arXiv:2602.05842v2 [cs.CL] 9 Feb 2026

# 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]
**Jiangfeng Gao** [† 2] **Zhou Yu** [† 1]

## 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 $\tau^2$ Bench and observe significant gains over the base model, despite being entirely *self-supervised*. When combined with task-success rewards, our method outperforms direct task-success reward RL by 6.9 and 5.7 points on ALFWorld and $\tau^2$ Bench respectively, while matching the performance of expert-data training.

## 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 (Brown et al., 2020; Wei et al., 2022; Lample & Conneau, 2019; Rozière et al., 2024; DeepSeek-AI et al., 2025; OpenAI & et al., 2024). These advances have motivated growing interests in using LLMs as autonomous agents to interact with realistic environments and complete long-horizon tasks (Yao et al., 2023; Deng et al., 2023). Despite strong linguistic and reasoning abilities, LLM-based agents struggle in many agentic settings that require anticipating action consequences and adapting to environment dynamics (Liu et al., 2025). This discrepancy highlights the distinction between *language competence* from pretraining and *agentic intelligence* required for LLM-based agents.

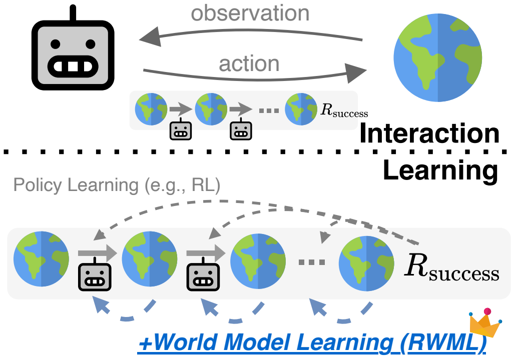


*Figure 1.* We propose RWML as a scalable, self-supervised method to improve the world modeling ability of LLM-based agent by learning from next-states, prior to downstream policy RL which learns from task-success reward.

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 (LeCun, 2022; Hu & Shu, 2023; Hao et al., 2023). The ability to model potential future outcomes of one's actions is central to biological intelligence. Research 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 (Craik, 1944; Tolman, 1948; Daw et al., 2005; Daw & Dayan, 2014; Bennett, 2023). We believe this capacity for world modeling is likewise essential for effective reasoning and planning in LLM-based agents.

---

[*] Project Lead [†] Equal Advising [1] Columbia University, New York [2] Microsoft Research, Redmond [3] Dartmouth College, Hanover. Correspondence to: Xiao Yu <xy2437@columbia.edu>.

*Preprint. February 10, 2026.*

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. (2025a); Yu et al. (2025c) 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.

In this paper, we propose Reinforcement World Model Learning (RWML), a self-supervised training method based on RL that learns action-conditioned world models for LLM-based agents. Rather than optimizing token-level fidelity with SFT, RWML trains LLMs to minimize the discrepancy between simulated next states produced by the model and realized next states observed from the environment, measured in a pre-trained embedding space. This sim-to-real alignment promotes semantic consistency between the agent's internal world model and real environment dynamics while preserving task-relevant transitions, making them suitable for downstream decision-making. We evaluate our method on two long-horizon agent benchmarks (ALFWorld and $\tau^2$ Bench), and find RWML significantly improved the base model performance by 19.6 and 6.9 points without using any expert data, strong LLMs, or task-success reward signal. When combined with task-success rewards, agents trained with RWML outperform direct task-success reward RL by 6.9 and 5.7 points on ALFWorld and $\tau^2$ Bench, respectively, while matching the performance of training with expert data.

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 $\tau^2$ Bench) and find that RWML significantly improves base model performance. When combined with task-success rewards, our models outperform standard RL and match the performance of training with expert data. (3) We conduct comprehensive analyses — including ablation studies, model forgetting, qualitative analysis, and more — to highlight the benefits of RL and world model learning for LLM-based agents.

## 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 $\pi_\theta$ receives a task instruction and an observation[1] from the environment $s_t \sim \mathcal{S}$, generates an 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 environment states $s_t$.

For example, in environments such as ALFWorld (Shridhar et al., 2021), 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 $\tau^2$ Bench (Barres et al., 2025), 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 B and C, 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 (Barres et al., 2025; Xie et al., 2024; Rawles et al., 2025; Zhou et al., 2024). As tasks and environments become more complex, this reliance introduces scaling challenges.

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

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

where "reason" 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 binary[2] reward function that compares the distance

---

[1] 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.

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

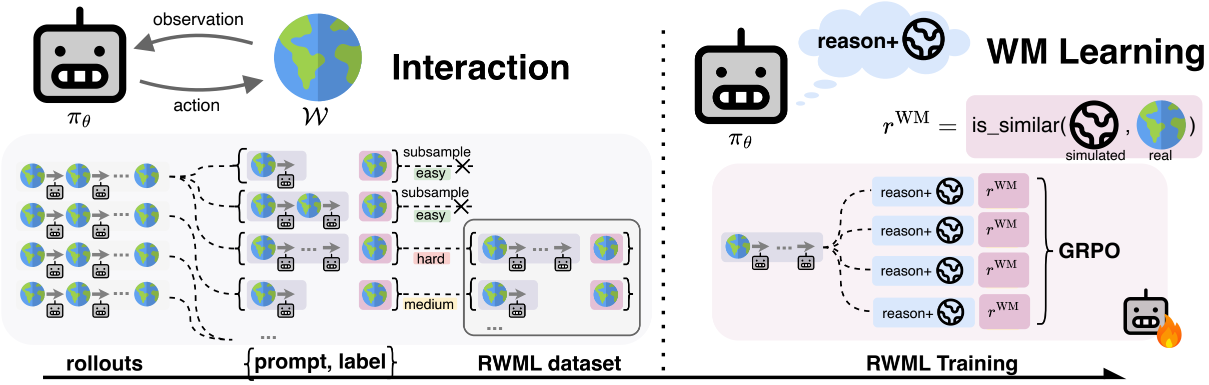


*Figure 2.* Overview of RWML. Given a target model $\pi_\theta$, we first collect training data for RWML by using $\pi_\theta$ to gather rollouts $(s_0, a_0, s_1, a_1, ...s_T)$ with the environment, and then convert these rollouts into $\langle s_{\leq t}, a_t, s_{t+1}\rangle$ triplets for all $t$, after subsampling "too easy" samples defined in Equation (1). We then train $\pi_\theta$ 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 $s_{t+1}$.

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 $\tau_d$ is a hyperparameter, and $d$ is implemented mainly using an off-the-shelf embedding model $E(\cdot)$ with cosine similarity (Karpukhin et al., 2020; Zhang et al., 2025b):

$$d(\hat{s}_{t+1}, s_{t+1}) = 1 - \cos(E(\hat{s}_{t+1}), E(s_{t+1})).$$

To optimize this reward, we use standard GRPO (Shao et al., 2024; DeepSeek-AI et al., 2025):

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

where $\rho_\theta = \pi_\theta(y|x)/\pi_{\theta_{\text{ref}}}(y|x)$ is the importance sampling ratio, $\beta$ 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.

To collect training data for RWML, we directly use the target model $\pi_\theta$ 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 (Snell et al., 2024; Sun et al., 2025) and subsample the portion of the dataset that are "too easy" to learn. Specifically, we first use SFT to train a separate LLM $\pi'_\theta$ capable of predicting $\hat{s}_{t+1}$ using 10% of the full dataset. Then, we use this $\pi'_\theta$ 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:

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

where $\tau_{\text{easy}}$ 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 (Shridhar et al., 2021) and $\tau^2$ Bench (Barres et al., 2025). 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. $\tau^2$ 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 $N$ trajecto-

ries per training task and then performs SFT training only on the trajectories that correctly solved the task (Touvron et al., 2023; Zelikman et al., 2022). Policy RL directly uses GRPO to train the base model $\pi_\theta$ to optimize for task-success reward using online rollouts (Feng et al., 2025c; Yu et al., 2025a).

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 consider Implicit World Modeling (IWM) and Self-Reflection (SR) from Zhang et al. (2025a); Yu et al. (2025c). 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 $\pi_\theta$. Then, either these data are converted to next-state prediction triplets $\langle s_{\le t}, a_t, s_{t+1}\rangle$ 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 using SFT (denoted as "Imitation Learning"). For more implementation details, please refer to Sections B.3 and C.4.

In addition to these training-based methods, we also evaluate REACT-style prompting (Yao et al., 2023) on closed-source LLMs such as GPT-5 (Singh et al., 2025) 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 (Feng et al., 2025c; Yu et al., 2025a; Zhang et al., 2025a), we train from Qwen2.5-7B-Instruct (Qwen et al., 2025) on ALFWorld for all methods. On $\tau^2$ Bench, we train from Qwen3-8B (Yang et al., 2025) 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 $\pi_\theta$ to rollout $N$ trajectories per training task with temperature $\tau = 1.0$, with $N = 3$ for ALFWorld and $N = 6$ for $\tau^2$ Bench. Then, we split all turns into triplets of $\langle s_{\le t}, a_t, s_{t+1}\rangle$ for all $t$, using 90% of the triplets for training and 10% for validation. Finally, we subsample "simple" training samples using a $\tau_{\text{easy}}$ that corresponds to ∼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 $\langle s_{\le t}, a_t, s_{t+1}\rangle$ 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 (Yu et al., 2025a) and allow a maximum of 30 steps per task. For $\tau^2$ Bench, due to cost concerns we train and evaluate using Qwen3-235B-A22B-Instruct (Yang et al., 2025) 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 C.3. All trainings are performed with B200 GPUs. For more training and hyperparameter details, please see Section B and Section C for ALFWorld and $\tau^2$ 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, strong LLMs, or task-success reward signals, RWML significantly improved agentic capability compared to the base model, advancing 19.6 and 7.9 points on ALFWorld and $\tau^2$ 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 $\tau^2$ Bench, our models achieve the second best overall score, 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 LLM-based agent performance.

### 3.3. RWML Forgets Less

In Table 3, we evaluate relative susceptibility of RL and SFT to catastrophic forgetting (Kirkpatrick et al., 2017; Luo et al., 2025c) in the context of world model learning. We evaluate our models trained on ALFWorld and $\tau^2$ Bench on (1) general knowledge benchmarks such as MMLU-Redux (Gema et al., 2025) and IFEval (Zhou et al., 2023); (2) math and STEM problems such as MATH-500 (Lightman et al., 2023), GSM8k (Cobbe et al., 2021), and GPQA-Diamond (Rein et al., 2023); and (3) coding tasks such as LiveCodeBench (Jain et al., 2024). 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 (Shenfeld et al., 2025; Chen et al., 2025a), 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.

*Table 1.* Performance on ALFWorld and $\tau^2$ 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 $\tau^2$ Bench.

| Method | ALFWorld | | | $\tau^2$ Bench | | | |
|---|---|---|---|---|---|---|---|
| | ID | OOD | AVG | Retail | Telecom | Airline | AVG |
| ReAct(Qwen2.5-7B) | 16.2±1.0 | 6.8±2.0 | 13.0±1.3 | 15.0±2.0 | 27.5±0.0 | 18.3±2.4 | 20.7±0.9 |
| ReAct(Qwen3-8B) | 40.9±1.5 | 31.3±2.6 | 37.7±1.8 | 37.7±4.9 | 31.2±4.2 | 21.6±5.2 | 31.9±2.9 |
| ReAct(Qwen3-235B*) | 38.0±0.4 | 32.3±2.7 | 36.1±0.7 | 50.6±2.7 | 48.8±3.8 | 51.3±5.5 | 50.0±3.4 |
| ReAct(GPT-4.1) | 42.5±1.0 | 47.4±0.7 | 44.1±0.5 | 55.8±2.4 | 41.7±4.3 | 48.3±2.4 | 48.7±4.5 |
| ReAct(GPT-5) | 51.6±1.3 | 44.8±0.7 | 49.3±0.9 | 55.8±7.2 | 65.0±5.4 | 55.0±4.1 | 59.3±0.5 |
| *Learning from task success reward* | | | | | | | |
| RFT | 34.4±3.8 | 34.4±3.4 | 34.4±2.6 | 43.3±3.1 | 33.3±3.1 | 13.3±2.4 | 33.3±1.7 |
| Policy RL | 82.1±3.6 | 79.2±2.0 | 81.0±1.6 | 40.8±1.2 | 39.2±1.2 | 30.0±8.2 | 38.0±1.6 |
| *Self-Supervised* | | | | | | | |
| WM SFT | 3.1±0.0 | 2.1±0.7 | 2.8±0.3 | 32.3±5.3 | 24.1±6.1 | 26.9±6.6 | 27.9±3.1 |
| RWML (ours) | 34.4±0.6 | 29.2±7.5 | 32.6±2.1 | 40.8±4.0 | 40.5±4.9 | 31.3±6.7 | 38.8±2.5 |
| *Self-Supervised + Policy RL* | | | | | | | |
| WM SFT + Policy RL | 76.2±3.4 | 82.3±0.7 | 80.4±1.5 | 40.8±4.2 | 45.0±5.4 | 30.0±7.1 | 40.3±3.9 |
| RWML + Policy RL (ours) | **86.7**±2.8 | **90.1**±0.7 | **87.9**±1.6 | **44.2**±2.1 | **45.8**±2.4 | **38.3**±2.4 | **43.7**±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. (2025a), 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*. Our models show competitive performance without using expert/strong LLM data.

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

### 3.4. Ablation Studies

In Table 4 we present an ablation study to investigate the contribution of different components in our RWML. Specifically, we consider: (1) replacing our embedding-based reward with LLM-as-a-judge (Zheng et al., 2023); (2) removing the data subsampling step which subsamples "too easy" samples, denoted as "w/o subsample"; (3) removing the RWML training entirely, denoted as "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 (Yang et al., 2025) as the judge model as it is a fast, strong, open-source LLM that can be hosted locally.

Results in Table 4 show that all components of our method are important in improving model performance. Additionally, we find that (1) weaker models such as Qwen2.5-7B are more susceptible to data quality/noisy reward functions; (2) LLM-as-a-judge is unreliable and can sometimes be hacked during training (see Section D for an example); and (3) subsampling "easy" training samples is beneficial to further improve model performance.

## 4. Discussion

### 4.1. Impact of RWML on Decision-Making

In this section, we provide some qualitative and quantitative analyses of model's decision-making behavior before and after RWML training. Qualitatively, in Figure 5 we find RWML-trained models produce more accurate and efficient decisions, utilizing its improved knowledge about the envi-

*Table 3.* Measuring forgetting after training on ALFWorld and $\tau^2$ 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 (ModelScope, 2024). Largest performance degradation ($\Delta$) is highlighted in dark red. Best viewed in color.

| | | ALFWorld | | | $\tau^2$ Bench | | |
|---|---|---|---|---|---|---|---|
| | | Qwen2.5-7B | +WM SFT | +RWML | Qwen3-8B | +WM SFT | +RWML |
| General | MMLU-Redux | 77.26 | 67.16$_{(\Delta\text{-}10.10)}$ | 74.88$_{(\Delta\text{-}2.38)}$ | 87.75 | 87.02$_{(\Delta\text{-}0.73)}$ | 87.42$_{(\Delta\text{-}0.33)}$ |
| | IFEval | 71.34 | 68.39$_{(\Delta\text{-}2.95)}$ | 69.32$_{(\Delta\text{-}2.02)}$ | 84.46 | 82.07$_{(\Delta\text{-}2.39)}$ | 83.36$_{(\Delta\text{-}1.10)}$ |
| Math & STEM | MATH-500 | 75.40 | 71.60$_{(\Delta\text{-}3.80)}$ | 75.40$_{(\Delta 0.00)}$ | 92.80 | 92.80$_{(\Delta 0.00)}$ | 92.80$_{(\Delta 0.00)}$ |
| | GSM8k | 91.66 | 90.45$_{(\Delta\text{-}1.21)}$ | 91.28$_{(\Delta\text{-}0.38)}$ | 96.13 | 95.53$_{(\Delta\text{-}0.60)}$ | 95.68$_{(\Delta\text{-}0.45)}$ |
| | GPQA-Diamond | 32.83 | 25.25$_{(\Delta\text{-}7.58)}$ | 28.79$_{(\Delta\text{-}4.05)}$ | 59.09 | 57.07$_{(\Delta\text{-}2.02)}$ | 58.08$_{(\Delta\text{-}1.01)}$ |
| Coding | LiveCodeBench | 19.23 | 15.38$_{(\Delta\text{-}3.85)}$ | 16.48$_{(\Delta\text{-}2.75)}$ | 43.41 | 41.21$_{(\Delta\text{-}2.20)}$ | 43.41$_{(\Delta 0.00)}$ |

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

| Method | ALFWorld | | | $\tau^2$ 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$_{\pm 2.9}$ |

ronment. 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 $\tau^2$ 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 $\tau^2$ 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

To understand the effectiveness of RWML, we also analyze how it reshapes model parameters during training. Following Zhu et al. (2025), we examine parameter-wise weight changes relative to the untrained base model, adopting the same definition and threshold $\eta = 10^{-3}$ to identify major point-wise updates:

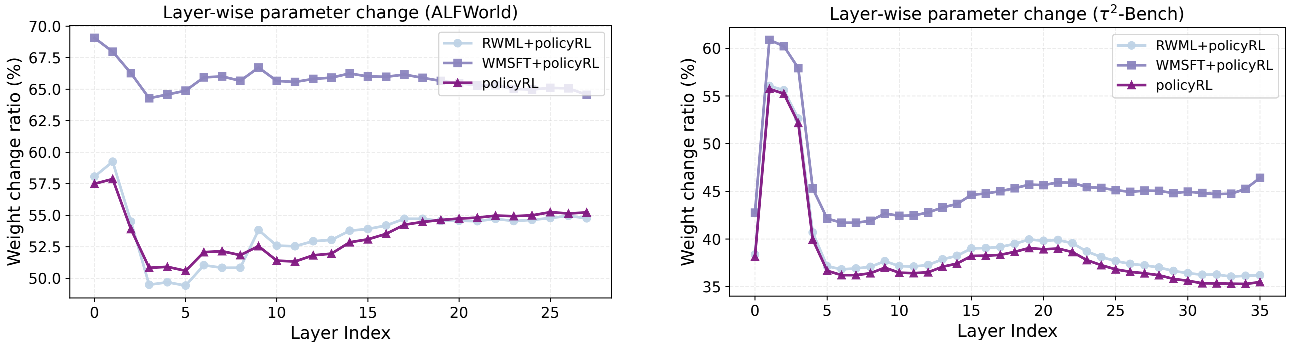
$$|\hat{w}_i - w_i| > \eta \cdot \max(|w_i|, |\hat{w}_i|),$$

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. Results for Qwen3-8B on $\tau^2$-Bench and Qwen2.5-7B-Instruct on ALFWorld are shown in Figure 3. Full results are in Section E. A consistent pattern emerges that RWML induces notably fewer parameter changes across layers compared to WM SFT, indicating that it encodes task-relevant information with a smaller and more targeted set of updates (also see Section E). This suggests that RWML learns in a more parameter-efficient and structurally conservative manner, avoiding widespread modifications to the pretrained representation space.

Importantly, this compact update behavior also help explain why RWML integrates well with subsequent policy learning, as shown in Figure 3. When followed by Policy RL, the resulting weight-change ratios remain remarkably close to those of Policy RL applied directly to the base model. In contrast, models initialized with 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



*(a)* ALFWorld

*(b)* $\tau^2$ 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.

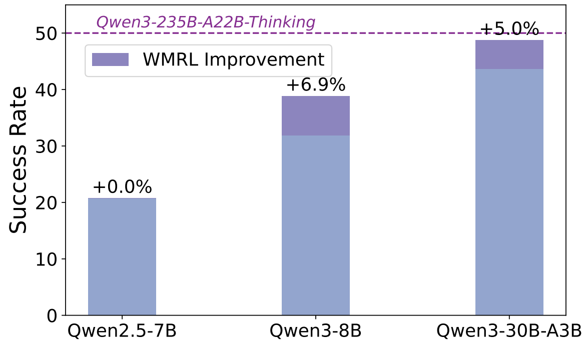


*Figure 4.* RWML training with different base models on $\tau^2$ Bench.

is consistent across both benchmarks and largely invariant to different transformer components, including attention (Q/K/V/O) and MLP projection layers (see Section E). 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 and consistent parameter updates, and may help explain the improved performance.

### 4.3. Impact of Base Model Capability

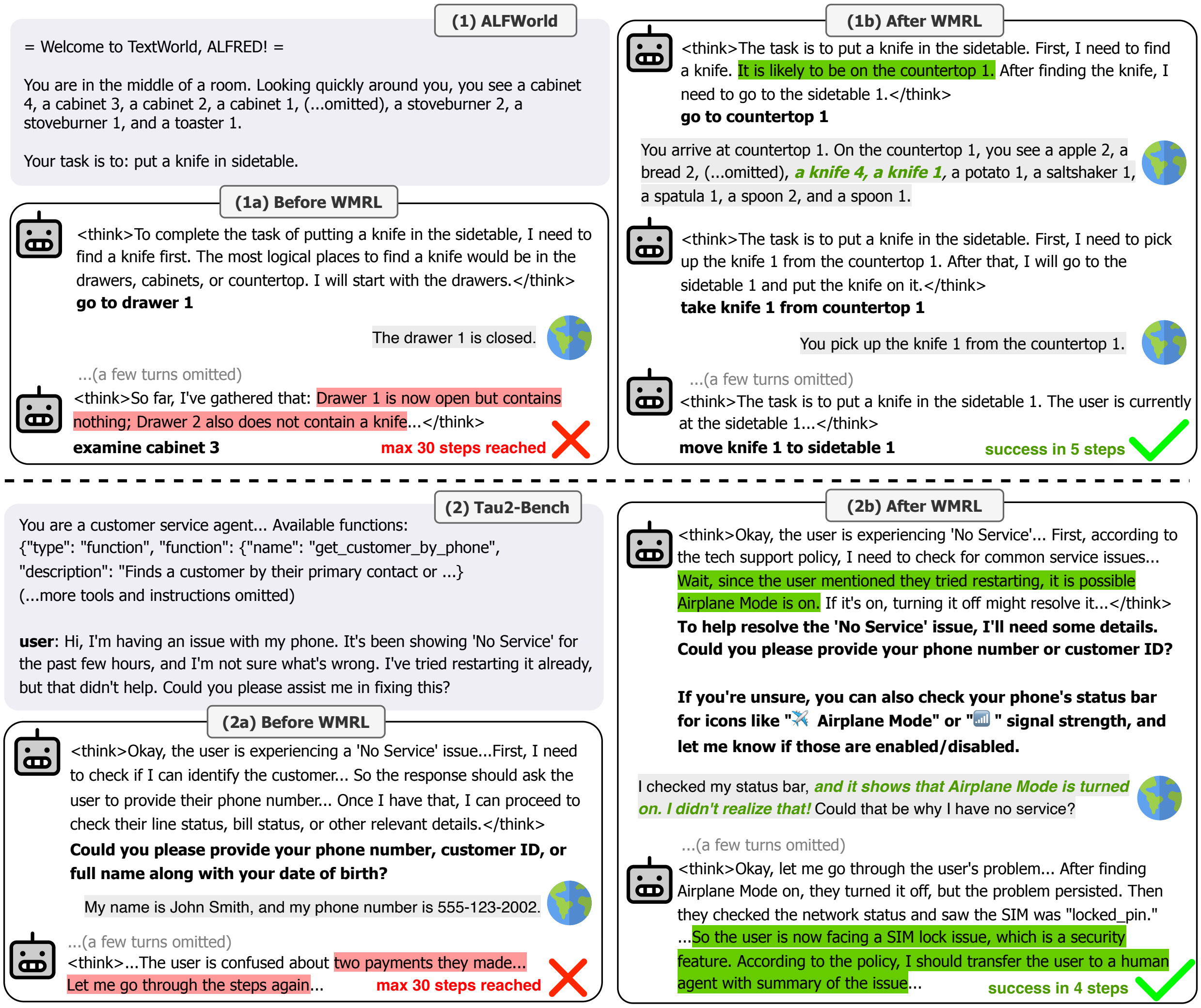
On the challenging $\tau^2$ bench, we find the ability to learn and transfer world model knowledge from RWML to decision-making is dependent on the capability of the base model. In Figure 4, we perform RWML training with three different base models: Qwen2.5-7B, Qwen3-8B, and Qwen3-30B-A3B[3]. We find that weaker models like Qwen2.5-7B struggle to transfer world knowledge to decision-making on the challenging $\tau^2$ 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. Related Work

**Training Decision-Making Agents** LLM-based agents (Yao et al., 2023; Shinn et al., 2023) has seen wide applications in domains such as interactive gaming (Wang et al., 2023; Feng et al., 2025c); software engineering (Jimenez et al., 2024; Yang et al., 2024); computer, phone, browser-use (Xie et al., 2024; Rawles et al., 2025; Zhou et al., 2024; Yu et al., 2025b), and more. Many early work on training language agents primarily rely on imitation learning (i.e., SFT), using either demonstrations from human experts (Deng et al., 2023; Chen et al., 2025b; Wang et al., 2025a) or trajectories synthesized from stronger LLMs often accompanied with a set of manually designed workflows/heuristics (Zeng et al., 2023; Chen et al., 2024; Su et al., 2025; Xu et al., 2025). While high-quality SFT data offers dense supervision signals, it is difficult to scale due to the high cost of collecting such demonstrations. Alternatively, recent efforts in RL bypasses the need for step-by-step demonstrations and instead directly learn from terminal rewards (i.e., task success) through trail and error. Recent work include Feng et al. (2025a); Tan et al. (2025); Luo et al. (2025a); Jin et al. (2025); Wang et al. (2025b), often powered by algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). However, designing task-success reward functions in complex environments still requires substantial human expertise (Chowdhury et al., 2024; Xie et al., 2024; Gou et al., 2025), 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.

---

[3] 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.



**(1) ALFWorld**

= 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.

**(1a) Before WMRL**

<think>To complete the task of putting a knife in the sidetable, I need to find a knife first. The most logical places to find a knife would be in the drawers, cabinets, or countertop. I will start with the drawers.</think>
**go to drawer 1**

The drawer 1 is closed.

...(a few turns omitted)

<think>So far, I've gathered that: Drawer 1 is now open but contains nothing; Drawer 2 also does not contain a knife...</think>
**examine cabinet 3**

**max 30 steps reached** ✗

**(1b) After WMRL**

<think>The task is to put a knife in the sidetable. First, I need to find a knife. It is likely to be on the countertop 1. After finding the knife, I need to go to the sidetable 1.</think>
**go to countertop 1**

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.

<think>The task is to put a knife in the sidetable. First, I need to pick up the knife 1 from the countertop 1. After that, I will go to the sidetable 1 and put the knife on it.</think>
**take knife 1 from countertop 1**

You pick up the knife 1 from the countertop 1.

...(a few turns omitted)

<think>The task is to put a knife in the sidetable 1. The user is currently at the sidetable 1...</think>
**move knife 1 to sidetable 1**

**success in 5 steps** ✓

**(2) Tau2-Bench**

You are a customer service agent... Available functions:
{"type": "function", "function": {"name": "get_customer_by_phone", "description": "Finds a customer by their primary contact or ...}
(...more tools and instructions omitted)

**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?

**(2a) Before WMRL**

<think>Okay, the user is experiencing a 'No Service' issue...First, I need to check if I can identify the customer... So the response should ask the user to provide their phone number... Once I have that, I can proceed to check their line status, bill status, or other relevant details.</think>
**Could you please provide your phone number, customer ID, or full name along with your date of birth?**

My name is John Smith, and my phone number is 555-123-2002.

...(a few turns omitted)

<think>...The user is confused about two payments they made... Let me go through the steps again...

**max 30 steps reached** ✗

**(2b) After WMRL**

<think>Okay, the user is experiencing 'No Service'... First, according to the tech support policy, I need to check for common service issues... Wait, since the user mentioned they tried restarting, it is possible Airplane Mode is on. If it's on, turning it off might resolve it...</think>
**To help resolve the 'No Service' issue, I'll need some details. Could you please provide your phone number or customer ID?**

**If you're unsure, you can also check your phone's status bar for icons like "✈ Airplane Mode" or "📶" signal strength, and let me know if those are enabled/disabled.**

I checked my status bar, ***and it shows that Airplane Mode is turned on. I didn't realize that!*** Could that be why I have no service?

...(a few turns omitted)

<think>Okay, let me go through the user's problem... After finding Airplane Mode on, they turned it off, but the problem persisted. Then they checked the network status and saw the SIM was "locked_pin." ...So the user is now facing a SIM lock issue, which is a security feature. According to the policy, I should transfer the user to a human agent with summary of the issue...

**success in 4 steps** ✓

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

**Training World Models** Beyond task-success rewards, real-world interaction data contains rich information that can be used to help decision-making. Early examples include Dyna algorithms (Sutton, 1991) which separately trains a world model to combine model-based with model-free learning for efficient policy training. Recent applications on LLM agents either train a *separate* world model to support inference-time algorithms such as MCTS (Hao et al., 2023; Wu et al., 2025; Chae et al., 2025; Gu et al., 2025), or jointly learn world models and policies within a single model to improve generalization (FAIR CodeGen team et al., 2025; Zhang et al., 2025a; Yu et al., 2025a;c; Feng et al., 2025b; Li et al., 2025; Qian et al., 2026). However, these approaches either require expensive training/inference of multiple models, or rely on additional annotations from experts/strong LLMs during world model learning. We propose RWML as a scalable, *self-supervised* method to improve the world knowledge and decision-making ability of a single model.

## 6. Conclusion

We propose RWML, a scalable, self-supervised method that enhances the environment understanding and decision-making ability of LLM-based agents prior to downstream RL with task-success reward. Without expert/strong LLM annotations or task-success signals, RWML trains the LLM as an action-conditioned world model by aligning the simulated next states with observed environment states in a pre-trained embedding space. We evaluate RWML on two long-horizon agent benchmarks, ALFWorld and $\tau^2$ Bench, and find significant performance gains while using only interaction data. When combined with task-success rewards in policy RL, our method outperforms direct policy RL on both benchmarks and matches training with expert data. We believe our work opens up new avenues for scalable, self-supervised training methods to further advance LLM-based agents in the era of agentic RL.

## 7. 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 (OpenAI, 2022; Inan et al., 2023; Luo et al., 2025b), and using isolated environments like sandboxes (AgentInfra Team, 2025; Pan et al., 2025), 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.

## References

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

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. $\tau^2$-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Bennett, M. *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.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Chae, H., Kim, N., iunn Ong, K. T., Gwak, M., Song, G., Kim, J., Kim, S., Lee, D., and Yeo, J. Web agents with world models: Learning and leveraging environment dynamics in web navigation, 2025. URL https://arxiv.org/abs/2410.13232.

Chen, H., Razin, N., Narasimhan, K., and Chen, D. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025a. URL https://arxiv.org/abs/2510.18874.

Chen, W., Cui, J., Hu, J., Qin, Y., Fang, J., Zhao, Y., Wang, C., Liu, J., Chen, G., Huo, Y., Yao, Y., Lin, Y., Liu, Z., and Sun, M. Guicourse: From general vision language models to versatile gui agents, 2025b. URL https://arxiv.org/abs/2406.11317.

Chen, Z., Liu, K., Wang, Q., Zhang, W., Liu, J., Lin, D., Chen, K., and Zhao, F. Agent-flan: Designing data and methods of effective agent tuning for large language models, 2024. URL https://arxiv.org/abs/2403.12881.

Chowdhury, N., Aung, J., Shern, C. J., Jaffe, O., Sherburn, D., Starace, G., Mays, E., Dias, R., Aljubeh, M., Glaese, M., Jimenez, C. E., Yang, J., Ho, L., Patwardhan, T., Liu, K., and Madry, A. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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

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

Daw, N. D., Niv, Y., and Dayan, P. 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.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web, 2023. URL https://arxiv.org/abs/2306.06070.

FAIR CodeGen team, Copet, J., Carbonneaux, Q., Cohen, G., Gehring, J., Kahn, J., Kossen, J., Kreuk, F., McMilin, E., Meyer, M., Wei, Y., Zhang, D., Zheng, K., Armengol-Estapé, J., Bashiri, P., Beck, M., Chambon, P., Charnalia, A., Cummins, C., Decugis, J., Fisches, Z. V., Fleuret, F., Gloeckle, F., Gu, A., Hassid, M., Haziza, D., Idrissi, B. Y., Keller, C., Kindi, R., Leather, H., Maimon, G., Markosyan, A., Massa, F., Mazaré, P.-E., Mella, V., Murray, N., Muzumdar, K., O'Hearn, P., Pagliardini, M., Pedchenko, D., Remez, T., Seeker, V., Selvi, M., Sultan, O., Wang, S., Wehrstedt, L., Yoran, O., Zhang, L., Cohen, T., Adi, Y., and Synnaeve, G. Cwm: An open-weights llm for research on code generation with world models, 2025. URL https://arxiv.org/abs/2510.02387.

Feng, J., Huang, S., Qu, X., Zhang, G., Qin, Y., Zhong, B., Jiang, C., Chi, J., and Zhong, W. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL https://arxiv.org/abs/2504.11536.

Feng, J., Zhang, Y., Zhang, C., Lu, Y., Liu, S., and Wang, M. Web world models, 2025b. URL https://arxiv.org/abs/2512.23676.

Feng, L., Xue, Z., Liu, T., and An, B. Group-in-group policy optimization for llm agent training, 2025c. URL https://arxiv.org/abs/2505.10978.

Gema, A. P., Leang, J. O. J., Hong, G., Devoto, A., Mancino, A. C. M., Saxena, R., He, X., Zhao, Y., Du, X., Madani, M. R. G., Barale, C., McHardy, R., Harris, J., Kaddour, J., van Krieken, E., and Minervini, P. Are we done with mmlu?, 2025. URL https://arxiv.org/abs/2406.04127.

Gou, B., Huang, Z., Ning, Y., Gu, Y., Lin, M., Qi, W., Kopanev, A., Yu, B., Gutiérrez, B. J., Shu, Y., Song, C. H., Wu, J., Chen, S., Moussa, H. N., Zhang, T., Xie, J., Li, Y., Xue, T., Liao, Z., Zhang, K., Zheng, B., Cai, Z., Rozgic, V., Ziyadi, M., Sun, H., and Su, Y. Mind2web 2: Evaluating agentic search with agent-as-a-judge, 2025. URL https://arxiv.org/abs/2506.21506.

Gu, Y., Zhang, K., Ning, Y., Zheng, B., Gou, B., Xue, T., Chang, C., Srivastava, S., Xie, Y., Qi, P., Sun, H., and Su, Y. Is your llm secretly a world model of the internet? model-based planning for web agents, 2025. URL https://arxiv.org/abs/2411.06559.

Hao, S., Gu, Y., Ma, H., Hong, J. J., Wang, Z., Wang, D. Z., and Hu, Z. Reasoning with language model is planning with world model, 2023. URL https://arxiv.org/abs/2305.14992.

Hu, Z. and Shu, T. Language models, agent models, and world models: The law for machine reasoning and planning, 2023. URL https://arxiv.org/abs/2312.05230.

Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., Mao, Y., Tontchev, M., Hu, Q., Fuller, B., Testuggine, D., and Khabsa, M. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. Dense passage retrieval for open-domain question answering, 2020. URL https://arxiv.org/abs/2004.04906.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. 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.

Lample, G. and Conneau, A. Cross-lingual language model pretraining, 2019. URL https://arxiv.org/abs/1901.07291.

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

Li, Y., Wang, H., Qiu, J., Yin, Z., Zhang, D., Qian, C., Li, Z., Ma, P., Chen, G., Ji, H., and Wang, M. From word to world: Can large language models be implicit text-based world models?, 2025. URL https://arxiv.org/abs/2512.18832.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013/`.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., and Tang, J. Agentbench: Evaluating llms as agents, 2025. URL `https://arxiv.org/abs/2308.03688`.

Luo, R., Wang, L., He, W., and Xia, X. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025a.

Luo, W., Dai, S., Liu, X., Banerjee, S., Sun, H., Chen, M., and Xiao, C. Agrail: A lifelong agent guardrail with effective and adaptive safety detection, 2025b. URL `https://arxiv.org/abs/2502.11448`.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025c. URL `https://arxiv.org/abs/2308.08747`.

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

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

OpenAI and et al. Openai o1 system card, 2024. URL `https://arxiv.org/abs/2412.16720`.

Pan, J., Wang, X., Neubig, G., Jaitly, N., Ji, H., Suhr, A., and Zhang, Y. 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.

Qian, C., Acikgoz, E. C., Li, B., Chen, X., Zhang, Y., He, B., Luo, Q., Hakkani-Tür, D., Tur, G., Li, Y., et al. Current agents fail to leverage world model as tool for foresight. *arXiv preprint arXiv:2601.03905*, 2026.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., Toyama, D., Berry, R., Tyamagundlu, D., Lillicrap, T., and Riva, O. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. URL `https://arxiv.org/abs/2405.14573`.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL `https://arxiv.org/abs/2311.12022`.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code, 2024. URL `https://arxiv.org/abs/2308.12950`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Shenfeld, I., Pari, J., and Agrawal, P. Rl's razor: Why online reinforcement learning forgets less, 2025. URL `https://arxiv.org/abs/2509.04259`.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning, 2023. URL `https://arxiv.org/abs/2303.11366`.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL `https://arxiv.org/abs/2010.03768`.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., Nathan, A., Luo, A., Helyar, A., Madry, A., Efremov, A., Spyra, A., Baker-Whitcomb, A., Beutel, A., Karpenko, A., Makelov, A., Neitz, A., Wei, A., Barr, A., Kirchmeyer, A., Ivanov, A., Christakis, A., Gillespie, A., Tam, A., Bennett, A., Wan, A., Huang, A., Sandjideh, A. M., Yang, A., Kumar, A., Saraiva, A., Vallone, A., Gheorghe, A., Garcia, A. G., Braunstein, A., Liu, A., Schmidt, A., Mereskin, A., and et al. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Su, H., Sun, R., Yoon, J., Yin, P., Yu, T., and Arik, S. O. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments, 2025. URL https://arxiv.org/abs/2501.10893.

Sun, Y., Cao, Y., Huang, P., Bai, H., Hajishirzi, H., Dziri, N., and Song, D. Rl grokking recipe: How does rl unlock and transfer new algorithms in llms?, 2025. URL https://arxiv.org/abs/2509.21016.

Sutton, R. S. 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.

Tan, S., Luo, M., Cai, C., Venkat, T., Montgomery, K., Hao, A., Wu, T., Balyan, A., Roongta, M., Wang, C., Li, L. E., Popa, R. A., and Stoica, I. 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.

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

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Wang, X., Wang, B., Lu, D., Yang, J., Xie, T., Wang, J., Deng, J., Guo, X., Xu, Y., Wu, C. H., Shen, Z., Li, Z., Li, R., Li, X., Chen, J., Zheng, B., Li, P., Lei, F., Cao, R., Fu, Y., Shin, D., Shin, M., Hu, J., Wang, Y., Chen, J., Ye, Y., Zhang, D., Du, D., Hu, H., Chen, H., Zhou, Z., Yao, H., Chen, Z., Gu, Q., Wang, Y., Wang, H., Yang, D., Zhong, V., Sung, F., Charles, Y., Yang, Z., and Yu, T. Opencua: Open foundations for computer-use agents, 2025a. URL https://arxiv.org/abs/2508.09123.

Wang, Z., Wang, K., Wang, Q., Zhang, P., Li, L., Yang, Z., Jin, X., Yu, K., Nguyen, M. N., Liu, L., Gottlieb, E., Lu, Y., Cho, K., Wu, J., Fei-Fei, L., Wang, L., Choi, Y., and Li, M. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025b. URL https://arxiv.org/abs/2504.20073.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Wu, J., Yin, S., Feng, N., and Long, M. Rlvr-world: Training world models with reinforcement learning, 2025. URL https://arxiv.org/abs/2505.13934.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., and Xiong, C. Aguvis: Unified pure vision agents for autonomous gui interaction, 2025. URL https://arxiv.org/abs/2412.04454.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Yu, X., Peng, B., Galley, M., Cheng, H., Wu, Q., Kulkarni, J., Nath, S., Yu, Z., and Gao, J. Dyna-mind: Learning to simulate from experience for better ai agents, 2025a. URL https://arxiv.org/abs/2510.09577.

Yu, X., Peng, B., Vajipey, V., Cheng, H., Galley, M., Gao, J., and Yu, Z. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning, 2025b. URL https://arxiv.org/abs/2410.02052.

Yu, X., Peng, B., Xu, R., Galley, M., Cheng, H., Nath, S., Gao, J., and Yu, Z. Dyna-think: Synergizing reasoning, acting, and world model simulation in ai agents, 2025c. URL https://arxiv.org/abs/2506.00320.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Zeng, A., Liu, M., Lu, R., Wang, B., Liu, X., Dong, Y., and Tang, J. Agenttuning: Enabling generalized agent abilities for llms, 2023. URL https://arxiv.org/abs/2310.12823.

Zhang, K., Chen, X., Liu, B., Xue, T., Liao, Z., Liu, Z., Wang, X., Ning, Y., Chen, Z., Fu, X., Xie, J., Sun, Y., Gou, B., Qi, Q., Meng, Z., Yang, J., Zhang, N., Li, X., Shah, A., Huynh, D., Li, H., Yang, Z., Cao, S., Jang, L., Zhou, S., Zhu, J., Sun, H., Weston, J., Su, Y., and Wu, Y. Agent learning via early experience, 2025a. URL https://arxiv.org/abs/2510.08558.

Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., and Zhou, J. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025b. URL https://arxiv.org/abs/2506.05176.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

Zhu, H., Zhang, Z., Huang, H., Su, D., Liu, Z., Zhao, J., Fedorov, I., Pirsiavash, H., Lee, J., Pan, D. Z., Wang, Z., Tian, Y., and Tai, K. S. 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. 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 (Zhang et al., 2025a; Yu et al., 2025c) | Required | ✗ | ✗ |
| SR (Zhang et al., 2025a; Yu et al., 2025c) | Required | Required | ✗ |
| Imitation Learning | Required | ✗ | ✗ |
| RFT | ✗ | ✗ | Required |
| RWML (ours) | ✗ | ✗ | ✗ |

# B. More Details on ALFWorld

ALFWorld (Shridhar et al., 2021) is a text-based, long-horizon agent environment designed to align with the embodied ALFRED benchmark (Shridhar et al., 2021). 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.

## B.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 $\tau^2$ Bench (see Section B.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 $r^{\text{WM}}$ during training, we using Qwen3-Embedding-8B (Zhang et al., 2025b) 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, 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 A4 and Table A5, 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.

## B.2. Policy RL Training Setup

For Policy RL, we mainly follow setups and prompts from Feng et al. (2025c); Yu et al. (2025a). 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.

*Table A2.* Hyperparameters introduced in RWML. *Value* represents value used in ALFWorld, $\tau^2$ Bench, respectively.

| Stage | Name | Value | Heuristics | Intuition |
|---|---|---|---|---|
| RWML Data | $\tau_d$ | 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 | $\tau_{\text{easy}}$ | 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 | $\tau_d$ | 0.2, 0.4 | Add 0.15 from $\tau_d$ 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. |

### B.3. Other Training Setup

For Imitation Learning, IWM, and SR, we use expert data from the official ALFWorld dataset, following Zhang et al. (2025a). For reflection data in SR, we follow Zhang et al. (2025a) and use a branching factor of 3 per expert action. Then, we use the prompt provided in Zhang et al. (2025a) 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. (2025c), we use a stronger LLM (i.e., GPT-4o) to generate reflections. After data collection, we follow Zhang et al. (2025a) 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.

## C. More Details on $\tau^2$ Bench

$\tau^2$ Bench (Barres et al., 2025) 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 $\tau^2$ 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 $\tau^2$ 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 $\tau^2$ Bench suitable for evaluating the tool/user understanding and modeling ability of an LLM-based agent.

### C.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 $\tau^2$ 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":

*Table A3.* Performance on $\tau^2$ 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$_{\pm2.0}$ | 30.0$_{\pm4.1}$ | 23.3$_{\pm4.7}$ | 33.7$_{\pm0.5}$ |
| ReAct(GPT-4.1) | 66.7$_{\pm3.1}$ | 50.0$_{\pm2.0}$ | 41.7$_{\pm0.0}$ | 55.0$_{\pm1.4}$ |
| ReAct(GPT-5) | 77.5$_{\pm7.4}$ | 97.5$_{\pm2.0}$ | 51.7$_{\pm2.4}$ | 80.3$_{\pm4.2}$ |
| *Learning from experts/strong LLMs* | | | | |
| Imitation Learning | 49.2$_{\pm4.5}$ | 50.0$_{\pm2.0}$ | 33.3$_{\pm3.3}$ | 46.3$_{\pm3.3}$ |
| SR | 52.5$_{\pm3.5}$ | 45.8$_{\pm6.5}$ | 43.3$_{\pm2.6}$ | 48.0$_{\pm2.5}$ |
| *Learning from task success reward* | | | | |
| Policy RL | 34.2$_{\pm3.1}$ | 45.0$_{\pm7.4}$ | 31.7$_{\pm5.3}$ | 38.0$_{\pm2.2}$ |
| *Self-supervised* | | | | |
| WM SFT | 40.8$_{\pm3.1}$ | 30.8$_{\pm1.2}$ | 30.0$_{\pm8.2}$ | 34.7$_{\pm3.3}$ |
| RWML (ours) | 43.3$_{\pm3.1}$ | 47.5$_{\pm2.0}$ | 45.0$_{\pm8.2}$ | 45.3$_{\pm3.1}$ |
| *Self-supervised + Policy RL* | | | | |
| RWML + Policy RL (ours) | 48.3$_{\pm4.7}$ | 41.7$_{\pm2.4}$ | 50.0$_{\pm0.0}$ | 46.0$_{\pm2.8}$ |

"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 A6 and Table A7. Then, we follow Section B.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 show in Table A2. In total, ~60% of the $s_{t+1}$ are tool-use responses and ~40% are user responses.

For $r^{\text{WM}}$ during training, we use Qwen3-Embedding-8B (Zhang et al., 2025b) similar to Section B.1. However, since tool-use responses are generally structured outputs, we find using rouge-score (Lin, 2004) is more effective at capturing these structures and any missing keys/values. Our final reward function in $\tau^2$ Bench is defined as:

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

with $\tau_d = 0.4$ as user responses are highly non-deterministic, and round(·, 0.2) is used to promote better training stability. 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.

### C.2. Policy RL Training Setup

We extend the codebase from Section B.2 to allow for multi-turn rollouts with $tau^2$ Bench. We use the official prompts from $tau^2$ 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.

### C.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 A3. 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.

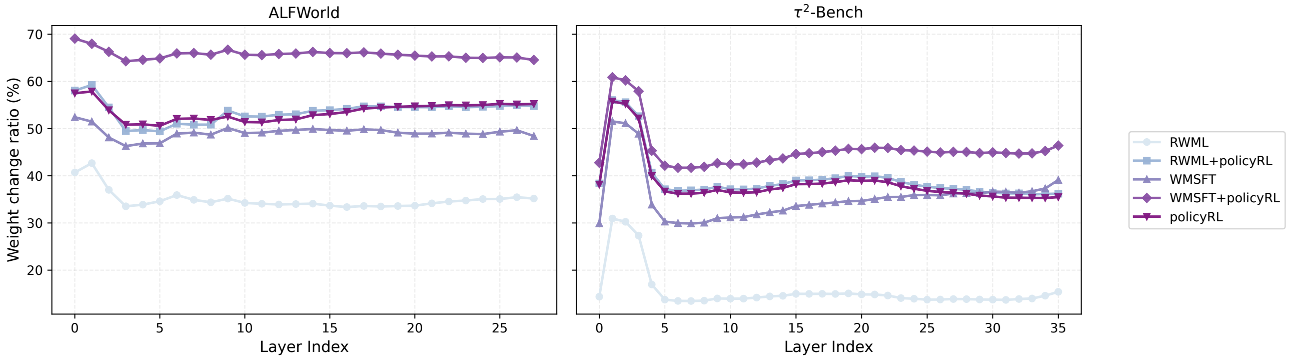
### C.4. Other Training Setup

Imitation Learning, IWM, and SR requires expert rollouts for training. However, since $\tau^2$ 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 C.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 B.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.

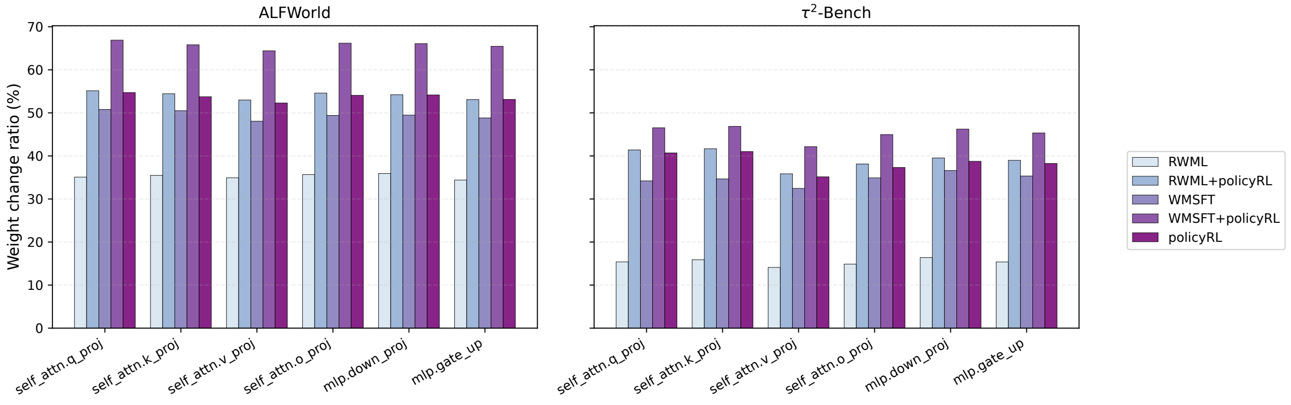
## D. More Details on Ablation Studies

For LLM-as-a-judge in Table 4, we present an example prompt and an example (hacked) response from ALFWorld in Table A8. We note that the model being trained is Qwen2.5-7B-Instruct, where as 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 does not show genuine understanding of the environment dynamics relevant to the task.

## E. More Details on Parameter Change Analysis



*(a)* Layer-wise parameter change ratios by transformer layer (ALFWorld and $\tau^2$-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 parameters, whereas WM SFT leads to noticeably broader parameter modifications.

*Table A4.* 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.

---

*Table A5.* Prompt for WM SFT in ALFWorld, given a current state $s_t$ and an action $a_t$. The label is the next state $s_{t+1}$.

---

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>`

---

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.

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 A6.* Prompt for RWML in $\tau^2$ 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 A7.* Prompt for WM SFT in $\tau^2$ Bench, given $s_{\leq t}$ and an action $a_t$. The label is the next state $s_{t+1}$.

---

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 A8.* 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.