THE OPPONENT LEARNS TOO: ALTERNATING ADVERSARIAL TRAINING FOR STRONGER CONVERSATIONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conversational agents need to coordinate dialogue and tool use across multiple turns to address evolving user needs. To develop these abilities, agents are trained with simulated users, and recent work has extended this paradigm to jointly train users and agents, yielding interactions that evolve throughout training. However, existing user simulators do not necessarily adapt to the current agent's remaining weaknesses, limiting their ability to provide targeted training signals for further improvement. In this paper, we propose **A**lternating **T**raining with **O**pponent **A**daptation (ATOA), which uses failures of the current agent to guide successive rounds of adversarial training between the agent and user simulators. Specifically, each round begins by fixing the agent and updating the user simulator using utterances from interactions in which the agent fails. The updated simulator is then fixed to train the agent using successful trajectories together with pairs of successful and failed attempts on the same task. Failures of the newly trained agent are used to guide the next simulator update, allowing the training challenges to evolve across successive rounds. To preserve task consistency, ATOA fixes the user's behavioral decisions in advance and trains only how those decisions are expressed in language, while task completion is independently verified from the resulting environment states. Experiments across different scenarios in τ^2-bench show consistent improvements in agent task success.

1 INTRODUCTION

Conversational agents need to coordinate dialogue and tool use across multiple turns to address evolving user needs (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024). In such interactions, users may provide information gradually, revise their requests, or lose patience before a task is finished, requiring agents to learn both what to ask and how to act (Shim et al., 2026). Recent methods develop these abilities by training agents on multi-turn interactions with simulated users (Prabhakar et al., 2025; Wu et al., 2025; Zhao et al., 2025; Qian et al., 2025; Gao et al., 2026a;b; Peng et al., 2026). However, most training pipelines keep the user simulator fixed while the agent continues to improve. As a result, the simulator cannot adapt to the agent's changing weaknesses, and the resulting training signal may become increasingly tied to a particular counterpart.

Increasing dialogue diversity alone does not resolve this limitation, since a fixed simulator still generates interactions under the same response policy. Training a user model to better reflect human behavior (Naous et al., 2026; Zhang et al., 2026a; Wu et al., 2026; Zhang et al., 2026b; Cao et al., 2026) or conditioning it on diverse personas (Chopra et al., 2026; Dou et al., 2025; Wang et al., 2025b) broadens the interactions available, but does not by itself make them adapt to the current agent's failures. As training proceeds, the agent may exploit regularities of this policy, producing gains that depend on the training opponent (Suh et al., 2026; Yu et al., 2026). Figure 1(a) illustrates this issue: improvements against a fixed training counterpart may not transfer to other interaction partners. IB-RL (Wang et al., 2026b) provides a clear example, showing that unilaterally trained negotiation agents achieve 86.4–89.7% agreement with the counterparts used during training, but only 46.2–56.3% when interacting with copies of the same trained policy. Earlier work similarly finds that training against a fixed partner can improve performance with that partner while reducing

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

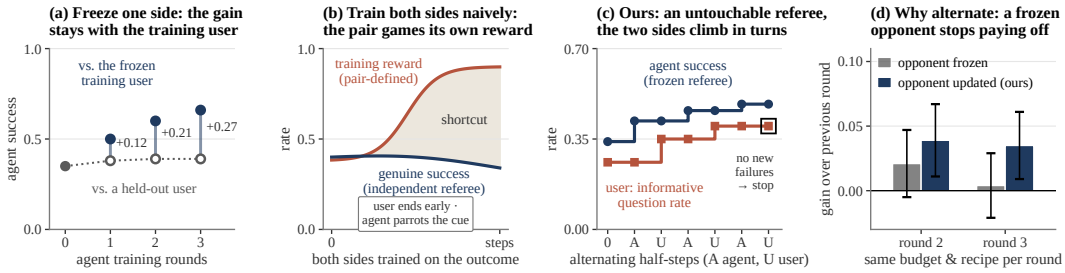


Figure 1: **Motivation and pilot study.** (a) A fixed training user can lead to counterpart-specific gains. (b) Naive bilateral training can increase training reward through shortcuts without improving independently verified success. (c) With fixed task rules and verification, alternating user and agent updates progressively improve agent success. (d) Updating the opponent provides larger gains than continuing with the previous opponent from the same agent checkpoint.

agreement with human users (Lewis et al., 2017). Together, these findings motivate adapting the training counterpart as the agent improves.

This limitation has motivated training schemes in which the user simulator evolves alongside the agent. Earlier work explored joint training of dialogue agents and user simulators (Liu & Lane, 2017; Takanobu et al., 2020; Tseng et al., 2021), while more recent methods such as SCOPE (Yu et al., 2026) and IB-RL (Wang et al., 2026b) develop adaptive opponents through population training, separate learning signals, and staggered updates. However, adapting the user creates a new problem: the user can change how difficult the task is instead of simply exposing the agent's weaknesses. For example, it may make the task artificially harder by refusing to provide necessary information or ending the interaction early, or artificially easier by becoming overly cooperative. Related failure modes have been observed in SEAD, SCOPE, and IB-RL (Dai et al., 2026; Yu et al., 2026; Wang et al., 2026b). Figure 1(b) illustrates the resulting mismatch: the pair's training reward rises, while task success measured by an independent evaluator does not improve accordingly. Together, these findings motivate a more controlled form of bilateral training, where user updates are separated from the task conditions that determine success.

We therefore study whether alternating user and agent updates remain effective under these fixed conditions. In a controlled pilot study, we keep the user decision rules and task verification fixed, and update the user and agent in turn. Figure 1(c) shows that agent task success improves progressively across these alternating rounds under the same evaluation standard. Figure 1(d) further isolates the effect of opponent adaptation by comparing continued agent training with the previous user against training with an updated user from the same agent checkpoint. The updated user provides additional gains, showing that adapting the opponent can continue to expose useful training challenges as the agent improves. These results motivate repeated adversarial alternation between the user and agent under fixed task conditions, with training challenges evolving alongside the agent.

In this paper, we propose **A**lternating **T**raining with **O**pponent **A**daptation (ATOA), a multi-round adversarial training framework that uses failures of the current agent to guide updates of its training opponent. To preserve task consistency, we construct the user simulator with a fixed decision controller and a trainable language model, building on controllable user simulation (Tennenholtz et al., 2026; Mehri et al., 2026). With the agent fixed, the user language model is updated using utterances from failed interactions to better expose the agent's current weaknesses while following the same task rules. The updated simulator is then fixed to train the agent using successful trajectories together with same-task successful–failed pairs. Failures of the newly trained agent guide the next simulator update, forming a multi-round loop in which the training challenges evolve with the agent. Throughout this process, the user decision rules and environment-based task verification remain fixed. To assess ATOA, we evaluate it across different scenarios in τ^2-bench (Barres et al., 2025). The results show improved agent task success, with further gains emerging over multiple training rounds. Comparisons with fixed-opponent training and component ablations further support the effectiveness of alternating updates and opponent adaptation. Our contributions are:

- We construct a controllable user simulator that samples diverse behavioral scripts, varying information disclosure, request changes, and patience while preserving fixed task rules.

- We propose ATOA, a multi-round alternating adversarial training framework that updates the user from the current agent's failures and then trains the agent against the updated user.
- We demonstrate improved task success across scenarios in τ^2-bench and validate multi-round alternating training through controlled comparisons and ablations.

2 Related Work

User simulation. User simulators provide repeatable interaction environments for training and evaluating conversational agents. UserLM learns the user role from human conversations, while UserLM-R1, HumanLM, and Turing-RL improve behavioral alignment through reasoning, latent states, or human-likeness rewards (Naous et al., 2026; Zhang et al., 2026a; Wu et al., 2026; Wang et al., 2026d). Persona-based methods broaden behavioral coverage through explicit or inferred user profiles (Wang et al., 2025a; Zhu et al., 2026; Chopra et al., 2026). Other approaches improve controllability by tracking user goals or explicitly controlling the intent expressed at each turn (Mehri et al., 2026; Tennenholtz et al., 2026; Wang et al., 2026a), while non-collaborative simulation introduces behaviors such as impatience and incomplete information (Shim et al., 2026). Despite these advances, simulators used for agent training are typically fixed after construction, so their behavior does not adapt to the agent's changing weaknesses.

Agent training with fixed user models. In many agent-training pipelines, the user simulator remains fixed while only the agent is optimized. One line of work uses simulated interactions to generate and select dialogues for agent fine-tuning (Prabhakar et al., 2025; Cao et al., 2026; Gu et al., 2026). Another optimizes the agent directly through simulated interaction using rewards for collaboration, task completion, or interaction cost (Wu et al., 2025; Zhao et al., 2025; Qian et al., 2025; Gao et al., 2026a;b). To maintain informative training signals, SEAD adapts user profiles, AT-GRPO adjusts interaction difficulty, and SENTINEL generates tasks from agent failures while leaving the user language model fixed (Dai et al., 2026; Peng et al., 2026; Wang et al., 2026e). These methods diversify the agent's training experience, but the user model itself does not repeatedly adapt to the agent's changing weaknesses. As a result, improvements can remain tied to the training counterpart (Suh et al., 2026; Yu et al., 2026). This limitation has motivated methods that update the user model alongside the agent.

Joint and adversarial training. Joint training and self-play address this limitation by allowing multiple dialogue roles to learn from their interactions. Early methods jointly optimize system and user policies through iterative learning or role-aware rewards (Liu & Lane, 2017; Takanobu et al., 2020; Tseng et al., 2021). Recent methods extend this idea to shared-policy social self-play and feedback-driven negotiation strategies (Jiang et al., 2026; Mangla et al., 2025). More closely related to our setting, SCOPE uses population co-training and an adaptive curriculum, while IB-RL combines separate learning signals with historical opponents and staggered updates (Yu et al., 2026; Wang et al., 2026b). These methods make the training counterpart adaptive, but bilateral optimization also introduces new failure modes. Their ablations report excessive cooperation, arbitrary refusal, and reward shortcuts under alternative update schemes (Yu et al., 2026; Wang et al., 2026b). ATOA instead combines multi-round adversarial alternation with fixed user decision rules and environment-based task verification, allowing the training opponent to evolve while keeping the conditions that define task success fixed.

3 Method

ATOA frames conversational-agent training as a repeated adversarial loop between the agent and an adaptive user simulator. In each round, failures of the current agent are first used to update the simulator, and the updated simulator then provides new interactions for training the agent. We first formalize this alternating process, then describe how user adaptation is constrained by fixed task rules and how the user and agent are updated in each round.

3.1 Interaction Setup and Alternating Training

We consider tasks $x \in \mathcal{X}_{\text{train}}$, each defined by an initial environment state, a user goal, and a set of task rules. An agent π_θ interacts with a user simulator U_ϕ through multi-turn dialogue and

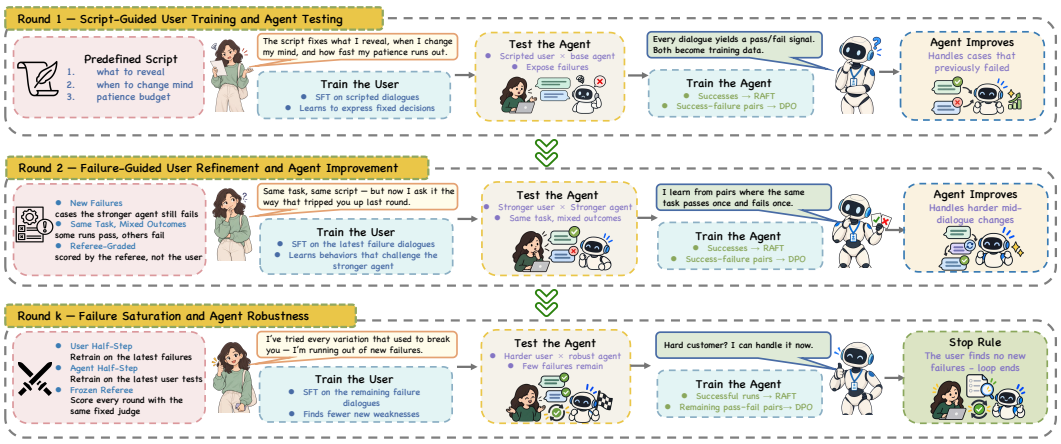


Figure 2: **Overview of ATOA.** The user simulator is first initialized from predefined behavioral scripts. In each subsequent round, failures of the current agent are used to adapt the user, after which the updated user generates new interactions for agent training with RAFT and DPO. User decision rules and task verification remain fixed throughout the alternating loop, which continues until further opponent adaptation no longer provides useful training challenges.

tool use. At each agent turn t, the agent observes the interaction history h_t and produces either a conversational response or a tool action:

$$a_t \sim \pi_\theta(\cdot \mid h_t). \tag{1}$$

Tool actions may modify the environment, and each completed interaction trajectory τ terminates at an environment state s_T. A fixed verifier assigns a binary task outcome $R_x(\tau) \in \{0, 1\}$ based on the terminal environment state. Since task success is determined from the resulting environment rather than either model's claim of completion, neither side directly controls the success criterion. The same verifier is used throughout training to identify the interactions that provide supervision for both user and agent updates.

ATOA organizes training into alternating rounds. Before the first round, the user simulator is initialized from behavioral scripts. In each subsequent round k, the current agent $\pi_{\theta_{k-1}}$ is first fixed, and its failed interactions are used to adapt the user simulator. The resulting user U_{ϕ_k} is then fixed and used to generate new interactions for training the agent, producing π_{θ_k}. Failures of the updated agent provide the signal for the next user update, closing the multi-round loop. Throughout this process, the user decision rules and task verifier remain fixed, while the user language model and agent are updated in turn, allowing the training challenges to evolve across successive rounds.

3.2 Controllable User Simulation

ATOA allows the user to adapt its language without changing the task conditions. To achieve this, we separate *what the user does* from *how the user expresses it*. We represent each simulated user with a behavioral script that specifies its task-consistent behavior throughout an interaction, including what information can be revealed, when requests may change, and how patience evolves. Before each interaction, we sample a script $\xi \sim p(\xi \mid x)$ for the current task x.

A fixed controller C follows this script and the evolving interaction history to determine the user's directive at each turn. The controller and user language model operate as

$$d_j = C(x, \xi, h_j), \qquad u_j \sim \mu_\phi(\cdot \mid h_j, d_j). \tag{2}$$

Here, d_j specifies the behavior and information permitted at user turn j, while μ_ϕ expresses this directive in natural language. The controller therefore determines information disclosure, request changes, and dialogue termination, whereas the trainable language model determines only how these decisions are expressed. Because the controller reacts to the evolving interaction history, different agent behaviors can still induce different conversations under the same underlying script.

Directive enforcement. Generated user utterances are checked against the controller directive before entering the interaction. The check ensures that required information is expressed, unrevealed information is not disclosed prematurely, and termination follows the controller decision. Invalid generations are resampled and, if necessary, replaced by a deterministic fallback. User-side tool actions, when required by the task, are also selected by the controller rather than the language model. These constraints keep user behavior fixed while allowing the language model to vary its expression.

User simulator initialization. Before alternating training begins, we first apply supervised fine-tuning (SFT) to teach the user language model to follow the behavioral scripts and controller directives. We construct an initialization dataset $\mathcal{D}_{\text{script}}$ of context–directive–utterance triples (h, d, u) from scripted interactions, covering different information-disclosure patterns, request changes, and patience states. The user model is trained with

$$\mathcal{L}_U(\phi; \mathcal{D}) = -\mathbb{E}_{(h,d,u)\sim\mathcal{D}} \left[\log \mu_\phi(u \mid h, d)\right]. \tag{3}$$

Starting from the pretrained checkpoint ϕ_{base}, this SFT stage produces the initial user simulator U_{ϕ_0}. The same script-based examples are retained during opponent updates as anchor data, helping preserve script-following behavior while the simulator adapts to failures of the current agent.

3.3 Failure-Guided Opponent Adaptation

Once user behavior is constrained by the fixed controller, ATOA adapts the user language model using failures of the current agent. After each agent update, we collect repeated interactions between the current agent $\pi_{\theta_{k-1}}$ and the retained user $U_{\phi_{k-1}}$. Rather than training on all failed tasks, we focus on tasks for which the agent exhibits mixed outcomes, indicating that the task is solvable but not yet handled consistently. Let $\mathcal{R}_{k-1}(x)$ denote the repeated attempts for task x, with $N_x = |\mathcal{R}_{k-1}(x)|$. We focus on tasks for which the current agent succeeds in some attempts but fails in others:

$$\mathcal{X}_{k-1}^{\text{mix}} = \left\{ x \in \mathcal{X}_{\text{train}} : 0 < \sum_{\tau\in\mathcal{R}_{k-1}(x)} R_x(\tau) < N_x \right\}. \tag{4}$$

These tasks identify interactions near the current agent's capability boundary.

For each $x \in \mathcal{X}_{k-1}^{\text{mix}}$, we extract user utterances from failed attempts together with their interaction contexts and controller directives. After filtering uninformative or invalid examples, these utterances form a failure-guided dataset $\mathcal{D}_{k-1}^{U}$. We combine the new examples with the fixed script-based data introduced above and train a candidate user from the same pretrained checkpoint:

$$\widetilde{\phi}_k = \text{SFT}_U\left(\phi_{\text{base}}; \mathcal{D}_{\text{script}} \uplus \mathcal{D}_{k-1}^{U}\right), \tag{5}$$

where $\uplus$ denotes dataset concatenation and SFT_U minimizes the user objective in Eq. 3.

The failure-derived utterances adapt the simulator to the current agent's weaknesses, while the original script examples preserve adherence to the fixed behavioral rules. Importantly, task outcomes are used only to select which user utterances enter the SFT dataset, rather than as a direct optimization reward for the user model. The adapted user is then fixed and used to generate new interactions that provide training data for the subsequent agent update.

3.4 Agent Learning against the Adapted User

Once the adapted user U_{ϕ_k} is selected, we freeze it and collect a new set of interactions $\mathcal{C}_k$ with the current agent $\pi_{\theta_{k-1}}$. Each trajectory is labeled as successful or failed by the same fixed verifier. ATOA then updates the agent in two stages: it first learns from verified successful trajectories, and then learns to prefer successful attempts over failed attempts on the same task. Starting from θ_{k-1}, these two stages produce the next agent π_{θ_k}.

Learning from successful trajectories. We first apply reward-filtered fine-tuning following RAFT (Dong et al., 2023), using only trajectories that achieve verified task success. Let $\mathcal{D}_k^+$ contain agent actions (h_t, a_t) extracted from successful trajectories in $\mathcal{C}_k$. We optimize

$$\mathcal{L}_{\text{RAFT}}(\theta) = -\mathbb{E}_{(h,a)\sim\mathcal{D}_k^+} \left[\log \pi_\theta(a \mid h)\right]. \tag{6}$$

Only agent outputs are supervised, while user utterances and tool results remain part of the interaction context. Starting from θ_{k-1}, this stage produces an intermediate agent $\pi_{\bar{\theta}_k}$ that learns from actions leading to verified task completion.

Learning from same-task comparisons. Successful trajectories provide positive demonstrations, while failed attempts reveal alternative behaviors on the same task. We therefore construct preference pairs $\mathcal{P}_k = \{(x, \tau^+, \tau^-)\}$ from $\mathcal{C}_k$, where τ^+ and τ^- correspond to the same task x, with $R_x(\tau^+) = 1$ and $R_x(\tau^-) = 0$. Each trajectory retains its own interaction history, so the comparison reflects the agent behaviors actually observed in the two interactions.

We then apply trajectory-level direct preference optimization (DPO) (Rafailov et al., 2023) over the agent actions in each pair. Let $\mathcal{I}_A(\tau)$ denote the agent action indices in trajectory τ. We define

$$\ell_\theta(\tau) = \sum_{t \in \mathcal{I}_A(\tau)} \log \pi_\theta(a_t \mid h_t), \qquad \Delta_\theta(\tau) = \ell_\theta(\tau) - \ell_{\bar{\theta}_k}(\tau). \tag{7}$$

Here, $\log \pi_\theta(a_t \mid h_t)$ denotes the summed token log-probability of the agent output at turn t. Using the post-RAFT agent $\pi_{\bar{\theta}_k}$ as the reference, we obtain the final agent π_{θ_k} by minimizing

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}_{(x,\tau^+,\tau^-)\sim\mathcal{P}_k} \left[\log \sigma \left(\beta \left[\Delta_\theta(\tau^+) - \Delta_\theta(\tau^-) \right] \right) \right], \tag{8}$$

where σ is the logistic function and $\beta > 0$ controls the scale of the preference objective. The two stages are complementary: RAFT learns from verified solutions, while DPO contrasts successful and failed behaviors on the same task.

3.5 Multi-Round Alternation and Stopping

ATOA repeats the two update steps to continually adapt the training opponent to the evolving agent. After obtaining π_{θ_k}, we collect new interactions with the retained user U_{ϕ_k}. Failures of the updated agent are then used to construct the supervision for the next user update, producing a new opponent candidate for round $k + 1$. In this way, each round uses the latest agent failures to refresh the challenges presented in the following round.

Candidate selection and stopping. A newly trained user candidate is retained only if it provides sufficiently useful challenges to the same fixed agent while satisfying the task-consistency constraints. Otherwise, the previous user is kept for the next agent update. Across rounds, ATOA applies a prespecified stopping rule and terminates when further opponent updates no longer provide useful training signal. The controller and task verifier remain unchanged throughout, so only the learned opponent and agent evolve across rounds.

4 Experiments

We evaluate ATOA from three perspectives: overall task performance across service scenarios, gains over successive alternating rounds, and the contribution of opponent adaptation to the resulting training signal.We further assess robustness with cooperative users and use controlled ablations to isolate the effects of individual components.

4.1 Experimental Setup

Tasks and models. We evaluate ATOA on the retail, airline, and telecom environments of τ^2-bench (Barres et al., 2025), using Qwen3.5-9B and Qwen3.6-27B as the backbones. Our main analyses focus on Qwen3.6-27B, while Qwen3.5-9B is included to examine whether the observed trends extend across model scales. Domain-specific task splits, evaluation sets, and the exact protocols used in each scenario are described in Appendix A.

Comparison methods. We compare ATOA with baselines spanning four complementary directions. *User-simulation methods* include Persona Policies (Chopra et al., 2026), RealUserSim (Zhu et al., 2026), and UserLM-R1 (Zhang et al., 2026a). *Adaptive training methods* include Calibrated Interactive RL (Wang et al., 2026c), SENTINEL (Wang et al., 2026e), and SEAD (Dai et al., 2026), which modify the simulator, curriculum, or training experience as the agent learns. *Agent self-improvement*

methods include EigenData (Gao et al., 2026a) and divergence-point preference learning (Tang, 2026). Finally, *bilateral training* is represented by an IB-RL-inspired cooperative variant (Wang et al., 2026b). We additionally report the untrained base agent and fixed-user training as reference baselines. Table 1 summarizes the full comparison.

Training. ATOA first initializes the user simulator with script-based SFT and then alternates failure-guided user adaptation with agent training. The agent learns from verified successful trajectories using RAFT, followed by trajectory-level DPO on same-task successful–failed pairs. All methods use matched agent backbones and shared task environments; complete optimization settings are provided in Appendix A.

Evaluation. We report task success under both strict and cooperative user settings. Scripted-user evaluation keeps the user decision rules fixed and verifies task completion from the final environment state; retail checkpoints are additionally evaluated with an external GPT-5.5 language renderer. Cooperative evaluation uses the standard GPT-4.1 user from τ^2-bench.

4.2 Main Results Across Scenarios

Evaluation protocol. Table 1 compares ATOA with representative baselines across three service scenarios and two model scales under a unified scripted-user evaluation. All methods are evaluated with the same frozen behavioral controller and the same script-trained language renderer, so differences in task success reflect the trained agents rather than changes in the evaluation user. For the Fixed-user training baseline, the training counterpart uses the same controller but an untrained language renderer; ATOA instead initializes the renderer with script-based SFT and subsequently allows it to adapt during alternating training. We report mean $\pm$ evaluation uncertainty estimated by resampling task cards as complete clusters, retaining all valid repeated interactions within each selected card. This uncertainty concerns evaluation sampling for fixed trained checkpoints, not variability across training seeds.

Overall performance. ATOA achieves the best performance in four of the six settings, ties the best baseline on Qwen3.5-9B Airline, and ranks second on Qwen3.6-27B Airline. Its advantage is most pronounced in Retail and Telecom, where ATOA consistently outperforms fixed-user training and other adaptive training approaches across both model scales.

Consistency across model scales. The gains remain consistent across model scales. On Qwen3.6-27B, ATOA reaches 47.84% on Retail and 85.76% on Telecom, exceeding the strongest competing baselines by 4.64 and 4.65 percentage points, respectively. Similar improvements appear with Qwen3.5-9B, indicating that the benefit of opponent adaptation is not limited to a single model scale. Airline remains substantially harder for all methods, with only modest separation among the strongest approaches. Overall, ATOA preserves its relative advantage as model scale increases.

The cross-domain score pattern differs from several existing τ^2-bench reports because our interaction protocol is also different. In Telecom, required user-side operations are assumed to be executed correctly, while our simulated user follows a more stochastic and demanding behavioral policy, including randomized information disclosure, request changes, and patience. Absolute scores and relative domain difficulty should therefore be interpreted under this protocol rather than directly compared with previously reported cross-domain rankings.

4.3 Multi-Round Alternating Training

Improvement across rounds. Figure 3(a) shows a steady improvement over successive rounds of alternating training. On Qwen3.6-27B Retail, task success increases from 25.32% for the base agent to 34.21%, 40.38%, 44.85%, and 47.84% after Rounds 1–4. From Round 1 to Round 4, this corresponds to a further gain of 13.63 percentage points, showing that the benefit of ATOA extends beyond the initial training round.

Effect of opponent adaptation. Figure 3(b) compares different continuations from the same agent checkpoints. After Round 2, training with the adapted user reaches 44.85%, compared with 42.92% when continuing with the previous user, corresponding to a 1.93-percentage-point improvement. After Round 3, the retained adapted user reaches 47.84%, compared with 47.06% using the previous user and 46.97% when reusing earlier data. These comparisons indicate that the gains from multi-

Table 1: Task success (%) across Retail, Airline, and Telecom under a unified scripted-user evaluation. Results are reported as mean ± evaluation uncertainty.

Method	Qwen3.5-9B			Qwen3.6-27B		
	Retail	Airline	Telecom	Retail	Airline	Telecom
Reference baselines						
Base	15.34 ±1.18	3.58 ±0.31	53.93 ±1.67	25.32 ±1.73	6.43 ±0.62	72.26 ±1.21
Fixed-user training	24.01 ±1.42	3.57 ±0.36	60.91 ±1.54	30.43 ±2.06	6.91 ±0.57	67.55 ±1.46
User simulation						
Persona Policies	22.28 ±1.27	3.29 ±0.28	60.36 ±1.83	30.24 ±1.89	6.79 ±0.74	73.09 ±1.62
RealUserSim	26.42 ±1.56	4.64 ±0.43	66.43 ±1.29	34.80 ±2.31	7.50 ±0.81	68.21 ±1.34
UserLM-R1	25.71 ±1.33	3.93 ±0.39	64.64 ±1.48	33.95 ±2.14	8.57 ±0.69	70.71 ±1.18
Adaptive training						
Calibrated Interactive RL	27.98 ±1.61	5.02 ±0.46	66.79 ±1.37	36.65 ±2.47	6.93 ±0.66	70.36 ±1.41
SENTINEL	24.63 ±1.49	3.07 ±0.24	61.73 ±1.72	31.19 ±2.23	6.14 ±0.53	67.50 ±1.57
SEAD	29.40 ±1.38	5.36 ±0.57	69.64 ±1.22	40.20 ±2.09	9.29 ±0.89	73.57 ±1.09
Agent self-improvement						
EigenData	28.69 ±1.24	5.71 ±0.49	68.57 ±1.45	37.64 ±2.36	7.86 ±0.77	70.33 ±1.28
Divergence-Point DPO	32.97 ±1.31	**6.43 ±0.42**	70.71 ±1.58	43.20 ±1.97	**10.04 ±0.86**	81.11 ±0.83
Bilateral training						
IB-RL-inspired (COOP)	31.96 ±1.67	5.79 ±0.51	67.93 ±1.31	41.47 ±2.58	9.36 ±0.96	74.93 ±1.16
ATOA	**34.94 ±1.09**	**6.43 ±0.58**	**73.57 ±1.36**	**47.84 ±2.82**	9.74 ±2.77	**85.76 ±0.94**

round training are not explained solely by additional optimization, and that refreshing the training opponent can provide additional learning signal as the agent improves.

4.4 Effectiveness of the User Simulator

Script-based initialization. Figure 3(c) examines how script-based SFT changes the interaction environment while keeping the agent and fixed behavioral controller unchanged. Compared with the user using an untrained language renderer, the user with a script-trained renderer reduces probe-agent success from 35.20% to 22.60%, indicating more challenging interactions. At the same time, the proportion of mixed-outcome tasks increases from 20.18% to 28.95%. Thus, script-based SFT does not simply turn more tasks into failures; it shifts more interactions into a regime where the agent succeeds in some attempts but fails in others. These mixed-outcome tasks provide both verified successes and failures on the same task, creating useful supervision for subsequent agent training.

Together with Figure 3(b), these results highlight the complementary roles of the two user-side training stages. Script-based initialization establishes a controlled but challenging interaction distribution, while failure-guided opponent adaptation adjusts the training opponent to the weaknesses of the current agent. Throughout both stages, the fixed behavioral controller preserves the underlying user decision rules, while the user language model changes only how those decisions are expressed.

4.5 Performance with Cooperative Users

Retention under cooperative interaction. ATOA is trained with a scripted user designed to provide more challenging and variable interactions. We therefore examine whether this training compromises the agent's ability to interact with the standard cooperative GPT-4.1 user in τ^2-bench. As shown in Figure 3(d), the resulting agents retain substantial performance under this different user policy. In Retail and Airline, ATOA shows only moderate decreases relative to the corresponding base agent, whereas in Telecom it improves task success by 27.50 percentage points.

These results indicate that training against a more demanding scripted user does not simply specialize the agent to that interaction style. The learned agent remains effective with a cooperative user, and can even transfer substantial gains in some scenarios. This suggests that ATOA can improve

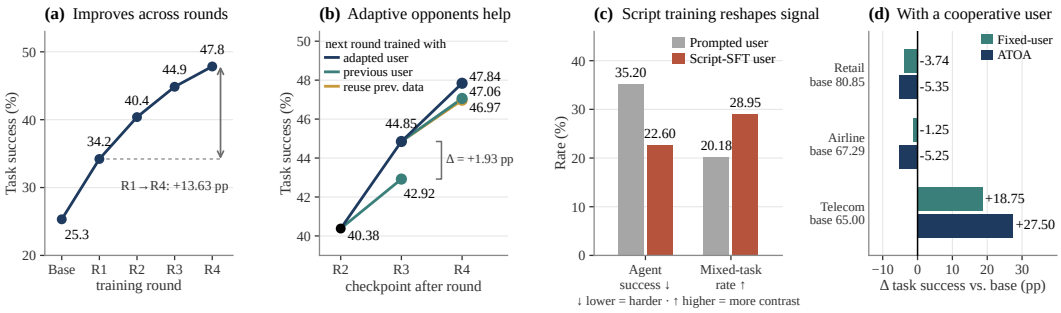


Figure 3: **Training dynamics and opponent adaptation on Qwen3.6-27B.** (a) Performance improves across ATOA rounds. (b) Adapted opponents outperform continued training with previous users or reused data. (c) Script-based user training increases difficulty and mixed-outcome tasks. (d) Cooperative-user evaluation shows domain-dependent transfer.

robustness to harder user behaviors without sacrificing the agent's basic ability to complete tasks with more standard cooperative users.

4.6 Ablation Study

Alternation and opponent adaptation. Removing alternation causes the largest degradation, with an average drop of 5.30 points. Keeping the script-initialized opponent fixed during training also reduces performance by 4.91 points. Random Update replaces failure-guided examples with random user utterances, while No Mixed Filter removes the restriction to mixed-outcome tasks; both consistently underperform full ATOA. These results show that the gain comes not only from repeatedly training the user, but from adapting it to the current agent before the next agent update, using failures from the current round.

Table 2: ATOA ablations on Qwen3.6-27B. Gray values denote drops from the full model.

Variant	Retail	Airline	Telecom
No Alternation	41.61 (-6.23)	7.38 (-2.36)	78.44 (-7.32)
Fixed Opponent	42.53 (-5.31)	8.16 (-1.58)	77.91 (-7.85)
Random Update	42.81 (-5.03)	8.02 (-1.72)	79.14 (-6.62)
No Mixed Filter	44.36 (-3.48)	8.10 (-1.64)	77.83 (-7.93)
No Script Init.	43.22 (-4.62)	7.71 (-2.03)	79.62 (-6.14)
No Script Retention	43.02 (-4.82)	8.18 (-1.56)	78.41 (-7.35)
Accept All Updates	45.54 (-2.30)	8.97 (-0.77)	80.22 (-5.54)
RAFT Only	42.68 (-5.16)	8.12 (-1.62)	78.73 (-7.03)
DPO Only	44.91 (-2.93)	8.95 (-0.79)	79.88 (-5.88)
Full ATOA	**47.84**	**9.74**	**85.76**

User-side design. No Script Init. skips the initial script-based SFT, while No Script Retention removes the original script examples from later opponent updates. Both cause consistent drops across all three scenarios, showing that user adaptation benefits from both a controlled initialization and continued exposure to the scripted behaviors. Accept All Updates replaces the retained opponent with every newly trained candidate without checking its usefulness; its smaller but consistent degradation indicates that rejecting unhelpful user updates improves multi-round stability.

Agent-side learning. RAFT Only removes trajectory-level DPO, whereas DPO Only removes RAFT on verified successful trajectories. Both underperform full ATOA, confirming their complementary effects. The larger 4.60-point average drop for RAFT Only shows that DPO adds substantial signal beyond successful demonstrations, while the 3.20-point drop for DPO Only confirms that verified successes remain important for anchoring agent learning.

5 Conclusion

In this work, we introduced ATOA, a multi-round adversarial training framework for conversational agents. ATOA alternates failure-guided user adaptation with agent training, allowing training challenges to evolve across rounds while keeping user decision rules and task verification fixed. Across Retail, Airline, and Telecom in τ^2-bench and two model scales, ATOA consistently improves agent task success, with ablations confirming the contributions of alternation and opponent adaptation.

AI USE STATEMENT

We used generative AI tools primarily for language polishing, grammar correction, and improving the clarity and consistency of the manuscript. The use of language models as agents and user simulators is described in Section 3. The authors reviewed the retained AI-assisted content and made the final methodological and editorial decisions. We take full responsibility for the manuscript, including its scientific claims, experimental results, and conclusions.

ETHICS STATEMENT

Our experiments use publicly available benchmark tasks from τ^2-bench and simulated interactions. We do not recruit human participants or collect new personal data, and all experimental interactions take place in simulated task environments. ATOA is intended to improve conversational agents through controlled adversarial training. Simulated users may not capture the full diversity of human behavior, and deployment with real users would require additional assessment of fairness, privacy, and safety.

REPRODUCIBILITY STATEMENT

We provide the key implementation code and configuration files in the Supplementary Material. Section 3 describes the ATOA framework, while Appendix A provides the task splits, training configurations, evaluation protocols, and statistical procedures. Appendices B–C detail the user simulator and multi-round training procedure, and Appendices D–F provide the baseline implementations, controlled experiments, additional results, and computational-cost analysis.

REFERENCES

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

Zhiyong Cao, Dunqiang Liu, Qi Dai, Haojun Xu, Huai Yuen Khor, Hao Wang, Huan He, Yafei Liu, Ke Ma, Ruqian Shi, Sicheng Zhou, and Sijia Yao. Simrpd: Optimizing recruitment proactive dialogue agents through simulator-based data evaluation and selection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 1359–1377, 2026.

Harshita Chopra, Kshitish Ghate, Aylin Caliskan, Tadayoshi Kohno, Chirag Shah, and Natasha Jaques. Beyond cooperative simulators: Generating realistic user personas for robust evaluation of llm agents. *arXiv preprint arXiv:2605.12894*, 2026.

Yuqin Dai, Ning Gao, Wei Zhang, Jie Wang, Ruiyuan Wu, Jinpeng Wang, Chaozheng Wang, et al. Sead: Self-evolving agent for multi-turn service dialogue. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3674–3684, 2026.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: reward ranked finetuning for generative foundation model alignment. *Trans. Mach. Learn. Res.*, 2023, 2023.

Yao Dou, Michel Galley, Baolin Peng, Chris Kedzie, Weixin Cai, Alan Ritter, Chris Quirk, Wei Xu, and Jianfeng Gao. Simulatorarena: Are user simulators reliable proxies for multi-turn evaluation of AI assistants? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 35212–35290, 2025.

Jiaxuan Gao, Jiaao Chen, Chuyi He, Shusheng Xu, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents. *arXiv preprint arXiv:2601.22607*, 2026a.

Ning Gao, Wei Zhang, Yuqin Dai, Ling Shi, Ziyin Wang, Yujie Wang, Wei He, Jinpeng Wang, and Chaozheng Wang. Reinforcing real-world service agents: Balancing utility and cost in task-oriented dialogue. *arXiv preprint arXiv:2602.22697*, 2026b.

Hengrui Gu, Xiaotian Han, and Kaixiong Zhou. Writ: Write-read intensive trajectory synthesis for multi-turn user-facing agents. *arXiv preprint arXiv:2606.02908*, 2026.

Bowen Jiang, Taiwei Shi, Ryo Kamoi, Yuan Yuan, Camillo J Taylor, Longqi Yang, Pei Zhou, and Sihao Chen. One model, all roles: Multi-turn, multi-agent self-play reinforcement learning for conversational social intelligence. *arXiv preprint arXiv:2602.03109*, 2026.

Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 2443–2453. Association for Computational Linguistics, 2017.

Bing Liu and Ian R. Lane. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017*, pp. 482–489. IEEE, 2017.

Shashank Mangla, Chris Hokamp, Jack Boylan, Demian Gholipour Ghalandari, Yuuv Jauhari, Lauren Cassidy, and Oisin Duffy. Negotiationgym: Self-optimizing agents in a multi-agent social simulation environment. *arXiv preprint arXiv:2510.04368*, 2025.

Shuhaib Mehri, Xiaocheng Yang, Takyoung Kim, Gokhan Tur, Shikib Mehri, and Dilek Hakkani-Tür. Goal alignment in llm-based user simulators for conversational AI. *Trans. Assoc. Comput. Linguistics*, 14:852–876, 2026.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. In *Proceedings of International Conference on Learning Representations*, volume 2026, pp. 82079–82103, 2026.

Kun Peng, Conghui Tan, Yu Liu, Guohua Tang, Zhongqian Sun, Wei Yang, Zining Zhu, Lei Jiang, Yanbing Liu, and Hao Peng. Dialogue model optimization via agent game and adaptive tree-based grpo. *arXiv preprint arXiv:2602.08533*, 2026.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In *Proceedings of Advances in Neural Information Processing Systems*, 2025.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, et al. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.

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

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

Jeonghoon Shim, Woojung Song, Cheyon Jin, Seungwon KooK, and Yohan Jo. Non-collaborative user simulators for tool agents. In *Proceedings of International Conference on Learning Representations*, volume 2026, pp. 64486–64534, 2026.

Joseph Suh, Ayush Raj, Minwoo Kang, and Serina Chang. Quantifying the utility of user simulators for building collaborative llm assistants. *arXiv preprint arXiv:2605.09808*, 2026.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pp. 625–638. Association for Computational Linguistics, 2020.

Jiaqiang Tang. Self-evolution for multi-turn tool-calling agents via divergence-point preference learning. *arXiv preprint arXiv:2606.23112*, 2026.

Guy Tennenholtz, Ofer Meshi, Amir Globerson, Uri Shalit, Jihwan Jeong, and Craig Boutilier. Controllable user simulation. *arXiv preprint arXiv:2605.11519*, 2026.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. Transferable dialogue systems and user simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pp. 152–166. Association for Computational Linguistics, 2021.

Bo Wang, Ruixing Zhang, Yunqi Liu, Yang Zhang, Liangzhe Han, Tongyu Zhu, and Leilei Sun. Intent speaks louder: Controllable user simulation beyond response imitation. *arXiv preprint arXiv:2608.09420*, 2026a.

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. Know you first and be you better: Modeling human-like user simulators via implicit profiles. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21082–21107, 2025a.

Senhao Wang, Chenghao Cai, Haitao Hu, Mingxing Huang, Xingguang Wang, Wenhao Li, and Zecheng Lin. Ib-rl: Isolated bilateral reinforcement learning for strategic dialogue agents. *arXiv preprint arXiv:2608.06735*, 2026b.

Xiaohua Wang, Jiakang Yuan, Zisu Huang, Muzhao Tian, Changze Lv, Kaitao Song, Tao Chen, and Xiaoqing Zheng. From static context to calibrated interactive rl: Mitigating distribution shift in multi-turn dialogue with aligned simulator. *arXiv preprint arXiv:2605.26403*, 2026c.

Yingshan Susan Wang, Cedegao E Zhang, Linlu Qiu, Zexue He, Pengyuan Li, Alex Pentland, Roger P Levy, and Yoon Kim. Learning user simulators with turing rewards. *arXiv preprint arXiv:2606.19336*, 2026d.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Jing Huang, and Dakuo Wang. Customer-r1: Personalized simulation of human behaviors via rl-based llm agent in online shopping. *arXiv preprint arXiv:2510.07230*, 2025b.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Qun Liu, Chen Luo, Jiri Gesi, Hanqing Lu, Yisi Sang, Manling Li, Jing Huang, et al. Sentinel: Failure-driven reinforcement learning for training tool-using language model agents. *arXiv preprint arXiv:2606.12908*, 2026e.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *Proceedings of Forty-second International Conference on Machine Learning, ICML*, volume 267, 2025.

Shirley Wu, Evelyn Choi, Arpandeep Khatua, Zhanghan Wang, Joy He-Yueya, Tharindu Cyril Weerasooriya, Wei Wei, Diyi Yang, Jure Leskovec, and James Zou. Humanlm: Simulating users with state alignment beats response imitation. *arXiv preprint arXiv:2603.03303*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of The Eleventh International Conference on Learning Representations*, 2023.

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

Feng Zhang, Shijia Li, Chunmao Zhang, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, Jingwen Xu, and Han Liu. Userlm-r1: Modeling human reasoning in user language models with multi-reward reinforcement learning. *arXiv preprint arXiv:2601.09215*, 2026a.

Yimeng Zhang, Tian Wang, Jiri Gesi, Ziyi Wang, Yuxuan Lu, Jiacheng Lin, Simon Zhan, Vianne Gao, Ruochen Jiao, Junze Liu, et al. Shop-r1: Rewarding llms to simulate human behavior in online shopping via reinforcement learning. In *Proceedings of International Conference on Learning Representations*, volume 2026, pp. 8406–8423, 2026b.

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

Ming Zhu, Juntao Tan, Rithesh Murthy, Jielin Qiu, Liangwei Yang, Wenting Zhao, Silvio Savarese, Shelby Heinecke, and Huan Wang. Realusersim: Bridging the reality gap in agent benchmarking via grounded user simulation. *arXiv preprint arXiv:2605.20204*, 2026.

A Experimental Setup and Reproducibility

This appendix provides the task construction, data partitions, model configurations, interaction protocols, and statistical procedures used in our experiments. We distinguish the task pools used for training and evaluation, as well as the training users and evaluation users used under different experimental settings. The training procedure and notation follow Section 3.

A.1 Tasks and Data Splits

Benchmark and task construction. We use τ^2-bench v1.0.1 with the Retail, Airline, and Telecom environments (Barres et al., 2025). Each executable task instance, referred to as a *task card* in our implementation, is deterministically compiled from a source task. A card specifies the initial request, information known and withheld by the user, persona information that affects language expression, behavioral decision parameters, and a fixed random seed. Behavioral decisions are controlled independently of language generation.

Before entering the experimental pool, each card passes a dual-probe check: a reference agent following the task solution must succeed, whereas an agent that performs no task-solving action must fail. This removes cases that cannot be mechanically distinguished under the benchmark environment. Task outcomes are determined without an LLM judge: Retail and Airline use database-state checks, while Telecom uses environment assertions.

Repeated interactions with the same card are treated as repeated trials rather than distinct tasks. Unless otherwise stated, each card is evaluated with eight trials, indexed from 0 to 7.

Table 3: Task pools and evaluation splits used in the three scenarios. All evaluation cards are disjoint from the corresponding training collection, and each main evaluation card is repeated for eight trials.

Setting	Retail	Airline	Telecom
Source pool	114 benchmark tasks plus 200 constructed task cards	50 benchmark tasks plus 400 template-resampled task cards	2,285 programmatically generated tasks
Training collection	114 task cards	44 task cards	106 task cards
Evaluation set	31 held-out task cards	35 newly sampled task cards	100 newly sampled task cards
Trials per evaluation card	8	8	8

Domain-specific task pools. Retail starts from 114 τ^2 benchmark tasks and 200 additional task cards constructed from existing training tasks and database templates. We use 114 cards for training and reserve 31 cards for held-out evaluation. Airline starts from 50 benchmark tasks together with 400 template-resampled cards; 44 cards are used for training and 35 newly sampled cards for evaluation. Telecom draws from 2,285 programmatically generated tasks, with 106 cards used for training and 100 newly sampled cards for evaluation.

Evaluation cards are disjoint from the corresponding training collections. We verify this separation using card IDs and source task IDs, with additional reference-solution checks where available. Repeated trials of the same card remain within the same partition and are never split across training and evaluation.

Candidate selection. User candidate selection is performed on training-side interactions before the held-out evaluation sets are accessed. A new user is evaluated against the same fixed agent with repeated interactions. It replaces the retained opponent only when it increases the number of mixed-outcome cards by at least $\lceil 0.05|\mathcal{X}| \rceil$, preserves the available success–failure pairing potential, and does not make the interaction easier according to the candidate criterion described in Appendix C.4. Otherwise, the previous user is retained. Section C provides the complete selection and stopping procedure.

Table 4: Script-based SFT data used to initialize the user language model. Counts denote utterance-level training examples.

Scenario	Training	Validation	Teacher model
Retail	2,735	302	`gpt-5.5`
Airline	4,080	451	`gpt-5.5`
Telecom	6,047	657	`gpt-5.5`

Table 5: Training configurations shared by Qwen3.5-9B and Qwen3.6-27B. The same stage-specific hyperparameters are used at both model scales.

Setting	User SFT	Agent RAFT	Agent DPO
Starting checkpoint	ϕ_{base}	θ_{k-1}	$\bar{\theta}_k$
LoRA rank / scaling	32 / 64	32 / 64	32 / 64
LoRA dropout	0.05	0.05	0.05
Learning rate	1×10^{-4}	5×10^{-5}	5×10^{-6}
Epochs	2	1	1
Gradient accumulation	16	16	8
Maximum sequence length	4,096	8,192	—
DPO prompt / response limit	—	—	12,288 / 1,536
Preference parameter β	—	—	0.1
Reference policy	—	—	Frozen post-RAFT agent

User-language-model data. The script-based SFT corpus contains context–directive–utterance examples and is split into training and validation portions by card. The teacher model generates the target language expression but does not judge task success. Table 4 summarizes the initialization corpora.

As an example of a later opponent update, the documented Retail update augments the 2,735 script examples with 165 user utterances extracted from failed interactions, yielding 2,900 user-SFT examples. Failure-derived examples are constructed independently in each round rather than accumulated across all previous rounds.

A.2 Models and Training Configurations

Model scales. We evaluate Qwen3.5-9B and Qwen3.6-27B as the two model scales. For the Qwen3.6-27B experiments, both the agent and user language model use Qwen3.6-27B. Qwen3.5-9B experiments use the same stage-specific optimization hyperparameters as the Qwen3.6-27B experiments.

The behavioral controller, script distribution, directive checks, and task verifier are fixed components and are not updated by gradient-based training.

Checkpoint flow. User initialization applies SFT to $\mathcal{D}_{\text{script}}$ starting from the pretrained checkpoint ϕ_{base}. Each later opponent candidate is also trained from ϕ_{base} using $\mathcal{D}_{\text{script}} \uplus \mathcal{D}^{U}_{k-1}$, following Eq. 5. Failure datasets from earlier rounds are not accumulated, and a rejected candidate does not replace the retained user.

The agent continues from its previous checkpoint θ_{k-1}. RAFT first learns from verified successful trajectories and produces the intermediate checkpoint $\bar{\theta}_k$. Trajectory-level DPO then starts from $\bar{\theta}_k$ and uses a frozen copy of the post-RAFT model as the reference policy.

SFT and RAFT optimization. User SFT and RAFT use a microbatch size of one with gradient accumulation over 16 examples, giving an effective batch size of 16. Both stages use LoRA with rank 32, scaling factor 64, dropout 0.05, and all linear layers as targets. Optimization uses AdamW with zero weight decay, gradient clipping at 1.0, a cosine learning-rate schedule, and 3% warmup. Training is performed in bf16 with thinking disabled and seed 20260814.

For user SFT, the loss is applied only to the target user utterance. For RAFT, successful trajectories are converted into agent-action examples, and only the target agent response or tool call contributes to the loss. Previous messages, user responses, and tool outputs remain part of the context.

Trajectory-level DPO. A preference pair consists of a successful and failed attempt on the same task card. We restrict the difference in agent-turn counts to at most four, retain at most 40 pairs per card, and discard complete pairs exceeding 24 agent turns or the supported sequence lengths. Training and validation preference pairs are separated by card.

DPO uses one epoch, learning rate 5×10^{-6}, gradient accumulation eight, and $\beta = 0.1$. The trajectory score is the sum of agent-token log probabilities and is not normalized by sequence length. A new LoRA adapter with rank 32, scaling factor 64, and dropout 0.05 is trained from the post-RAFT checkpoint, while a frozen copy of the post-RAFT agent is used as the reference policy.

Recorded training-data volumes. The standard Qwen3.6-27B Retail Round-4 agent uses 2,709 RAFT examples and 109 DPO pairs, corresponding to 14 DPO optimizer updates. The first-round Airline agent uses 733 RAFT examples and 13 DPO pairs, corresponding to two updates. The first-round Telecom run uses 2,700 RAFT examples sampled from 12,103 candidate rows originating from 612 successful interactions, together with 108 DPO pairs and 14 optimizer updates.

These counts reflect the usable training data produced by each interaction collection rather than a fixed target imposed across domains. Matching the number of collected interactions therefore does not necessarily match the number of successful RAFT examples, preference pairs, agent tokens, or optimizer updates.

Hardware and software. Experiments are run on a pool of four NVIDIA H200 GPUs with approximately 140 GB memory per GPU. Each training job and each inference service uses one GPU. Training uses bf16 arithmetic. The software stack consists of vLLM 0.18.0, Transformers 5.3, PyTorch 2.10, and PEFT 0.19.1. Trajectory-level DPO uses our own trainer rather than TRL.

Inference uses `max-num-seqs=1` and disables prefix caching. This single-sequence setting avoids batching-dependent forward differences observed with the Qwen3.6 GDN implementation. Tool calls are parsed with the `qwen3_coder` parser.

A.3 Interaction and Evaluation Protocols

Environment orchestration. We build our interaction pipeline on the native τ^2-bench framework and replace only the user simulator with our controllable user model. The remaining environment dynamics, tool execution, and task verification follow the benchmark protocol. Each episode then proceeds as a multi-turn interaction between the agent and the scripted user simulator.

At every user turn, the frozen behavioral controller first determines which information may be revealed and whether the interaction should terminate. The user language model then expresses this directive in natural language. The generated utterance is checked before entering the interaction: required information must appear verbatim, unrevealed information cannot be disclosed prematurely, and the stop marker is permitted if and only if termination has been authorized by the controller. Invalid generations are resampled up to three times, after which a deterministic template fallback is used. The user language model never determines the underlying behavioral decision.

Shared behavioral randomness. Behavioral randomness is fixed at the task-card and trial level. The card seed is obtained from `sha256("20260814:" + card_id)`, and the trial index defines the trial-specific seed. Consequently, different methods evaluated on the same card and trial receive the same sampled behavioral script. Script hashes for common card–trial cells are checked for equality before comparison.

The patience limit is sampled uniformly from $[12, 18]$ in Retail and Airline and from $[14, 22]$ in Telecom. At each user turn, each undisclosed information item has probability 0.25 of being temporarily withheld, and at most two information items may be revealed in a single user turn. Fixed randomness does not imply a fixed conversation: the controller still reacts to the interaction history, so different agent actions can produce different dialogue trajectories under the same script.

Decoding. Both the agent and user language model use temperature 0.7. The agent may generate up to 2,048 tokens per turn and the user language model up to 200 tokens. Agent thinking is dis-

Table 6: Domain-specific interaction settings for scripted-user evaluation.

Setting	Retail	Airline	Telecom
Patience range	$[12, 18]$	$[12, 18]$	$[14, 22]$
Maximum controller turns	40	40	80
Agent context length	16,384	16,384	32,768
User-model context length	8,192	8,192	8,192
Main-evaluation trials per card	8	8	8

abled. Agent requests use the trial index as the request seed. The benchmark additionally enforces a maximum of 200 orchestration steps, ten consecutive tool errors, and 1,800 seconds per episode.

Termination. The controller emits `###STOP###` when the user patience budget is exhausted, or after the task is resolved, the agent closes the interaction, and no required information remains undisclosed. Retail and Airline allow at most 40 controller turns, whereas Telecom allows 80 turns including tool turns.

Telecom additionally supports user-side tool operations through 15 read tools and 15 write tools. These operations are selected by the fixed controller rather than the user language model. Required user-side operations are assumed to execute correctly. After each write operation, the environment checks whether the issue has been resolved; four consecutive unresolved write operations cause the user to abandon the interaction. If the agent transfers the user to a human operator, the simulator emits `###TRANSFER###`.

Training and evaluation users. The main evaluation in Table 1 uses the same frozen behavioral controller and script-trained language renderer for every agent. We refer to this condition as the scripted-user evaluation. Holding the evaluation user fixed prevents changes in the user model from being conflated with changes in agent performance.

Fixed-user training differs only in the training counterpart: it uses the same fixed behavioral controller but an untrained language renderer, which remains unchanged throughout agent training. This setting is distinct from the *Fixed Opponent* ablation in Table 2, which first applies script-based SFT and then prevents subsequent opponent adaptation.

The external-renderer evaluation keeps the same behavioral controller and task cards while replacing only language generation with GPT-5.5. The cooperative-user evaluation, reported in Figure 3(d), instead uses the official `gpt-4.1-2025-04-14` user configuration from τ^2-bench. It contains 67 Retail tasks with six trials each, 27 Airline tasks with four trials each, and 40 Telecom tasks with two trials each.

Task verification. The official τ^2 mechanical evaluator supplies the environment reward. An eligible interaction is counted as successful when the environment reward exceeds 0.99 and the episode reaches a normal protocol termination. Interactions that end because of a hard limit or other abnormal termination are not counted as successful completions.

Infrastructure failures are retried with the same seed up to three times. Attempts that remain invalid are removed from the primary denominator. Turn-limit and context-limit cases are tracked separately; we additionally report a sensitivity analysis in which capped attempts are counted as failures when the exclusion-rate difference between two methods exceeds three percentage points.

A.4 Statistical Analysis

Task success. A basic evaluation unit is a card–trial pair. Let $y_{m,x,r} \in \{0, 1\}$ denote the outcome of method m on card x and trial r, and let $\mathcal{I}_{m,x}$ contain the valid trial indices for that card. The reported task-success rate pools all valid card–trial cells:

$$\widehat{S}_m = 100 \frac{\sum_x \sum_{r \in \mathcal{I}_{m,x}} y_{m,x,r}}{\sum_x |\mathcal{I}_{m,x}|}. \tag{9}$$

Thus, when all eight trials are valid, each card contributes eight binary outcomes. If an attempt is excluded because of an infrastructure failure, the denominator contains only the remaining valid cells.

Evaluation uncertainty. We estimate evaluation uncertainty using a whole-card bootstrap. In each bootstrap replicate, we sample the observed number of task cards with replacement and retain all repeated interactions associated with each sampled card. We then recompute Eq. 9. We use 10,000 bootstrap replicates with seed 20260814.

Let $\widehat{S}_m^{*(b)}$ denote the task-success estimate in bootstrap replicate b, and let $\overline{S}_m^* = B^{-1}\sum_{b=1}^{B}\widehat{S}_m^{*(b)}$. The bootstrap standard error is

$$\widehat{\mathrm{SE}}_{\mathrm{boot}}(\widehat{S}_m) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\widehat{S}_m^{*(b)} - \overline{S}_m^*\right)^2}, \tag{10}$$

where $B = 10{,}000$. The $\pm$ values reported in Table 1 are these bootstrap standard errors, expressed in percentage points.

This uncertainty characterizes evaluation variability for a fixed trained checkpoint. It does not measure variability across independent training runs, since each reported method is trained once.

B Controllable User Simulator

This appendix provides additional implementation details for the controllable user simulator introduced in Section 3. The key design principle is to separate user decisions from their natural-language realization. A behavioral script specifies the allowed behavior, a fixed controller converts the script and interaction history into a turn-level directive, and a trainable language model expresses that directive in natural language.

B.1 Behavioral Scripts

Each task is associated with a behavioral script ξ that specifies how the simulated user may behave during the interaction. The script contains the opening request, information that may or may not be disclosed, possible request changes, patience-related parameters, persona information, and a random seed. Telecom additionally includes the user-side device operations that may be executed during the task. Table 7 summarizes the main fields.

Table 7: Main fields of a behavioral script. Behavioral fields determine user decisions, while persona information affects only language realization.

Field	Description
Opening request	Initial task request presented at the beginning of the interaction.
Information units	Facts that the user knows, does not know, or may reveal only after the agent requests them.
Request changes	Task-consistent changes in the user's request that may become active during the interaction.
Patience	Parameters controlling how long the user continues when sufficient progress is not made.
Persona	Language-level attributes that affect wording and tone but not the underlying behavioral decisions.
User-side actions	Telecom-only device operations that the controller may execute when appropriately requested by the agent.
Seed	Random seed used to instantiate stochastic behavioral choices for a particular trial.

Behavioral randomness is instantiated at the task-card and trial level. In our implementation, the patience budget is sampled from $[12, 18]$ for Retail and Airline and from $[14, 22]$ for Telecom. A requested information unit is temporarily withheld with probability 0.25, and at most two information units may be disclosed in one user turn. These choices are shared across methods evaluated on the same card and trial, as described in Appendix A.3.

Importantly, the script fixes the behavioral constraints but not the complete dialogue trajectory. The controller responds to the evolving interaction history, so agents that ask different questions or act at different times may receive different user responses under the same underlying script.

B.2 Fixed Behavioral Controller

At user turn j, the fixed controller computes

$$d_j = C(x, \xi, h_j), \tag{11}$$

where x is the task, ξ is the sampled behavioral script, and h_j is the interaction history. The directive d_j specifies what the user should do at that turn, such as stating the opening request, revealing permitted information, withholding unavailable information, expressing a request change, performing a user-side action, or terminating the dialogue.

For information disclosure, a fact becomes eligible only after the agent requests the corresponding information. If several facts are eligible, the controller follows the per-turn disclosure budget and keeps the remaining facts pending for later turns. Information that has not yet become eligible cannot be revealed. The controller similarly tracks patience and determines whether the interaction should continue or terminate.

Telecom additionally includes user-side device actions. These actions are selected by the controller rather than the language model. When the agent provides an appropriate instruction, the controller may execute the corresponding device operation and expose its result to the subsequent interaction.

The controller itself is not learned. Its rules, behavioral parameters, and task constraints remain fixed throughout user and agent training. Thus, opponent adaptation changes how the user expresses a directive without changing the rules used to generate that directive.

B.3 Trainable User Language Model

The user language model receives the interaction context together with the controller directive and generates the natural-language user response:

$$u_j \sim \mu_\phi(\cdot \mid h_j, d_j). \tag{12}$$

The model learns how to realize the directive naturally, including wording, tone, and persona-consistent expression. It does not determine which information becomes available, when a request changes, how patience evolves, whether the interaction terminates, or which user-side action is executed. These decisions remain under the fixed controller.

Script-based SFT initializes this language model from context–directive–utterance examples. During later opponent adaptation, failure-derived examples provide additional supervision for how the same controller-defined behaviors are expressed against the current agent. In both stages, the controller and behavioral scripts remain unchanged. The construction of these training examples and the complete alternating update procedure are described in Appendix C.

B.4 Directive Enforcement

Before a generated user utterance enters the interaction, we mechanically check whether it follows the controller directive. The checks cover three main conditions:

- **Required information.** Information specified by the directive must be present in the generated response.
- **Premature disclosure.** Information that has not yet been released by the controller must not be exposed in advance.
- **Termination.** Stop or transfer markers may appear only when the corresponding controller decision has been triggered.

If an utterance violates these constraints, it is regenerated up to three times. If all attempts fail, the simulator uses a deterministic template rendering of the directive. User-side tool actions in Telecom are handled directly by the controller and therefore do not depend on language-model generation.

These checks ensure that changes in the user language model affect only how the predefined behavior is expressed. Task completion remains independently determined by the environment-based verifier described in Appendix A.3.

B.5 Illustrative Examples

The following examples illustrate how the behavioral script, controller, and language model interact. They are simplified for presentation; the controller logic and directive checks remain the same as in the experiments.

Example 1 (Controlled information disclosure)

Suppose the script specifies that the user knows their name and phone number, but may reveal them only when requested.

Agent: Could you describe the problem you are having?

Controller directive: *Restate the problem. Do not reveal identity information.*

User: I have been unable to send MMS messages for the past few hours.

The identity information remains hidden even though it is available in the script. If the agent later asks for it:

Agent: Could I have your name and phone number?

Controller directive: *Reveal the name and phone number.*

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

The controller determines when the information becomes available, while the language model determines how it is expressed.

Example 2 (Same decision, different language realization)

Consider a directive that requires the user to report that they are currently traveling in France.

Controller directive: *Reveal that the user is currently abroad in France.*

User: **Realization 1:** Yes, I am currently in France.

User: **Realization 2:** I'm traveling in France at the moment, if that matters.

Both responses satisfy the same directive. The language model may vary wording and tone, but it cannot change the underlying fact or introduce information that has not been released by the controller.

Example 3 (Controller-managed user-side action)

In Telecom, some tasks require the user to perform an operation on their own device.

Agent: Please restart your phone and let me know whether the issue is still present.

Controller directive: *Execute the permitted device action* `reboot_device`.

The controller executes the corresponding user-side action in the environment. After receiving the result, it issues a new directive specifying what may be reported to the agent, and the language model only verbalizes that result.

If the requested operation is not permitted by the script, the controller does not execute it and instead produces a directive to decline the request. The language model therefore cannot independently choose or execute user-side actions.

C Detailed Alternating Training Procedure

This appendix describes the complete training loop of ATOA. The key idea is simple: the current user is used to expose failures of the agent, these interactions are used to train the agent, and failures of the improved agent are then used to construct the next user update. The behavioral controller and task verifier remain fixed throughout the process.

C.1 Overall Training Loop

ATOA begins with a script-initialized user U_{ϕ_0} obtained from $\mathcal{D}_{\text{script}}$. We first collect repeated interactions between the base agent π_{θ_0} and U_{ϕ_0} to obtain the failures used for the first opponent update.

At round k, failures of the current agent $\pi_{\theta_{k-1}}$ are used to construct the next user-training dataset. A candidate user is trained from ϕ_{base} and compared with the retained user against the same fixed agent. The selected user U_{ϕ_k} is then frozen and used to collect new interactions for agent training. RAFT on verified successful trajectories, followed by trajectory-level DPO on same-task successful–failed pairs, produces the updated agent π_{θ_k}. Its failures provide the signal for the following opponent update, closing the alternating loop.

Algorithm 1 Alternating Training with Opponent Adaptation

Require: Training tasks $\mathcal{X}_{\text{train}}$, base agent π_{θ_0}, base user model ϕ_{base}, script data $\mathcal{D}_{\text{script}}$

Train the initial user U_{ϕ_0} from ϕ_{base} on $\mathcal{D}_{\text{script}}$
Collect repeated interactions $\mathcal{R}_0$ between π_{θ_0} and U_{ϕ_0}
for $k = 1, 2, \ldots$ **do**
  Select mixed-outcome tasks from $\mathcal{R}_{k-1}$ and extract user utterances from their failed attempts to form $\mathcal{D}^U_{k-1}$
  Train candidate user $\widetilde{U}_{\phi_k}$ from ϕ_{base} on $\mathcal{D}_{\text{script}} \uplus \mathcal{D}^U_{k-1}$
  Compare $\widetilde{U}_{\phi_k}$ with the retained user $U_{\phi_{k-1}}$ against the same fixed agent $\pi_{\theta_{k-1}}$
  if the candidate satisfies the selection criterion **then**
    $U_{\phi_k} \leftarrow \widetilde{U}_{\phi_k}$
  else
    $U_{\phi_k} \leftarrow U_{\phi_{k-1}}$
  end if
  Freeze U_{ϕ_k} and collect interactions $\mathcal{C}_k$ with the current agent $\pi_{\theta_{k-1}}$
  Apply RAFT to verified successful trajectories in $\mathcal{C}_k$, producing $\pi_{\bar{\theta}_k}$
  Apply trajectory-level DPO to same-task successful–failed pairs in $\mathcal{C}_k$, producing π_{θ_k}
  Collect repeated interactions $\mathcal{R}_k$ between π_{θ_k} and U_{ϕ_k}
  if the stopping criterion is met **then**
    break
  end if
end for
return Final agent π_{θ_k}

C.2 Construction of User Adaptation Data

For each training card, the current agent is evaluated over eight repeated interactions under different trial seeds. Following Eq. 4, we retain tasks on which the agent succeeds in at least one trial and fails in at least one other trial. These mixed-outcome tasks lie near the current agent's capability boundary and provide both positive and negative outcomes under the same task specification.

For each selected task, we take the user turns from failed interactions together with the corresponding interaction context and controller directive. Uninformative or invalid turns are removed before constructing $\mathcal{D}^U_k$. Importantly, task rewards are used only to select the interactions from which these examples are drawn; they are not used as a direct training reward for the user language model.

The user candidate is trained as

$$\widetilde{\phi}_{k+1} = \mathrm{SFT}_U\left(\phi_{\text{base}}; \mathcal{D}_{\text{script}} \uplus \mathcal{D}^U_k\right). \tag{13}$$

Each candidate starts again from ϕ_{base}. Failure-derived examples from earlier rounds are not accumulated. The fixed script data are always retained, providing an anchor for directive-following behavior while the new examples adapt language realization to the latest agent failures.

C.3 Agent Training Data

Agent learning uses the interactions collected from the retained user and contains two complementary stages.

RAFT. All trajectories that satisfy the fixed task verifier are converted into supervised agent examples. Only agent outputs contribute to the training loss; user utterances and tool results remain part of the context. This stage produces the intermediate agent $\pi_{\bar{\theta}_k}$.

Trajectory-level DPO. We then construct preference pairs from successful and failed attempts on the same task card. A pair is retained only when the difference in agent-turn counts is at most four. We keep at most 40 pairs per card and remove pairs whose complete trajectories exceed 24 agent turns or the supported sequence length.

For each trajectory, the DPO score sums the log probabilities of agent tokens only:

$$\ell_\theta(\tau) = \sum_{t \in \mathcal{I}_A(\tau)} \log \pi_\theta(a_t \mid h_t). \tag{14}$$

We do not normalize this score by trajectory length. DPO starts from the post-RAFT checkpoint and uses a frozen copy of the same post-RAFT model as the reference policy. The optimization objective is therefore identical to Eq. 8 in the main text.

C.4 Candidate Selection

A newly trained user candidate does not automatically replace the retained opponent. We compare the candidate and the retained user against the same fixed agent, using the same set of training cards and the same eight trial seeds per card. This keeps the agent checkpoint, tasks, and behavioral randomness fixed, so the comparison reflects only the change in the user language model.

For each card x, let n_x^+ and n_x^- denote the numbers of successful and failed trials. We compute

$$S = \left|\left\{x : n_x^+ > 0,\ n_x^- > 0\right\}\right|, \qquad P = \sum_x \min(n_x^+, n_x^-), \tag{15}$$

where S is the number of mixed-outcome cards and P measures the available success–failure pairing potential. We also compute the agent pass rate $\bar{r}$ under each user.

The candidate replaces the retained user only if

$$\begin{aligned} S_{\text{cand}} &\geq S_{\text{cur}} + \lceil 0.05|\mathcal{X}| \rceil, \\ P_{\text{cand}} &\geq P_{\text{cur}}, \\ \bar{r}_{\text{cand}} - \bar{r}_{\text{cur}} &\leq 0.02, \end{aligned} \tag{16}$$

where $\mathcal{X}$ is the candidate-evaluation card set.

The first condition requires the candidate to expose more tasks on which the current agent succeeds inconsistently. The second ensures that this increase does not reduce the amount of success–failure contrast available for subsequent agent training. The final condition prevents a candidate from being accepted simply because it makes the interaction easier. User abandonment rate and dialogue length are monitored separately but do not enter the acceptance decision.

For example, after the second Retail agent-training round, we evaluate the retained user and the newly trained candidate against the same Round-2 agent on the same 88 training cards. The candidate increases the number of mixed-outcome cards from 23 to 31, exceeding the required increase of five, and increases the pairing potential from 45 to 61. At the same time, agent success decreases from 40.4% under the retained user to 37.8% under the candidate user. The candidate therefore provides more mixed-outcome training signal without making the interaction easier and is retained for the following round.

The pass-rate constraint was added when finalizing the candidate-selection protocol after this Retail update. Re-evaluating the same comparison under the final criterion shows that the accepted candidate also satisfies this constraint.

If a candidate does not satisfy all selection criteria, the previous user is retained for the next agent-training round. Candidate rejection and stopping are treated separately: rejecting one user update does not by itself terminate the alternating procedure.

C.5 Training Rounds and Selected Checkpoints

The number of alternating rounds is not fixed across scenarios. We continue training while successive opponent and agent updates provide meaningful additional improvement, and retain the latest checkpoint before the gains saturate or begin to reverse. Retail is trained for four agent rounds, while Airline and Telecom are trained for three. Table 8 summarizes the progression of the Qwen3.6-27B runs used in our main analyses.

Table 8: Task success (%) across alternating training rounds for Qwen3.6-27B. The bold value denotes the checkpoint retained for the main results. The final non-retained value reports the next-round continuation probe used for stopping.

Scenario	Base	R1	R2	R3	R4	R5	Selected
Retail	25.32	34.21	40.38	44.85	**47.84**	47.6	R4
Airline	6.43	7.86	9.16	**9.74**	8.74	—	R3
Telecom	72.26	79.88	84.42	**85.76**	84.12	—	R3

Retail exhibits sustained improvement over four retained training rounds. Starting from 25.32%, task success increases to 34.21%, 40.38%, 44.85%, and 47.84% over Rounds 1–4, corresponding to successive gains of 8.89, 6.17, 4.47, and 2.99 percentage points. We then perform the next-round continuation probe, which reaches 47.60% and does not improve over the Round-4 checkpoint. We therefore retain Round 4 as the final model, for an overall gain of 22.52 points over the base agent.

Airline shows the same pattern over a shorter horizon. Task success rises from 6.43% to 7.86%, 9.16%, and 9.74% over the first three rounds. The corresponding gains decrease from 1.43 to 1.30 and 0.58 points, indicating that the benefit of further alternation is beginning to saturate. The next update reduces performance to 8.74%, so we stop training and retain the Round-3 checkpoint. This yields an overall improvement of 3.31 points over the base agent.

Telecom improves more rapidly in the early rounds. Performance increases from 72.26% to 79.88%, 84.42%, and 85.76% over Rounds 1–3, with successive gains of 7.62, 4.54, and 1.34 points. The diminishing increments indicate that the agent is approaching saturation under the current training distribution. A further round decreases success to 84.12%, and the Round-3 checkpoint is therefore retained. Relative to the base agent, the selected checkpoint improves task success by 13.50 points.

Across the three scenarios, the same pattern emerges: alternating training provides large gains in the earlier rounds, while the marginal benefit decreases as the agent improves. Importantly, ATOA does not require a fixed number of rounds across scenarios. Retail continues to benefit through four rounds, whereas Airline and Telecom peak after three. Once an additional round no longer improves task success, the previous best checkpoint is retained. This stopping behavior prevents unnecessary continued training after the benefit of opponent adaptation has saturated or begun to reverse.

D Baseline and Ablation Implementations

This appendix describes how the comparison methods and ablation variants are implemented in our experimental framework. Because the original methods were developed with different backbones, simulators, and training environments, we adapt their defining mechanisms to the same task setting rather than directly comparing against their originally reported checkpoints.

Across all methods, we use the same Qwen backbone at the corresponding model scale, the same training-side task pools, and the same environment-based task verification. All final results in Table 1 are evaluated with the same scripted-user protocol described in Appendix A.3. Methods that primarily modify the training user or data distribution use the same downstream agent-training recipe when possible, while methods whose main contribution is a distinct optimization objective retain that objective.

D.1 Baseline Implementations

Table 9 summarizes the implementation of each comparison method. The goal is to preserve the mechanism that distinguishes each baseline while keeping the task environment and final evaluation comparable.

Fairness of the comparison. All adaptations operate on training-side tasks only and use the same environment-based success criterion as ATOA. No method has access to the final evaluation cards during training or method selection. Simulator-focused baselines differ in how the training counterpart is constructed, whereas failure-driven and self-improvement baselines retain their characteristic data generation or optimization mechanism. Regardless of the training counterpart, the final agents are evaluated against the same frozen scripted user, so the reported differences reflect the trained agent rather than a change in the evaluation user.

D.2 Ablation Variants

The ablations in Table 2 modify individual components of ATOA while keeping the remaining training and evaluation settings unchanged. Unless stated otherwise, the same task pools, model backbones, interaction budgets, and agent-training configurations are used as in full ATOA. Table 10 summarizes each intervention.

Alternation and opponent adaptation. The distinction between *Fixed Opponent*, *No Alternation*, and full ATOA isolates two different effects. Fixed Opponent disables failure-guided opponent adaptation entirely after script-based initialization. No Alternation retains the first failure-guided user update but removes the repeated feedback loop: later failures of the improving agent no longer refresh the training opponent. Full ATOA repeatedly updates the user from failures of the latest agent before subsequent agent training. Thus, Fixed Opponent tests the contribution of opponent adaptation, while No Alternation further tests the contribution of repeated alternating updates.

For No Alternation, the first opponent update follows the same temporal order as full ATOA. In particular, it uses the same agent checkpoint and failure data that would produce the first adaptive user in the complete training loop. The comparison also matches the number of agent-training rounds, per-round interaction collection, and agent optimization budget, so differences cannot be attributed simply to less agent training.

User-side design. Random Update tests whether the benefit comes specifically from conditioning the user update on current agent failures rather than from additional user fine-tuning alone. No Mixed Filter removes the capability-boundary criterion used to identify informative failures. No Script Init. tests the role of script-based initialization, whereas No Script Retention tests whether continued exposure to the original script examples is useful during later opponent updates. Accept All Updates removes the mechanism that rejects candidate users that do not provide sufficiently useful training challenges.

Agent-side learning. RAFT Only and DPO Only isolate the two stages of agent learning. RAFT Only retains learning from verified successful trajectories but removes direct successful–failed comparisons. DPO Only retains same-task preference learning but removes the preceding supervised update on verified successes. All user-side components remain identical to full ATOA in these two variants.

E Additional Controlled Experiments

This appendix provides additional controlled comparisons for the main mechanisms of ATOA. We focus on whether the gain from later rounds is caused by adapting the training opponent, rather than simply performing additional agent optimization, and whether selecting user-update data from the current agent's failures is more effective than simply performing additional user fine-tuning.

E.1 Controlled Comparison of Opponent Adaptation

To isolate the effect of opponent adaptation, we branch from the same agent checkpoint and vary only the source of the interactions used for the next agent update. The *adapted-user* branch collects

Table 9: Implementation of the baselines in Table 1. Methods are grouped following the organization of the main results.

Method	Implementation in our framework
Reference baselines	
Base	The pretrained agent is evaluated directly without task-specific post-training.
Fixed-user training	The agent is trained with the same RAFT and trajectory-level DPO stages as ATOA, but all training interactions are collected from a fixed user consisting of the behavioral controller and an untrained language renderer. The user is never updated during agent training.
User simulation	
Persona Policies	We sample task-preserving persona policies from the persona generator and use them to condition the training user, introducing variation in behaviors and communication styles without changing the underlying task goal. The resulting interactions are used for agent training under the common agent-training recipe.
RealUserSim	We condition the training simulator on grounded behavioral profiles derived from the released RealUserSim profile pool. Profiles are sampled independently of agent performance and remain fixed within an interaction. The resulting trajectories are used to train the agent with the common agent-training recipe.
UserLM-R1	We construct the training user following the profile-conditioned, goal-driven reasoning formulation of UserLM-R1, including its supervised and multi-reward user-model training stages. Once trained, the resulting user model is held fixed while collecting interactions for agent training.
Adaptive training	
Calibrated Interactive RL	We first align the simulator using training-side interaction data and then optimize the agent through interactive rollouts with the calibrated simulator. The simulator is calibrated before agent optimization and is not repeatedly updated from the current agent's failures.
SENTINEL	We follow the failure-driven Controller–Proposer–Solver loop. Failures of the current agent are summarized into error patterns and used to generate targeted training tasks, after which the solver is trained on the resulting interactions. The user language model itself is not updated from these failures.
SEAD	We retain the separation between a Profile Controller and a user role-play model. The Profile Controller adapts the distribution of user states presented during training, while the role-play model realizes these states in dialogue. Agent training proceeds on the resulting adaptive curriculum.
Agent self-improvement	
EigenData	We use the self-evolving data-generation process to construct additional tool-grounded training interactions from the training-side task schemas and verify them with executable task checks. The agent is then optimized on the verified synthetic interactions following the method's verifier-based post-training procedure.
Divergence-Point DPO	We align successful and failed trajectories from the same task and identify the earliest agent decision at which their behaviors diverge. Preference examples are constructed around these divergence points and used for DPO, focusing the update on decisions that separate successful from failed trajectories.
Bilateral training	
IB-RL-inspired (COOP)	We implement a bilateral variant in which the user and agent are updated through separate parameter paths from joint interaction rollouts. The user-side objective is cooperative and rewards interactions that lead to task completion. This is an adaptation of the bilateral-training principle rather than a reproduction of the original strategic-dialogue setting.

new interactions with the newly accepted opponent. The *previous-user* branch continues with the opponent from the preceding round. We additionally consider a *reused-data* branch that trains from interactions collected in the previous round instead of collecting new interactions.

Table 10: Definitions of the ATOA ablations reported in Table 2.

Variant	Change from full ATOA
No Alternation	The first failure-guided opponent adaptation is performed exactly as in full ATOA, using the same agent checkpoint, failure-derived data, and user-training configuration. The resulting user is then frozen and is not updated from failures of later agents. Agent-training rounds, per-round interaction collection, and agent optimization budgets are matched to full ATOA.
Fixed Opponent	The user receives the same script-based initialization as full ATOA and is then kept fixed throughout agent training. No failure-guided opponent adaptation is performed. Agent-training rounds and budgets are matched to full ATOA.
Random Update	Failure-derived user examples are replaced with the same number of randomly sampled user turns from training-side interactions. This preserves the amount of user-training data while removing the connection between the update and the current agent's failures. Script-based anchor data and candidate selection remain unchanged.
No Mixed Filter	Failure-derived examples are collected without restricting the underlying tasks to mixed-outcome cards. User turns from failed interactions may therefore come from tasks on which the current agent fails consistently, rather than only from tasks near its current capability boundary.
No Script Init.	The initial opponent skips script-based SFT and uses the pretrained user language model directly. Subsequent failure-guided opponent updates follow the full ATOA procedure, including the script-based anchor data.
No Script Retention	The initial user receives the same script-based SFT as full ATOA, but the original script examples are omitted from subsequent opponent updates. Each later candidate is trained from ϕ_{base} using only the newly collected failure-derived examples.
Accept All Updates	Every newly trained user candidate replaces the retained opponent. The candidate-selection criteria in Eq. 16 are disabled, while user training, interaction collection, and agent training remain unchanged.
RAFT Only	The agent is trained only on verified successful trajectories using RAFT. The trajectory-level DPO stage is removed, while opponent adaptation and the remaining training procedure are unchanged.
DPO Only	The RAFT stage is removed. Same-task successful–failed pairs are constructed as in full ATOA, and trajectory-level DPO is applied directly from the incoming agent checkpoint, with a frozen copy of that checkpoint used as the reference policy.

Table 11 reports the Qwen3.6-27B Retail comparisons corresponding to Figure 3(b).

Table 11: Controlled continuation experiments on Qwen3.6-27B Retail. Comparisons within each block start from the same agent checkpoint. Bold values denote training with the adapted opponent.

Starting checkpoint	Continuation	Task success (%)
Round 2	Previous user	42.92
Round 2	Adapted user	**44.85**
Round 3	Reused previous data	46.97
Round 3	Previous user	47.06
Round 3	Adapted user	**47.84**

After Round 2, replacing the previous opponent with the adapted user improves task success from 42.92% to 44.85%. Starting from the Round-3 checkpoint, training with the retained adapted opponent reaches 47.84%, compared with 47.06% when continuing with the previous user and 46.97% when reusing earlier interaction data.

Because the compared branches start from the same agent checkpoint, these differences cannot be attributed to a stronger initialization. The comparison also separates opponent adaptation from simply taking another optimization step: additional training with the previous user or reused data

improves the agent less than collecting interactions from the updated opponent. This supports the role of opponent adaptation in refreshing the training signal as the agent improves.

E.2 Budget-Matched Agent Optimization

The adapted opponent can change not only the interaction distribution but also the amount of usable agent-training data. To isolate the effect of the training opponent from differences in optimization budget, we therefore construct a budget-matched comparison in Telecom.

Both branches start from the same agent checkpoint and use the same number of same-task successful–failed pairs and the same number of DPO updates. The only difference is whether the new interactions are collected with the previous opponent or the adapted opponent.

Table 12: Budget-matched comparison on Qwen3.6-27B Telecom. Both branches use the same agent initialization and DPO optimization budget.

Training opponent	Pairs	DPO steps	Task success (%)
Previous user	63	31	83.12
Adapted user	63	31	**85.76**

With the agent initialization and DPO optimization budget held fixed, training against the adapted opponent improves task success from 83.12% to 85.76%, a gain of 2.64 percentage points. Both branches use the same 63 preference pairs and 31 DPO updates, ruling out a simple training-budget explanation. Together with the continuation results above, this shows that the benefit of opponent adaptation comes from the training signal provided by the updated interaction distribution, rather than from performing more DPO optimization.

E.3 Effect of Failure-Guided User Updates

We further test whether later user updates benefit specifically from the current agent's failures. Two ablations provide complementary controls (Table 13). *Random Update* uses the same amount of additional user-training data but replaces failure-guided examples with randomly sampled user turns. *No Mixed Filter* retains failed interactions but removes the requirement that they come from mixed-outcome tasks.

Table 13: Effect of the user-update data on Qwen3.6-27B task success (%). All variants retain the remaining ATOA training procedure.

Variant	Retail	Airline	Telecom
Random Update	42.81	8.02	79.14
No Mixed Filter	44.36	8.10	77.83
Full ATOA	**47.84**	**9.74**	**85.76**

Random Update underperforms full ATOA by 5.03, 1.72, and 6.62 percentage points on Retail, Airline, and Telecom, respectively. Thus, the gain cannot be explained by additional user fine-tuning alone. Removing the mixed-outcome filter also reduces performance by 3.48, 1.64, and 7.93 points, showing that failures are more useful when they come from tasks on which the current agent can sometimes succeed and sometimes fail.

Together with the continuation experiments above, these results support two separate aspects of ATOA. Failure-guided user updates change the interaction distribution in a way that provides more useful training challenges, while repeated opponent adaptation refreshes these challenges as the agent becomes stronger.

E.4 Reliability of Directive Enforcement

We additionally audit whether the directive-enforcement layer reliably prevents the trainable user language model from changing controller-defined behavior. We distinguish three quantities: *raw*

Table 14: Directive-enforcement audit over evaluation interactions generated by the trained user language models. Raw violations are rejected generations before regeneration; executed violations are measured on the final utterances sent to the agent.

Metric	Retail	Airline	Telecom	Overall
User utterances	90,866	47,756	156,249	294,871
Raw violations	8 (0.009%)	16 (0.034%)	214 (0.137%)	238 (0.081%)
Dialogues with $\geq$1 rejected generation	0.09%	0.56%	2.05%	1.04%
Fallbacks	0	0	0	0
Executed violations	0	0	0	0

violations, where an initial language-model generation fails a directive check; *fallbacks*, where all regeneration attempts fail and the deterministic template is used; and *executed violations*, where the final utterance actually sent to the agent violates the controller directive.

Across 51 evaluation batches with the trained user renderers, covering approximately 22K dialogues and 294,871 user utterances, violations before enforcement are rare and no violating utterance is ultimately executed. Table 14 summarizes the results.

The trained renderers follow controller directives reliably even before enforcement: only 0.081% of candidate utterances are rejected, and no turn requires the deterministic fallback. After enforcement, no directive violation is observed in the final utterances sent to the agent.

To verify that the zero executed-violation rate is not caused by an insensitive checker, we perform an injected-violation audit. We insert 4,950 known violations into Retail examples and 10,727 into Telecom examples. The same checking pipeline detects all 15,677 injected violations, yielding 100% detection in this positive-control test. This supports the interpretation that the observed zero executed violations reflect effective enforcement rather than a failure of the auditing procedure.

The trained user renderers also require substantially less intervention than alternative language renderers under the same guard. With GPT-5.5, 2.91% of candidate generations are rejected and 0.59% of user turns fall back to the deterministic template, with 6.7% of dialogues containing at least one fallback. For the untrained base renderer used by the fixed-user training baseline, raw rejection rises to 16–20% and fallback to 4.8–6.5% of user turns, with 71–83% of dialogues requiring at least one fallback.

These results show that script-based user training improves not only the interaction distribution but also direct compliance with controller directives. The enforcement layer therefore acts primarily as a safeguard for the trained user rather than as a mechanism that frequently overrides its outputs.

F Generalization and Complete Results

This appendix provides additional evaluations under different user models together with the computational-cost analysis. We examine whether the final agents remain effective when the language renderer or user policy is changed. Round-by-round training results are reported separately in Appendix C.5.

F.1 Evaluation with a Different Language Renderer

The main evaluation uses a fixed script-trained language renderer. We further ask whether ATOA remains effective when the same user behavior is expressed by a different language model. To test this, we keep the task cards, behavioral controller, and user decisions unchanged, but replace the language renderer with GPT-5.5.

This setting isolates robustness to language realization: the user follows the same behavioral rules and reveals the same information, but expresses these decisions with different wording. Therefore, strong performance under GPT-5.5 would indicate that the learned agent is not specialized to the particular language patterns of the renderer used in the main evaluation.

Table 15: Retail task success (%) under the original script-trained renderer and an external GPT-5.5 renderer. The behavioral controller, task cards, and user decisions are unchanged.

Method	Script-trained renderer	GPT-5.5 renderer	Change
Base	25.32	27.14	+1.82
Fixed-user training	30.43	31.76	+1.33
ATOA	**47.84**	**46.92**	*-0.92*

Changing only the language renderer produces relatively small changes in task success. ATOA reaches 46.92% with GPT-5.5, compared with 47.84% under the original script-trained renderer, a decrease of only 0.92 percentage points. It also remains substantially above both the base agent and fixed-user training under the external renderer.

Because the behavioral controller, task cards, and user decisions are unchanged, this comparison isolates variation in language realization. The small performance change indicates that ATOA's improvement is not tied to the particular wording patterns of the renderer used during the main evaluation. Instead, the trained agent remains effective when the same underlying user behavior is expressed by a different language model.

F.2 Complete Cooperative-User Results

We further evaluate the final agents with the standard cooperative `gpt-4.1-2025-04-14` user from τ^2-bench. Unlike the external-renderer evaluation above, this setting replaces the scripted user policy rather than only its language realization. The evaluation contains 67 Retail tasks with six trials each, 27 Airline tasks with four trials each, and 40 Telecom tasks with two trials each.

Table 16 expands Figure 3(d) by reporting the absolute task-success rates.

Table 16: Task success (%) with the standard cooperative GPT-4.1 user. Δ is measured relative to the corresponding base agent.

Scenario	Base	Fixed-user	ATOA	Δ vs. Base	
				Fixed-user	ATOA
Retail	80.85	77.11	75.50	-3.74	-5.35
Airline	67.29	66.04	62.04	-1.25	-5.25
Telecom	65.00	83.75	**92.50**	+18.75	+27.50

The effect of scripted-user training transfers differently across domains. In Retail and Airline, ATOA remains effective with the cooperative user but shows decreases of 5.35 and 5.25 percentage points relative to the base agent. The corresponding absolute success rates remain 75.50% and 62.04%. Telecom shows the opposite pattern: ATOA improves cooperative-user task success from 65.00% to 92.50%, a gain of 27.50 points, and also exceeds the fixed-user-trained agent.

These results show that ATOA does not simply produce a uniform shift under a different user policy. Its transfer is domain dependent: training with the more challenging scripted user preserves substantial cooperative-user performance in Retail and Airline while transferring strongly in Telecom. Together with the external-renderer results above, this suggests that the learned agents are not tied to a single evaluation user.

F.3 Computational Cost

We report the computational cost of the complete Qwen3.6-27B Retail run, where ATOA performs four retained agent-training rounds. We separate the cost into interaction collection, user adaptation, and agent optimization with RAFT and trajectory-level DPO. GPU hours are aggregated over the corresponding training and interaction jobs.

Table 17: Computational cost of the Qwen3.6-27B Retail run. Task success corresponds to the round-by-round evaluation reported in Figure 3(a). The final R5 row is a stopping probe rather than an additional retained agent-training round; no further RAFT or DPO optimization is performed. GPU hours are independently rounded to one decimal place, so displayed components may differ from their summed totals by 0.1 GPU hours.

Stage	Collection	User update	RAFT+DPO	Stage cost	Cumulative	Success (%)
Preparation	—	3.6	—	3.6	3.6	25.32
Round 1	40.2	—	13.0	53.3	56.9	34.21
Round 2	33.1	1.1	15.0	49.2	106.1	40.38
Round 3	29.9	3.7	11.8	45.4	151.4	44.85
Round 4	27.3	3.5	13.5	44.3	195.7	**47.84**
R5 stopping probe	31.8	7.2	—	39.0	234.7	47.60

The complete run uses 234.7 GPU hours, of which 162.3 hours are spent on interaction collection, 19.1 hours on user-model training, and 53.3 hours on agent optimization. Interaction collection therefore accounts for the largest part of the total cost, while the additional computation required to update the user language model is comparatively small.

The cost also reflects the diminishing returns observed across training rounds. Task success improves from 25.32% for the base agent to 34.21%, 40.38%, 44.85%, and 47.84% after Rounds 1–4. The first two rounds provide the largest gains, while later rounds continue to improve performance with smaller increments. By the end of Round 4, the retained model has gained 22.52 percentage points over the base agent at a cumulative cost of 195.7 GPU hours.

The R5 stopping probe incurs an additional 39.0 GPU hours for interaction collection and candidate-user evaluation. Its task success reaches 47.60%, slightly below the retained Round-4 result of 47.84%. Because the probe does not justify another agent-training round, no additional RAFT or DPO optimization is performed. This illustrates the practical role of the stopping criterion: the procedure evaluates whether further alternation remains useful before incurring the cost of another full agent update.

G Prompts and Qualitative Examples

This appendix provides abbreviated examples of the inputs used by the agent and user language model, illustrates how failure-guided user-training examples are constructed, and analyzes representative failures of the final agents. These examples complement the formal descriptions in Appendices B and C by showing how controller directives, language realization, and agent-training signals appear in actual interactions. We also include examples of remaining failures to clarify which challenges persist after alternating training and to distinguish improvements in the training signal from the underlying difficulty of long-horizon tool use.

The examples are based on logged interactions and are abbreviated for readability; omitted portions are marked with "[...]". The examples are selected for illustration rather than as an exhaustive taxonomy of behaviors. Complete prompts, tool schemas, and additional implementation details are included in the Supplementary Material.

G.1 Prompt and Training Formats

Agent input. The agent follows the standard τ^2-bench service-agent format. It receives the domain policy, available tools, dialogue history, and previous tool results, but does not observe the behavioral script, controller directive, or task verifier. In Telecom, user-side device actions are also hidden from the agent except for information subsequently communicated by the user.

Example 4 (Agent input)

System instruction: You are a customer service agent that helps the user according to the provided domain policy. At each turn, either respond to the user or make an available tool call.
Context: Domain policy, available tools, dialogue history, and previous tool results.
Output: The next agent response or tool action.

User language-model input. The user language model receives fixed role instructions and persona information together with the current agent message and the directive generated by the behavioral controller. The directive specifies exactly what may be expressed at that turn. The renderer does not observe the verifier or the controller's internal decision state.

The following example is taken from a Telecom interaction.

Example 5 (User language-model input and output)

Agent: Thank you, John. I've found your account. [. . .] Could you please tell me which phone number is experiencing the MMS issue?

Controller directive: *Reveal that the user is currently abroad in France.*

User: I think the issue might be related to the fact that I am currently abroad in France. I'm not sure if that changes which line it is, but that is the situation right now.

The controller determines that the location may be disclosed at this turn; the language model determines only how this information is expressed.

User SFT. Script-based SFT uses the same directive-to-language format as inference. A teacher model provides the target utterance, while the controller determines which information is permitted.

Example 6 (Script-based user SFT example)

Agent: Could you tell me whether you are currently at home or traveling abroad?

Controller directive: *Reveal that the user is currently abroad in France.*

User: I'm currently abroad in France. Let me know what you need me to check next.

The target teaches the language model how to realize the directive. The decision to disclose the location, rather than other pending information, remains fixed by the controller.

Agent training. RAFT converts each agent turn from a verified successful trajectory into a supervised training example. Trajectory-level DPO instead pairs a successful and failed trajectory from the same task, with only agent tokens contributing to the preference objective.

Example 7 (Same-task preference pair)

Task: The user's phone shows "No Service." Airplane Mode is enabled on the device, while a suspended account line acts as a distractor.
Preferred trajectory: The agent identifies the account, checks the relevant lines, asks the user to inspect the device status, identifies Airplane Mode, and instructs the user to turn it off. The verifier confirms that service is restored.
Rejected trajectory: The agent focuses on the suspended distractor line, attempts unrelated account actions, and eventually transfers the interaction without disabling Airplane Mode.

Both trajectories correspond to the same underlying task. DPO compares the agent-token probabilities over their complete interaction histories.

G.2 Example of Failure-Guided User Training

Failure-guided opponent adaptation uses user utterances from failed attempts of mixed-outcome tasks. The following Retail example shows how a turn from a failed interaction becomes a user-training example.

Example 8 (From a failed interaction to a user-training example)

User: Look, let's speed this up. Your name is Lei Wilson and your zip code is 32255. Can we please find the account now?

This utterance comes from a failed interaction of the Round-2 Retail agent on a mixed-outcome task. The controller had already determined that the identity information could be disclosed at this turn. The guard-approved utterance is therefore retained as a target for the next user-language-model update; the underlying disclosure decision itself is unchanged.

In this Retail update, the failure-guided data contain 165 user utterances. Against the same Round-2 agent, cards, and trial seeds, the resulting candidate user increases the number of mixed-outcome cards from 23 to 31 and the success–failure pairing potential from 45 to 61, while agent success decreases from 40.4% to 37.8%. The candidate therefore exposes more tasks near the agent's capability boundary without making the interaction easier and is accepted for the subsequent round.

The following agent updates improve scripted-user task success from 40.38% to 44.85% and then 47.84%. We interpret these results at the aggregate task set level rather than attributing improvement on an individual task to the specific user utterances extracted from that task.

G.3 Representative Remaining Failures

ATOA substantially improves task success, but some difficult interactions remain unresolved. We provide two representative examples from the final agents to illustrate the types of errors that persist.

Example 9 (Retail: incomplete multi-step execution)

User: Hi. I need to cancel my order, number W7109609. I ordered it by mistake.

Agent: [...] Do you confirm you want to cancel this order with the reason "ordered by mistake"?

User: Yes, I confirm. Go ahead and cancel it.

Agent: Your order W7109609 has been successfully cancelled. [...] Is there anything else I can help you with?

User: No, that is all. Thank you for your help. ###STOP###

The interaction reaches a partially completed state, but the full task requirements are not satisfied. This illustrates a remaining failure mode in which the agent completes an intermediate operation but does not finish the full multi-step task before terminating the dialogue.

Example 10 (Telecom: incomplete account-side resolution)

The agent correctly identifies that the line has exceeded its data allowance and proposes the appropriate account-side action. However, the interaction returns to repeated diagnostic steps before that action is completed, and the dialogue terminates without reaching the required final state.

These examples show that the remaining errors are primarily associated with incomplete multi-step execution and recovery during long-horizon interactions. ATOA improves the training signal presented to the agent, but difficult planning and execution failures can still remain.