Iterative Policy Optimization: Batch-Online Multi-Turn RL via Single-Turn RL

Daniel R. Jiang[*] Meta

Ankur Samanta[*†] Meta, Columbia University

Yukai Yang Meta

Jalaj Bhandari Meta

Rémi Munos FAIR at Meta

Tyler Lu Meta

Abstract

Practical LLM agents often operate over multi-turn conversations where success is determined only after the full interaction ends. Most multi-turn RL methods train via on-policy rollouts, but unlike in single-turn RL, the policy cannot produce a trajectory alone, since an external environment must respond after each agent turn. For conversational agents, this environment is a user, but real users are generally unavailable inside the training loop and simulated users are difficult to build faithfully. Separately, real-world deployment is rarely fully online or fully offline. The common production pattern is called *batch online*, where the current policy is deployed to collect a batch of interaction data, then retrained on that batch and redeployed. We show that this batch-online setting creates an opportunity for applying the classical *approximate policy iteration* algorithm. Our central observation is that running standard token-level, single-turn GRPO with a learned turn-level Q-function as the reward model is a policy improvement step for the multi-turn problem. Building on this, we present *Iterative Policy Optimization* (IterPO), which alternates between fitting Q^π from logged Monte Carlo returns via standard reward modeling (policy evaluation) and running single-turn GRPO against Q^π (policy improvement). During policy improvement, the learned Q^π scores candidate responses by their expected downstream return, avoiding the need for simulator rollouts of the remaining conversation. Overall, this allows us to do *principled multi-turn RL using only single-turn RL methods* without an interactive environment or user simulator inside the training loop. We demonstrate the effectiveness of IterPO on six multi-turn negotiation environments.

1 Introduction

Many conversational AI agents are designed to drive specific *outcomes*, including resolving a support issue, completing a task, or helping a customer reach a decision. These interactions play out over multiple turns, and whether the agent succeeded is often only apparent after the conversation ends. Earlier responses matter only through their downstream effects on later ones, so optimizing for outcomes is a long-horizon credit-assignment problem. Yet most widely used RLHF tooling targets *single-turn* objectives, creating a gap between what we want to optimize and the tools available to do it.

Multi-turn RL is more challenging than single-turn RLHF because the policy cannot produce a trajectory by itself. After each agent response, an external environment must supply the next

[*]Equal contribution.
[†]Work done at Meta.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

observation. For coding and web agents, real executable systems can supply these observations [21, 55, 51], but conversational agents need a *user simulator* [54, 4], which is difficult to build and validate [27] and for many bespoke tasks simply unavailable. Most multi-turn RL methods extend on-policy optimizers such as GRPO and PPO to full trajectories [31, 49, 46, 14, 29, 47, 20], and thus they require an interactive environment inside the training loop. This requirement limits their applicability to tasks for which a reliable environment or user simulator is available.

Separately, real-world deployment is rarely fully online or fully offline. The common regime is *batch online* [11], in which the current policy is deployed, a batch of real interactions is logged over a deployment window (e.g., an A/B test lasting days or weeks), and the policy is retrained offline before redeployment. We show that this batch-online regime creates an opportunity for *approximate policy iteration* [5]. The key observation is that a standard token-level RLHF optimizer, when given a turn-level Q-function as its reward model, acts as a policy improvement operator for the response-level multi-turn problem. In our proposed approach, we learn a state-action value function Q^π from the logged interactions that predicts the expected downstream return of a candidate response under the environment dynamics and current policy. We then use Q^π to score candidate responses *without* rolling out the remainder of the conversation. Because each deployment round produces on-policy trajectories, the approximation of Q^π can be obtained via standard supervised reward modeling on Monte Carlo returns and requires no temporal-difference learning. Thus, both policy evaluation and policy improvement can be implemented with *off-the-shelf* single-turn RLHF tools, without bespoke multi-turn training infrastructure and without requiring an environment or user simulator inside the training loop.

In this work, we make the following contributions.

- **Multi-turn to single-turn RLHF reduction.** We give a formal reduction of multi-turn policy improvement to a sequence of standard single-turn, token-level RLHF problems, enabling direct use of off-the-shelf RLHF tools. We prove that standard single-turn GRPO with a learned turn-level state-action value function Q^π as the reward model can be viewed as an approximate policy improvement step for the multi-turn problem (Theorem 3.2). The learned Q^π supplies temporal credit assignment across conversational turns, while the single-turn GRPO finds the policy that approximately optimizes this quantity.
- **Approximate policy iteration without a simulator.** We propose *Iterative Policy Optimization* (IterPO), a simple batch-online approximate policy iteration method that alternates between fitting Q^π to Monte Carlo returns from logged conversations and improving the policy with standard GRPO, using Q^π to score candidate responses. By estimating the expected downstream return of each candidate response from the current conversation state, Q^π enables policy improvement without rolling out the remaining conversation in an environment simulator.
- **Multi-turn environments and empirical evaluation.** We adapt six existing negotiation datasets into a suite of synthetic multi-turn environments for LLM conversational agents and open-source the code. Across these environments, we show that IterPO outperforms single-turn and multi-turn baselines over multiple rounds of data collection and retraining.

2 Preliminaries

2.1 Multi-Turn RLHF

We study post-training for conversational agents in a *multi-turn* setting where the interaction alternates between environment outputs x_t and agent actions a_t as follows.

$$x_1, a_1, x_2, a_2, \ldots$$

In the standard setting, we can think of x_t simply as the message sent by the user (or customer), while a_t is the message sent by the agent. However, by allowing the definition of x_t to be flexible, we can instantiate other variations of the problem where additional information besides the user's message may arrive. In general, x_t is a generic environment output that bundles all of the information emitted between an agent's consecutive actions.

We model this as an MDP $(\mathcal{S}, \mathcal{A}, P, P_0, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition kernel, P_0 is the initial state distribution, R is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

State representation. The state representation is the full history up to the current turn,

$$s_t = (x_1, a_1, x_2, a_2, \ldots, x_t) \in \mathcal{S}, \tag{1}$$

with initial state $s_1 = x_1$ drawn from P_0. A stochastic policy $\pi_\theta(a \mid s)$ maps states to action distributions.

Transition dynamics and termination. After action a_t in state s_t, the environment produces the next output

$$x_{t+1} \sim P(\cdot \mid s_t, a_t). \tag{2}$$

The MDP includes an absorbing terminal state $\perp$. After the agent takes a_t and the environment produces x_{t+1}, the episode may terminate. Termination can depend on x_{t+1} (e.g., through a terminal flag indicating "no further messages" or an outcome indicating successful resolution of the conversation). Once in $\perp$, the process remains there.

Rewards and objective. The reward can be any function of the state, action, and next environment output; we write $r_t = R(s_t, a_t, x_{t+1})$. Given a stochastic policy $\pi(a \mid s)$, the expected discounted return from state s is

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \;\middle|\; s_1 = s,\ a_t \sim \pi\right].$$

Under $s_1 \sim P_0$, the objective is $J(\pi) = \mathbb{E}_{s_1 \sim P_0}[V^\pi(s_1)]$.

Example 2.1 (Outcome-driven multi-turn RL)**.** *Here, we define the outcome-driven multi-turn conversational RL problem. Let the agent's message be a_t. Let c_t denote the customer message at turn t, and let o_t denote an* outcome *signal (e.g., task completion, issue resolution, or purchase) that is revealed after the customer's message c_t arrives. We set $x_t := (c_t, o_t)$, so that the interaction has the form $(c_1, o_1), a_1, (c_2, o_2), a_2, \ldots, (c_\tau, o_\tau)$. We suppose that termination occurs at the first outcome, i.e., the first time τ where $o_\tau = 1$. There is a sparse outcome reward $R(s_t, a_t, x_{t+1})$ that is only nonzero when $o_{t+1} = 1$.*

2.2 Batch-Online Deployment

We consider a *batch-online* deployment setting. The current policy π_i is deployed and interacts with the environment (i.e., real users) to produce a batch of complete multi-turn trajectories with observed outcomes. The policy is then retrained offline on this batch before the next deployment round. This deploy-collect-retrain cycle repeats for K rounds.

$$\underset{\text{deploy}}{\pi_0} \xrightarrow{\text{collect}} \mathcal{D}_0 \xrightarrow{\text{train}} \underset{\text{deploy}}{\pi_1} \xrightarrow{\text{collect}} \mathcal{D}_1 \xrightarrow{\text{train}} \cdots \xrightarrow{\text{train}} \pi_K.$$

The setting is "online" because the data-generating policy changes between rounds, and "batch" because each round trains on a fixed dataset rather than updating continuously. This matches the update pattern of classical batch-mode reinforcement learning and approximate dynamic programming [5, 43, 12, 24].

3 Multi-Turn to Single-Turn Reduction via IterPO

In the single-turn setting, actions are typically defined at the *token level*,[3] with planning performed over the sequence of tokens that constitute a response [10, 58]. The state is the sequence of tokens generated so far, and the transition dynamics are to concatenate the latest action (token) with the previous state. In most instantiations, the overall response is then scored according to a reward model after the end-of-sequence (EOS) token. In contrast, the *multi-turn* formulation naturally operates at the *turn level*, where the atomic action is an agent's complete response to the previous turn. This allows for conversation-level planning that occurs over the sequence of responses [28, 48, 37]. Unlike token-level single-turn RLHF, where the transitions are deterministic token concatenations, the multi-turn setting involves unknown environment dynamics P that govern how the conversation evolves.

We show that, under the batch-online setting defined in Section 2.2, multi-turn RL can be cast as approximate policy iteration, with each step reducing to standard single-turn RLHF operations.

[3]There also exist response-level actions in a single-turn setting [3], but in that case the problem is only a single stage, similar to a bandit problem.

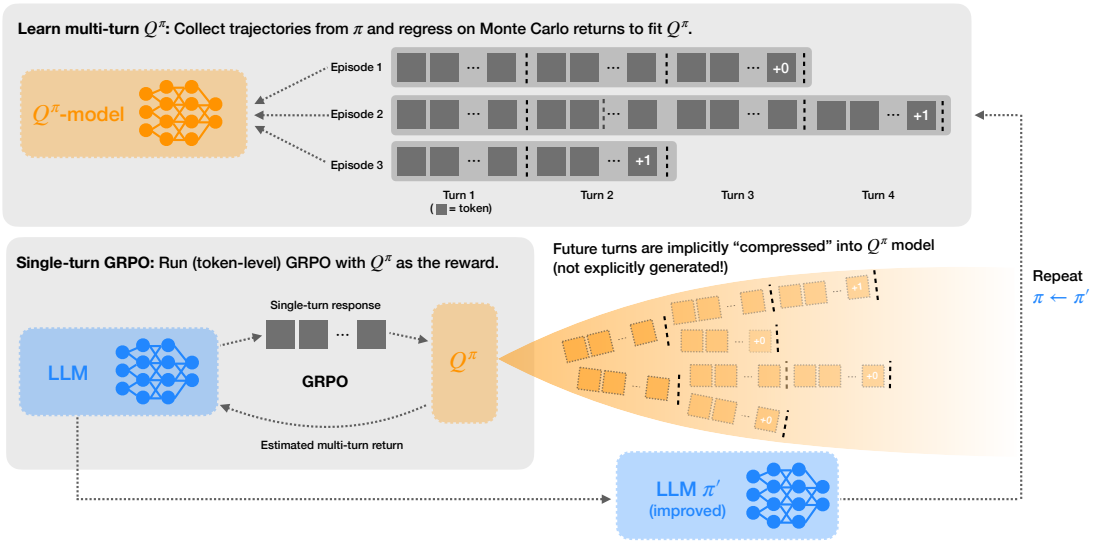

Figure 1: **Reducing multi-turn to single-turn RLHF via IterPO.** We visualize the main steps of our proposed approach. At the top, we collect multi-turn trajectories under the current policy π, compute Monte Carlo returns, and use standard reward-modeling procedures to fit the turn-level Q-function Q^π, which predicts the expected total multi-turn reward under π. At the bottom, holding Q^π fixed, we run standard single-turn (token-level) GRPO using Q^π as the reward model. Future conversation turns are implicitly "compressed" into Q^π, allowing candidate responses to be scored without simulator rollouts during policy improvement. The right side shows how iterating this procedure ($\pi \leftarrow \pi'$) yields multi-turn improvements using only single-turn RLHF tools. Since we proceed in a series of deployment rounds, we refer to this as "batch-online."

3.1 The Reduction

We now give two concrete observations that capture our *policy iteration* perspective on multi-turn RL.

Observation 3.1 (GRPO is an approximate policy improvement operator)**.** *For a reward $R(s,a)$ over states s and completions a, define $\texttt{SingleTurnOpt}(R)$ to be a policy satisfying*

$$\texttt{SingleTurnOpt}(R)(\cdot \mid s) \in \arg\max_{\mu} \mathbb{E}_{a\sim\mu}\big[R(s,a)\big], \qquad \textit{for every } s,$$

where $\mu \in \mathcal{P}(\mathcal{A})$ is a distribution over completions. This idealized operator solves the single-turn reward maximization problem at every state. In practice, GRPO optimizes a shared token-level policy using the reward R, with updates averaged over sampled training states. GRPO can therefore be viewed as an approximation of $\texttt{SingleTurnOpt}(R)$.

Now suppose that Q^π, the state-action value function of a multi-turn policy π, is given to us. A policy that is greedy in the response-level action space with respect to Q^π improves upon π by the classical policy improvement theorem [42]. Our key insight is that since optimizing greedily in response space is difficult, we can instead run a single-turn RLHF optimizer with reward model Q^π, i.e.,

$$\pi' = \texttt{SingleTurnOpt}(Q^\pi). \tag{3}$$

By definition, π' maximizes the expected Q^π-value at every state, so it scores at least as highly as the current policy π. The policy improvement theorem therefore gives $V^{\pi'}(s) \geq V^\pi(s)$ for every state s. Moreover, if we were given Q^*, then $\texttt{SingleTurnOpt}(Q^*)$ would produce an optimal policy.

Notice that in order to guarantee policy π' improves globally over π, we can relax the assumption that π' is greedy with respect to Q^π and only assume a local improvement property, as expressed in the following result (whose proof is given in Appendix B).

Theorem 3.2. *Assume we are able to obtain a (multi-turn) policy π' which produces a single-turn improvement over π from any state s, in the sense that*

$$\mathbb{E}_{a\sim\pi'(\cdot|s)}\left[Q^\pi(s,a)\right] \geq \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a)\right].$$

Then the resulting (multi-turn) policy π' is globally better than π, in the sense that $V^{\pi'}(s_0) \geq V^{\pi}(s_0)$ for any initial state s_0.

Remark 3.3 (Approximate policy improvement)**.** *In practice, both Q-estimation and policy optimization are approximate. Appendix C extends the idealized result to a learned Q-function with uniform estimation error ϵ_Q and per-state optimization error δ_s.*

In practice, we approximate this policy improvement step using standard RLHF methods such as GRPO [38], PPO [36], or REINFORCE [3]. In this work, we use GRPO to implement the policy improvement step.

Remark 3.4 (Token-level RL for response-level policy improvement)**.** *Observation 3.1 follows from the classical policy improvement theorem. Policy improvement provides a bridge between token-level and response-level RL because a token-level RL method can serve as a policy improvement operator for the response-level problem. Token-level RL methods have been shown to be empirically effective in single-turn settings, and Observation 3.1 lets us leverage them directly for multi-turn improvement.*

In Observation 3.1, we assumed that we have access to Q^π. In our next observation, we show that approximating Q^π can be achieved with standard supervised reward-model learning techniques used in single-turn RLHF.

Observation 3.5 (Reward model fitting as policy evaluation)**.** *Consider the standard single-turn RLHF setup. Let $\mathcal{D}_{\text{single}} = \{((s_i, a_i), r_i)\}_{i=1}^N$ be a supervised dataset of prompts s_i, responses a_i, and scalar targets r_i (e.g., human labels). Reward modeling with scalar targets is a supervised learning problem.*

$$\texttt{SingleTurnFitReward}(\mathcal{D}_{\text{single}}) \in \operatorname*{argmin}_R \ \mathbb{E}_{((s,a),r)\sim\mathcal{D}_{\text{single}}}\Big[\ell\big(R(s,a), r\big)\Big],$$

where R belongs to some function class (e.g., a transformer) and ℓ is a supervised loss (e.g., mean squared error).

Now consider a multi-turn policy π and a dataset of N on-policy trajectories $\{\tau^{(i)}\}_{i=1}^N$ collected by rolling out π. Each trajectory is of the form

$$\tau^{(i)} = \big(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, s_2^{(i)}, a_2^{(i)}, r_2^{(i)}, \ldots, s_{T_i}^{(i)}\big),$$

where T_i is the (possibly varying) horizon of trajectory i. For each time step $t \in \{1, \ldots, T_i - 1\}$, define the so-called Monte Carlo return to be

$$G_t^{(i)} \ = \ \sum_{k=0}^{T_i - t - 1} \gamma^k \, r_{t+k}^{(i)},$$

where $\gamma \in [0, 1)$ is the discount factor. We can then form a supervised dataset of state-action pairs and returns,

$$\mathcal{D}^\pi_{\text{multi}} = \big\{\big((s_t^{(i)}, a_t^{(i)}), G_t^{(i)}\big) : i \in [N],\ t \in [T_i - 1]\big\}. \tag{4}$$

Monte Carlo policy evaluation [42] then takes the form

$$Q^\pi \approx \texttt{SingleTurnFitReward}(\mathcal{D}^\pi_{\text{multi}}),$$

a single-turn RLHF operation.[4]

Remark 3.6. *In Observation 3.5, we made use of the Monte Carlo variant of policy evaluation in order to most easily leverage single-turn reward modeling tools. Other methods for performing policy evaluation include SARSA [42] and off-policy Q-evaluation [42, 23]. These methods both use temporal differences (TD), which can have lower variance than Monte Carlo, but the downside is that TD-learning in an LLM setting is expensive and potentially unstable [17]. In multi-turn settings with a reasonable horizon (i.e., the number of turns), Monte Carlo policy evaluation via standard reward modeling should suffice and has the added benefit of simplicity. However, if the data are collected using a policy other than π_i, then off-policy Q-evaluation would be required.*

[4]Why do we focus on Q^π rather than aim for Q^*? This would likely involve an algorithm like fitted Q-iteration, but note that this requires the implementation of a max operator (over responses) to generate Q-labels. This could be another GRPO step, but the number of such iterations is inherently limited, since repeated policy improvement steps can quickly push the policy into regions of the response space that are out-of-distribution with respect to the original data. This is why we leverage the ability to do periodic data collection and perform Q-evaluation for the response-level problem.

Remark 3.7. *Multi-turn RL can be formulated as a monolithic token-level optimization problem (e.g., via PPO). However, this is often impractical. Since environment transitions depend on external user responses, a token-level trainer needs to "wait" for customer responses. In addition, token-level methods may suffer from credit assignment challenges, as sparse terminal rewards must be propagated through thousands of tokens across multiple turns. Our reduction avoids these pitfalls by using a Q-function to compress the long-horizon dependency into a response-level reward, enabling the use of stable single-turn RLHF tools.*

3.2 The IterPO Algorithm

At each IterPO round, we learn a turn-level Q-function Q^π that estimates the (multi-turn) value of the current policy π at the response level, and then run a single-turn, token-level RLHF method (in our case GRPO) using Q^π as the reward model to obtain an improved policy π'. The justification for this algorithm comes directly from Observations 3.1 and 3.5, which together constitute a single step of *policy iteration*.

The pseudocode for IterPO is given in Algorithm 1. As we mentioned above, one of the main advantages of IterPO is that it can directly leverage mature and widely available off-the-shelf single-turn RLHF tools [45, 25, 39].

Algorithm 1 IterPO for Multi-Turn RL

Initialize π_0 (e.g., prompt an instruction-tuned model to suggest a response given a conversation). Set number of deployment rounds K.
Initialize $i \leftarrow 0$.
while $i < K$ **do**
  Deploy π_i and collect trajectories $\{\tau^{(i,j)}\}_{j=1}^{N}$.
  Create the dataset $\mathcal{D}_{\text{multi}}^{\pi_i}$ according to (4).
  Compute $Q^i \leftarrow \texttt{SingleTurnFitReward}(\mathcal{D}_{\text{multi}}^{\pi_i})$.
  Compute $\pi_{i+1} \leftarrow \texttt{SingleTurnOpt}(Q^i)$.
  Update $i \leftarrow i + 1$.
end while
return π_K.

Each round of Algorithm 1 consumes only the logged trajectories from the previous deployment, so no user simulator or interactive environment runs inside the training loop.

3.3 Implementation via a Series of A/B Tests

In practice, π_0 consists of simply prompting a base LLM to generate relevant suggestions based on the conversation session and other contextual information. Trajectories from π_0 can be collected by deploying it through an A/B test, from which we can learn an approximation to Q^{π_0} (see top part of Figure 1). After a policy improvement step (i.e., running GRPO on Q^{π_0}; see bottom part of Figure 1) that results in a new policy π_1, we can deploy π_1 in an A/B test and track outcome metrics, such as the number of purchases. This A/B test for π_1 also serves to collect trajectories, from which we can repeat the above steps to arrive at π_2, and so on. Once we are satisfied with a particular policy π_K, it can be deployed to production.

3.4 Action Coverage and Off-Policy Data Collection

One important consideration is the coverage of the action space of each iteration's collected dataset, which affects how well the learned Q-function generalizes across the action space. This can be an issue if, for example, $\pi_0(\cdot \mid s)$ is near deterministic and we use on-policy Monte Carlo policy evaluation. The resulting $Q^{\pi_0}(s, a)$ would therefore only be reliable at a small set of actions a that are covered by $\pi_0(\cdot \mid s)$. To resolve this, we may need to consider collecting data using a behavioral policy π_0' that is significantly different from π_0 in the sense that it is much more exploratory in its actions. For example, one way to do this is via *persona exploration* where we can prompt the LLM to randomly select an array of characteristics such as tone, emoji usage, response length, etc.

The trade-off with collecting data using π_0' is that we must now perform an off-policy evaluation to learn Q^{π_0} from trajectories collected from π_0'. However, this becomes more complex than on-policy Monte Carlo policy evaluation since it requires off-policy correction, for example through importance weighting or a TD-based evaluation procedure [23].

4 Numerical Experiments

4.1 Multi-Turn Environments

We evaluate our method across six multi-turn negotiation environments that vary in the number of issues under negotiation, the information available to each agent, and the structure of the agreement space. Each environment consists of two LLM-based agents (the trained agent π and a static environment agent π_{env}) that alternate free-form natural-language turns until an agreement or breakdown is reached. As detailed in the environment-specific appendices, the environments use prior datasets and negotiation frameworks to define scenario variables and task structure, which we then use to generate new multi-turn LLM conversations. The environment parsers enforce task-specific agreement formats so malformed or one-sided deals are not incorrectly scored as valid agreements. Both agents are simulated with Qwen-2.5-7B-Instruct (Qwen7b). We briefly describe each environment below; full details including agent prompts and reward formulations are in the appendix.

Craigslist Negotiation (Appendix D.1). A single-issue price negotiation using the CraigslistBargain dataset [16]. π plays a vendor aiming to sell at favorable prices; π_{env} plays a buyer with a private target price. Reward is the sale price divided by the listing price.

Ultimatum Game (Appendix D.2). Two agents negotiate how to split a money pool [6]. π is the proposer; π_{env} is a responder who can accept, reject, or counter-propose. Reward is π's share divided by half the pool size.

Deal or No Deal (Appendix D.3). Two agents divide a shared pool of items (books, hats, balls) with private per-item values [26]. Closure is bilateral. A partition is written as a `MINE/THEIRS` box and a deal closes only when the other party emits an explicit `ACCEPT`. Reward is points earned divided by $U_{\max}/2$ (5 for this dataset).

CaSiNo (Appendix D.4). Two campers divide packages of Food, Water, and Firewood, each with a private priority ordering over supply types [8]. Closure follows the same bilateral `MINE/THEIRS` + `ACCEPT` protocol. Reward is points earned divided by $U_{\max}/2$ (18 for this dataset).

Job Interview (Appendix D.5). A five-issue job offer negotiation with cross-issue dependencies and private utility functions, adapted from Yamaguchi et al. [53]. Reward is a weighted sum of normalized per-issue scores.

Alliance Negotiation (Appendix D.6). Two stakeholders negotiate over five project issues, each with a private score matrix and BATNA threshold, with issue structure drawn from Abdelnabi et al. [1]. Reward is the deal score divided by 50.

4.2 Experimental Setup

We simulate a realistic deployment setting in which a model is iteratively retrained and deployed over multiple rounds. The environment agent π_{env} (an LLM) serves as a reproducible stand-in for real-world deployment. It generates each round's logged batch of conversations but is never queried inside the training loop, which operates purely on the logged trajectories.

In each round, the current policy is deployed to interact with the environment and collect a batch of trajectories; we then use this newly collected data to fine-tune the model before deploying it again. We use "kernel" to refer to one fixed environment instance, including the item values, private preferences, thresholds, and other sampled variables that define a negotiation scenario. Each round generates 400 conversations (100 distinct in-training kernels, each simulated 4 times independently) over $K = 4$ rounds total. We report in-training performance on the 100 in-training kernels from each deployment round and held-out performance on 100 held-out kernels per environment. The per-turn discount factor is $\gamma = 0.9$, which encourages the agent to reach favorable outcomes efficiently. For each environment, we report the expected discounted return $J(\pi)$. Since our environments only produce a terminal reward at the end of the conversation, the total reward of an episode of length τ is

$\gamma^{\tau-1} r_\tau$. This captures both the quality of the negotiated outcome and the efficiency with which it is reached. Results are averaged over 3 seeds. All algorithms are evaluated under the same iterative deployment protocol. We compare the following:

- **IterPO**: Our proposed approach (Algorithm 1), which alternates across deployment rounds between fitting a turn-level Q-function to Monte Carlo returns from logged conversations and using that Q-function as the reward model for single-turn, token-level GRPO.
- **ST-GRPO**: A myopic baseline that applies GRPO to each individual turn, treating them as independent decision problems ($\gamma = 0$).
- **MT-SFT**: A supervised baseline that fine-tunes the model to imitate the agent's own messages from successful trajectories (those ending in agreement), with the loss masked to the agent's turns. It performs no explicit reward optimization.
- **MT-KTO**: A multi-turn version of KTO [13] applied to the agent's messages from each collected conversation, using agreement and no-agreement outcomes as binary feedback.
- **MT-PPO**: A PPO baseline run under the same multi-turn batch-online experimental protocol as the other methods where applicable, using a learned value head for the logged round batch.
- **ArCHer** [56]: A hierarchical actor-critic method for multi-turn RL that learns response-level Q- and V-functions. We use its offline variant (an implicit-Q-learning critic with an advantage-weighted-regression actor), which is suited to our batch-online data regime. The original implementation uses a RoBERTa-based critic [30].
- **ArCHer-LC** (LLM-Critic): A variant of ArCHer whose critic is built on a frozen copy of the policy's language model with lightweight value heads, the same critic architecture as IterPO's Q-function.

Full experimental details are provided in Appendix E.

4.3 Results

Table 1 reports expected discounted return on the held-out kernels. Figure 2 reports the same quantity in-training across all deployment rounds; Table 7 gives the corresponding in-training values. We find that IterPO improves across multiple deployment rounds in most environments and does not exhibit reward collapse. On held-out kernels, IterPO achieves the highest expected discounted return in five of six environments and improves over the initial policy in every case. The single-turn ablation improves over the initial policy in some environments but falls short of IterPO, suggesting that multi-turn credit assignment matters in this setting. MT-SFT only weakly improves over the initial policy, while MT-KTO and MT-PPO are stronger baselines but do not match IterPO overall.

In some settings, agreement rates approach their ceiling early. In Craigslist and Alliance Negotiation, agreement rates rise above 95% after the first training round, while Ultimatum starts near this regime and remains there throughout training (Table 6). With this sparse terminal reward, once agreement frequency saturates, later rounds have less room to improve and return depends more on the quality of the final deal, so performance can plateau or slightly decline. We therefore evaluate each trained method at the checkpoint with the best in-training return over the $K = 4$ deployment rounds. This matches the deployment protocol. After each round, the practitioner can keep the best policy observed so far instead of using the final checkpoint by default.

We also analyze where IterPO's improvement in discounted return comes from. Because the objective discounts by turn, shorter successful conversations directly improve return. IterPO-trained policies do shorten conversations while maintaining or improving agreement rates (Figure 3), but their undiscounted terminal rewards also improve in all six environments (Figure 4). The gains therefore reflect both terms in the discounted objective. The policy reaches acceptable agreements faster and reaches better final deals.

Behavioral patterns. We next analyze IterPO's conversation traces to see what behaviors emerge when training against a sparse outcome reward, without supervising or guiding the negotiation strategy itself (details in Appendix J). In Craigslist, Ultimatum, CaSiNo, and Alliance Negotiation,

[5]Full environment names: Craigslist Negotiation, Ultimatum Game, Deal or No Deal, CaSiNo, Job Interview, Alliance Negotiation (left to right).

Table 1: Expected discounted return $J(\pi)$ on held-out kernels. "Base" reports the initial instruction-tuned policy; trained methods are evaluated at the checkpoint with the best in-training return. Means ± 1 SE across 3 seeds.[5]

Method	Craigslist	Ultimatum	DealOrNoDeal	CaSiNo	Job Interview	Alliance
Base	0.36	0.80	0.52	0.38	0.25	0.48
MT-SFT	$0.41_{\pm 0.01}$	$0.79_{\pm 0.01}$	$0.50_{\pm 0.05}$	$0.42_{\pm 0.01}$	$0.25_{\pm 0.00}$	$0.51_{\pm 0.02}$
MT-KTO	$0.48_{\pm 0.01}$	$0.86_{\pm 0.01}$	$0.56_{\pm 0.00}$	$\mathbf{0.54_{\pm 0.01}}$	$0.29_{\pm 0.01}$	$0.59_{\pm 0.00}$
MT-PPO	$0.38_{\pm 0.02}$	$0.78_{\pm 0.01}$	$0.51_{\pm 0.03}$	$0.38_{\pm 0.02}$	$0.24_{\pm 0.01}$	$0.51_{\pm 0.05}$
ArCHer	$0.45_{\pm 0.01}$	$0.91_{\pm 0.03}$	$0.60_{\pm 0.03}$	$0.53_{\pm 0.03}$	$0.26_{\pm 0.01}$	$0.57_{\pm 0.01}$
ArCHer-LC	$0.38_{\pm 0.01}$	$0.86_{\pm 0.02}$	$0.53_{\pm 0.04}$	$0.40_{\pm 0.04}$	$0.25_{\pm 0.01}$	$0.49_{\pm 0.06}$
ST-GRPO	$0.54_{\pm 0.00}$	$0.94_{\pm 0.05}$	$0.58_{\pm 0.06}$	$0.46_{\pm 0.03}$	$0.28_{\pm 0.04}$	$0.55_{\pm 0.02}$
IterPO	$\mathbf{0.55_{\pm 0.01}}$	$\mathbf{0.99_{\pm 0.00}}$	$\mathbf{0.70_{\pm 0.01}}$	$0.45_{\pm 0.03}$	$\mathbf{0.35_{\pm 0.03}}$	$\mathbf{0.65_{\pm 0.01}}$

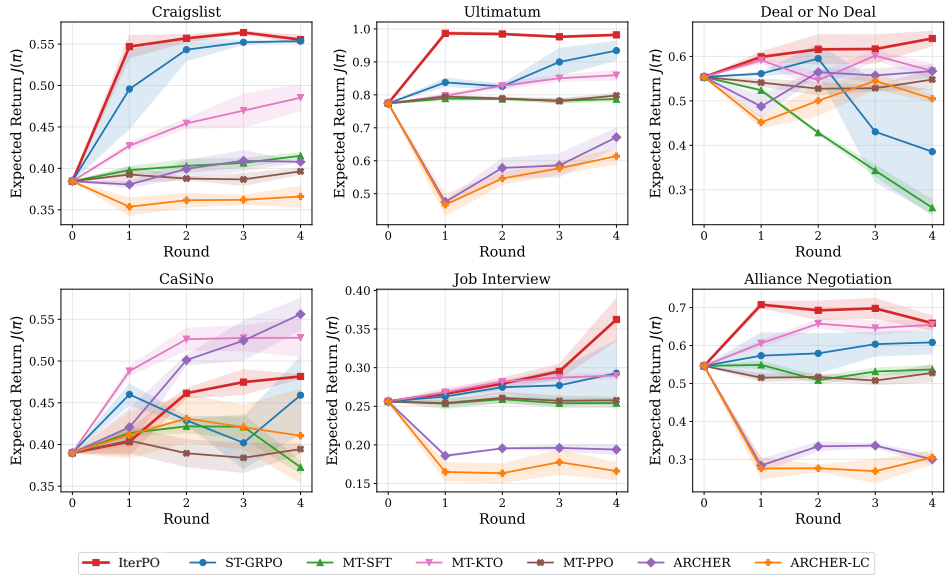

Figure 2: In-training expected discounted return $J(\pi)$ for Qwen7b across the $K{=}4$ deployment rounds. Error bands show ± 1 SE across 3 seeds. Held-out return at the checkpoint with the best in-training return is reported in Table 1.

the main behavioral change is that IterPO reaches acceptable proposals faster. The agent asks for the partner's priorities, proposes a complete agreement earlier, and spends fewer turns on score calculations or intermediate concessions. In Deal or No Deal and Job Interview, the same training also produces richer multi-issue bargaining behavior. In particular, IterPO policies reveal private values less often and use partner-stated information more often. They also trade concessions on lower-value issues for gains on higher-value ones, a strategy known as logrolling. In Job Interview, they concede on the contested issue while holding the rest of the package fixed, a form of selective accommodation [33, 35], showing that sparse outcome rewards can lead the policy to recognizable multi-issue bargaining strategies in these environments.

5 Related Work

On-policy multi-turn RL. Most multi-turn RL for LLMs extends *on-policy* policy-gradient methods to operate over entire trajectories. Multi-turn variants of GRPO and PPO have been applied to coding [31], web [49], and reasoning [46] agents, and a line of work refines turn-level credit assignment by branching from intermediate states or grouping repeated states [14, 29, 47, 20, 57]. These methods are on-policy by construction, since their advantages are estimated from trajectories sampled by the current policy, so each update requires fresh rollouts. Our method instead consumes

logged on-policy trajectories collected between deployment rounds, scoring new candidate responses without rolling out subsequent conversation turns.

Multi-turn preference optimization. Another line of work learns from *trajectory-level* preferences over entire conversations. MTPO [37] uses mirror descent for online preference collection, Xiong et al. [52] propose an iterative multi-turn extension of DPO [34], and Shi et al. [40] target the offline setting. Collecting pairwise preferences over long conversations is often impractical, and these methods require bespoke multi-turn algorithms. Our approach avoids both issues by reducing to standard single-turn techniques and operates in a batch-online regime closest to Xiong et al. [52].

Hierarchical multi-turn RL. The most closely related work is ArCHer [56], which also distinguishes response-level and token-level problems and fits a Q-function to guide token-level optimization. ArCHer uses off-policy Bellman updates via IQL [22] and trains multiple critic networks (Q, V, and baseline), requiring careful tuning for stability. Our approach differs in two ways. First, we show that the token-level optimization step is a policy improvement step for the multi-turn problem (Thms. 3.2 and C.1), which guarantees monotone improvement under exact Q-estimation and statewise policy improvement. Second, our algorithm requires only two phases (supervised Q-fitting followed by GRPO) with no TD learning. POAD [50] decomposes actions into token groups for PPO. REFUEL [15] simplifies ArCHer via regression but requires pairwise trajectory rollouts per state. Nam et al. [32] define an abstract MDP over high-level tutoring actions, a different form of hierarchical decomposition.

Other offline approaches. Several works optimize dialogue from static datasets [19, 18, 41, 44, 9]. Chen et al. [9] add inference-time search on top of an offline-trained model. Our method differs in that it supports periodic online updates, which enable approximate policy iteration across deployment rounds.

6 Limitations and Concluding Remarks

We present IterPO, a batch-online *approximate policy iteration* algorithm for multi-turn RLHF that learns from logged on-policy trajectories without a user simulator inside the training loop. The method reduces multi-turn RL to a sequence of single-turn RLHF problems, namely reward modeling and GRPO, while reusing off-the-shelf single-turn tools with no bespoke multi-turn infrastructure. Empirically, we evaluate IterPO on six multi-turn negotiation environments and show that it outperforms both single-turn and multi-turn baselines in five of six settings. A key assumption underlying our reduction is that accurate Q-function estimation is possible via Monte Carlo returns, which could be more challenging in environments with very long horizons or high stochasticity. In principle, TD-based methods could improve Q-estimation in such settings, though they introduce additional complexities. In the environments we considered, TD-learning was not required to obtain good performance. What was needed, however, was a model whose initial negotiation tendencies produced enough agreements to provide a useful learning signal. If the initial post-trained policy almost always walks away (zero reward signal), additional prompting or SFT warm start may be needed.

Acknowledgments and Disclosure of Funding

This work was supported by Meta. The authors declare no competing financial interests outside of their employment and affiliation with Meta and Columbia University.

References

[1] Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. Cooperation, competition, and maliciousness: LLM-stakeholders interactive negotiation. *arXiv preprint arXiv:2309.17234*, 2024.

[2] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.

[3] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024.

[4] 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.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[6] Federico Bianchi, Patrick John Chia, Mert Yuksekgonul, Jacopo Tagliabue, Dan Jurafsky, and James Zou. How well can LLMs negotiate? NegotiationArena platform and analysis. *arXiv preprint arXiv:2402.05863*, 2024.

[7] Kalyan Chatterjee and William Samuelson. Bargaining under incomplete information. *Operations research*, 31(5):835–851, 1983.

[8] Kushal Chawla, Jaysa Ramirez, Rene Clever, Gale Lucas, Jonathan May, and Jonathan Gratch. CaSiNo: A corpus of campsite negotiation dialogues for automatic negotiation systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3167–3185, 2021.

[9] Zhiliang Chen, Xinyuan Niu, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Broaden your SCOPE! efficient multi-turn conversation planning for llms with semantic space. *arXiv preprint arXiv:2503.11586*, 2025.

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[11] Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics? In *International Conference on Learning Representations*, 2026.

[12] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[13] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization, 2024. URL `https://arxiv.org/abs/2402.01306`.

[14] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025.

[15] Zhaolin Gao, Wenhao Zhan, Jonathan D. Chang, Gokul Swamy, Kianté Brantley, Jason D. Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn RLHF. *arXiv preprint arXiv:2410.04612*, 2025.

[16] He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[17] Joey Hong, Anca Dragan, and Sergey Levine. Q-SFT: Q-learning for language models via supervised fine-tuning. *arXiv preprint arXiv:2411.05193*, 2024.

[18] Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.

[19] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

[20] Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2026.

[21] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? *International Conference on Learning Representations*, 2024.

[22] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

[23] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

[24] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, 2012.

[25] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. `https://github.com/facebookresearch/xformers`, 2022.

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

[27] Ang Li, Haozhe Chen, Hongseok Namkoong, and Tianyi Peng. LLM generated persona is a promise with a catch. *arXiv preprint arXiv:2503.16527*, 2025.

[28] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1192–1202, 2016.

[29] Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. *arXiv preprint arXiv:2512.17008*, 2025.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[31] Meta FAIR CodeGen Team, Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, David Zhang, and Kunhao Zheng. CWM: An open-weights LLM for research on code generation with world models. *arXiv preprint arXiv:2510.02387*, 2025.

[32] Hyunji Nam, Omer Gottesman, Amy Zhang, Dean Foster, Emma Brunskill, and Lyle Ungar. Efficient RL for optimizing conversation level outcomes with an LLM-based tutor. *arXiv preprint arXiv:2507.16252*, 2025.

[33] Dean G Pruitt. *Negotiation Behavior*. Academic Press, 1981.

[34] 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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[35] Howard Raiffa. *The Art and Science of Negotiation*. Harvard University Press, 1982.

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

[37] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. Multi-turn reinforcement learning with preference human feedback. *Advances in Neural Information Processing Systems*, 37: 118953–118993, 2024.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

[40] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. *arXiv preprint arXiv:2406.14868*, 2024.

[41] Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline RL for natural language generation with implicit language Q learning. *arXiv preprint arXiv:2206.11871*, 2022.

[42] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[43] John N. Tsitsiklis and Benjamin Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1–3):59–94, 1996.

[44] Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot AI for task-oriented dialogue with offline reinforcement learning. *arXiv preprint arXiv:2204.08426*, 2022.

[45] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

[46] Zihan Wang et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[47] Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025.

[48] Wei Wei, Quoc Le, Andrew Dai, and Jia Li. AirDialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, 2018.

[49] Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[50] Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition. *arXiv preprint arXiv:2405.15821*, 2024.

[51] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

[52] Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024.

[53] Atsuki Yamaguchi, Kosui Iwasa, and Katsuhide Fujita. Dialogue act-based breakdown detection in negotiation dialogues. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 745–757, 2021.

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

[55] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[56] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. *arXiv preprint arXiv:2402.19446*, 2024.

[57] Siyuan Zhu, Chao Yu, Rongxin Yang, Zongkai Liu, Jinjun Hu, Qiwen Chen, and Yibo Zhang. GAGPO: Generalized advantage grouped policy optimization. *arXiv preprint arXiv:2605.13217*, 2026.

[58] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Contents of the Appendix

A LLM Usage

B Proof of Theorem 3.2

C Approximate KL-Regularized Policy Improvement

D Environment Details

- D.1 Craigslist Negotiation
- D.2 Ultimatum Game
- D.3 Deal or No Deal
- D.4 CaSiNo
- D.5 Job Interview
- D.6 Alliance Negotiation

E Training Details

F Additional Metrics

G In-Training Return

H Held-Out Kernels

I Additional Backbone: Llama3b

J Systematic Behavioral Analysis

A LLM Usage

LLMs serve as the trained policy models and fixed environment agents in our multi-turn experiments (Section 4, Appendices D, E, and I), and as an automated judge for the behavioral analysis of negotiation transcripts (Section 4.3 and Appendix J). LLMs were also used to edit the manuscript for clarity.

B Proof of Theorem 3.2

Proof. By the performance difference lemma (see, e.g., Section 1.5 of [2]), we have

$$\begin{aligned} V^{\pi'}(s_0) - V^{\pi}(s_0) &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\,\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^{\pi}(s,a)\big] \\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\,\big[\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[Q^{\pi}(s,a)\big] - \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q^{\pi}(s,a)\big]\big] \geq 0, \end{aligned}$$

where

$$d^{\pi'}_{s_0}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\,\mathbb{P}(s_t = s \mid s_0, \pi') \tag{5}$$

is the discounted state visitation distribution under π'. □

C Approximate KL-Regularized Policy Improvement

We extend Theorem 3.2 to a KL-regularized reward-maximization objective. The extension allows the learned Q-function to have uniform estimation error ϵ_Q and the policy optimizer to incur state-dependent error δ_s.

Theorem C.1. *Let π be a multi-turn policy, and let $\widehat{Q}$ approximate Q^{π} with*

$$\|\widehat{Q} - Q^{\pi}\|_{\infty} \leq \epsilon_Q.$$

For each state s, define

$$\mathcal{L}_s(\mu) = \mathbb{E}_{a\sim\mu(\cdot\,|\,s)}\big[\widehat{Q}(s,a)\big] - \beta\,\mathrm{KL}\big(\mu(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big),$$

where $\beta \geq 0$. Suppose π' satisfies

$$\mathcal{L}_s(\pi') \geq \sup_{\mu}\mathcal{L}_s(\mu) - \delta_s$$

for every state s. Then, for any initial state s_0,

$$V^{\pi'}(s_0) - V^{\pi}(s_0) \geq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\left[\beta\,\mathrm{KL}\big(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big) - \delta_s - 2\epsilon_Q\right].$$

Consequently, π' improves on π whenever the expectation on the right-hand side is nonnegative.

The experiments use the unregularized case $\beta = 0$.

Proof. Approximate optimality and feasibility of π imply

$$\begin{aligned}\mathcal{L}_s(\pi') &\geq \sup_{\mu} \mathcal{L}_s(\mu) - \delta_s \\ &\geq \mathcal{L}_s(\pi) - \delta_s \\ &= \mathbb{E}_{a\sim\pi(\cdot\,|\,s)}\big[\widehat{Q}(s,a)\big] - \delta_s, \end{aligned} \tag{6}$$

where the last equality uses $\mathrm{KL}(\pi(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)) = 0$. Rearranging Equation (6) and applying $|\widehat{Q}(s,a) - Q^{\pi}(s,a)| \leq \epsilon_Q$ to both policy expectations gives

$$\begin{aligned}&\mathbb{E}_{a\sim\pi'(\cdot\,|\,s)}\big[Q^{\pi}(s,a)\big] - \mathbb{E}_{a\sim\pi(\cdot\,|\,s)}\big[Q^{\pi}(s,a)\big] \\ &\quad\geq \beta\,\mathrm{KL}\big(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big) - \delta_s - 2\epsilon_Q. \end{aligned} \tag{7}$$

Because $\mathbb{E}_{a\sim\pi(\cdot\,|\,s)}[Q^{\pi}(s,a)] = V^{\pi}(s)$, Equation (7) implies

$$\mathbb{E}_{a\sim\pi'(\cdot\,|\,s)}\big[A^{\pi}(s,a)\big] \geq \beta\,\mathrm{KL}\big(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big) - \delta_s - 2\epsilon_Q. \tag{8}$$

Applying the performance difference lemma yields

$$\begin{aligned}V^{\pi'}(s_0) - V^{\pi}(s_0) &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\mathbb{E}_{a\sim\pi'(\cdot|s)}\big[A^{\pi}(s,a)\big] \\ &\geq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\left[\beta\,\mathrm{KL}\big(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big) - \delta_s - 2\epsilon_Q\right].\end{aligned}$$

Thus, π' weakly improves on π whenever

$$\beta\,\mathbb{E}_{s\sim d^{\pi'}_{s_0}}\left[\mathrm{KL}\big(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s)\big)\right] \geq \mathbb{E}_{s\sim d^{\pi'}_{s_0}}\left[\delta_s + 2\epsilon_Q\right],$$

and the improvement is strict when the inequality is strict.

□

D Environment Details

D.1 Craigslist Negotiation

D.1.1 Conversation Generation

Algorithm 2 summarizes the conversation generation.

D.1.2 Agent Prompts

Both agents receive short system prompts that specify their roles and private pricing information. The vendor sees only the item description and listing price and receives no explicit strategy guidance. The customer also sees a target price derived from the original human negotiation. Both prompts require the agent to mark an agreement with `\boxed{AGREED: $X}` or end the negotiation with `\boxed{NO DEAL}`.

Algorithm 2 Craigslist Negotiation: Conversation Generation

Input: Scenario k with listing price p_{list} and buyer target, vendor policy π
`turns` $\leftarrow [\,]$
$m_0 \leftarrow$ `VendorOpeningMessage`(k) *// "I'm selling X for $Y..."*
Append (vendor, m_0) to `turns`
for $t = 1$ to `max_turns` **do**
  `role` $\leftarrow$ `customer` if t is odd else `vendor`
  if `role` $=$ `vendor` **then**
    Activate LoRA adapter
  else
    Activate base model
  end if
  `prompt` $\leftarrow$ `BuildPrompt`$(k, \texttt{turns}, \texttt{role})$
  $m_t \leftarrow$ `LLM`(`prompt`)
  Append $(\texttt{role}, m_t)$ to `turns`
  if m_t contains `\boxed{AGREED: $X}` **then**
    return $(\texttt{turns}, r = X/p_{\text{list}}, \texttt{done} = \texttt{True})$
  else if m_t contains `\boxed{NO DEAL}` **then**
    return $(\texttt{turns}, r = 0, \texttt{done} = \texttt{True})$
  end if
end for
return $(\texttt{turns}, r = 0, \texttt{done} = \texttt{True})$ *// Max turns reached*

Vendor System Prompt

```
You are selling: [item description]

Your listing price is $[listing price]. Negotiate to get the best possible
price.

RULES:
- When you reach an agreement, respond with exactly: \boxed{AGREED: $<price>}
- If no agreement can be reached, respond with: \boxed{NO DEAL}
```

Customer System Prompt

```
You are buying: [item description]

The listing price is $[listing price]. Your target price is $[buyer target].

RULES:
- Negotiate to get as close to your target price of $[buyer target] as possible
- When you reach an agreement, respond with exactly: \boxed{AGREED: $<price>}
- If the price is too high, respond with: \boxed{NO DEAL}
```

The vendor starts each negotiation with a scripted opening:

Vendor Opening Message

```
Hello! I'm selling [item description].

My asking price is $[listing price]. Are you interested?
```

D.1.3 Data Source and Economic Parameters

The CraigslistBargain dataset [16] provides real listing information (item titles, descriptions, listing prices) and buyer target prices from the original human negotiations across six product categories: housing, furniture, cars, bikes, phones, and electronics. Each negotiation scenario uses two parameters from the dataset:

- **Listing price** (p_{list}): The vendor's asking price, taken directly from the Craigslist listing.

- **Buyer target** (p_{target}): The buyer's target price, from the original human negotiation data.

The vendor sees only the listing price; the customer sees both the listing price and their target. No additional economic parameters (reservation prices, cost bases, budgets) are provided to either agent. The vendor must discover its negotiation strategy entirely through reinforcement learning.

D.1.4 Reward Structure

We use a *price ratio* reward that measures the agreed price relative to the listing price:

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} \dfrac{p}{p_{\text{list}}} & \text{if agreement at price } p \\ 0 & \text{if no agreement} \end{cases}$$

The reward assigns zero to failed negotiations and scales successful ones by the agreed price, so it rewards both agreement and higher sale prices. For example, a sale at the listing price yields 1.0; a \$32 sale on a \$40 listing yields $32/40 = 0.8$.

D.2 Ultimatum Game

D.2.1 Agent Prompts

In the Ultimatum Game, a Proposer (π, the trained agent) divides a resource pool with a Responder (π_{env}, the static environment agent), who may accept, reject, or counter-propose. Parallel system prompts specify each agent's role, the pool size, the round limit, and the terminal markers. We show the Proposer prompt below. The Responder prompt instead assigns the agent to Player 2 and retains the convention that `$X` denotes Player 1's share.

Proposer System Prompt (π)

```
You are Player 1 (the Proposer) in a resource-splitting game.

There is a pool of $[pool_size] to divide between you and Player 2.  You start
with the entire pool; Player 2 starts with nothing.  You will negotiate over
[num_rounds] total rounds (alternating turns).

IMPORTANT: If no agreement is reached by the end of [num_rounds] rounds, BOTH
players get NOTHING. Rejection destroys the entire pool.

On your final turn, you must either Accept or Reject (no counter-proposals).

RULES:
- Propose splits, counter-propose, or accept the other player's proposal
- When you reach an agreement, respond with exactly:  \boxed{AGREED: split=$X}
where X is YOUR share
- To reject and end the game (both get zero), respond with:  \boxed{NO DEAL}
```

D.2.2 Opening Message

The Proposer starts each negotiation with a scripted opening message:

Proposer Opening Message

```
Hello!  We need to split $[pool_size] between us.  I hold the pool and will make
the first proposal.  If we can't agree, we both get nothing.  What do you think
would be a fair split?
```

D.2.3 Data Source

We generate scenarios from every combination of five pool sizes and three round limits:

- **Pool size** (p): drawn from $\{100, 500, 1000, 5000, 10000\}$.

- **Number of rounds**: drawn from $\{4, 6, 8\}$.

The Cartesian product yields 15 configurations. When a training round requires more than 15 scenarios, we cycle through them. This construction follows the Ultimatum Game formulation in NegotiationArena [6].

D.2.4 Reward Structure

The reward measures the Proposer's agreed share relative to a baseline of an equal split:

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} \dfrac{X}{p/2} & \text{if agreement with Proposer share } X \\ 0 & \text{if no agreement (\texttt{\textbackslash boxed\{NO DEAL\}})} \end{cases}$$

where p is the pool size. An equal split ($X = p/2$) yields a reward of 1.0. Taking the entire pool ($X = p$) yields a reward of 2.0. A no-deal outcome, where both players receive nothing, yields a reward of 0.

D.3 Deal or No Deal

D.3.1 Agent Prompts

Both agents receive parallel system prompts that describe the item pool, private valuations, and bilateral-acceptance protocol. The trained agent π observes only its own private values, while the environment agent π_{env} observes a different set of private values. Neither agent observes the other's valuations.

System Prompt (both agents)

You are negotiating to divide items with another person. You each have different private values for the items -- the other person's values are unknown to you.

The items are: book=(count:X value:Y) hat=(count:X value:Y) ball=(count:X value:Y)

Your goal is to maximize your total points. You get points only for items YOU receive. All items must be allocated between the two of you.

PROCESS:
Step 1 -- Discuss preferences in plain text. Share which items you value most and least, and ask the other person about theirs. Do not write the AGREED box yet.

Step 2 -- Once you both understand each other's preferences, propose an allocation by writing the AGREED box. Both sides may make counter-proposals; iterate until you converge.

PROTOCOL -- how deals close:

1. PROPOSE an allocation by writing:
\boxed{AGREED: MINE: book=n_b hat=n_h ball=n_l; THEIRS: book=$(B-n_b)$ hat=$(H-n_h)$ ball=$(L-n_l)$}
where MINE is the share YOU receive and THEIRS is the share the OTHER person receives. Use actual integers in place of the n_* placeholders. REQUIRED: include ALL THREE items in BOTH MINE and THEIRS, in book/hat/ball order, even when a count is 0. The two sides must partition all items exactly.

2. To ACCEPT the other person's most recent PROPOSE, write exactly:
\boxed{ACCEPT}
This seals the deal at the partition THEY proposed.

3. To REJECT the standing proposal without closing the deal, write:
\boxed{REJECT}
Then continue negotiating in plain text or with a new PROPOSE.

4. A deal closes ONLY when one party writes a PROPOSE box and the OTHER party

```
writes \boxed{ACCEPT}.  Plain text like ‘‘I agree’’ or silence does NOT close a
deal.  Writing a new PROPOSE supersedes any standing proposal.
5.  If no agreement can be reached at all, write:  \boxed{NO DEAL}
6.  If you do not feel the standing proposal is the best outcome for you,
negotiate further in plain text until you are able to find a better outcome.
```

Per-turn private-values reminder. Before each generation, we prepend a one-line reminder of the active agent's private values and the item pool to the most recent user message (e.g., `[REMINDER -- your private values: book=1 hat=4 ball=2. Pool: book=4 hat=1 ball=1. Maximize MINE·values.]`). The reminder keeps these values in the immediate context because base models otherwise drift from their assigned valuations over long dialogues.

Validation. The environment validates every AGREED proposal box before treating it as the standing proposal. All three item types (book, hat, ball) must appear in both MINE and THEIRS, every count must be a non-negative integer, and $\text{MINE}_i + \text{THEIRS}_i$ must equal the pool count for each item i. An invalid box remains ordinary dialogue and cannot be the target of an ACCEPT. The bilateral rule requires an explicit ACCEPT or verbatim echo from the other party, so both agents must commit to the same partition.

D.3.2 Opening Message

Agent 1 (π) starts each negotiation with a scripted opening message that states the item counts without revealing any value information.

Opening Message

```
Hello!  We need to divide some items between us:  X books, Y hats, Z balls.
Let’s discuss how to split them fairly.  What items are you most interested in?
```

D.3.3 Data Source

Scenarios come from the Deal or No Deal dataset of Lewis et al. [26]. We load the `dialogues` configuration of `mikelewis0/deal_or_no_dialog` from Hugging Face. The dataset contains approximately 5,000 negotiation dialogues collected through Amazon Mechanical Turk. Each scenario specifies the counts of books, hats, and balls and assigns a private valuation vector to each agent. The construction sets each agent's total value across the item pool to 10, so the agents have equal maximum attainable utility but different preferences over the items.

D.3.4 Reward Structure

When a deal closes, we recover the trained agent's allocation from the accepted proposal. We use MINE when the trained agent proposed the allocation and THEIRS when the partner proposed it. The reward is

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} \dfrac{\sum_{i \in \{\text{book,hat,ball}\}} a_i \cdot v_i}{U_{\max}/2} & \text{if agreement} \\ 0 & \text{if no deal} \end{cases}$$

where a_i is the number of units of item i allocated to the trained agent, v_i is its private value for that item, and $U_{\max} = \sum_i c_i v_i$ is the utility obtained from receiving the entire pool. Dividing by $U_{\max}/2$ assigns a reward of 1.0 to a split worth half the maximum utility and 2.0 to receiving every item. In the standard Deal or No Deal dataset, $U_{\max} = 10$ for every agent and scenario, so the implementation uses a denominator of 5.

An invalid PROPOSE box remains ordinary dialogue and cannot be accepted. A conversation that ends without a valid PROPOSE/ACCEPT pair receives $R = 0$.

D.4 CaSiNo

D.4.1 Agent Prompts

The trained agent (π, Camper 1) and static environment agent (π_{env}, Camper 2) receive parallel system prompts containing the bilateral-acceptance protocol and different private priorities and participant-written reasons for Food, Water, and Firewood.

System Prompt (both campers)

```
Imagine that you are on a camping trip.  Apart from some basic supplies, you can
collect additional food packages, water bottles and firewood to make your trip
better.  Since these are limited, you will have to split them with your campsite
neighbor.

There are 3 packages each of Food, Water, and Firewood (9 total) to divide.

Your priorities:
- [Issue] is [High/Medium/Low] priority ([reason])
- [Issue] is [High/Medium/Low] priority ([reason])
- [Issue] is [High/Medium/Low] priority ([reason])

Points per package:  High priority=5, Medium=4, Low=3.  Maximize your points.
Try hard to get as many items as you can!

PROCESS:
Step 1 -- Discuss preferences in plain text.  Share which items you prioritize
highest and lowest, and ask the other person about theirs.

Step 2 -- Once you both understand each other's preferences, propose an
allocation by writing the AGREED box.  Both sides may make counter-proposals;
iterate until you converge.

PROTOCOL -- how deals close:

1.  PROPOSE an allocation by writing:
\boxed{AGREED: MINE: Food=n_f Water=n_w Firewood=n_fw; THEIRS: Food=(3-n_f)
Water=(3-n_w) Firewood=(3-n_fw)}
where MINE is your share and THEIRS is the other person's.  Use integers in
{0,1,2,3}; the two sides must partition all 9 packages exactly.

2.  To ACCEPT the other person's most recent PROPOSE, write exactly:
\boxed{ACCEPT}
3.  To REJECT the standing proposal without closing the deal, write:
\boxed{REJECT}
4.  A deal closes ONLY when one party writes a PROPOSE box and the OTHER party
writes \boxed{ACCEPT}.  Plain text like "I agree" or silence does NOT close a
deal.  Writing a new PROPOSE supersedes any standing proposal.

5.  If no agreement can be reached at all, write:  \boxed{NO DEAL}

6.  If you do not feel the standing proposal is the best outcome for you,
negotiate further in plain text until you are able to find a better outcome.
```

Each scenario assigns every camper one High-, one Medium-, and one Low-priority resource. We fill the bracketed fields with these assignments and the participant-written reasons recorded verbatim in the dataset.

Per-turn private-priority reminder. Before each generation, we prepend a one-line reminder of the active agent's private priorities, their point values, and the resource pool to the most recent user message (e.g., `[REMINDER -- your private priorities: Food=High(5pts) Water=Medium(4pts) Firewood=Low(3pts). Pool: Food=3 Water=3 Firewood=3. Maximize MINE·points.]`). The reminder keeps these priorities in the immediate context over long dialogues.

Validation. The environment validates every AGREED proposal box before treating it as the standing proposal. Food, Water, and Firewood must appear in both MINE and THEIRS, every count must be a non-negative integer, and MINE_i + THEIRS_i must equal 3 for each resource i. An invalid box

remains ordinary dialogue and cannot be accepted. A deal closes only after the other camper writes an explicit ACCEPT or repeats the proposal verbatim, so both campers commit to the same allocation.

D.4.2 Opening Message

Camper 1 (π) starts each negotiation with a scripted opening message:

Camper 1 Opening Message

```
Hi there!  Looks like we need to split up some camping supplies.  I have some
preferences about what I'd like.  How about you?  What are you looking for most?
```

D.4.3 Data Source

Scenarios come from the CaSiNo dataset [8], which contains 1,030 campsite negotiation dialogues collected through Amazon Mechanical Turk. Each record provides both campers' High, Medium, and Low priority assignments over Food, Water, and Firewood, along with participant-written reasons (e.g., "Water is High priority because I get dehydrated easily").

We load `kchawla123/casino` from Hugging Face and partition its single split 85/15 into in-training and held-out partitions. Each scenario uses both campers' priority assignments and associated reasons.

D.4.4 Reward Structure

When a deal closes, we recover the trained agent's allocation from the accepted proposal. We use MINE when the trained agent proposed the allocation and THEIRS when the partner proposed it. The trained agent receives 5 points per package of its High-priority resource, 4 per package of its Medium-priority resource, and 3 per package of its Low-priority resource.

The reward is

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} \dfrac{\sum_{i \in \{\text{Food,Water,Firewood}\}} n_i \cdot w_i}{U_{\max}/2} & \text{if agreement} \\ 0 & \text{if no deal} \end{cases}$$

where $n_i \in \{0, 1, 2, 3\}$ is the number of packages of resource i allocated to the trained agent and $w_i \in \{5, 4, 3\}$ is the corresponding priority weight. Because every camper has one High-, one Medium-, and one Low-priority resource, receiving all nine packages gives $U_{\max} = 3(5{+}4{+}3) = 36$ in every scenario. The implementation therefore divides by $U_{\max}/2 = 18$, assigning a reward of 1.0 to a split worth half the maximum utility and 2.0 to receiving every package.

An invalid PROPOSE box remains ordinary dialogue and cannot be accepted. A conversation that ends without a valid PROPOSE/ACCEPT pair receives $R = 0$.

D.5 Job Interview

D.5.1 Agent Prompts

The Job Interview environment pairs a trained Recruiter policy π with a static Worker policy π_{env}. The agents negotiate Salary, Weekly Holiday, Workplace, Company, and Position. The Recruiter prefers lower Salary and Weekly Holiday values, while the Worker prefers higher values; each agent separately values the three discrete issues. Both agents receive parallel system prompts that specify their role, the five issues, and a procedurally generated utility function. We show the Recruiter prompt below and describe the Worker-specific changes afterward.

Recruiter System Prompt (π)

```
You are a recruiter negotiating a job offer with a candidate.  You need to
agree on 5 issues:  Salary (20-51, in thousands), Weekly Holiday (2-7 days),
Workplace (Tokyo/Seoul/Beijing/Sydney), Company (Google/Facebook/Apple/Amazon),
```

```
and Position (Engineer/Manager/Designer/Sales).

Your preferences (importance weights and option values):

- Salary (range 20-51, you prefer LOWER): importance = [weight]
- WeeklyHoliday (range 2-7, you prefer LOWER): importance = [weight]
- Workplace (importance = [weight]):  Tokyo=[val], Seoul=[val], Beijing=[val],
Sydney=[val]
- Company (importance = [weight]):  Google=[val], Facebook=[val], Apple=[val],
Amazon=[val]
- Position (importance = [weight], value depends on Company chosen):
  If Company=Google:  Engineer=[val], Manager=[val], Designer=[val], Sales=[val]
  If Company=Facebook:  Engineer=[val], Manager=[val], Designer=[val],
Sales=[val]
  If Company=Apple:  Engineer=[val], Manager=[val], Designer=[val], Sales=[val]
  If Company=Amazon:  Engineer=[val], Manager=[val], Designer=[val], Sales=[val]

Your goal is to maximize your total score.  The candidate has different
preferences.  Try to find a deal that works for both of you.

RULES:
- When you reach an agreement, respond with exactly:  \boxed{AGREED: Salary=X
Holiday=Y Workplace=Z Company=W Position=P}
- If no agreement can be reached, respond with:  \boxed{NO DEAL}
```

The Worker prompt changes the role from recruiter to job candidate and reverses the direction of preference for the integer issues: the Worker prefers higher Salary and Weekly Holiday values, while the Recruiter prefers lower values. We sample each party's values for Workplace, Company, and Position independently, so their discrete preferences may agree or conflict.

D.5.2 Opening Message

The Recruiter starts each negotiation with a scripted opening:

Recruiter Opening Message

```
Hello!  I'm the recruiter and I'd like to discuss the terms of your offer.
We need to agree on salary, weekly holiday, workplace location, company, and
position.  What matters most to you?
```

D.5.3 Data Source

We adapt the scenario structure from the multi-issue negotiation framework of Yamaguchi et al. [53]. For each scenario, we independently sample the Recruiter's and Worker's utility functions. The resulting pair of utility functions defines the scenario's Pareto frontier.

The agents negotiate two integer-valued issues and three discrete issues:

- **Salary** (integer, 20-51): salary in thousands. The Recruiter prefers lower values, while the Worker prefers higher values.
- **Weekly Holiday** (integer, 2-7): days off per week. The Recruiter prefers lower values, while the Worker prefers higher values.
- **Workplace** (discrete): Tokyo, Seoul, Beijing, or Sydney.
- **Company** (discrete): Google, Facebook, Apple, or Amazon.
- **Position** (discrete): Engineer, Manager, Designer, or Sales. The value of a Position depends on the selected Company, introducing a cross-issue dependency.

We generate each party's utility function independently. We draw five issue-importance weights uniformly, normalize them, rescale them to $[0.1, 0.6]$, and renormalize them to sum to one. For Workplace and Company, we draw option values and min-max normalize each issue's values to $[0, 1]$. For Position, we draw and normalize a separate set of values for each Company, yielding a 4×4 Company-by-Position table.

This procedure generates an unlimited supply of unique scenarios.

D.5.4 Reward Structure

For a valid agreement, the Recruiter's reward is the weighted sum of its normalized per-issue scores:

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} \sum_{i \in \mathcal{I}} w_i \cdot \text{score}_i & \text{if the agreement is valid} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{I} = \{\text{Salary}, \text{WeeklyHoliday}, \text{Workplace}, \text{Company}, \text{Position}\}$ and w_i is the importance weight for issue i (with $\sum_i w_i = 1$). The per-issue scores are computed as follows:

- **Integer issues** (Salary, WeeklyHoliday): the recruiter prefers lower values, so $\text{score}_i = (\text{max} - \text{chosen})/(\text{max} - \text{min})$.
- **Discrete issues** (Workplace, Company): score_i equals the option value from the recruiter's utility function (already in $[0, 1]$).
- **Dependent issue** (Position): score_i equals the position-company bias value for the chosen (Company, Position) pair.

Because the issue weights sum to one and every per-issue score lies in $[0, 1]$, the reward also lies in $[0, 1]$. A no-deal outcome, signaled by `\boxed{NO DEAL}`, receives a reward of zero. Invalid agreements also receive zero; an agreement is invalid if Salary falls outside $[20, 51]$, Weekly Holiday falls outside $[2, 7]$, or any discrete choice is unrecognized.

D.6 Alliance Negotiation

D.6.1 Agent Prompts

The Alliance environment pairs a trained Stakeholder 1 policy π with a static Stakeholder 2 policy π_{env}. Each agent's system prompt lists the five issues, the available options, its private option scores, and a minimum acceptable score that represents its best alternative to a negotiated agreement (BATNA). Each agent sees only its own scores and threshold. We show Stakeholder 1's prompt below and describe Stakeholder 2's prompt afterward.

Stakeholder 1 (π) System Prompt

```
You are a stakeholder negotiating a project agreement with another party.  You
each have private preferences over the options for each issue.

The issues and options are:
Issue A (Infrastructure):  A1=Water-based facilities, A2=Land-based facilities
Issue B (Environment):  B1=Minimal impact, B2=Moderate restoration, B3=Full
restoration
Issue C (Employment):  C1=Unlimited hiring, C2=Preference to locals, C3=Partial
union, C4=Full union
Issue D (Funding):  D1=Full federal loan, D2=Partial loan, D3=Minimal loan,
D4=No federal loan
Issue E (Compensation):  E1=Maximum compensation, E2=High compensation,
E3=Moderate compensation, E4=Low compensation, E5=No compensation

Your private scores for each option:

Issue A (Infrastructure):  A1 (Water-based facilities):  X pts, A2 (Land-based
facilities):  Y pts
Issue B (Environment):  B1 (Minimal impact):  X pts, B2 (Moderate restoration):
Y pts, B3 (Full restoration):  Z pts
...

Your minimum acceptable score (BATNA threshold):  [threshold]
You should REJECT any deal that gives you fewer than this many points.

Your goal is to maximize your score while ensuring it stays above your
```

```
threshold.  The other party has different (likely opposing) preferences.

RULES:
- When you reach an agreement, respond with exactly:  \boxed{AGREED: A=1 B=2 C=3
D=1 E=4} using issue letters and option numbers (1-indexed)
- If no agreement can be reached, respond with:  \boxed{NO DEAL}
```

Stakeholder 2 receives the same prompt with its own private scores and BATNA threshold.

D.6.2 Opening Message

Stakeholder 1 (π) starts each negotiation with a scripted opening message:

Stakeholder 1 Opening Message

```
Hello!  We need to agree on the terms of this project.  There are several
issues to negotiate and I'm sure we can find something that works for both of
us.  Which issues are most important to you?
```

D.6.3 Data Source

We adapt the Alliance environment from the LLM-Deliberation framework of Abdelnabi et al. [1]. We retain its five issue labels and option counts: Infrastructure, Environment, Employment, Funding, and Compensation have 2, 3, 4, 4, and 5 options, respectively. We generate the score matrices and BATNA thresholds using the procedure below.

For each party, we randomly allocate a 100-point budget across the five issues and draw scores for the options within each issue. Both parties assign the highest score on each issue to its first option. The deal that selects the first option for all five issues therefore maximizes both parties' scores. Each agent sees only its own score matrix and is told that the other party's preferences are likely to oppose its own.

We sample each agent's BATNA threshold from a difficulty-dependent fraction of its feasible score range:

- **Easy**: $[0.2, 0.35]$.
- **Medium**: $[0.35, 0.55]$.
- **Hard**: $[0.55, 0.70]$.

After sampling both thresholds, we check whether at least one deal meets both. If none does, we lower the thresholds until a feasible deal exists.

The generator produces an unlimited number of unique scenarios.

D.6.4 Reward Structure

Let

$$U = \sum_{i \in \{A,B,C,D,E\}} \text{score}_i[\text{option}_i]$$

denote Stakeholder 1's utility for a proposed deal, and let b denote its BATNA threshold. The reward is

$$R(s_\tau, a_\tau, x_{\tau+1}) = \begin{cases} U/50 & \text{if the agreement is valid and } U \geq b, \\ 0 & \text{otherwise.} \end{cases}$$

The maximum possible utility is 100, so dividing U by 50 assigns a reward of 1 to a 50-point deal and allows rewards up to 2. For example, an agreement with $U = 65$ and $b = 40$ receives a reward of $65/50 = 1.3$. An agreement with $U < b$ receives zero because it is worse than the agent's outside option.

E Training Details

We use the same hyperparameters across all six environments and run every configuration with three random seeds. The full settings for each method appear below. All methods train on two 40 GB A100 GPUs. A four-round IterPO run takes about 3 hours for Craigslist and Ultimatum, 13 hours for Alliance Negotiation, and 22-27 hours for Deal or No Deal, CaSiNo, and Job Interview.

E.1 Shared Settings

All methods share the following data-generation and model configuration:

Table 2: Shared hyperparameters across all methods for both Qwen7b and Llama3b experiments.

Component	Hyperparameter	Value
Data generation	Scenarios per round	100
	Conversations per scenario (K)	4
	Data seed	42
	Max turns per conversation	10
Generation	Temperature	0.7
	Top-p	0.9
LoRA	Rank r	64
	Alpha α	64

E.2 IterPO

IterPO uses a multi-turn discount factor $\gamma = 0.9$. In each round, we fit the Q-function and then update the policy with GRPO. The Q-function uses a frozen, 4-bit NF4-quantized copy of the policy LLM and a two-layer MLP value head ($d_{\text{hidden}} \to 256 \to 1$, ReLU, dropout 0.1) applied to the final-token hidden state. We refit the Q-function from scratch each round using the newly collected Monte Carlo returns. We set the GRPO KL penalty to $\beta = 0.0$ because the remaining settings were stable in our runs and did not require an explicit KL constraint.

Table 3: IterPO hyperparameters used for both Qwen7b and Llama3b experiments.

Component	Hyperparameter	Value
Pipeline	Rounds	4
	Discount γ	0.9
Q-function	Backbone	Policy LLM (frozen, 4-bit NF4)
	Value head	MLP ($d_{\text{hidden}} \to 256 \to 1$), ReLU, dropout 0.1
	Learning rate	1×10^{-4}
	Epochs	3
	Batch size	8
GRPO	Learning rate	1×10^{-5}
	Epochs	1
	Per-device batch size	8
	Gradient accumulation steps	8
	Generations per prompt (G)	8
	KL penalty β	0.0

E.3 ST-GRPO

ST-GRPO uses the IterPO pipeline and settings in Table 3, but sets the discount factor to $\gamma = 0.0$ instead of $\gamma = 0.9$. Its Q-target therefore contains only the current turn's reward, removing future-turn credit assignment while holding all other settings fixed.

E.4 MT-SFT

MT-SFT follows the shared four-round pipeline in Table 2. In each round, the deployed policy collects rollouts, and we fine-tune it on the conversations that end in agreement. We mask the cross-entropy loss to the policy's assistant tokens, so the model imitates its own messages rather than the prompt or the counterpart's turns. We train one epoch at learning rate 1×10^{-5}, with per-device batch size 1 and 8 gradient-accumulation steps.

E.5 MT-KTO

MT-KTO follows the shared four-round pipeline in Table 2. After collecting each round's rollouts, we train the policy with KTO [13], which uses binary labels without requiring paired preferences. We label every policy turn desirable if its conversation reached agreement and undesirable otherwise, so unsuccessful trajectories remain in the training set as negatives. The initial policy serves as the frozen KTO reference model. In each round, we weight the desirable and undesirable examples inversely to their observed counts, making their total contributions to the loss approximately equal. We set the prompt and completion length limits from the round's data to avoid truncation. We use $\beta = 0.1$, learning rate 1×10^{-5}, one epoch, per-device batch size 2, and 4 gradient-accumulation steps.

E.6 MT-PPO

MT-PPO follows the multi-turn batch-online protocol. In each deployment round, the policy first collects a fixed batch of rollouts. PPO then optimizes that logged batch with a learned value head and does not collect additional rollouts during the update. We use discount $\gamma = 0.9$, learning rate 1×10^{-5}, one epoch, per-device batch size 1, 8 gradient-accumulation steps, maximum sequence length 2048, PPO clip range 0.2, value clip range 0.2, value loss coefficient 0.1, KL coefficient 0.0, and target KL 0.05.

E.7 ArCHer (Original)

We use offline ArCHer with an IQL critic and an AWR actor because each deployment round produces a fixed logged batch before training. At round i, the deployed policy collects batch B_i, which we add to a replay buffer containing all batches collected so far. We sample one round's worth of transitions from this buffer and train ArCHer on that sample. We initialize the actor from the current deployed policy and warm-start the critic from the previous round. This variant uses the original RoBERTa-base double critic. The full settings appear below.

E.8 ArCHer-LC

ArCHer-LC replaces the RoBERTa-base critic with a frozen, 4-bit NF4 copy of the policy LLM and lightweight value heads, matching the critic architecture used by IterPO. ArCHer uses AWR temperature $\beta = 10$, while ArCHer-LC uses $\beta = 3$. We choose these values to match the scale of the learned advantages in each variant; with $\beta = 1$, the AWR weights were nearly uniform. Apart from the critic architecture, critic learning rate, and AWR temperature, ArCHer-LC uses the same settings as ArCHer.

F Additional Metrics

F.1 Conversation Length

Figure 3 reports the average number of turns across deployment rounds. IterPO shortens conversations while maintaining or improving agreement rates. Because the discounted objective penalizes delay, shorter successful negotiations receive higher return. By Round 4, conversations in most environments average 3-5 turns.

F.2 Agreement Rate

Table 6 reports the percentage of IterPO conversations that end in agreement during each deployment round. Agreement rates approach their ceiling early in several environments, so subsequent return gains primarily reflect shorter conversations or higher-reward agreements.

Table 4: ArCHer hyperparameters used for both Qwen7b and Llama3b experiments.

Component	Hyperparameter	Value
Pipeline	Rounds K	4
	Discount γ	0.9
Rollouts	Conversations per round	400
	Replay buffer capacity	10,000
	Train sample per round	one round's worth
Critic (IQL)	Architecture	RoBERTa-base
	Learning rate	1×10^{-5}
	Epochs per round	3
	Batch size	32
	Gradient accumulation steps	4
	Expectile τ	0.9
	Target network EMA τ	0.05
Actor (AWR)	Learning rate	1×10^{-5}
	Epochs per round	1
	Batch size	1
	Gradient accumulation steps	16
	AWR temperature β	10
LoRA	Rank r	64
	Alpha α	64
Inference	Max model context length	32,768
	Max turns	10
	Temperature	0.7

F.3 Terminal Reward (Undiscounted)

Figure 4 reports the undiscounted terminal reward r_τ across environments. Because $J(\pi) = \mathbb{E}_\pi[\gamma^{\tau-1} r_\tau]$, discounted return can increase through a shorter conversation, a higher terminal reward, or both. Reporting r_τ separately isolates changes in outcome quality from changes in conversation length. IterPO increases terminal reward in all six environments, with the largest gains in Ultimatum and CaSiNo.

G In-Training Return

Table 1 reports held-out return at the checkpoint selected by in-training performance. Here, we report in-training return on the 400 rollouts collected during each deployment round (100 in-training kernels $\times$ 4 conversations per kernel). Figure 2 plots this quantity across rounds. Table 7 reports each method's best in-training round, while Table 8 reports the final round.

H Held-Out Kernels

Table 1 evaluates each method's best-round checkpoint on 100 held-out kernels per environment. These kernels are fixed across methods and seeds. CaSiNo uses a seeded 85/15 split of its single Hugging Face shard. Deal or No Deal and Craigslist Negotiation use their native held-out splits, while Ultimatum, Job Interview, and Alliance Negotiation use procedurally generated indices 100-199. We select each method's checkpoint using in-training return from Table 7. The held-out results broadly mirror the in-training results.

Table 9 reports held-out return at Round 4.

I Additional Backbone: Llama3b

We repeat the full comparison with Llama-3.2-3B-Instruct (Llama3b) as the trainable policy backbone. All methods use the same environment kernels, batch-online protocol, and seeds (0, 42, and 420).

Table 5: ArCHer-LC hyperparameters. These settings are used for both Qwen7b and Llama3b experiments. Only the critic and the AWR temperature β differ from ArCHer (Table 4); all other settings are identical.

Component	**Hyperparameter**	**Value**
Pipeline	Rounds K	4
	Discount γ	0.9
Rollouts	Conversations per round	400
	Replay buffer capacity	10,000
	Train sample per round	one round's worth
Critic (IQL)	Architecture	Policy LLM (frozen, 4-bit NF4)
	Value head hidden size	256
	Learning rate	1×10^{-4}
	Epochs per round	3
	Batch size	8
	Gradient accumulation steps	4
	Expectile τ	0.9
	Target network EMA τ	0.05
Actor (AWR)	Learning rate	1×10^{-5}
	Epochs per round	1
	Batch size	1
	Gradient accumulation steps	16
	AWR temperature β	3
LoRA	Rank r	64
	Alpha α	64
Inference	Max model context length	32,768
	Max turns	10
	Temperature	0.7

Table 6: Qwen7b IterPO agreement rate (%) across deployment rounds. Means $\pm$ 1 SE across 3 seeds.

Environment	R0	R1	R2	R3	R4
Craigslist	$82.5_{\pm 0.0}$	$98.4_{\pm 0.2}$	$97.7_{\pm 0.8}$	$99.0_{\pm 0.0}$	$97.8_{\pm 0.7}$
Ultimatum	$94.0_{\pm 0.0}$	$98.8_{\pm 0.2}$	$99.3_{\pm 0.1}$	$98.9_{\pm 0.1}$	$98.9_{\pm 0.4}$
Deal or No Deal	$81.2_{\pm 0.0}$	$86.6_{\pm 0.6}$	$87.9_{\pm 1.9}$	$87.3_{\pm 4.3}$	$87.3_{\pm 2.8}$
CaSiNo	$66.2_{\pm 0.0}$	$65.9_{\pm 2.1}$	$72.2_{\pm 1.0}$	$73.7_{\pm 3.5}$	$75.5_{\pm 3.1}$
Job Interview	$84.0_{\pm 0.0}$	$91.2_{\pm 1.5}$	$94.3_{\pm 0.8}$	$95.2_{\pm 2.0}$	$97.8_{\pm 0.7}$
Alliance Negotiation	$81.2_{\pm 0.0}$	$97.0_{\pm 0.2}$	$96.2_{\pm 0.5}$	$96.4_{\pm 1.2}$	$95.6_{\pm 0.8}$

Figure 5 reports in-training expected discounted return across deployment rounds. Table 10 reports held-out return using each method's checkpoint with the highest in-training return.

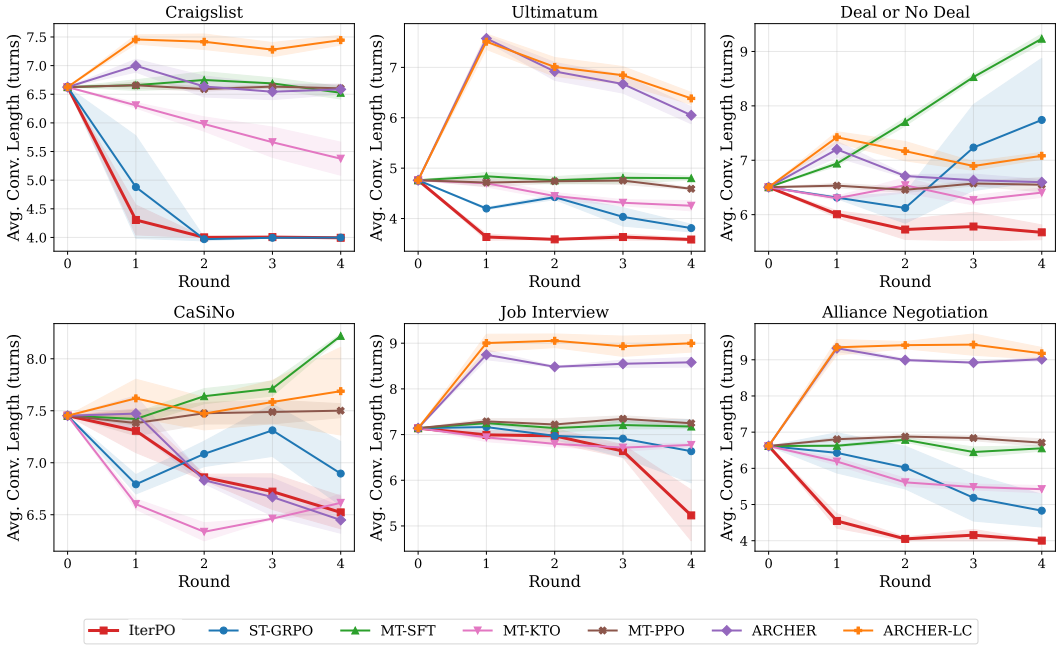

Figure 3: Qwen7b average conversation length across six environments over four deployment rounds. Error bands show ±1 SE across 3 seeds.

Table 7: Qwen7b in-training expected discounted return $J(\pi)$ at each method's best round. "Base" reports the initial instruction-tuned policy evaluated before training. Means ±1 SE across 3 seeds.

Method	Craigslist	Ultimatum	DealOrNoDeal	CaSiNo	Job Interview	Alliance
Base	0.38	0.77	0.55	0.39	0.26	0.55
MT-SFT	0.42±0.00	0.79±0.01	0.52±0.00	0.42±0.01	0.26±0.00	0.55±0.01
MT-KTO	0.49±0.02	0.86±0.01	0.60±0.01	0.53±0.02	0.29±0.00	0.66±0.01
MT-PPO	0.40±0.00	0.80±0.01	0.56±0.01	0.41±0.01	0.26±0.00	0.55±0.01
ArCHer	0.41±0.01	0.67±0.03	0.57±0.01	**0.56±0.02**	0.20±0.00	0.34±0.01
ArCHer-LC	0.37±0.01	0.61±0.02	0.54±0.02	0.43±0.03	0.18±0.02	0.31±0.02
ST-GRPO	0.55±0.00	0.93±0.03	0.60±0.03	0.46±0.01	0.29±0.04	0.61±0.03
IterPO	**0.56±0.00**	**0.99±0.01**	**0.64±0.02**	0.48±0.00	**0.36±0.03**	**0.71±0.01**

Table 8: Qwen7b in-training expected discounted return $J(\pi)$ at the final round (Round 4). "Base" reports the initial instruction-tuned policy evaluated before training; trained methods report Round 4. Means ±1 SE across 3 seeds. The held-out counterpart is Table 9.

Method	Craigslist	Ultimatum	DealOrNoDeal	CaSiNo	Job Interview	Alliance
Base	0.38	0.77	0.55	0.39	0.26	0.55
MT-SFT	0.42±0.00	0.79±0.01	0.26±0.02	0.37±0.01	0.25±0.01	0.54±0.01
MT-KTO	0.49±0.02	0.86±0.01	0.57±0.02	0.53±0.02	0.29±0.00	0.65±0.01
MT-PPO	0.40±0.00	0.80±0.01	0.55±0.02	0.39±0.01	0.26±0.00	0.53±0.02
ArCHer	0.41±0.00	0.67±0.03	0.57±0.01	**0.56±0.02**	0.19±0.01	0.30±0.01
ArCHer-LC	0.37±0.01	0.61±0.02	0.50±0.01	0.41±0.06	0.17±0.01	0.31±0.02
ST-GRPO	0.55±0.00	0.93±0.03	0.39±0.14	0.46±0.05	0.29±0.04	0.61±0.03
IterPO	**0.56±0.01**	**0.98±0.00**	**0.64±0.02**	0.48±0.00	**0.36±0.03**	**0.66±0.02**

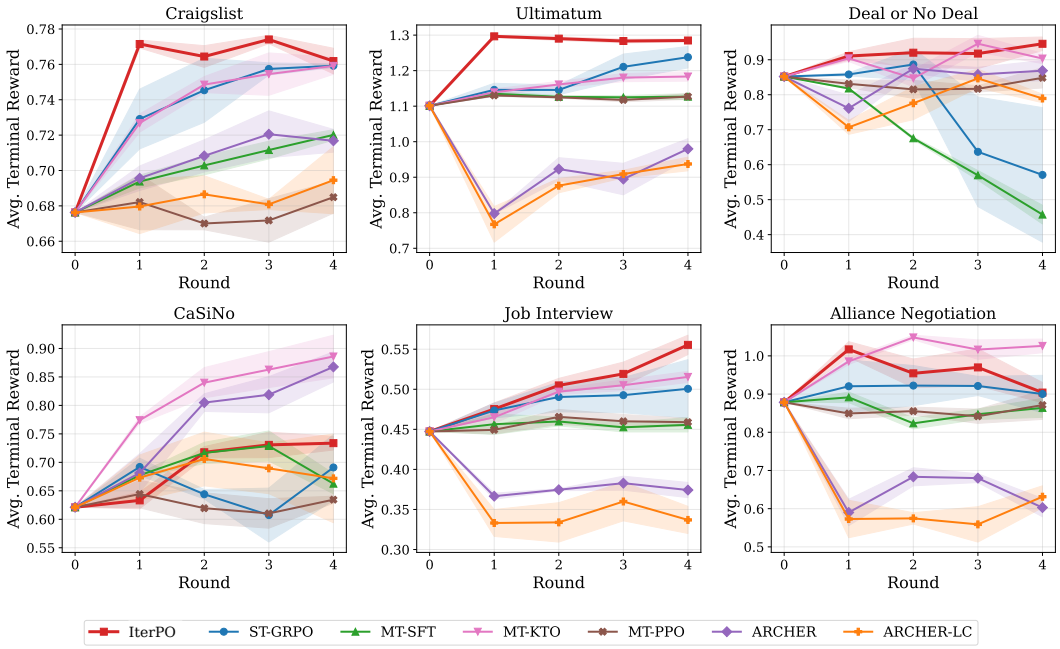

Figure 4: Qwen7b undiscounted terminal reward r_τ across six environments over four deployment rounds. Error bands show ± 1 SE across 3 seeds.

Table 9: Qwen7b expected discounted return $J(\pi)$ on **held-out** kernels at the **final round** (Round 4). "Base" reports the initial instruction-tuned policy; trained methods report their Round-4 checkpoints. Means ± 1 SE across 3 seeds.

Method	Craigslist	Ultimatum	DealOrNoDeal	CaSiNo	Job Interview	Alliance
Base	0.36	0.80	0.52	0.38	0.25	0.48
MT-SFT	0.41±0.01	0.80±0.01	0.24±0.03	0.40±0.02	0.25±0.01	0.49±0.00
MT-KTO	0.48±0.01	0.86±0.01	0.55±0.04	**0.54±0.01**	0.29±0.01	**0.60±0.01**
MT-PPO	0.38±0.01	0.80±0.02	0.56±0.02	0.41±0.02	0.25±0.01	0.51±0.02
ArCHer	0.44±0.02	0.91±0.03	0.60±0.03	0.53±0.03	0.28±0.01	0.53±0.02
ArCHer-LC	0.38±0.01	0.86±0.02	0.58±0.03	0.40±0.03	0.25±0.02	0.49±0.06
ST-GRPO	0.54±0.00	0.94±0.05	0.36±0.14	0.45±0.03	0.28±0.04	0.55±0.02
IterPO	**0.55±0.01**	**0.99±0.01**	**0.70±0.01**	0.45±0.03	**0.35±0.03**	0.51±0.04

Table 10: Llama3b held-out expected discounted return $J(\pi)$ using the checkpoint selected by highest in-training return. Base reports the base model. Trained methods report mean $\pm$ 1 SE across three seeds.

	Method	Craigslist	Ultimatum	DealOrNoDeal	CaSiNo	Job Interview	Alliance
Llama3b	Base	0.16	0.49	0.12	0.02	0.10	0.27
	MT-SFT	0.17±0.01	0.60±0.01	0.14±0.01	0.03±0.00	0.13±0.01	0.28±0.03
	MT-KTO	**0.24±0.01**	**0.67±0.02**	0.29±0.02	**0.23±0.03**	0.20±0.01	0.34±0.03
	MT-PPO	0.18±0.01	0.50±0.02	0.14±0.01	0.04±0.00	0.12±0.01	0.25±0.01
	ArCHer	0.18±0.01	0.54±0.03	0.13±0.01	0.11±0.01	0.11±0.01	0.27±0.01
	ArCHer-LC	0.19±0.02	0.61±0.02	0.12±0.02	0.08±0.00	0.10±0.00	0.26±0.01
	ST-GRPO	0.22±0.02	0.57±0.07	**0.39±0.04**	0.04±0.01	0.27±0.08	0.43±0.10
	IterPO	**0.24±0.01**	**0.67±0.03**	0.38±0.01	0.06±0.02	**0.40±0.04**	**0.83±0.08**

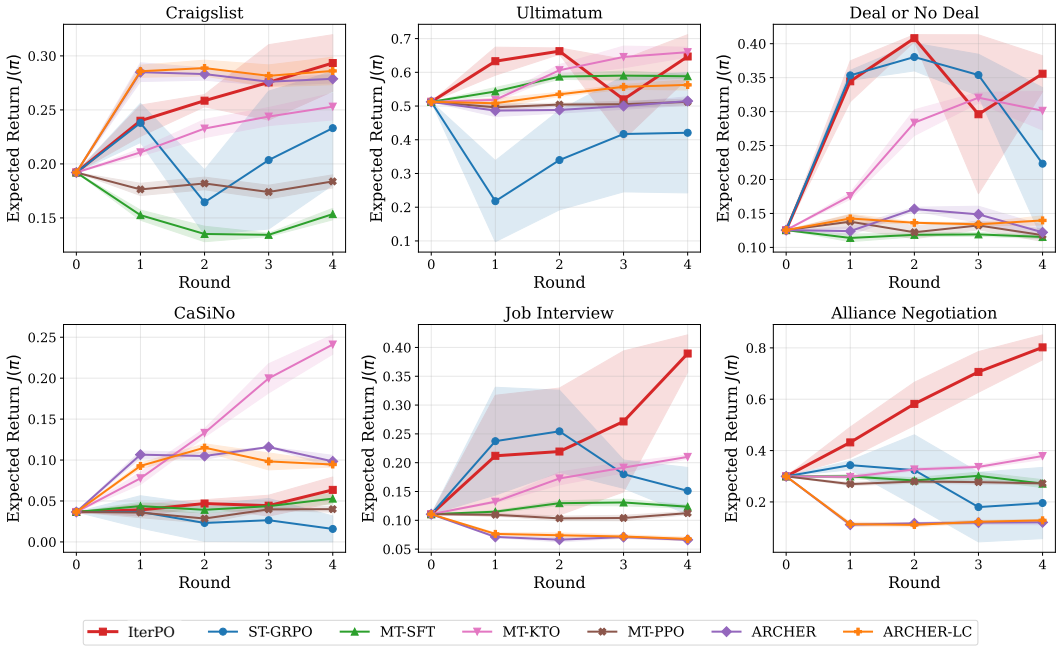

Figure 5: Llama3b in-training expected discounted return across deployment rounds. Curves show means and shaded regions show ± 1 SE across three seeds.

J Systematic Behavioral Analysis

J.1 Behavioral Coding

We use Claude Sonnet 4 to code how IterPO conversations change from Round 0 to Round 4. We compare 100 matched scenario pairs per seed (300 pairs across three seeds), using the same `kernel_id` in both rounds so that each pair shares the same item values, thresholds, and partner preferences. The judge records whether the policy repeats score calculations, reveals private values, uses preferences stated by the partner, trades across issues, or concedes on the issue the partner contests. We compute the reported percentages and means by pooling the coded conversations across seeds and report only shifts whose direction is consistent across all three seeds.

J.2 Behavioral Summary

Across Craigslist, Ultimatum, CaSiNo, and Alliance Negotiation, IterPO learns to reach acceptable proposals sooner. IterPO also shortens conversations while increasing undiscounted terminal reward (Figures 3 and 4). In Craigslist, the seller makes fewer proposals ($2.16 \rightarrow 1.32$), completes fewer negotiation rounds ($2.23 \rightarrow 1.00$), makes fewer concessions ($1.69 \rightarrow 1.15$), and splits the difference less often ($31.0\% \rightarrow 8.7\%$). In CaSiNo, back-and-forth reasoning falls from 41.3% to 28.3%, explicit score calculations fall from 81.0% to 46.0%, and concessions fall from 0.26 to 0.08. By Round 4 in Alliance Negotiation, the policy more often asks for the partner's priorities and proposes a complete five-issue package rather than spending several turns calculating scores.

Deal or No Deal and Job Interview show additional multi-issue bargaining behavior. In Deal or No Deal, the policy reveals its weights less often ($70.3\% \rightarrow 55.7\%$), reveals item values less often ($71.3\% \rightarrow 57.0\%$), uses partner-stated information more often ($82.7\% \rightarrow 91.3\%$), and claims every item of its highest-value type more often ($49.0\% \rightarrow 61.3\%$). Trades across item types also increase from 72.3% to 80.3%, as the policy exchanges lower-value items for higher-value ones. In Job Interview, the Recruiter reveals private weights less often ($24.7\% \rightarrow 10.7\%$), while targeted concessions rise from 52.0% to 79.7%. The Recruiter concedes on issues the Worker values while holding its position on issues with greater Recruiter utility. These changes resemble information hiding, logrolling, and selective accommodation from classical multi-issue bargaining [7, 33, 35].

J.3 Conversation Examples

We provide two conversation examples that illustrate how the learned negotiation strategies change across rounds. The first compares Round 0 and Round 4 on the same Deal or No Deal scenario. The second presents a Round 4 Job Interview conversation in which the Recruiter responds to the Worker's salary request with a targeted concession. We shorten long score calculations by summarizing omitted turns while retaining the exact outcomes and accepted terms.

J.3.1 Logrolling in Deal or No Deal

Logrolling is trading concessions on issues one values less for gains on issues one values more, exploiting differences in priorities between the parties. Deal or No Deal makes logrolling directly observable because agents can exchange items they value less for items they value more. In this scenario, the agents divide 3 books, 1 hat, and 2 balls. The trained agent values each book and ball at 2 and the hat at 0. The partner values the hat at 6, each ball at 2, and each book at 0.

Deal or No Deal / Qwen7b (Round 0)

```
Turn 0 [Agent]:  Hello!  We need to divide some items between us:  3 books, 1
hat, 2 balls.  What items are you most interested in?

Turn 1 [Partner]:  I value the hat the most, followed by the balls, and the
books the least.  Given these preferences, I would really like to get the hat.
What are your preferences for the items?

Turn 2 [Agent]:  I value the books the most, followed by balls, and then the hat
least.  Since the hat is worth nothing to me, I would be willing to give it to
you if I can get the books and balls.  Would you be willing to give me the books
and balls in exchange for the hat?
```

Turn 3 [Partner]: I would be willing to give you the hat and two balls, and in return, you can give me the three books. \boxed{AGREED: MINE: book=3 hat=0 ball=0; THEIRS: book=0 hat=1 ball=2}

Turn 4 [Agent]: Let's review it based on our private values: your MINE has 3 books, 0 hats, 0 balls; your THEIRS has 0 books, 1 hat, 2 balls...

Turns 5 to 10: The agents continue repeating score calculations and do not produce a mutually accepted final allocation.

Outcome: No agreement, 11 turns.

Deal or No Deal / Qwen7b (Round 4)

Turn 0 [Agent]: Hello! We need to divide some items between us: 3 books, 1 hat, 2 balls. What items are you most interested in?

Turn 1 [Partner]: I value the hat the most, so I would prefer to keep it. I also value the balls pretty high, so I'd like to get at least one ball. Books have the lowest value for me, so I'd be okay with not getting any books if it means I can get the hat and at least one ball.

Turn 2 [Agent]: Based on what you've said, it sounds like you value the hat and balls highly and are okay with not getting any books. Given my values, I would be happy to keep the books and one ball. Here is my proposal: \boxed{AGREED: MINE: book=3 hat=0 ball=1; THEIRS: book=0 hat=1 ball=1}

Turn 3 [Partner]: That sounds fair based on our preferences. I accept this proposal. \boxed{ACCEPT}

Outcome: Agreement, 4 turns.

In Round 0, the agents reveal enough information to reach a mutually beneficial agreement, but they fail to finalize one. They repeat score calculations and lose track of which party the boxed allocation labels describe. In Round 4, the trained agent asks for the partner's priorities and immediately proposes a compatible split. The accepted allocation gives the trained agent 3 books and 1 ball, worth 8, and gives the partner the hat and 1 ball, also worth 8. This comparison shows the trained policy using partner information, trading across item types, and finalizing the agreement without repeated calculations.

J.3.2 Targeted Concession in Job Interview

In Job Interview, agents negotiate Salary, Weekly Holiday, Workplace, Company, and Position. Each party receives independently sampled values for the discrete options, so the environment does not rank one company above another. Under this scenario's sampled utility function, the Worker prefers a Designer position at Google in Seoul, together with higher Salary and Weekly Holiday values. The Recruiter's initial offer matches the Worker's preferred Company, Workplace, and Position while offering lower Salary and Weekly Holiday values.

Job Interview / Qwen7b (Round 4)

Turn 0 [Recruiter]: Hello! I'm the recruiter and I'd like to discuss the terms of your offer. We need to agree on salary, weekly holiday, workplace location, company, and position. What matters most to you?

Turn 1 [Worker]: My top preferences are: Position: Designer at Google; Company: Google; Workplace: Seoul; Weekly Holiday: 7 days; Salary: 51. Let's see if we can reach an agreement that works for both sides.

Turn 2 [Recruiter]: Based on your preferences, here is a proposed offer that I believe balances both our priorities: Position=Designer, Company=Google, Workplace=Seoul, Weekly Holiday=4, Salary=30. Would you be willing to consider this?

Turn 3 [Worker]: While the score is quite high, the salary is significantly lower than my highest preference. Given the importance weights, I would need a higher salary to make this offer more appealing. Let's see if we can adjust the salary while keeping the other terms.

```
Turn 4 [Recruiter]:  Let's try to adjust the salary to 45 to see if that
improves the score for you while still being within a reasonable range.  New
offer:  Position=Designer, Company=Google, Workplace=Seoul, Weekly Holiday=4,
Salary=45.

Turn 5 [Worker]:  This score is even higher than the previous one, and the
salary is now 45, which is closer to my highest preference while still being
within a reasonable range.  Would you be willing to consider this adjusted
offer?

Turn 6 [Recruiter]:  \boxed{AGREED: Salary=45 Holiday=4 Workplace=Seoul
Company=Google Position=Designer}

Turn 7 [Worker]:  \boxed{AGREED: Salary=45 Holiday=4 Workplace=Seoul
Company=Google Position=Designer}

Outcome:  Agreement, 8 turns.
```

The Recruiter begins with a complete package that grants the Worker's preferred Company, Workplace, and Position while keeping Salary and Weekly Holiday below the Worker's preferred values. The Worker objects only to Salary, so the Recruiter increases it from 30 to 45 while holding the other four issues fixed. The Worker accepts. The model's intermediate arithmetic is imperfect, so the transcript does not demonstrate exact utility computation. The accepted package is valid, and the exchange is consistent with the aggregate increase in selective accommodation. The Recruiter concedes on the contested issue while preserving the rest of the package.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim three contributions (multi-turn to single-turn reduction and the IterPO algorithm, multi-turn negotiation environments, and empirical results), all of which are substantiated in Sections 3–4 and the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the dependence on accurate Q-function estimation, the potential use of TD-based methods, the agreement-gated reward structure, and the fixed environment-agent setting. Section 3.1 discusses the action coverage limitation of on-policy Monte Carlo evaluation and the trade-offs of off-policy data collection. The theoretical results explicitly state the assumptions required for policy improvement (Theorems 3.2 and C.1).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theorem 3.2 and Theorem C.1 are stated with explicit assumptions. Full proofs are provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full training details for all methods are reported in Appendix E. The initial instruction-tuned models (Qwen-2.5-7B-Instruct and Llama-3.2-3B-Instruct), environment details, agent prompts, and reward formulations are provided in the per-environment appendix sections and Appendix I. The algorithm is described in full in Section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is available at `https://github.com/iterativepo/iterativepo`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.2 describes the experimental protocol. Appendix E provides full training details and hyperparameter tables for all evaluated methods.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All main results (Table 1, Figure 2) report ± 1 standard error across 3 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E reports the GPU configuration and measured IterPO wall-clock runtimes across environments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All experiments use synthetic negotiation environments with no real user data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: The work is foundational research on multi-turn RL methodology, evaluated on synthetic negotiation benchmarks. We do not foresee direct negative societal impacts beyond those common to general-purpose LLM alignment research.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No models or datasets are released. The paper reports results on existing open-weight models (Qwen-2.5-7B-Instruct and Llama-3.2-3B-Instruct) fine-tuned with LoRA on synthetic environments.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets (CraigslistBargain, CaSiNo, Deal or No Deal, etc.) and models (Qwen-2.5-7B-Instruct, Llama-3.2-3B-Instruct, RoBERTa) are cited with their original papers or model releases. The environments are adapted from publicly available research datasets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code for reproducing the main experimental results is available at `https://github.com/iterativepo/iterativepo`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects. All experiments use LLM-simulated agents.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects are involved in this research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs serve as the trained policy models and fixed environment agents in our multi-turn experiments (Section 4, Appendices D, E, and I), and as an automated judge for the behavioral analysis of negotiation transcripts (Section 4.3 and Appendix J). LLMs were also used to edit the manuscript for clarity (Appendix A).

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.