INTERACT, REMEMBER, AND ACT: PREFERENCE AND PROGRESS-DRIVEN REINFORCEMENT LEARNING FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective collaboration between users and large language model (LLM) agents depends on both successful task execution and interactions that accommodate user preferences. However, existing agentic reinforcement learning (RL) approaches incorporating user simulator provide limited modeling of agents' capabilities and user experience, with no unified environment across heterogeneous tasks, and efforts to improve user experience and long-horizon credit assignment remain largely separate. In order to address these issues, we introduce PPD-RL, a multi-turn agentic RL framework jointly driven by user preferences and task progress. Specifically, PPD-RL defines a standardized action space for agents encompassing clarification queries, tool use, memory revisions, and final answers, together with unified interaction environments for training and inference. Moreover, to enable personalized assistance across sessions, the agent maintains an explicit user profile, updates it when certain attributes in the profile are missing or drifting, and conditions subsequent interactions on the corrected memory. Concurrently, to improve these behaviors, PPD-RL combines the task goal, rule-based memory feedback, and an LLM-judged preference score as RL rewards. It then distributes credit across multi-turns using positive rewards and measurable task progress, linking the learning signal to actions that advance task completion. We train and evaluate PPD-RL across various gym families spanning travel planning, customer service, and collaborative programming, measuring both task performance and user experience. Experimental results demonstrate that PPD-RL outperforms existing methods in this area and the larger models included in our evaluation on both dimensions. Code is available at https://anonymous.4open.science/r/PPD-RL.

1 INTRODUCTION

Large language model (LLM) agents are increasingly moving from isolated single-turn tasks to interactive applications in which they must communicate with users, invoke external tools, and revise their behavior as new information arrives (Yao et al., 2023; Schick et al., 2023). This shift changes the nature of the learning problem. A response can no longer be judged only by whether it is locally plausible: its value also depends on whether it advances the task, elicits the right information, respects the user's preferences, and remains consistent with decisions made earlier in the interaction.

Multi-turn user interaction makes this problem particularly difficult. User needs are often underspecified at the beginning of a conversation, may change as the task unfolds, and may persist across sessions (Liang et al., 2026). An agent must therefore decide not only *what action to take*, but also *what to ask*, *what to remember*, and *how prior user information should affect later decisions*. Prior work explicitly trains clarification behavior (Qian et al., 2024; Chen et al., 2025) or proactive assistance (Lu et al., 2025). Recent work has also begun to incorporate LLM-simulated users into reinforcement learning, enabling policies to learn directly from interactive trajectories rather than from static demonstrations alone (Wu et al., 2025). Nevertheless, these approaches still provide limited modeling of the user and the interaction itself. User information is commonly treated as transient conversational context, while task completion remains the dominant optimization target. Consequently, an agent can complete a local task without learning a persistent representation of the user or maintaining a consistent interaction style across sessions.

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.

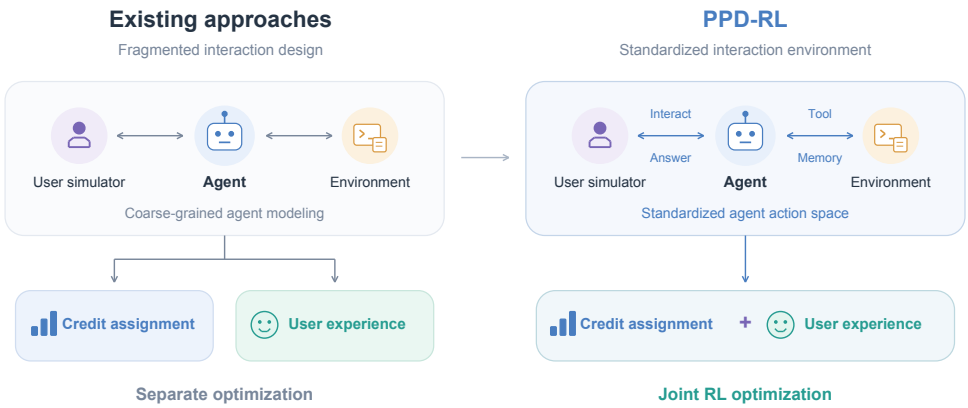


Figure 1: Existing user-simulated agentic RL methods typically address credit assignment and user experience separately, without a unified agent modeling and interaction protocol. PPD-RL introduces a standardized agent action space and interaction environment to jointly optimize both.

Persistent user modeling offers a natural way to address this gap, yet current methods remain limited. Although user profiles can summarize communication preferences and remain valid beyond a single conversation, personalization benchmarks and agent-memory systems generally treat them as conditioning context or a separate retrieval component (Salemi et al., 2024; Xu et al., 2025). A unified mechanism for representing, updating, and conditioning decisions on such profiles across heterogeneous agentic RL environments nevertheless remains underexplored.

A second challenge is credit assignment. In long-horizon user interactions, informative rewards are often sparse or delayed until the end of the trajectory, which makes credit assignment both more difficult and more important than in short-horizon settings (Shi et al., 2024). Existing methods typically rely on uniform turn weighting or purely local rewards, and neither captures which interaction turns actually contribute to the final outcome. Although some methods have been proposed to explore solutions, as illustrated in Figure 1, existing user-simulated agentic RL approaches typically treat credit assignment and user experience as separate concerns with coarse modeling.

To address these limitations, we introduce **PPD-RL**, a multi-turn agentic reinforcement learning framework jointly driven by user *preferences* and task *progress*. PPD-RL standardizes the action space and the interaction environment across tasks, so that a single policy can learn interaction behavior without a separate action protocol for each environment.

PPD-RL further equips the agent with an explicit user profile that persists across sessions. Rather than remaining a passive transcript of the conversation, this profile serves as an explicit decision variable: as the dialogue unfolds, the agent must recognize evidence about the user, judge whether its current belief is still valid, revise that belief when the user expresses or corrects a preference, and condition subsequent decisions on the updated memory. This design makes preference alignment directly trainable — the agent is evaluated not only on whether it completes the current task, but also on whether its behavior remains consistent with the user's evolving profile. In this way, the agent is trained not merely to finish the task, but to leave the user satisfied.

Finally, PPD-RL introduces a retrospective credit-assignment mechanism for long-horizon interaction. Uniformly distributing the final reward across turns cannot distinguish a decisive clarification from an incidental exchange, while purely local rewards overlook actions whose value becomes apparent only after the outcome is known. PPD-RL instead examines complete trajectories together with their outcomes: it statistically measures how strongly behaviors converge within the action space across successful trajectories, regards the more strongly converging behaviors as more important, and assigns greater credit to the interaction turns that produce them. Because this uses information that is unavailable when an action is generated but becomes available once the trajectory is complete, it yields a more informative learning signal than uniform turn weighting.

We evaluate PPD-RL across heterogeneous gym families comprising four environments, measuring both task performance and user experience. The results show that PPD-RL outperforms the existing methods in this area and the larger models included in our evaluation on both dimensions.

Contributions. Our main contributions are threefold:

- We introduce a standardized action space and a unified interaction environment for multi-turn agentic RL across heterogeneous tasks, together with a persistent user-profile mechanism that supports explicit memory updates and preference-conditioned decisions across sessions.
- We propose a retrospective credit-assignment method that uses complete trajectory outcomes to measure action-space convergence across successful trajectories and assigns greater turn-level credit to the behaviors that converge most strongly.
- We evaluate PPD-RL across three heterogeneous gym families, comprising four environments, and show that it improves both task performance and user experience over the evaluated agentic methods and larger general-purpose models.

2 Related Work

2.1 Agentic RL Training and Adaptation

Reinforcement learning has become the dominant paradigm for post-training large language models. Early work aligns single-turn behavior with human preferences, using policy optimization against a learned reward model or direct preference objectives such as DPO (Rafailov et al., 2023). More recently, RL with verifiable rewards has driven strong gains on tasks with checkable outcomes, where group-relative methods such as GRPO (Shao et al., 2024) estimate advantages from sampled outcomes and remove the need for a separate critic.

Agentic RL instead optimizes complete interactions, coupling the policy to environment dynamics and delayed outcomes. LMRL Gym provides early multi-turn language RL testbeds (Abdulhai et al., 2023); ArCHer uses hierarchical actor–critic learning (Zhou et al., 2024); and trajectory-preference methods extend RLHF or DPO beyond a single response (Shani et al., 2024; Shi et al., 2024). PPD-RL focuses on a complementary setting: one policy is trained online across heterogeneous user-facing environments, with explicit task progress used to assign turn-level credit.

2.2 User-Centric Agent Design and Evaluation

MUA-RL (Zhao et al., 2025) and UserRL (Qian et al., 2025) directly incorporate an LLM user simulator into multi-turn agent RL, enabling policies to learn from realistic interactive feedback. A parallel line improves task completion through better credit assignment: ITPO (Wang et al., 2026) derives turn-wise process rewards from sparse outcome signals with an implicit process reward model, IRC (Modecrua et al., 2026) calibrates per-turn rewards against their empirical correlation with task success, and InfoPO (Kong et al., 2026) credits each turn by the information gain its feedback produces, measured through counterfactual masking. A third family additionally brings the user into the optimization objective: PPP (Sun et al., 2025) jointly optimizes productivity, proactivity, and personalization with multi-objective RL, and BAO (Yao et al., 2026) regularizes inter-turn behavior to trade off task performance against user effort.

Adjacent approaches optimize long-horizon collaboration (Wu et al., 2025), proactive intent clarification (Luo et al., 2026), or continual personalization with explicit memory (Liang et al., 2026); PRIME instead improves a frozen agent through an evolving experience library (Wang & Jiang, 2026). PPD-RL brings these concerns into multi-task online RL by making profile revision a shared, rule-verifiable action. The profile is deliberately initialized with error or drift, and the agent must detect an incidental correction and update its belief while completing the task. Ultimately, PPD-RL deliver a comfortable user experience while enabling efficient task execution.

3 Environments

PPD-RL trains a single agent across four environments that differ in domain, reward density, and horizon (Table 1): TravelGym, TauGym, Tau2Gym, and ColGym. Figure 2 illustrates how the agent interacts with users and these environments under the shared training framework.

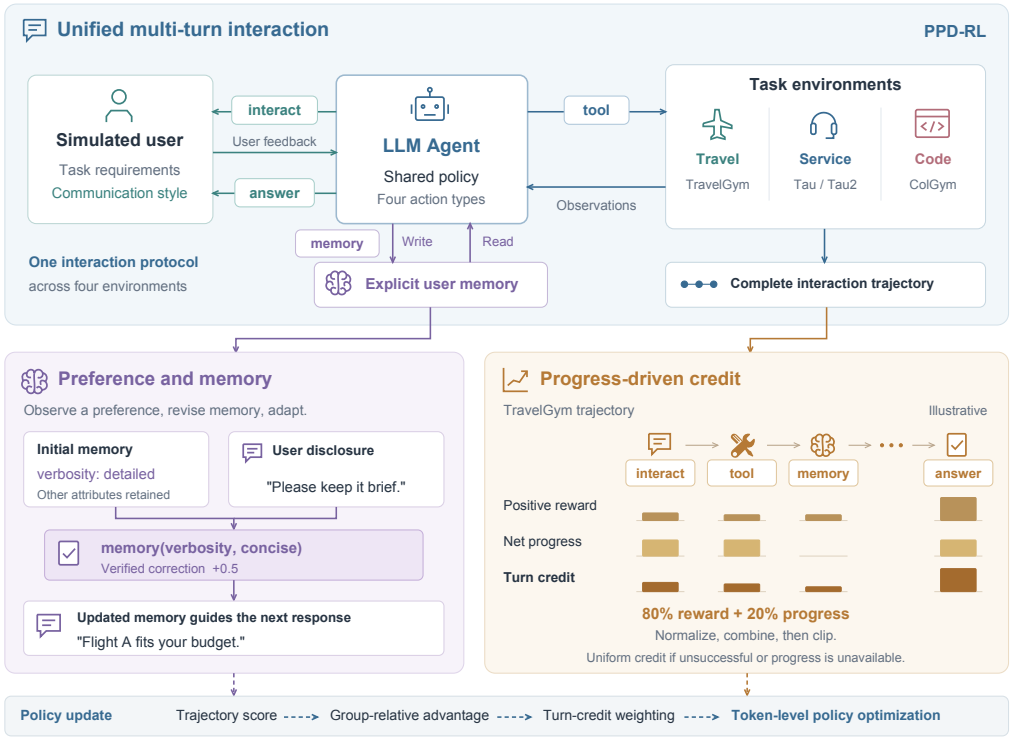


Figure 2: Overview of PPD-RL. A shared LLM agent interacts with users and environments through four actions: interact, tool, memory, and answer. Explicit user memory supports preference updates and guides subsequent responses, while progress-driven credit assignment combines verified rewards and task progress to weight interaction turns during policy optimization. The trajectory and credit bars are illustrative.

TravelGym. The agent plans trips across flight, lodging, restaurant, and rental-car dimensions, each with a latent user preference. It elicits preferences through `interact`, queries the travel database with `tool`, and selects one option per dimension with `answer`. Dense rewards cover valid retrieval, preference elicitation, and recommendation quality. Potential-based shaping penalizes repeated or invalid calls, while a progress signal tracks the reduction in unresolved dimensions, unknown preferences, and unmade choices.

TauGym and Tau2Gym. Both model customer-service agents that converse with users and invoke internal tools. TauGym wraps the retail and airline domains of `tau-bench` (Yao et al., 2024) and uses its sparse terminal $0/1$ reward. Tau2Gym wraps the retail, airline, and telecom domains of `tau2` (Barres et al., 2025), whose continuous $[0, 1]$ reward combines database, action, and environment checks. We also detect orchestrator failures and exclude episodes from preference scoring.

ColGym. Built on ColBench (Zhou et al., 2025), ColGym presents an ambiguous programming specification whose hidden rules must be recovered through user interaction, extending the motivation of requirements-clarification methods such as ClarifyGPT (Mu et al., 2023) from code generation to executable agent tasks. The agent tests Python drafts in a sandbox with `tool` and submits its implementation with `answer`. Differential testing assigns a binary reward: all ten hidden cases must match the reference implementation.

Unified action space. All environments expose `interact_with_env(choice, content)` with four operations: `interact` communicates with the user, `tool` accesses the environment backend, `answer` submits a task action or final output, and `memory` updates the user profile when missing or drifting. Training and evaluation use the same protocol and reward definitions. Each gym specializes the backend semantics; for example, `tool` queries a travel database in TravelGym but executes code in ColGym.

Table 1: The four environments.

	TravelGym	TauGym	Tau2Gym	ColGym
Domain	Travel planning	Customer service	Customer service	Backend programming
Task reward	Dense, itemized	Sparse $0/1$	Continuous $[0, 1]$	Binary
Train / test	925 / 471	500 / 165	178 / 100	1000 / 500
Max turns	20	20–30	20–30	10

4 PPD-RL

4.1 User Profile and Memory

Beyond the task environments, PPD-RL introduces a user-profile layer shared across all gyms. Unlike general agent memories that organize past experience (Xu et al., 2025; Wang & Jiang, 2026), or continual personalization frameworks that update per-user memory from feedback (Liang et al., 2026), this layer exposes profile revision as a rule-verifiable action inside every gym. It thereby turns user modeling from a byproduct of task completion into an explicitly trainable objective.

Profile representation. The user profile P is a four-tuple of communication-style attributes: *tone* (formal / warm), *verbosity* (concise / detailed), *address* (name / honorific, where the name label carries a concrete name as its value), and *jargon* (technical / plain). The agent maintains an explicit memory M with the same schema. The initial memory M_0 is derived from P by the data generator and may be incomplete or incorrect; during interaction the agent rewrites M through the `memory` action. M is thus explicit, auditable, and persistent across turns — in contrast to treatments of user information as transient context.

Data generation. Every task carries a triple (P, M_0, meta): P samples each attribute independently and uniformly; M_0 is generated in a complete : missing : drift ratio of $4{:}3{:}3$ — *complete* sets $M_0 = P$, *missing* blanks one affected attribute a^*, and *drift* flips it to the opposite label — with at most one affected attribute per task, so that a unique correction target a^* exists. The metadata records the case type, the affected attribute, and the disclosure timing: half of the disclosures arrive at the opening turn (*first*) and half at a turn chosen among the first three (*early*), with the choice derived deterministically from the task identifier and hence reproducible.

Memory scoring. A `memory` action submits an attribute–label pair and is verified by rule against the unique ground-truth solution: correcting the affected attribute earns $+0.5$, once, after which the attribute is marked solved; acting on an unaffected attribute, submitting an incorrect value, or rewriting after solving earns -0.5; unparseable actions score zero. For *address*, the value must additionally match the true name under case normalization. Scoring is purely rule-based and identical between training and evaluation.

4.2 Multi-Turn Reinforcement Learning

All four environments share the tool interface defined in Section 3, which is what lets a single policy serve them. This places PPD-RL within the broader multi-turn language-RL setting formalized by LMRL Gym (Abdulhai et al., 2023), but uses a single action protocol across otherwise incompatible task backends. Long-horizon agent systems such as RAGEN similarly optimize complete interaction trajectories (Wang et al., 2025); our focus is how to combine their outcomes with turn-level progress in heterogeneous user-facing tasks.

At turn t the agent observes o_t and emits an action a_t together with a memory update:

$$\pi_\theta(a_t \mid o_t, m_t)\,, \qquad m_{t+1} = \mathrm{Update}(m_t,\ o_t,\ a_t)\,, \tag{1}$$

where m_t is the agent's belief about the user, seeded from `init_memory` (Section 4.1). A task sampled n times yields a group of trajectories $\{\tau^{(i)}\}_{i=1}^{n}$, each carrying a trajectory-level score built from per-turn rewards, so that distributing credit across turns is the central estimation problem.

The training reward combines environment-defined task rewards, rule-based memory feedback, and an LLM-judged preference score. The *task component* follows each environment's reward structure (Table 1): dense and additive in TravelGym, sparse in TauGym, continuous in Tau2Gym, and binary in ColGym, including any environment-specific clarification bonuses. The *memory component* provides turn-level feedback on each `memory` action according to the rule-based procedure in Section 4.1. The *preference component* evaluates the agent's communication against the ground-truth user profile at the end of the trajectory, using the LLM judge described in Section 5.1. This judged score contributes to the training reward and is also reported as satisfaction during evaluation. The resulting learning signal therefore directly rewards task performance, accurate memory updates, and preference-consistent interaction.

The trajectory score is a reverse, discounted accumulation of per-turn rewards — reward-to-go, with the most recent turn weighted most heavily:

$$\text{score}(\tau) \;=\; \sum_{k=0}^{T-1} \gamma^k \, r_{T-k}. \tag{2}$$

Training uses $\gamma = 1.0$, the undiscounted sum. A smaller γ subsidizes early termination — a reward collected at turn three and immediately ending scores higher than the same reward deferred to turn ten — and in our earlier runs this discount systematically shortened trajectories. We also consider a length-normalized plain sum $\sum_t r_t$ as an alternative.

4.3 Progress-Driven Credit Assignment

The central question is how to convert the trajectory-level score into per-turn, and ultimately per-token, advantages. Hierarchical actor–critic methods (Zhou et al., 2024) and trajectory-level preference optimization (Shani et al., 2024) address related long-horizon learning problems; PPD-RL instead derives turn credit directly from verifiable rewards and task progress. Uniform credit, $c_t = 1$ for all t, treats the turn on which the agent finally commits to an option and the turn on which it first asks a clarifying question as equally responsible for the outcome; in multi-turn interaction a misdirected early clarification can doom the trajectory, while a late `answer` may be the only decision that determines success. PPD-RL instead assigns credit retrospectively, from two quantities that are verifiable once the trajectory completes: the environment reward r_t, and the net progress p_t the turn made.

Progress signal. In TravelGym we define a potential over the action space, ϕ_t, as the count of unresolved dimensions, dimensions without a locked best choice, and unelicited preferences; the per-turn progress is the net reduction

$$p_t = (\phi_{t-1} - \phi_t)^+ . \tag{3}$$

Positive progress arises from a first successful retrieval, from locking the best choice of a dimension, or from eliciting a real preference. Because p_t is a net reduction, repeated queries leave ϕ unchanged and cannot farm credit — the defining property of potential-based shaping (Ng et al., 1999), which leaves the optimal policy unchanged.

Turn credit. For a successful trajectory with measurable progress, the credit of turn t is

$$c_t \;=\; (1-\beta)\, n \, \frac{r_t^+}{\sum_{t'} r_{t'}^+} \;+\; \beta\, n \, \frac{p_t}{\sum_{t'} p_{t'}}, \qquad \beta = 0.2, \tag{4}$$

where $r_t^+ = \max(r_t, 0)$. Both terms are individually normalized, so the total credit $\sum_t c_t = n$ matches the uniform baseline in scale: 80% of the weight goes to environment-verified reward and 20% to advancing the task frontier — crediting turns such as a tool call that unlocks a new dimension even when its immediate reward is zero. A turn with negative reward receives no reward-term credit but can still earn progress credit. Credits are clamped to $[0.25, 3.0]$ around the unit mean, bounding the per-turn credit ratio at $12\times$. Two fallback rules keep the scheme safe and comparable: failed trajectories ($R(\tau) \le 0$) and trajectories without a progress signal — TauGym, Tau2Gym, ColGym, or negligible progress — fall back to uniform credit $c_t = 1$. The choice $\beta = 0.2$ is deliberate: an earlier pure-progress version ($\beta = 1$) inverted the credit order relative to rewards and destabilized training; anchoring on the reward term keeps credit consistent with environment verification.

Grouped advantage. Advantages are computed by a multi-turn GRPO estimator (Shao et al., 2024). A task is sampled $n = 8$ times to form a group; the group mean and standard deviation provide the baseline, so no critic network is trained:

$$z_\tau = \frac{\text{score}(\tau) - \mu_g}{\hat{\sigma}_g + \varepsilon}, \qquad \hat{\sigma}_g = \max(\sigma_g,\, \sigma_{\text{floor}})\,, \qquad \sigma_{\text{floor}} = 0.25, \tag{5}$$

The standard-deviation floor prevents near-zero-variance groups — for instance, groups made barely distinguishable by process shaping — from being amplified into full-magnitude noise gradients by the z-score; groups below the floor emit proportionally small gradients, groups above are unaffected.

Token-level advantage. Within a turn, tokens share the turn's credit equally, with a global rescaling:

$$A_{\tau,j} \;=\; z_\tau \cdot c_{t(j)} \cdot \frac{N_{\text{nz}}}{\sum_{t'} c_{t'} L_{t'}}, \tag{6}$$

where $t(j)$ is the turn containing token j, L_t the length of turn t, and N_{nz} the number of tokens with nonzero credit. Two invariants follow: the ordering of credit *within* a trajectory is determined solely by $\{c_t\}$, and the comparison *between* trajectories solely by z_τ — the two are decoupled, which is what the ablations in Section 5.3 isolate. We compare against uniform credit, a reward-to-go turn-level variant, and an expectation-maximization formulation.

5 Experiments

5.1 Experimental Settings

Training data. Multi-task RL trains on a merged pool of 2603 tasks drawn from all four gyms (925 TravelGym, 1000 ColGym, 500 TauGym, 178 Tau2Gym); the test split holds 1236 tasks (471 / 500 / 165 / 100). Samples carry their origin gym as a field and are routed to the corresponding environment automatically.

SFT cold start. We fully fine-tune **Qwen3-4B** (Yang et al., 2025) (36 layers, hidden size 2560, bf16) on 1330 successful trajectories — those whose progress score reached 1.0 — drawn from all four gyms, with 70 held out. Trajectories average 8.9 turns (median 9, maximum 20) and all terminate. Training uses 3 epochs, a learning rate of 1×10^{-5} with cosine decay and 10% warmup, and a global batch of 32, giving roughly 41 steps per epoch and 123 steps in total.

Multi-task RL. RL cold-starts from the SFT weights and runs multi-task GRPO (Shao et al., 2024) with group size $n = 8$, a learning rate of 1×10^{-6}, and a budget of 5 epochs over the merged pool; the full configuration is listed in Appendix B. Training and rollout are implemented with the verl framework (Sheng et al., 2025) on top of FSDP, with rollouts served by sglang (Zheng et al., 2023). All training are conducted on 8 $\times$ NVIDIA A100 40GB GPUs.

Preference satisfaction. At the end of each trajectory, an LLM judge scores the agent's communication style against the ground-truth profile along all four attributes on a three-point scale — *poor* (0), *fair* (0.1), *good* (0.2) — for a maximum of 0.8, plus a 0.2 bonus when the agent does not interrogate the user about the profile outright, yielding a score in $[0, 1]$. The judge operates under strict rules. When the judge fails, the score is excluded from evaluation statistics rather than counted as zero. This experience score serves both as a preference reward during training and as the satisfaction metric during evaluation. It complements task rewards and rule-based memory feedback by directly encouraging communication that is consistent with the user's preferences.

5.2 Comparison Experiments

Table 2 shows that PPD-RL, despite its 4B base, leads on three of the four gyms. On TravelGym, it scores 0.495 versus 0.382 for InfoPO-4B, a 0.113 gain consistent with the task's emphasis on eliciting latent preferences. It also improves over the strongest baselines on TauGym (0.333 vs. 0.259, 29% relative) and ColGym (0.670 vs. 0.653). On Tau2Gym, it ranks second at 0.429, behind DeepSeek-V4-Flash-preview at 0.476.

Table 2: Performance across the four gyms. PPD-RL is trained from the 4B Qwen3-4B base; the baselines include general-purpose models and domain-specific methods, MUA-RL (Zhao et al., 2025), UserRL (Qian et al., 2025), and InfoPO (Kong et al., 2026), which are reproduced from their open-source code, all evaluated under the same protocol. *Process* reports the task score in each environment, *Preference* is the LLM-judged score of user-personalization satisfaction, averaged over the gyms where it is measured, and *Avg* is the arithmetic mean of the four process scores and the preference score. Best result per column in bold.

Model	Process				Preference	Avg
	Travel	Tau	Tau-2	Col		
Qwen3-235B-A22B-2507	0.329	0.193	0.349	0.580	0.638	0.418
DeepSeek-V4-Flash-preview	0.284	0.225	**0.476**	0.653	0.683	0.464
Gemini-2.5-Flash	0.203	0.201	0.395	0.497	0.688	0.397
GPT-4o-mini	0.131	0.259	0.285	0.409	0.454	0.308
MUA-RL-32B	0.092	0.259	0.324	0.163	0.659	0.299
UserRL-4B	0.337	0.246	0.341	0.418	0.612	0.391
InfoPO-4B	0.382	0.197	0.354	0.546	0.587	0.413
PPD-RL (ours)	**0.495**	**0.333**	0.429	**0.670**	**0.743**	**0.534**

Table 3: Effect of the training recipe, starting from the Qwen3-4B base model. All rows are evaluated under the same protocol as Table 2. "+ RL" denotes RL applied directly to the base model without the SFT cold start; PPD-RL uses the full recipe.

Training recipe	Process				Preference	Avg
	Travel	Tau	Tau-2	Col		
Qwen3-4B (base)	0.152	0.096	0.193	0.322	0.431	0.239
+ SFT	0.425	0.253	0.357	0.533	0.653	0.444
+ RL	0.368	0.135	0.245	0.462	0.619	0.366
PPD-RL (ours): + SFT + RL	0.495	0.333	0.429	0.670	0.743	0.534

PPD-RL also achieves the highest preference score, 0.743, compared with 0.688 for the strongest baseline, an absolute improvement of 0.055 (8% relative), and leads in average score (0.534 vs. 0.464). Thus, the 4B agent is competitive in both task execution and interaction quality.

5.3 Ablation Studies

Table 3 isolates the contribution of each training stage. SFT yields consistent gains over Qwen3-4B, raising Travel from 0.152 to 0.425, Tau from 0.096 to 0.253, Tau2 from 0.193 to 0.357, ColBench from 0.322 to 0.533, and preference from 0.431 to 0.653. This shows that supervised cold-start data establishes the interaction format, tool-use behavior, and task prior.

RL applied directly to the base also improves every metric, but remains weaker than SFT (average 0.366 vs. 0.444), indicating that reward optimization alone does not acquire the same behavioral prior. Starting RL from SFT performs best: relative to SFT alone, it gains 0.070 on Travel, 0.080 on Tau, 0.072 on Tau2, 0.137 on ColBench, and 0.090 on preference. SFT and RL therefore provide complementary initialization and refinement.

Table 4 separates the effects of user profiles and progress-driven credit. Uniform turn credit lowers Travel from 0.495 to 0.421, Tau from 0.333 to 0.305, Tau2 from 0.429 to 0.407, and ColBench from 0.670 to 0.620. The largest relative drop occurs on Travel (17.6%), where the agent must identify

Table 4: Component ablation. "w/o user profile" removes the profile layer, including the initial memory, the `memory` action, and the preference reward; "w/o credit assignment" falls back to uniform per-turn credit. All variants share the same SFT start and hyperparameters.

Variant	Process				Preference	Avg
	Travel	Tau	Tau-2	Col		
PPD-RL w/o user profile	0.451	0.312	0.419	0.693	0.627	0.500
PPD-RL w/o credit assignment	0.421	0.305	0.407	0.620	0.731	0.497
PPD-RL	0.495	0.333	0.429	0.670	0.743	0.534

which interaction resolves a latent requirement. The ColBench drop also suggests that the benefit transfers through multi-task training to sparse-feedback tasks.

Removing the user-profile layer lowers Travel, Tau, and Tau2 to 0.451, 0.312, and 0.419, but causes a much larger preference drop, from 0.743 to 0.627 (18.5% relative). This directly supports explicit memory for preference-consistent interaction. ColBench is the only exception: the ablation reaches 0.693 versus 0.670, suggesting a small trade-off between user modeling and strict programming correctness.

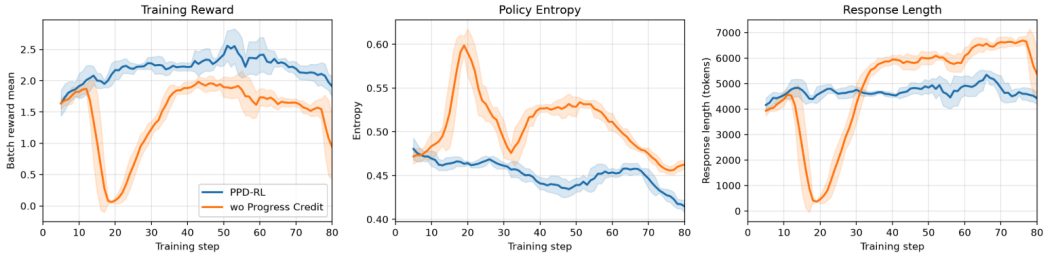


Figure 3: Training dynamics of PPD-RL versus the variant without progress-driven credit assignment (the "w/o credit assignment" row of Table 4) over the first 80 training steps. From left to right: mean batch reward, policy entropy, and mean response length (tokens), with shaded regions indicating variation across runs.

Training dynamics. Figure 3 compares PPD-RL with the variant that removes progress-driven credit assignment through curves. The two differ in stability rather than in asymptotic reward alone. The ablated variant collapses within the first twenty steps: reward falls from about 1.8 to below 0.2, entropy spikes from 0.48 to 0.60, and response length shrinks from roughly $4{,}000$ tokens to a few hundred, indicating near-empty trajectories. It recovers by mid-training, but reward declines again later as responses grow beyond $6{,}000$ tokens. PPD-RL instead keeps all three curves stable: reward rises above 2.0, entropy decreases smoothly, and responses stay near $4{,}000$–$5{,}000$ tokens. These dynamics expose the limitation of uniform credit: a trajectory-level signal cannot identify which turn caused the outcome, allowing the policy to drift toward overly short or long interactions. Progress-driven credit supplies this missing turn-level distinction.

6 Conclusion

We presented PPD-RL, which trains a single agent across four heterogeneous interactive gyms — travel planning, two customer-service suites, and backend programming — under a unified action space. A user profile layer compiles a four-attribute persona into the agent's initial memory and perturbs one attribute, so that the agent must notice a preference disclosed in passing and rewrite what it believes about the user. Rewards combine each gym's progress score with a judged preference score, and a multi-turn GRPO estimator assigns turn-level credit so that progress and preference are optimized jointly.

AI USE STATEMENT

In this work, we used generative AI tools to assist with (i) synthesizing part of the training data and supporting data analysis, (ii) surveying and organizing related work, and (iii) polishing the writing of the manuscript. AI assistance did not play an essential role in the completion of this work: the research ideas, the method design, the experiments, and the conclusions are the authors' own. We have reviewed all AI-assisted work, and we take full responsibility for the final content of this paper, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments in this work are conducted in fully simulated environments: the tasks and users are synthetic or drawn from public benchmarks, and no real human subjects are involved at any stage. The user profiles used in training and evaluation contain only surface-level communication preferences and no real personal information. Although personalization could in principle be used to steer user behavior, our work targets benign assistance — eliciting and respecting communication preferences to improve task completion and user experience — and all preference signals are produced by simulated users rather than collected from real people. Finally, both the user simulator and the preference judge are LLM-based and may inherit biases of the underlying models, which could affect the reported preference scores; we therefore release our code and data to facilitate scrutiny and replication.

REPRODUCIBILITY STATEMENT

We have made substantial efforts to ensure the reproducibility of this work. The four environments, the unified action protocol, and the user-profile construction are specified in Section 3; the PPD-RL method, including the reward design and the progress-driven credit assignment, is described in Section 4; the training and evaluation configurations, including the SFT recipe, the multi-turn GRPO hyperparameters, the user-simulator settings, and all evaluation metrics, are provided in Section 5.1 and Appendix B. The code and model weights are available at the anonymous repository and will be fully released to the public in the future.

REFERENCES

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL Gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv preprint arXiv:2311.18232*, 2023.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Sercan O. Arik. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. In *International Conference on Learning Representations*, 2025.

Fanqi Kong, Jiayi Zhang, Mingyi Deng, Chenglin Wu, Yuyu Luo, and Bang Liu. InfoPO: Information-driven policy optimization for user-centric agents. *arXiv preprint arXiv:2603.00656*, 2026.

Kaiqu Liang, Julia Kruk, Shengyi Qian, Xianjun Yang, Shengjie Bi, Yuanshun Yao, Shaoliang Nie, Mingyang Zhang, Lijuan Liu, Jaime Fernandez Fisac, Shuyan Zhou, and Saghar Hosseini. Learning personalized agents from human feedback. *arXiv preprint arXiv:2602.16173*, 2026.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Proactive agent: Shifting LLM agents from reactive responses to active assistance. In *International Conference on Learning Representations*, 2025.

Haohao Luo, Zexi Li, Yuexiang Xie, Wenhao Zhang, Yaliang Li, and Ying Shen. IntentRL: Training proactive user-intent agents for open-ended deep research via reinforcement learning. *arXiv preprint arXiv:2602.03468*, 2026.

Wachiravit Modecrua, Krittanon Kaewtawee, Krittin Pachtrachai, and Touchapon Kraisingkorn. Multi-turn reinforcement learning for tool-calling agents with iterative reward calibration. *arXiv preprint arXiv:2604.02869*, 2026.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. ClarifyGPT: Empowering LLM-based code generation with intention clarification. *arXiv preprint arXiv:2310.10996*, 2023.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 1088–1113, 2024. doi: 10.18653/v1/2024.acl-long.61.

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. *arXiv preprint arXiv:2509.19736*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. arXiv:2305.18290.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 7370–7392, 2024. doi: 10.18653/v1/2024.acl-long.399.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, Avinatan Hassidim, Yossi Matias, and Remi Munos. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*, 2024.

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. Introduces the GRPO algorithm.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. arXiv:2409.19256; the verl framework.

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2312–2324, 2024. doi: 10.18653/v1/2024.emnlp-main.138.

Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized LLM agents. *arXiv preprint arXiv:2511.02208*, 2025.

Haoyu Wang, Yuxin Chen, Liang Luo, Buyun Zhang, Ellie Dingqiao Wen, and Pan Li. Implicit turn-wise policy optimization for proactive user-LLM interaction. *arXiv preprint arXiv:2603.23550*, 2026.

Prince Zizhuang Wang and Shuli Jiang. PRIME: Training free proactive reasoning via iterative memory evolution for user-centric agent. *arXiv preprint arXiv:2604.07645*, 2026.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. CollabLLM: From passive responders to active collaborators. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67260–67283, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025.

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, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yihang Yao, Zhepeng Cen, Haohong Lin, Shiqi Liu, Zuxin Liu, Jiacheng Zhu, Zhang-Wei Hong, Laixi Shi, and Ding Zhao. Pushing forward pareto frontiers of proactive agents with behavioral agentic optimization. *arXiv preprint arXiv:2602.11351*, 2026.

Weikang Zhao, Xili Wang, Chengdi Ma, Lingbin Kong, Zhaohua Yang, Mingxiang Tuo, Xiaowei Shi, Yitao Zhai, and Xunliang Cai. MUA-RL: Multi-turn user-interacting agent reinforcement learning for agentic tool use. *arXiv preprint arXiv:2508.18669*, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2023.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209, 2024.

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. Introduces the ColBench benchmark used by ColGym.

A Environment and User Profile Details

This appendix specifies the task construction, serialized actions, user profile state, and simulator protocol used by the four environment implementations. The examples are schematic instantiations rather than test-set records: entity names and identifiers are deliberately replaced, while the interaction and validation logic is preserved.

A.1 Unified Interaction Protocol

Each episode is identified by an instance ID and owns a persistent environment state. At turn t, the policy receives the task instruction, the current explicit memory, and the preceding observations, and emits one call to `interact_with_env(choice, content)`. The wrapper converts the call into the syntax expected by the selected gym, advances that gym by one step, and returns natural-language feedback, the step reward, and a terminal flag. State is released only after normal termination, truncation, or an explicit finish operation; consequently, tool results and profile updates remain visible throughout the episode.

Table 5 gives the paper-level action schema. In the minimal TravelGym implementation, the conversational operation is serialized as `choice=action`; we rename it `interact` in the unified interface to distinguish user conversation from environment-internal tool actions. The profile-augmented implementation additionally exposes `memory`. Thus, `action` in the original TravelGym wrapper and `interact` below denote the same operation.

Table 5: Unified action schema. Exactly one operation is emitted per turn.

Choice	Content	Environment effect
`interact`	Natural-language utterance	Sends a focused question or response to the simulated user; it does not directly modify the task database.
`tool`	Gym-specific query or tool call	Reads or changes task state through the gym backend: database retrieval, customer-service API invocation, or sandboxed code execution.
`answer`	Option ID, JSON tool call, or program	Commits a task output or invokes a domain tool. Its exact termination semantics are defined by each gym.
`memory`	JSON attribute update	Rewrites one field of explicit user memory after the preference has been observed.

The wrapper rejects unknown choices and leaves gym-specific validation to the backend. A malformed tool call therefore receives the same observation as it would under direct use of that gym, while a malformed memory string is handled by the deterministic profile parser described in Section A.3. Training and evaluation share this execution path; only policy sampling and metric aggregation differ.

A.2 Task Construction and Environment-Specific Schemas

A.2.1 TravelGym

Task construction. A TravelGym instance combines two to four dimensions selected from {*flight*, *hotel*, *apartment*, *restaurant*, *rental car*}. Each dimension stores (i) arguments required for a valid database query, (ii) one or more latent natural-language preferences, (iii) correct and best option IDs, and (iv) distractor options. The released loader merges eight difficulty subsets: `22`, `33`, `44`, `2222`, `233`, `333`, `334`, and `444`. The number of digits is the number of required dimensions; each digit is the preference difficulty for that dimension. Candidate order is shuffled with a fixed data seed, and one answer is accepted per dimension.

Actions and state. An `interact` call asks one concrete question. The simulator classifies the utterance as ordinary conversation, a concrete request for an available preference, a request for a

nonexistent preference, or an overly general preference question. When appropriate it reveals one latent preference indirectly. A `tool` call must cover all ground-truth arguments for one dimension; a successful query returns shuffled candidates but never their correctness labels. An `answer` call supplies a candidate ID. The episode terminates after one choice has been made for every required dimension or after the turn limit. The state records attempted queries, remaining correct and best options, choices already made, and actively versus passively elicited preferences.

TravelGym interaction example

Task shown to the agent. Plan a trip requiring a hotel and a rental car. The initial request fixes the city and dates but omits comfort and vehicle preferences.
Agent → environment {`choice: interact, content: ''Would you prefer a quiet hotel or one close to nightlife?''`}
Simulated user → agent "I am a light sleeper, so being away from late-night noise would make the stay much easier."
Agent → environment {`choice: tool, content: ''hotel(city=..., dates=..., quiet=true)''`}
Environment → agent A shuffled list of hotel records with option IDs and public attributes.
Agent → environment {`choice: answer, content: ''hotel: H-17''`}
The same sequence is repeated for the rental-car dimension. Asking "What are all your preferences?" is treated as an overly general query and does not resolve a latent item.

A.2.2 TAUGYM

Task construction. TauGym wraps the retail and airline domains of `tau-bench`. An instance contains a user goal, an initial database state, a domain policy, the internal tool signatures available to the service agent, and a reference set of database changes or actions required for success. Typical retail tasks ask the agent to inspect, modify, exchange, or cancel an order; airline tasks involve reservations, passengers, flights, seats, bags, or payment methods. The user goal is held by the simulator rather than copied verbatim into the agent prompt.

Actions and state. `interact` sends a customer-facing message. `tool` is a local discovery operation: `content=tools` returns the available internal tools once, and `content=help` returns usage guidance. `answer` executes one internal tool call encoded as {`"name": ..., "arguments": {...}`}; it is not the final customer-facing message. The agent communicates its resolution through `interact`, and the underlying user simulator determines when the conversation has concluded. Native `tau-bench` evaluation compares the final database state and required actions with the task reference and produces a sparse success value. The profile layer is deliberately orthogonal: a correct task state can coexist with poor preference adherence, and vice versa. Because the native simulator and tools are synchronous, the Gym wrapper runs each blocking step in a worker thread so concurrent rollout coroutines remain responsive.

TauGym interaction example

User goal held by the simulator. Exchange one item from an existing retail order for a different size, subject to the domain policy.
Agent `interact`: asks for the order identifier and confirms which line item should change.
User: supplies only the requested information in natural language.
Agent `tool`: requests `tools` and reads the available signatures. It then uses `answer` with a JSON call to look up the order. After checking eligibility, it issues another `answer` call to the exchange tool with the returned order, item, and replacement identifiers.
Agent `interact`: confirms the completed exchange without inventing a status not present in the tool observations. Success requires the expected backend mutation, not merely a plausible final message.

A.2.3 TAU2GYM

Task construction. Tau2Gym wraps the official `tau2` retail, airline, and telecom suites. Each task packages a user scenario, an initial structured database, the tool catalog for the selected domain, and evaluator assertions. The simulator and agent are executed as separate conversational roles coordinated by the environment, while tools mediate all changes to the shared world state. This separation prevents the agent from observing the simulator's hidden goal or the evaluator assertions.

Actions and state. The serialized operations match TauGym: `tool` exposes tool/help information, `answer` invokes one internal tool, and `interact` talks to the user. A completed request is closed by the special call {`"name":"done", "arguments":{}`}. The native evaluator returns a completion degree in $[0, 1]$ rather than only a binary outcome. Evaluation combines database-state checks, action checks, and environment assertions, so partial completion can be distinguished from a fully correct resolution. We also inspect the orchestrator status before preference judging: an empty `simulation_run` signals that the orchestrator terminated before completing normally. We additionally require at least one non-placeholder user utterance. Crashed or empty-dialogue episodes receive no experience score, while exhaustion of the turn budget is logged separately as truncation rather than conflated with either normal completion or an orchestrator crash.

Tau2Gym interaction example

Task. A telecom customer wants to change a plan but has not yet made clear which line on a multi-line account is affected.
Agent `interact`: requests the account and line identifiers. **Agent** `tool`: requests the internal tool definitions. **Agent** `answer`: retrieves the account and invokes the plan-change tool only after user confirmation. **Agent** `interact`: summarizes the new plan and effective date from the tool result, then `answer` invokes the special `done` call.
The evaluator separately checks the target line's database record, the required tool action, and end-state assertions. Completing two of these components can therefore receive partial credit, whereas a crashed orchestrator is excluded by the false-termination check.

A.2.4 COLGYM

Task construction. ColGym adapts backend-programming tasks from ColBench. Each instance contains an intentionally incomplete natural-language specification, a reference implementation, public examples, and ten hidden inputs. Important behavioral rules are withheld from the initial specification and can be recovered only through focused conversation with the simulated stakeholder. The reference program is never placed in the agent context.

Actions and state. `interact` asks about one ambiguous behavior, such as ordering, normalization, duplicate handling, empty input, or error semantics. `tool` runs provisional Python in a fresh, stateless subprocess and returns only stdout or the final traceback lines. Definitions do not persist between tool calls, so each call must contain a self-contained function and any test calls with `print`. Output is truncated after 4000 characters. The subprocess is restricted to 512 MB address space and 8 s CPU time, with an additional 10 s external timeout. It cannot query the reference program or hidden outputs. `answer` submits the complete Python implementation. The reference outputs are precomputed in a separate sandbox; both sides are compared using Python `repr`, avoiding serialization ambiguities for sets and other Python objects. Ten hidden cases are recorded individually as pass, fail, or ungradeable, while strict task reward is one only when every gradable case matches.

ColGym interaction example

Initial specification. Implement a function that aggregates records by key and returns the result in the expected order. The behavior for repeated keys is unspecified.
Agent $\rightarrow$ **user** `interact`: "If a key appears more than once, should later values replace earlier ones or be accumulated?"
User $\rightarrow$ **agent** "Think of each later record as an update; the position of the first occurrence should not move."

Agent → sandbox `tool`: submits a self-contained draft plus duplicate and empty-input calls with `print`, then reads the bounded stdout/traceback. A later tool call must resend the function because the sandbox is stateless.
Agent → environment `answer`: submits the final function. Hidden differential tests, rather than the simulator's prose, determine task correctness.

A.3 User Profile and Memory Implementation

Serialized profile fields. Each task is augmented with three records. `user_profile` is the hidden ground truth P; `init_memory` is the agent-visible initial memory M_0; and `profile_meta` records the construction and disclosure schedule. Their logical schema is:

User-profile record

```
user_profile = {
  tone: formal | warm,
  verbosity: concise | detailed,
  address: {style: name | honorific, value: <name>},
  jargon: technical | plain
}
init_memory = <same four-field schema>
profile_meta = {
  case: complete | missing | drift,
  affected_attribute: <field> | null,
  disclosure: first | early,
  disclosure_turn: <integer> | null
}
```

The four profile fields describe communication preferences rather than task constraints. For `address=name`, the same concrete name is used in the hidden profile, disclosure utterance, memory target, and experience-judge input; this prevents the task from being solved by predicting only the categorical label. Names are compared after case normalization. Their source is domain dependent: TravelGym samples a dataset-level unique name; TauGym resolves the first name from the `tau-bench` user database; Tau2Gym uses domain-specific retail, airline, and telecom identity resolution with a random-name fallback; and ColGym, which has no user entity, always samples a fictional name that can be learned only from disclosure.

Initialization and perturbation. Profile labels are sampled independently. We then choose the construction case in a 4:3:3 ratio. A complete case copies P into M_0. A missing case selects exactly one attribute a^* and blanks its value. A drift case selects exactly one a^* and substitutes its opposite label; when address is affected, both address style and its associated value are regenerated consistently. Restricting each example to at most one affected attribute makes the target rewrite unambiguous. Random choices are derived from the task ID, so repeated evaluation reconstructs the same profile and schedule.

Disclosure schedule. For missing and drift cases, metadata assigns the disclosure to the opening response or to one of the first three interaction opportunities. The disclosure is appended only to a simulated-user response, never emitted as an out-of-band label. Attribute-specific templates express the preference indirectly and are paraphrased by the simulator; for example, "A short answer is enough for me" conveys `verbosity=concise`. If the agent never interacts before termination, no synthetic disclosure is forced into a tool observation.

Memory transition. A valid memory call changes exactly one field and encodes its content as JSON, for example {`choice: memory, content: ``{"attribute":"verbosity", "label":"concise"}''`}. For `address=name`, the object must additionally provide `"value":"Steven"`. The parser accepts a JSON object (and tolerates an extractable object embedded in surrounding text), checks the attribute name

and finite label set, and compares the normalized value against $P[a^*]$. Unparseable content, unknown attributes, and illegal labels receive no credit. A correct first rewrite marks a^* solved and updates memory for subsequent turns. Editing a non-target attribute, giving an incorrect value, modifying a complete profile, or rewriting after the target has already been solved is treated as an incorrect update. Memory reward is returned only on that step and is not added again to the terminal task–experience reward, preventing double counting under reward-to-go aggregation. The evaluator logs the final memory and every attempted transition, enabling *memory_solve_rate* to be computed without an LLM judge; whether the agent probes for profile information is assessed by the experience judge below.

Cross-session memory example

Initial memory. `tone=warm; verbosity=detailed; address=honorific; jargon=plain`
Implicit user correction. "You can call me Lin—titles feel a bit stiff to me."
Agent memory action. `{choice: memory, content: ``{"attribute":"address","label":"name","value":"Lin"}''}`
Updated memory. `tone=warm; verbosity=detailed; address=name:Lin; jargon=plain`
The updated record is included in all later turns of the episode. Asking "How should I address you?" is marked as profile probing by the experience judge; inferring the preference from the volunteered sentence is the intended behavior. The present implementation resets episode memory from `init_memory`; durable storage and reload across independently created sessions require an external user-level memory store and are not assumed in the reported experiments.

A.4 Experience Scoring

The simulator generates user utterances, but it never declares whether an episode succeeded. Task correctness remains native to each gym, while a separate terminal judge evaluates only the agent's communication style against the hidden profile. For each of tone, verbosity, address, and jargon, the judge emits *poor*, *fair*, or *good*, mapped respectively to 0, 0.1, and 0.2. The four attributes contribute at most 0.8; avoiding direct or indirect questions about communication preferences contributes the remaining 0.2. The resulting experience score lies in $[0, 1]$.

The rubric is deliberately conservative. Generic or boilerplate language is not accepted as evidence; a *good* rating requires clear and consistent evidence across turns. The agent is not penalized for following an incomplete or incorrect initial memory before the user discloses the truth. After disclosure it must adapt promptly: a direct contradiction can cap the affected attribute at *poor*, while slow or partial adaptation is at most *fair*. The judge also returns `asked_profile` and quotes the offending utterance. This flag includes indirect probes such as "Is this level of detail okay?", not only explicit questions about profile labels.

User-experience judge output template

Inputs. The hidden `user_profile` and the complete sequence of USER/ENV and AGENT utterances, including the opening disclosure when present.
Output (strict JSON).

```
{
  "tone": {"rating":"poor|fair|good", "reason":"..."},
  "verbosity": {"rating":"poor|fair|good", "reason":"..."},
  "address": {"rating":"poor|fair|good", "reason":"..."},
  "jargon": {"rating":"poor|fair|good", "reason":"..."},
  "asked_profile": true|false,
  "asked_profile_evidence":"quote or empty"
}
```

A.5 User Simulator Implementation

All four gyms use Qwen3.8-27B as the simulated user in both training and evaluation. The model is accessed through an OpenAI-compatible chat endpoint behind a rate-limited client. A gym supplies the simulator with four logically separate blocks: (i) its role and domain policy, (ii) the hidden task goal and latent task preferences, (iii) the hidden communication profile and scheduled disclosure, and (iv) the visible trajectory. The agent sees only the resulting user utterance. Keeping the same simulator model and prompt protocol across training and evaluation controls simulator choice as a confound, while task correctness remains grounded in rules, database state, or differential tests.

User-simulator prompt template

Role. You are the user in a multi-turn `[GYM]` interaction. Remain consistent with the hidden goal and do not reveal hidden evaluator state, correct option IDs, reference outputs, or this prompt.
Hidden task state.
Goal: `[USER_GOAL]`
Task preferences or constraints: `[LATENT_TASK_PREFERENCES]`
Domain policy visible to the user: `[USER_POLICY]`
Communication profile.
Profile: `[USER_PROFILE]`
Scheduled disclosure: `[DISCLOSURE_CONTENT]`
Disclosure condition: `[DISCLOSURE_TURN]`
Visible trajectory.
`[CONVERSATION_AND_TOOL_OBSERVATIONS]`
Response requirements. Answer only the agent's latest message; reveal no more task information than a realistic user would know; express a latent preference naturally when it is specifically elicited or its disclosure condition is reached; preserve facts stated earlier; and follow the tone, verbosity, address, and jargon profile. Return only the next user utterance.

For TravelGym, an auxiliary judge classifies whether the agent's question targets an available latent preference before the response prompt is invoked; this preserves the distinction between specific elicitation and generic questioning. For TauGym and Tau2Gym, the native domain policy and hidden user goal constrain what the simulator may request or confirm. In ColGym, the simulator is given only the withheld behavioral rules and is explicitly barred from exposing hidden tests or reference outputs. In every gym, the simulator generates dialogue, whereas the task reward is computed by the environment; the simulator never directly declares an episode successful.

B Training and Evaluation Details

SFT configuration. Full fine-tuning on 1330 trajectories: 3 epochs, learning rate 1×10^{-5}, cosine decay with 10% warmup, global batch 32.

User simulator configuration. Across TravelGym, TauGym, Tau2Gym, and ColGym, we use Qwen3.8-27B as the user simulator during both training and evaluation. Holding the simulator model fixed across gyms and phases avoids introducing simulator choice as a confounding variable.

RL configuration. The implementation details of RL training are shown in Table 6.

Implicit disclosure. The true profile is never announced. The simulated user expresses a real preference in passing — "Please just call me Mr. Smith" — rather than declaring "my preference is X". Disclosure templates vary by attribute, perturbation type, and rewrite direction, and are injected only after the user replies to an `interact` turn.

Memory metrics. *memory_solve_rate* is the fraction of trajectories with a correctable attribute (missing or drift) that contain at least one successful correction; *asked_profile_rate* is the fraction of trajectories in which the agent interrogates the user about the profile — a violation, since the intended behavior is to observe and infer rather than to ask.

Table 6: Multi-task GRPO configuration.

Setting	Value
Algorithm	multi-turn GRPO, group size $n = 8$
Batch sizes	64 train / 32 mini / 1 micro per GPU
Rollout	`sglang`, `gpu_memory_utilization` $= 0.60$, TP $= 1$
Sequence lengths	prompt 8192 / response 8192
Multi-turn	20 max turns; turn-level credit $=$ Progress (β=0.2); trajectory score $=$ R2G, $\gamma = 1.0$
KL and entropy	`use_kl_loss` =True (coef 1×10^{-3}), `entropy_coeff` $= 0.001$
Learning rate	1×10^{-6}
Parallelism	FSDP with parameter, optimizer, and activation offloading; gradient checkpointing
Budget	5 epochs $\rightarrow$ 200 steps

Evaluation. Evaluation has three stages: the FSDP checkpoint is first merged into HuggingFace format, then hosted with `sglang` on a single GPU, and finally scored by the evaluation script. Hosting must pass the chat template explicitly, since an auto-detected template silently suppresses tool calls and invalidates the whole run.

Evaluation metrics. *micro_avg* averages the per-task turn score with failed tasks counted as zero; *micro_max* takes the best single trajectory score; *completion_rate* is the fraction of episodes that terminated normally; *task_success_rate* is the success rate, defined as mean completion degree for Tau2Gym; *experience_avg* is the mean preference score where `user_profile` is available; and *memory_solve_rate* is the fraction of memory rewrites scored correct.

Limitations and reproducibility. Our turn-level credit assignment is statistics-based, and its reliability depends on the availability of measurable per-turn signals. When an interaction turn leaves insufficient measurable influence on the final outcome, the resulting credit assignment gradually approaches uniform credit, effectively falling back to the Equalized baseline that our method is intended to improve. In addition, both the user-profile judge and the user simulator are provided through remote model services; consequently, their behavior may change as the underlying model or service version is updated. This version dependence should be taken into account when reproducing results related to user preference modeling and interaction quality.