EAU: ENVIRONMENT-AGENT-USER CO-EVOLUTION FOR TRAINING USER-CENTRIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training LLM agents to interact with users over multiple turns is challenging because the learning signal depends not only on the agent, but also on the user simulator and environment that generate its experience. Simulated users can be stochastic or inconsistent, while environments may contain unreliable tools, incomplete state coverage, and incorrect verifiers. Prior work mainly addresses the resulting instability through reward shaping or denser supervision, while treating these components as fixed. We instead introduce **EAU**, a meta-agent framework for **e**nvironment–**a**gent–**u**ser co-evolution. During training, **EAU** monitors trajectories, tool traces, rewards, and training statistics; diagnoses failures and uninformative interactions; and proposes targeted modifications to the user simulator, environment, reward design, and RL configuration. Validated changes are hot-swapped into the ongoing run, allowing the training setup to evolve together with the policy. These adaptations also persist across runs, so later training can build on previously discovered improvements rather than restarting from a fixed setup. We enable this process through an editable symbolic representation of user and environment dynamics, while the agent remains a neural LLM optimized with RL. Across three interactive-agent benchmarks and two backbone models, **EAU** consistently outperforms prior RL methods, improving the overall score over the strongest baseline by 13.1–15.0% and increasing successful task completion on ColBench by up to 36.7%, showing that co-evolving the training system with the policy can substantially improve multi-turn RL.

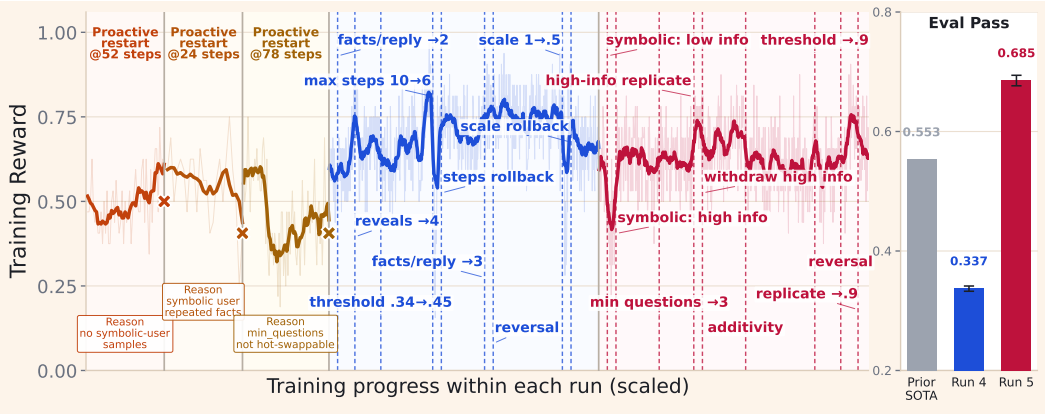

Figure 1: Illustration of how our meta-agent, **EAU**, manages RL training runs. **EAU** surpasses the prior SOTA on ColBench. **Left:** For the first three runs, **EAU** proactively restarts when it identifies issues that can only be resolved with offline patches. **Middle:** Run 4 and 5 displayed in blue and red are two full runs. Dashed lines indicate online, intra-run modifications. Run 4 explored individual modifications through local perturbations, whereas Run 5 used endpoint interventions followed by replication, reversal, and interaction tests.

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.

1 INTRODUCTION

Large language models are increasingly deployed as agents that collaborate with humans rather than simply execute fully specified tasks. As shown in Figure 2, in many realistic applications, users can provide incomplete or evolving specifications, requiring an agent to interact over multiple turns: asking clarifying questions, resolving ambiguities, negotiating constraints, and adapting its behavior as new information emerges. This has motivated a growing body of interactive benchmarks in which agents communicate with simulated users while acting in an external environment (Yao et al., 2024; Barres et al., 2025; Qian et al., 2025a). Such simulation makes large-scale evaluation and reinforcement learning (RL) possible, but it also introduces a fundamental challenge: *the learning signal is only as reliable as the interaction that produces it*. LLM-based user simulators are stochastic and can behave inconsistently, for example by drifting from their assigned persona, revealing information prematurely, or changing their intent across otherwise similar episodes. At the same time, environments may contain incomplete state coverage, unreliable tool behavior, unreachable tasks, or incorrect verifiers. Moreover, because both components are typically fixed throughout training, the interactions they generate need not remain informative as the policy evolves. Together, these issues can make rewards highly variable or systematically misleading, substantially destabilizing RL for interactive agents.

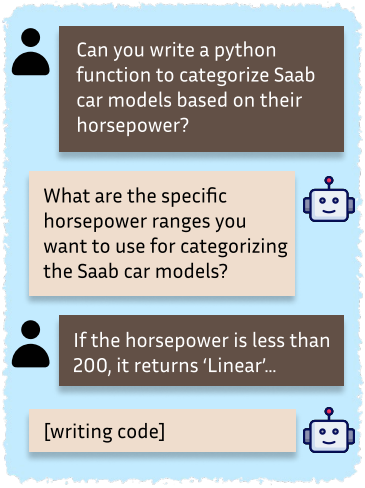

Figure 2: Illustration of a user-centric agentic task. An agent needs to ask for clarifications about a user query.

Prior work has largely addressed this instability by improving the supervision presented to the policy, for example through reward shaping that credits partial progress (Qian et al., 2025b) or auxiliary supervision that provides denser learning signals (Kong et al., 2026; Wang et al., 2026). These approaches can make better use of a noisy signal, but they leave the sources of that signal—the user simulator and the environment—largely unchanged. We argue that this misses an important root cause of instability. If a tool silently fails for a particular argument, a verifier incorrectly rejects a successful trajectory, or a simulated user reveals information that the agent was supposed to elicit, modifying the downstream reward cannot recover the intended interaction. More generally, as the policy improves, a fixed simulator and environment may continue producing trajectories that are redundant, uninformative, or poorly targeted to the policy's current weaknesses. We therefore propose shifting part of the intervention upstream: rather than treating the user simulator and environment as fixed infrastructure and optimizing only the policy's learning signal, we continually diagnose and revise the interaction process itself during training.

To realize this idea, we introduce **EAU**, a meta-agent that continuously supervises RL training and improves the components that generate its learning experience. Throughout training, **EAU** observes rollout transcripts, tool traces, rewards, and aggregate training statistics, and diagnoses whether problematic or uninformative trajectories arise from the policy, the user simulator, the environment, or the training configuration. As shown in Figure 1, it proposes targeted interventions based on this diagnosis, including correcting environment or verifier failures, refining the simulator's persona and information-disclosure behavior, and adjusting reward terms or training hyperparameters. Importantly, these interventions are not limited to fixing errors. As the policy improves, **EAU** can also adapt the simulator and environment to expose unresolved weaknesses and generate more informative training trajectories. Candidate modifications are validated by a bounded integrity gate before being accepted and hot-swapped into the ongoing training loop, allowing the training system itself to evolve without repeatedly restarting from scratch.

This produces a different view of interactive-agent training: rather than optimizing a policy against a fixed user and environment, **EAU** enables the environment, agent, and user to co-evolve over the course of training. The user simulator becomes more consistent and appropriately challenging, the environment becomes more reliable and informative, and the policy learns from the improved interactions that result. To prevent this adaptation from artificially inflating performance by making

the task easier, we constrain modifications using a frozen, human-verified evaluation set and retain a versioned record of accepted changes.

We evaluate **EAU** on three interactive-agent benchmarks and two backbone models. Across these settings, **EAU** discovers interpretable modifications to both simulators and environments and improves final agent performance by up to 36.7% over the previous state of the art, while our analysis shows how the three components evolve together throughout training.

To summarize, our contributions are:

- We identify fixed user simulators and environments as a key bottleneck in interactive RL.
- We introduce **EAU**, a meta-agent that diagnoses and updates the training system online.
- We show that environment–agent–user co-evolution yields strong improvements across benchmarks and models.

2 Related Work

User Centric Agentic Reinforcement Learning. User-centric tasks (Yao et al., 2024; Barres et al., 2025; Qian et al., 2025a) focus on interacting and fulfilling the requirement of a simulated user under multi-turn conversations. Different from previous multi-turn Reinforcement Learning (Jimenez et al., 2024; Xi et al., 2025; Zhang et al., 2026b) that solely interacts with a fixed deterministic environment, user-centric RL (Qian et al., 2025b; Kong et al., 2026; Yu et al., 2026) poses an additional challenge: the introduction of LLM user simulator may cause several issues such as stochastic state transition, inconsistent behaviors and collapse in training due to overfitting a fixed behavior (Yu et al., 2026). Recent methods mainly focus on improving supervision signal presented to the policy (Zhou et al., 2025; Kong et al., 2026). Yu et al. (2026) co-train the user to avoid overfitting. Our work identifies that issues may lie in all three parts, i.e., RL training dynamics, user simulation and environment, and we endeavor to address all at once with a meta agent, **EAU**.

Meta-Agents and Automated Optimization of Training Components. A growing body of work uses one LLM to optimize another system or the system surrounds itself (Hu et al., 2025; Lee et al., 2026; Zhang et al., 2026a; Liu et al., 2026; Chen et al., 2026b; Surina et al., 2025). **EAU** differs mainly in that the meta-agent acts *within* a run: modifications are validated and hot-swapped into a live training loop rather than applied between runs or before training begins.

Co-evolution of Agent and its World. More recently, there's a trend in improving the world, e.g., environment and user simulator, that an agent is interacting with. While most of them prepare the worlds before training (Huang et al., 2026; Tu et al., 2026), some begin to discover how the world can evolve along with the agent training (Yu et al., 2026; Kim et al., 2026; Yang et al., 2026). Unlike prior methods that evolve the world through predefined update rules or workflows, **EAU** can flexibly determine what to modify, when to intervene, and how to adapt the training system online.

3 Problem Setup

We focus on user-centric agentic tasks, where an agent interacts with both an environment through tool calls and a (simulated) user over multiple turns. The goal is to accomplish a task while satisfying the user's intent, which may be underspecified at the outset and gradually refined through the interaction.

We formulate each task in our setting as a partially observable Markov decision process (POMDP). Formally, the environment for a task is $\mathcal{E} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{O})$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$, observation space $\mathcal{Z}$, and a function $\mathcal{O} : \mathcal{S} \to \Delta(\mathcal{Z})$ that maps states to observations. Δ here refers to the stochasticity in transitions and observations. We consider each task to have a simulated user as a part of the environment, therefore an agent needs to interact with the environment by tool calling and the user simulator by conversations, i.e., $\mathcal{S} = \mathcal{S}_{\text{tool}} \otimes \mathcal{S}_{\text{user}} \otimes \mathcal{S}_{\text{query}}$, where $\mathcal{S}_{\text{query}}$ can be considered as the first state. And we have $\mathcal{A} = \mathcal{A}_{\text{tool}} \cup \mathcal{A}_{\text{user}}$, $\mathcal{Z} = \mathcal{Z}_{\text{tool}} \cup \mathcal{Z}_{\text{user}}$. Specifically, for transition $\mathcal{T}$, interactions with the environment through tool calls are deterministic, while interactions with simulated users are stochastic. Thus, we have $\mathcal{T}_{\text{tool}} : (s_{\text{tool}}, a_{\text{tool}}) \mapsto (s'_{\text{tool}}, z_{\text{tool}})$ and $\mathcal{T}_{\text{user}} : (s_{\text{user}}, a_{\text{user}}) \to (\mathcal{S}_{\text{user}}, \mathcal{Z}_{\text{user}})$.

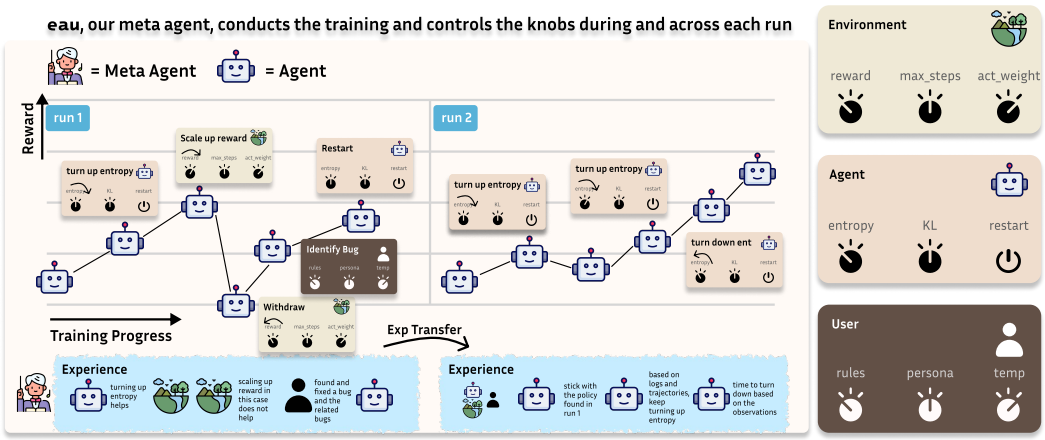

Figure 3: Illustration of how **EAU**, our meta agent, explores, learns, and absorbs experience online during and across each run. Reading logs and trajectories, **EAU** explores different strategies online by controlling the *knobs* of each module, learns which strategies help and which do not. Experience is brought across different runs, serving as a solid foundation for continual exploration and learning.

Let π_θ be the policy (agent), our goal is to maximize the number of successful tasks, i.e., $\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} R(s_t,a_t)\,\middle|\,\text{query}\right]$. To achieve this, a policy needs to first interact, negotiate, and even persuade the user, while at the same time understanding users' intents and calling the correct tools. Thus, we term it a bi-level optimization:

$$\max_{\theta}\ \mathbb{E}_{\tau\sim\pi_\theta}\left[R_{\text{task}}(\tau,u)\right] \qquad \text{s.t.} \qquad \pi_\theta = \arg\max_{\pi}\ \mathbb{E}_{\tau'\sim\pi}\left[\sum_{t=0}^{T-1}\gamma^t r_{\text{conv}}(s_t,a_t)\right], \tag{1}$$

where R_{task} refers to rewards related to task success, and r_{conv} refers to rewards related to conversations specifically designed for each environment. Notably, R_{task} is usually verifiable yet sparse while r_{conv} can be dense but noisy.

4 Method: Environment-Agent-User Co-Evolution

In this section, we introduce **EAU** (Environment-Agent-User Co-Evolution) for user-centric agentic RL. Recall that agentic RL with a simulated user leads several challenges, as 1) sparse reward with stochastic user simulator and environments may cause the agent hard to learn, and 2) user simulator may not be reliable, e.g., providing unreliable reward, distracting from the instruction and leaking answer in the conversation.

4.1 Method Overview

To address these challenges, the core idea of our method is to enable the user simulator, environment, and agent to evolve synergistically throughout training. The agent is updated at every gradient step, while the user simulator and environment are adapted on demand in response to the agent's evolving needs. This co-evolution progressively improves agent-user and agent-environment interactions, leading to more stable and effective user-centric agentic RL training.

To realize this co-evolution, we introduce a meta-agent that oversees the entire RL training process, as opposed to relying on careful manual expert intervention which is labor-intensive and hard to scale. As illustrated in Figure 3, the meta-agent treats tunable aspects of the user simulator, environment, and agent as *knobs*. It continuously monitors training dynamics and adjusts these knobs when necessary to improve the training process. In the later of this section, we will describe how the meta-agent drives the evolution of each of the three components: the User, Environment, and Agent.

4.2 Evolving User

LLM-simulated users are inherently stochastic and may imperfectly follow its specification, for example by revealing information that should remain hidden or by improvising answers to questions unsupported by the task. To provide a more controllable signal, we introduce a *symbolic user simulator*, u_{sym}, which consists of a set of explicit rules that deterministically map interaction states, obtained from parsed agent outputs, to user responses. Figure 4 illustrates an example of such a rule. These rules provide precise control over what information is revealed and under which conditions. However, designing and updating them manually throughout training is intricate and labor-intensive. We therefore expose the symbolic rules as tunable *knobs*, allowing the meta-agent to revise them in response to the agent's evolving behavior.

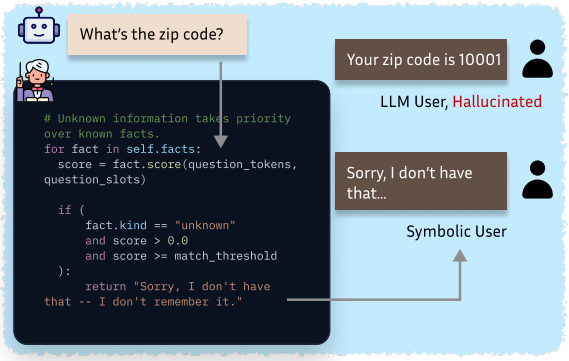

Figure 4: Illustration of symbolic user simulator.

The symbolic and LLM simulators offer complementary strengths: the former provides controllability and determinism, while the latter offers richer and more natural interactions. We combine them through an ϵ-symbolic strategy:

$$u_\epsilon := \begin{cases} u_{\text{sym}}, & \text{with probability } \epsilon, \\ u_{\text{LLM}}, & \text{with probability } 1-\epsilon, \end{cases} \qquad \epsilon \in [0,1]. \tag{2}$$

The mixing coefficient ϵ thus controls the balance between structured, deterministic supervision and flexible LLM-based interaction, and can itself be adapted by the meta-agent during training.

Beyond the symbolic rules and mixture coefficient ϵ, **EAU** can also evolve the LLM user by tuning controllable generation parameters, including temperature and persona-specific prompt instructions. Together, these mechanisms allow the user side of the training loop to adapt alongside the policy rather than remaining fixed throughout training.

4.3 Evolving Environment

To evolve the environment $\mathcal{E}$, **EAU** primarily adapts intermediate reward signals of $\mathcal{R}$. Recall that user-centric agentic tasks typically involve multi-turn interactions, where verifiable rewards are often sparse, such as task success. As a result, the agent may struggle to explore effectively when it rarely reaches successful outcomes. When such behavior is observed, **EAU** can introduce intermediate reward rules that identify and reward states indicative of progress toward task completion. An illustrative example is shown in Figure 5.

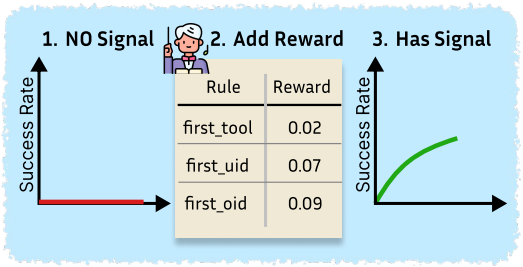

Figure 5: Illustration of environment evolution by introducing additional rewards.

EAU can also revise existing manually designed rewards when they induce undesirable behaviors or divert the agent from the final objective. By dynamically adding and modifying intermediate rewards, **EAU** provides more informative learning signals while also accumulating experience about which reward designs facilitate or hinder learning. This experience can in turn guide future environment evolution.

4.4 EVOLVING AGENT

The agent naturally evolves through gradient updates during RL training. We keep the underlying *training algorithm fixed* and let the meta-agent continuously monitor the optimization process and intervene in its configurable training dynamics when necessary.

Training Algorithm. We use and fix Proximal Policy Optimization (Schulman et al., 2017, PPO) as RL algorithm. Recent work on LLM RL mainly use critic-free methods, such as GRPO (Shao et al., 2024), for low training cost and unbiased advantage estimation. However, current critic-free methods suffer from the sparse credit assignment, where intermediate rewards are accumulated to a single trajectory-level reward (Qian et al., 2025b), or are not applicable to user-centric tasks where the same states are hard to replicate for group advantage estimation (Feng et al., 2026).

We therefore use a learned critic V_ϕ, parameterized by ϕ, with Generalized Advantage Estimation (GAE). The critic is optimized by minimizing the value error $\mathcal{L}_\phi^{\mathrm{VF}} = \mathbb{E}[(V_\phi(q, y_{<t}) - R)^2]$, where R is the cumulative reward. Both critic and policy are restricted to action tokens: the critic is evaluated only where the policy acts, and GAE is computed across actions rather than across all tokens, so the advantage of a turn is not diluted by the observation text surrounding it (Hou et al., 2026). Formally, the advantage $\hat{A}_t^{\mathrm{GAE}}$ is computed as:

$$\hat{A}_t^{\mathrm{GAE}} = \sum_{l=0}^{|y|-t-1} (\gamma\lambda)^l \delta_{t+l}, \tag{3}$$

where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, and s_t refers to the t-th action token.

Monitoring Training During training, the meta agent is allowed to tune the hyper-parameters such as entropy coefficient and KL coefficient, based on training logs, including metrics like estimated rollout entropy and log likelihood. It can also read the rollout trajectories and the interactions with the environment and the user.

4.5 INTRA- AND INTER-RUN INTERVENTION

A notable design choice of **EAU** is to support flexible intervention at two timescales: both *intra-run* and *inter-run*. As illustrated in Figure 3, for intra-run intervention, updates can be hot-swapped into the training loop without restarting from scratch. For example, **EAU** can revise symbolic-user rules, add or modify environmental rewards, and adjust RL hyperparameters during training.

It can also decide when a restart (inter-run) is necessary, such as after an irreversible failure, or resume from an intermediate checkpoint. Inter-run intervention, in contrast, enables **EAU** to consolidate observations from completed runs and use them to revise the setup for subsequent training.

Unlike previous paradigm that only allows inter-run modifications (Chen et al., 2026a; Xia et al., 2026; Liu et al., 2024; Yamada et al., 2025; Ferreira et al., 2026), by granting this flexibility, **EAU** can explore different operations on-the-fly, which is more efficient than inter-run interventions alone. Therefore, it can find out more effective operations based on the intra- and inter-run experience.

4.6 INSTANTIATION OF META-AGENT

We instantiate our meta agent with coding agents, such as Claude Code and Codex, following recent works on meta agent (Lee et al., 2026). The meta agent has access to a local filesystem which includes the source code, training logs, etc. It reads the logs with a fixed interval during training, and decides which part to modify for environment-user-agent evolution. A list of *knobs* to tune is prepared at the beginning, but **EAU** can also modify the list if necessary. It is also asked to maintain a experience pool as a directory of documents in the filesystem. Before each modification, it may reference to the experience pool and whenever there's a new finding, it updates the pool accordingly.

Unlike previous methods that allow the meta agent to collect feedback on a evaluation/test set (Panfilov et al., 2026; Lee et al., 2026; Xia et al., 2026), the only information visible for **EAU** is the logs. All described above are implemented as a skill of instructions along with necessary scripts.

Table 1: Comparison across UserGym and ColBench on *Qwen3-4B* and *Qwen2.5-7B-Instruct*. The best and second-best scores in each category are highlighted in **bold** and underlined respectively. Standard deviations across 3 repeated evaluations are shown in the subscripts.

Type	Method	UserGym								ColBench			Overall
		ID					*OOD*		Avg.	Pass	Succ.	Avg.	
		Travel	Func.	Pers.	Tau	Turtle	Inten.	Tele.					
Closed-Source Models													
Prompting	Gemini-3-Flash	0.574	0.423	0.695	0.167	0.153	1.718	0.829	0.651	0.515	0.382	0.449	0.606
Prompting	GPT-4.1	0.554	0.051	0.599	0.109	0.267	1.867	0.732	0.597	0.529	0.403	0.466	0.568
Prompting	GPT-4o-mini	0.596	0.089	0.683	0.091	0.117	1.277	0.512	0.481	0.463	0.342	0.403	0.463
Qwen3-4B													
Prompting	Qwen3	0.277	0.026	0.452	0.006	0.071	1.660	0.488	0.426	0.272	0.153	0.213	0.378
Prompting	ReAct	0.279	0.053	0.476	0.005	0.147	1.782	0.454	0.457	0.269	0.145	0.207	0.401
Prompting	Reflexion	0.291	0.045	0.474	0.035	0.124	1.717	0.501	0.455	0.303	0.184	0.244	0.408
RL Training	RAGEN	0.477	0.104	0.511	0.006	0.117	1.572	0.488	0.468	0.479	0.361	0.420	0.457
RL Training	Search-R1	0.482	0.103	0.425	0.059	0.113	1.711	0.412	0.472	0.467	0.355	0.411	0.459
RL Training	UserRL	0.538	0.095	0.507	0.053	0.121	1.732	0.520	0.509	0.468	0.342	0.405	0.486
RL Training	InfoPO	0.589	0.115	0.556	0.097	0.154	1.862	**0.542**	0.559	0.553	0.439	0.496	0.545
RL Training	PPO	0.468	0.051	0.560	0.006	0.180	1.880	0.512	0.525	$0.284_{\pm 0.046}$	$0.200_{\pm 0.044}$	0.242	0.462
RL Training	**Ours**	**0.615**	**0.186**	**0.726**	**0.109**	**0.188**	**2.023**	0.512	**0.622**	$\mathbf{0.685_{\pm 0.009}}$	$\mathbf{0.600_{\pm 0.015}}$	**0.643**	**0.627**
Qwen2.5-7B-Instruct													
Prompting	Qwen2.5	0.441	0.026	0.289	0.000	0.062	1.254	0.292	0.338	0.242	0.140	0.191	0.305
Prompting	ReAct	0.452	0.064	0.325	0.037	0.078	1.378	0.316	0.379	0.238	0.135	0.187	0.336
Prompting	Reflexion	0.445	0.052	0.312	0.022	0.074	1.320	0.305	0.361	0.276	0.168	0.222	0.330
RL Training	RAGEN	0.538	0.124	0.548	0.000	0.148	1.815	0.416	0.513	0.449	0.348	0.399	0.487
RL Training	Search-R1	0.565	0.113	0.412	0.043	0.154	1.805	0.416	0.501	0.457	0.352	0.405	0.480
RL Training	UserRL	0.546	0.115	0.444	0.048	0.152	1.826	0.424	0.508	0.436	0.327	0.382	0.480
RL Training	InfoPO	0.588	0.167	0.535	**0.091**	**0.178**	1.892	0.488	0.563	0.534	0.426	0.480	0.544
RL Training	PPO	**0.596**	0.229	0.651	0.018	0.135	1.860	0.488	0.568	$0.356_{\pm 0.012}$	$0.240_{\pm 0.000}$	0.298	0.508
RL Training	**Ours**	0.595	**0.333**	**0.730**	0.048	0.170	**2.035**	**0.512**	**0.620**	$\mathbf{0.659_{\pm 0.017}}$	$\mathbf{0.533_{\pm 0.018}}$	**0.596**	**0.615**

5 EXPERIMENT

5.1 EVALUATION PROTOCOL

To evaluate **EAU**, we consider two policy models, *Qwen3-4B* (Yang et al., 2025) and *Qwen2.5-7B-Instruct* (Qwen et al., 2025) and three benchmarks. For fair comparison, we strictly follow the evaluation in Kong et al. (2026). Details about evaluations are provided below. Notably, as mentioned above, we *do not treat evaluation results as part of the feedback* for **EAU**.

Benchmarks. We evaluate **EAU** on three representative user-centric benchmarks, i.e., **ColBench** (Zhou et al., 2025), **UserGym** (Qian et al., 2025b), and τ^2**-bench** (Barres et al., 2025). Each task requires an agent to uncover hidden information through conversation and tool calls in order to fulfill a user's request. Details about each benchmark are deferred to Appendix B.

Baselines. We compare against two families: prompting methods (ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023)) and RL training methods (RAGEN (Wang et al., 2025b), Search-R1 (Jin et al., 2025), UserRL (Qian et al., 2025b), InfoPO (Kong et al., 2026)). We also compare with vanilla PPO. For UserGym and ColBench, except PPO, baseline numbers are taken from Kong et al. (2026).

User Simulator. We default to *CoSER* (Wang et al., 2025a), a *Llama-3.1-8B* fine-tuned model, based on the competitive performance in user simulation shown in Zhou et al. (2026). The simulator is not fixed, however: **EAU** may replace it when the training logs justify doing so. For evaluation, we follow Kong et al. (2026) to use `GPT-4o-mini`.

5.2 IMPLEMENTATION DETAILS

Backbone of EAU. We use `Claude Code` with *Claude Opus-5* as the backbone if not otherwise addressed. A comparison between different meta agent backbones can be found in Table 5.

Hyper-Parameters. We match the compute budget of Kong et al. (2026). An episode is capped at 16 turns on UserGym, 10 on ColBench and 50 on τ^2-bench, with a per-turn generation budget of 1024 tokens, and a total rollout budget of 8192, 8192 and 16384 tokens respectively. For each experiment, we cap the maximum number of inter-run iteration at 5, including the active restart of experiments. Remaining optimization settings are given in Appendix C.

5.3 Main Results

As shown in Table 1, **EAU** consistently outperforms existing RL training methods across both benchmarks and both backbone models. Compared with the strongest baseline, InfoPO, **EAU** improves the overall score from 0.545 to **0.627** on *Qwen3-4B* and from 0.544 to **0.615** on *Qwen2.5-7B-Instruct*, corresponding to relative gains of 15.0% and 13.1%, respectively. Notably, with only a 4B backbone, **EAU** also achieves a higher overall score than all evaluated closed-source models, including Gemini-3-Flash (0.606) and GPT-4.1 (0.568). These results indicate that the gains are not specific to a particular model or benchmark, but transfer consistently across model scales and interaction settings.

Table 2: Results on τ^2-bench.

Method	Retail	Airline	Telecom	Avg.
UserRL	16.9	7.5	13.8	13.8
InfoPO	**24.4**	10.0	15.6	18.0
PPO	16.3	23.8	18.1	18.5
EAU	18.1	**28.8**	**21.3**	**21.5**

UserGym. Inference-time scaffolding such as ReAct and Reflexion provides only limited improvements over the base model, suggesting that prompting alone is insufficient to induce robust user-interaction behavior. RL training yields substantially larger gains, with interaction-aware methods such as InfoPO outperforming more generic RL baselines. **EAU** further improves over InfoPO, raising the average UserGym score from 0.559 to **0.622** on *Qwen3-4B* (+11.3%) and from 0.563 to **0.620** on *Qwen2.5-7B-Instruct* (+10.1%). Importantly, the improvement is broad rather than driven by a single environment. On *Qwen3-4B*, **EAU** improves over the strongest baselines on six of the seven UserGym environments, with particularly large gains on Func., Pers., and Inten. The latter is especially notable because Inten. and Tele. are held out during training and therefore directly test cross-environment generalization. On Inten., **EAU** reaches **2.023**, substantially outperforming InfoPO (1.862) as well as all evaluated closed-source models. On Tele., performance remains competitive, although InfoPO performs slightly better (0.542 vs. 0.512). Overall, these results suggest that the interaction strategies acquired during training transfer beyond the environments on which they were learned.

Table 3: Ablation on the user simulator. Results are on ColBench.

Variant	Pass mean	Succ. mean
LLM	$0.284_{\pm 0.046}$	$0.200_{\pm 0.044}$
Symbolic	$0.229_{\pm 0.004}$	$0.177_{\pm 0.012}$
Hybrid	$0.294_{\pm \mathbf{0.019}}$	$0.213_{\pm \mathbf{0.032}}$

ColBench. The gains transfer similarly to a distinct interaction benchmark. On *Qwen3-4B*, **EAU** improves over InfoPO from 0.553 to **0.685** on Pass (+23.9%) and from 0.439 to **0.600** on Succ. (+36.7%), increasing the ColBench average from 0.496 to **0.643** (+29.6%). The same trend holds for *Qwen2.5-7B-Instruct*, where Pass improves from 0.534 to **0.659** (+23.4%) and Succ. from 0.426 to **0.533** (+25.1%). The consistently larger improvements on the stricter success metric show that **EAU** does more than increase partial task completion: it substantially increases the fraction of interactions that are carried through to successful completion.

τ^2-bench. Table 2 compares **EAU** with UserRL, InfoPO, and PPO. **EAU** consistently outperforms prior state-of-the-art methods, achieving a 16.2% relative improvement over the strongest baseline.

5.4 Ablations & Analysis

To better understand (1) the effect of the symbolic user simulator in **EAU**, (2) how our design benefits the meta agent, (3) whether **EAU** is applicable to different training algorithms and (4) how backbone model affects the performance of **EAU**, we conduct the following ablations and analyses.

Table 4: Comparison of PPO without meta-agent, naïve meta-agent without **EAU**, and **EAU**.

Model	Method	UserGym	ColBench	
		Avg	Pass	Succ.
Qwen3-4B				
RL	PPO	0.525	$0.284_{\pm 0.046}$	$0.200_{\pm 0.044}$
RL	Meta-Agent	0.574	$0.317_{\pm 0.038}$	$0.207_{\pm 0.035}$
RL	**Ours**	**0.622**	$\mathbf{0.685_{\pm 0.009}}$	$\mathbf{0.600_{\pm 0.015}}$
Qwen2.5-7B-Instruct				
RL	PPO	0.568	$0.356_{\pm 0.012}$	$0.240_{\pm 0.000}$
RL	Meta-Agent	0.512	$0.415_{\pm 0.007}$	$0.277_{\pm 0.006}$
RL	**Ours**	**0.620**	$\mathbf{0.659_{\pm 0.017}}$	$\mathbf{0.533_{\pm 0.018}}$

Table 5: Ablations on training algorithm and meta-agent backbone. Results are on τ^2-bench.

Method	Retail	Airline	Telecom	Avg.
UserRL	16.9	7.5	13.8	13.8
+ **EAU**	**22.5**	**15.0**	**21.9**	**20.8**
InfoPO	**24.4**	10.0	15.6	18.0
+ **EAU**	20.0	**16.3**	**24.4**	**21.0**
+ **EAU** (Codex)	20.0	13.8	20.0	18.8

Ablation on symbolic user. Table 3 shows the results of changing the user simulator. For Hybrid, we set the mixing coefficient to $\epsilon = 0.5$. Although the symbolic user alone achieves the lowest performance, mixing it with an LLM user outperforms using the LLM user alone. We attribute this to the complementary behaviors of them: the LLM user provides a more diverse interaction distribution, while the symbolic user offers more reliable but narrower responses. Their combination therefore balances diversity and reliability, resulting in more effective training signals for the agent.

Ablation on EAU design. Table 4 shows the ablation on our design. By removing the symbolic user and the skill that teaches intra-run operations over the user, environment, and agent, we obtain a naïve meta-agent with only a generic instruction to improve RL training. Results show that without proper design, the naïve meta-agent may even lead to worse performance, and only improve PPO with a small margin compared to **EAU**.

Analysis: EAU is agnostic to the training algorithm and meta-agent backbone. As shown in Table 5, **EAU** improves both UserRL and InfoPO, suggesting that its gains are not tied to a specific training algorithm. Replacing the default meta-agent with Codex (GPT-5.6-Sol) also yields improvements, though smaller ones, indicating that **EAU** is broadly applicable while still benefiting from a stronger coding agent.

Analysis: Intra-run modification enables better exploration and exploitation for meta agent An important design choice of **EAU** is to allow intra-run modifications in addition to inter-run modification. To understand the benefit of this, we provide a qualitative view of (1) how intra-run produces experience efficiently and (2) how intra-run exploits previous experience effectively.

A key benefit of intra-run modification is its flexibility in allocating training budget. As shown in Figures 1 and 7, **EAU** proactively restarts runs once severe issues are detected, avoiding wasted training and generating useful experience more efficiently. On ColBench, **EAU** restarts early in its first three runs and already surpasses the previous SOTA by the second valid run, while the naïve inter-run meta-agent improves much more slowly. Moreover, **EAU** explores multiple modifications within run 4, producing richer feedback for run 5 to exploit. Figure 8 further isolates this effect: given the same first four runs of experience, enabling intra-run modification at run 5 leads to a stronger improvement than inter-run adaptation alone.

6 Conclusion

We introduced **EAU**, a meta-agent framework that jointly evolves the environment, agent, and user during training. We show that a frontier coding agent can effectively manage and adapt these components based solely on training feedback, including trajectories and optimization logs. To provide finer control over user behavior, we introduce a symbolic user simulator that the meta-agent can modify programmatically. We further enable intra-run interventions, allowing **EAU** to explore and exploit training modifications more flexibly. Across three user-centric benchmarks, **EAU** improves upon prior state-of-the-art performance by up to 36.7%. This highlights the value of adapting the training system itself. Future work may investigate post-training open-source models to perform such co-evolution autonomously.

Ethics Statement

Our work studies reinforcement learning for user-centric agents using simulated users and benchmark environments. All experiments are conducted with publicly available models and benchmarks and involve only simulated or mocked users; we do not collect or use data from real human subjects or personally identifiable information. As our framework allows a meta-agent to modify components of the training system, we additionally constrain such modifications to the training process and evaluate the resulting agents on a frozen evaluation set, reducing the risk of artificially improving performance through direct adaptation to evaluation instances.

Reproducibility Statement

We provide the main implementation and experimental details necessary to reproduce our results. Section 4 describes the EAU framework, including the symbolic user simulator, environment evolution, agent optimization, intra- and inter-run interventions, and the instantiation of the meta-agent. Section 5 specifies the evaluated benchmarks, policy models, user simulators, baselines, training budgets, and evaluation protocol. Additional details on the tunable knobs and benchmark-specific configurations are provided in the appendix. We also report repeated evaluations with standard deviations where applicable.

AI Use Statement

Generative AI systems are an integral part of the method studied in this work. In particular, we instantiate the EAU meta-agent using frontier coding agents, including Claude Code and Codex, which inspect training logs and trajectories and propose modifications to the training system. LLMs are also used as policy models, user simulators, and evaluation components as described in Section 5. All experimental design, analysis, interpretation of results, and final claims remain the responsibility of the authors. As for the research itself, we only use AI to help polish the writing of the paper.

References

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

Guhong Chen, Yingcheng Shi, Yongbin Li, Binhua Li, Xander Xu, Hu Wei, Shiwen Ni, Min Yang, and Jieping Ye. Evotrainer: Co-evolving llm policies and training harnesses for autonomous agentic reinforcement learning. *arXiv preprint arXiv:2606.03108*, 2026a.

Xiaoyin Chen, Canwen Xu, Yite Wang, Boyi Liu, Zhewei Yao, and Yuxiong He. Learning to self-evolve, 2026b. URL https://arxiv.org/abs/2603.18620.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, Yifei Ming, Jiaqi Pan, Bhavana Dalvi Mishra, Jiaxin Huang, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. Envharness: Awakening static worlds for agent learning, 2026. URL https://arxiv.org/abs/2608.19880.

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

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

Minseon Kim, Zhengyan Shi, Emiliano Penaloza, Christopher Cui, Roger Creus Castanyer, Maryam Hashemzadeh, Isadora White, Jonathan Light, Jeonghye Kim, Matheus Pereira, Darya Moldavskaya, Chinmay Singh, Fabio Vera, Baolin Peng, Xingdi Yuan, Marc-Alexandre Côté, and Alessandro Sordoni. Frognano: Training a 4b coding agent via online task synthesis, 2026. URL https://arxiv.org/abs/2609.07925.

Fanqi Kong, Jiayi Zhang, Mingyi Deng, Chenglin Wu, Yuyu Luo, and Bang Liu. Infopo: Information-driven policy optimization for user-centric agents, 2026. URL https://arxiv.org/abs/2603.00656.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization. *arXiv preprint arXiv:2402.01881*, 2024.

Zewen Liu, Zhan Shi, Yisi Sang, Bing He, Minhua Lin, Tianxin Wei, Dakuo Wang, Benoit Dumoulin, Wei Jin, and Hanqing Lu. Adaptive auto-harness: Sustained self-improvement for agentic system deployment on open-ended task streams, 2026. URL https://arxiv.org/abs/2606.01770.

Alexander Panfilov, Peter Romov, Igor Shilov, Yves-Alexandre de Montjoye, Jonas Geiping, and Maksym Andriushchenko. Claudini: Autoresearch discovers state-of-the-art adversarial attack algorithms for llms. *arXiv preprint arXiv:2603.24511*, 2026.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userbench: An interactive gym environment for user-centric agents, 2025a. URL https://arxiv.org/abs/2507.22034.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userrl: Training interactive user-centric agent via reinforcement learning, 2025b. URL https://arxiv.org/abs/2509.19736.

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

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Anja Surina, Amin Mansouri, Lars Quaedvlieg, Amal Seddas, Maryna Viazovska, Emmanuel Abbe, and Caglar Gulcehre. Algorithm discovery with llms: Evolutionary search meets reinforcement learning, 2025. URL https://arxiv.org/abs/2504.05108.

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, 2026. URL https://arxiv.org/abs/2602.06820.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents, 2026. URL https://arxiv.org/abs/2510.14967.

Xintao Wang, Heng Wang, Yifei Zhang, Xinfeng Yuan, Rui Xu, Jen tse Huang, Siyu Yuan, Haoran Guo, Jiangjie Chen, Shuchang Zhou, Wei Wang, and Yanghua Xiao. CoSER: Coordinating LLM-based persona simulation of established roles. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=BOrR7YqKUt.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.08755.

Sirui Xia, Yikai Zhang, Aili Chen, Siye Wu, Siyu Yuan, and Yanghua Xiao. From ai assistant to ai scientist: Autonomous discovery of llm-rl algorithms with llm agents. *arXiv preprint arXiv:2603.23951*, 2026.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shidong Yang, Ziyu Ma, Tongwen Huang, Yiming Hu, Yong Wang, and Xiangxiang Chu. Coevolve: Training llm agents via agent-data mutual evolution, 2026. URL https://arxiv.org/abs/2604.15840.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Simon Yu, Nicholas Tomlin, Marwa Abdulhai, Ximing Lu, Derek Chong, Abe Hou, Dilara Soylu, Sergey Levine, Christopher D. Manning, and Weiyan Shi. One frozen simulator is not enough: Simulator collapse in multi-agent rl, 2026. URL https://arxiv.org/abs/2608.12253.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents, 2026a. URL https://arxiv.org/abs/2505.22954.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, Chenyang Zhao, Henry Peng Zou, Haoyun Deng, Sathish Reddy Indurthi, Shujian Liu, Simin Ma, Xiaoyang Wang, Xin Eric Wang, and Song Wang. Cm2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use, 2026b. URL https://arxiv.org/abs/2602.12268.

Xuhui Zhou, Weiwei Sun, Qianou Ma, Yiqing Xie, Jiarui Liu, Weihua Du, Sean Welleck, Yiming Yang, Graham Neubig, Sherry Tongshuang Wu, and Maarten Sap. Mind the sim2real gap in user simulation for agentic tasks, 2026. URL https://arxiv.org/abs/2603.11245.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

A KNOBS TABLE

Qualitative examples of what and how modifications are made during training. Table 6 shows the *knobs* and the online modifications for ColBench. Table 7 demonstrates an example on UserGym.

Table 6: Knobs the meta-agent may write on ColBench, with what each was measured to do. Raising difficulty requires the recent success rate to reach 0.80; `max_steps` was refused four times under that rule before the run qualified. Across both runs, 15 knob writes produced one confirmed causal result, two measured harms, and three attributions later retracted — while the defects that mattered (a 41% deadlock, a reward-hacking channel, template collapse) were all found by reading trajectories, not by moving knobs.

Module	Knob	Range	Measured effect (2 runs, 624 steps each)
Environment	`max_steps`	3–60	$10 \to 6$ cost **Pass** -0.246, revert $+0.177$; both $> 2\sigma$. The only knob write in either run with a confirmed causal effect.
	`min_questions`	0–3	Did not count a turn that asked *and* answered, deadlocking 41% of episodes. Fixed in code; $1 \to 2$ then null, $2 \to 3$ harmful.
	`human_response_char_limit`	–	Not exercised.
User (symbolic)	`facts_per_reply`	1–3	$1 \to 3$ made the user dictate the reference solution (837-char replies); reverted. Reward attribution did not survive the gap band.
	`match_threshold`	$[0, 1]$	Four legs (0.9/0.6/0.35/0.9); **none cleared the measured noise floor** of 0.228.
	`volunteer_interval`	0–10	$10 \to 5$ raised flow success while the question rate *fell* 40%→26%; reverted as masking.
	`max_reveals`	0–20	**Structurally inert**: re-disclosing an already-revealed fact does not charge the budget, so the one fact that matters is free forever.
	`reveal_return_shape`	{T,F}	**Misnamed**: filters only `kind=="shape"` (8% of facts); the observed leak is `kind=="return"` (18%).
Reward	`per_gym_scale`	$[0, 2]$	Not exercised.
	`turn_reward_clip`	$[-1, 2]$	Not exercised. `predicate_weight` is unavailable: ColBench has no predicate library.
User (mix)	`user_strategy_mix`	$[0, 1]$	Held at 0.5. The two arms differ by 0.29 in success rate throughout, so the blended metric averages two different environments.
	`user_persona`	–	Unavailable (`travel` only).
Agent	`entropy_coeff`	$[0, 0.05]$	$0 \to 0.004$ raised `actor/entropy` 50% while behavioural diversity fell; effect lag exceeded the 10-step verification window. Reverted.

B DETAILS ON EXPERIMENTS

Baselines. We compare EAU against both prompting-based and RL-based baselines. For prompting, we include ReAct (Yao et al., 2022), which interleaves reasoning with actions, and Reflexion (Shinn et al., 2023), which improves agent behavior through verbal feedback and self-reflection. For RL training, we consider RAGEN (Wang et al., 2025b), a general multi-turn agent RL frame-

Table 7: Knobs the meta-agent may write on UserGym, by gym. Every write passes a gate enforcing the declared range and rejecting no-ops. Raising difficulty additionally requires the gym's recent success rate to reach 0.80 (`difficulty_gate.py`); below that the write is refused, so a gym the policy is struggling with cannot have its degenerate modes contained. [a] `function` is the only gym with a predicate library (five `probe_*` predicates); `predicate_weight` is rejected elsewhere. [b] `user_strategy_mix` is keyed `travel/colbench/tau2`, so the four other UserGym gyms resolve to `None` and cannot be split into symbolic and LLM arms – the key `tau` does not match `tau2`.

Module	Knob (range)	travel	tau	persuade	turtle	function
Environment	`max_steps` (3–60)	✓	✓	✓	✓	✓
	`search_failure_interval` (1–50)	✓	–	–	–	–
	`elicitation_interval` (1–20)	✓	–	–	–	–
	`noise_choice_number` (0–30)	✓	–	–	–	–
	`wrong_choice_number` (0–30)	✓	–	–	–	–
	`success_threshold`	–	–	–	✓	–
	`max_repeated_probes`	–	–	–	–	✓
Reward	`default_scale` $[0, 2]$	✓	✓	✓	✓	✓
	`per_gym_scale` $[0, 2]$	✓	✓	✓	✓	✓
	`turn_reward_clip` $[-1, 2]$	✓	✓	✓	✓	✓
	`predicate_weight` $[-0.5, 0.5]$	–	–	–	–	✓[a]
User	`user_strategy_mix` $[0, 1]$	✓	–[b]	–[b]	–[b]	–[b]
	`user_persona` (≤ 8)	✓	–	–	–	–
Agent	`entropy_coeff` $[0, 0.05]$	shared across gyms; ≤ 0.004 per write				

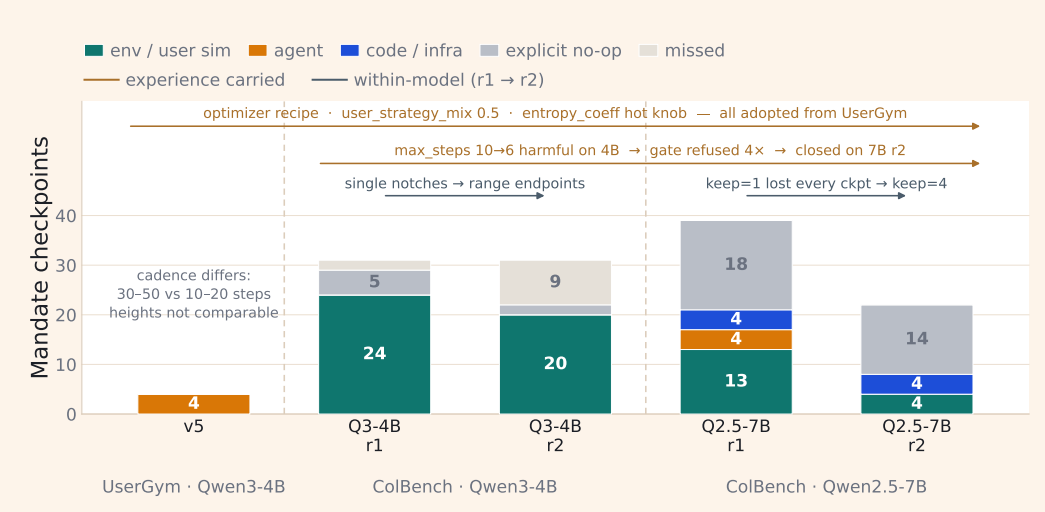

Figure 6: Experience flow from UserGym, *Qwen3-4B* to ColBench, *Qwen2.5-7B*.

work; Search-R1 (Jin et al., 2025), which trains agents through reinforcement learning over multi-step search trajectories; UserRL (Qian et al., 2025b), which targets interactive user-centric agents; and InfoPO (Kong et al., 2026), which introduces information-driven optimization for user-agent interaction. We additionally include vanilla PPO as a standard RL baseline to isolate the gains from our co-evolution framework. For UserGym and ColBench, all baseline results except PPO are taken from Kong et al. (2026); PPO and EAU are evaluated under our implementation.

Benchmarks. We evaluate our method on three representative user-centric benchmarks, i.e., **ColBench** (Zhou et al., 2025), **UserGym** (Qian et al., 2025b), and τ^2**-bench** (Yao et al., 2024; Barres et al., 2025). ColBench evaluates collaborative artifact creation, where the agent works with an LLM-simulated collaborator that has privileged access to a hidden reference artifact. We evaluate **EAU** on the *Backend Programming* subset, where the agent implements a Python function from an

incomplete specification, and is evaluated with hidden unit tests. UserGym is a suite of eight Gymnasium environments unified under a single tool interface, spanning preference elicitation, intent clarification, entity guessing, lateral-thinking puzzles, rule induction, policy-constrained tool use, persuasion, and search. Each episode pairs a rule-based task with an LLM-simulated user who holds the ground-truth intent and reveals it only when queried. Because the initial request is systematically underspecified, the agent must repeatedly trade off asking the user against committing to an action. We report the environment reward averaged within and across gyms. Specifically, we exclude the SearchGym for evaluation due to the cost of APIs. It requires the agent to conduct online search using a charged api. Notably, this does not affect the RL training, since it's an OOD gym solely for evaluation. τ^2-bench casts customer service as a dual-control problem in which both the agent and the simulated user can act on a shared world state. Unlike prior benchmarks where the user is a passive information source, the user here has its own tools, so the agent must verbally guide the user through actions it cannot perform itself while complying with a domain policy. We evaluate on the retail, airline, and telecom domains, and score a task as solved only when the resulting environment state matches the annotated target and all required information has been conveyed to the user.

C Additional Implementation Details

We show the hyperparameters in Table 8.

Table 8: Key hyperparameters for training on UserGym, ColBench, and τ^2-Bench environments.

Parameter	**UserGym**	**ColBench**	**τ^2-Bench**
Data Configuration			
Train batch size	64	64	32
Max prompt length	1152	2048	8192
Max response length	8192	8192	16384
Actor Configuration			
Learning rate	3×10^{-7}	3×10^{-7}	1×10^{-6}
PPO minibatch size	16	16	16
PPO microbatch size per GPU	4	4	2
KL loss coef	0.001	0.001	0.001
Entropy coefficient	0.001	0.001	0.001
Rollout Configuration			
Rollout engine	SGLang	SGLang	SGLang
Log prob micro batch size per GPU	4	4	2
Parallel rollouts (n)	5	5	5
Max turns	16	10	50
Response length (per turn)	1024	1024	1024
GPU memory utilization	0.5	0.5	0.5
Training Configuration			
GPUs per node	4	4	4
Total epochs	2	2	2

D Additional Analysis

Observations from the main results. We observe that with a strong model, i.e., *Qwen3-4B*, **`EAU`** is able to achieve generally better performance. On benchmarks that require capabilities that the model is more familiar with – namely coding (ColBench) rather than tool-call (UserGym), **`EAU`** can achieve relatively greater performance gain. This may suggest that **`EAU`** is more adept at eliciting better interaction behavior from the model than at overcoming its intrinsic capability limitations.

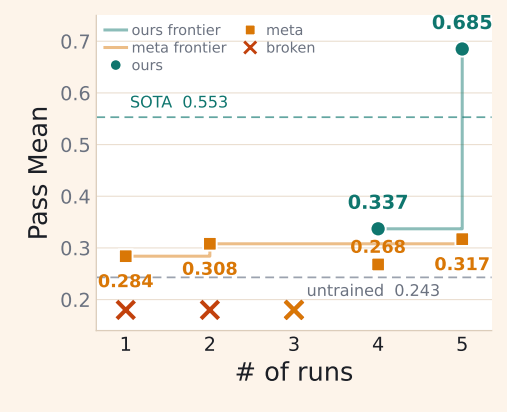

Figure 7: Best-score frontier comparison between Ours and Meta-agent. The frontier keeps flat (ties) when the score degrades or breaks.

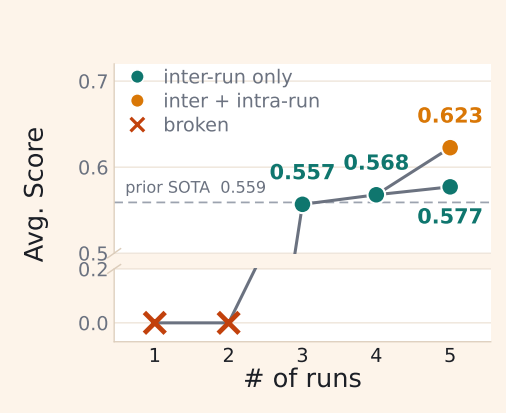

Figure 8: Results progression from UserGym with *Qwen3-4B*. Comparison between experiments that consider only inter-run experience sharing and those that consider both intra- and inter-run sharing.