model learning prompts. We present an example in Table A15 and Table A16. Then, we follow Section D.1 to subsample "too easy" triplets with $\tau_d = 0.15, \tau_{\text{easy}} = 0.0$ and obtained 5,578 triplets for training. We set $p = 0.1$ to subsample these "too easy" training samples, so that the final dataset retains mostly medium-to-hard examples while preserving sufficient dataset diversity. An overview of hyperparameter heuristics/intuitions is shown in Table A2. In total, $\sim$60% of the s_{t+1} are tool-use responses and $\sim$40% are user responses.

For r^{WM} during training, we use Qwen3-Embedding-8B [85] similar to Section D.1. For results with other embedding models, please see Section F.3. However, since tool-use responses are generally structured outputs (e.g., JSON with specific keys and values), embedding-based distances can be insensitive to missing or incorrect fields. We therefore use a generic, off-the-shelf lexical-overlap metric, ROUGE-L [39], as the sim-to-real gap metric for tool-use responses, which better captures such structure. We emphasize that ROUGE-L is a general-purpose metric that is neither designed nor tuned for τ^2 Bench; no task-specific rules or field-level checks are used. Our final reward function in τ^2 Bench is defined as:

$$r^{\text{WM}} = \begin{cases} 1.0, & \text{if } d(\hat{s}_{t+1}, s_{t+1}) < \tau_d \text{ and } s_{t+1} \text{ is a user response,} \\ \text{round}(\text{rouge}(\hat{s}_{t+1}, s_{t+1}), 0.2), & \text{if } s_{t+1} \text{ is a tool-use response,} \\ 0.0, & \text{otherwise.} \end{cases}$$

with $d(\hat{s}_{t+1}, s_{t+1})$ is the embedding-based distance, $\tau_d = 0.4$ as user responses are highly non-deterministic, and round($\cdot$, 0.2) is used to promote better training stability.

Nevertheless, we emphasize that embedding-based rewards alone remain **effective** on τ^2 Bench. As shown in Table A3, RWML (embed-only) still significantly improves over the base model (34.7 vs. 31.9), though the gain is smaller than the ROUGE+embedding combination (38.8). To test whether this holds after downstream training, we further run Policy RL on the embed-only checkpoint with the same 3-run protocol. RWML (embed-only) + Policy RL reaches 40.2 ± 1.4, outperforming direct Policy RL (38.0 ± 1.6) by more than one standard deviation, though it remains below the full RWML + Policy RL (43.7 ± 2.1). We also note that on ALFWorld, the entire pipeline already uses the embedding-only reward. Thus, embedding-based self-supervision alone benefits downstream Policy RL, while a structure-aware (but still generic) metric such as ROUGE-L for tool responses provides further gains on structured outputs. This result reflects a known limitation of off-the-shelf embedding models on structured outputs (e.g., JSON tool responses), where a single missing key or value

Table A3: Ablating the ROUGE term in r^{WM} on τ^2 Bench (Qwen3-8B). "RWML (embed-only)" uses embedding-based reward for both user and tool-use responses. The embedding-only variant still significantly improves over the base model.

Model	Retail	Telecom	Airline	AVG
Base (prompt)	37.7	31.2	21.6	31.9
RWML	40.8	40.5	31.3	38.8
RWML (embed-only)	40.8	32.2	28.3	34.7
Policy RL	40.8	39.2	30.0	38.0
RWML + Policy RL	44.2	45.8	38.3	43.7

Table A4: Performance on τ^2 Bench using the official evaluation setting (GPT-4.1 as user simulator, and a maximum step of 100).

Method	Retail	Telecom	Airline	AVG
REACT(Qwen3-8B)	$42.5_{\pm 2.0}$	$30.0_{\pm 4.1}$	$23.3_{\pm 4.7}$	$33.7_{\pm 0.5}$
REACT(GPT-4.1)	$66.7_{\pm 3.1}$	$50.0_{\pm 2.0}$	$41.7_{\pm 0.0}$	$55.0_{\pm 1.4}$
REACT(GPT-5)	$77.5_{\pm 7.4}$	$97.5_{\pm 2.0}$	$51.7_{\pm 2.4}$	$80.3_{\pm 4.2}$
Learning from experts/strong LLMs				
Imitation Learning	$49.2_{\pm 4.5}$	$50.0_{\pm 2.0}$	$33.3_{\pm 3.3}$	$46.3_{\pm 3.3}$
SR	$52.5_{\pm 3.5}$	$45.8_{\pm 6.5}$	$43.3_{\pm 2.6}$	$48.0_{\pm 2.5}$
Learning from task success reward				
Policy RL	$34.2_{\pm 3.1}$	$45.0_{\pm 7.4}$	$31.7_{\pm 5.3}$	$38.0_{\pm 2.2}$
Self-supervised				
WM SFT	$40.8_{\pm 3.1}$	$30.8_{\pm 1.2}$	$30.0_{\pm 8.2}$	$34.7_{\pm 3.3}$
RWML (ours)	$43.3_{\pm 3.1}$	$47.5_{\pm 2.0}$	$45.0_{\pm 8.2}$	$45.3_{\pm 3.1}$
Self-supervised + Policy RL				
RWML + Policy RL (ours)	$48.3_{\pm 4.7}$	$41.7_{\pm 2.4}$	$50.0_{\pm 0.0}$	$46.0_{\pm 2.8}$

may barely change the embedding distance. It also shows that RWML is compatible with different sim-to-real gap metrics: pairing natural-language states with an embedding distance and structured outputs with a structure-sensitive distance is a simple, general recipe rather than task-specific reward engineering. We leave a deeper study of metric choice, including finetuned embedding models for structured outputs, to future work.

For RWML, we train over 2 epochs using a learning rate of 1e-6, batch size of 32, and group size of 16 with 4xB200 GPUs. For WM SFT, we train over 2 epochs using a learning rate of 2e-6 and an effective batch size of 32 over 4xB200 GPUs.

E.2 Policy RL Training Setup

We extend the codebase from Section D.2 to allow for multi-turn rollouts with τ^2 Bench. We use the official prompts from τ^2 Bench for both training and evaluation. However, since the official setting requires using GPT-4.1 as user simulator, to save cost we use the open-source Qwen3-235B-A22B-Instruct as user simulator during Policy RL. During Policy RL, we use a maximum step of 30 and $\gamma = 1.0$ to propagate terminal task success rewards to every turn in the trajectory. All Policy RL runs are trained with GRPO over 200 steps with a group size of 8 on 8xB200 GPUs. Average training time is 5 days.

E.3 Additional Evaluation Results

To augment our evaluation in Table 1 and Table 2, we additionally evaluate our models using the official setting with GPT-4.1 as user simulator and a maximum step of 100. Since GPT-4.1 is expensive, we evaluate the representative models in each category of Table 1 and Table 2. We present the results in Table A4. In general, we find our method surpasses all training methods that does not use experts/strong LLMs; and is competitive compared to methods that use experts/strong LLMs.

Table A5: Example output generated by the RWML-trained model on ALFWorld. The model first reasons about the action's effect within <think></think> tags, then predicts the next state within <next_state></next_state> tags. Here, the model predicts a semantic description of the next state rather than surface-level object IDs. The outputs in the <next_state> tags and the ground truth are used to compute $r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1})$ during RWML.

Prompt: = Welcome to TextWorld, ALFRED! = You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, (...omitted), a stoveburner 2, a stoveburner 1, and a toaster 1. Your task is to: put a knife in sidetable. (...past action and observation omitted for brevity...) Potential next action: go to countertop 1
Model generation: <think> The action 'go to countertop 1' means the agent will move to countertop 1. After moving, the agent will observe the objects located on that countertop. A countertop typically contains multiple kitchen items such as knives, spoons, and forks, and possibly other items like bowls and bottles. </think> <next_state> Observation: You are now at the countertop 1. On the countertop 1, there is a knife, a spoon, a bowl, a bottle, and a fork. </next_state>
Ground truth next state: On the countertop 1, you see a butterknife 2, a butterknife 1, a dishsponge 3, a dishsponge 1, a fork 2, a fork 1, a mug 2, a soapbottle 1, a spatula 2, a spoon 3, a winebottle 3, and a winebottle 2.

E.4 Other Training Setup

Imitation Learning, IWM, and SR requires expert rollouts for training. However, since τ^2 Bench does not provide expert trajectory annotations, we collect "expert" rollouts using rejection sampling with a strong LLM. Specifically, we use Qwen3-235B-A22B-Thinking-2507 and perform rollouts with $N_{\text{total}} = 6$, with $N = 3$ using GPT-4.1 as user simulator and $N = 3$ using Qwen3-235B-A22B-Instruct (same as Section E.1). We then only keep rollouts that successfully solved the tasks as expert rollouts. For reflection data in SR, we follow the same procedure as ALFWorld (Section D.1). Similar to ALFWorld, due to the weak performance of the target model on the benchmark, we use GPT-4.1 instead of Qwen3-8B to generate reflection data. All training is done on 8xB200 GPUs.

F Additional Examples and Results for RWML Training

F.1 RWML Prompts and Outputs

To illustrate what the predicted next state looks like in practice, we present example outputs generated by our RWML-trained model on ALFWorld and τ^2 Bench in Tables A5 and A6. In each example, the model first generates a reasoning trace within <think></think> tags, then predicts the next state within <next_state></next_state> tags. We also include the ground truth next state from the environment for reference. Note that RWML-trained model learned to (1) predict semantic description of next-state instead of surface forms (e.g., in ALFWorld item ids are NOT generated); and (2) think about a distribution of possible next outcomes for underterministic states such as user responses in τ^2 Bench (see the blue highlighted texts in Table A6).

F.2 World Model Prediction Accuracy After RWML

To quantify the effect of RWML training, we measure next-state prediction accuracy on 512 unseen samples per benchmark before and after RWML. We define accuracy as the fraction of predictions that receive $r^{\text{WM}} = 1$. As shown in Table A7, RWML substantially improves this accuracy on both benchmarks: from 16.8% to 76.2% on ALFWorld, and from 18.9% to 73.8% on τ^2 Bench. Example model predictions after our RWML training are shown in Table A5 and Table A6.

Since the accuracy above is measured with the same reward used for training, we additionally conduct a blinded human evaluation that is independent of r^{WM}. We hired two human annotators to label 100

Table A6: Example output generated by the RWML-trained model on τ^2 Bench. The model first reasons about the user's likely follow-up response within <think></think> tags, then predicts the next state within <next_state></next_state> tags. In this stochastic environment, the model reasons about a distribution of plausible user responses (highlighted in blue) before predicting one likely next state. The outputs in the <next_state> tags and the ground truth are used to compute $r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1})$ during RWML.

Prompt: <instructions>
You are a customer service agent that helps the user according to the <policy> provided below.
In each turn you can either:
- Send a message to the user.
- Make a tool call.
You cannot do both at the same time.
Tools
You may call one or more functions to assist with the user query.
You are provided with function signatures within <tools></tools> XML tags:
<tools>
{'type': 'function', 'function': {'name': 'get_customer_by_phone'}} (...more tools and instructions omitted for brevity...)
user: Hi, I'm having an issue with my phone. It's been showing 'No Service' for the past few hours, and I'm not sure what's wrong. I've tried restarting it already, but that didn't help. Could you please assist me in fixing this?
(...past action and observation omitted for brevity...)
Potential assistant response: Since the tool to check APN settings isn't available, I'll guide you manually. Here's how to check and verify APN settings for your carrier:
Steps to Check APN Settings:
1. Open Settings: Go to Settings > Network & Internet (... other instructions omitted for brevity)
If APN Settings Are Correct:
- Try sending an MMS again.
- If it still fails, your issue might be with the carrier's MMS server or a device-specific setting.
Would you like help finding your carrier's correct APN details?

Model generation:
<think> Okay, let's see. The user is having trouble sending MMS messages. They followed the previous steps: reseated the SIM, enabled mobile data, changed network mode to 4G/5G, and now they have excellent signal. But when they tried sending an MMS, it didn't work. (...omitting some content for brevity) So the assistant sends the message with the manual steps. Then, the user would reply. So, the user might say something like, "I checked the APN settings, and they are set correctly. What should I do next?" or "I found that the APN is incorrect. How do I reset it?" or maybe the user says they can't find the APN settings. So, the next_state would be the user providing feedback after following the assistant's instructions. Therefore, the next_state is the user's response, which could be any of those possibilities. Since the exact response isn't known, but based on the context, the user might be indicating that they checked the APN and MMSC settings and may still be experiencing issues with sending MMS messages. </think>
<next_state> User: I checked my APN settings. MMSC is set to "http://mms.carrier.com/mms/wapenc" according to your instruction, and the MMS proxy and port are correct. I reset the APN settings just to be sure, but I still can't send MMS messages. </next_state>

Ground truth next state:
I restarted the phone and checked my APN settings. Now the MMSC URL is set to "http://mms.carrier.com/mms/wapenc," which looks correct. But I still can't send MMS messages. I'm a bit frustrated—everything seems to be set up right, but it's still not working. What else can we try?

held-out generations per model per benchmark (400 samples in total), without knowing which model produced each prediction. Given the history $s_{\leq t}$, the action a_t, the reference next state s_{t+1} from the environment, and the predicted next state $\hat{s}_{t+1}$, annotators label a prediction as correct only if it captures *all* task-critical information necessary to solve the task; minor missing or incorrect details

Table A7: Next-state prediction accuracy (%) before and after RWML training. *Reward-based*: fraction of 512 unseen samples per benchmark that receive $r^{\mathrm{WM}} = 1$ under the benchmark-specific reward function. *Human-judged*: fraction of 100 held-out samples per model per benchmark that blinded human annotators judge to capture all task-critical information (see Section F.2).

Benchmark	Reward-based		Human-judged	
	Before RWML	After RWML	Before RWML	After RWML
ALFWorld	16.8	76.2	11.0	67.0
τ^2 Bench	18.9	73.8	13.0	61.0

Table A8: Instructions given to human annotators for the blinded evaluation of predicted next states (Section F.2). Each sample page additionally shows one example of a good and one example of a bad prediction, followed by the context, the reference next state, and the predicted next state. The identity of the model that produced each prediction is hidden from the annotators.

Instruction
Your task is to determine the quality of a model's predicted next state in a given context. Specifically, you will be given the current state of the environment that the agent is acting in, an action, and the model's predicted next state, and your task is to determine whether the predicted next state captures all task-critical information necessary to solve the task.

Qualities of a good next state prediction:
- The predicted next state covers most of the content of the ground-truth next state.
- Some minor details in the predicted next state can be missing or incorrect — if and only if the missing or incorrect details likely do not critically relate to solving the task and should NOT mislead/affect the agent's ability to make a correct decision in the next step.

Qualities of a bad next state prediction:
- Critical information related to solving the task is missing or incorrect.
- A very generic description of the next state that does not provide any useful information for the agent to make a correct decision in the next step.

Example of a good next state prediction: *// omitted for brevity*
Example of a bad next state prediction: *// omitted for brevity*

Sample to annotate
Context: {history $s_{\leq t}$}
Next action: {action a_t}
Reference next state: {s_{t+1} from the environment}
Predicted next state: {model's predicted $\hat{s}_{t+1}$}

are tolerated only when they would not affect the agent's next decision. This is a deliberately strict criterion that targets sparse but decision-critical errors (e.g., object identity or location in ALFWorld, tool-call arguments or user intent in τ^2 Bench) that an embedding-based metric may miss. Before the final annotation, both annotators labeled a shared screening set of 25 samples, reaching a Cohen's κ of 0.83. The full annotation instructions are shown in Table A8. As shown in Table A7, human-judged accuracy improves from 11.0% to 67.0% on ALFWorld and from 13.0% to 61.0% on τ^2 Bench. As expected from the strict criterion, human-judged accuracy is lower than reward-based accuracy, but the improvement after RWML is consistent across both measures. This indicates that the gains in Table A7 reflect genuinely better next-state predictions rather than merely satisfying the training proxy.

F.3 Choice of Embedding Model for RWML

We use Qwen3-Embedding-8B [85] across both benchmarks as a lightweight but strong general-purpose embedding model. To study the sensitivity of RWML to this choice, we additionally train on τ^2 Bench with Qwen3-Embedding-4B and Llama-Embed-8B [3]. As shown in Table A9, RWML

Table A9: RWML performance on τ^2 Bench with different embedding models.

Embedding Model	Retail	Telecom	Airline	AVG
– (no RWML training)	37.7	31.2	21.6	31.9 ± 2.9
Qwen3-Embedding-8B	40.8	40.5	31.3	38.8 ± 2.5
Llama-Embed-8B	41.7	39.2	31.3	38.5 ± 2.5
Qwen3-Embedding-4B	37.5	40.8	37.5	37.0 ± 2.1

Table A10: Preliminary RWML results on Terminal-Bench using Qwen3-8B with Kimi-CLI.

Setting	Success Rate (↑)	Tool-call Error Rate (↓)
Before RWML	$3.1\pm0.0\%$	41.8%
After RWML	$5.6\pm1.3\%$	31.6%

Table A11: RWML ablation on ALFWorld with high-quality scripted rollouts vs. Qwen2.5-7B's own rollouts. With clean data, removing subsampling no longer collapses performance.

Setting	Qwen2.5-7B data	Scripted data
w/ subsample	34.4	26.5
w/o subsample	3.1	22.0
- (no training)	16.2	16.2

consistently improves over the base model across reasonable embedding choices, with performance roughly tracking the general quality of the embedding model. In general, as with any reward-based training method, RWML depends on the quality of the sim-to-real gap metric: weak embedding models or metrics poorly matched to structured outputs can provide noisy supervision. This suggests that stronger or task-adapted embedding models may further improve performance, which we view as an orthogonal direction for future work.

F.4 Preliminary Extension to TerminalBench

We also present a preliminary extension of RWML to Terminal-Bench [44], a more complex coding environment. We train on Terminal-Bench Pro and evaluate on Terminal-Bench 2.0 using Qwen3-8B with Kimi-CLI [46]. For next-state representation, we use a simple truncation scheme: tool responses with at most 128 tokens are kept as-is, while longer responses are represented by their first 50 and last 50 tokens.

As shown in Table A10, RWML yields a modest but significant improvement in task success rate, from $3.1\pm0.0\%$ to $5.6\pm1.3\%$. We find that Qwen3-8B still lacks sufficient knowledge for understanding large codebases and generating complex code, which limits the absolute success rate on this difficult benchmark. Nevertheless, RWML reduces the tool-call error rate by over 10 percentage points (41.8% to 31.6%), suggesting improved understanding of Kimi-CLI and the environment despite using only self-supervised training. In general, we believe that how to best represent next states in complex environments such as OS, web, and phone-use remains an open question, as also reflected in the JEPA line of work on representation learning [2, 4, 47]. We leave the use of stronger base models and better next-state representations for complex, multimodal environments to future work.

G More Details on Ablation Studies

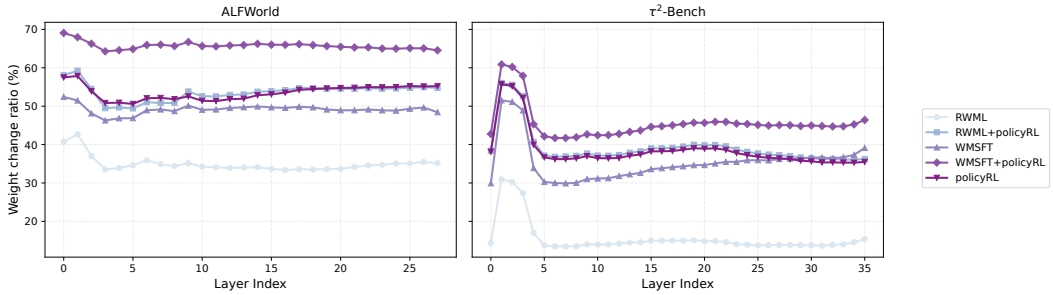
For LLM-as-a-judge in Table 4, we present an example prompt and an example (hacked) response from ALFWorld in Table A17. We note that the model being trained is Qwen2.5-7B-Instruct, whereas the judge model is Qwen3-235B-A22B-Instruct. In general, we find LLM-as-a-judge can be unreliable, awarding high scores to predictions that do not show genuine understanding of the environment dynamics relevant to the task.

Table A12: Isolation control for the next-state prediction objective. "RWML (format-only)" is trained on the *same* interaction data with the *same* training budget as RWML, but its reward only checks whether the output is well-formed, i.e., the next-state prediction reward is removed entirely. We use Qwen2.5-7B-Instruct on ALFWorld and Qwen3-8B on τ^2 Bench.

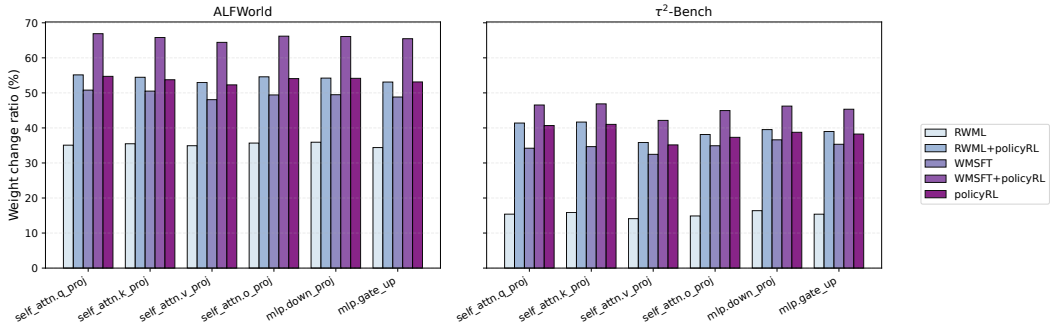
Method	ALFWorld	τ^2 Bench
REACT (no training)	13.0	31.9
RWML (format-only)	15.1	32.2
RWML	**32.6**	**38.8**

A natural alternative explanation for the gains of RWML is that they stem from more exposure to environment trajectories or from improved action formatting, rather than from learned world-model knowledge. To isolate the effect of the next-state prediction objective, we train a control variant, RWML (format-only), on the *same* interaction data and with the *same* training budget as RWML, but with a reward that only checks whether the generated output is well-formed; the next-state prediction reward is removed entirely. As shown in Table A12, format-only training improves only marginally over no training (15.1 vs. 13.0 on ALFWorld; 32.2 vs. 31.9 on τ^2 Bench), whereas full RWML yields much larger gains (32.6 and 38.8, respectively). This indicates that the improvements of RWML stem from the next-state prediction objective rather than from environment exposure or better formatting.

H More Details on Parameter Change Analysis



(a) Layer-wise parameter change ratios by transformer layer (ALFWorld and τ^2-Bench).



(b) Module-wise parameter change ratios aggregated across layers (attention Q/K/V/O and MLP projections).

Figure A1: Parameter change analysis across datasets and training variants. **Top:** layer-wise weight change ratios by transformer layer. **Bottom:** main module-wise change ratios aggregated across layers.

We report a fine-grained analysis of parameter updates at both the layer and module levels. Specifically, we compute the fraction of parameters exhibiting major changes within each transformer layer, with results summarized in Figure A1a. Across both benchmarks, we observe highly consistent trends throughout the network depth. Models trained with RWML display the lowest proportion of updated

Table A13: Prompt for RWML in ALFWorld, given a current state s_t and an action a_t. Final output within the <next_state> </next_state> tags is used to compare with the next state in $r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1})$ during RL.

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to .../*/ omitting task instruction and previous action history here for brevity*
Your current observation is: {current_state} **Potential action:** {action}
Now, your task is to predict the immediate next observation after taking the potential action above. You should first briefly reason step-by-step about the previous steps and current situation — summarize key information you've learned about the environment that is relevant to the task. This reflection and reasoning process **must** be enclosed within <think> </think> tags. Once you've finished your reasoning, you should describe the next observation (use the past and current observations as examples!) and present them within <next_state> </next_state> tags.

parameters, whereas WM SFT leads to noticeably broader parameter modifications. This pattern aligns with the forgetting behavior analyzed in Section 3.3, where RWML models retain prior knowledge more effectively.

In addition, we compare the effects of policy RL when applied to different mid-trained initializations. Notably, the parameter change profiles of policy RL remain largely similar regardless of whether RWML is applied beforehand, and closely resemble those obtained when policy RL is performed directly on the base model. In contrast, initializing policy RL from a WM SFT model results in substantially elevated update ratios. These results indicate that RWML preserves a parameter configuration that is more amenable to subsequent policy optimization, reducing the extent of disruptive updates during post-training.

To further validate that this behavior is not an artifact of layer aggregation, we conduct a module-wise breakdown over attention projections (Q/K/V/O) and MLP projection parameters, as shown in Figure A1b. The observed trends closely mirror the layer-wise results, suggesting that the relative stability induced by RWML is consistent across different architectural components rather than being localized to specific modules.

We also note that WM SFT exposes a practical limitation of SFT for world model learning. Since transition data contains ground-truth next states but not ground-truth reasoning traces, token-level SFT must either omit reasoning supervision or rely on synthetic rationales. The latter would require additional expert or strong-LLM annotations, which is not scalable and contrasts with RWML, which only requires observed next states from interaction data. We also find that the degradation of WM SFT is unlikely to be explained solely by suppressing reasoning tokens: the same pattern appears for both Qwen2.5-7B-Instruct on ALFWorld, which is not a reasoning-specialized model, and Qwen3-8B on τ^2 Bench, where reasoning behavior is more natural.

Overall, these detailed analyses provide additional evidence that employing RL during both mid-training and post-training leads to more coherent and stable parameter update than the conventional SFT-then-RL pipeline.

Finally, we emphasize that the analyses presented here are empirical and are intended to provide descriptive evidence rather than a complete mechanistic explanation. While the observed parameter update patterns offer insights into how RWML influences subsequent training dynamics, a more systematic understanding of RL-based training—particularly from the perspectives of optimization dynamics and mechanistic interpretability—remains an important direction for future work. We hope these findings motivate further investigation into the internal representations and learning trajectories induced by RL at different stages of model training.

Table A14: Prompt for WM SFT in ALFWorld, given a current state s_t and an action a_t. The label is the next state s_{t+1}. Since SFT requires target tokens and our transition data does not contain ground-truth reasoning traces, we train WM SFT with empty `<think> </think>` tags followed by the next-state label. We find this direct-prediction setup gives the best performance for both WM SFT and the subsequent Policy RL stage.

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to .../*omitting task instruction and previous action history here for brevity*

Your current observation is: {current_state}
Potential action: {action}

Now, your task is to predict the immediate next observation after executing the potential action above.
Directly present your final prediction of the next observation (use the past and current observations as examples!) within `<next_state> </next_state>` tags. DO NOT generate anything else.

Label: `<think> </think>`
`<next_state>` {next_state} `</next_state>`

Table A15: Prompt for RWML in τ^2 Bench, given $s_{\leq t}$ and an action a_t. Final output within the `<next_state> </next_state>` tags is used to compare with the next state in $r^{\text{WM}}(\hat{s}_{t+1}, s_{t+1})$ during RL.

You are a customer service agent that helps the user according to the <policy> provided below.../*omitting task instruction, policy, and tool usage details here for brevity*

User Information
The following information about the user is available:
{available_user_info}

History
{current_state}
Potential assistant response: {action}

Your Task
Now, your task is to predict the immediate next user/tool response if the above 'potential assistant response' is used based on the available information above.
Once you've finished your thinking, format your final prediction of the next user/tool response and task completion status within `<next_state> </next_state>` tags. Note that a user response should be written as plain text. A tool response may be a short status message (e.g., no data found, error, transaction success, etc.,) or a JSON object; if it is JSON, only predict the JSON schema in OpenAPI format rather than actual values (e.g., {"type": "object", "properties": {"customer_id": {"type": "string"}, "full_name": {"type": "string"}},...}).

Table A16: Prompt for WM SFT in τ^2 Bench, given $s_{\leq t}$ and an action a_t. The label is the next state s_{t+1}. Since SFT requires target tokens and our transition data does not contain ground-truth reasoning traces, we train WM SFT with empty `<think> </think>` tags followed by the next-state label. We find this direct-prediction setup gives the best performance for both WM SFT and the subsequent Policy RL stage.

You are a customer service agent that helps the user according to the <policy> provided below...// *omitting task instruction, policy, and tool usage details here for brevity*

User Information
The following information about the user is available:
{available_user_info}

History
{current_state}
Potential assistant response: {action}

Your Task
Now, your task is to predict the immediate next user/tool response if the above 'potential assistant response' is used based on the available information above.
DO NOT perform any thinking. Directly present your final prediction of the next user/tool response within `<next_state> </next_state>` tags. Note that a user response should be written as plain text. A tool response may be a short status message (e.g., no data found, error, transaction success, etc.,) or a JSON object; if it is JSON, only predict the JSON schema in OpenAPI format rather than actual values (e.g., {"type": "object", "properties": {"customer_id": {"type": "string"}, "full_name": {"type": "string"}},...}).

Label: `<think> </think>`
`<next_state>` {next_state} `</next_state>`

Table A17: Example prompt for LLM-as-a-judge in our ablation studies. In general, we find LLM-as-a-judge is unreliable and can award high scores to predictions that are not meaningful. "predicted_next_state" are contents extracted from the <next_state></next_state> tags in the generated response.

You are a helpful judge AI agent.
Current state
{current_state}

Action proposed by an AI agent
{action_text}

Predicted next observation
After executing the above action, another AI agent predicted the next observation as follows:
{predicted_next_state}

Actual next observation
The actual next observation from the environment is as follows:
{actual_next_state}

Evaluate the predicted next observation
Now, your task is to evaluate how well the predicted next observation matches the actual next observation. Specifically, you need to judge whether the prediction demonstrates a *genuine* understanding of the environment dynamics relevant to the task, using the actual next observation as reference.

- If all important task-related information in the actual next observation is present in the predicted next observation, assign a score of 1.0.
- Otherwise, assign a score of 0.0.

Your output format
Your task is to output a JSON object in the following format:

```
<json>
{
"analysis": "which important task-related content is
present/missing in the predicted next observation.",
"score": 0.0 or 1.0
}
</json>
```

Directly output the JSON object. DO NOT generate anything else.

Example generation that achieved a score of 1.0:
<think> The potential action is to go to countertop 1. After this action, I should observe that I am at countertop 1, and the contents of the countertop will be displayed. The task is not yet completed as I have not found the tomato or started the process of putting it in the microwave. </think>
<next_state> The immediate next observation after going to countertop 1 will describe the contents of countertop 1 or indicate if there is anything special about the countertop.</next_state>

Example actual next state:
You arrive at countertop 1. On the countertop 1, you see a apple 2, a bread 2, (...omitted), a knife 4, a knife 1, a potato 1, a saltshaker 1, a spatula 1, a spoon 2, and a spoon 1.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim RWML as a scalable, self-supervised method to improve the world modeling ability of LLM-based agent and hence the decision-making performance in long-horizon tasks. Experiments and analyses in the rest of the paper support this claim.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Section B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the training algorithm and data collection process in Sections 2 and 3.1, benchmark-specific training details in Sections D and E, hyperparameter details in Table A2, and LLM prompts in Tables A13 to A16.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release code and data upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We use standard training/test splits from the original ALFWorld and τ^2 Bench papers. We provide benchmark-specific training details, including data splits, hyperparameters, optimizer settings, and reward functions, in Sections D and E, with hyperparameter for RWML summarized in Table A2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report standard deviations across runs in our main results, and our models exceed the baselines by more than 2 standard deviations in most cases.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the GPU type/count and training time or training steps for each major experiment in Sections D and E, including the RWML, WM SFT, policy RL, and other baseline training setups.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper does not involve human subjects, personally identifiable information, or using any sensitive data beyond standard academic benchmarks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please see Section A.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Please see Section A.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the datasets and models used in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release data and code upon acceptance, and we will provide documentation alongside the assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Please see Section C.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.