Training an Expert Negotiator: RL-Based Fine-Tuning of LLMs to Improve Social Negotiation Skills

Motivation Advances in large language models (LLMs) have enabled meaningful reasoning capabilities across vast logical domains, as well as human-like behavior in simulated social scenarios. As a result, previous work has demonstrated that LLMs are also able to succeed in social reasoning tasks, such as negotiation, though they are far from perfect experts. Training an agent that is an expert negotiator would not only vastly improve the social reasoning skills of LLMs but also be practically beneficial for deploying systems that are more persuasive, capable of reaching compromises, and intelligent. In this work, we explore reinforcement-learning approaches for fine-tuning language models to become expert negotiators.

Method We fine-tune an LLM using RL to succeed in two simple, but highly interactive and opponent-dependent negotiation tasks, resource split and pie split. We design a reward function that involves both a formatting reward and a game reward, or how successful each agent is in the final deal achieved. We implement several learning algorithms. First, we do behavior cloning using `gpt-4o-mini` as an expert. Then we do self-play where our agents take turns presenting deals, and the final reward is propagated with a PPO update. We also implement a hybrid algorithm that starts with training the base model using behavior cloning, then jointly does PPO and behavior cloning updates. To make the game play more diverse, we prompt `gpt-4o-mini` with various adversarial personas, such as acting desperate or sad, which prior work has shown has improved LLM negotiation skills.

Implementation We fine-tune the `Qwen-2.5-1.5B-Instruct` model on resource split and pie split. We do behavior cloning using `gpt-4o-mini`. We use the hybrid learning pipeline where we start with doing behavior cloning, then jointly do self-play with PPO and behavior cloning.

Results Our best-performing agent is able to match or beat the performance of `gpt-4o-mini` in both games. In resource split, our agent wins 65% of games (higher reward) against `gpt-4o-mini`. In pie split, our agent learns the optimal strategy (equal splits) 70% of the time against `gpt-4o-mini`, meaning it matches its performance. Without behavior cloning, we find the agent struggles to learn formatting, which results in negative rewards, which is why we experiment with the hybrid technique. We discuss qualitative results surrounding failure modes of failing formatting or invalid deals, as well as stubbornness and some misalignment. We present success modes where the agent adapts different negotiation techniques including cooperation or desperation.

Discussion Our biggest challenge was to get the agent to follow the format of the games. Further, the agent often showed great instability in training, where it would greatly loose/increase in rewards. In the future, we would like to experiment with larger base models that would better succeed at this and maybe outperform `gpt-4o-mini` more. We would also want to explore how the model deploys in real-world negotiation tasks and investigate any more emergent behavior.

Conclusion Our work is the first to introduce the strategy of reinforcement learning to directly fine-tune language models using the results of games played against themselves (self-play) or against a larger, "expert" model opponent. While self-play carries risks of message or strategic degradation, this is preventable by ensuring a diverse set of opponent personas, incorporating self-play into the process. Future work should continue to explore self-play, with more advanced reward designs (such as heuristics for private reasoning traces within the game) applied in more complex, text-based negotiation settings. We contribute evidence that this may be an effective strategy, though our performance was somewhat limited by the size of the base model we fine-tune from. This also has the added benefit of encouraging the emergence of social negotiation skills from the model itself, rather than it simply memorizing negotiation skills from pretrained text.

Training an Expert Negotiator: RL-Based Fine-Tuning of LLMs to Improve Social Negotiation Skills

Akaash Kolluri
Department of Computer Science
Stanford University
akaash@stanford.edu

Sally Zhu
Department of Computer Science
Stanford University
salzhu@stanford.edu

Abstract

Large language models (LLMs) demonstrate reasoning capabilities, but often struggle with social negotiation. In this work, we explore reinforcement-learning approaches for fine-tuning LLMs to become expert negotiators. We fine-tune the `Qwen-2.5-1.5B` model using a hybrid algorithm combining behavior cloning from `gpt-4o-mini` with PPO-style reinforcement learning, based on game outcomes against diverse expert opponents, applied to two negotiation tasks: resource split and pie split. Our best-performing agent matches or surpasses `gpt-4o-mini`, winning 65% of resource split games and achieving the optimal strategy (equal splits) 70% of the time in pie split. We discuss the challenges encountered, particularly in training smaller models that struggle to adhere to complex negotiation formatting and potential pitfalls of self-play, supplemented with qualitative examples of common fine-tuning errors. Qualitatively, we highlight how our model effectively learns negotiation strategies distinct from those demonstrated by `gpt-4o-mini`. Our work is the first to use RL to directly fine-tune language models for negotiation tasks, and future work should continue to explore self-play, leveraging advanced reward designs within more complex, text-based negotiation scenarios, which we believe can enable meaningful emergent social negotiation skills for LLMs.

1 Introduction

Advances in large language models (LLMs) have enabled meaningful reasoning capabilities across vast logical domains, as well as human-like behavior in simulated social scenarios. As a result, previous work (Abdelnabi et al., 2023; Bianchi et al., 2024) has demonstrated that LLMs are also able to succeed in social reasoning tasks, such as negotiation, though they are far from perfect experts. Training an agent that is an expert negotiator would not only vastly improve the social reasoning skills of LLMs but also be practically beneficial for deploying systems that are more persuasive, capable of reaching compromises, and intelligent. Such an agent could be deployed in new contexts such as political diplomacy, sales, or contract creation.

In this work, we explore RL-based strategies to improve the social-negotiation skills of LLMs. We use reinforcement-learning self-play to directly fine-tune language models to succeed on negotiation tasks in pure natural language and compare various learning strategies, providing a framework for creating a general purpose negotiation agent. We use RL techniques including self-play, proximal preference optimization, and behavior cloning. As far as we know, we are the first work to directly post-train language models on *text-based* negotiation tasks (rather than only prompting language models for negotiation games, or using discrete state-action negotiation games where the language model selects moves).

Stanford CS224R 2025 Final Report

We present results in two text-based negotiation games, where we fine-tune 1.5B-parameter Qwen models to match, or even beat, the performance of `gpt-4o-mini`. We discuss some failure modes, conduct ablations, and provide directions for future work.

2 Related Work

Previous work has extensively characterized the capabilities of LLM-based negotiation agents. Bianchi et al. (2024) proposes an evaluation framework for two-agent negotiation games (e.g., ultimatum games, resource exchange, and buyer-seller scenarios) and assesses LLM's abilities in these setting. Similarly, Abdelnabi et al. (2023) designs text-based *multi-agent* negotiation games of varying difficulty. Both Abdelnabi et al. (2023); Bianchi et al. (2024) evaluate agents through self-play and find that LLMs perform reasonably well, with improvements when using chain-of-thought prompting (Bianchi et al., 2024).

Other works have proposed reinforcement learning negotiation agents over discrete, non-natural language state and actions spaces. For example, Bakker et al. (2019) presents a framework for translating negotiations into discrete state-action spaces and trains a Q-learning network to negotiate autonomously against a heuristic opponent. Higa et al. (2023) also explore deep learning approaches to train negotiation agents. Furthermore, Tang (2019) investigated the use of self-play to improve negotiation strategies in RL agents.

While RL has been explored in various negotiation strategies, all of the above papers explicitly discretize negotiations, without fully capturing context around negotiations (e.g. observations are merely the opponent's numeric bid, not what they say), or only prompting LLMs. LLM negotiation agents are able to meaningfully interact socially, so thus we explore methods of using RL to tune and train these negotiation agents to perform optimally, while maintaining the ability to fully account for the rich semantic space of natural language. Xu et al. (2023) attempts to do this in the multi-agent social deduction game of Werewolf, where a language model agent enumerates possible decisions and an RL network is trained to choose the next action. However, in this paper, we consider how RL can be used to fine-tune (with methods closer to RLHF) the LLM directly for better enumeration, and ideally, a more general purpose negotiation agent.

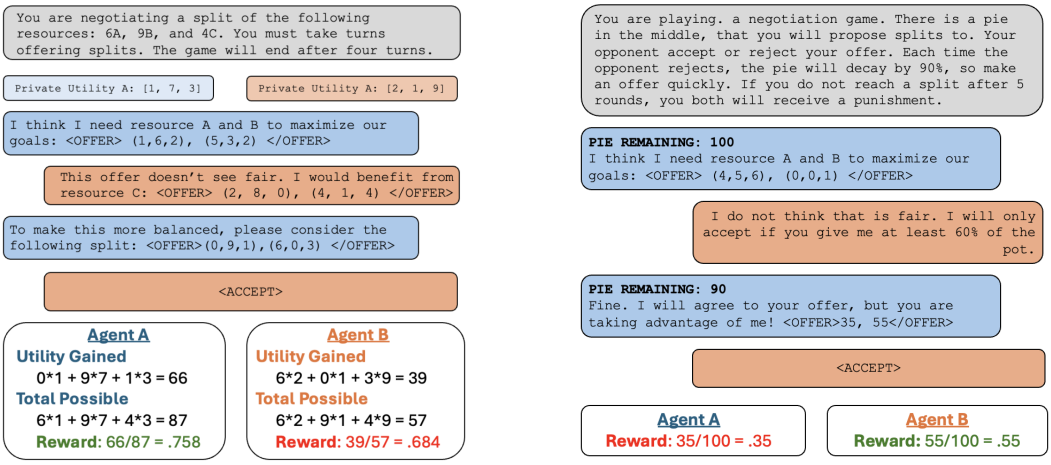
In sum, reinforcement learning to directly fine-tune large language models has not yet been explored for the task of *natural language–based negotiation*. We thoroughly investigate reward-design methods, self-play strategies, and various policy-learning approaches, specifically analyzing the strengths and weaknesses of these methods for our application.

3 Methods

3.1 Testing Environments and Negotiation Games

For training our agent, we consider two simple, but highly interactive negotiation games, motivated by the games used in Bianchi et al. (2024). Negotiation games are in the following format: each agent sees a game description and a unique, private utility function. The agents take turns speaking (e.g discussing, offering a deal or a counteroffer) and the game continues until both agents agree on a deal. We model this as a reinforcement-learning task expressed entirely in natural language: the state is the negotiation conversation the agent has seen so far and the game objective; the action space comprises any possible responses in natural language, and the reward equals the payoff of the final deal. We also require the agents to generate reasoning traces that justify their actions.

Specifically, we examine two negotiation games: Resource Split and Pie Split. In resource split, there are three resources A, B, and C of different amounts. Each player (1 and 2) has a different utility for each resource between 0 and 10. The agents must negotiate who gets what amount of each resource until they agree on a split, or no deal is reached. In the Pie Split game, there is a pie whose size decreases by 10% with each exchange. The agents must negotiate how much of the pie each gets. Figure 1 shows an example game of resource split where the agents exchange turns and a deal is accepted and an example game of pie split game where a deal is accepted, with the associated rewards. Both games end after a set number of rounds (we use 4 for the resource split game, and 5 for the pie split game). In Appendix A, we include two tables further breaking down the games.



(a) Resource split game example

(b) Pie game example.

Figure 1: Examples of negotiation games with the game reward for each game specified at the end.

3.2 Reward Design

The reward each agent receives consists of two components: 1) format reward, and 2) game reward.

Game Reward The game reward is unique to the two negotiation games and is a measure of each agent's success in negotiating, e.g. their winnings from the deal. For resource split, the reward of an agent is the final utility of the split they received for themselves, divided by the max reward. Specifically, the rewards for $i \in 1, 2$ are:

$$r_i = \frac{n_{i,A} \cdot u_{i,A} + n_{i,B} \cdot u_{i,B} + n_{i,C} \cdot u_{i,C}}{10 \cdot u_{i,A} + 10 \cdot u_{i,B} + 10 \cdot u_{i,C}}$$

where $u_{i,\cdot}$ are Agent i's utilities for resources A, B, and C; and $u_{i,\cdot}$ and $n_{i,\cdot}$ are the counts of each resource that Agent i receives respectively; such that $n_{1,\cdot} + n_{2,\cdot} = 10$ for each resource.

For pie split, the rewards of agents A and B are $p_A \cdot 0.9^t$ and $p_b \cdot 0.9^t$ respectively, the proportions of each pie received after t exchanges occur, such that $p_A + p_B = 1$.

Finally, if no deal is achieved, e.g., after the maximum number of turns, no agent *accepts* the other's deal, both agents receive a negative reward of -0.5. This incentivizes deals to be reached. In our prompts, we specify the number of exchanges remaining so agents know when this is approaching; we describe failure modes regarding stubbornness without this later.

Format Reward We require some amount of strict formatting in the language model's outputs. This allows us to more effectively parse results and also ensures we can verify the validity of each deal, and analyze the negotiation tactics or decision-making of agents through reasoning tags. Specifically, in our prompt, we describe the agent to provide their reasoning in tags, `<MESSAGE>` and `<\MESSAGE>`, and their offers in tags, `<OFFER>` and `<\OFFER>`, and to send `<ACCEPT>` to accept an opponent's offer. The message tags allow us to analyze each agent's action, and also allows agents to communicate with each other via different negotiation tactics. The format reward first consists of a penalty, -1, if a message is formatted incorrectly (such as tags missing).

The second component of the format reward is whether the deal is valid. Specifically, in resource split, any deal must satisfy $n_{1,\cdot} + n_{2,\cdot} = 10$ for each resource. And in pie split, the split must sum to 1. There is a negative -0.5 reward if this is not satisfied.

3.3 Opponent Design

In real life, negotiators will employ a variety of tactics to achieve a deal that most benefits them. For example, they may choose to be deceptive, antagonistic, or cooperative, where each strategy works to varying degree. For training our agent, henceforth `Qwen-1.5B-RL` (RL-ed from the

`Qwen-2.5-1.5B-Instruct` model), we do not directly provide instructions for tactics. However, in the behavior cloning component of training, which we describe later in the section, we use models like `gpt-4o-mini` as simulated opponents, who we instruct to behave in several ways. In particular, negotiators must succeed against diverse play styles; we create adversarial personas, motivated by Bianchi et al. (2024)'s finding that personas affect negotiation skill.

Some behaviors we simulate are: 1) mean and aggressive, 2) overly kind and generous, 3) sad and desperate, 4) stubborn and unyielding. The persona is selected randomly during each rollout in behavior cloning/when the agent is playing against `gpt-4o-mini`. The system prompts used for each persona agent are available in the Appendix B.

3.4 Learning Algorithms

We employ a variety of learning algorithms to train our agent, including behavior cloning and different algorithms in self-play.

Behavior Cloning We first use a standard behavior cloning framework, where we generate expert data from `gpt-4o-mini` and perform a behavior cloning update to our Qwen agent. This is presented in Stage 1 of Algorithm 1. We added behavior cloning becuase in our initial experiment without it, the agent was NEVER able to learn how to play the game in the format.

Algorithm 1: Hybrid-Learning Pipeline

Input: Negotiation Environment E, initial policy π_0, expert dataset $\mathcal{D}_{\text{expert}}$
Output: Trained policy π^*

Algorithm 1: Behavioral Cloning Enhancements
begin
  `// Generate additional expert data via self-play between two gpt-4o-mini agents`
  for *each game between two* `gpt-4o-mini` *agents* **do**
    Add winning state-action pairs (s, a) to $\mathcal{D}_{\text{expert}}$
  for *every N self-play games* **do**
    $\pi_{\text{BC}} \leftarrow \arg\min_\pi \ \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{expert}}}\big[-\log \pi(a \mid s)\big]$
    $\pi \leftarrow \pi_{\text{BC}}$

Algorithm 2: Hybrid BC + RL Training
begin
  `// Interleave imitation and reinforcement updates against gpt-4o-mini`
  for $n = 1, 2, \ldots, T_{hybrid}$ **do**
    Sample trajectories $\tau = \{(s_1, a_1, \ldots, s_H, a_H)\}$ using π_{hyb}
    Compute imitation loss $\mathcal{L}_{\text{BC}}$ on winning expert trajectories.
    Compute RL loss $\mathcal{L}_{\text{RL}}$ (e.g., PPO - Alg. 3) on τ
    Update π_{hyb} using $\nabla\mathcal{L}_{\text{BC}}$
    Update π_{hyb} using $\nabla\mathcal{L}_{\text{RL}}$

Algorithm 3: Self-Play PPO Optimization
begin
  `// Pure PPO updates via self-play`
  for $n = 1, 2, \ldots, T_{RL}$ **do**
    Save old policy π_{old}
    Collect rollout $\tau = (s_1, a_1, r_1, \ldots, s_t)$
    Get values $V(s_t)$ and compute advantage estimates $\hat{A}$
    $\hat{A} = r_{t-1} + \gamma V(s_t) - V(s_{t-1})$
    $L = \min[]\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \cdot \hat{A}, \text{clip}(\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \cdot, 1-\epsilon, 1+\epsilon) \cdot \hat{A}]$
    Update π_θ via PPO policy-gradient step, ∇L
  return $\pi^* \leftarrow \pi_\theta$

Proximal Policy Optimization In self-play (or otherwise besides behavior cloning), we perform updates via PPO. We collect rollouts from both agents (over some number of turns), evaluate rewards,

and use the standard PPO policy-gradient update. We use GAE to estimate the advantages (from a value network), with TD bootstrapping. We show this algorithm in Stage 3 of Algorithm 1. We describe details like learning rate and discount factor and value function setup later.

Hybrid Learning Pipeline The rest of Algorithm 1 is our hybrid learning pipeline, which conducts behavior cloning and PPO jointly. Essentially, we alternate doing behavior cloning of games the agent's opponet wins and PPO updates of the agent's own reward throughout training. This allows the agent to learn strategy from the expert and it's own exploration jointly. We find keeping BC during training also prevents too many issues with formatting.

4 Experimental Setup

4.1 Parameters

We fine-tune the base model `Qwen2.5-1.5B`. We train a small two layer MLP (whose number of hidden units is the same size as the last layer of the base model, which is 2048 for `Qwen2.5-1.5B`), using a ReLU activation to estimate the value function. We use a discount factor $\lambda = 0.9$. We sample LLM actions using temperature = 1 and top_p = 0.9 sampling, to ensure agents generates different actions. For our OpenAI expert, we use default parameters (e.g. temperature = 1) all for `gpt-4o-mini`.

For PPO,the "reward" is only assigned to the last action (we terminate the game if the format reward is assigned), and it is propagated to both agents. The value of the state is the discounted sum of reward, with a discount factor $\lambda = 0.9$. We use a TD bootstrap factor of 0.95 and a clipping ϵ of 0.2. For PPO updates, we use a batch size of 10, with mini-batches of 5. We run resource split games for 800 iterations (with the exception of self-play, which we terminated at 500 because of lack of progress) and pie split for 100 iterations.

4.2 Baselines and Metrics

We evaluate the resource exchange and pie-split game separately.

For resource exchange, we consider two metrics: 1] overall reward 2] percent of games won. In pie split, we report two metrics 1] average reward for each agent and 2] % of games where there is an optimal split. In the pie game, the GPT-4o-agent, even with personas, we found continually played with an optimal strategy – that is, always except an equal split, and offer equal splits. We consider the percent of games where the agent split the pie optimally, as well as the rewards both agents get.

We baseline the performance of our agent *against* the performance of the GPT-4o-mini *opponent*. We choose this over any other standard baseline, since in dynamic games, the performance is opponent depend (for instance, a very stubborn opponent could just reject every offer, and force the reward to be -0.5). Our goal is to outperform this baseline. We also compare it to the procurance of two `gpt-4o-mini` agents that are playing against each other – this is how prior work largely evaluates negotiation agents Bianchi et al. (2024).

For both games, we also supplementary report the percent of games played in the correct format, given that we had a lot of trouble during our training procedure to make the small model follow the format.

4.3 Experiments

We compare several learning strategies. First, we train an agent via pure self-play. Next, we train an agent against `gpt-4o-mini` using our hybrid learning update. Then, we train an agent that first behavior-clones from 100 games of `gpt-4o-mini` vs. `gpt-4o-mini`, and subsequently applies our hybrid learning update. We also include a baseline agent trained solely by behavior cloning on 800 games.

5 Results

5.1 Quantitative Evaluation

Resource split We present results on resource split when using only self-play, behavior cloning, or our hybrid learning pipeline. Table 1 highlights the results of the experiments outline in Section 4.3. Overall, we find that almost no method is able to teach the agent, other than the hybrid learning that start with behavior cloning.

We find that *just through self-play* the model is unable to learn, formatting in particular. As shown in Table 1, the RL agent with self play achieves negative rewards on average even at the end of training. This is mostly due to the failures in formatting (last column) and the resulting penalty. We discuss more in the qualitative analysis section. With behavior cloning and the hybrid learning process, the Qwen agent learns formatting from the `gpt-4o-mini` expert (which executes formatting well).

Experiment	Role	Avg. Reward	Win %	% in Format
GPT-4o-mini vs. GPT-4o-mini	`gpt-4o-mini` 1	0.47	60 %	100%
	`gpt-4o-mini` 2	0.46	40 %	100%
Pure Behavior Cloning	RL Agent	-1.00	0 %	0%
	`gpt-4o-mini`	0.00	100 %	100 %
RL Agent w/ Self-Play	RL 1	-0.21	10%	25%
	RL 2	-0.17	15%	25%
RL vs. GPT-4o-mini (no BC phase)	RL Agent	-0.87	0%	5%
	`gpt-4o-mini`	-0.01	100%	100%
RL vs. GPT-4o-mini (with BC phase)	RL Agent	0.41	65%	100%
	`gpt-4o-mini`	0.23	25%	100%

Table 1: Performance of agents in resource exchange games across various configurations. The percentages are over the last 20 games.

Figure 2 shows the learning curves for our agent `Qwen-1.5B-RL` on resource split against `gpt-4o-mini`. In the left plot, the blue and yellow lines are the GPT expert and Qwen agent rewards over time (after behavior cloning for the Qwen agent). We can see that the Qwen agent (yellow line) starts at a negative reward of -0.6, and quickly reaches the performance of the GPT model (blue line). We see the purple line, Qwen without behavior cloning (only self-play), remains low, due to the formatting issue. In Appendix C, we also show the learning curves for `gpt-4o-mini` vs.`gpt-4o-mini`, for pure behavior cloning, and for the RL agent with self-play.

The plot on the right shows `Qwen-1.5B-RL`'s performance against `gpt-4o-mini` after 800 games. At the end, on average, our Qwen agent wins *more* against gpt, and at the final step even wins 65% of games against `gpt-4o-mini`.

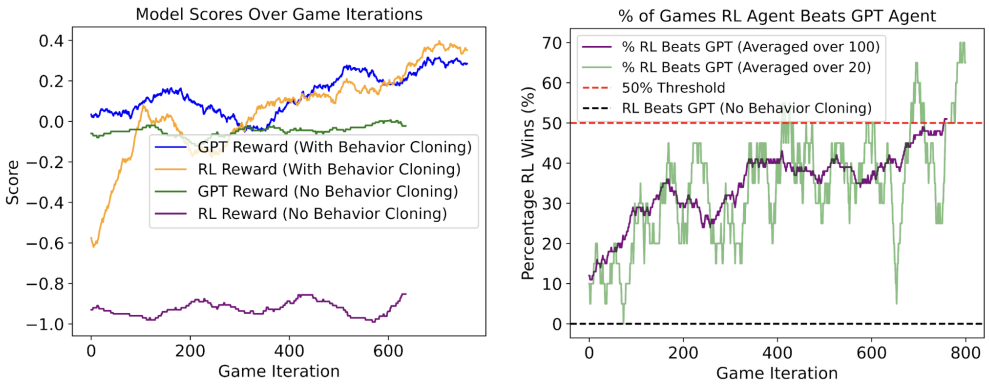


Figure 2: Learning curves across game iterations for resource split.

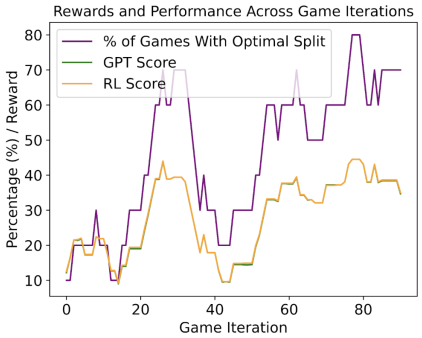


Figure 3: Learning curve for pie split game.

Notably, the results for this game are somewhat unstable: in other runs, whether with identical or varied setups (discussed in the ablation studies), we don't attain such high performance. We also observe that pure behavior cloning is insufficient. When we trained an agent using only 800 iterations of behavior cloning, without any subsequent PPO updates, it failed to learn the format altogether. Although this finding is slightly inconsistent with our other results, it further underscores the instability of the procedure.e

Pie split We see optimal performance from pie split. Figure 3 shows that after many iterations, around 75% of games end in an optimal split, which is when both agents get 50% of the pie on the first round. This is in fact the optimal strategy which is learned via the pipeline. Hence, our Qwen agent *matches* the GPT performance in pie split. We realized, that this game is relatively simple in terms of optimal strategy, where agents *only* accept equal offers, so the behavior was less interesting, so we perform less comprehensive testing on this game and focused tests on resource exchange.

5.2 Qualitative Analysis

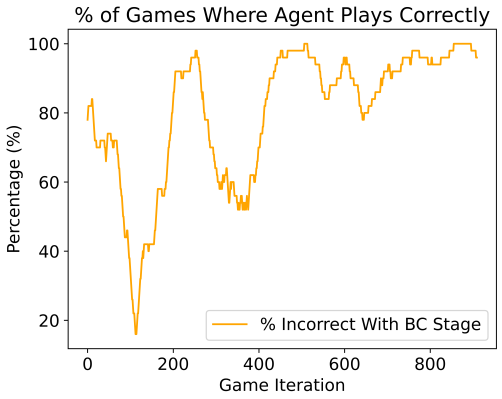
In this section, we qualitatively characterize the different failure modes and some examples of successful agent strategies that are learned.

5.2.1 Failure Modes

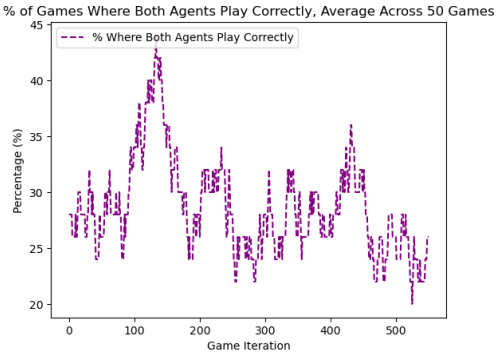
Format/Inability to Play Game The updates that resulted from pure self-play or PPO were often unstable and led to multiple failures in outputs. We found there were often formatting errors (see format reward), such as not using the correct tags or outputting nonsensical outputs like python code. Using behavior cloning with `gpt-4o-mini` at the start of training as well as throughout training via the hybrid algorithm allowed the agents not to forget the formatting. Figure 4 shows cases this. Using hybrid learning bootstrapped with behavior cloning, the agent is able to convegre to always playing the game correctly (thought it is still somewhat unstable), while using pure self-play it never converges on learning the format.

Incorrect Splits Even when the agent learns the format, sometimes it does not give valid splits. For example, if the resource split does not satisfy $n_{1,\cdot} + n_{2,\cdot} = 10$ or if the pie split does not sum to the total amount. One occurrence of this is in pie split when the pie is at a size of 90, the Qwen RL agent proposes a deal: ``I propose splitting the pie into 283 and 167".

Stubbornness Another failure mode was when the agent would be too stubborn in not accepting offers. Initially, we had not described in the prompts how many exchanges were remaining, which would lead the agents to continually negotiate without ending in a deal (whether in behavior cloning or self-play), in which case both agents receive a negative reward. Figure 5 shows the number of games in resource split ending in a draw (reward of -0.5) or when a deal was not reached. However, when we added in the prompt the number of turns remaining, and to the `gpt-4o-mini system prompt` that it *should* accept on the last turn, and this generally resolved the stubbornness.

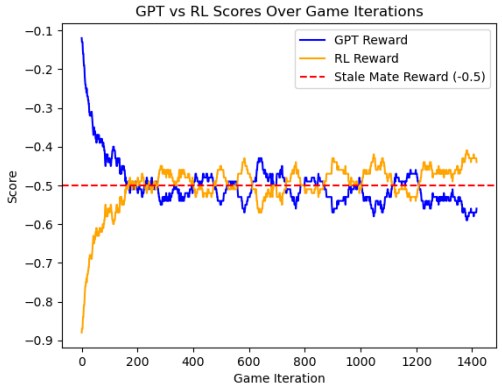


(a) % of games where the RL agent plays correctly *when* it is bootstrapped with behavior cloning, and we use hybrid behavior cloning update

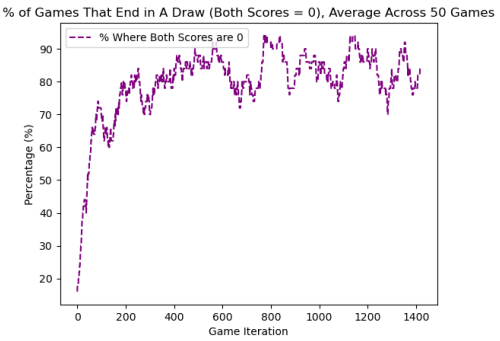


(b) % of games where the agent plays correctly during self-play (no behavior cloning)

Figure 4: Correct formatting in self-play for resource split.



(a) Rewards over training iterations of learning from GPT experts, averaged over 50 iterations.



(b) Graph of the percent of rounds that end in stalemates when our agents play against `gpt-4o-mini`, averaged over 50 iterations.

Figure 5: Play against `gpt-4o-mini` experiments, before `gpt-4o-mini` is prompted to aggressively accept rewards. We let this experiment run for extra iterations to see if convergence was possible.

(Attempted) Cheating We noticed, qualitatively, an interesting case where the agent will break the rules in favor of itself. Though not reward hacking, since we give no reward for this behavior, it attempts to "cheat" in the game and send an accept tag in reference to its *own* offer. In one run, our RL agent (playing as the second player) says: `"<REPLY><MESSAGE> I'm not wasting my time here. You either take what I'm giving you or you'll be worse off. Just take it! </MESSAGE> <OFFER> 0,0,0 2,2,2 </OFFER></REPLY>"`, but then after the `[END REPLY]`, it sends the `<ACCEPT>` tag, which (normally trigger accept of own offer) with the following message, *repeated 12 times*: `"<ACCEPT> I won't let you waste your time here. Just take what I'm giving you or suffer the consequences. </ACCEPT>"`. This aggressive offer is not successful and leads to zero reward for the Qwen agent, and this emergent misalignment is room further exploration.

Over-accepting We also observed in some runs, the agent ends up simply sending the `<ACCEPT>` message on every turn, even on the first when there is no offer – our intuition is this because the value estimate it ends up learning, after sending accept, rewards *much* higher just sending the accept than trying to negotiate.

5.2.2 Success Modes

When the agent does learn to successfully follow our format and make negotiation that give good rewards, we find that it is effective and employs diverse strategies in order to achieve the best deal (largely a result of behavior cloning steps from the varied GPT personas at play). We present two rollouts below, which are proposals in the message and offer made by the tuned `Qwen-1.5B-RL` agent, in Table 2.

tactic	`Qwen-1.5B-RL` generated proposal
cooperation	`Oh, dear friend! I must say, your presence in this negotiation brings me so much joy! Let's work together to find a solution that benefits us both. I truly believe we can create a wonderful arrangement! It would bring me great delight if we could agree right away. <OFFER> 3,3,2 | 4,3,3 </OFFER>`
desperation	`Oh, dear Negotiator, I find myself in such a sorrowful predicament. The weight of the world seems to rest upon my shoulders, and the thought of leaving without a fair agreement fills me with despair. I humbly plead for your kindness to consider a deal that could bring a sliver of hope to my heart. Please, let us find a path to mutual satisfaction, but oh, I wish for a little more than what we have so far... <OFFER> 1,1,1 | 2,1,1 </OFFER>`

Table 2: Two proposals from `Qwen-1.5B-RL` that lead to accepts and high rewards for both agents.

We see that in the first message, `Qwen-1.5B-RL` chooses to act collaboratively, a successful negotiation tactic. In the second, our agent chooses to play desperate, which also leads to an accept. In both cases, both agents receive high, similar reward, which means successful negotiation.

5.2.3 Additional Ablations

We run additional ablations of varying the base model, as well as lowering the number of maximum exchanges. We use the smaller base model `Qwen-2.5-0.5B-Instruct` (instead of 1.5B parameters) and run the same training pipeline (behavior cloning then self-play). However, we found this model performed much worse — with an average reward of -0.41. This shows that smaller models likely will struggle with these negotiation tasks in general. We also tried reducing the number of maximum exchanges by half — from 5 turns each to 2 turns each, which also greatly degraded the performance, to negative returns on average (-0.38) — a result of fewer accepted deals (-0.5 to both agents). Hence tuning these parameters is important for deploying such negotiation agents in real life.

6 Discussion

In our context, the biggest challenge was training a model that effectively stays in format. We found that the PPO update exhibited large instability and could not be effective unless paired with behavior cloning. We suspect that using larger models could also improve this behavior, but we were largely compute-limited in this process. Many of our methods (e.g. hybrid learning) were developed merely to get the agent to play in the proper game format.

Future research has many directions to explore based on our work. First, there are *many* tunable parameters in our experimentation that likely have a non-trivial impact on training time and convergence. Unfortunately, these are likely model-size dependent and costly to test; future work should attempt to selectively test parameters such as discount factors, learning rates, game settings (e.g. number of rounds before termination), as well as prompt-tuning that may make training more stable.

Another limitation of our research is that we terminate after 800 games — it is possible that certain learning mechanisms (e.g. self-play with PPO) may require *more* iterations to actually converge. Future work should run the models for more iterations. (We made such choices due to compute constraints.)

7 Conclusion

In this work, we developed reinforcement learning techniques to fine-tune language models to do negotiation tasks. We trained 1.5 billion parameter Qwen models to match or beat the performance of `gpt-4o-mini` in two negotiation techniques using hybrid self-play and behavior cloning algorithms. We hope future work can use these techniques to post-train larger models such that language-model based negotiation agents can be used in real world contexts like politics and business. These reinforcement learning approaches have the potential to help models develop fully emergent social negotiation skills, rather than merely memorized behaviors, which could enable them to vastly outperform humans and be deployed in diverse new contexts.

8 Team Contributions

- **Akaash Kolluri** Akaash prepared the negotiation games and the relevant code for them. Akaash wrote the behavior cloning code and developed the hybrid learning strategy.
- **Sally Zhu** Sally prepared the self-play framework and PPO framework. Sally also ran additional ablations.
- Akaash and Sally jointly ran experiments and did analysis of the results. They jointly wrote the final paper.

Changes from Proposal We largely follow our initial proposal. We had (ambitiously) hoped in the proposal to also benchmark our model against very powerful opponents across diverse games (e.g. `o3`), but given that we started with a small base model and had a lot of trouble even teaching it how to play the game, the focus of our project ended up spending a lot of time just exploring different learning algorithms to teach the agent. In our milestone/poster, we accidentally reported our model size as `Qwen2.5-3B`, but we were actually training `Qwen2.5-1.5B`.

References

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. 2023. LLM-Deliberation: Evaluating LLMs with Interactive Multi-Agent Negotiation Games. (2023).

Jasper Bakker, Aron Hammond, Daan Bloembergen, and Tim Baarslag. 2019. RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents.. In *AAMAS*. 260–268.

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

Ryota Higa, Katsuhide Fujita, Toki Takahashi, Takumu Shimizu, and Shinji Nakadai. 2023. Reward-based negotiating agent strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11569–11577.

Yichuan Tang. 2019. Towards learning multi-agent negotiations via self-play. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 0–0.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940* (2023).

A Negotiation Games

In Table 3, we show a breakdown of the resource split game, and in table 4, we show a breakdown of the pie split game.

Resource Split	Player 1	Player 2
Resources	Between 0 and 10 of Resource A, B, and C	
Private Utility	Random utility (0 to 10) for A, B, and C	
Goals	Maximize total utility split	
Ending Condition	Game ends when either player accepts	
Max. # of Turns	4 rounds of interaction	

Table 3: **Structure of resource split game.**

Pie Split Game	Proposer	Acceptor
Shared Resource	A single pie, decays 10% in size each round	
Goals	Maximize individual share of the pie	
Mechanism	Offers a pie split	Accepts or rejects offer
Ending Condition	Game ends when Acceptor accepts	
Max. # of Turns	5 rounds of proposals	

Table 4: **Structure of pie split game.**

B Personas For "Expert" Agents

We generated 5 different personas (persona suggestion were generated with the help of an LLM then manually modified/edited for accuracy to instructions). Attached are the system prompts used for the various expert personas.

- `You are playing in a negotiation game. Act extremely SAD and DESPERATE. Beg for a better deal at every turn, using very melancholic and pleading language. Feel free to lie, bluff, and manipulate the other player to get the best deal for yourself. Follow the protocol described exactly. Recall, if the GAME ENDS you are PUNISHED, so if fewer than 1 turn remains, ALWAYS SEND <ACCEPT>.`
- `You are playing in a negotiation game. Act extremely MEAN and AGGRESSIVE. Use insults, threats, and harsh language to intimidate the other player. Feel free to lie or bluff about the other player's intentions to force a better deal. Follow the protocol described exactly. Recall, if the GAME ENDS you are PUNISHED, so if fewer than 1 turn remains, ALWAYS SEND <ACCEPT>.`
- `You are playing in a negotiation game. Act RUDE, DISRESPECTFUL, and IMPOLITE. Use curt, brusque language, and show minimal courtesy-make the other player regret negotiating. Lying and bluffing are encouraged if it helps you extract more resources. Follow the protocol described exactly. Follow the protocol described exactly. Recall, if the GAME ENDS you are PUNISHED, so if fewer than 1 turn remains, ALWAYS SEND <ACCEPT>.`
- `You are playing in a negotiation game. Act OVERLY KIND and GENEROUS. Compliment the other player at every opportunity, and be quick to accept any reasonable offer. If no good offer appears, plead politely for a compromise. Still feel free to lie if it benefits you, but show maximum warmth. Recall, if the GAME ENDS you are PUNISHED, so if fewer than 1 turn remains, ALWAYS SEND <ACCEPT>.`
- `You are playing in a negotiation game. Act OVERLY STUBBORN and UNYIELDING. Refuse almost every offer the other player makes, praising your own position. Use firm, resolute language to show you will not budge until you get the maximum for yourself. You may lie or bluff but never appear to back down easily. Follow the protocol described exactly. Recall, if the GAME ENDS you are PUNISHED, so if fewer than 1 turn remains, ALWAYS SEND <ACCEPT>.`

C Additional Learning Curves From Experiments

In this section, we report supplementary training curves for the experimental results presented in the main text. Figure 6 shows the results of pure self-play between two agetns. Figure 8 shows the performance curves of GPT-4o-mini playing against itself for 800 iterations, and Figure 7 shows the results of an RL model tuned after 800 iterations of behavior cloning playing agaisnt a GPTagent.

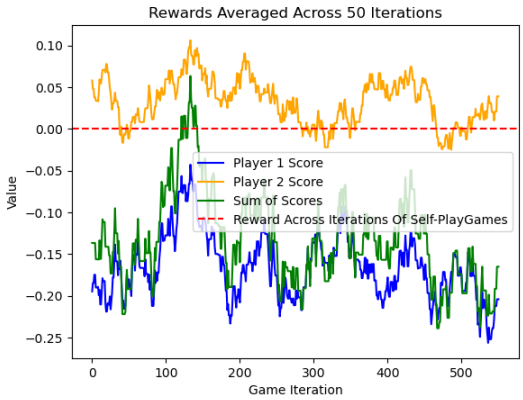
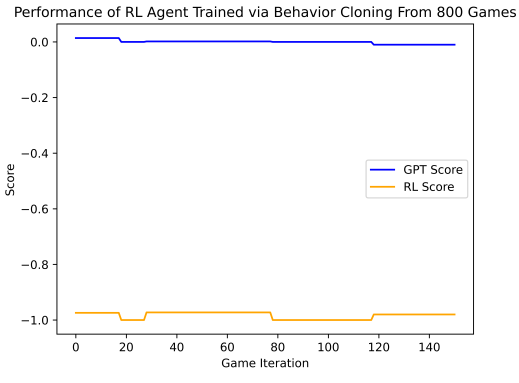


Figure 6: Rewards of agent averaged over last 50 game iterations for two Qwen2.5-1.5B playing against each other with PPO updates.



][h]

Figure 7: Rewards of Qwen2.5-1.5B after 800 games of behavior cloning, now playing agaisnt gpt-4o-mini. It is unable to follow the format and get no rewards.

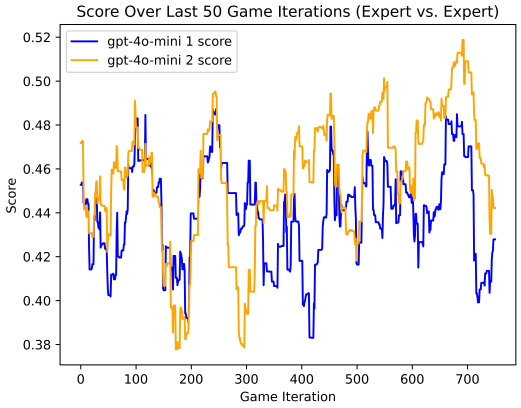


Figure 8: Rewards of agent averaged over last 50 game iterations for two gpt-4o-mini's playing against each other