REPLAYWORLD: EVOLVING AGENTIC WORLD MODEL FOR AGENT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based environment simulators increasingly synthesize trajectories for agent training, reducing reliance on costly real environments. However, prompted and fine-tuned simulators generalize poorly across environments, and their environment knowledge stays fixed after deployment. We introduce ReplayWorld, an agentic world model for eight environments in web navigation, CoWork and tool use. It searches for information, executes code and consults a repertoire of simulator skills that encode how environments respond to actions. Replay-driven skill evolution writes recurring discrepancies between predicted and recorded observations into these skills, so environment knowledge evolves without weight updates. Semantic balanced sampling (SBS) promotes replay diversity, learnability-prioritized filtering (LPF) focuses selection on learnable traces, and multi-criteria cycle arbitration (MCCA) selects the best repertoire through independent validation. A hierarchical skill router (HiSkill) routes relevant skills during trajectory synthesis. On ReplayBench, our benchmark of 2,080 real environment transitions scored on six criteria, ReplayWorld outperforms its prompted backbone by 4.2 points. The agent trained on ReplayWorld trajectories gains 12.0 points on average across eight agent benchmarks. Code is available at `https://anonymous.4open.science/r/replayworld-anon-2027-3ABB/`.

1 INTRODUCTION

LLM agents benefits from trajectories, but collecting them at scale from real environments is costly (Chen et al., 2025b; Zhang et al., 2025b; Wang et al., 2026). A growing line of work uses language models as environment simulators, predicting the observation that follows each action to synthesize trajectories for supervised fine-tuning or reinforcement learning (Li et al., 2025; Wang et al., 2025; Zuo et al., 2026; Xiao et al., 2026; Chen et al., 2025b). Such simulators enable large-scale, controllable data synthesis tailored to specific training needs. The effectiveness of this approach depends on how accurately the simulated responses reflect the behavior of the target environments.

Existing environment simulators based on LLMs broadly follow two approaches: prompting and model training. Prompting enables flexible environment simulation without additional model training (Li et al., 2025; Zhuo et al., 2025; Lee et al., 2026). However, reliance on environment priors and rules supplied by human experts makes adaptation dependent on human effort and limits generalization to new environments. Training world models on large datasets of interaction trajectories improves simulation across diverse environments (Zuo et al., 2026; Xiao et al., 2026), but encodes environment knowledge in fixed model parameters. Keeping this knowledge current as environments change typically requires new data and retraining. Figure 1 illustrates this problem: both baselines generate file contents for a read operation even though the requested file does not exist and the environment returns an error. In contrast, ReplayWorld uses skills learned from experience to model error conditions and code to track file state, enabling it to predict the error.

We introduce ReplayWorld, a general agentic world model for environment simulation spanning eight environments across three domains, including web navigation, CoWork, and tool use. Using 29,666 trajectories, we build its experience as simulator skills stored in editable text files. Whereas agent skills guide how an agent acts (Mi et al., 2026; Jiang et al., 2026; Yu et al., 2025; Zheng et al., 2025; Ye et al., 2026), simulator skills capture how an environment responds and are retrieved as needed to guide predictions. Code execution maintains persistent state in local files and performs computations for complex simulations, while web search provides access to current information.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

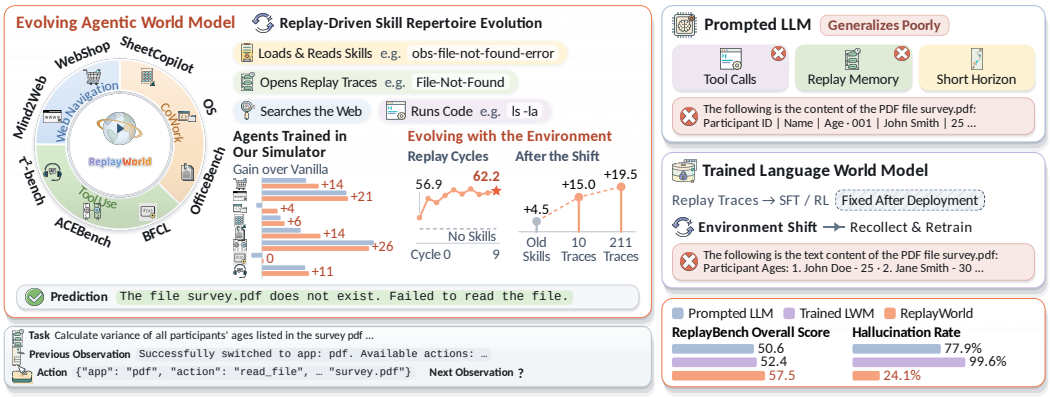

Figure 1: Left and bottom right: ReplayWorld is an evolving agentic world model that achieves consistent gains over baselines on ReplayBench and boosts the performance of agents trained in its environment simulator. Top right: when a PDF does not exist, a prompted LLM and a trained language world model wrongly invent the file's contents, while ReplayWorld predicts the error.

As environments change, ReplayWorld refines its skills by replaying newly collected trajectories, updating its experience without updating model parameters.

The key challenge is to generalize from individual prediction errors to reusable causal knowledge of environment behavior. We propose replay-driven skill evolution (Section 3.2) to build generalizable simulator skills during training and retrieve relevant ones at inference time. During training, semantic balanced sampling (SBS) promotes *diversity* by balancing replay across semantic clusters; learnability-prioritized filtering (LPF) focuses *selection* on trajectories with a single prediction error, so each error points to a specific gap; and multi-criteria cycle arbitration (MCCA) provides *validation*, scoring revised repertoires on trajectories withheld from revision and keeping the best. At inference time, a hierarchical skill router (HiSkill) retrieves relevant skills to guide trajectory synthesis.

To measure simulation accuracy, we introduce ReplayBench, which pairs 2,080 test traces from eight environments (260 per environment) with recorded observations. It rates simulators on six criteria: Format, Consistency, Factuality, LongHorizon (free runs on the simulator's own predictions), EnvReaction, and ActionPreserve. Factuality scores agree with three human annotators at Spearman 0.80. For agent training, we train Qwen3-8B and Qwen3-14B on simulated trajectories, test them in the eight real environments, and assess ReplayWorld as a reinforcement learning environment.

ReplayWorld-397B scores 59.4 on ReplayBench, above the same model prompted directly (55.2) and Qwen-AgentWorld-35B-A3B (52.4), with gains concentrated in EnvReaction, Factuality, and LongHorizon. In supervised training, its trajectories improve Qwen3-8B and Qwen3-14B by 12.0 and 10.0 points across eight benchmarks, beating the prompted simulator's trajectories at both scales. As an RL environment, ReplayWorld answers 24.1% of illegal actions with a normal observation, against 77.9% for the prompted simulator and 99.6% for EnvACE-8B (Xu et al., 2026b). When the operating system behind OS changes from Ubuntu to Alpine, one replay cycle on the new trajectories lifts the repertoire's advantage from 4.5 to 19.5 points without weight updates.

Our contributions are threefold: ❶ We introduce ReplayWorld, a general agentic world model for synthesizing agent training trajectories, which adapts to changing environments through evolving experience, code execution for state tracking, and web search for current information, without updating model parameters; ❷ we introduce ReplayBench, comprising 2,080 test traces across eight environments and six evaluation dimensions, and validate its automated evaluation against human judgments; ❸ we demonstrate that ReplayWorld outperforms both prompted LLMs and various trained language world models on ReplayBench for observation prediction, while also boosting agent performance after training on our environment simulator.

2 Related Work

LLM-based environment simulation. A language model can simulate an environment by predicting the observation that follows each action (Li et al., 2026; Chu et al., 2026). Prompted simulators

draw this knowledge from pre-training and a prompt (Li et al., 2025; Lee et al., 2026), and trained ones from interaction data (Zuo et al., 2026; Xiao et al., 2026; Chen et al., 2025b). Either way, a behavior absent from both becomes a plausible guess (Liu et al., 2026), and a changed environment costs new data and a training run unless the model revises its context online (Zhang et al., 2026; Ren et al., 2026) or its prompt templates before training (Zuo et al., 2026).

Executable environment synthesis. The other route builds the environment as code: tools, databases and state transitions (Wang et al., 2026; Song et al., 2026; Xu et al., 2026a), or whole collections of environments (Dong et al., 2026; Zhang et al., 2025a). Code gives deterministic transitions and checkable rewards and avoids the hallucinations of LLM simulators (Song et al., 2026; Tu et al., 2026), but every environment must be implemented before an agent can train in it (Zuo et al., 2026).

3 REPLAYWORLD: AN AGENTIC WORLD MODEL WITH SKILL EVOLUTION

ReplayWorld stores its knowledge as simulator skills, short text rules on how the environment responds, and revises them by replaying logged trajectories against the recorded responses, a form of what Chu et al. (2026) call evidence-driven model revision. Because the knowledge lives in text rather than in weights, it is consulted at prediction time: ReplayWorld runs as an agent, and each prediction is an agentic turn, a short loop of replies with tool calls that ends in the predicted observation (Figure 2). Sections 3.1 to 3.4 state the problem, the replay loop that evolves the skills, the agentic simulator that uses them, and the training of agents inside it.

3.1 PROBLEM FORMULATION

Environment and world model. We consider an agent solving a task q in an environment E. At step t the agent has observed o_t, takes the action a_t and carries the history $h_t = (q, o_0, a_0, \ldots, o_t)$; a logged trajectory $\tau = (q, o_0, a_0, \ldots, o_{T_\tau})$ with $T_\tau \geq 1$ is what we call a trace. Each test trace of ReplayBench isolates one step of such a trajectory, namely the history h_t, the action a_t and the observation o_{t+1} that the environment returned. To compare a simulator with the environment it stands in for, we write P_E for the transition distribution of E and $\hat{P}_S$ for that of a world model reading a skill repertoire S, and mark model-generated observations and histories $\hat{h}_t$:

$$o_{t+1} \sim P_E(\cdot \mid h_t, a_t), \qquad \hat{o}_{t+1} \sim \hat{P}_S(\cdot \mid h_t, a_t). \tag{1}$$

Skill Repertoire The model learn the repertoire S of skills and keep the weights θ of its language model p_θ fixed. We refer to the form of an agent skill (Mi et al., 2026; Huang et al., 2026): an identifier and an applicability condition, listed in the skill index, and a body opened when needed. The agentic world model will load the read a skill and the body of the skill states the observation format, the state carried over and the failure behavior, as in an OS skill that keeps a deleted file missing (Appendix F.2). Besides the skills, the benchmark supplies its interface, the action format and the parser that its agents use, and a format exemplar α_E holds one real observation per response type (Appendix D.2).

3.2 REPLAY-DRIVEN SKILL EVOLUTION

Having reduced learning to the repertoire, the model learns by editing individual skills, so that each edit names the rule it corrects and where that rule applies. Replay-driven skill evolution finds these edits in the discrepancies that recur between replayed predictions and recorded observations. Replaying a step of a trace means predicting its logged observation o_{t+1} from the logged history h_t and action a_t in an agentic turn (Section 3.3), with nothing re-executed in E. The logged traces used for replay form the replay corpus, which we split into replay traces $\mathcal{D}_{\mathrm{re}}$, from which revisions are written, and arbitration traces $\mathcal{D}_{\mathrm{arb}}$, used only to score repertoires; ReplayBench is disjoint from this corpus at the trajectory level (Appendix C.1). Starting from an initial repertoire S_0 of one to twelve skills per environment (Appendix B.1), cycle $k = 1, 2, \ldots$ (Algorithm 1) turns S_{k-1} into the alternative S_k, the repertoire that the cycle proposes, and multi-criteria cycle arbitration (MCCA) keeps the best alternative so far as the winner (Agrawal et al., 2025). In optimization terms, the repertoire is the simulator's parameter, each cycle a step estimated from one batch, and MCCA checkpoint selection with early stopping.

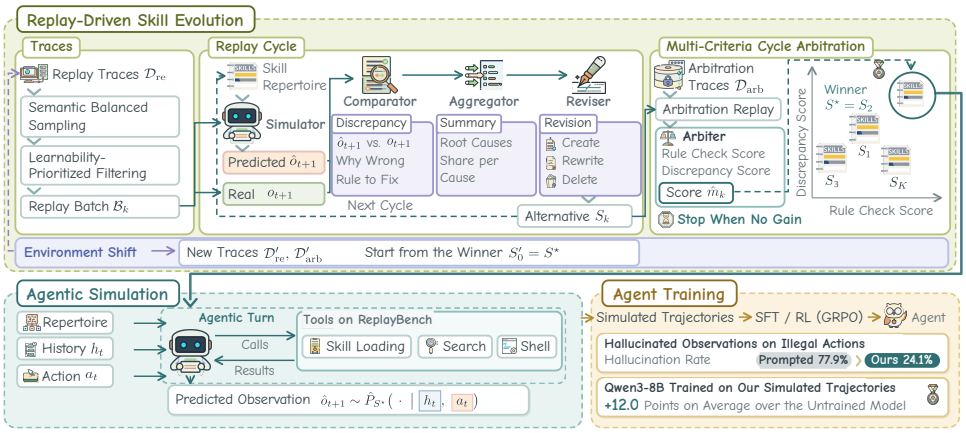

Figure 2: ReplayWorld overview. Top: each replay cycle turns recurring discrepancies into the next alternative, arbitration keeps the winner $S^\star$, and an environment shift restarts from it. Bottom left: to predict one observation, the simulator runs as an agent over its repertoire, opening skills and calling tools. Bottom right: the policy trains on trajectories the simulator generates.

Semantic Balanced Sampling. A uniform draw of replay traces would over-represent common task families and spend replay on predictions the simulator already gets right, so we select each batch in two stages (Figure 3). Semantic balanced sampling (SBS) first draws a pre-sample $\mathcal{P}_k$ of at most n_{pre} replay traces not yet batched; as in the data curation of Siméoni et al. (2025), each draw picks a k-means cluster ℓ of query embeddings, which holds n_ℓ traces, with probability p_ℓ and then one trace of that cluster. Learnability-prioritized filtering (LPF) then keeps up to n_{sl} traces of $\mathcal{P}_k$ with the highest learnability g_k, a product of three percentile ranks taken within $\mathcal{P}_k$, namely ρ_k^r of the recorded reward and $\rho_k^{\max}$ and ρ_k^{mean} of the largest and the mean replay error over a trace's sampled steps (Appendix B.1). Finally, SBS draws the batch $\mathcal{B}_k$ of at most B traces from those that LPF keeps, so that no single task family fills the batch. Formally,

$$p_\ell \propto \sqrt{n_\ell}, \qquad g_k = \rho_k^r \, \rho_k^{\max} \left(1 - \rho_k^{\text{mean}}\right), \\ e_{\tau,t} = 1 - \cos\left(\text{emb}(\hat{o}_{t+1}), \text{emb}(o_{t+1})\right), \tag{2}$$

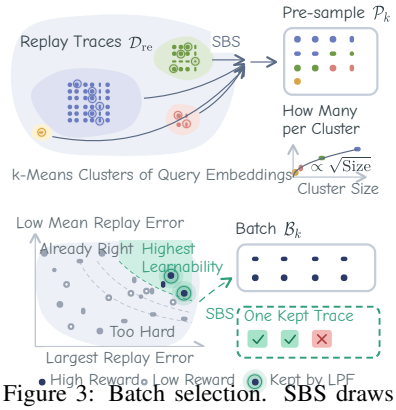

Figure 3: Batch selection. SBS draws traces cluster by cluster, so rare task families are not crowded out; LPF keeps the traces worth replaying, rewarded ones the simulator gets right except at one step, and SBS then draws the batch from them

where $e_{\tau,t}$ is the replay error at a sampled step, computed with an observation embedding emb. High learnability thus marks rewarded traces that the simulator predicts well except at one step (Wang et al., 2025). The prediction $\hat{o}_{t+1} \sim \hat{P}^{(1)}_{S_{k-1}}(\cdot \mid h_t, a_t)$ in this error comes from the single-reply instance, which uses no tools (Appendix B.1).

Learnability-prioritized filtering. Because a discrepancy seen on one trace can belong to that trace alone, we replay the whole batch, one uniformly drawn step per trace, so that the discrepancies come from distinct interactions. To keep any stronger model out of the repertoire, we let three roles of the same model p_θ, each with its own prompt, turn the batch into edits (Zhang et al., 2025c). The Comparator C_θ reads the logged history h_t, the action a_t, the prediction and the recorded observation, and writes a discrepancy δ_τ: what differs, which behavior of E was missed, and a rule that covers that behavior. The Aggregator A_θ groups the discrepancies by root cause into the discrepancy summary F_k, which lists each cause with the share of discrepancies behind it. For the causes whose share passes a frequency threshold, the Reviser R_θ writes a revision $\mathcal{U}_k$, a set of create, rewrite and delete edits on distinct skills. The update operator $\oplus$ applies a revision deterministically (Mi et al., 2026): $S \oplus \mathcal{U}$ removes every skill that an edit of $\mathcal{U}$ targets and adds the skill of every create and rewrite edit. Because separate cycles can write the same rule twice, a periodic consolidation merges skills whose wording overlaps above a threshold. With τ ranging over the traces of $\mathcal{B}_k$ whose prediction $\hat{o}_{t+1} \sim \hat{P}_{S_{k-1}}(\cdot \mid h_t, a_t)$ parses and each role written as the distribution of its output under p_θ, cycle

k computes

$$\begin{aligned} \delta_\tau &\sim \mathsf{C}_\theta\big(\cdot \mid h_t, a_t, \hat{o}_{t+1}, o_{t+1}\big), & F_k &\sim \mathsf{A}_\theta\big(\cdot \mid (\delta_\tau)_\tau\big), \\ \mathcal{U}_k &\sim \mathsf{R}_\theta\big(\cdot \mid S_{k-1}, F_k, \mathcal{B}_k\big), & S_k &\leftarrow S_{k-1} \oplus \mathcal{U}_k, \end{aligned} \tag{3}$$

then consolidates S_k when due. Discrepancies therefore reach the repertoire only through the summary F_k.

Multi-criteria cycle arbitration. An edit written from one batch can change predictions wherever its skill is opened, so MCCA scores whole repertoires, each on a fresh sample of arbitration traces. The Arbiter, a fixed scoring rule, averages two criteria into the arbitration score $\hat{m}_k$, a rule check score $\hat{m}_k^{\text{rule}}$ from benchmark-specific checks against the recorded observation and a discrepancy score $\hat{m}_k^{\text{sum}}$ from the Comparator's discrepancies and the Aggregator's summary of them. An arbitration pass that returns no score counts as $-\infty$, and $\hat{m}_0 = -\infty$ stands for the unscored S_0. Arbitration also decides when to stop: we count in η_k the consecutive cycles without a strict improvement of the best score so far, $\hat{m}^\star$, and the loop stops when η_k reaches the patience κ or at the cycle cap $K_{\max}$, returning the winner $S^\star = S_{k^\star}$ as the selected repertoire, with K the number of cycles run:

Algorithm 1: Replay-driven skill evolution.

Input : Replay traces $\mathcal{D}_{\text{re}}$; Arbitration traces $\mathcal{D}_{\text{arb}}$; Initial repertoire S_0; Model p_θ in roles $\mathsf{C}_\theta, \mathsf{A}_\theta, \mathsf{R}_\theta$; Sizes $n_{\text{pre}}, n_{\text{sl}}, B$; Patience κ; Cycle cap $K_{\max}$
Output : Selected repertoire $S^\star$

$S^\star \leftarrow S_0$; $\hat{m}^\star \leftarrow -\infty$; $\eta \leftarrow 0$
for $k \leftarrow 1$ **to** $K_{\max}$ **do**
  // Batch Selection (Eq. 2); Cold-Start Ranking at $k = 1$
  $\mathcal{P}_k \sim \text{SBS}\big(\mathcal{D}_{\text{re}} \setminus \bigcup_{i<k} \mathcal{B}_i,\ n_{\text{pre}}\big)$
  $\mathcal{B}_k \sim \text{SBS}\big(\text{Top}_{n_{\text{sl}}}(\mathcal{P}_k;\ g_k),\ B\big)$
  // Replay and Revision (Eqs. 5 and 3)
  foreach $\tau \in \mathcal{B}_k$ **do**
    $t \sim \text{Unif}\{0, \ldots, T_\tau - 1\}$;
    $\hat{o}_{t+1} \sim \hat{P}_{S_{k-1}}(\cdot \mid h_t, a_t)$
    if $\hat{o}_{t+1}$ *parses* **then**
      $\delta_\tau \sim \mathsf{C}_\theta(\cdot \mid h_t, a_t, \hat{o}_{t+1}, o_{t+1})$
  // $F_k = \emptyset$ When No Prediction Parses
  $F_k \sim \mathsf{A}_\theta(\cdot \mid (\delta_\tau)_\tau)$; $\mathcal{U}_k \sim \mathsf{R}_\theta(\cdot \mid S_{k-1},\ F_k,\ \mathcal{B}_k)$
  $S_k \leftarrow S_{k-1} \oplus \mathcal{U}_k$, Consolidated when due
  // MCCA: Arbitration and Early Stopping (Eq. 4)
  $\hat{m}_k \leftarrow$ Score of S_k on a fresh sample of $\mathcal{D}_{\text{arb}}$
  if $\hat{m}_k > \hat{m}^\star$ **then** $S^\star \leftarrow S_k$; $\hat{m}^\star \leftarrow \hat{m}_k$; $\eta \leftarrow 0$
  else $\eta \leftarrow \eta + 1$; **if** $\eta = \kappa$ **then break**
return $S^\star$

$$\hat{m}_k = \tfrac{1}{2}\big(\hat{m}_k^{\text{rule}} + \hat{m}_k^{\text{sum}}\big), \quad k^\star = \min \arg\max_{0 \le k \le K} \hat{m}_k, \quad K = \min\big\{K_{\max},\ \min\{k : \eta_k = \kappa\}\big\}. \tag{4}$$

The inner minimum is $K_{\max}$ when the patience is never reached, and ties go to the earlier cycle. The Reviser never sees these scores, which only select the returned repertoire and stop the loop. When an environment shift from E to a new environment E', we set $S'_0 = S^\star$ and run the same cycles on the replay and arbitration traces $\mathcal{D}'_{\text{re}}$ and $\mathcal{D}'_{\text{arb}}$ of E', so that $S'_k = S'_{k-1} \oplus \mathcal{U}'_k$ and Equation 4 selects $S'^\star$.

3.3 Agentic Simulation with the Evolved Repertoire

Prediction as an agentic turn. Since the agentic world model has to consult its skills while it writes the next observation, we formalize each prediction by $\hat{P}_S$ as an agentic turn in a harness, the program that runs p_θ as an agent, executes its tool calls with a deterministic tool executor T and parses its replies. A turn starts from an initial context $c_1 = (h_t, a_t, S, \alpha_E)$, with S rendered as the harness configuration prescribes and α_E present when supplied. The harness appends each reply and its tool results to the context until a reply calls no tool, and it disables tools in the M-th reply, so a turn has at most M replies. The prediction is the last reply y_J, and $\hat{P}_S$ is the distribution of y_J that the turn induces:

$$\begin{aligned} &y_j \sim p_\theta(\cdot \mid c_j), \qquad c_{j+1} = \big(c_j,\ y_j,\ \mathsf{T}(y_j)\big), \\ &J = \min\{\, j \le M : y_j \text{ calls no tool} \,\}, \\ &\hat{P}_S(\hat{o}_{t+1} \mid h_t, a_t) = \textstyle\sum_{y_{1:J}} \mathbf{1}[\, y_J = \hat{o}_{t+1} \,] \prod_{j=1}^{J} p_\theta(y_j \mid c_j). \end{aligned} \tag{5}$$

The prediction therefore conditions on every tool result.

Skill access. To keep the context short, on ReplayBench we render S by progressive disclosure: the context shows only the skill index, and a body enters the context when a tool opens it, as do a skill's reference traces, the logged traces attached to it. The unified repertoire can thus hold the selected skills of all eight environments: a turn sees their index and opens bodies as needed.

3.4 Training the Policy in the Environment Simulator

With $S^\star$ selected, a policy π_ϕ, a smaller language model with parameters ϕ, learns from trajectories that the simulator generates in place of E. For environment synthesis we switch to the injected

instance $\hat{P}_S^{\text{inj}}$, which places every skill body in the context and replaces the tools with a format-example lookup We also introduce HiSkill, a hierarchical skill router, then points the simulator to the rules a task needs. With its observation from ϱ, a simulated trajectory has probability under a policy π

$$\hat{\mathbb{P}}_\pi(\tau) = \varrho(q, o_0) \prod_{t=0}^{T_\tau - 1} \pi\big(a_t \mid \hat{h}_t\big)\, \hat{P}_{S^\star}^{\text{inj}}\big(\hat{o}_{t+1} \mid \hat{h}_t, a_t\big), \tag{6}$$

Here π is a behavior policy μ during synthesis and π_ϕ during reinforcement learning. For agentic supervised fine-tuning (Jiao et al., 2026), we train π_ϕ on the pool $\mathcal{D}_{\text{sim}}$ that the filters of Appendix B.2 select from trajectories of μ, the simulator's language model p_θ acting as the agent without simulator skills and reading q and the last w steps of $\hat{h}_t$. Because the simulator wrote the observations, the loss covers only the action tokens, $\mathcal{L}_{\text{SFT}}(\phi) = -\frac{1}{N_{\text{tok}}} \sum_{\tau \in \mathcal{D}_{\text{sim}}} \sum_{t,j} \log \pi_\phi(a_{t,j} \mid \hat{h}_t, a_{t,<j})$, where $a_{t,j}$ is the j-th token of a_t and $N_{\text{tok}} > 0$ counts the action tokens in $\mathcal{D}_{\text{sim}}$. The simulator can also replace E in agentic reinforcement learning with GRPO (Shao et al., 2024): a language-model judge ψ reads the final state the simulator wrote and returns the binary outcome reward $r_\psi(\tau) \in \{0, 1\}$, and the policy maximizes the expected reward $\mathbb{E}_{\tau \sim \hat{\mathbb{P}}_{\pi_\phi}}[r_\psi(\tau)]$ (Section 4.3; the GRPO objective is in Appendix B.3). In both cases the policy trains in $\hat{P}_{S^\star}^{\text{inj}}$, the environment that the selected repertoire defines.

4 Experiments

4.1 Experimental Setup

Environments. Eight real environments span three application areas: web navigation (WebShop (Yao et al., 2022), Mind2Web (Deng et al., 2023)), CoWork (OS (Liu et al., 2023), SheetCopilot (Li et al., 2023), OfficeBench (Wang et al., 2024)) and tool use (ACEBench (Chen et al., 2025a), BFCL (Patil et al., 2023), τ^2-bench (Barres et al., 2026)). The replay corpus shares no trajectory with the ReplayBench test traces (Appendix C.1).

World models. Each world model predicts the next observation from the logged history h_t and action a_t of a test trace at temperature 0. We compare models prompted directly, trained language world models (Qwen-AgentWorld-35B-A3B (Zuo et al., 2026), WebWorld-14B (Xiao et al., 2026), EnvACE-Qwen3-8B (Xu et al., 2026b), EnvScaler-Qwen3-8B (Song et al., 2026)) and ReplayWorld on Qwen3.5-397B-A17B and Qwen3.6-27B. LLM-as-WM represents prompted LLMs.

Policies. We fine-tune Qwen3-8B and Qwen3-14B (Yang et al., 2025) on trajectories from each simulator and score them in the real environments with each benchmark's own metric; Vanilla is the untrained model (Appendix B.3).

4.2 Evaluating World Models on ReplayBench

4.2.1 Benchmark Construction and Evaluation Protocol

ReplayBench test models' capabilities of observation prediction. Its 2,080 test traces, 260 per environment, each pair a logged history and the agent's next action with the observation the environment returned (Figure 4). A LLM judge that reads the recorded observation rates Format, Consistency, Factuality and LongHorizon from 1 to 5, rescaled to 0 to 100 as $100\,(r-1)/4$; EnvReaction, whether the predicted response type matches the recorded one, and ActionPreserve, whether an agent keeps its next action after reading the prediction, are rates on the same scale. We choose GPT-5.6-terra as main judge since it takes no part in synthesis and agrees with human annotators at Spearman 0.80 on Factuality. ReplayBench curation and metric design details are in Appendix C.1 and Appendix D.1, respectively.

4.2.2 Main Results

ReplayWorld-397B raises Overall from 55.2 for LLM-as-WM to 59.4. The gain lies in EnvReaction, Factuality and LongHorizon, while Format and Consistency move by less than a point: skills written from what the environment returned change the kind and content of a prediction, not its format. On LongHorizon, where the simulator consumes its own predictions, ReplayWorld-27B scores highest: it rereads its skill index at every step, whereas a model predicting from its weights alone carries its first error forward.

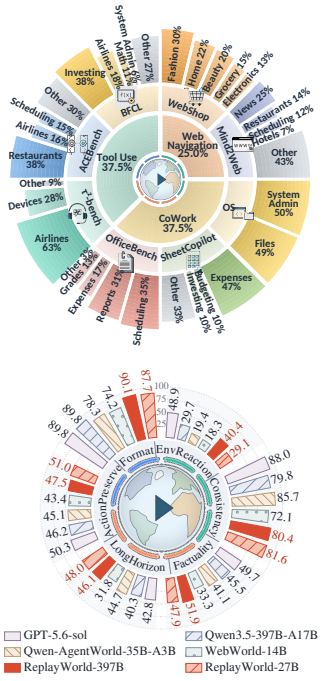

Figure 4: Overview of ReplayBench: environments and task sub-sectors (top); per-criterion scores (bottom).

Table 1: Evaluation results on ReplayBench. Overall: mean of the six criteria (Table 12). **Bold** and underline: best and second-best per column, above and below the double line.

	Web Navigation		CoWork		Tool Use		Overall↑
	Judge	Rule	Judge	Rule	Judge	Rule	
Proprietary Models							
Claude Opus 5	62.3	46.8	68.1	53.0	**73.5**	58.5	**63.7**
GPT-6-astra	59.8	43.8	**71.4**	**55.6**	70.3	**62.3**	63.6
Grok 4.6	60.8	**47.2**	68.5	54.8	70.6	49.3	61.7
GPT-5.6-sol	61.3	41.9	68.7	54.1	70.8	50.2	61.6
Claude Sonnet 5	59.6	39.3	67.5	51.9	70.8	49.5	60.5
Claude Opus 4.6	**63.7**	36.2	65.2	51.1	69.9	50.6	60.1
Open-Source Foundation Models (>100B)							
Qwen3.5-397B-A17B	59.4	26.7	63.9	42.5	66.8	41.0	55.2
GLM-5	58.5	26.1	65.1	41.4	66.8	40.3	55.0
Kimi K2.5	58.6	25.7	62.0	40.7	67.8	43.8	55.0
DeepSeek-V3.2	58.8	24.7	62.7	43.9	65.7	42.2	54.7
Llama 4 Maverick	57.2	23.5	59.0	32.9	63.1	38.4	50.9
Open-Source Models (<50B)							
Qwen3.6-27B	62.6	24.4	64.0	45.6	69.1	43.2	56.9
Qwen3-14B	56.8	18.8	59.0	39.6	65.2	35.1	51.5
Qwen3.5-35B-A3B	51.8	18.4	57.6	39.0	65.8	37.7	50.6
Qwen3-8B	55.6	18.1	53.8	33.7	63.9	37.8	49.2
Qwen3.5-9B	49.7	15.6	56.4	32.8	65.2	41.1	49.2
Language World Models							
Qwen-AgentWorld-35B-A3B	59.8	18.5	59.3	32.8	67.1	40.8	52.4
EnvScaler-Qwen3-8B	55.5	20.2	54.9	33.5	64.2	36.2	49.5
EnvACE-Qwen3-8B	56.3	18.5	52.2	30.9	59.5	30.1	46.5
WebWorld-14B	54.2	23.9	51.3	31.7	53.9	34.6	45.5
Ours							
REPLAYWORLD-27B	60.0	25.1	**67.2**	**49.5**	**69.8**	40.6	57.5
REPLAYWORLD-397B	**64.9**	**45.4**	66.6	43.0	69.3	**44.0**	**59.4**

Table 2: Evaluation results on AgentWorldBench.

	AgentWorldBench						
	Format	EnvReac	Consist.	Factual.	LongHor.	ActPres	Avg↑
Qwen3.5-397B-A17B	94.0	83.3	72.6	57.7	32.9	45.3	64.3
GLM-5	88.9	66.8	75.2	56.5	33.2	42.9	60.6
Kimi K2.5	95.0	75.3	75.1	54.2	35.7	43.3	63.1
Qwen-AgentWorld-35B-A3B	94.3	**83.8**	81.1	57.3	**39.5**	46.3	67.1
EnvScaler-Qwen3-8B	57.9	32.3	42.9	26.6	27.0	25.5	35.4
EnvACE-Qwen3-8B	55.6	29.9	42.3	27.0	23.9	25.4	34.0
REPLAYWORLD-27B	85.1	69.3	81.2	**57.8**	39.4	**46.4**	63.2
REPLAYWORLD-397B	**95.8**	83.7	**81.3**	57.7	39.4	46.2	**67.4**

Table 3: Agent performance on different domains after fine-tuning in each environment simulator. Subscripts: change from Vanilla.

	Web Navigation			CoWork				Tool Use				Overall
	WebShop	Mind2Web	Avg.	OS	SheetCopilot	OfficeBench	Avg.	ACEBench	BFCL	τ^2-bench	Avg.	
Qwen3-8B												
Vanilla	50.9	16.3	33.6	28.5	13.1	35.6	25.7	49.1	61.0	17.7	42.6	34.0
LLM-as-WM	61.6$_{\uparrow 10.8}$	36.8$_{\uparrow 20.5}$	49.2$_{\uparrow 15.7}$	27.1$_{\downarrow 1.4}$	17.6$_{\uparrow 4.5}$	45.0$_{\uparrow 9.4}$	29.9$_{\uparrow 4.2}$	**76.3**$_{\uparrow 27.1}$	58.4$_{\downarrow 2.6}$	28.2$_{\uparrow 10.4}$	54.3$_{\uparrow 11.6}$	43.9$_{\uparrow 9.8}$
EnvScaler-8B	54.5$_{\uparrow 3.6}$	23.8$_{\uparrow 7.6}$	39.2$_{\uparrow 5.6}$	29.9$_{\uparrow 1.4}$	9.0$_{\downarrow 4.1}$	25.2$_{\downarrow 10.4}$	21.4$_{\downarrow 4.4}$	55.7$_{\uparrow 6.6}$	58.8$_{\downarrow 2.3}$	17.8$_{\pm 0.0}$	44.1$_{\uparrow 1.4}$	34.3$_{\uparrow 0.3}$
EnvACE-8B	50.2$_{\downarrow 0.6}$	19.7$_{\uparrow 3.5}$	35.0$_{\uparrow 1.4}$	29.2$_{\uparrow 0.7}$	10.4$_{\downarrow 2.7}$	39.5$_{\uparrow 3.9}$	26.4$_{\uparrow 0.6}$	50.6$_{\uparrow 1.5}$	59.4$_{\downarrow 1.6}$	28.2$_{\uparrow 10.5}$	46.1$_{\uparrow 3.4}$	35.9$_{\uparrow 1.9}$
AWM-8B	48.9$_{\downarrow 1.9}$	15.6$_{\downarrow 0.6}$	32.3$_{\downarrow 1.3}$	25.0$_{\downarrow 3.5}$	10.9$_{\downarrow 2.3}$	37.7$_{\uparrow 2.1}$	24.5$_{\downarrow 1.2}$	47.0$_{\downarrow 2.1}$	60.1$_{\downarrow 1.0}$	16.7$_{\downarrow 1.1}$	41.2$_{\downarrow 1.4}$	32.7$_{\downarrow 1.3}$
EnvFactory-8B	52.2$_{\uparrow 1.4}$	20.7$_{\uparrow 4.5}$	36.5$_{\uparrow 2.9}$	26.4$_{\downarrow 2.1}$	9.1$_{\downarrow 4.1}$	22.3$_{\downarrow 13.3}$	19.3$_{\downarrow 6.5}$	47.5$_{\downarrow 1.6}$	54.1$_{\downarrow 7.0}$	23.8$_{\uparrow 6.0}$	41.8$_{\downarrow 0.8}$	32.0$_{\downarrow 2.0}$
REPLAYWORLD	**64.7**$_{\uparrow 13.9}$	**37.2**$_{\uparrow 21.0}$	**51.0**$_{\uparrow 17.4}$	**32.3**$_{\uparrow 3.8}$	**18.6**$_{\uparrow 5.5}$	**49.8**$_{\uparrow 14.2}$	**33.6**$_{\uparrow 7.9}$	75.1$_{\uparrow 26.0}$	**61.4**$_{\uparrow 0.4}$	**29.1**$_{\uparrow 11.4}$	**55.2**$_{\uparrow 12.6}$	**46.0**$_{\uparrow 12.0}$
Qwen3-14B												
Vanilla	53.5	17.1	35.3	31.9	15.4	49.8	32.4	61.4	63.9	29.5	51.6	40.3
LLM-as-WM	67.0$_{\uparrow 13.5}$	38.1$_{\uparrow 21.0}$	52.6$_{\uparrow 17.3}$	31.9$_{\pm 0.0}$	15.8$_{\uparrow 0.5}$	49.8$_{\uparrow 0.1}$	32.5$_{\uparrow 0.2}$	77.3$_{\uparrow 15.9}$	61.5$_{\downarrow 2.3}$	33.3$_{\uparrow 3.9}$	57.4$_{\uparrow 5.8}$	46.8$_{\uparrow 6.5}$
AWM-14B	54.2$_{\uparrow 0.7}$	17.3$_{\uparrow 0.2}$	35.7$_{\uparrow 0.4}$	30.6$_{\downarrow 1.4}$	13.6$_{\downarrow 1.8}$	49.8$_{\pm 0.0}$	31.3$_{\downarrow 1.1}$	62.0$_{\uparrow 0.6}$	63.8$_{\downarrow 0.1}$	24.9$_{\downarrow 4.5}$	50.2$_{\downarrow 1.4}$	39.5$_{\downarrow 0.8}$
REPLAYWORLD	**67.3**$_{\uparrow 13.7}$	**38.3**$_{\uparrow 21.2}$	**52.8**$_{\uparrow 17.5}$	**35.4**$_{\uparrow 3.5}$	**19.5**$_{\uparrow 4.1}$	**56.3**$_{\uparrow 6.6}$	**37.1**$_{\uparrow 4.7}$	**84.9**$_{\uparrow 23.5}$	**65.8**$_{\uparrow 2.0}$	**35.1**$_{\uparrow 5.7}$	**61.9**$_{\uparrow 10.3}$	**50.3**$_{\uparrow 10.0}$

Out-of-distribution evaluation. ReplayWorld's capability generalizes toward other domains. On AgentWorldBench (Zuo et al., 2026), ReplayWorld-397B raises the average from 64.3 for its prompted base to 67.4, above the 67.1 of Qwen-AgentWorld-35B-A3B (full table in Appendix E.1).



Figure 5: RL training dynamics of the agent inside the simulator (left) and its checkpoints' performance on OfficeBench (right).

Table 4: Ablation studies on skill repertoire and harness capabilities. Shaded: the Table 1 configuration.

Skills	Search	Coding	Ref.	OpenCode	Claude Code	Codex
✗	✗	✗	✗	54.0	55.4	55.9
Unified	✗	✗	✗	56.8	58.2	59.0
Unified	✓	✓	✓	59.4	**61.0**	**61.7**
Unified	✗	✓	✓	57.2	58.5	59.1
Unified	✓	✗	✓	57.8	59.1	59.8
Unified	✓	✓	✗	58.2	59.5	60.3
Per-Env	✓	✓	✓	**60.6**	60.8	61.0
Per-Domain	✓	✓	✓	58.3	59.7	60.1

Table 5: Component ablation studies on the WebShop subset of ReplayBench.

	WebShop (ReplayBench)						WebShop†
	Format	EnvReac	Consist.	Factual.	ActPres	Avg↑	Score
Full	**71.2**	**51.5**	**80.4**	**70.2**	30.4	**60.7**	**64.7**
−*MCCA*	70.4	47.2	74.0	68.4	31.8	58.4	61.9
−*MCCA* −*LPF*	70.7	48.6	72.0	67.6	**32.4**	58.3	61.4
−*MCCA* −*LPF* −*SBS*	71.0	45.8	72.9	66.9	31.1	57.5	60.9

WebShop†: Agent with backbone Qwen3-8B tested on WebShop dataset.

4.3 World Models as Environment Simulators for Agent Training

Serving as an environment simulator, ReplayWorld also boosts the performance of agents trained in it, beyond what other trained language world models achieve. where observations carry state across steps (OS, OfficeBench, ACEBench), which a skill that records what an earlier action left behind supplies. With reinforcement learning in the simulated environment of ReplayWorld, the agent performs better than that trained on the simulator of prompted LLM .

4.4 Ablation Study

Harness capabilities. With Qwen3.5-397B-A17B as the world model, the evolved repertoire and then the full set of capabilities each add 2.6 to 3.1 points of Overall under all three harnesses (Table 4), and removing any one capability lowers Overall under every harness, search most, then code execution, then the reference traces. The repertoire organizations differ only in the skills the index offers An edit written from one batch can change predictions wherever its skill is opened, and taking the last cycle instead of MCCA's winner costs 2.3 points for the world model and 2.8 for its policy (Table 5). Without LPF, error, empty and refusal traces, whose type is learnable but whose content is not, enter the batch and trade Consistency and Factuality for EnvReaction; without SBS, batches concentrate on the majority flow and EnvReaction drops.

4.5 Further Analysis

Skill Evolution. Replay helps most where the corpus holds real executions, since the loop writes its skills from what the environment returned: at the selected cycle of Figure 6 the largest gains are on ACEBench, OS and WebShop, whose corpora are executed rollouts or logged runs on live commerce sites.

What the repertoire contributes. The gain is knowledge specific to each environment, and it lands on the kind of reply the environment gives. Under the unified repertoire, 83% of the skills the simulator loads belong to the trace's own environment, and traces that read only such skills gain 11.2 points of Consistency over the same traces without skills, whereas Mind2Web traces that read WebShop skills under per-domain repertoires lose 8.1. On the 27% of traces where LLM-as-WM predicts the wrong response type, ReplayWorld gains 16.7 points.

Environment shift. Because that knowledge is text tied to traces, it updates by replay rather than retraining. After OS moves from Ubuntu to Alpine, the repertoire evolved on Ubuntu keeps only 4.5 points of outcome agreement over the harness without a repertoire on the steps where Alpine replies differently; one replay cycle on ten new traces raises the margin to 15.0, and one on all new traces to 19.5 (Appendix E.6).

Case study. Search and code execution let the simulator predict what no stored experience holds (Figure 7). On a live-shell command outside the eight environments, ReplayWorld-397B fetches

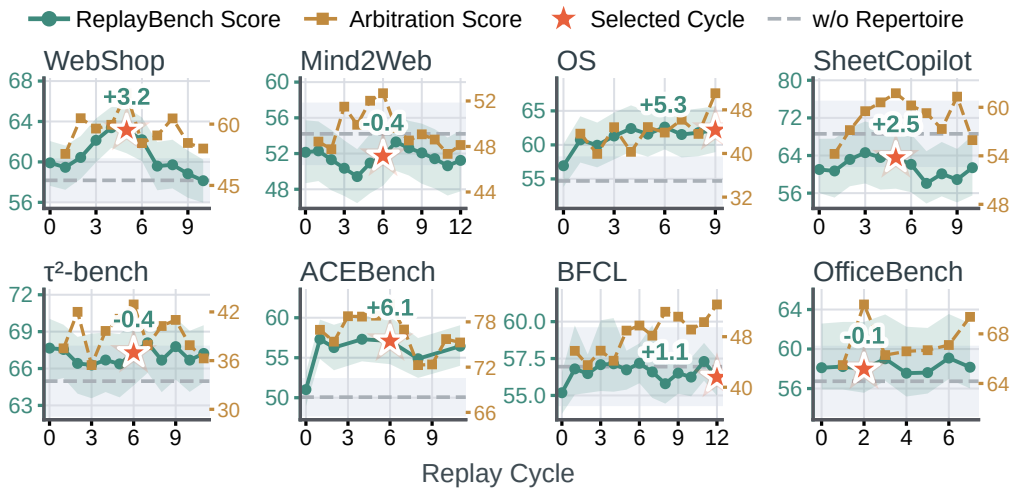

Figure 6: ReplayBench score and arbitration score across replay cycles; ★: the selected cycle.

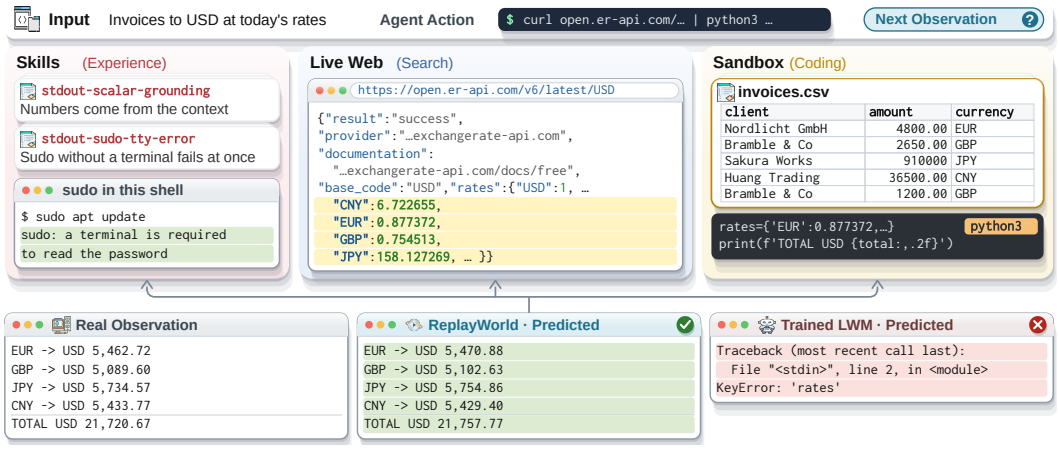

Figure 7: Case study on a new live-shell task whose reply depends on real-time exchange rates, where ReplayWorld looks up and computes based on skills and harness capabilities.

the exchange rates from the page the command calls, computes the totals in a sandbox and lands within 0.4% of the shell's reply, while Qwen-AgentWorld-35B-A3B, predicting from its weights alone, returns a crash (Appendix F.1).

5 Conclusion

We introduce ReplayWorld, an evolving agentic world model that combines simulator skills, code execution, and web search to synthesize agent training trajectories. Replay-driven skill evolution updates its environment knowledge from recorded experience without modifying model parameters. Evaluations on ReplayBench and downstream tasks show that agentic capabilities improve environment response prediction and agent performance.

AI use statement

We used large language models to polish the writing (grammar and clarity). They were not used to generate research ideas, design experiments, or analyze results; all AI-assisted text was reviewed by the authors, who take responsibility for the final content.

Ethics statement

This work trains and evaluates language agents on eight public benchmarks; the environments run in sandboxed containers and local executors. The replay corpus and ReplayBench are derived from publicly released datasets and from logged interactions of open-weight agents with those environments, redistributed under the upstream licenses (Appendix C.1); cluster identifiers that appeared inside observations were pseudonymized. The one human component is the annotation study of Appendix D.4, in which annotators rate simulator outputs against a written brief; it involves no personal data and no judgements about people, and we see no ethical concerns beyond those common to research on agent training.

Reproducibility statement

Algorithm 1 and the displays of Section 3 specify the loop, the agentic turn, and the training objectives. Appendix A collects the notation, Appendix G the prompts of every role, and Appendices F.2 and F.3 example skills and one replay cycle end to end. Appendix B.1 lists the per-benchmark settings of the loop, Appendix B.2 the synthesis filters, Appendix B.3 the fine-tuning configuration, and Appendix C.1 the data-isolation audits and licenses; Appendices D.1, D.3 and D.4 give the ReplayBench criteria and scoring, the judges, and the human study. Code, the skill repertoires, the simulated trajectory pools and the ReplayBench traces will be made available.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *ICLR 2026*, 2025. arXiv:2507.19457.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *ICML 2026*, 2026. arXiv:2506.07982.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, Wulong Liu, Xinzhi Wang, Defu Lian, Baoqun Yin, Yasheng Wang, and Wu Liu. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*, 2025a.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, Huaxiu Yao, Hengduo Li, Jiacheng Zhu, Xian Li, Dawn Song, Bo Li, Jason Weston, and Dat Huynh. Scaling agent learning via experience synthesis. In *ICLR 2026*, 2025b. arXiv:2511.03773.

Meng Chu, Xuan Billy Zhang, Kevin Qinghong Lin, Lingdong Kong, Jize Zhang, Teng Tu, Weijian Ma, Ziqi Huang, Senqiao Yang, Wei Huang, Yeying Jin, Zhefan Rao, Jinhui Ye, Xinyu Lin, Xichen Zhang, Qisheng Hu, Shuai Yang, Leyang Shen, Wei Chow, Yifei Dong, Fengyi Wu, Quanyu Long, Bin Xia, Shaozuo Yu, Mingkang Zhu, Wenhu Zhang, Jiehui Huang, Haokun Gui, Runyi Li, Chenyu Tang, Dong Huang, Xuhang Chen, Rui Liu, Chengzu Li, Shiyi Du, Xu Huang, Haoxuan Che, Long Chen, Qifeng Chen, Wenya Wang, Wenxuan Zhang, Xiaojuan Qi, Yang Deng, Yanwei Li, Mike Zheng Shou, Zhi-Qi Cheng, See-Kiong Ng, Ziwei Liu, Philip Torr, and Jiaya Jia. Agentic world modeling: Foundations, capabilities, laws, and beyond. *arXiv preprint arXiv:2604.22748*, 2026.

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. In *NeurIPS 2023*, 2023. arXiv:2306.06070.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-World: Scaling Real-World Environment Synthesis for Evolving General Agent Intelligence. arXiv:2604.18292, 2026.

Zisu Huang, Jingwen Xu, Yifan Yang, Ziyang Gong, Qihao Yang, Muzhao Tian, Xiaohua Wang, Changze Lv, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Xue Yang, Dongdong Chen, Xiaoqing Zheng,

and Chong Luo. From raw experience to skill consumption: A systematic study of model-generated agent skills. *arXiv preprint arXiv:2605.23899*, 2026.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents. In *ICML 2026*, 2026. arXiv:2603.12056.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Xuan Ren, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. Agentic proposing: Enhancing large language model reasoning via compositional skill synthesis. *arXiv preprint arXiv:2602.03279*, 2026.

Seanie Lee, Sanjoy Chowdhury, Chao Jiang, Cheng-Yu Hsieh, Ting-Yao Hu, Alexander T. Toshev, Oncel Tuzel, and Raviteja Vemulapalli. Environment-free Synthetic Data Generation for API-Calling Agents. arXiv:2607.16900, 2026.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. Sheetcopilot: Bringing software productivity to the next level through large language models. In *NeurIPS 2023*, 2023. arXiv:2305.19308.

Yixia Li, Hongru Wang, Peng Lai, Zhiwen Ruan, He Zhu, Youxin Zhu, Ganlong Zhao, Minda Hu, Yun Chen, Sibei Yang, Peng Li, Jeff Z. Pan, Jia Pan, Guanhua Chen, Yang Liu, and Guanbin Li. Bridging the agent-world gap: Text world models for llm-based agents. *arXiv preprint arXiv:2606.09032*, 2026.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025.

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. In *ICLR 2024*, 2023. arXiv:2308.03688.

Yi Liu, TingFeng Hui, Wei Zhang, Li Sun, Ningxin Su, Jian Wang, and Sen Su. Envsimbench: A benchmark for evaluating and improving llm-based environment simulation. *arXiv preprint arXiv:2605.07247*, 2026.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. In *ICML 2026*, 2026. arXiv:2602.01869.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *NeurIPS 2024*, 2023. arXiv:2305.15334.

Baochang Ren, Yunzhi Yao, Rui Sun, Shuofei Qiao, Ningyu Zhang, and Huajun Chen. Aligning agentic world models via knowledgeable experience learning. In *Findings of EMNLP 2026*, 2026. arXiv:2601.13247.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling Tool-Interactive Environments for LLM Agent via Programmatic Synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, 2026. ACL Anthology 2026.findings-acl.407; arXiv:2601.05808.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. Scaleenv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training. In *ICML 2026*, 2026. arXiv:2602.06820.

Yiming Wang, Da Yin, Yuedong Cui, Ruichen Zheng, Zhiqian Li, Zongyu Lin, Di Wu, Xueqing Wu, Chenchen Ye, Yu Zhou, and Kai-Wei Chang. Llms as scalable, general-purpose simulators for evolving digital agent training. *arXiv preprint arXiv:2510.14969*, 2025.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *ICML 2026*, 2026. arXiv:2602.10090.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. Webworld: A large-scale world model for web agent training. In *ICML 2026*, 2026. arXiv:2602.14721.

Minrui Xu, Zilin Wang, Mengyi Deng, Zhiwei Li, Zhicheng Yang, Xiao Zhu, Yinhong Liu, Boyu Zhu, Baiyu Huang, Chao Chen, Heyuan Deng, Fei Mi, Lifeng Shang, Xingshan Zeng, and Zhijiang Guo. EnvFactory: Scaling Tool-Use Agents via Executable Environments Synthesis and Robust RL. arXiv:2605.18703, 2026a.

Zishan Xu, Zhiyuan Yao, Yuxin Chen, Yifu Guo, Zhengxi Lu, Yuquan Lu, Jinyang Huang, Yan Xu, Yasheng Wang, Weinan Zhang, Xingshan Zeng, and Weiwen Liu. EnvACE: Internalizing Environment Dynamics via World Rehearsal for Agentic Reinforcement Learning. arXiv:2608.06197, 2026b.

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. *arXiv preprint arXiv:2505.09388*, 2025.

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

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. In *ICML 2026*, 2026. arXiv:2601.21557.

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. Polyskill: Learning generalizable skills through polymorphic abstraction. In *ICLR 2026*, 2025. arXiv:2510.15863.

Jiayi Zhang, Yiran Peng, Fanqi Kong, Cheng Yang, Yifan Wu, Zhaoyang Yu, Jinyu Xiang, Jianhao Ruan, Jinlin Wang, Maojia Song, HongZhang Liu, Xiangru Tang, Bang Liu, Chenglin Wu, and Yuyu Luo. AutoEnv: Automated Environments for Measuring Cross-Environment Agent Learning. arXiv:2511.19304, 2025a.

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. In *ICML 2026*, 2025b. arXiv:2510.08558.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *ICLR 2026*, 2025c. arXiv:2510.04618.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for llm agent planning. In *Findings of EMNLP 2026*, 2026. arXiv:2606.30639.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Cyber-zero: Training cybersecurity agents without runtime. In *ICLR 2026*, 2025. arXiv:2508.00910.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, Yantao Liu, Qin Zhu, Yinger Zhang, Bowen Yu, Haiquan Zhao, Haiyang Xu, Jianxin Yang, Jiayang Cheng, Junyang Wang, Lianghao Deng, Mingfeng Xue, Tianyi Bai, Yang Fan, Yubo Ma, Yucheng Li, Zeyu Cui, Zhihai Wang, Zhihui Xie, Zhuorui Ye, An Yang, Dayiheng Liu, Jingren Zhou, and Ning Ding. Qwen-agentworld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026.

APPENDIX CONTENTS

A Notation **15**

B ReplayWorld: Implementation Details **15**

B.1 Replay Loop Settings 15

B.2 Trajectory Synthesis Filters 16

B.3 Policy Training Configuration 17

C ReplayBench: Data **18**

C.1 Replay Corpus and Test Set Construction. 18

C.2 Composition 19

C.3 Characterization and Validity 20

D ReplayBench: Evaluation Protocol **20**

D.1 Criteria and Scoring 20

D.2 Harness Configuration 22

D.3 Judge Selection and Agreement 22

D.4 Human Study. 22

E Additional Results **24**

E.1 AgentWorldBench: Per-Criterion Results. 24

E.2 Arbitration Score across Replay Cycles 24

E.3 Repertoire Evolution: Strategies and Edit Census 24

E.4 Retrieval-Grounded Prompting. 27

E.5 Gain by Trace Difficulty 27

E.6 Environment Shift. 27

E.7 Inference Cost 27

F Qualitative Examples. **28**

F.1 Case Study: One Step per Capability 28

F.2 Example Skills 29

F.3 A Replay Cycle, End to End. 32

G Prompts **34**

A NOTATION

Table 6 lists the symbols of Section 3 and of Appendices B.1 and B.2; a hat marks a simulated quantity or an estimate, $\sim$ a draw, and $\leftarrow$ an assignment.

Table 6: Symbols of Section 3 and of the settings and synthesis-filter appendices.

Symbol	Meaning
Environment and interaction	
E, P_E	Environment and its transition distribution
q, o_t, a_t, h_t	Task; observation, action and history at step t
τ, T_τ	Trajectory, a trace when logged, and its length
π_ϕ	Agent policy with parameters ϕ
Simulator and skills	
p_θ, θ	Language model of the simulator and its fixed weights
S	Repertoire of simulator skills
$\hat{P}_S$, $\hat{o}_{t+1}$, $\hat{h}_t$	Simulator over S; predicted observation; simulated history
α_E	Format exemplar, one real observation per response type
c_j, y_j	Context and reply j of an agentic turn
T, J	Tool executor of the harness; index of the reply taken as the prediction
M	Reply cap of a turn
$\hat{P}_S^{\mathrm{inj}}$, $\hat{P}_S^{(1)}$	Injected instance for synthesis; single-reply instance for LPF
Replay cycle and arbitration	
$\mathcal{D}_{\mathrm{re}}$, $\mathcal{D}_{\mathrm{arb}}$	Replay traces; arbitration traces
k; S_0, S_k	Cycle index; initial repertoire; alternative after cycle k
$\mathcal{P}_k$, $\mathcal{B}_k$, B	Pre-sample; batch; batch-size cap
ℓ, n_ℓ, p_ℓ	Query cluster; its eligible trace count; its SBS probability
$\mathrm{SBS}(D, n)$, $\mathrm{Top}_n(D; g)$	Balanced draw of at most n traces; the at most n traces of largest g
g_k; ρ_k^{r}, $\rho_k^{\max}$, ρ_k^{mean}	Learnability; percentile ranks of reward, largest and mean replay error
$e_{\tau,t}$, emb, J_τ, n_{step}	Replay error; observation embedding; scored transitions; their requested number
n_{pre}, n_{sl}	Size caps of the pre-sample and of LPF's retained set
C_θ, A_θ, R_θ	Comparator, Aggregator and Reviser as output distributions of p_θ
δ_τ, F_k	Discrepancy; discrepancy summary
$\mathcal{U}_k$, $\oplus$	Revision; update operator
$\hat{m}_k$; $\hat{m}_k^{\mathrm{rule}}$, $\hat{m}_k^{\mathrm{sum}}$; $\hat{m}^\star$	Arbitration score; rule check and discrepancy scores; best score so far
η_k, κ	Cycles without improvement; patience
K, $K_{\max}$, $k^\star$, $S^\star$	Cycles run; cycle cap; winner index; selected repertoire
E', $\mathcal{D}'_{\mathrm{re}}$, $\mathcal{D}'_{\mathrm{arb}}$	Shifted environment and its replay and arbitration traces
S'_k, $\mathcal{U}'_k$, $S'^\star$	Alternatives, revisions and selected repertoire after the shift
Synthesis and training	
$\hat{\mathbb{P}}_\pi$, ϱ	Simulated-trajectory distribution under π; distribution of a task and its first observation
$T_{\max}$	Step cap of the benchmark's interface
μ, w	Behavior policy; number of recent steps it reads
$\mathcal{D}_{\mathrm{sim}}$	Training pool of simulated trajectories
$\mathcal{L}_{\mathrm{SFT}}$, $a_{t,j}$, N_{tok}	Action-token loss; token j of a_t; number of action tokens
ψ, r_ψ	Judge and its binary outcome reward

B REPLAYWORLD: IMPLEMENTATION DETAILS

We list the settings of the replay loop, the filters applied to synthesized trajectories, and the training configuration of the policy models.

B.1 REPLAY LOOP SETTINGS

We cap a skill body at 5,000 characters on all benchmarks, set the patience to $\kappa = 5$, and stop at the cycle cap $K_{\max} = 12$, or 15 on WebShop. After every fifth cycle, consolidation merges skills whose word-overlap ratio exceeds 0.7; the ratio divides the number of shared words by the size of the smaller word set. On ACEBench the Reviser also receives a table of recorded outcome counts per tool, and each trace is replayed at its last transition. After an environment shift, we cluster the traces of E' apart from those of E and replay each at its last transition. The harness then takes the interface and any exemplar from E', and arbitration runs on the arbitration traces $\mathcal{D}'_{\mathrm{arb}}$ of E'.

The initial repertoires hold 6 (WebShop), 5 (Mind2Web), 10 (SheetCopilot), 8 (OS), 12 (OfficeBench), 1 (ACEBench), 11 (BFCL) and 8 (τ^2-bench) skills. In the same order, the selected repertoires used in Table 1 hold 14, 5, 10, 14, 12, 5, 20 and 19.

Mind2Web runs a multiple-choice variant of the loop that has no Comparator, Aggregator or Arbiter and accepts a revision only if it raises the element accuracy on a fixed training batch by a margin.

The frequency threshold. The frequency threshold is 0.10: only causes whose share passes it go to the Reviser, except on WebShop, where all causes do. An earlier filter sits in the Aggregator's prompt, which asks for root causes found in at least 20% of the batch's discrepancies. On ACEBench, BFCL and OfficeBench a second instruction in that prompt asks for 10%, and the SheetCopilot prompt has no such filter.

SBS, LPF and the cold-start ranking. An offline two-level k-means over BGE-M3 query embeddings gives the clusters ℓ of Equation 2. SBS first draws $n_{\mathrm{pre}} = 2{,}000$ traces, or the whole pool if it is smaller. LPF keeps the $n_{\mathrm{sl}} = 1{,}000$ traces of largest learnability g_k (600 on SheetCopilot, 350 on OS and 2,000 on OfficeBench), and SBS draws the batch of $B = 200$ from them (30 on SheetCopilot). In cycle k, the scoring pass of LPF reads h_t, a_t and o_{t+1} from a trace $\tau \in \mathcal{P}_k$ and draws a fresh prediction $\hat{o}_{t+1}$ at each sampled step. Here $\hat{P}^{(1)}_{S_{k-1}}$ is the current simulator run with a single reply and no tools, and the sampled steps and their predictions are local to the scoring pass. The embedding emb maps observations to nonzero real vectors of a common fixed dimension. We request $n_{\mathrm{step}} = 3$ scored transitions per trace (one on WebShop), so J_τ holds $\min(n_{\mathrm{step}}, T_\tau)$ transition indices drawn uniformly without replacement from $\{0, \dots, T_\tau - 1\}$. The error ranks of g_k are taken over the replay errors

$$e_{\tau,t} = 1 - \cos\big(\mathrm{emb}(\hat{o}_{t+1}),\, \mathrm{emb}(o_{t+1})\big), \qquad \hat{o}_{t+1} \sim \hat{P}^{(1)}_{S_{k-1}}(\cdot \mid h_t, a_t), \qquad t \in J_\tau. \tag{7}$$

In the first cycle a cold-start ranking replaces LPF: a trace scores one point for a recorded reward in $[0.3, 0.9]$ and one for a length between the 50th and 90th percentiles of the pre-sample.

The arbitration score. The rule check score $\hat{m}^{\mathrm{rule}}_k$ averages, over the arbitration sample, benchmark-specific checks of three kinds against the recorded observation: identifier recall, field coverage and outcome class. The discrepancy score $\hat{m}^{\mathrm{sum}}_k$ draws on the discrepancy summary and the Comparator's labels. It is a weighted sum of three terms. One minus the mean root-cause share has weight 50, the share of high or medium quality labels has weight 25, and a term decreasing in the number of distinct root causes has weight 25. Each cycle draws a fresh arbitration sample of at most 300 traces (100 on SheetCopilot); on BFCL and OfficeBench, each cycle scores all arbitration traces instead. Ties go to the earlier cycle, and $S^\star = S_0$ when no pass returns a score.

Calls per cycle. With the WebShop settings a cycle makes about 3,000 model calls, two thirds of them for LPF's scoring pass, which the first cycle omits.

B.2 Trajectory Synthesis Filters

The benchmark's interface sets a step cap $T_{\max}$, and a simulated trajectory ends at the first step $t \geq 1$ at which

$$\begin{gathered} a_{t-1} \text{ is terminal}, \qquad t \geq T_{\max}, \qquad \text{the harness cannot parse } \hat{o}_t, \\ \text{or } \; a_{t-1} = a_{t-2} = a_{t-3} = a_{t-4} \; \text{ with } \; t \geq 4. \end{gathered} \tag{8}$$

The filters below run after synthesis; the pool $\mathcal{D}_{\mathrm{sim}}$ of Section 3.4 holds the trajectories that pass them and, on ACEBench and BFCL, the refusal rows described at the end of this subsection. We deduplicate by query on the WebShop LLM-as-WM pool and on SheetCopilot, and by upstream identifier and trajectory on BFCL; on the other benchmarks the queries are already distinct at generation.

The WebShop LLM-as-WM pool also drops trajectories that end without clicking an offered option, a filter that the ReplayWorld pipeline does not apply. Mind2Web keeps a simulated step when the chosen element is the annotated one; the annotated element must rank in the ranker's top ten, and four hard negatives must be available. On WebShop a codename stands in for the benchmark's name until the trajectories are written to the pool.

OS keeps successful trajectories of at least three steps ending in a terminal action, and drops those whose reasoning leaks into observations, whose final answer is a shell command, or whose numbers no observation supports. SheetCopilot executes each simulated program once in a Python spreadsheet sandbox and keeps the best of three synthesis passes per query if at least 70% of its checked cells match the reference workbook. OfficeBench keeps successful and failed trajectories alike if they have at least three steps, at least 60% non-empty observations and no premature termination.

τ^2-bench keeps successful trajectories with at least two tool calls if some action uses one of the domain's write verbs or the query requests no change. BFCL's composite multi-step tasks come from its multi-turn slice, so we admit them only after a verbatim-substring decontamination check against the four multi-turn evaluation files. The ACEBench pools carry 2,250 generated refusal rows (1,350 for missing parameters, 450 for wrong values and 450 for requests the tools cannot serve). BFCL's pools carry 900 refusals of unservable requests and 796 template refusals to the upstream irrelevance questions. All refusal rows are formatted like the positive examples and are identical in both simulators' pools.

B.3 Policy Training Configuration

We fine-tune all policies with FSDP on four H200 GPUs in bf16, using the no-think chat template of each policy's family. We use a learning rate of 10^{-5}, an effective batch size of 128, a cosine schedule and two epochs, and compute the loss on assistant turns only. Training on both simulators' pools with the same configuration keeps the comparison fair. The context length is the smallest of 8,192, 12,288 and 16,384 tokens that fits the benchmark's longest training example. It is 12,288 on OS and SheetCopilot, and 12,288 or 16,384 on τ^2-bench, depending on the policy. In the pools expanded step by step (WebShop, Mind2Web, OS, τ^2-bench), we supervise only the last step of each example. The shared hyperparameters have two exceptions. The SheetCopilot LLM-as-WM policy uses an effective batch size of 32, a learning rate of 2×10^{-5} and four epochs. On OfficeBench, both Qwen3-14B policies use a learning rate of 5×10^{-6}, and the one trained on the LLM-as-WM pool runs for one epoch.

Departures from the shared design. The two pools of each benchmark start from the same queries except on SheetCopilot. They share one query file on WebShop (8,025 queries), OS (3,516), OfficeBench (7,075), ACEBench (3,851) and τ^2-bench (800), and one query list on BFCL. On Mind2Web, both pools use the steps of the official training split. On SheetCopilot, the 2,207 trajectories of the ReplayWorld pool come from two query files, 885 from the 1,821-query set of the LLM-as-WM pool and 1,322 from a later 2,731-query set.

Reinforcement learning configuration. The runs of Section 4.3 further train the OfficeBench Qwen3-8B ReplayWorld policy with GRPO on 27 of the 93 single-application tasks outside the 106 evaluation tasks. The selection keeps the tasks whose pass rate under the initial policy lies strictly between 0.1 and 0.9, so that a task's rollouts can differ in reward. Each step draws 27 groups of 8 rollouts; the outcome reward is the binary verdict of a judge, Qwen3.5-397B-A17B without reasoning at temperature 0. We keep the parameters and the optimizer state in fp32, train on the same four H200 GPUs and use a learning rate of 10^{-6}. Training runs for 60 steps and then continues for 60 more as a new run from the step-60 policy, with a fresh optimizer state and the step-60 policy as the KL reference. The tasks, seed, rollout budget, rubric and simulator are the same in both runs. We export a policy every ten steps and evaluate it twice on the 106 real tasks.

GRPO objective. With π_{old} the policy that produced the current batch, π_{ref} the reference policy, $G = 8$ rollouts $\{\tau_i\}$ per task, ϵ the clipping range and $\beta = 0.001$ the KL weight, one update maximizes

$$\mathcal{J}_{\text{GRPO}}(\phi) = \mathbb{E}_{\substack{(q,o_0)\sim\varrho \\ \{\tau_i\}_{i=1}^{G}\sim\hat{\mathbb{P}}_{\pi_{\text{old}}}}} \Big[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|\tau_i|} \sum_t \frac{1}{|a_{i,t}|} \sum_j \min\big(\rho_{i,t,j} A_i,\ \text{clip}(\rho_{i,t,j}, 1\pm\epsilon) A_i\big) \Big] - \beta\, \mathbb{D}_{\text{KL}}\big(\pi_\phi \,\|\, \pi_{\text{ref}}\big), \tag{9}$$

where $|\tau_i|$ is the number of steps of rollout i, $\rho_{i,t,j} = \pi_\phi(a_{i,t,j} \mid \hat{h}_{i,t}, a_{i,t,<j}) / \pi_{\text{old}}(a_{i,t,j} \mid \hat{h}_{i,t}, a_{i,t,<j})$ is the token-level ratio, and $A_i = \big(r_\psi(\tau_i) - \text{mean}_k\, r_\psi(\tau_k)\big)/\text{std}_k\, r_\psi(\tau_k)$ is the group-normalized advantage (Shao et al., 2024). The reference policy is the initial policy of each run.

Evaluation protocols. We score policies with each benchmark's own metric:

- WebShop: average reward on 500 tasks.
- Mind2Web: step success rate on 5,911 steps.
- OS: success rate on 144 tasks.
- SheetCopilot: Pass@1 on 221 tasks.

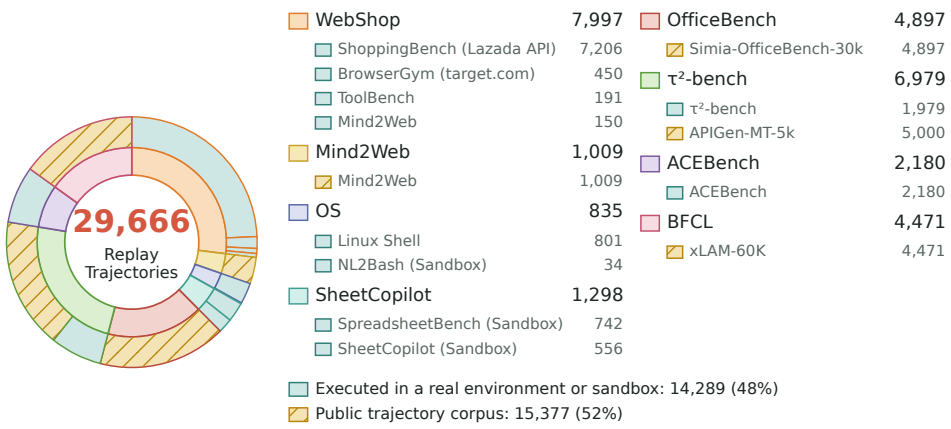

Figure 8: Upstream sources of the replay corpus.

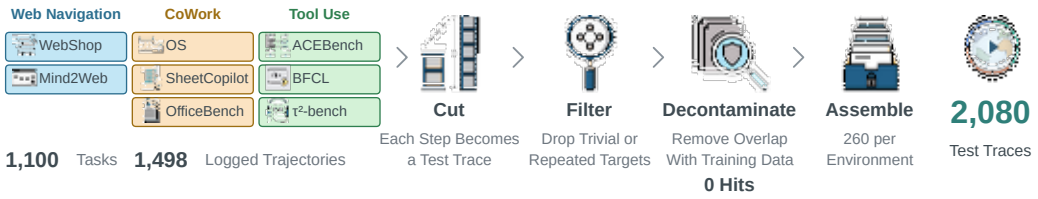

Figure 9: ReplayBench data curation pipeline.

- OfficeBench: mean of the two- and three-application pass rates on 106 tasks.
- ACEBench: average accuracy over the 12 Normal and 3 Special categories.
- BFCL: overall accuracy by the v3 formula over the v3 categories.
- τ^2-bench: macro pass^1 on 114 retail, 50 airline and 114 telecom tasks.

C REPLAYBENCH: DATA

We describe where the replay corpus and the test traces come from, how the test traces are composed, and what they look like.

C.1 REPLAY CORPUS AND TEST SET CONSTRUCTION

We write $\mathcal{D}_{\text{real}} = \mathcal{D}_{\text{re}} \cup \mathcal{D}_{\text{arb}}$ for the replay corpus and $\mathcal{R}_{\text{bench}}$ for the ReplayBench traces. The splits are disjoint at the trajectory level,

$$\mathcal{B}_k \cap \mathcal{D}_{\text{arb}} = \emptyset \ \ \forall k, \qquad \mathcal{R}_{\text{bench}} \cap \mathcal{D}_{\text{real}} = \emptyset. \tag{10}$$

The replay corpus and the fine-tuning and reinforcement-learning task pools contain no evaluation episode of the eight benchmarks. No replay batch shares a trace with the arbitration traces, and no ReplayBench trace is cut from a trajectory of $\mathcal{D}_{\text{real}}$, the corpus from which the repertoires of Tables 1 and 3 evolved. At the coarser level of tasks, where an episode is one run of a task, the fine-tuning and reinforcement-learning pools of ReplayWorld share no task with the eight test sets except for one ACEBench question. The four benchmarks with a fixed task set and no training split (SheetCopilot, OfficeBench, ACEBench, τ^2-bench) contribute other logged runs on those tasks to the replay corpus. No selected skill contains the text of an evaluation task.

Table 7: Sub-sectors of the ReplayBench test traces.

Environment	Sub-sectors (share of the environment's traces)
WebShop	Fashion & Apparel 30%, Home, Furniture & Garden 22%, Beauty & Personal Care 20%, Grocery & Everyday Goods 15%, Electronics & Gadgets 13%, Other 2%
Mind2Web	News & Publishing 25%, Restaurants & Delivery 14%, Scheduling, Email & Meetings 12%, Hotels & Lodging 7%, Other 43%
OS	System Administration 50%, Files & Data Processing 49%, Other 1%
SheetCopilot	Expenses, Invoices & Accounting 47%, Budgeting & Financial Analysis 10%, Investing & Stocks 10%, Other 33%
OfficeBench	Scheduling, Email & Meetings 35%, Reports, Forms & Document Management 31%, Expenses, Invoices & Accounting 17%, Academic Records & Grading 13%, Other 3%
τ^2-bench	Airlines & Flights 63%, Devices & Connectivity 28%, Other 9%
ACEBench	Restaurants & Delivery 38%, Airlines & Flights 16%, Scheduling, Email & Meetings 15%, Other 30%
BFCL	Investing & Stocks 38%, Airlines & Flights 18%, Math & Computation 11%, System Administration 6%, Other 27%

Trace definition. WebShop traces come from the original WebShop text store (goals 500 and up, outside the 0 to 499 evaluation split). OS traces come from an unprivileged user's shell on a Linux host behind a command blacklist, and SheetCopilot traces from spreadsheet programs re-executed in a Python/openpyxl sandbox. The τ^2-bench, ACEBench and BFCL traces come from each benchmark's official simulator or executor, and the OfficeBench traces from its application backends. Mind2Web traces are WebLINX demonstrations on live websites, and on τ^2-bench a target can also be a turn of the benchmark's language-model user. Qwen3.5-397B-A17B collected every trace except the 260 Mind2Web demonstrations. Figure 10 shows how long the contexts and targets of the traces are.

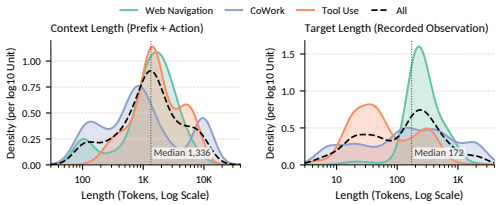

Figure 10: Context and target lengths of ReplayBench by domains.

In the replay corpus of Figure 8, the SheetCopilot, OS, ACEBench and τ^2-bench portions contain executions of the environment itself, and the WebShop portion contains logged runs against live commerce sites.

From trajectories to test traces. Figure 9 summarizes the construction. The logged trajectories come from agent runs in the eight environments and from human demonstrations for Mind2Web. We first cut each logged trajectory, so that each of its L steps becomes a potential test trace $((o, a)_{0:t-1}, o_t)$. A trace survives if its target is a complete observation that the environment returned and clears a novelty floor of 5% against the prefix; at most six traces may share a target within an environment. Its trajectory must also be absent from every replay corpus and from every fine-tuning or reinforcement-learning pool. A final selection keeps at most five traces per task, spread over early, middle and late steps with deeper steps first, until each environment holds 260 traces. In all, 1,498 trajectories of 1,100 tasks yield the test set of 2,080. We use no model score at any stage.

Isolation audit. The OS, OfficeBench and SheetCopilot subsets are splits of the same collection campaigns as their corpora, yet no test trace is cut from a trajectory of its environment's replay corpus. Since the τ^2-bench traces cover 49 of the 50 airline tasks, we isolate at the rollout level: the corpus excludes the 188 rollouts the traces were cut from.

License. The upstream environments are licensed under MIT (WebShop, τ^2-bench, ACEBench), Apache-2.0 (AgentBench, BFCL, OfficeBench), GPL-3.0 (SheetCopilot, of which we redistribute only traces), and CC BY-NC-SA 4.0 (WebLINX, the main source of the Mind2Web subset). We therefore license ReplayBench as a whole under CC BY-NC-SA 4.0 for research use.

C.2 Composition

Sub-sectors of the test traces. We assign sub-sectors with a locally served Qwen3.5-397B-A17B at temperature 0 from a closed list of 37 sub-sectors under 12 industries (Table 7).

Table 8: ReplayBench statistics and validity checks.

Dimension	Measure	Value
Size	Test traces; per environment; source trajectories / tasks	2,080; 260; 1,498 / 1,100
Domain distribution	Application areas / environments / action types	3 / 8 / 24
Prefix length	Median steps before the requested one; median context	4 steps; 1.3K tokens
	First-step traces; contexts above 10K tokens	14.6%; 2.6%
Target length	Median recorded observation; at most 50 tokens; above 1K	172 tokens; 31.5%; 6.6%
Target novelty	Median share of target tokens absent from task and prefix	61% (18.4% of targets at least 90% new)
Refusals	Traces whose target is an error, a failure or an unchanged state	329 (15.8%)
Similarity (Exact / Type)	Qwen3.5-397B-A17B prompted	2.6% / 72.8%
	REPLAYWORLD-397B	5.2% / 81.7%
	Qwen-AgentWorld-35B-A3B	2.0% / 63.9%
	Copy last observation	0.0% / 45.6%
Prediction stability	Same weights and settings predicted twice at temperature 0, 520 traces: identical text (prompted / REPLAYWORLD)	27.3% / 26.3%
	Identical judged score per trace, Format / Consistency / Factuality	84 to 87% / 73 to 78% / 64 to 68%
	Row-level judged score, second run minus first, on the traces neither run truncated	-1.8 / $+0.7$
Judge stability	Same predictions judged twice, 520 traces: identical score per trace, Format / Consistency / Factuality (within one point)	88% / 82% / 74% (93% / 87% / 87%)
	Row-level judged score, second judging minus first	-0.5
Ranking agreement	Spearman between seats over 26 system rows (terra and DeepSeek / terra–Sonnet 5 / DeepSeek–Sonnet 5)	0.966 / 0.988 / 0.965
Human evaluation	100 traces, three annotators: response-type κ; classifier κ; judge and human ρ on Factuality	0.945; 0.911; 0.80 [0.71, 0.87]

C.3 CHARACTERIZATION AND VALIDITY

Table 8 summarizes the test set, the stability of its scores under repeated prediction and judging, and the agreement of the judges with each other and with the annotators.

D REPLAYBENCH: EVALUATION PROTOCOL

We define the six criteria and their scale, describe the harness the simulator runs in, and report the judge and human agreement studies.

D.1 CRITERIA AND SCORING

Criteria. We rate each judged criterion in its own call, so no rationale leaks into another criterion's score. Format asks whether the prediction is a well-formed observation that an agent could consume in place of the real one, whatever its content. EnvReaction asks whether the prediction has the recorded response type (a result, an empty result, an error, an unchanged page, a receipt). A deterministic classifier labels both the predicted and the recorded type. Consistency asks whether the entities, values and states already in the history are carried forward correctly under the latest action. Factuality asks whether the prediction gets right the content of the ground truth that the history does not contain. Its rubric requires a match for a computed result that follows from the given inputs and does not penalize a plausible value for unseen state, such as a catalog entry or a file body. LongHorizon is the mean of Consistency and Factuality on the free-running subsample. ActionPreserve runs one agent, Qwen3.5-397B-A17B with thinking off, once on the predicted and once on the real observation. It counts the two next actions as equivalent when they match after normalization or when gpt-oss-120b

Table 9: EnvReaction under alternative agreement statistics.

	Adjusted BA (EnvReaction)	**Cohen's κ**	**MCC**	**Raw agreement**
Qwen3.5-397B-A17B prompted	29.7	30.6	35.7	72.8
REPLAYWORLD-397B	40.4	43.0	45.6	81.7
Qwen-AgentWorld-35B-A3B	19.4	23.4	27.8	63.9
Claude Opus 5	54.2	56.0	58.7	81.7
REPLAYWORLD − prompted	+10.7	+12.4	+9.9	+8.9

judges them functionally equivalent. We divide the agreement rate by that of the ground-truth copy on the same traces.[1]

Coverage. Consistency needs an earlier real observation and is scored on the traces with $t > 1$ (1,777 of 2,080); ActionPreserve needs a following action and is scored on the 1,707 non-final traces.

Traces for LongHorizon. The 300 free-running traces are a fixed subsample of the 2,080, shared by all rows and drawn from the traces whose prefix leaves three to twelve steps to run. SheetCopilot has none, because the execution report of a spreadsheet program does not continue into a trajectory, and LongHorizon therefore averages over seven environments.

Generation budgets. We allow 30,000 tokens for the Qwen3-8B, Qwen3.5-9B and Qwen3-14B rows, 16,384 for the frontier rows, and 12,288 for the prompted Qwen3.5-397B and the single-call harness rows built on it. The trained world models get 32,768 or 8,192 in their own input formats. A prediction cut off at the length limit gets one retry with 32,768 tokens, the budget that the ReplayWorld rows of Table 1 and all free-running rows used from the start. Where a prefix leaves less than the budget inside the context window (40,960 tokens up to 14B, 131,072 for Qwen3.5-397B), we reduce the budget to fit.

Parsing and empty predictions. The prediction is the first closed `<predicted_observation>` block of a reply, or the whole reply when there is none, and we do not strip reasoning tags from it. Format deducts any reasoning in front of the observation, since the agent would receive that text. An empty or whitespace-only prediction skips the judges and receives the minimum rating and the EnvReaction label `empty`, while `[]`, `{}` and `None` are judged as legitimate observations.

Common scale. We anchor every criterion so that an empty prediction scores 0 and a copy of the real observation 100. EnvReaction, a balanced accuracy over the response types of each environment, needs a correction to meet these anchors. With k response types in an environment, a simulator that always guesses the majority class scores $1/k$, or 50 points on the six environments where $k = 2$. We therefore apply the standard chance correction,

$$\text{EnvReaction} \;=\; \frac{\text{BA} - 1/k}{1 - 1/k} \times 100,$$

computed and clipped at 0 within each environment and then macro-averaged.

Table 9 recomputes EnvReaction on the same classified responses under Cohen's κ, Matthews correlation and raw agreement. The simulators rank the same under the three chance-corrected statistics, and raw agreement ties Claude Opus 5 and ReplayWorld-397B; the chance correction therefore does not drive the results.

Reference rows. Among the reference rows, copying the last observation scores high on Consistency and low on Factuality (human ratings in Table 11), so the two criteria measure different properties. Predicting the most frequent observation scores 0 on EnvReaction, as the chance correction requires.

[1] Ceiling per environment: WebShop 96.1, Mind2Web 83.5, SheetCopilot 98.4, OS 83.5, τ^2-bench 84.2, ACEBench 93.1, BFCL 94.7, OfficeBench 94.2.

Aggregation robustness. The ranking depends little on how the criteria are combined. An ordinal Borda aggregate orders the rows almost identically to the equal-weight mean (rank correlation 0.946), and the two disagree only on pairs of rows whose scores are within about five points of each other.

D.2 Harness Configuration

Harness configuration per benchmark. The harness answers WebShop order confirmations and OfficeBench application switches programmatically, validates outputs on SheetCopilot, OS, τ^2-bench and OfficeBench, and passes α_E to the injected instance and the ReplayBench configurations with all capabilities. The tools available in a turn are `load_skill`, a web-search tool and a page-extraction tool, and the shell and file tools `bash`, `read_file`, `write_file` and `edit_file`; the injected instance uses only `validate_observation` and `get_format_example`. ReplayBench prediction turns have at most $M = 4$ replies, or $M = 6$ when a web-search or shell tool group is enabled. LPF instead uses $M = 1$ with no tools.

HiSkill. After the loop, HiSkill treats each selected skill as an operation, and k-means over BGE-M3 embeddings of the operations' one-sentence summaries groups them into at most six named strategies (Table 14). During synthesis the task text and the first 200 characters of the initial observation select the two nearest strategies. A second call picks three of their operations (five on ACEBench) for the context, with the routed strategy and at most eight observation focus hints.

Harness inputs. Besides the repertoire, the harness supplies the format exemplar α_E, the reference traces (logged trajectories that skills point to) and a note on the shell tool's interface. On WebShop and Mind2Web this material comes from the traces' own environments. There the reference traces are 40 WebShop sessions (goals ≥ 500) and 40 WebLINX demonstrations, task-disjoint from ReplayBench, and the WebShop exemplar holds one verbatim page per response type from six such sessions. The tool note is environment-neutral, and the reference traces of the other six environments are real executions.

D.3 Judge Selection and Agreement

We score the rows of Table 10 with three independent judges from three model families: GPT-5.6-terra at medium reasoning effort, Claude Sonnet 5, and DeepSeek-V3.2. Every judge gets the same reply budget of 16,384 tokens, with one retry at 65,536 when a reply cannot be parsed, and runs at temperature 0 where the model accepts the parameter. Claude Sonnet 5 runs without extended thinking.

Qualification. Before the full run, we screened five judges on 30 calibration traces against criteria fixed in advance. The ground-truth copy must score at least 87.5 on Format, and an empty prediction the minimum on every criterion. The Format gap between a wrong and a right response type in a valid envelope must satisfy $|\Delta| \leq 7.5$ points, which bounds how much content leaks into Format. Every screened judge passed.

Agreement across judges. Table 10 reports the rows of Table 1 that all three judges rated on all 2,080 test traces. The other two judges score 12 to 18 points below GPT-5.6-terra and rank the 26 systems almost as it does (Spearman 0.96 to 0.99, Table 8).

D.4 Human Study

The human study checks the rule-based EnvReaction classifier and the judges against three annotators; no number in the main table comes from it.

Sample. Three annotators independently scored 100 of the 2,080 test traces under one interface and rubric. Each environment contributes eight to twenty-one traces. The predictions come from four simulators, 18 to 23 traces each (ReplayWorld-397B with per-environment repertoires, the prompted 397B model, Claude Opus 5 and prompted Qwen3-8B), and from the copy-last reference row (19 traces). Consistency is defined on the 85 later-step traces ($t > 1$).

Table 10: Evaluation on ReplayBench using different judges. Cells: mean of Format, Consistency and Factuality. Marks as in Table 1.

	GPT-5.6-terra	DeepSeek-V3.2	Claude Sonnet 5	Mean
Proprietary Models				
Claude Opus 5	**76.4**	**63.7**	**64.3**	**68.1**
GPT-6-astra	75.8	62.7	62.9	67.1
Grok 4.6	75.1	59.9	60.5	65.2
GPT-5.6-sol	75.9	60.7	60.8	65.8
Claude Sonnet 5	74.9	60.2	59.6	64.9
Claude Opus 4.6	74.0	60.9	57.4	64.1
Open-Source Foundation Models (>100B)				
Qwen3.5-397B-A17B	71.7	56.3	54.7	60.9
GLM-5	71.3	55.4	53.7	60.1
Kimi K2.5	70.0	55.0	52.7	59.2
DeepSeek-V3.2	69.8	55.0	52.7	59.2
Llama 4 Maverick	66.6	50.8	48.7	55.4
Open-Source Models (<50B)				
Qwen3.6-27B	72.2	57.1	55.1	61.5
Qwen3-14B	66.8	51.9	48.8	55.9
Qwen3-8B	63.7	50.0	46.3	53.3
Qwen3.5-9B	65.2	51.1	47.9	54.7
Language World Models				
Qwen-AgentWorld-35B-A3B	68.3	54.0	51.1	57.8
Ours				
REPLAYWORLD-27B	72.4	57.6	56.6	62.2
REPLAYWORLD-397B	**74.2**	**59.6**	**58.9**	**64.2**

Blinding in two stages. The annotator first states what kind of response the prediction expresses and only then sees the real observation and gives the 1-to-5 scores for Format, Consistency and Factuality. The interface enforces this order and stores the answer given before the reveal, so the label that tests the classifier is chosen blind. The rating scales follow the judge rubrics.

Pre-registered analysis. Before the study we fixed two tests. If the classifier's accuracy against the annotators' majority label falls below 0.70, EnvReaction is dropped from the main table. On the copy-last traces, Consistency, which scores carried-over content, must exceed Factuality, which scores new content.

Results. The classifier agrees with the three-annotator majority label on 91 of the 99 traces that have one, an accuracy above the 0.70 threshold, and its Cohen's κ against that label is 0.911. The annotators agree almost perfectly on the response type (Fleiss' $\kappa = 0.945$, 93 of 100 traces unanimous).

The pre-registered copy-last contrast holds for every annotator: on the 16 copy-last traces with an earlier step, Consistency exceeds Factuality by $+82.8$, $+85.9$ and $+84.4$ points (paired Wilcoxon test, $p \leq 3.8 \times 10^{-4}$ for each). Table 11 gives the per-row means and the agreement statistics. Format and Consistency sit near the top of the scale for every row, copy-last included, with 81% and 86% of their ratings at the top score. Factuality separates the rows, with 5.7 for copy-last and 30.1 to 49.2 for the models. This is the intended division of labor: the first two criteria catch malformed or self-contradicting predictions, and Factuality measures whether the prediction is true.

On Factuality the judge tracks the annotators: Spearman $\rho = 0.80$ $[0.71, 0.87]$ with the three-annotator mean, a mean offset of $+2.7$ points, and the same order of the five rows (Spearman 1.0). For each choice of two annotators, we correlate their mean with the judge and with the third annotator. Averaged over the three choices, the judge's Factuality correlation is 0.776 and the third annotator's is 0.707. The judge therefore agrees with the annotators as closely as they agree with each other. On Format the judge reaches $\rho = 0.72$. On Consistency the judge is 14.7 points stricter than the annotators, with $\rho = 0.29$. With 86% of the ratings at the top score, the annotators leave little to

Table 11: Human study. Top: annotator and judge means on the same traces (0 to 100); ±: 95% half-width. Bottom: agreement per criterion.

		Format		Consistency		Factuality	
Simulator	Traces	Annotators	Judge	Annotators	Judge	Annotators	Judge
REPLAYWORLD-397B, per-env. repertoires	19	90.4 ± 5.1	90.8	92.6 ± 5.4	75.0	43.9 ± 9.8	44.7
Qwen3.5-397B-A17B, prompted	18	93.5 ± 4.1	91.7	95.0 ± 4.6	80.0	35.2 ± 10.2	38.9
Claude Opus 5, prompted	21	93.3 ± 4.0	88.1	97.8 ± 3.1	85.5	49.2 ± 9.5	56.0
Qwen3-8B, prompted	23	84.4 ± 5.8	85.9	81.0 ± 8.6	56.9	30.1 ± 9.1	32.6
Copy last observation	19	97.8 ± 3.1	92.1	91.1 ± 6.4	87.5	5.7 ± 4.6	5.3
Judge vs. annotators, Spearman ρ [95% CI]		0.72 [0.54, 0.86]		0.29 [0.08, 0.50]		0.80 [0.71, 0.87]	
Judge minus annotators, mean over traces		-2.1		-14.7		$+2.7$	
Annotators vs. each other, Krippendorff's α		0.58		0.49		0.62	
Ratings at the top score		81%		86%		16%	

rank on Consistency; its test is the copy-last contrast above, which holds for every annotator. Claude Sonnet 5 reaches $\rho = 0.73$ on Factuality and 0.46 on Format, and DeepSeek-V3.2 reaches 0.52 and 0.46. Ordinal Krippendorff's α among the annotators is 0.62, 0.58 and 0.49 for Factuality, Format and Consistency.

E ADDITIONAL RESULTS

We collect here the tables and figures that the main text summarizes.

E.1 AGENTWORLDBENCH: PER-CRITERION RESULTS

Table 13 reports the per-criterion results on AgentWorldBench, and Table 12 the same breakdown on ReplayBench. We convert 1,338 AgentWorldBench traces, at most 200 per domain, to the benchmark's official message layout and parse outputs with its official output parser. Consistency is scored on the 1,050 non-first traces of this set, LongHorizon on 301 free-running traces and ActionPreserve on 1,158 non-final ones.

E.2 ARBITRATION SCORE ACROSS REPLAY CYCLES

The curves of Figure 6 plot the ReplayBench score of each cycle's repertoire as the mean of the five per-step criteria; LongHorizon is not scored per cycle. The Mind2Web, SheetCopilot, OS and OfficeBench curves come from re-runs of the loop, and the WebShop, τ^2-bench, BFCL and ACEBench curves come from the runs whose selected repertoires Table 1 evaluates. For Mind2Web, we recompute the arbitration score of every cycle on the same 150 development traces, and the star marks the highest one.

For the same runs, Figure 11 plots the arbitration score, which MCCA uses to pick the winner, as a function of the number of traces replayed.

E.3 REPERTOIRE EVOLUTION: STRATEGIES AND EDIT CENSUS

Recurring strategy themes. Table 14 lists verbatim the strategies HiSkill induced in one run per environment. Grouped by hand, they form four families that recur across environments, while most operations under them stay specific to one environment. Schema and field compliance appears in all eight environments; realistic, non-fabricated content in seven; state grounding in six; and mutation versus query semantics in five.

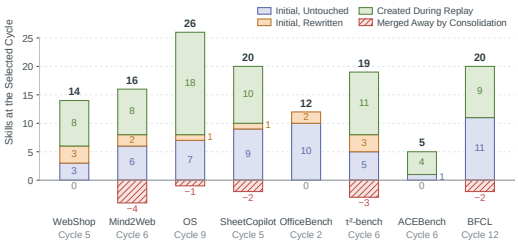

Figure 12: Composition of the selected repertoires.

We then embed the name, description and usage condition of every strategy and operation with the router's own encoder (BGE-M3, 1024 dimensions) and, for each, find the nearest neighbor from any other environment. At the strategy level, 4 of 45 strategies have such a neighbor at a cosine of 0.80

Table 12: Per-criterion results on ReplayBench. Marks as in Table 1.

	ReplayBench						
	Format	EnvReac	Consistency	Factuality	LongHorizon	ActPres	**Avg↑**
Proprietary Models							
Claude Opus 5	87.3	54.2	85.2	**56.8**	**45.9**	**52.8**	**63.7**
GPT-6-astra	88.9	**57.8**	**88.1**	50.6	43.7	52.5	63.6
Grok 4.6	89.3	52.7	86.6	49.3	43.6	49.0	61.7
GPT-5.6-sol	89.8	48.9	88.0	49.7	42.8	50.3	61.6
Claude Sonnet 5	**90.6**	46.7	85.2	49.0	42.3	49.0	60.5
Claude Opus 4.6	89.5	45.4	83.4	48.9	44.6	49.0	60.1
Open-Source Foundation Models (>100B)							
Qwen3.5-397B-A17B	89.8	29.7	79.8	45.5	40.3	46.2	55.2
GLM-5	90.6	28.8	80.8	42.5	41.9	45.6	55.0
Kimi K2.5	**91.7**	29.9	77.2	41.0	43.7	46.3	55.0
DeepSeek-V3.2	90.6	30.3	78.1	40.6	41.9	46.7	54.7
Llama 4 Maverick	86.2	22.8	76.9	36.8	40.3	42.4	50.9
Open-Source Models (<50B)							
Qwen3.6-27B	89.4	30.6	82.5	44.7	46.3	48.2	56.9
Qwen3-14B	87.0	20.7	75.8	37.7	43.3	44.8	51.5
Qwen3.5-35B-A3B	84.4	21.6	72.2	40.8	39.7	45.1	50.6
Qwen3-8B	84.7	20.6	72.5	33.9	41.5	42.1	49.2
Qwen3.5-9B	84.7	19.7	74.0	36.8	36.5	43.5	49.2
Language World Models							
Qwen-AgentWorld-35B-A3B	78.3	19.4	**85.7**	41.1	44.7	45.1	52.4
EnvScaler-Qwen3-8B	85.9	20.8	71.1	34.1	43.4	41.6	49.5
EnvACE-Qwen3-8B	80.6	14.6	70.4	32.5	40.6	40.5	46.5
WebWorld-14B	74.2	18.3	72.1	33.3	31.8	43.4	45.5
Ours							
REPLAYWORLD-27B	87.7	29.1	81.6	47.9	**48.0**	**51.0**	57.5
REPLAYWORLD-397B	90.1	**40.4**	80.4	**51.9**	46.1	47.5	**59.4**

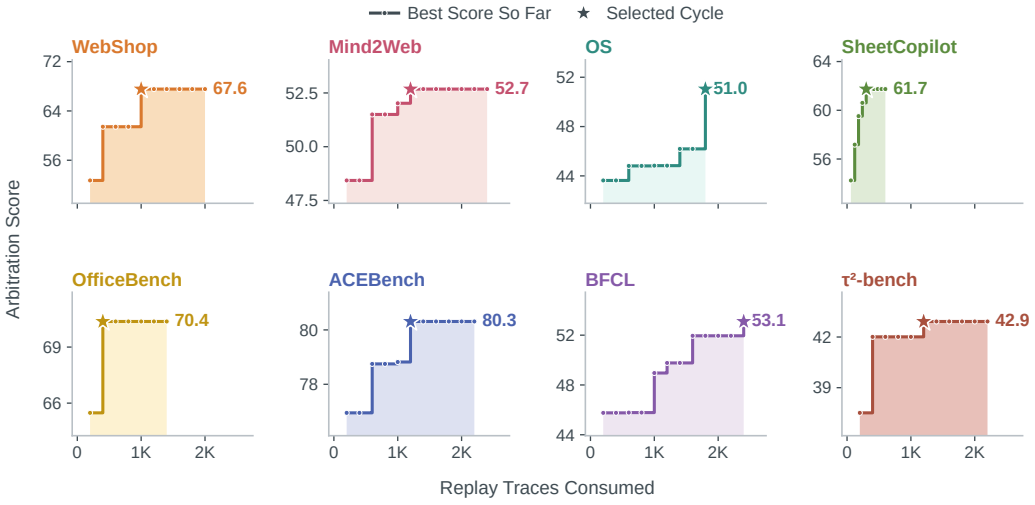

Figure 11: Arbitration score against replay traces consumed per environment.

or above, all four between ACEBench and BFCL, with the highest pair at 0.846. At the operation level, 21 of 119 do (17.6%), again concentrated on the two function-calling environments and on τ^2-bench. The tool-response conventions of ACEBench, BFCL and τ^2-bench are near-duplicates (empty-on-success 0.924, response schema 0.902, value realism 0.885), and identifier continuity recurs across Mind2Web, τ^2-bench and ACEBench. Outside those groups, the only such pair is the observation-format rules of BFCL and OfficeBench (0.819).

Table 13: Per-criterion results on AgentWorldBench. Marks as in Table 1.

	AgentWorldBench						
	Format	EnvReac	Consistency	Factuality	LongHorizon	ActPres	**Avg↑**
Proprietary Models							
Claude Opus 5	94.8	76.3	78.9	66.6	40.2	49.0	67.6
GPT-6-astra	93.6	82.2	**86.3**	**68.8**	40.3	**49.3**	**70.1**
Grok 4.6	**96.5**	**84.7**	82.4	67.7	39.6	48.7	69.9
GPT-5.6-sol	95.6	78.8	84.8	66.9	**43.6**	48.1	69.6
Claude Sonnet 5	96.1	82.6	75.5	61.5	37.5	47.6	66.8
Claude Opus 4.6	95.4	78.0	76.9	62.5	41.3	47.0	66.9
Open-Source Foundation Models (>100B)							
Qwen3.5-397B-A17B	94.0	83.3	72.6	57.7	32.9	45.3	64.3
GLM-5	88.9	66.8	75.2	56.5	33.2	42.9	60.6
Kimi K2.5	95.0	75.3	75.1	54.2	35.7	43.3	63.1
DeepSeek-V3.2	93.1	69.4	72.0	51.7	34.7	42.0	60.5
Llama 4 Maverick	85.3	64.4	69.7	48.0	35.7	40.3	57.2
Open-Source Models (<50B)							
Qwen3.6-27B	93.2	79.1	73.0	55.9	34.8	42.5	63.1
Qwen3-14B	61.9	31.4	44.1	29.6	23.0	27.4	36.2
Qwen3.5-35B-A3B	83.1	73.3	64.2	48.1	31.7	38.9	56.6
Qwen3-8B	56.8	30.6	41.4	27.3	24.8	24.9	34.3
Qwen3.5-9B	85.8	67.3	64.6	47.0	29.1	40.0	55.6
Language World Models							
Qwen-AgentWorld-35B-A3B	94.3	**83.8**	81.1	57.3	**39.5**	46.3	67.1
EnvScaler-Qwen3-8B	57.9	32.3	42.9	26.6	27.0	25.5	35.4
EnvACE-Qwen3-8B	55.6	29.9	42.3	27.0	23.9	25.4	34.0
WebWorld-14B	17.8	13.2	39.5	27.9	13.7	23.2	22.5
Ours							
REPLAYWORLD-27B	85.1	69.3	81.2	**57.8**	39.4	**46.4**	63.2
REPLAYWORLD-397B	**95.8**	83.7	**81.3**	57.7	39.4	46.2	**67.4**

Table 14: Strategies HiSkill induced per environment. In parentheses: the number of operations under each strategy.

Environment	**Strategies (operations)**
WebShop	Reliable Agent Interface Validation (4); Diverse Realistic Search Simulation (4); E-commerce Data Fidelity Strategy (3); Consistent Observation Structure Strategy (1); Mandatory Field Validation Strategy (1); Rich Semantic Observation Generation (1)
Mind2Web	Dynamic Accessibility State Synchronization (3); Stable Tree Prediction Strategy (1); Strict DOM Identity Enforcement (1); Strict Ground Truth Matching (1); Stable AXTree and Identifier Management (5); Grounded State Prediction Strategy (7)
SheetCopilot	Header-Based Row Iteration (2); Excel Data Manipulation Strategy (4); Excel File Preservation Strategy (2); Numeric Data Type Enforcement (1); Categorical Data Aggregation Strategy (1)
OS	Generate Realistic Bash Outputs for Downstream Parsing (2); Simulate Realistic Shell Counting Outputs (1); Simulating Sudo TTY Requirement Failures (1); Enforcing Realistic Linux Command Execution and Output (7); Simulate realistic Linux environment outputs (2)
τ^2-bench	Standardize error responses for invalid tool inputs (1); Enforcing Strict Adherence to Ground Truth Constraints (12); Strict Domain Adherence and Natural Dialogue (3); Enforce Airline Reservation Modification Policies (1); Verify Identity and Confirm Before State Changes (2)
ACEBench	Realistic Tool Response Structuring (5); Realistic Mutation Response Strategy (2); Complete Tool Response Validation (1); AceBench Output Standardization (1); Strict Tool Output Filtering (1); Schema-Compliant Tool Response Generation (3)
BFCL	Realistic Deterministic Response Generation (5); Strict Schema Compliance Strategy (6); Strict JSON Schema Compliance (5); Action and Query Response Differentiation (2); Strict Schema Compliance Strategy (1); Minimal Mutation Confirmation (1)
OfficeBench	Standardized Excel Data Formatting (1); Standardized Observation and Validation Strategy (6); Explicit Finish Signal Strategy (1); Excel Operation Verification (1); File Existence Verification Strategy (1); Standardized App Response Generation (2)

Edit census (create, rewrite, merge). Over the eight runs that produce the cycle curves of Figure 6, the loop created 115 skills, rewrote 121 and merged 30 away. Figure 12 shows the composition of the repertoire selected in each run of Figure 6.

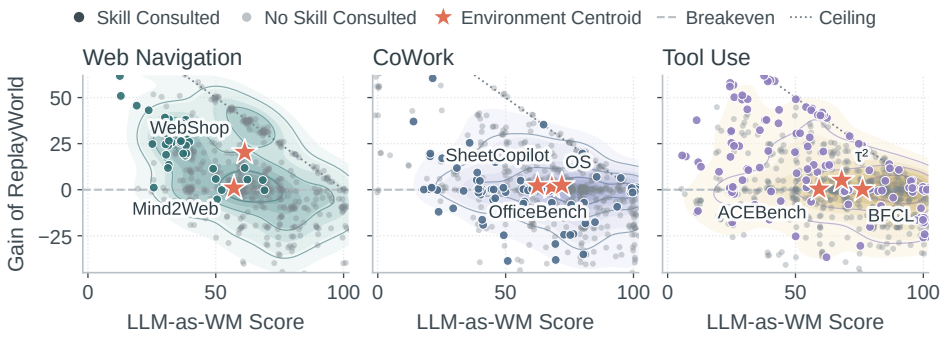

Figure 13: Gain of ReplayWorld over LLM-as-WM per test trace.

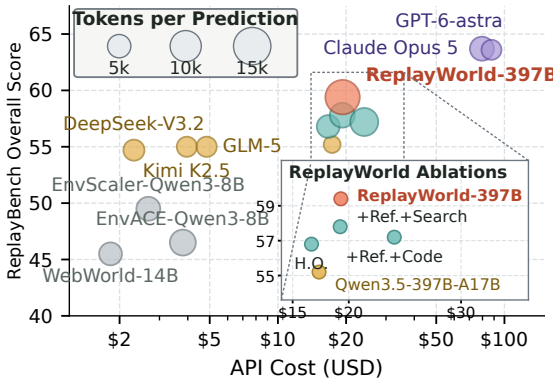

Figure 14: Cost analysis on ReplayBench. API cost of one evaluation at September 2026 rates. H.O.: Harness Only (repertoire, no tools); Ref.: Reference. ■ Proprietary ■ Open-weight ■ Trained world model ■ ReplayWorld ablations ■ ReplayWorld.

E.4 Retrieval-Grounded Prompting

With the three most similar logged transitions of other tasks in context, the prompted model reaches 58.6 Overall, 3.4 above prompting alone. The unified configuration with all capabilities (Table 4) reaches 59.4 without reading any retrieved transitions. Its repertoire of 99 skill files already holds most of what retrieval fetches from the corpus at inference time. With the same-task rollouts masked, the Factuality gain of retrieval comes from traces whose nearest logged transition is close: $+14.1$ at similarity ≥ 0.90 and $+7.8$ between 0.80 and 0.90 against -0.1 below 0.80. Retrieval therefore rewards a corpus that already holds the transition, and we do not rank it in Table 1.

E.5 Gain by Trace Difficulty

Figure 13 shows the per-trace gains that Section 4.5 groups by the LLM-as-WM score.

Error propagation in free runs. On the 300 free-running ReplayBench traces, most failing last steps repeat an earlier error: 93% for the prompted model and 87% for the unified configuration with all capabilities. The unified configuration also goes wrong at the first simulated step less often, on 29.0% of the traces against 37.7% for the prompted model.

E.6 Environment Shift

Table 15 gives the outcome agreement of Section 4.5 for each pool of new traces.

E.7 Inference Cost

Figure 14 plots the Overall score of each simulator against the API cost of one ReplayBench evaluation.

Table 15: Outcome agreement on the changed OS steps after the shift from Ubuntu to Alpine.

Repertoire	Points over no repertoire [95% interval]
Evolved on Ubuntu, unchanged	+4.5 [+0.5, +8.5]
One cycle on 10 traces	+15.0 [+10.5, +20.0]
One cycle on 25 traces	+18.0 [+12.5, +24.0]
One cycle on 54 traces	+20.5 [+14.0, +27.0]
One cycle on 211 traces	+19.5 [+13.5, +25.5]

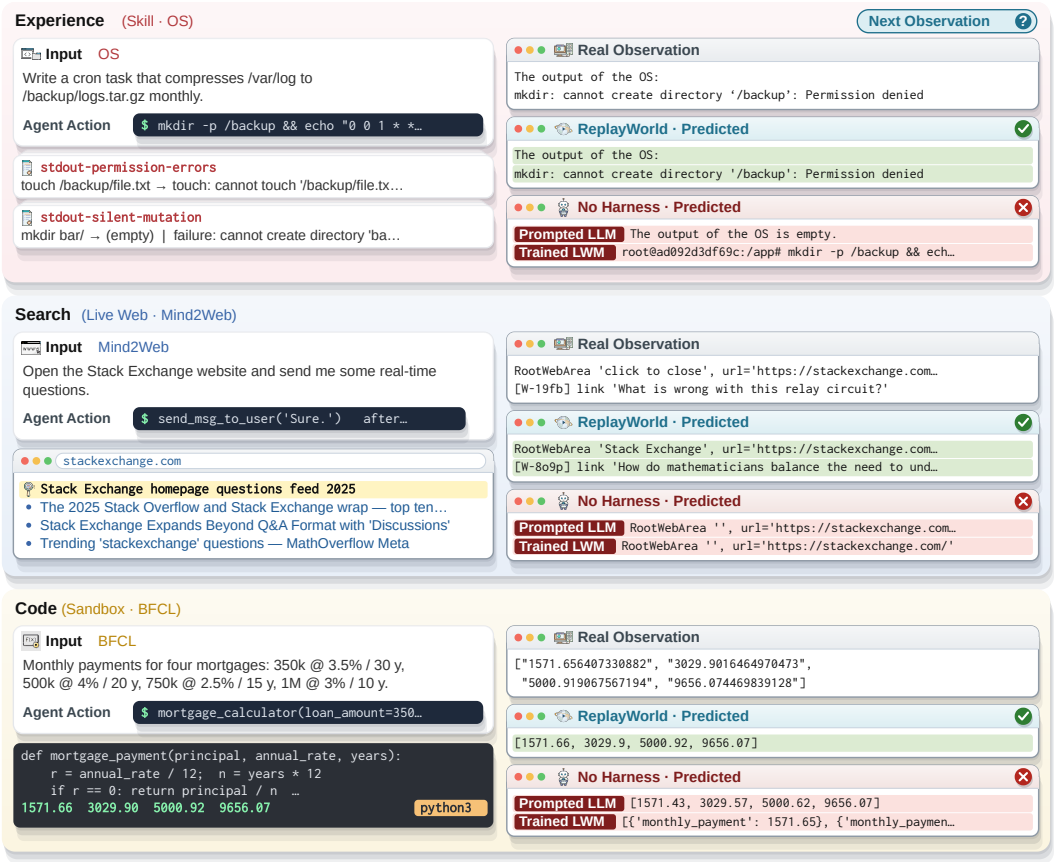

Figure 15: Additional case studies. Experience from the skill repertoire and tools such as web search and a sandbox help the agentic world model identify environment updates.

F Qualitative Examples

This section gives a case study, example skills, and one complete replay cycle.

F.1 Case Study: One Step per Capability

Figure 15 follows one step from each of three ReplayBench traces, in OS, Mind2Web and BFCL, and reads the same way as Figure 7: the left column is what the simulator is given and what it consults, the right column is the real next observation above the predictions. The prompted LLM is Qwen3.5-397B-A17B without its harness, and the trained LWM is Qwen-AgentWorld-35B-A3B. In the first step the agent creates a directory at the root of the file system, which the shell refuses with a permission error. The OS repertoire holds two skills that give the shell's wording for a write into a restricted directory and ask for the tool's exact phrasing, and the prediction reproduces the error line; the prompted LLM predicts an empty output and the trained LWM echoes the command as if it had succeeded. In the second step the agent has opened stackexchange.com and the last observation is still a blank page. The simulator searches the live web, reads the site and predicts a loaded front page with current question titles; the prompted LLM and the trained LWM repeat the blank page. In the third step the agent calls a mortgage calculator four times. The simulator writes the amortization

formula, runs it in the sandbox and returns the four payments, which match the real ones to the cent; the prompted LLM and the trained LWM each get one of the four right within a cent. The judge rates the factuality of the ReplayWorld prediction at 100 in all three steps; the prompted LLM scores 0, 0 and 50, and the trained LWM 0, 0 and 25.

F.2 Example Skills

Each box below shows one skill of an environment's selected repertoire $S^\star$ as the harness stores it. The front matter carries the identifier (`name`) and the applicability condition (`description`), which together make up the skill's line in the skill index; the body below it is the skill body. The boxes quote the files verbatim, [. . .] marks omitted text, and WM in the files is the world model, the simulator.

The WebShop rule ties each action to the page it leads to and carries the title, price, and identifier of the selected item over to the product page that follows.

WebShop: `cnt_reflect_user_actions`

```
---
name: cnt_reflect_user_actions
description: USE WHEN: generating a next observation after an action. Ensures observation
  matches the action's outcome, tool context, and query domain.
---
# Reflect User Actions & Tool Context
[...]
## State Transition Requirements
  - **Search Action**: Observation MUST show search results relevant to the query, not a
  product detail page or home page.
  - **Click Product**: Observation MUST transition from Search Results to Product Detail
  Page (PDP) for the specific clicked item ID.
  - **Pagination (Next/Prev)**: Observation MUST show a different set of products and
  update the page number indicator (e.g., Page 1 -> Page 2).
[...]
  - **Back Navigation**: Observation MUST return to the previous valid state (e.g., PDP ->
  Search Results).

## Consistency Checks
  - **Item Identity**: If viewing a product detail, the Title, Price, and ID must match
  the item selected in the previous step.
[...]
```

The SheetCopilot rule encodes which way of writing a workbook keeps its other sheets and which way replaces them.

SheetCopilot: `preserve-existing-sheets`

```
---
name: preserve-existing-sheets
description: "USE WHEN: writing to one sheet without losing other sheets in the workbook."
---

# Preserve Existing Sheets

Don't call df.to_excel() with default mode='w' -- it overwrites all sheets. Either: (a)
  load the workbook with openpyxl + only modify target sheet + save, or (b) use
  ExcelWriter(path, mode='a', if_sheet_exists='replace').

Example (preserve via openpyxl):
wb = openpyxl.load_workbook('./initial.xlsx')
ws = wb['Target']
# modify ws...
wb.save('./output.xlsx')  # other sheets intact
```

A second SheetCopilot rule states what a cell holds after a formula is written: the formula string itself, which the spreadsheet evaluates when the file is opened.

SheetCopilot: `write-formula-as-string`

```
---
name: write-formula-as-string
description: "USE WHEN: the task asks for an Excel formula (SUM, AVERAGE, IF, VLOOKUP) in
  a cell, not a precomputed value."
---
```

```
# Write Formula As String

Set cell.value = '=SUM(A1:A10)' (note the leading '='). Excel will evaluate when the file
  is opened. Computed values defeat the purpose.

Example:
ws['B1'] = '=SUM(A1:A10)'
ws['C1'] = '=IF(A1>0, "pos", "neg")'
```

The OS rule encodes the error line that each command prints for a missing path and keeps a file removed on an earlier step missing on every later one.

OS: `stdout-file-existence`

```
---
name: stdout-file-existence
description: USE WHEN: WM generates bash stdout for commands targeting non-existent files
  or directories. Ensures 'No such file or directory' errors are emitted instead of silent
  success or empty results.
---
# Stdout: File Existence Errors

## CRITICAL: No Silence on Missing Paths
[...] **If a file or directory is missing, the command MUST emit an error.**
[...]
## Canonical Error Messages
Use exact Linux phrasing for missing resources:

| Command | Exact Error String |
|---|---|
| `cat <missing>` | `cat: <path>: No such file or directory` |
| `ls <missing>` | `ls: cannot access '<path>': No such file or directory` |
| `cd <missing>` | `-bash: cd: <path>: No such file or directory` |
| `rm <missing>` | `rm: cannot remove '<path>': No such file or directory` |
[...]
5. **State Consistency**: If a file was `rm`'d in a previous turn, subsequent accesses
  must fail with this error.
```

The τ^2-bench rule encodes which parts of the database state persist through a conversation: identifiers never change, stored attributes change only through a write call, and a new identifier appears only after a creation call succeeds.

τ^2-bench: `tool-id-continuity-across-turns`

```
---
name: tool-id-continuity-across-turns
description: USE WHEN: WM generates [tool] JSON responses in multi-turn trajectories.
  Ensures IDs (user_id, order_id, line_id) and core attributes remain immutable and
  consistent across all turns, preventing hallucinated swaps.
---
# Tool Response: ID & Attribute Continuity

## The Invariant: Immutable IDs
Once an identifier appears in a `[tool]` response or is provided by the `[user]`, it is
  **immutable** for the remainder of the trajectory.
[...]
## Attribute Consistency (Value Fidelity)
Core attributes associated with an ID must remain stable unless explicitly mutated by a
  prior tool call.
[...]
- If `get_order_details` returns `status: "shipped"`, later turns cannot revert to
  `"pending"` unless a specific reversal tool was called.
[...]
## Handling New Entities
New IDs are **only** generated when a creation tool succeeds (e.g., `book_reservation`,
  `create_order`).
- In this case, the new ID is captured and becomes the immutable reference for subsequent
  turns.
- Do not generate new IDs for read-only operations or failed mutations.
[...]
```

The ACEBench rule encodes when the environment refuses a call on an entity it does not hold, together with the exact refusal message, and keeps an entity deleted earlier in the episode refused.

ACEBench: `env-entity-existence`

```
---
name: env-entity-existence
description: USE WHEN: evaluating if a tool call involving users, senders, receivers, or
  reminders should succeed or fail based on entity validity.
---
# Entity Existence Validation

## Core Principle
- The environment refuses calls involving non-existent entities with specific error
  messages. Do not assume an entity exists unless it was explicitly created, listed, or
  successfully queried in the interaction history.
[...]
- **Users/Senders/Receivers**:
  - If a tool like `view_messages_between_users`, `search_messages`, or `send_message` is
  called with a username not previously confirmed (via successful login, user creation, or
  listing), predict refusal.
  - Expected refusal messages: "User does not exist", "Sender does not exist", "Receiver
  does not exist", or "Sender or receiver does not exist".
[...]
- **Reminders**:
  - If `view_reminder_by_title` is called with a title not previously confirmed (via
  `add_reminder` success or `search_reminders` result), predict refusal.
  - Expected refusal message: "No reminder found with the title <value>."
[...]
## State Continuity
- If an entity was deleted in the history (e.g., via `delete_message` or user removal),
  subsequent calls referencing it must be refused as non-existent.
[...]
```

The BFCL rule encodes that a response repeats the identifiers of its call with the same type and value, so that the next call of a chain refers to the same entity.

BFCL: `func-id-consistency`

```
---
name: func-id-consistency
description: "USE WHEN: Query or prior function_call carries a specific ID and WM
  generates the response. Response IDs must echo the call's IDs so downstream chained
  calls reference the same entity."
---
# BFCL Function Response ID Consistency

## Core Invariant

If function called `get_X(x_id=V)`, the response's `x_id` (or equivalent) MUST be `V`.
  Otherwise the next function call that passes the response's ID downstream references a
  phantom entity.

## ID Echoing Rules

1. **Echo input IDs verbatim** -- same type, same value
2. **Don't string-cast integers**: if `is_id=123` input, response is `{"is_id": 123, ...}`
  (int), not `"is_id": "123"` (string)
3. **Multi-call parallel** -- each response's IDs echo its corresponding call's IDs
4. **Generated IDs** (create verbs) can be new but must use realistic format
[...]
```

The OfficeBench rule encodes the exact error for an action issued in the wrong application and fixes the current application as the last one switched to successfully, which the error leaves unchanged.

OfficeBench: `obs-wrong-app-error`

```
---
name: obs-wrong-app-error
description: "USE WHEN: WM predicts a tool call result when the agent calls a tool while
  in the wrong app."
---
# OfficeBench Wrong-App Error Format

## Canonical Error Message

`Error: you must switch to the {correct_app} app before executing this action. Use the
  switch_app action to switch apps.`

## Determining Current App
```

```
The current app is established by the LAST successfully-switched app in the trajectory
  (last obs starting with "Successfully switched to app: {app}").

## When to Emit

Emit wrong-app error when the agent's action JSON `"app"` field does NOT match the current
  app, AND the action is not a `system` action.
[...]
3. After wrong-app error, current app does NOT change
[...]
```

F.3 A Replay Cycle, End to End

We follow one cycle of an illustrative τ^2-bench run of replay-driven skill evolution that starts from a one-skill initial repertoire. Its first cycle adds eight skills to the format skill of S_0; MCCA then replays the resulting S_1 on the arbitration traces, and box (a) shows one step of that pass with the Comparator's discrepancy. The agent passed a list argument as a string, the environment answered with its own exception text, and S_1 predicted another error in the same form.

(a) A step of an arbitration trace replayed under S_1, and the Comparator's discrepancy

a_t	`[{"id": [...], "name": "modify_pending_order_items", "arguments": {"item_ids": "[\"6117189161\", \"9879255677\", \"7453605304\", \"3799046073\", \"9851293632\"]", [...]}}]`
o_{t+1}	`[tool] Error: [ not found`
$\hat{o}_{t+1}$	`[tool] Error: modify_pending_order_items() got an unexpected keyword argument 'item_ids'`
differs	The real observation reports a JSON parsing error ('Error: [not found'), indicating the tool failed to parse the stringified list arguments. [...]
behavior	The predicted error hallucinates a function signature issue ('unexpected keyword argument') that contradicts the successful schema usage implied by the trajectory context, whereas the real error correctly reflects a data formatting failure common when LLMs stringify JSON arguments.
rule	When predicting tool errors, ensure the error message matches the likely cause: if arguments are stringified JSON (e.g., "[\"id\"]"), predict a JSON parsing error, not a 'unexpected keyword argument' error. [...]

Cycle $k = 2$ then replays, under S_1, a batch $\mathcal{B}_2$ of $B = 200$ traces that SBS draws from the $n_{\text{sl}} = 1{,}000$ traces LPF keeps out of a pre-sample of $n_{\text{pre}} = 2{,}000$; the harness parses 199 of the 200 predictions. Box (b) gives the first three issues of the discrepancy summary F_2 in the Aggregator's words; their shares of 63%, 27% and 23% lie above the frequency threshold (Appendix B.1), so all three go to the Reviser. The revision $\mathcal{U}_2$ rewrites five skills and creates two (box (c)), and $S_2 = S_1 \oplus \mathcal{U}_2$ holds eleven. The created skill of box (d), which the Reviser wrote from F_2 without reading the arbitration pass, states the error behavior that box (a) shows S_1 getting wrong: a malformed argument yields a parsing error, a failed call is reported as plain text after the tag, and the message keeps the environment's wording. Boxes (a), (b) and (d) quote verbatim, [...] marks omitted text, and box (c) lists the edits of $\mathcal{U}_2$.

(b) Discrepancy summary F_2: first three root causes

```
63% of predictions exhibit severe
  entity/state hallucination where tool
  response[...]
27% of predictions show domain or intent
  drift where the simulated user [...]
23% of predictions fail conditional
  instruction following or persona
  constraints[...]
```

(c) Revision $\mathcal{U}_2$: edits

```
rewrite  tool-value-grounding
rewrite  tool-schema-fidelity
rewrite  user-info-reveal-discipline
rewrite  user-voice-realism
rewrite  policy-state-validation
create   tool-error-response-canonical
create   user-memory-of-prior-turns
```

(d) Created skill `tool-error-response-canonical`

```
---
name: tool-error-response-canonical
description: USE WHEN: generating [tool] responses for failed calls. Ensures the error
  type (syntax vs. logic) matches the input cause and uses exact environment phrasing.
---
[...]
## Syntax vs. Logic Distinction
- If the agent's input contains malformed JSON or invalid syntax, the predicted `[tool]`
  observation must return a **parsing/syntax error** message, not a logical validation
  error.
```

```
[...]
## Exact Error Phrasing
- Use the exact exception text or error message format defined in the environment's ground
  truth for the specific failure mode.
[...]
- Failed calls are always reported as plain text: `[tool] Error: <message>`.

## No JSON Error Objects
- Never wrap error messages in a JSON object (e.g., `{"error": "..."}`). The error message
  itself is the payload following the `[tool] Error:` tag.
```

G PROMPTS

Every role below is played by one model with no weight update, Qwen3.5-397B-A17B in the main rows: the simulator, the Comparator, Aggregator and Reviser of replay-driven skill evolution, and the HiSkill inducer and router. The simulator prompts serve every prediction of the loop, on ReplayBench and in trajectory synthesis, the three role prompts every replay cycle of Algorithm 1 (the Comparator and the Aggregator also run on the arbitration passes, whose outputs the Arbiter scores), and the HiSkill prompts run once per selected repertoire and once per task. HiSkill first summarizes groups of related skills into strategies, each with a trigger condition; at synthesis it routes a task to the closest strategies and injects their skills, which the prompts call operations. In the Reviser prompt, add, update and retire are the create, rewrite and delete edits of Section 3.2. The prompts call the simulator a world model (WM) and its skill repertoire the reference skills. A name in angle brackets, set in the color of its box, such as ⟨*task*⟩, marks a slot filled at run time; [...] marks omitted text, and tool names that carry a provider's name are shown generically. The rest is quoted from the prompts. Where a prompt has one instance per benchmark we show the τ^2-bench instance.

Simulator $\hat{P}_S$ on ReplayBench · reads the history, the action a_t and the skill index · writes $\hat{o}_{t+1}$

SYSTEM You are an environment simulator for an AI agent.
[...]
- Output ONLY the environment's response/observation, no explanations or commentary
[...]
 # Reference Skills
 # Role and Objective
You are a **Tool-Use World Model** — a precise simulator of tool-execution backends and, where the trajectory is conversational, of the user side of the dialogue. [...]
 # Format Requirements
Preserve the observation representation used in this trajectory exactly [...] Do not wrap outputs in markdown fences or add agent-side artifacts (`Thought:`, `Action:`, `<answer>`). Representation mismatch is a correctness error, not a stylistic preference.
Output exactly one complete and closed <predicted_observation></predicted_observation> block and nothing else.
Task the agent is pursuing: ⟨*task*⟩
 # External information tools (web search)
You can call `web_search(query)` to search the live web and `web_extract(urls)` to open a page found by a search. [...] Whatever you look up, your final answer must still be nothing but the predicted observation, in exactly the format the instructions above require
[...]
 # Local tools (shell and files)
[...] Whatever you compute, your final answer must still be nothing but the predicted observation in the required format.
⟨*format-consistency rules of the environment*⟩
Format Reference
⟨*format exemplar α_E*⟩
 # Skill Repertoire (progressive disclosure)
You have ⟨*number of skills*⟩ skills available. Each line below is `name — when to use it`. [...] REQUIRED workflow: (1) pick the 1-3 skills whose 'when to use' best matches this environment's observation format and the latest action, (2) call the tool `load_skill(name)` for each of them to read the full text [...] (3) only then output the next observation, following the loaded skills. Output ONLY the observation.
- ⟨*identifier*⟩ — ⟨*applicability condition*⟩
[...]
TOOLS {"name": "load_skill", "description": "Read the full text of one skill from the skill repertoire by its exact name.", [...]
USER Current Observation:
⟨*initial observation o_0*⟩
Agent Action: ⟨*action a_0*⟩
Next Observation:
ASSISTANT ⟨*recorded observation o_1*⟩

```
[...]
USER Agent Action: ⟨latest action a_t⟩
Next Observation:
```

Simulator $\hat{P}_S^{\text{inj}}$ at synthesis · reads every skill body and the routed strategies · writes $\hat{o}_{t+1}$

```
SYSTEM You are an environment simulator for an AI agent.
[...]
# Reference Skills
⟨format-consistency rules of the environment⟩
## HiSkill routing
## Strategy: ⟨strategy name⟩
When to use: ⟨trigger condition⟩
Operations:
  - ⟨identifier⟩: ⟨first 200 characters of its body⟩
[...]
[...]
## Observation Focus (current strategy: ⟨strategy name⟩)
[...]
## Reference Skills
## Skill: ⟨identifier⟩
⟨body⟩
[...]
## Format Reference
⟨format exemplar α_E⟩
```

Comparator · reads the task, the actions, a prediction and the recorded observation · writes a discrepancy δ_τ

```
You are evaluating a World Model's predicted observation against a
REAL observation from a customer-service conversation environment
(tau2-bench: airline / retail / telecom).
[...]
## Task Context
Query: ⟨task⟩
Agent's action at step ⟨t⟩: ⟨action⟩
Previous actions: ⟨earlier actions⟩
## Predicted Observation (from World Model)
⟨prediction⟩
## Real Observation (from actual tau2 environment)
⟨recorded observation⟩
## CRITICAL RULES
[...]
4. **The real observation is ground truth.** Only report semantic
   differences.
## Your Evaluation (TWO parts)
[...]
8. **Trajectory quality**: Overall, does this predicted observation
   help train a customer-service agent? Rate: high / medium / low
Return ONLY a JSON object (no markdown fences):
{
  "content_diff": "semantic content in real but not in predicted [...]
  "content_accuracy": "JSON schema + ID consistency + value fidelity
  assessment",
  "missing_in_predicted": "specific info present in real but absent in
  predicted",
  "behavioral_issues": "speaker-tag + policy-gate + ID-continuity +
  realistic-voice issues (or 'none')",
  "trajectory_quality": "high or medium or low",
  "improvement_rules": "1-3 rules to improve tool-schema fidelity /
  user-sim realism / policy compliance"
}
```

Aggregator · reads the discrepancies of a batch · writes the discrepancy summary F_k

```
You are analyzing a batch of ⟨n⟩ replay comparison results [...]
## Individual Feedbacks
--- Feedback ⟨i⟩ (quality=⟨quality⟩) ---
  content_diff: ⟨what differs⟩
  content_accuracy: ⟨accuracy of values⟩
  missing_in_predicted: ⟨what the prediction lacks⟩
  behavioral_issues: ⟨behavior misrepresented⟩
  improvement_rules: ⟨rule to fix it⟩
[...]
From these ⟨n⟩ comparisons, extract common patterns in TWO categories:
### A. BEHAVIORAL issues (trajectory logic, HIGHEST PRIORITY)
[...]
For each issue, report its frequency (fraction of comparisons). Only
report issues appearing in 20%+ of comparisons.
Return ONLY a JSON object (no markdown fences):
{
  "behavioral_issues": ["specific issue with frequency, [...]
  "attribute_issues": ["specific attribute issue with frequency"],
  "behavioral_suggestions": ["actionable rule to fix behavioral
  issue"],
  "attribute_suggestions": ["actionable rule for realism"],
  "issue_frequencies": {"policy_violation_refund": 0.40, "id_drift":
  0.22}
}
```

Reviser · reads the repertoire S_{k-1} and the discrepancy summary F_k · writes a revision $\mathcal{U}_k$

```
You are a skill optimization agent for a World Model that simulates
customer-service conversations [...]
## Current Skills
### Skill folder: ⟨folder⟩/
Name: ⟨identifier⟩
Description: ⟨applicability condition⟩
Content:
⟨body⟩
---
[...]
## Replay Batch Feedback (Cycle ⟨cycle⟩)
Batch size: ⟨number of discrepancies⟩ comparisons (⟨number rated high or medium⟩
valid)
### Issues Found:
- ⟨issue⟩
[...]
### Issue Frequencies:
- ⟨root cause⟩: ⟨share⟩
## CRITICAL RULES
[...]
7. **Skill structure**: YAML frontmatter (`name: kebab-case`,
   `description: USE WHEN: ...`), then `# Title`, then 2-4 `##`
   sections, bullet lists. 30-80 lines each.
Return ONLY a JSON object (no markdown fences):
{
  "updates": [
    {
      "action": "update" or "add",
      "skill_name":
      "<tool|user|policy|flow>_<category>_in_snake_case",
      "rationale": "link to specific feedback",
      "content": "---\nname: <kebab-case>\ndescription: USE WHEN:
      <trigger>. <purpose>\n---\n# <Title>\n\n## <Section>\n-
      <bullet>\n..."
    }
  ],
  "no_change_reason": "string" or null
}
```

```
## RETIRE
Besides "add" and "update" you may use "action": "retire" to take an
existing skill OUT of the repertoire.
- Use it only when real observations quoted in the feedback CONTRADICT
  what the skill says the environment does [...]
- A retire entry has: "skill_name" = the existing skill's name,
  "rationale" = one sentence on why it must go, "content" = the real
  observation text from the feedback that contradicts it, copied
  verbatim (one or more quotes). A retire without such a quote is
  ignored.
[...]
```

HiSkill · reads the selected repertoire, then a task · writes strategies, then the routed operations

SYSTEM You are summarizing a group of related skills into ONE strategy. Output ONLY a JSON object with exactly three string fields —— no markdown fences, no prose before or after:
{"name": "<3-8 word noun phrase>", "description": "<1 sentence, 15-30 words, what this strategy accomplishes>", "when_to_use": "<1 sentence, phrased 'When X, use this strategy to Y', 15-30 words>"}
[...]
USER Goals of skills in this group:
- ⟨*one-sentence summary of a skill*⟩
Output JSON:

SYSTEM You are a skill routing assistant. Given the user query, current observation, and a small list of candidate operations grouped by strategy, pick the ⟨*number of operations*⟩ most relevant operations to inject into the simulator's prompt. Output ONLY a JSON array of operation names, e.g. ["opA", "opC"]. No prose, no markdown fences, no preamble.
USER User query: ⟨*task*⟩
Current obs (truncated): ⟨*first observation*⟩
Candidate operations (across ⟨*number of strategies*⟩ strategies):
Strategy '⟨*strategy name*⟩' (cos=⟨*similarity*⟩):
  When to use: ⟨*trigger condition*⟩
  Operations:
  - ⟨*identifier*⟩: ⟨*first 200 characters of its body*⟩
[...]
Return JSON list of top-⟨*number of operations*⟩ operation names: