UserRL: A Gym-Based Testbed for User-Centric Reinforcement Learning

Anonymous ARR submission

Abstract

Reinforcement learning (RL) has shown promise for training agentic models that engage in dynamic, multi-turn interactions. However, the value of such agents ultimately depends on their ability to assist users with diverse goals, preferences, and evolving interaction patterns. We introduce **UserRL**, a reproducible and extensible testbed for training and evaluating user-centric agents through standardized gym environments paired with simulated users. UserRL supports flexible reward specification by decoupling turn-level reward shaping from trajectory-level scoring, enabling controlled studies of credit assignment in multi-turn RL. Across Qwen3 models, our experiments show that SFT cold start provides an effective initialization for user-centric RL; trajectory-level reward-to-go scoring improves both effectiveness and interaction efficiency; and stronger simulated users improve training, while open-source simulators remain a cost-effective alternative with positive cross-simulator and preliminary real-user transfer. These results highlight reward design and user simulation as key to user-centric agent training, positioning UserRL as a practical foundation for interactive agent RL.

1 Introduction

Reinforcement learning (RL) has emerged as a powerful approach for training *agentic* large language models (LLMs), offering greater generalizability than supervised fine-tuning (Chu et al., 2025). This advantage is crucial for agents, which must operate in varied and unpredictable environments, solving tasks that often require extended reasoning and adaptation (Xi et al., 2023). A key enabler in this direction is multi-turn rollout, where the agent engages in multiple steps of interaction with its environment, producing rich trajectories that RL algorithms can exploit. For tool-using agents, multi-turn rollout is also necessary: many real tasks demand sequences of tool calls, intermediate reasoning steps, and iterative refinements, rather than solvable in a single step (Deng et al., 2024b; Qian et al., 2025a).

Recent progress combining RL with multi-turn rollouts has produced increasingly capable agents (Wang et al., 2025; Zeng et al., 2025). These systems can coordinate multiple tool uses, navigate diverse domains such as web, code and games, and execute multi-stage reasoning pipelines (Deng et al., 2024a; Qian et al., 2024a; Zhu et al., 2025). Examples include iterative searches for QA (Jin et al., 2025), stepwise debugging for programming (Golubev et al., 2025), and staged decision-making for complex browsing tasks (Lù et al., 2024). Such abilities mark a significant step toward agents that function as general problem-solvers.

However, an agent's ultimate value is not determined by its performance in abstract benchmarks or closed environments, but by its effectiveness in assisting **users**. Whether the context is scientific research, professional analysis, or everyday information gathering, the agent's role is collaborative (Baek et al., 2024; Sun et al., 2025; Zhang et al., 2025). This reframing shifts the perspective: the user is not merely a goal-setter or evaluator, but an integral and dynamic part of the agent's operating environment. The most capable agent is one that can understand, adapt to, and actively support the user (Qian et al., 2024b; Lu et al., 2025).

Placing robust user assistance at the center exposes two critical interaction traits for training:

- **Diversity**: User behavior is heterogeneous and shaped by individual preferences, goals, and communication styles. This diversity demands that agents master a broad range of interaction skills.
- **Dynamics**: User interaction unfolds over multiple turns and can shift in constraints as the conversation progresses, making pre-collected datasets unable to capture evolving patterns.

These traits, while essential to real-world assis-

tance, create challenges for model training: there is no standardized framework to represent diverse user abilities, and it is difficult to simulate realistic, dynamic interactions within existing training pipelines. This leads to our core research question: **How can we train and evaluate agentic models that effectively acquire user-centric abilities, while accounting for the inherent diversity and dynamics of user interactions?**

To address *diversity*, we build a **unified suite of user-centric gym environments**, each targeting a distinct interaction skill and supporting both benchmarking and RL training. A standardized interface and configurable rewards make each gym easy to adapt and extend to new scenarios. To address *dynamics*, we couple **multi-turn RL rollouts with LLM-based user simulation**, allowing the agent to interact with adaptive, context-aware users during both training and evaluation. These design choices together define **UserRL**: a general, extensible testbed for user-centric RL, instead of merely a collection of isolated benchmarks or a single prescribed reward design.

Empirically, we use UserRL to study supervision and credit assignment choices under GRPO (Guo et al., 2025). Because each gym provides structured feedback and verifiable outcomes, our testbed supports controlled comparisons of reward and credit assignment in interactive rollouts along two axes: (1) how to aggregate a trajectory-level score and (2) how to assign turn-level rewards. Across 4B and 8B Qwen3 models, we find that trajectory-level scoring has a larger effect than fine-grained turn differentiation. SFT cold start provides a strong initialization and prevents early RL plateaus, while SFT+RL achieves the best final performance in our ablations. We also find that stronger simulated users can improve training, but models trained with cost-effective open-source simulators still retain gains when evaluated with stronger simulated users and in a preliminary real-user study. Overall, these findings illustrate the kinds of controlled analyses UserRL is designed to enable. We summarize our contributions as follows:

- We introduce UserRL, a unified suite of eight user-centric gyms with LLM-simulated users for multi-turn training and benchmarking.
- We provide a standardized tool interface and modular components that enable plug-and-play extension (gyms, simulators, rewards).
- Using UserRL, we report comprehensive empirical observations including credit structure, initialization, and simulator choice, etc.

2 Gym Construction

Gymnasium environments have long supported RL training by offering clean, reproducible agent–environment interfaces (Towers et al., 2024). Building on this foundation, we introduce eight standardized gym environments, each emphasizing a distinct aspect of user interaction. This design supports both training and benchmarking while capturing key nuances of real-world interaction.

General Gym Components. Each gym environment is built around two core components: the *task* and the *user*. From the **task perspective**, the environment can be seen as a finite automaton. After initialization through the `reset()` function, the environment transitions step-by-step based on agent actions, where each `step()` updates the internal state according to deterministic, rule-based transition functions. Along with the new state, the environment emits rewards that reflect whether progress toward task completion has been made. This structure ensures that the evaluation remains rigorous, transparent, and reproducible.

From the **user perspective**, certain agent actions are interpreted as input to the simulated user. The environment then returns a response generated by a LLM, which acts as the user simulator. Depending on the specific task, this feedback may take the form of a conversational utterance, a preference judgment, or an answer to a posed query. By employing LLMs, the user responses remain dynamic and contextually adaptive, while the underlying task completion remains strictly rule-based. This dual design introduces both the rigor of formal evaluation and the richness of natural user interaction.

Standardized Tool Interface. A key innovation in our construction is the *standardized tool interface*, which governs how agents interact with the environment. As shown in Figure 1, this interface reduces all interactions to three core operation types: `Action`, `Search`, and `Answer`.

- **Action:** Direct communication with the simulated user. The agent provides utterances as input, and the gym returns user responses.
- **Search:** Retrieval of external knowledge. The agent issues a search query, and the gym interacts with a backend to provide retrieved content.
- **Answer:** Submission of a candidate solution. The agent provides a answer, and the gym verifies

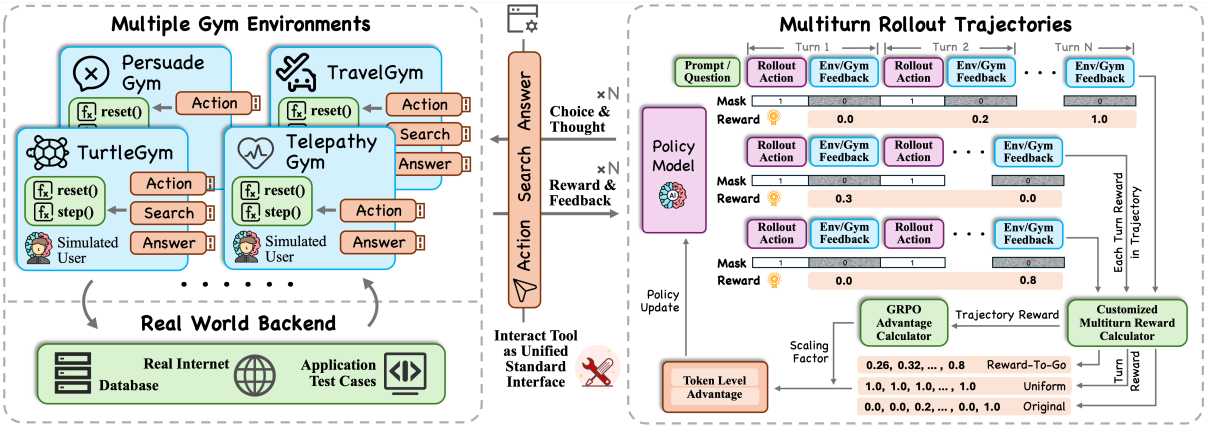

Figure 1: UserRL framework. *Left*: example gym environments each with a simulated user and standardized interface. *Right*: during training, the policy performs online multi-turn rollouts through the interface, producing actions and receiving feedback plus turn-level rewards from the gym. After each trajectory forms, a reward calculator maps the rollout to (i) a trajectory-level score used for advantage estimation and (ii) turn-level rewards, which are scaled and aggregated into token-level advantages for policy updates.

correctness if the task is goal-oriented.

This interface is deliberately minimal yet expressive: it captures the essential modes of agent behavior, keeps implementation simple, and allows straightforward extensibility. Importantly, the set of available options may vary by environment. For instance, TurtleGym permits all three operation types, while PersuadeGym restricts agents to `Action`, as persuasion tasks lack verifiable answers and completion is defined by user attitude change.

Specific Gym Designs. We design eight distinct gym environments, each targeting a different aspect of agent ability. The details are summarized in Table 1. Some gyms are newly curated, while others adapt existing benchmarks under our unified interface, demonstrating both the flexibility and extensibility of our framework. These environments collectively test a spectrum of user-centric skills: from intent understanding and persuasive communication, to personalized planning and tool use. They also probe different reasoning capacities, including creative, strategic, and mathematical reasoning. Taken together, our gyms provide a principled yet dynamic platform for advancing agent training and evaluation, ensuring that agents are not only competent in abstract tasks but also genuinely helpful in supporting users. For additional construction details, user simulation prompts and task design logic, please refer to Section C.

3 UserRL Framework

Alongside the gym environments, we release a reinforcement-learning *training scaffold* for user-centric, multi-turn interaction. As shown in Figure 1 (right), UserRL adopts GRPO as the underlying optimizer and exposes *modular hooks* for (i) assigning turn-level rewards during a rollout and (ii) computing a trajectory-level score for advantage estimation. Rather than prescribing a single reward scheme, the framework provides a unified interface that allows different credit-assignment and aggregation strategies to be compared under the same training loop, supporting empirical study and task-specific customization.

Multi-Turn Rollouts and Notation. Let π_θ denote the policy and T the number of interaction turns. A *multi-turn rollout trajectory* is

$$\tau = \{(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots, (s_T, a_T, r_T)\},$$

where $a_t \sim \pi_\theta(\cdot \mid s_t)$, the environment transitions to s_{t+1}, and emits a *turn reward* r_t. Each turn t corresponds to a generated sequence of tokens $x_t = (x_{t,1}, \ldots, x_{t,L_t})$.

Motivation. In multi-turn interaction, feedback is naturally incremental: some turns are exploratory, others solve subgoals, and later turns may finalize the outcome. Our gyms provide turn-level rewards, making it possible to use these dense signals for assigning credit across turns. This motivates a framework that can flexibly redistribute supervision and evaluate different hypotheses about what matters most in user interaction.

Turn-Level Reward Shaping. The gym produces raw turn rewards $\{r_t\}_{t=1}^{T}$. Before broadcasting them to tokens, we transform them into turn-level signals $\{\tilde{r}_t\}_{t=1}^{T}$:

$$R(x_{t,k}) = \tilde{r}_t, \quad \forall k \in \{1, \ldots, L_t\}.$$

This ensures that tokens within a turn share the same signal, while different turns can be treated differently depending on the chosen scheme. We experiment with the following reward shapings:

Gym Name	Description	Data Source	Capability	Interface
IntentGym	Reveal user's real intent from agentic tasks	IN3 (Qian et al., 2024b)	Intent understanding, ambiguity resolution	Action: ask clarifying question
TurtleGym	Play turtle soup game with user to reveal hidden twists	Newly curated	Creative reasoning, contextual adaptation	Action: inquire story details; Answer: unveil the hidden story
PersuadeGym	Persuade user with opposing claims and arguments	Persuasion (Durmus et al., 2024)	Strategic reasoning, persuasive communication	Action: provide persuasive arguments
TelepathyGym	Guess out what entity user is thinking	Newly curated	Strategic reasoning, hypothesis testing	Action: interact for clues; Answer: guess out the entity
FunctionGym	Reveal user's hidden mapping rule for a set of numbers	Newly curated	Math reasoning, pattern generalization	Action: get number's mapping results; Search: retrieve test case; Answer: give test case result
TravelGym	Help user make personalized travel booking	UserBench (Qian et al., 2025b)	Preference elicitation, personalized planning	Action: interact for preferences; Search: retrieve travel options; Answer: provide travel recommendation
TauGym	Fulfill user requirements through tools and conversation	Tau-Bench (Yao et al., 2024)	Tool use, task-oriented interaction	Action: interact for user details; Search: retrieve available tools; Answer: perform tool call for task solving
SearchGym	Search and answer general user questions	Bamboogle (Press et al., 2022)	General helpfulness and reasoning	Search: retrieve online information; Answer: respond to user query

Table 1: Eight gym environment details including data source, tested capability, and interface design.

- **Naive:** $\tilde{r}_t = r_t$, rewards remain unchanged, but in practice this often makes them too sparse, which quickly leads to training collapse.
- **Equalized:** $\tilde{r}_t = c$, a constant reward is assigned to every turn, effectively treating them all equally. This mirrors the approach used in the original GRPO, where each turn's reward is uniform.
- **Reward-to-Go (R2G):** Each turn accumulates discounted future rewards:

$$\tilde{r}_t = \sum_{j=t}^{T} \gamma^{j-t} r_j, \quad \gamma \in [0,1].$$

- **Exponential Mapping (EM):** A nonlinear rescaling of $r_t \in [0,1]$ into $[0.5,1]$:

$$\tilde{r}_t = \phi_k(r_t) = 0.5 + 0.5 \cdot \frac{1-\exp(-kr_t)}{1-\exp(-k)}, \quad k > 0.$$

Insight. These shaping rules impose different turn-level credit biases: Equalized treats turns uniformly, R2G propagates feedback backward to earlier actions, and EM preserves small progress under sparse rewards. UserRL keeps this choice modular, enabling straightforward comparisons without changing the optimizer or environments.

Trajectory-Level Scoring. GRPO requires a single scalar trajectory score for group normalization. As our gyms produce incremental feedback, we define two strategies:

- **Sum:** As turn rewards reflect incremental gains, summing naturally recovers total progress.
- **Reward-to-Go (R2G):** This variant encourages efficient strategies that achieve progress earlier within fewer turns.

$$R_{\text{traj}}^{\text{sum}}(\tau) = \sum_{t=1}^{T} r_t, \qquad R_{\text{traj}}^{\text{r2g}}(\tau) = \sum_{j=1}^{T} \gamma^{j-1} r_j.$$

Insight. Trajectory scoring collapses incremental feedback into the scalar used for GRPO normalization. Sum reflects total progress, while R2G discounts late rewards and favors earlier gains. UserRL supports both to study how aggregation affects learning and interaction behavior.

Grouped Advantage Estimation and Objective. For each query Q, a rollout group $G_Q = \{\tau^{(i)}\}_{i=1}^{n}$ is collected. Using a chosen trajectory scorer R_{traj}, we compute:

$$\mu_Q = \frac{1}{n}\sum_{i=1}^{n} R_{\text{traj}}(\tau^{(i)}), \quad \sigma_Q = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(R_{\text{traj}}(\tau^{(i)}) - \mu_Q\right)^2}.$$

Each token $x_{t,k}$ in trajectory i is then assigned a normalized advantage:

$$A(x_{t,k}|Q) = \frac{\tilde{r}_t^{(i)} - \mu_Q}{\sigma_Q + \eta}, \quad \eta > 0.$$

Objective. Our design preserves GRPO-style group normalization over comparable rollouts, while keeping the definitions of turn-level rewards and trajectory-level scores modular. This separation lets us instantiate and compare different credit-assignment biases under a consistent optimization pipeline. Finally, the policy is trained with the clipped PPO objective, adapted to our advantage formulation. We omit the KL loss in original objective to encourage exploration and alignment to our new interface.

Summary. UserRL does not introduce a new RL algorithm; rather, it provides a benchmark-integrated abstraction for studying credit assignment in user-centric multi-turn RL. By decoupling turn-level reward shaping from trajectory-level scoring within the same gym suite, simulator protocol, and GRPO-style training loop, UserRL makes reward formulation a controlled experimental variable across

Gym	Travel	Turtle	Function	Tau	Persuade	Intention	Telepathy	Search	Total
Train Num.	925	423	460	500	378	380	360	0	**3426**
Test Num.	471	48	78	165	42	40	41	125	**1010**
Metric	Choice Correctness (Follow UserBench)	Sum Turn Reward	Test Case Correctness	Task Completion (Follow Tau-Bench)	Sum Turn Reward	Sum Turn Reward	Final Guess Correctness	Final Answer Exact Match	-

Table 2: Statistics of eight gym environments, including data point numbers and metrics for evaluation.

heterogeneous interaction tasks. This design is empirically useful: an intermediate question may not receive immediate reward, but can still enable later task success. The framework therefore allows researchers to compare how different credit structures weight such intermediate behavior under matched environments and optimization settings.

4 Experiments

Building on the gyms and training scaffold in UserRL, we study which design choices most affect learning in user-centric, multi-turn tasks. We focus on two configurable components exposed by our framework: *trajectory-level scoring* and *turn-level reward shaping*, which are made possible by the dense, turn-wise feedback available in gym environments. This setup enables controlled comparisons of reward structures and their effects on policy learning in interactive rollouts.

4.1 Experiment Settings

Settings. We evaluate four representative configurations, denoted as A/B where A refers to the turn-level reward shaping method and B refers to the trajectory-level scoring method: Equalized/-Sum, Equalized/R2G, EM/R2G, and R2G/R2G. The *Naive* setting is excluded because in practice its sparse effective reward signal, where many turns yield zero reward, quickly leads to training collapse. Among these variants, Equalized/Sum serves as the most natural extension of the original GRPO algorithm into the multi-turn setting, treating every turn equally while summing the incremental rewards to produce the trajectory score. From this baseline, we vary either the turn-level reward or the trajectory-level score to enable fair comparison across methods.

Training. All RL models are initialized with an SFT cold start, which stabilizes optimization and improves downstream RL performance; we analyze this effect in later section. We employ VERL (Sheng et al., 2024) for RL training. Each step samples a batch of 128 with 8 responses per query, training for 15 epochs in total (see Section E for full configuration). To encourage exploration, we remove KL regularization and use temperature 1.0. For all gyms, we employ Qwen3-32B as the simulated user model, set max turns to 16, and keep other settings as in Section C.

Data. We use TravelGym, TurtleGym, FunctionGym, TauGym, and PersuadeGym for training, while reserving IntentionGym, TelepathyGym, and SearchGym as held-out evaluation environments. This ensures test-time evaluation requires generalization to unseen interaction purposes. For the training split, we also retain in total 1K trajectories from the five training gyms as supervised fine-tuning data. These trajectories are distilled using GPT-4o as both agent and simulated user, providing consistent high-quality supervision for SFT initialization. All remaining trajectories are then used for RL training. Please refer to detailed statistics in Table 2. In Section G, we additionally test robustness to the gym train/test composition by retraining under an alternative split and evaluating held-out gyms.

Model and Metrics. We conduct experiments with Qwen3 models of 4B and 8B, and include larger variants (14B, 32B) for raw performance comparison. For closed-source baselines, we report GPT and Gemini families as references. Evaluation metrics follow each gym's definition: for TravelGym and TauGym, we adopt UserBench and Tau-Bench protocols and score by correctness of the final choice or state; for others, the score is the sum of turn rewards since each turn reflects incremental gain (equivalent to the metric definitions in Table 2). In addition to per-gym results, we report micro-averaged performance across all eight gyms for overall comparison.

4.2 Experiment Results

We present our main results in Table 3. The key findings are presented in the following.

Equalized/R2G generally outperforms other training settings. The *Equalized/R2G* setting achieves the highest average performance across all tasks for both 4B and 8B models, while *Equalized/-Sum* almost always performs worst. This highlights the importance of trajectory-level score calculation: R2G outperforms simple summation because it better captures the cumulative value of intermediate steps toward reward-yielding turns. Importantly, zero reward does not imply zero contribution: for

Model	TravelGym	TurtleGym	FunctionGym	TauGym	PersuadeGym	*IntentionGym*	*TelepathyGym*	*SearchGym*	Avg.
Open-Source (Trained Model)									
Qwen3-8B (Equalized/R2G)	**0.5730**	0.1854	**0.4231**	0.1818	0.5317	1.8175	0.5610	0.8880	**0.5652**
Qwen3-8B (EM/R2G)	0.5025	0.1917	0.4103	0.2000	**0.5397**	**1.9025**	0.5366	0.8640	0.5343
Qwen3-8B (R2G/R2G)	0.5724	0.1615	**0.4231**	0.1394	0.5238	1.8525	0.5854	0.8480	0.5539
Qwen3-8B (Equalized/Sum)	0.5054	0.1323	0.2692	**0.2121**	0.5040	1.6275	0.5366	0.8320	0.5076
Qwen3-4B (Equalized/R2G)	0.5086	0.1844	0.3333	0.2000	0.4643	1.8075	0.6098	0.8640	0.5269
Qwen3-4B (EM/R2G)	0.5076	0.1417	0.3333	0.1576	0.4563	1.7375	0.6341	0.8640	0.5154
Qwen3-4B (R2G/R2G)	0.4629	0.1687	0.3974	0.1333	0.5794	1.5975	0.4634	0.8640	0.4895
Qwen3-4B (Equalized/Sum)	0.4456	0.1615	0.2308	0.1333	0.4524	1.7150	0.4878	0.8400	0.4656
Open-Source (Raw Model)									
Qwen3-32B (Raw)	0.1724	0.1510	0.1538	0.0000	0.4841	1.8300	0.5610	0.7920	0.3128
Qwen3-14B (Raw)	0.1924	0.1417	0.1667	0.1030	0.5317	1.7000	0.5854	0.5120	0.3027
Qwen3-8B (Raw)	0.1689	0.1135	0.1474	0.1090	0.5190	1.7150	0.5122	0.7920	0.3060
Qwen3-4B (Raw)	0.1405	0.0854	0.0769	0.0364	0.4048	1.7400	0.4878	0.8560	0.2929
Closed-Source									
Gemini-2.5-Pro	0.3468	0.2740	0.4103	0.1939	0.4246	1.5900	**0.9024**	**0.9280**	0.4702
Gemini-2.5-Flash	0.2553	0.1958	0.3205	0.1212	0.4087	1.6850	0.6341	**0.9280**	0.3973
GPT-4o	0.3643	**0.2917**	0.2821	0.0303	0.3770	1.8975	0.8537	0.8800	0.4449
GPT-4o-mini	0.0976	0.0906	0.1538	0.2061	0.5317	0.2500	0.0488	0.3520	0.1729

Table 3: The main evaluation results on gym environments. For each gym, the highest performance within section is marked with underline, while the global best performance is highlighted in **bold**.

example, in *TelepathyGym*, asking clarifying questions earns no reward but helps narrow the answer space and thus supports final success. Assigning such turns zero advantage undermines learning, which explains the collapse observed under naive turn-level reward training. Finally, we find that the choice of turn-wise reward assignment (Equalized, EM, Distance-based) has less impact on performance and the simple Equalized scheme already suffices, suggesting trajectory-level scoring is more decisive than fine-grained turn differentiation. We further discuss this in Section 5.

Gym-trained models are strong under the UserRL protocol. Under the standardized UserRL protocol, trained Qwen3 models achieve strong overall performance and outperform closed-source reference models on several gyms, including TravelGym, PersuadeGym, and IntentionGym. The result should not be read as a general claim that smaller trained open-source models are superior. Instead, it shows that targeted user-centric training can substantially improve performance within a controlled interaction interface. Closed-source models remain stronger on gyms such as TelepathyGym and SearchGym, which require strategic guessing and external information use, suggesting that user-centric performance depends on both interaction adaptation and model capabilities.

Scaling raw model size is less effective without interaction training. We observe that scaling raw Qwen3 models alone yields only marginal and inconsistent gains under the UserRL protocol: larger raw models improve on some gyms but do not reliably close the gap to interaction-trained models. By contrast, RL in our environments unlocks improvements brought by scaling: for 4B and 8B models trained under the best setting, their performance gap can surpass that of the raw 4B and 32B models. This shows scaling effects emerge most clearly after adaptation to user-centric interaction, suggesting foundational capacity alone is insufficient unless paired with training that elicits effective communication and user adaptability.

Adapting existing benchmarks to user-centric settings can reduce performance. In *TravelGym* and *TauGym*, we observe that models perform far worse than in the original UserBench (Qian et al., 2025b) and Tau-Bench (Yao et al., 2024), even with unchanged test data and metrics. Our only modification is the introduction of a standardized tool interface to mediate user interactions. This result suggests two insights. First, interface design can substantially affect measured agent performance, even when the underlying test data and metrics are unchanged. Second, interacting correctly through standardized tools remains challenging for current models, indicating that user-centric abilities such as structured communication, preference elicitation, and consistent tool use are not yet robust.

4.3 Analysis

Effectiveness of SFT cold start. To validate our choice of using SFT cold start, we further trained raw Qwen3 4B and 8B models under the *Equalized/R2G* setting, which gave the best results in Table 3. As shown on the left of Figure 2, SFT cold start consistently allows RL to boost performance, sometimes even exceeding 100% gains. The train-

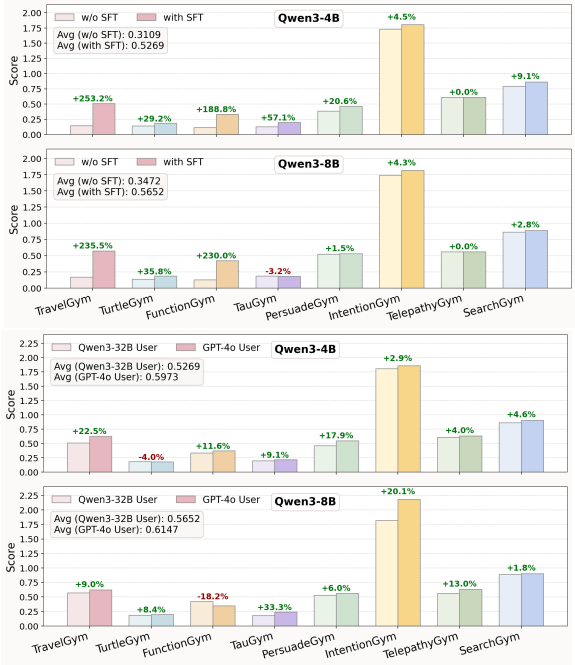

Figure 2: **Top:** SFT cold start improves RL training performance compared to direct RL on raw Qwen3 models. **Bottom:** GPT-4o as simulated user during training yields better downstream performance than Qwen3-32B as simulated user in gym environments.

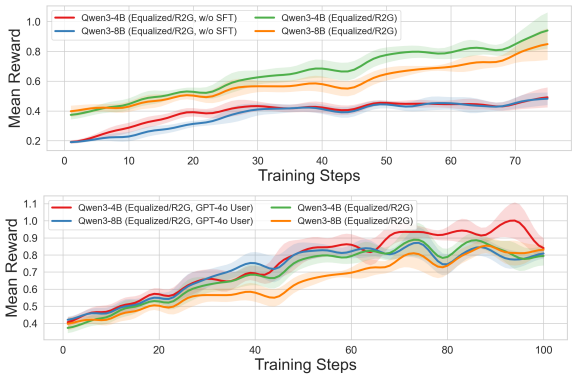

Figure 3: Comparisons of Qwen3 training curves under the Equalized/R2G setting: **Top:** w/ vs. w/o SFT cold start and **Bottom:** using GPT-4o vs. Qwen3-32B as the simulated user.

ing curves on the left of Figure 3 also show that SFT-initialized models begin from a stronger baseline and keep improving, while models without SFT plateau early at lower levels. Therefore, we conclude that SFT cold start equips the models with essential initial user interaction abilities, which RL can then better refine and extend. In Section G, we further compare raw vs. SFT-only vs. SFT+RL settings to clarify what SFT contributes versus RL.

Choice of user simulation model for training. In all the main experiments, we used Qwen3-32B as the simulated user for cost reasons, while evaluation always employed GPT-4o (detailed justifications in Section F). To probe the effect of simulator choice, we additionally trained Qwen3 4B and 8B

Model	Effective Turns$^{\uparrow}$	Time-Weighted Performance$^{\uparrow}$
Qwen3-8B (Equalized/R2G)	**6.6463**	**0.6516**
Qwen3-8B (EM/R2G)	5.8792	0.5586
Qwen3-8B (R2G/R2G)	5.1050	0.6510
Qwen3-8B (Equalized/Sum)	6.1842	0.4530
Qwen3-4B (Equalized/R2G)	6.1307	0.6423
Qwen3-4B (R2G/R2G)	5.7317	0.6213
Qwen3-4B (EM/R2G)	5.6743	0.5355
Qwen3-4B (Equalized/Sum)	5.3881	0.5118
Qwen3-32B (Raw)	2.8079	0.2852
Qwen3-14B (Raw)	3.4129	0.3211
Qwen3-4B (Raw)	2.2545	0.2674
Gemini-2.5-Pro	5.7731	0.5263
Gemini-2.5-Flash	4.2465	0.4525
GPT-4o	3.4087	0.4024
GPT-4o-mini	1.9461	0.1614

Table 4: Gym-based RL trained models has more effective turns on average, while gaining non-zero reward more efficiently.

with GPT-4o as the simulated user under *Equalized/R2G* setting. As shown on the right of Figure 2 and Figure 3, GPT-4o as simulated user generally yields higher performance, shows faster growth and higher plateaus (albeit subtly). This improvement likely arises from the alignment of interaction patterns in training and evaluation, which enables the model to adapt more effectively. These results suggest a practical tradeoff: stronger simulated users can improve training, but budget-friendly simulators such as Qwen3-32B still produce policies that retain gains when evaluated with a stronger GPT-4o user. We view this as evidence of cross-simulator transfer within the UserRL protocol. We further discuss effective user simulation in Section 5.

Effectiveness and efficiency of user interaction. cross all settings, we first measure the average effective turns for each evaluated gym, defined as the number of turns a model takes before obtaining its last non-zero turn-wise reward. This metric captures whether the model can meaningfully leverage interaction turns to achieve reward. As shown in the left column of Table 4, our trained models generally exploit more interaction turns, while raw Qwen3 models often stop gaining reward after just two or three turns. Nevertheless, the evaluation maximum is set to 16 turns, and even the best model achieves only 6.6463 effective turns on average, suggesting much room for improvement. Building on the interaction efficiency definition in Qian et al. (2025b), we further conduct weighted-timing analysis, where the reward at turn i is scaled by a weight of $1/(i+1)$, and the final metric is the sum of all weighted rewards. This emphasizes early rewards, encouraging models to achieve gains quickly rather than delaying progress. As shown in the right column of Table 4,

Model / User	Turtle	Telepathy	Persuade	Travel
Trained / GPT-4o	0.1844	0.6098	0.4643	0.5025
Trained / Real	0.2631	0.7073	0.5476	0.5144
Raw / GPT-4o	0.0854	0.4878	0.4048	0.1444
Raw / Real	0.1042	0.5366	0.4524	0.1522

Table 5: Real-user evaluation on four UserRL gyms. We compare trained and raw Qwen3-4B models under both GPT-4o simulated users and real users.

trained models consistently score higher, with *R2G*-based models outperforming those using *Sum* as the trajectory score. This validates that the *R2G* design not only improves effectiveness but also fosters efficiency by incentivizing earlier successes in interaction. We further elaborate on this issue of interaction efficiency in Section 5.

Interaction with real users. To examine whether the gains observed with simulated users remain visible under human interaction, we conduct real-user evaluation on four gyms: TurtleGym, TelepathyGym, PersuadeGym, and TravelGym. We compare both the UserRL-trained Qwen3-4B model and its raw counterpart under the same protocol, using GPT-4o simulated users and real users as two evaluation conditions. As shown in Table 5, real users often provide slightly more informative interaction than GPT-4o users, but the raw model gains only modestly from this additional cooperation and does not close the gap to the trained model. This suggests that the improvement is not solely explained by pre-existing base-model capability or by simulator-only evaluation. At the same time, we treat this as limited but direct evidence rather than a definitive proof of human robustness: the human study covers four tasks and a modest participant pool, and broader population-scale validation remains future work. Additional protocol details and participant information are provided in Section F.3.

Additional diagnostics. Beyond the main results, we include additional diagnostics in Section G to clarify the interpretation of UserRL training. First, Section G.1 tests robustness to the train/test gym composition and shows that performance remains comparable under an alternative split. Second, Section G.2 compares raw, SFT-only, and SFT+RL models, showing that SFT provides a strong interaction foundation while RL further improves performance under the online multi-turn reward structure. Third, Section G.3 evaluates MATH500 and LiveCodeBench and finds that UserRL-trained models retain over 95% of the raw baselines, suggesting limited trade-off with general capabilities. Finally, Section G.4 and Section G.5 examine two possible artifacts: raw and closed-source models already achieve high format-compliance rates, while overt simulator-specific exploitation is uncommon in sampled trajectories. Together, these diagnostics indicate that UserRL gains are not primarily explained by gym partition choice, SFT alone, general-capability degradation, output parseability, or obvious simulator shortcuts.

5 Discussions

Turn-level signals: what we learn and what they enable. UserRL exposes a central limitation of outcome-based credit assignment: simple mappings such as EM and R2G provide usable signals, but often confuse progress with action quality. EM collapses many unrewarded turns, while R2G overvalues turns near reward events, making informative questions hard to distinguish from shallow probing. This suggests that multi-turn RL needs richer turn-level supervision—such as relevance, information gain, or verifier-graded question quality. UserRL makes this problem measurable by allowing such evaluators to be compared under the same optimizer and environments.

Balance of rigor and flexibility in user simulation. User simulation enables scalable multi-turn RL, but it creates a core tradeoff: realistic users require diverse behaviors, while rigorous benchmarks require controlled and comparable interactions. UserRL addresses this by combining LLM-generated responses with rule-based task tracking, as in TravelGym, where user replies are guided by classified agent intents. This design preserves fairness while allowing limited interactional diversity, and points to a broader challenge: building simulators that are both natural enough for robust training and structured enough for reliable evaluation.

6 Conclusion

In this paper, we presented UserRL, a reproducible testbed for user-centric, multi-turn RL. By unifying eight gym environments under a standardized tool interface, UserRL enables controlled benchmarking and training across diverse interaction skills. Our GRPO experiments show that trajectory scoring and turn-level reward shaping meaningfully affect learning, with early real-user results suggesting simulator gains can transfer beyond simulated evaluation. More broadly, UserRL provides a foundation for studying agents that learn to clarify, adapt, and collaborate with users over time.

Limitation

UserRL is a controlled and extensible testbed for user-centric agent RL, and its current scope reflects this design goal. LLM-based simulated users enable scalable and reproducible multi-turn training, but cannot fully capture the diversity of real users; our real-user evaluation and simulator-audit diagnostics provide encouraging evidence beyond simulator-only settings, while larger-scale human studies remain future work. The eight gyms cover representative interaction capabilities rather than all forms of human-agent collaboration, and future extensions can add richer user profiles, longer-horizon tasks, and more complex domains. Finally, UserRL is intended as a benchmark-integrated framework for studying reward design and credit assignment, not a new RL algorithm. Its results highlight the complementary roles of SFT and RL: SFT provides a strong initialization, while RL further adapts agents to online multi-turn feedback.

Impact Statement

This paper introduces UserRL, a reproducible testbed for studying and training user-centric, multi-turn RL agents. Our goal is to advance the methodology for evaluating and improving interactive assistance, which may benefit research on more reliable and helpful AI systems. The work is primarily conducted in simulated gym environments; no personally identifiable data is used, and any human evaluation involves consenting participants. Potential risks include biases introduced by simulated users and the possibility that improved interactive agents could be misused (e.g., for manipulation or deceptive dialogue). We mitigate these risks by emphasizing transparent, standardized environments and reporting results across both simulated and real-user settings, enabling the community to audit behaviors and develop safer training and evaluation practices.

References

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2024. Researchagent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*.

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

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

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, and 1 others. 2025b. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024a. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024b. On the multi-turn instruction following for conversational web agents. *arXiv preprint arXiv:2402.15057*.

Esin Durmus, Liane Lovitt, Alex Tamkin, Stuart Ritchie, Jack Clark, and Deep Ganguli. 2024. Measuring the persuasiveness of language models.

Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. 2024. Aligning llm agents by learning latent preference from user edits. In *Advances in Neural Information Processing Systems*.

Alexander Golubev, Maria Trofimova, Sergei Polezhaev, Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Sergey Abramov, Andrei Andriushchenko, Filipp Fisin, and 1 others. 2025. Training long-context, multi-turn software engineering agents with reinforcement learning. *arXiv preprint arXiv:2508.03501*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *Preprint*, arXiv:2406.04770.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, and 1 others. 2025. Proactive agent: Shifting llm agents from reactive responses to active assistance. In *The Thirteenth International Conference on Learning Representations*.

Shuhaib Mehri, Xiaocheng Yang, Takyoung Kim, Gokhan Tur, Shikib Mehri, and Dilek Hakkani-Tür. 2025. Goal alignment in llm-based user simulators for conversational ai. *arXiv preprint arXiv:2507.20152*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025a. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, and 1 others. 2024a. Escapebench: Pushing language models to think outside the box. *arXiv preprint arXiv:2412.13549*.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024b. 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*.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, and 1 others. 2025b. Userbench: An interactive gym environment for user-centric agents. *arXiv preprint arXiv:2507.22034*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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 Rémi Munos. 2024. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*.

Hao Shen, Pengcheng Liu, Jiarun Li, Chenyou Fang, Yifan Ma, Jing Liao, Qiu Shen, Zhili Zhang, Kai Zhao, Qi Zhang, and 1 others. 2025. VLM-R1: A stable and generalizable R1-style large vision-language model. *arXiv preprint arXiv:2504.07615*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Harmanpreet Singh, Nikhil Verma, Yixiao Wang, Manasa Bharadwaj, Homa Fashandi, Kevin Ferreira, and Chul Lee. 2024. Personal large language model agents: A case study on tailored travel planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. 2025. Lambda: A large model based data agent. *Journal of the American Statistical Association*, pages 1–13.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. 2016. Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, and 1 others. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

Jiayin Wang, Fengran Mo, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024a. A user-centric multi-intent benchmark for evaluating large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2024b. MINT: Evaluating llms in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, 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. 2025. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. *arXiv preprint arXiv:2505.11821*.

Michael J. Q. Zhang, W. Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. *Preprint*, arXiv:2410.13788.

Weizhi Zhang, Yangning Li, Yuanchen Bei, Junyu Luo, Guancheng Wan, Liangwei Yang, Chenxuan Xie, Yuyao Yang, Wei-Chieh Huang, Chunyu Miao, and 1 others. 2025. From web search towards agentic deep research: Incentivizing search with reasoning agents. *arXiv preprint arXiv:2506.18959*.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025a. Do llms recognize your preferences? evaluating personalized preference following in llms. In *International Conference on Learning Representations*.

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

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and 1 others. 2025. Multiagentbench: Evaluating the collaboration and competition of llm agents. *arXiv preprint arXiv:2503.01935*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. 2025. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

Appendix

A Related Works

User-centric agent design and evaluation. As LLM agents are deployed in everyday settings, recent work has studied how to evaluate and improve assistance under underspecified goals, evolving preferences, and interactive feedback. On the evaluation side, benchmarks grounded in real user interactions capture multi-intent or in-the-wild queries (Wang et al., 2024a; Qian et al., 2024b; Lin et al., 2024). Multi-turn testbeds further probe whether agents can incorporate feedback, use tools, and adapt to shifting goals (Wang et al., 2024b; Yao et al., 2024; Barres et al., 2025; Mehri et al., 2025), while personalized suites examine whether models can infer and sustain user-specific preferences over longer conversations (Zhao et al., 2025a). Complementing evaluation, agent design advances teach models to clarify ambiguity rather than hallucinate intent (Zhang et al., 2024; Chen et al., 2025a) and to personalize outputs through explicit or latent user modeling (Gao et al., 2024; Singh et al., 2024). Meanwhile, RL efforts such as MAU-RL have begun to improve robustness in user-facing behaviors (Zhao et al., 2025b).

Our work is aligned with these directions but targets a different gap: many existing user-facing benchmarks emphasize evaluation, specific domains, or one-off setups, whereas **UserRL** provides a *unified suite* of gym environments that are *designed for both benchmarking and multi-turn RL training* under a standardized interaction interface. This enables controlled comparisons of training and reward design choices across diverse user-centric skills within one reproducible platform.

Agentic RL training and adaptation. Supervised fine-tuning with curated data has been widely used to improve agentic abilities such as tool use (Schick et al., 2023; Qin et al., 2024). Reinforcement learning has recently emerged as a scalable alternative that can promote stronger reasoning and generalization. Early methods such as PPO (Schulman et al., 2017) and RLHF (Ouyang et al., 2022) have evolved into preference-based and trajectory-level algorithms, including DPO (Rafailov et al., 2023), SimPO (Meng et al., 2024), and trajectory-level approaches such as GRPO (Guo et al., 2025) and variants including DAPO (Yu et al., 2025) and VAPO (Yuan et al., 2025). In parallel, agentic RL has extended from single-turn alignment to multi-turn interaction and long-horizon reasoning, including trajectory-level preference optimization (Shani et al., 2024), direct trajectory optimization (Wang et al., 2025), and tool learning without step-level supervision (Singh et al., 2025), with applications to search (Jin et al., 2025), tool use (Qian et al., 2025a), and value judgment (Chen et al., 2025b). RL also intersects with test-time optimization (Zuo et al., 2025) and multimodal learning (Shen et al., 2025).

Relative to these literatures, **UserRL** does not present as a new RL algorithm: rather, it is a *user-centric multi-turn training and evaluation testbed* that makes RL design choices comparable across tasks. In particular, our framework exposes modular hooks for both turn-level reward assignment and trajectory-level scoring within the same optimization pipeline (e.g., GRPO), enabling systematic study of credit assignment and user simulation effects in interactive settings.

B Novelty and Contributions.

UserRL builds on established RL and simulation components, but its novelty lies in *how these elements are unified into a reproducible, extensible testbed* for user-centric, multi-turn RL. Rather than proposing a single fixed benchmark or a universally optimal reward formulation, UserRL provides infrastructure that supports controlled comparisons of design choices across diverse interactive settings. Concretely, we contribute:

- **A unified suite of user-centric gym environments.** We introduce eight gyms that cover a broad range of user-centric abilities (e.g., clarification, persuasion, personalized planning, and tool-mediated interaction), and are designed to support both benchmarking and RL training.
- **A standardized interaction interface for plug-and-play extension.** All gyms share a minimal tool interface, enabling researchers to add new environments, swap user simulators, or integrate new verifiers without modifying the core training loop.
- **Modular reward specification with decoupled credit structure.** UserRL separates turn-level reward assignment from trajectory-level scoring within the same pipeline, making it straightforward to explore alternative credit assignment and aggregation strategies across gyms.
- **Evaluation and analysis geared toward interaction quality.** Beyond task success, we include

analyses that reflect interaction efficiency and multi-turn behavior, facilitating more diagnostic study of user-centric agents.

Overall, UserRL is best viewed as a *research platform* for user-centric agent RL: it unifies training and evaluation under standardized interfaces, while remaining flexible enough for community-driven extensions in environments, simulators, and reward definitions.

C Gym Construction Details

IntentionGym. IntentionGym is an environment designed to evaluate an agent's ability to uncover a user's true intent when given vague or underspecified tasks. The core task logic requires the agent to iteratively ask targeted clarifying questions until all critical missing details are revealed. Rewards are assigned based on the importance of the uncovered detail (high, medium, low), with penalties for unfocused or overly broad questions, encouraging efficient and precise inquiry. The dataset includes diverse user tasks across domains such as travel, education, and technology, each annotated with missing details and importance levels. The environment thus provides a principled testbed for intent understanding and ambiguity resolution through multi-round user–agent interaction.

The IntentionGym environment processes each action through a sophisticated two-step evaluation system. When an agent submits a question as input (corresponding to `Action` operation choice), the environment performs two parallel language model calls: (1) a response generation call that creates a natural conversational response from the simulated user's perspective without knowing the ground truth missing details (using temperature 0.7 for naturalness), and (2) an evaluation call that analyzes which specific missing details the question addresses (using temperature 0.0 for consistency). The evaluation model receives the complete list of remaining missing details with their importance levels (1=Low, 2=Medium, 3=High) and returns indices of covered details plus analysis metadata. The reward calculation uses a tiered system where High importance details yield 1.0 base reward, Medium yield 0.7, and Low yield 0.4, with a multi-detail penalty of 0.2 per additional detail covered (encouraging focused questions), multiplied by the configurable reward scale (default 1.0), then reduced by the step penalty (default 0.0), and finally normalized to [0,1] if configured. The environment tracks conversation history, updates covered/remaining detail lists, and terminates when all missing details are covered or maximum steps (default 20) is reached. The simulated user role is to respond authentically as someone seeking help while maintaining ignorance of what specific details need clarification, creating a realistic intention-guessing scenario where the agent must strategically ask questions to uncover the hidden missing information.

Please refer to Figure 4 and Figure 5 for system instruction details about the response generation and coverage evaluation processes.

PersuadeGym. PersuadeGym is an environment designed to evaluate an agent's ability to persuade a user to change its stance on controversial statements through strategic argumentation. The core task logic requires the agent to iteratively present compelling arguments to move the user from an initial "Strongly Agree" position toward "Strongly Disagree" on various claims. Rewards are assigned based on the magnitude of stance change achieved, with exponential scaling that encourages significant persuasion rather than incremental shifts. The dataset includes diverse controversial statements across domains such as technology policy, social issues, and governance, each with detailed initial arguments supporting the the user's starting position. The environment thus provides a principled testbed for persuasive communication and strategic reasoning through multi-round adversarial dialogue.

The PersuadeGym environment processes each action through a stance evaluation system. When an agent submits a persuasive argument as input (corresponding to `Action` operation choice), the environment performs a single language model call that both generates a natural response and evaluates stance change (using temperature 0.0 for consistency). The evaluation model that simulates the user receives the current statement, the initial argument, conversation history, and the agent's latest argument, then returns a JSON response containing the feedback, updated stance, and internal reasoning. The stance progression follows seven discrete levels: Strongly Agree (0.0), Agree (0.167), Partly Agree (0.333), Neutral (0.5), Partly Disagree (0.667), Disagree (0.833), and Strongly Disagree (1.0). The reward calculation uses exponential scaling where the reward equals the number of levels changed toward disagreement divided by 6 (the maximum possible change), with no reward for moving back toward agreement or no change. The

IntentionGym LLM Instruction for Response Generation

```
You are a person who has posted a vague request for help and is now responding to someone who is
trying to help clarify your needs.

Your job is to respond naturally as the person who originally made the request.  Follow
these guidelines:

1. If the question is asking about your specific preferences for this task:
- Provide an authentic and coherent response
- Share realistic preferences that someone might have for this type of task
- Be conversational and natural

2. If the question is NOT directly about your preferences for this task:
- Try to answer helpfully if you can
- Guide the conversation back to clarifying what you need for your task
- Be polite but redirect: "That's interesting, but what I'm really trying to figure out is..."
- Do NOT provide what missing details need to be clarified or give any examples.
- Do NOT provide concrete help or solutions - you're the one seeking help!

Please respond in the following json format: {
    "thought": "Your thought process about whether the question is about your preferences and how
to respond",
    "response": "Your natural conversational response"
}

IMPORTANT:
- Respond only as the person seeking help, not as an evaluator
- Be natural and conversational
- Don't reveal any "ground truth" or act like you know what details are missing
- Just respond authentically as someone who made this request
```

Figure 4: IntentionGym LLM system instruction for response generation.

environment tracks conversation history, updates the simulated user's stance, and terminates when the user reaches "Strongly Disagree" (complete persuasion success) or maximum steps (default 20) is reached. The simulated user's role is to respond authentically as someone holding a strong initial position while being open to compelling evidence, creating a realistic persuasion scenario where the agent must strategically craft arguments to overcome resistance.

Please refer to Figure 6 for system instruction details about the AI's response generation and stance evaluation process.

TurtleGym. TurtleGym is an environment designed to evaluate an agent's ability to engage in creative reasoning and contextual adaptation through interactive story-based puzzle solving with a user. The core task logic requires the agent to play a "turtle soup" game where they must uncover hidden twists in mysterious stories by asking strategic questions and providing comprehensive explanations. The environment presents the agent with a surface-level story description and challenges them to discover the underlying "bottom" truth through iterative inquiry and creative interpretation. Rewards are assigned based on how well the agent's final story explanation covers the evaluation criteria, with weighted scoring that emphasizes the most critical story elements. The dataset includes diverse mysterious scenarios with hidden twists, each annotated with evaluation criteria and importance weights. The environment thus provides a principled testbed for creative reasoning and contextual adaptation through multi-round collaborative story exploration.

The TurtleGym environment processes each action through a dual-mode evaluation system. When an agent submits an inquiry action (corresponding to `Action` operation choice), the environment performs a single language model call that evaluates whether the question is helpful for understanding the story, returning "Yes", "No", or "Maybe" responses (using temperature 0.0 for consistency). When an agent submits an answer action (corresponding to `Answer` operation choice), the environment evaluates the story explanation against multiple weighted criteria, where each criterion receives a score of 0.0 (completely incorrect), 0.5 (partially correct), or 1.0 (completely correct), multiplied by the criterion's weight to produce a final score between 0.0 and 1.0. The reward calculation uses

IntentionGym LLM Instruction for Coverage Evaluation

```
You are evaluating how well a user's question addresses missing details in a vague task.

Your job is to evaluate which (if any) of the missing details are addressed by the latest
question.

Rules for evaluation:
- If the question is NOT related to clarifying the user's intent or task requirements,
covered_detail_indices should be an empty list
- For each missing detail that is directly addressed by the question, note its index
- A question "addresses" a detail if it would help reveal the information needed for that detail

Please respond in the following json format:
{
    "analysis": "Brief explanation of what missing details (if any) were covered by this question",
    "is_task_related": true/false (whether the question is about clarifying the task requirements),
    "covered_detail_indices": [list of indices from the missing details list that this question
addresses]
}

IMPORTANT: You are ONLY evaluating, not generating responses. Focus solely on which details
are addressed.
```

Figure 5: IntentionGym LLM system instruction for coverage evaluation.

PersuadeGym LLM Instruction for Response Generation and Stance Evaluation

```
## Task
You are participating in a persuasion conversation. You hold a position on a statement, and someone
is trying to persuade you to change your mind. For each of their arguments, you should:

1. Respond naturally and thoughtfully to their argument
2. Express your current stance after considering their point
3. You may change your mind if presented with compelling evidence
4. Be respectful but critical - point out logical fallacies if you see them

Follow the format strictly:
“‘json
{
    "thought": "Your internal reasoning about their argument",
    "response": "Your public response to their argument (max 200 tokens)",
    "stance": "One of: Strongly Agree, Agree, Partly Agree, Neutral, Partly Disagree, Disagree,
Strongly Disagree"
}
“‘
```

Figure 6: PersuadeGym LLM system instruction for response generation and stance evaluation.

incremental improvement: only answers that exceed the current best score receive positive rewards equal to the score improvement multiplied by the reward scale (default 1.0), with step penalties (default 0.0) applied cumulatively. The environment tracks the best score achieved and terminates when an answer reaches the success threshold (default 0.9) or maximum steps (default 20) is reached. The simulated user's role is to provide objective evaluation of the agent's questions and story explanations without revealing the ground truth, creating a realistic collaborative puzzle-solving scenario where the agent must strategically explore and creatively interpret mysterious scenarios.

Please refer to Figure 7 and Figure 8 for system instruction details about the inquiry evaluation and story explanation scoring processes.

TelepathyGym. TelepathyGym is an environment designed to evaluate an agent's ability to engage in strategic reasoning and hypothesis testing through interactive mind reading games with a user. The core task logic requires the agent to guess what entity the user is thinking of by asking strategic yes/no questions and making final guesses. The environment presents the agent with a category description and challenges them to systematically narrow down the possibilities through binary questioning, building a hypothesis space that converges on the correct answer. Rewards are assigned based

TurtleGym LLM Instruction for Inquiry Evaluation

```
## Task
You are a helpful assistant to respond to the user query based on the given story scenario (surface)
and ground truth (bottom) in a Turtle Soup game. Please follow the instructions below.

## Instructions
1. You can only give three values: "Yes", "No", or "Maybe" in your response.
2. "Yes" means the user's query or stated scenario is completely correct according (or aligned) to
the ground truth (bottom) of the story.
3. "No" means the user's query is incorrect or contradicts the ground truth (bottom) of the story,
or the user's query is not even close to the ground truth.
4. "Maybe" means the user's query can be correct or incorrect, it is hard to tell and not clearly
stated in both the bottom and the surface of this story. "Maybe" is usually used when the user's
query is not quite relevant to the ground truth. Please try to be determinant and use as less
"Maybe" in your response as possible.

## Example Format

### Your Response
“‘json
{
    "thought": "Your thought about how to evaluate the user's query, and justify the response you
give.",
    "response": "Yes" or "No" or "Maybe"
}
“‘
```

Figure 7: TurtleGym LLM system instruction for inquiry evaluation.

on the accuracy of the final guess, with binary scoring that emphasizes successful entity identification. The dataset includes diverse entities across categories such as famous people, animals, objects, and landmarks, each with detailed descriptions for the simulated user's knowledge base. The environment thus provides a principled testbed for strategic reasoning and hypothesis testing through multi-round interactive deduction.

The TelepathyGym environment processes each action through a dual-mode evaluation system. When an agent submits an inquiry action (corresponding to `Action` operation choice), the environment performs a single language model call that evaluates the question against the target entity and returns "Yes", "No", or "Maybe" responses (using temperature 0.0 for consistency). When an agent submits an answer action (corresponding to `Answer` operation choice), the environment evaluates whether the final guess correctly identifies the target entity, returning a binary score of 1.0 for exact matches or 0.0 for incorrect guesses. The final reward is calculated by multiplying the reward scale (default 1.0), with step penalties (default 0.0) applied cumulatively. The environment tracks clue history from previous questions and responses, updates the best score achieved, and terminates when a correct guess is made (score 1.0) or maximum steps (default 20) is reached. The simulated user's role is to respond honestly to yes/no questions based on the target entity while maintaining the mystery, creating a realistic mind reading scenario where the agent must strategically formulate questions to systematically eliminate possibilities and converge on the correct answer.

Please refer to Figure 9 and Figure 10 for system instruction details about the question response and entity guessing evaluation processes.

FunctionGym. FunctionGym is an environment designed to evaluate an agent's ability to engage in mathematical reasoning and pattern generalization through interactive function discovery with a user. The core task logic requires the agent to uncover a hidden mathematical mapping rule by testing different number combinations and analyzing the results to identify the underlying pattern. The environment presents the agent with a four input numbers and challenges them to discover the mapping function through systematic experimentation and logical deduction. Rewards are assigned based on the accuracy of the final answer for a test case, with binary scoring that emphasizes successful mathematical reasoning. The dataset includes diverse mathematical functions with varying complexity levels, each containing a hidden mapping rule, test case, and expected result. The environment thus provides a principled testbed for mathematical reasoning and

TurtleGym LLM Instruction for Inquiry Evaluation

```
## Task
You are a helpful agent to help me evaluate the correctness of the user's story against the ground
truth in a Turtle Soup game. You should give both your score and evaluation feedback based on a
evaluation protocol provided. Please follow the instructions below.

## Instructions
1. There may exist multiple evaluation criteria based on the evaluation protocol. You should give a
score for each criteria.
2. Your score can only take three values: 0, 0.5, 1.0, where 0 means the user's answer is completely
incorrect (not even close to the ground truth), 0.5 means the user's answer partially aligns with
the ground truth, and 1.0 means the user's answer is completely correct.
3. After giving the score, you should give an overall feedback about which part in the user's
answer is correct (or the story is all wrong and totally not aligned). Do not say which
part is incorrect or not aligned with the ground truth. Do not release anything else about the
ground truth (bottom) or the evaluation protocol. Try to keep your feedback concise and to the point.

## Example Format

### Your Response
“‘json
{
    "scores":[
        {
            "statement": "Copy the exact statement from the evaluation protocol.",
            "thought": "Your thought about how to evaluate the statement, and justify the score
you will give based on the protocol statement and comparison between the ground truth and user's
answer.",
            "score": 0 or 0.5 or 1.0
        },
    ... (the number of scores should be the same as the number of criteria in the evaluation protocol,
and the order should also exactly match)
    ],
    "feedback": "Your feedback to the user's answer about which part is correct. Do not release
anything about the ground truth (bottom) and the evaluation protocol. Be concise and to the point.
Use the second person tone (you / your) to address the user."
}
“‘
```

Figure 8: TurtleGym LLM system instruction for story explanation scoring.

pattern generalization through multi-step interactive problem solving.

The FunctionGym environment processes each action through a three-mode evaluation system. When an agent submits a four input numbers (corresponding to `Action` operation choice), the user evaluates them by applying the hidden function rule and returns the calculated result, with no reward for calculation actions. When an agent submits a search action (corresponding to `Search` operation choice), the user retrieves the test case numbers, providing the agent with the specific input values for the final evaluation. When an agent submits an answer action (corresponding to `Answer` operation choice), the environment evaluates whether the numerical answer matches the expected result within a tolerance of 1e-6, returning a reward of 1.0 for correct answers or 0.0 for incorrect ones. The final reward is calculated by applying the correct answer reward (default 1.0) or incorrect answer reward (default 0.0), with step penalties (default 0.0) applied cumulatively. The environment tracks action history, updates the answer status, and terminates when a correct answer is submitted or maximum steps (default 20) is reached. The simulated user's role is to provide accurate mathematical calculations based on the hidden rule while maintaining the mystery of the underlying mapping function, creating a math reasoning scenario where the agent must systematically test hypotheses and generalize from observed patterns to discover the correct mapping rule.

All the user response in this gym is rule-based so there is no LLM simulation for user specifically in this gym.

SearchGym. SearchGym is an environment designed to evaluate an agent's ability to demonstrate general helpfulness and reasoning through interactive web search and question answering with a user. The core task logic requires the agent to an-

TelepathyGym LLM Instruction for Question Response

```
## Task
You are a telepathic entity playing a mind reading game. The user is trying to guess what entity
you are thinking of by asking yes/no questions. You should respond honestly based on the target
entity you're thinking of.

## Instructions
1. You are thinking of a specific entity (person, object, concept, etc.) - this is the "target_entity"
provided to you.
2. The user will ask questions to narrow down what you're thinking of.
3. Answer "Yes" if the question is true about your target entity.
4. Answer "No" if the question is false about your target entity.
5. Answer "Maybe" only if the question is ambiguous or you genuinely cannot determine a clear yes/no
answer.
6. Be helpful and honest - the goal is for them to eventually guess correctly through good questions.

## Example Format

### Your Response
“‘json
{
    "thought": "Your reasoning about how the user's question relates to the target entity.",
    "response": "Yes" or "No" or "Maybe"
}
“‘
```

Figure 9: TelepathyGym LLM system instruction for question response.

swer general knowledge questions from the user by performing web searches to retrieve relevant information and synthesizing the results into accurate responses. The environment presents the agent with diverse questions across various domains and challenges them to leverage online information sources effectively to provide correct answers. Rewards are assigned based on the accuracy of the final answer, with binary scoring that emphasizes successful information retrieval and synthesis. The dataset includes diverse general knowledge questions from the Bamboogle benchmark, each requiring multi-step reasoning and information gathering. The environment thus provides a principled testbed for general helpfulness and reasoning through multi-step search-based question answering.

The SearchGym environment processes each action through a dual-mode evaluation system. When an agent submits a search action (corresponding to `Search` operation choice), the environment performs a web search using the Serper API and returns formatted search results with titles and snippets, with no reward for search actions but a limit on maximum search steps (default 5). When an agent submits an answer action (corresponding to `Answer` operation choice), the environment evaluates the answer using either rule-based comparison (normalized string matching) or LLM-based evaluation (using temperature 0.0 for consistency, default method), returning a reward of 1.0 for correct answers or 0.0 for incorrect ones. The final reward is calculated by applying the correct answer reward (default 1.0) or incorrect answer reward (default 0.0), with step penalties (default 0.0) applied cumulatively. The environment tracks search history, updates the answer status, and terminates when a correct answer is submitted or maximum steps (default 20) is reached. The simulated user's role is to provide accurate web search results by interacting with the real web backend and evaluate answer correctness while maintaining the challenge of information synthesis, creating a search-based question answering scenario where the agent must strategically formulate search queries and synthesize information from multiple sources to provide accurate responses.

Please refer to Figure 11 for system instruction details about answer evaluation processes.

TauGym. TauGym is an environment designed to evaluate an agent's ability to demonstrate tool use and task-oriented interaction through comprehensive user requirement fulfillment with a simulated user. The core task logic requires the agent to accomplish complex user requests by leveraging available internal tools, gathering necessary information through conversation, and executing appropriate tool calls to complete the specified objectives. The environment presents the agent with realistic user scenarios from retail and airline domains and

TelepathyGym LLM Instruction for Entity Guessing Evaluation

```
## Task
You are a telepathic entity playing a mind reading game. The user is trying to guess what entity
you are thinking of based on the clues you've given through your "Yes" or "No" responses to their
questions. You need to evaluate if their final guess is correct.

## Instructions
1. You are thinking of a specific entity (person, object, concept, etc.) - this is the "target_entity"
provided to you.
2. The user has been asking questions about this entity and is now making a final guess.
3. You should evaluate if their guess correctly identifies the target entity you were thinking of.
4. Only return "Yes" if their guess is exactly correct or a clearly equivalent/synonymous
identification of the target entity.
5. Return "No" if their guess is wrong, partially correct, or close but not exact.
6. There is NO partial credit - it's either completely right (Yes) or wrong (No).
7. Address the user in second person tone (e.g., "You", "Your", "You're") in your feedback.
8. Your feedback should be concise and do not release anything about the target entity. Just state
your judgment and encourage or congratulate the user.

## Example Format

### Your Response
"'json
{
    "thought": "Your reasoning about whether the user's guess matches the target entity you were
thinking of.",
    "judgment": "Yes" or "No",
    "feedback": "Brief feedback explaining why their guess is correct or incorrect. Do not reveal
the correct answer if they are wrong."
}
"'
```

Figure 10: TelepathyGym LLM system instruction for entity guessing evaluation.

challenges them to navigate multi-step workflows involving tool discovery, user communication, and systematic task execution. Rewards are assigned based on the original Tau-Bench implementation. The dataset includes diverse task-oriented scenarios from the existing Tau-Bench benchmark, each containing detailed user requirements and available tool sets. The environment thus provides a principled testbed for tool use and task-oriented interaction through multi-step problem solving.

The TauGym environment processes each action through a three-mode evaluation system. When an agent submits an interaction action (corresponding to `Action` operation choice), the environment facilitates direct communication with the simulated user and returns conversational responses. When an agent submits a search action (corresponding to `Search` operation choice), the environment retrieves available internal tools information or help documentation, providing the agent with comprehensive tool specifications and usage guidelines. When an agent submits an answer action (corresponding to `Answer` operation choice), the environment parses and executes tool calls using the existing Tau-bench framework, returning tool execution results and task completion status. The final reward is calculated by the tau-bench evaluation system for each step based on task completion quality and user satisfaction, with additional step penalties (default 0.0) applied cumulatively. The environment tracks action history, maintains tool information, and terminates when the task is successfully completed or maximum steps (default 30) is reached. The simulated user's role is to provide realistic task requirements and respond authentically to agent interactions while maintaining the complexity of task-oriented scenarios, creating a comprehensive evaluation environment where the agent must strategically combine conversation, tool discovery, and systematic execution to fulfill user requirements effectively.

All the user responses are based on the implementation of original Tau-Bench so there is no additional LLM prompt or instruction introduced in this specifically adapted gym environment.

TravelGym. TravelGym is an environment designed to evaluate an agent's ability to demonstrate preference elicitation and personalized planning through comprehensive travel booking assistance with a user. The core task logic requires the agent to

SearchGym LLM Instruction for Answer Evaluation

```
## Task
You are asked to judge whether the answer for a question is correct or not.

## Instructions
1. You will be provided with the question, the model's answer, and the correct answer.
2. If the answer is exactly the same, or a clearly equivalent/synonymous identification of the
correct answer, return "Yes". Please base your answer judgment on the given question scenario,
instead of just comparing the answers.
3. If the answer is wrong, return "No".
4. In your feedback, you could provide a succinct explanation for your judgment, but you should
never reveal the correct answer.
5. In your feedback, please use second person tone (e.g., "You", "Your", "You're").
6. In your feedback please do not give any hint or any information about the correct answer.

## Example Format

### Your Response
“‘json
{
    "reasoning": "Your reasoning about whether the answer is correct or incorrect.",
    "judgment": "Yes" or "No",
    "feedback": "Brief feedback explaining why the answer is correct or incorrect. Do not reveal
the correct answer if they are wrong."
}
“‘
```

Figure 11: SearchGym LLM system instruction for answer evaluation.

help users make personalized travel arrangements by eliciting their preferences across multiple dimensions (flight, hotel, rental car, apartment, restaurant) and providing tailored recommendations. The environment presents the agent with diverse travel scenarios and challenges them to systematically gather user preferences, search for relevant options, and make optimal choices that align with the user's needs. The dataset includes complex travel planning scenarios with multiple preference dimensions, each containing correct, wrong, and noise options to test the agent's ability to distinguish quality recommendations. The environment thus provides a principled testbed for preference elicitation and personalized recommendation through multi-round user-agent interaction.

The TravelGym environment processes each action through a three-mode evaluation system. When an agent submits an action (corresponding to `Action` operation choice), the environment performs a single LM call that evaluates whether the agent's utterance relates to preference elicitation, returning type classifications (1: normal conversation, 2: preference-related, 3: unavailable preference, 4: too vague) with corresponding rewards of 0.0, 0.2, 0.0, and 0.0 respectively (using temperature 0.0 for consistency). When an agent submits a search action (corresponding to `Search` operation choice), the environment evaluates search request alignment and returns travel options for valid dimensions, with successful searches yielding 0.2 reward and system errors simulated every 5 search attempts. When an agent submits an answer action (corresponding to `Answer` operation choice), the environment evaluates option selections against ground truth, with best options yielding 1.0 reward, correct but not the best options yielding 0.8 reward, and wrong choices incurring penalties (default 0.0). The final reward is calculated by multiplying the reward scale (default 1.0), with step penalties (default 0.0) applied cumulatively. The environment tracks conversation history, preference elicitation progress, and remaining options, terminating when all best options are selected or maximum steps (default 20) is reached. The simulated user's role is to respond authentically to preference questions while maintaining realistic travel planning constraints, creating a comprehensive scenario where the agent must strategically balance information gathering, search execution, and recommendation quality.

All the user responses are based on the implementation of UserBench, so there is no additional LLM prompt or instruction introduced in this specifically adapted gym environment.

D Clarifications and Justifications

Fidelity of User Simulator. Designing user simulators requires balancing flexibility and task rigor. To maintain naturalness while ensuring reproducibility, our LLM-based users are not entirely free-form but are guided by task instructions and ground-truth annotations. Conceptually, the user can be viewed as an "oracle with constraints": it classifies the agent's intent, evaluates responses against the ground truth, and produces replies according to carefully crafted instruction templates. This design ensures that feedback remains consistent, interpretable, and task-specific rather than arbitrary. To prevent trivial solution strategies (e.g., direct leakage of answers), certain gyms deliberately restrict the user's accessible information. We manually audited interactions under GPT-4o and Qwen3-32B that we employed in main experiments, including adversarial prompts, and found no evidence of answer leakage or instability. Empirically, the simulators consistently generated faithful and robust responses, which supports their reliability for rigorous training and evaluation. In summary, the deliberate instruction design of each gym ensures robustness without sacrificing reproducibility, a critical property for benchmarking user-centric agents.

Parameter Selection and Sensitivity. Our parameter settings are motivated by empirical tuning rather than arbitrary choice. For reward-to-go (R2G), we selected $\gamma = 0.8$ based on a grid search over $\{0.6, 0.7, 0.8, 0.9\}$ using Qwen3-4B. Values ≤ 0.5 caused premature decay of future rewards and unstable training, while $\gamma = 1$ reduced to raw summation, leading to overemphasis on the first turn. $\gamma = 0.8$ yielded the best overall micro-average across gyms, balancing stability and forward credit propagation. For the Exponential Mapping (EM), we experimented with $k \in \{0.5, 1, 2, 3\}$. $k = 0.5$ consistently underperformed, whereas $k = 1, 2, 3$ produced nearly identical results (variance $< 1\%$), indicating robustness of the mapping curve within this range. We chose $k = 2$ as a representative setting that offers a sharper but stable reward gradient. These choices were thus informed by both principled considerations and empirical validation.

Trajectory and Turn-Level Reward Shaping. We selected our shaping strategies to cover both theoretically grounded and practically intuitive variants. R2G follows classic formulations of temporal credit assignment (Tamar et al., 2016), rewarding actions in proportion to their downstream impact and thereby encouraging efficiency in multi-turn interactions. EM, on the other hand, applies a monotonic convex transformation that emphasizes decisive turns by amplifying informative signals while compressing noise, reflecting the practical dynamics of dialogue where early clarifications are disproportionately valuable. We also included "baseline" strategies such as naive, equalized, and summation, which serve as natural points of comparison and reveal how much gain arises from more structured shaping. Together, these strategies form a principled yet diverse spectrum for probing credit assignment in user-centric environments. While our work highlights the effectiveness of R2G and EM, our open-source framework makes it straightforward to extend this analysis with additional shaping functions, ensuring that our method can evolve alongside future research.

On External Benchmark Transfer. We have not include additional external benchmarks beyond those already integrated into our gym suite for several reasons. First, several widely used user-centric benchmarks such as UserBench, IN3, and Tau-Bench are already instantiated within our framework as gyms, ensuring direct comparability without redundant evaluation. Second, our emphasis is on interactive agents operating through standardized tool-use abstractions, which many existing benchmarks do not natively support; adapting them faithfully would require significant re-engineering and could compromise comparability. Third, our main experiments already span a broad spectrum of capabilities, including reasoning, planning, retrieval, and dialogue grounding, with three gyms explicitly held out as unseen test environments in the main results table. This design allows us to study transfer and generalization without requiring every possible benchmark. Moreover, evaluating on heterogeneous external datasets often introduces confounding differences in annotation quality, interface format, or reward granularity, which may obscure rather than clarify the effects of reward shaping and trajectory scoring. Finally, we view further adaptation of external resources into our gym environment as a natural avenue for community-driven extension, which our open-source release is designed to facilitate. In this way, we balance rigor, breadth, and clarity of evaluation, while keeping

RL Training Config		
Parameter	Value	Notes
algorithm.gamma	0.8	Discount factor
algorithm.k	2.0	Scaling coefficient
data.train_batch_size	128	Training batch size
data.max_prompt_length	1152	Max input length
data.max_response_length	8192	Max output length
actor.optim.lr	1e-6	Learning rate
actor.ppo_mini_batch_size	16	PPO minibatch size
actor.use_kl_loss	False	KL loss disabled
actor.entropy_coeff	0	Entropy coefficient
rollout.name	sglang	Rollout engine
rollout.n	8	Parallel rollouts
rollout.multi_turn.max_turns	16	Multi-turn limit
trainer.n_gpus_per_node	8	GPUs per node
trainer.nnodes	1	Number of nodes
trainer.total_epochs	15	Training epochs
SFT Cold Start Config		
Parameter	Value	Notes
finetuning_type	full	Full parameter tuning
cutoff_len	16384	Max input length
per_device_train_batch_size	2	Batch size per GPU
gradient_accumulation_steps	4	Grad accumulation
learning_rate	1.0e-5	Base LR
num_train_epochs	3.0	Training epochs
lr_scheduler_type	cosine	Scheduler type
warmup_ratio	0.1	Warmup fraction
bf16	true	Mixed precision

Table 6: RL and SFT Training Configurations.

the benchmark extensible for future work.

E Training Experiment Details

Training Data. As described in Section 3, we use 1k samples for SFT cold start before RL training. To construct this dataset, we collect all training tasks from five environments, where GPT-4o acts as both the simulated user and the agent model interacting with each gym. We then rank all interaction trajectories within each environment. For each environment, we select the top-K ranked trajectories and combine them to form the final 1k-sample dataset used for SFT training. Beyond these 1k samples, all remaining training tasks are used for RL training. Since RL requires the actor model to roll out multiple trajectories, only task descriptions are needed rather than distilled trajectories.

Training Configuration. The detailed training configurations are listed in Table 6. For SFT, we use 4 Nvidia H200 GPUs, and each training takes approximately 1 hour. For RL training, we use 8 Nvidia H200 GPUs, and each training takes approximately 1.5 days. We use LlamaFactory (Zheng et al., 2024) for SFT training and VERL (Sheng et al., 2024) for RL training. Especially for RL, we further reserve 5% of data as validation set and pick the best saved checkpoint for final result evaluation for each setting.

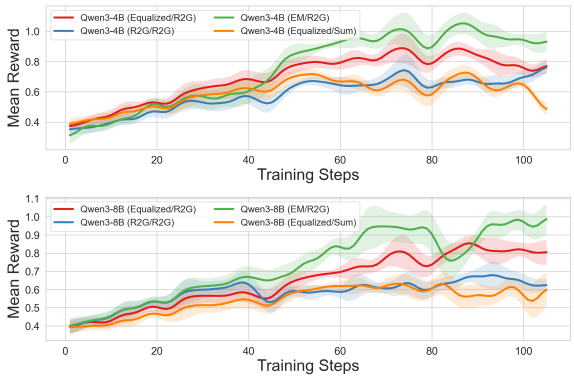

Figure 12: Training curves of Qwen3 4B (**top**) and 8B (**bottom**) models across different reward strategies.

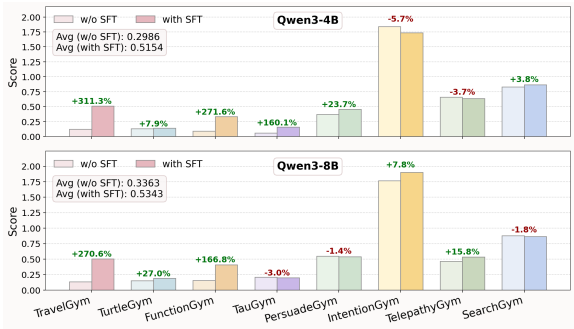

Figure 13: Additional experiment to validate SFT cold start improves RL training performance compared to direct RL on raw Qwen3 models.

Evaluation Configuration. For evaluation, we keep all the model's temperature to 0.0 to ensure consistency. The maximum interaction turn is set to 16, while all other settings are set to default aligning with the description in Section C.

Instructions and Tools. We present the system prompt template for the agent in Figure 16. The content inside the placeholders varies depending on the gym that the agent interacts with; in the figure, we show an example from TauGym's system prompt. All tasks within the same gym share the same system prompt. Importantly, both training and evaluation for each gym use the same system prompt to ensure consistency. In addition, we provide the YAML schema of the `interact` tool, which defines the interface for agent-gym interaction, in Figure 15.

F Analysis Details

F.1 Details of Training Curves.

For all curves in Figure 3, we report rewards using a consistent standard: the sum of turn-level rewards

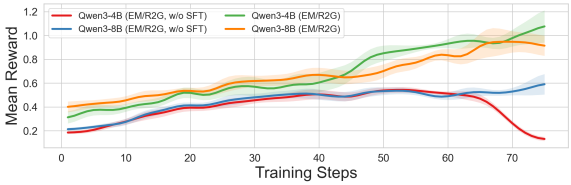

Figure 14: Additional experiment on comparisons of the training curves under the EM/R2G setting: w/ vs. w/o SFT cold start.

per trajectory. This ensures comparability across methods, even though the actual training signals may differ (e.g., R2G computes trajectory scores differently, and GRPO further transforms them into token-level advantages). Thus, the plotted rewards should be interpreted as a normalized measure for comparison rather than the raw training signal.

Training curves (solid lines) are smoothed with a Gaussian 1D filter ($\sigma = 2$). The shaded regions indicate fluctuations during training, defined as the deviation between the raw and smoothed values at each step. Their boundaries are further smoothed with the same filter for clarity.

F.2 Justification for User Simulation.

We employ Qwen3-32B for user simulation during training due to the large number of requests involved. Specifically, the total number of requests in one training run is approximately: data size (2k) $\times$ epochs (15) $\times$ rollouts (8) $\times$ maximum turns per rollout (16) $\approx$ 4M requests. Using a closed-source model for such scale would be prohibitively expensive, making an open-source model a more budget-friendly choice for training.

During inference, however, the total number of requests is much smaller. Therefore, we use GPT-4o for user simulation at test time, as it provides stronger instruction-following and more realistic user behavior. This setup also allows us to verify whether a model trained with weaker user simulations can generalize effectively to interactions with a stronger simulated user during final evaluation.

F.3 Real user participation.

For real-user testing, we recruited 20 participants with diverse backgrounds, including undergraduate and PhD students from Computer Science, Physics, and Economics. Participation was voluntary with no payment, and all participants consented to the use of their interaction data for research purposes. Before the experiment, users were instructed on the purpose of the gyms, the evaluation procedure, and how to interact with the tested models. In all gyms, their role was to act as the user/oracle: they responded to the model's questions and judged answers against the ground truth, while avoiding direct disclosure of the correct answers.

Participants were assigned tasks across both gyms, and results were aggregated across users. We emphasize that user behavior naturally varies across individuals, leading to score variance; nevertheless, the overall trend is consistent across models (Table 5). As observed in Section 4.3, models tend to perform better with real users than with the simulated user. A plausible explanation from log inspection is that, although users were instructed not to reveal answers, they often provided subtle, non-revealing cues rather than only short judgments, reflecting a more cooperative interaction style.

The study was determined to be exempt from full external ethics review because it involved only non-sensitive interaction tasks with consenting adult participants. We include this clarification to distinguish the real-user evaluation from a large-scale behavioral or clinical human-subjects study. The purpose of the study is to provide preliminary evidence that UserRL-trained behaviors remain visible under human interaction, not to claim population-level robustness across all user groups.

F.4 Follow-up experiment results.

For the main results, we provide the training curve regarding different turn-level rewarding strategies in Figure 12. We also present supplementary analysis of training raw Qwen3 4B and 8B models under the *EM/R2G* setting. The corresponding results and training curves are shown in Figure 13 and Figure 14. These findings further validate that models initialized with SFT cold start consistently outperform those trained without it, even under different reward shaping strategies. This strengthens our main claim that SFT initialization is critical for unlocking and sustaining effective user-centric RL training.

G Additional Experiments

G.1 Robustness to train/test gym composition.

Our main results use a fixed train/test split across gyms. To test sensitivity to the gym composition, we retrain Qwen3-4B under the same setting (Equalized/R2G) but with a different training set: {TravelGym, IntentionGym, TelepathyGym, FunctionGym, TauGym}. We then evaluate *zero-shot* on

YAML Tool Schema: interact_with_env

```yaml
type: "function"
function:
    name: "interact_with_env"
    description: "A tool for interact with a target environment. The detailed environment description and action space is provided in the system prompt, so please follow the system prompt when calling this tool. You can use this tool to interact with the target environment step by step."
    parameters:
        type: "object"
        properties:
            choice:
                type: "string"
                enum: ["action", "answer", "search"]
                description: "Your choice of what to do next, must be one of action, answer or search. Please follow system prompt about the scope of choices you can make and how to decide your choice."
            content:
                type: "string"
                description: "The content of your choice, must be a string. If you choose action, you should provide the action you want to take. If you choose answer, you should provide the answer that you want to submit. If you choose search, you should provide the search query. The specific format of the content is determined by the environment description in the system prompt. Please follow the format strictly in order to successfully use this tool."
            required: ["choice", "content"]
```

Figure 15: Interact tool's schema for environment interaction used in all the training experiments. This tools serves as the standardized interface for agent's interaction with multiple gym environments.

the held-out gyms {SearchGym, TurtleGym, PersuadeGym}, while reporting all gym metrics for completeness.

As shown in Table 7, performance remains comparable to the original split across both trained and held-out gyms. This suggests that our conclusions are not driven by a particular gym partition and that the training scaffold captures broadly useful interaction behaviors across environments.

G.2 Clarifying SFT vs. RL capabilities.

To disentangle what is learned from SFT versus RL, we compare three Qwen3-4B variants: (i) raw (no post-training), (ii) SFT-only (trained on the same distilled trajectories used for cold start), and (iii) SFT+RL (Equalized/R2G). We evaluate on TravelGym and TurtleGym (seen during RL training) and IntentionGym (held-out), keeping the SFT dataset and hyperparameters identical across conditions.

Table 8 shows that SFT alone yields large gains over the raw model, consistent with SFT providing a stable interaction foundation. However, SFT+RL consistently improves further, suggesting RL contributes additional adaptation beyond memorizing interaction templates. The held-out IntentionGym gap is also larger, consistent with RL improving generalization in intention-evolving settings.

G.3 Impact on general capabilities.

To assess potential trade-offs on non-interactive skills, we evaluate raw and UserRL-trained models on MATH500 and LiveCodeBench. We compare Qwen3-4B and Qwen3-8B, with the trained models corresponding to Equalized/R2G.

Table 9 shows only minor degradation relative to raw baselines; UserRL-trained models retain over 95% of baseline performance on these benchmarks. This suggests that the primary effect of UserRL training is to better integrate general reasoning/coding abilities into user-driven, multi-turn contexts rather than replacing them.

G.4 Protocol Compliance and Interface Artifacts

One concern is that improvements under UserRL may come primarily from learning the Action/Search/Answer interface rather than from better user-centric interaction. To diagnose this, we measure protocol-compliance rates across four representative gyms: TravelGym, TurtleGym, FunctionGym, and TauGym. A response is counted as compliant if it follows the required action schema and can be parsed by the environment.

As shown in Table 10, trained models reach perfect compliance, but raw and closed-source models also achieve high compliance rates, typically above 95%. Moreover, compliance does not track final

Model	Travel	Turtle	Func.	Tau	Persuade	Intent	Telepathy	Search
Qwen3-4B (original)	0.5086	0.1844	0.3333	0.2000	0.4643	1.8075	0.6098	0.8640
Qwen3-4B (new train)	0.5054	0.1587	0.3333	0.2121	0.4563	1.8175	0.6341	0.8640

Table 7: Sensitivity to train/test gym composition for Qwen3-4B (Equalized/R2G). "New train" uses {Travel, Intent, Telepathy, Function, Tau} for training and evaluates zero-shot on held-out gyms {Search, Turtle, Persuade}.

Model	TravelGym	TurtleGym	IntentionGym
Qwen3-4B (Eq./R2G)	0.5086	0.1844	1.8075
Qwen3-4B (SFT only)	0.3921	0.1633	1.7375
Qwen3-4B (Raw)	0.1405	0.0854	1.7400

Table 8: Ablation on Qwen3-4B: raw vs. SFT-only vs. SFT+RL (Equalized/R2G). TravelGym and TurtleGym are seen during RL training; IntentionGym is held-out.

Model	MATH500	LiveCodeBench
Qwen3-4B (Equalized/R2G)	95.2	52.1
Qwen3-4B (Raw)	97.0	54.2
Qwen3-8B (Equalized/R2G)	96.0	54.5
Qwen3-8B (Raw)	97.4	57.5

Table 9: General capability check on MATH500 and LiveCodeBench: raw vs. UserRL-trained (Equalized/R2G) models.

Model	TravelGym	TurtleGym	FunctionGym	TauGym
Qwen3-4B (Eq./R2G)	100.00	100.00	100.00	100.00
Qwen3-4B (Raw)	95.75	96.25	95.19	96.67
Gemini-2.5-Pro	98.09	98.96	99.29	98.09
GPT-4o	98.73	97.60	99.04	98.64

Table 10: Protocol-compliance rates (%) across four gyms. Format compliance is high for both trained and untrained/closed-source models, suggesting that task-performance differences are not explained solely by whether the model can emit parseable Action/Search/Answer outputs.

Model	Grounded	Partially Guessed	Fully Guessed
Qwen3-4B (Equalized/R2G)	83%	12%	5%
Qwen3-8B (Equalized/R2G)	86%	12%	2%

Table 11: Manual audit of 80 sampled trajectories for simulator-specific pattern exploitation. Most successful trajectories are grounded in information elicited during interaction, while fully unsupported guessing is rare.

task success monotonically. We therefore view protocol learning as a necessary condition for UserRL performance, but not a sufficient explanation for the observed gains.

G.5 Manual Audit for Simulator-Specific Pattern Exploitation

Because both the agent and the simulated user are LLM-based, a potential failure mode is that the agent may exploit simulator-specific response patterns instead of learning generally useful interaction strategies. We conduct a manual diagnostic audit of 80 randomly sampled test trajectories, consisting of 10 cases from each gym for Qwen3-4B and Qwen3-8B under the Equalized/R2G setting. Each trajectory is categorized as: *grounded*, if the agent's answer is supported by information elicited during the interaction; *partially guessed*, if the agent uses some grounded information but also makes unsupported assumptions; and *fully guessed*, if the answer is mostly unsupported by interaction.

The audit in Table 11 does not rule out simulator bias, but it provides a qualitative diagnostic that overt pattern exploitation is uncommon in the sampled trajectories. We treat the audit as complementary to cross-simulator evaluation, held-out gym transfer, and real-user evaluation, rather than as definitive evidence of human-level robustness.

H Use of LLMs

In this work, LLMs are used strictly for research support rather than as sources of substantive content. Their use falls into three categories: (i) serving as simulated users in our gym environments, (ii) providing baselines for evaluating trained models, and (iii) assisting with language refinement during paper writing. Both open-source and closed-source LLMs are employed for simulation and evaluation to ensure a comprehensive analysis. For writing support, we used GPT-5 solely to polish text (improving coherence and grammar) while all ideas, logic, results, and technical contributions originate from the authors. To safeguard rigor, we have carefully reviewed all LLM-refined texts to confirm that no hallucinated content was introduced and that the original arguments, findings, and perspectives were faithfully preserved.

I Discussions

Turn-level signals: what we learn and what they enable. UserRL lets us study how turn-level credit schemes shape multi-turn behavior beyond

overall trajectory scoring. Our results suggest that simple, generic mappings (e.g., EM or R2G applied to outcome-based rewards) can provide a useful training signal, but they do not always reflect the *quality* of intermediate actions. For instance, EM assigns the same value to many zero-reward turns, and R2G tends to favor turns close to reward events; both can blur the difference between genuinely informative questions and unproductive probing (such as those in the IntentionGym). The underlying challenge is that many environments reliably measure *progress*, but provide limited supervision about *which turn caused it* and *why it helped*. A key advantage of UserRL is that it makes this gap actionable: one can plug in richer turn-level evaluators (e.g., relevance, information gain, or verifier-graded question quality) and compare them under the same environments, enabling systematic progress on turn-level credit assignment while retaining the benefits of trajectory-level scoring.

Balance of rigor and flexibility in user simulation. User simulation is essential for scaling RL training and enabling dynamic multi-turn rollouts. To make trained agentic models robust to diverse user behaviors, simulations must incorporate variation in responses, such as different strategies, tones, and communication styles. Future extensions of our framework could introduce richer user profiles in gym environments to better approximate real-world diversity and further strengthen robustness. At the same time, benchmarks must preserve rigor: when user interaction is the primary challenge, fairness across models is critical to ensure comparability. This inevitably reduces naturalness and requires rule-based structures to maintain consistency. In our eight gym designs, we address this tension by combining LLM-driven dynamic responses with rule-based strategies that track task completion. For example, in TravelGym the simulated user categorizes the agent's utterance before generating a guided response. This illustrates the inherent tradeoff between flexibility and rigor in user simulation, highlighting an direction for future work on balancing the two for robust training.

Balance of efficiency and effectiveness in user interaction. In Section 4.3, we introduced two complementary metrics to evaluate user interaction: *Effective Turns* and *Time-Weighted Performance*. While higher values for both are desirable, we should note that inherently they may conflict. Models that achieve effectiveness by exploiting many turns risk creating tedious, inefficient conversations, while overly efficient agents may fail to fully capture user intent. Our metrics address this by accounting for not only the number of turns but also the rewards they deliver. This also mirrors real-world tradeoffs: users may tolerate a few clarifying questions, but prolonged or repetitive probing quickly becomes frustrating. Therefore, training and evaluation should consider how to balance efficiency and effectiveness, potentially through improved metrics and reward designs that encourage models to be both accurate and concise in user interaction.

J Significance, Scope, and Clarifications

J.1 Why UserRL Matters

A Missing Testbed for User-Centric Agent RL. Recent agent research has made rapid progress on tool use, planning, web interaction, and reasoning in structured environments. These settings are important, but they often treat the user as an external goal provider rather than as a dynamic part of the environment. In realistic user-facing scenarios, however, the agent must interact with users over multiple turns, clarify underspecified goals, elicit preferences, adapt to partial feedback, and decide when to act. UserRL focuses on this under-studied setting: **reinforcement learning for user-centric multi-turn interaction**. The goal is not only to evaluate whether an agent can solve a task, but also to study how it learns to communicate, clarify, search, persuade, and make decisions in response to evolving user signals.

From Isolated Benchmarks to a Unified RL-Ready Suite. A central contribution of UserRL is to turn heterogeneous user-interaction tasks into a unified, trainable testbed. Prior benchmarks often study one interaction pattern at a time, such as preference elicitation, task-oriented dialogue, persuasion, or tool use. UserRL instead places these abilities under a shared Gymnasium-style abstraction with standardized observations, actions, rewards, and rollout interfaces. This makes it possible to train and evaluate agents across multiple user-centric skills within one framework. The contribution is therefore not merely a collection of tasks, but a reproducible environment for studying how user-facing agents improve under matched training, evaluation, and reward-design conditions.

A Practical Interface for Studying Interactive Agents. UserRL uses a deliberately simple Action/Search/Answer interface. This abstraction is expressive enough to cover direct user communication, external information retrieval, and final task completion, while remaining easy to implement and extend. This design matters because agent benchmarks often differ in their interaction protocols, making it difficult to determine whether performance differences come from model capability, task design, or interface-specific behavior. By standardizing the interaction interface, UserRL provides a controlled platform for comparing models, training objectives, reward formulations, and simulated users across diverse environments.

Reward Design as a Controlled Experimental Variable. Multi-turn user interaction naturally produces delayed and incremental feedback. Some turns may directly yield reward, while others may only become useful later by eliciting information or narrowing the solution space. UserRL makes this structure explicit by separating turn-level reward shaping from trajectory-level scoring. This abstraction is not intended as a new RL algorithm; rather, it provides a benchmark-integrated way to study how different credit structures affect learning under the same optimizer and environments. Our results show that trajectory-level scoring, especially reward-to-go scoring, has a substantial effect on both effectiveness and efficiency. This highlights UserRL's value as a testbed for understanding design choices that are difficult to isolate.

A Bridge Between Simulation and Real User Interaction. Fully human-in-the-loop training and evaluation are valuable but expensive, variable, and difficult to reproduce. UserRL therefore uses LLM-based simulated users to support scalable multi-turn rollouts, while also including real-user evaluation to test whether learned behaviors remain useful beyond simulator-only settings. The expanded real-user results in Table 5 compare raw and trained models across four gyms, showing that the trained model maintains a clear advantage under real human interaction. This supports the practical relevance of the framework: UserRL-trained policies do not merely optimize against a fixed simulator, but learn interaction behaviors that remain visible when the user is a real person.

A Testbed for Both Evaluation and Improvement. UserRL is designed not only to rank models, but also to support agent improvement. Because the environments expose rollouts, rewards, user responses, and task outcomes through a standard interface, they can be used for supervised fine-tuning, reinforcement learning, reward-design studies, simulator comparisons, and behavioral analysis. This makes UserRL useful as a research tool: it enables controlled studies of how initialization, simulator choice, trajectory scoring, and turn-level credit assignment affect user-centric agent behavior. The framework therefore connects benchmark construction with model training, which is essential for making progress on interactive agents rather than only measuring their current limitations.

J.2 Design and Empirical Clarifications

UserRL Targets User-Centric Interaction, Not a Single Atomic Skill. User-centric assistance is broad, and no single suite can cover every possible form of human-agent collaboration. UserRL deliberately focuses on a representative set of interaction skills that are central to user-facing agents: clarification, preference elicitation, persuasion, strategic questioning, tool use, search, and final decision-making. The eight gyms should therefore be understood as an extensible coverage of common user-centric interaction patterns rather than an exhaustive taxonomy. This scope is intentional: it allows the benchmark to remain controlled and trainable while still capturing multiple complementary dimensions of user interaction.

The Contribution Is More Than Engineering Integration. Integrating multiple tasks into a common RL-ready interface is itself important for agent research. Without a shared environment abstraction, it is difficult to compare reward designs, simulator choices, training strategies, or transfer behavior across tasks. UserRL provides this missing infrastructure for user-centric agent RL. The framework makes heterogeneous interaction tasks jointly trainable, exposes common reward and rollout hooks, and enables controlled cross-environment analysis. Thus, the value of UserRL lies not in claiming that every individual task is new, but in making user-centric interaction a standardized object of RL study.

The Reward Framework Is Benchmark-Integrated Rather Than Algorithm-Centric. UserRL does not claim that turn-level or trajectory-level credit assignment is conceptually unavailable in prior RL frameworks. The key distinction

is that UserRL integrates these reward abstractions directly into a suite of user-centric gym environments, allowing reward formulations to be compared under matched tasks, simulators, and optimization settings. This makes reward design an experimental variable rather than an implementation detail. In this sense, UserRL complements general-purpose agent RL frameworks by providing a focused testbed where credit assignment can be studied systematically for user interaction.

SFT and RL Play Complementary Roles. The purpose of the SFT cold start is not to replace RL, but to provide the model with an initial interaction policy that makes online multi-turn optimization effective. As shown in Section G.2, SFT-only models already acquire useful interaction behavior, confirming that supervised trajectories provide a strong foundation. However, SFT+RL achieves the strongest final performance, showing that online reinforcement learning further adapts the policy to the reward structure and environment dynamics. The intended conclusion is therefore complementary: SFT bootstraps user interaction, and RL refines it through task-grounded feedback.

Raw Model Scaling Alone Does Not Solve the Interaction Problem. The added Qwen3-8B raw baseline strengthens the comparison between model scale and interaction training. Larger raw models can improve on some tasks, but scaling alone does not consistently close the gap to UserRL-trained models. This suggests that user-centric performance is not simply a byproduct of general model capacity. Instead, agents benefit from training that explicitly exposes them to multi-turn interaction, user feedback, and task-grounded reward signals. UserRL is therefore useful for studying how interaction-specific training unlocks capabilities that may remain underdeveloped in raw models.

Protocol Learning Does Not Explain the Main Gains. A natural concern is that UserRL-trained models may improve mainly because they learn to follow the Action/Search/Answer format. We address this in Section G.4. Raw and closed-source models already achieve high format-compliance rates across representative gyms, often above 95%, while still differing substantially in task success. This indicates that parseability is necessary but not sufficient: successful agents must also ask useful questions, retrieve relevant information, use feedback appropriately, and make correct final decisions. The performance gains therefore reflect more than learning a surface-level output schema.

Closed-Source Models Are Used as Reference Points Under a Shared Protocol. The comparison with closed-source models should be interpreted within the UserRL protocol. These models are strong general-purpose references, but they are not specifically trained for the UserRL interaction interface. The main takeaway is not that smaller trained open-source models are universally superior to closed-source systems. Rather, the results show that targeted user-centric training can substantially improve performance under a controlled interaction protocol, while closed-source models remain strong on tasks requiring broader capabilities such as search, strategic guessing, and general reasoning. This comparison helps identify where interaction training helps and where broader model capability remains important.

LLM-Based Simulated Users Are a Scalable Training Mechanism. UserRL uses LLM-based simulated users because multi-turn user interaction requires adaptive responses at scale. The simulator's role is to provide context-aware user feedback while the environment maintains rule-based task states and verifiable outcomes. This design balances flexibility with reproducibility: user responses can adapt to the conversation, but task completion and reward computation remain controlled. Simulator choice is therefore an experimental component of the framework, not an uncontrolled confound. Our results further show that stronger simulators can improve training, while open-source simulators can still produce policies that transfer positively to stronger simulated users and preliminary real-user evaluation.

Simulator-Specific Exploitation Is Diagnosed Through Multiple Signals. Because both agents and simulated users can be LLM-based, it is important to check whether the agent simply exploits simulator-specific patterns. UserRL addresses this concern through several complementary signals: cross-simulator evaluation, held-out gym evaluation, real-user evaluation, and manual trajectory audit. In Section G.5, most inspected trajectories are grounded in information elicited during interaction, and fully unsupported guessing is uncommon. This does not require claiming that simulator bias

is impossible; rather, it shows that the observed gains are not easily reduced to an obvious simulator shortcut.

Real-User Evaluation Tests Practical Transfer Beyond Simulation. The real-user study is designed as a practical transfer check from simulated interaction to human interaction. The expanded results compare raw and trained models under both GPT-4o simulated users and real users across four gyms. The raw model benefits only modestly from human cooperation, while the trained model maintains a clear advantage. This supports the conclusion that UserRL training improves interaction behavior in a way that remains useful with real users. The study should be viewed as preliminary but meaningful evidence: it is not intended to exhaustively characterize all human populations, but it directly tests whether simulator-trained behaviors persist under human interaction.

Ethics and Human-Study Scope Are Clearly Delimited. The real-user evaluation involves minimal-risk, non-sensitive interaction tasks with voluntary participation and anonymized logs, as detailed in Section F.3. The study is used to evaluate task interaction behavior, not to collect sensitive personal information or make claims about human subjects beyond the benchmark setting. This scope keeps the evaluation focused, reproducible, and appropriate for the paper's purpose: validating whether UserRL-trained agents remain effective when interacting with real users under controlled tasks.

General Capabilities Are Largely Preserved. User-centric RL should improve interaction behavior without substantially damaging general capabilities. The additional evaluation on MATH500 and LiveCodeBench in Appendix G shows that UserRL-trained models retain more than 95% of the raw baseline performance in all tested cases. This suggests that the training procedure improves user-centric interaction while largely preserving mathematical and coding abilities. The result supports the practical viability of UserRL-style training as a targeted adaptation method rather than a narrow optimization that severely compromises general competence.

UserRL Is Designed for Extensibility. The current suite includes eight gyms, but the framework is not limited to these tasks. New environments can be added by implementing the same reset, step, action, reward, and user-simulation interfaces. New simulators, reward functions, trajectory scorers, and training algorithms can also be plugged into the same framework. This extensibility is important because user-centric interaction is a broad research area: UserRL provides an initial standardized foundation on which richer user profiles, more complex task dependencies, and larger-scale human evaluations can be built.

Final Takeaway. UserRL positions user-centric interaction as a concrete frontier for agent RL. Its main contribution is to provide a unified, reproducible, and extensible testbed where agents can be trained and evaluated through multi-turn interaction with simulated and real users. The framework clarifies how reward design, initialization, simulator choice, protocol compliance, and model scale affect user-facing behavior. By combining diverse gyms, standardized interfaces, controlled rewards, simulator-based training, real-user evaluation, and diagnostic analyses, UserRL offers a practical foundation for developing agents that are not only capable task-solvers, but adaptive collaborators in interactive settings.

Agent System Prompt Template (TauGym as Example)

```
## Task
You are an agent that actively interact with a specific environment. The following are the details
of the environment and your action space.


## Environment Description
{{Placeholder: The environment description for this gym; Example of TauGym in the following}}

TauGym is an environment where you need to interact with both the user and internal tools
to fulfill the user's request. You should thoroughly understand the user's goal, figure out what
information is needed, and get this information through querying the user or leveraging the internal
tool step by step.


## Action Space
You should call the tool interact_with_env to interact with the environment. The action should be
one of the following: search, action, or answer.


## Action Description
{{Placeholder: The definition of particular action under this specific gym's context; Example of
TauGym in the following}}

    * search:  If you choose search, you must specify either 'tools' or 'help' in the
content field. Giving 'tools' will return a list of internal tools, including their descriptions
and required arguments, which you can later call through choosing answer. Giving 'help' will return
a general guidance on how to interact with the environment effectively.
    * action: If you choose action, you will communicate directly with the user through the message
you write in the content field. Ask clear and specific questions to gather the information needed
to fulfill the user's request. Keep in mind that the user may not have all the necessary details,
so you might need to both request additional user input and call internal tools step by step to
reach the goal.
    * answer: If you choose answer, you must provide an internal tool call in the content field,
with the tool name and its arguments in JSON format (e.g. {"name": tool_name, "arguments": {"arg_1":
"value_1", "arg_2": "value_2"}}).


## Important Notes
    * In each step of interaction, first write your thoughts and analysis between <think> and
</think> to carefully decide your next step. Only after providing this reasoning should you call
the interact_with_env tool to interact with the environment. Always present your reasoning before
making the tool call.
    * The total number of rounds that you can interact with the environment is limited. You should
smartly {{Placeholder: Remind the task's goal}}, so that you can fulfill the user's request in the
most efficient way.
    * Usually you should {{Placeholder: Hint on the common way to interact with this gym in order
to achieve the task goal}}.
    * Be bold, creative and smart in your interaction with the environment! Let's begin!
```

Figure 16: System prompt template for agent before its interaction with the specific gym environment.