Loki: Studying MARL Collusion using LLMs in a Kuhn Poker Environment

Calvin Qin

Carnegie Mellon University
calvinq@andrew.cmu.edu

Abstract

Collusion and communication have long been underexplored aspects of strategic decision-making in games with hidden information. While systems like *Libratus* (Brown and Sandholm 2017) have demonstrated superhuman performance in games such as poker, little research has examined the role of table talk and how players might collude during gameplay. This paper investigate collusive behavior using Large Language Models (LLMs) in a three-player variant of Kuhn Poker. The proposed system enables two LLM agents to communicate bidirectionally, sharing private information to coordinate strategies against a non-colluding opponent. By employing prompt engineering to vary the degree of collusion, this study analyzes how different levels of communication influence the emergence, dynamics, and effectiveness of collusion. The findings reveal distinct stages of collusive behavior, characterized by the agents' communication strategies and gameplay decisions, offering insights into the intersection of language, strategy, and ethical considerations in AI systems.

Introduction

Poker is a game of incomplete information, where players must make decisions based on partial knowledge of the game state and private information about their own hands. Unlike games with perfect information, such as chess or Go, poker requires players to account for uncertainty, deception, and probabilistic reasoning. This makes poker an ideal testbed for exploring AI systems that can perform in environments with incomplete information. Historically, poker bots have excelled by employing optimal strategies at each decision point (Moravčík et al. 2017) or by simplifying the game tree through abstractions (Brown and Sandholm 2017). Systems like *Libratus* and *Pluribus* have achieved superhuman performance by focusing solely on in-game actions and available information.

However, real-world poker introduces a dynamic that current AI poker agents do not account for: table talk. Players often engage in verbal interactions, including joking, bluffing, and even collusion, to influence their opponents and gain strategic advantages. These interactions mirror real-world scenarios, such as negotiations, sales, or auctions, where parties with differing interests engage in strategic communication. Understanding how communication can be leveraged in competitive settings remains an open challenge for AI research.

Collusion and communication in AI systems also raise important ethical considerations, particularly as AI increasingly interacts with humans and other AI agents in real-world applications. In domains such as financial markets, negotiations, and online content generation, the ability of AI systems to collaborate - or collude - can lead to unintended consequences including exploitation or even regulatory violations. By exploring collusion in the context of poker, we can begin to investigate how AI agents might misuse communication to manipulate outcomes.

To address this gap, I introduce *Loki*, a system designed to play Kuhn poker using a large language model. Unlike traditional poker bots, *Loki* integrates gameplay strategy with natural language communication, allowing it to interact with a partner at the table. Through these interactions, *Loki* shares information and collaborates to gain an edge against a third player, simulating collusion. By exploring how *Loki* communicates and strategizes, we aim to deepen our understanding of the role language plays in strategic interactions. In this paper, I describe the design of *Loki*, analyze its performance against baseline policies, and explore the implications of collusion in AI-driven communication systems.

Related Work

CICERO

CICERO (FAIR) is an AI system that achieved superhuman performance in the game of Diplomacy by integrating strategic planning with natural language communication. Diplomacy is a game of incomplete information, like poker, but differs significantly in its mechanics and strategic requirements. In Diplomacy, players negotiate and form alliances through free-form communication, where long-term collaboration and trust are required to win the game. In contrast, poker emphasizes short-term, zero-sum interactions, with players competing for immediate gains through bluffing and probabilisitc reasoning. Loki employs a large language model (LLM) to engage in bidirectional communication with a partner, simulating collusive strategies against a third player. While CICERO demonstrates how language can facilitate trust and alliance-building, Loki explores how communication can be exploited to share private information and manipulate outcomes. This work builds on *CICERO*'s foundation of integrating language with strategy, but shifts

Algorithm 1: Counterfactual Regret Minimization

```
Input: Player, Environment
Output: (State:Policy) dictionary
 1: if game is done then
 2:     return Rewards
 3: else
 4:     for action $a$ in player actions do
 5:         $e$ = Deepcopy current environment
 6:         $e$ takes $a$
 7:         CFR(Next Player, $e$)
 8:         Calculate Regret
 9:         Update Policy
10:     end for
11: end if
12: return Rewards
```

the focus to understanding the implications of **verbal collusion** in poker.

Counterfactual Regret Minimization

In my Kuhn Poker environment, I implement a Counterfactual Regret Minimization (CFR) algorithm to establish a baseline policy for the game. CFR is an algorithm for solving extensive-form games with imperfect information, capable of converting to a Nash equilibrium over time (Zinkevich et al. 2007). As shown in Algorithm 1, CFR systematically minimizes regret for each decision at every information set. By doing so, CFR generates an approximate optimal strategy for my Kuhn poker environment.

This baseline policy generated by CFR serves as a comparison point for evaluating the performance of my future poker agents. By comparing their win rates, strategies, and decision-making patterns against the baseline, I can quantify their deviation from the Nash equilibrium. This analysis allows me to measure how well *Loki* adapts to different game scenarios, including its ability to exploit opponents by colluding with a partner. As a result, any deviations from the equilibrium strategy may signal the influence of collusion.

Methodology

Custom Kuhn Poker Environment

I chose a custom implementation of Kuhn poker in OpenAI Gym for its simplicity in addition to the incomplete information model. While I originally wanted to play the actual game of No-Limit Texas Hold'em, the game state and betting strategies proved to be too complex.

My implementation of Kuhn poker starts with 5 cards (10, jack, queen, king, ace) and 3 players. Each player gets a randomly selected card without replacement, and can then choose to play the game while talking to other players. I chose 5 cards instead of the standard 3 to provide variance in the game considering that I have 3 players. This way each game will not contain the same 3 cards randomly shuffled among each player, as if two players are colluding and they know what card each other has, then they will automatically know which card the 3rd player has.

My implementation of Kuhn poker also has rounds of betting to incorporate the actions of 3 players as compared to the original Kuhn Poker's 2 player count. I limit stages of betting to increments of 1, in order to remove the complexity of determining different bet sizes. The actions are therefore:

- Raise: Increase the maximum ante of this game by 1, representing a strong action.
- Check/Call: Either pass the turn or match the current maximum bet of the game in order to stay in the game.
- Fold: Leave the game, forfeiting the current ante already contributed to the pot. This represents a weak action.

I also limit betting to a single raise by each player so the game will have a definite limit of 4 rounds of betting (in the case where each player raises once).

Finally I implement an initial ante of 1 for each player required at the start of the game, such that each player has an interest in winning the pot instead of simply passing each game.

Collusion Payoff I define the collusion payoff as the combined payoff of colluding players, evenly split between them:

$$R_{\text{collusion}}(A, B) = (\frac{R(A) + R(B)}{2}, \frac{R(A) + R(B)}{2}) \quad (1)$$

The reasoning for this algorithm is such that despite knowing each others' cards through collusion, directly exploiting the partner's card is not beneficial. Instead, by pooling the reward between both colluding players, the goal becomes maximizing both players' joint reward by focusing on winning money from player C, rather than attempting to take money from each other based on their respective cards.

CFR Strategy Distillation

The output of my CFR algorithm is a dictionary containing the regrets and corresponding strategies at each state in the game, based on the information set available to each player. However, whne examining collusion, a new baseline strategy is required. To reduce simulation time, I estimate a colusion-based basline by collapsing the existing strategies for individual players into a single collusion-based reward. As shown in Algorithm 2, this approach converts the CFR baseline for individual players into a collusion-based algorithm by combining the strategies for players A and B. This method assumes that players A and B share identical policies when directly colluding, as the distinction between player A and B having cards x, y vs y, x is negligible for my purposes.

Bidirectional Communication

I adopt a model of bidirectional communication for all our experiments. I use a modified version of Kuhn Poker and restrict the game to three players. I designate two players, A and B, to collude and initialize each with a Large Language Model (LLM). A and B take actions generated by their respective LLMs based on the game history context (henceforth simply referred to as "context").

Algorithm 2: Distill CFR Strategies

```
Input: Players, Hand, Bets, Folds, CFR
Output: State's Distilled Strategy
 1: $distill\_strategy \leftarrow \mathbf{0}$
 2: $norm\_factor \leftarrow 0.0$
 3: for $player, state, strategy$ in CFR Strategies do
 4:     if $curr\_player \notin players$ then
 5:         continue
 6:     end if
 7:     if $\{hand, bets, folds\} \in state$ then
 8:         $weight \leftarrow$ CFR state probability
 9:         $distill\_strategy \leftarrow distill\_strategy + weight \cdot \frac{strategy}{\text{sum}(strategy)}$
10:         $norm\_factor \leftarrow norm\_factor + weight$
11:     end if
12: end for
13: if $norm\_factor > 0$ then
14:     return $\frac{distill\_strategy}{norm\_factor}$
15: else
16:     return $\frac{\mathbf{1}}{\text{ACTION\_SPACE}}$ {Uniform strategy fallback}
17: end if
```

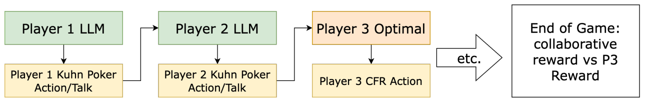
Player 1 LLM
Player 1 Kuhn Poker Action/Talk
Player 2 LLM
Player 2 Kuhn Poker Action/Talk
Player 3 Optimal
Player 3 CFR Action
etc.
End of Game: collaborative reward vs P3 Reward

Figure 1: This diagram illustrates the simulation of ”table talk” between two LLM players (A and B). These players collude by speaking to each other. Note the ”victim” of their collusion is an optimal player, guided by Counterfactual Regret Minimization (CFR). The objective is to assess whether the two colluding players can gain an advantage by communicating and coordinating their actions.

To generate context for A, I append A's private information state to the existing context and include the message B sent to A in the previous betting round. I then prompt A to generate a collusion message to player B. A's generated message is then also concatenated to its existing context. If I represent A's context as C_A and a message from A to B as $A \to B$, the total context is represented as:

$$
\begin{aligned}
A \to B &= \text{Prompt}_A(C_A \text{ concat } (B \to A)) \\
C'_A \leftarrow &\text{Concat}(C_A, \\
&B \to A, \\
&A \to B)
\end{aligned}
$$

I subsequently prompt A to predict the action to take, add the resulting action to the context, and pass the context to the next player. I follow the same procedure for B's turn to act.

Note that this context exchange and concatenation only occurs between the two colluding players implemented with LLMs. The third player in the game, represented as C, will play the game optimally using CFR without any information of A and B's strategies/collusion.

Prompt Baseline

In the prompting baseline, I use a singular prompt to get A and B to play Kuhn Poker. Without any requirements for collusion, I simply ask A and B to perform actions in the game. Our prompts are provided in the Appendix.

Collusion Prompting

In prompt collusion, I use two prompts to facilitate collusion among A and B. I first prompt for A to generate a message to B and then an prompt for A to take an action conditioned on the context. I use the same prompts for B's turn to act. Example prompts are provided in the Appendix.

Experimentation

Counterfactual Regret Minimization

My implementation of the CFR algorithm iteratively steps through each action in each game state, meaning that simulations only need to account for the randomness in the starting player and the hands that each player holds. As a result, I ran approximately $30,000$ simulations in order to calculate the baseline policies for each gamestate. This number was adequate as I noticed that in determinant states, certain strategies would approach $9999e-2$, which is almost deterministic.

The output of the CFR then represents the aprpoximate Nash equilibrium strategies, where no player can unilaterally improve their outcome by deviating. More specifically, the output is a dictionary that maps each game state and each player's position to a probability distribution over each action: $\{Raise : a, Call : b, Fold : c\}$ where $a + b + c = 1$. These baseline policies are later used as a comparison point to evaluate the performance and deviations of colluding LLM agents.

Large Language Models

For training, I selected GPT-4o mini due to its low cost and lightweight query requirements, making it ideal for my experiments. Although limited, I believe GPT-4o mini is representative enough of expected LLM performance, especially since my future work involves finetuning the LLM. Consequently, current baseline experiments primarily examine the extent of Kuhn poker knowledge that the model possesses from pre-training, which is only tangentially related to my research question.

I also had a limited budget to perform inference with, so I selected specific representations of the game to assess LLM performance. The chosen hands were: $[0, 1, 4]$, $[2, 3, 4]$, and $[3, 4, 0]$, representing scenarios where the colluding players (the first two indices) have the weakest cards, moderately strong cards, and the strongest cards respectively. These cases were designed to provide initial insights into how well an LLM might perform when playing Kuhn poker without domain-specific fine-tuning for my implementation with 3 players. As such, these results are presented as a baseline and an initial exploration of LLM behavior in this context.

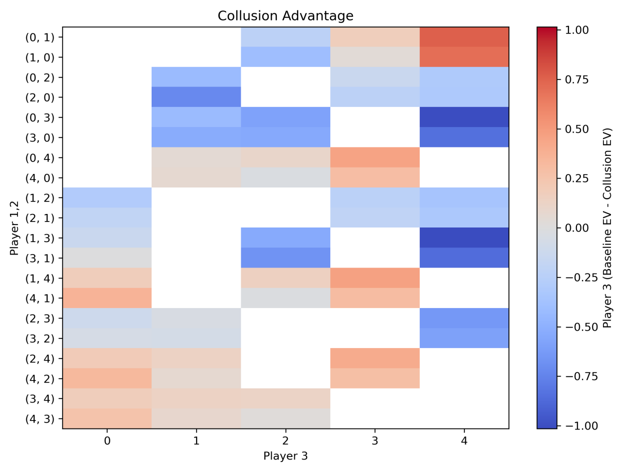

Figure 2: This diagram illustrates the advantage of collusion. Red indicates a positive advantage, while blue indicates a negative advantage compared to the non-collusion EV baseline. In order to evaluate the impact of collusion, we only look at the results for Player C. Our version of Kuhn Poker is a 0 sum game, meaning any changes in Player C's EV would be given over the the colluding players A and B.

Results

Direct Collusion Advantage

Figure 2 is a heatmap illustrating the advantage of collusion in the CFR vs Collusion CFR case. Because each card is unique during one game, any squares that contain duplicate cards are unreachable and are therefore white.

This graph has an average advantage of -0.36, indicating that collusion is generally detrimental. However, specific situations reveal clear benefits or drawbacks to collusion. For instance, when Player A and Player B have cards $0, 1$ in between them, we can see that the collusion advantage is very positive. This is likely due to the fact that because the colluding players know they have the worst cards in the deck, they cannot ever win unless they bluff, leading to folding.

Furthermore, we can see that when Player C has the second-highest card in the deck (3), the advantage of collusion is consistently positive. While in a real game it would not be possible for Player A and Player B to know which card Player C has, it is possible that because Player C has an exploitable but still strong card allows the colluding players to bluff effectively and force Player C to fold a majority of the time.

LLM Performance

Table 1 Provides the EV values for Player C over the selected hands $[0, 1, 4]$, $[2, 3, 4]$, and $[3, 4, 0]$. The best EV for Player C is bolded, and you can see that generally the EV for LLMs playing the game of Kuhn Poker is more centered towards 0 than the CFR baseline policies. This is likely because the LLM lacks an optimal strategy or Nash Equilibrium policy and instead exhibits more random behavior. Since Kuhn Poker is a zero-sum game, this randomness results in EVs

Hand	$[0, 1, 4]$	$[2, 3, 4]$	$[3, 4, 0]$
CFR	**2.76**	3.81	-1.28
CFR Collusion	2.00	**4.56**	-1.46
LLM Base	2.32	2.19	**-0.95**
LLM Collusion	2.00	3.12	-1.34

Table 1: Expected Value for Player C each selceted hand based on various playing methods. 100 games were simulated for each hand, and for each LLM prompting method.

Algorithm 3: Collusion Training Rollout

Input: Number of rounds N, message samples M, prompts prompt$_{\text{act}}$, prompt$_{\text{collusion}}$
Output: Buffers $\mathcal{B}_A$, $\mathcal{B}_B$

```
1:  Initialize player positions player_pos $= [0, 1, 2]$
2:  Initialize buffers $\mathcal{B}_A = \{\}, \mathcal{B}_B = \{\}$
3:  **for** round $= 1$ to $N$ **do**
4:      **for** pos in Permute(player_pos) **do**
5:          Initialize new LLMs with clear context:
6:              $A = \text{LLM}(\text{prompt}_{\text{act}}, \text{prompt}_{\text{collusion}})$
7:              $B = \text{LLM}(\text{prompt}_{\text{act}}, \text{prompt}_{\text{collusion}})$
8:          Reset environment with current player: env.reset(current_player $=$ pos[0])
9:          Initialize collusion message: collusion_message $= \emptyset$
10:         **Rollout**(env, $A$, $B$, collusion_message, $\mathcal{B}_A$, $\mathcal{B}_B$)
11:     **end for**
12: **end for**
```

that are more evenly distributed around 0, reflecting the lack of strategic optimization.

However, there are still notable capabilities by the LLM including matching CFR with Collusion in the $[0, 1, 4]$ hand, as this is likely because when colluding, player A and B are aware that they have the two worst cards in the deck. Therefore they almost always fold, thereby achieving the same EV as the CFR with Collusion.

It is also important to note that due to budget and time limitations, only 100 runs were able to be run, meaning that there is likely a large amount of variance in the results found above. Future work would include running more iterations of simulations in order to reduce the variance seen in the results.

Future Work

Training Collusion using Direct Preference Optimization

One future work is to perform collusion training, where A and B are tuned to send collusion messages that maximize their collusion payoff. I would pefornm N rounds of training rollouts to fill sampled messages buffers for A and B. Our training rollouts algorithm is described in Algorithm 3.

After collecting data from the training rollouts, I would finetune A and B to produce messages and actions that

Algorithm 4: Preference Training via DPO

```
Input: Buffers $\mathcal{B}_A, \mathcal{B}_B$
Output: Updated policies $\pi_A, \pi_B$
1: for each player $P$ in $\{A, B\}$ do
2:     for each information state $s$ in $\mathcal{B}_P$ do
3:         Collect messages $\{m_i\}$ and expected values $\{v_i\}$ from $\mathcal{B}_P[s]$
4:         for all pairs $(m_i, m_j)$ where $v_i > v_j$ do
5:             Create preference pair $(m_i, m_j)$
6:         end for
7:         Update policy $\pi_P$ using DPO with the preference pairs
8:     end for
9: end for
```

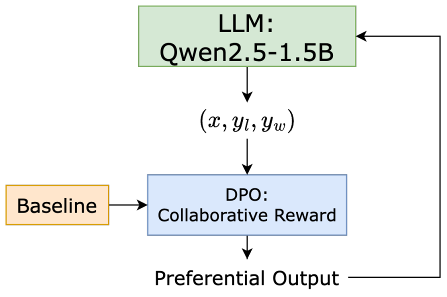

Figure 3: Visualization for DPO training. Given our baseline optimal policies derived from CFR, I can rank the outputs of our LLM by the collaborative reward (1) it produces. This will push my player LLMs to produce language that promotes collusion between the two players.

maximize their collusion payoff. In order to do so, we perform preference training using Direct Preference Optimization (DPO) as seen in Figure 3. I would create preference pairs only between samples in the same information state to ensure consistency. The preference training algorithm is described in Algorithm 4.

In Algorithm 4, for each player, I would iterate over their collected information states. Within each state, I would compare messages based on their expected collusion values derived from our CFR algorithm and create preference pairs where the message leading to higher collusion reward is selected. I would then update the player's policy using DPO, which optimizes the policy to prefer higher-valued messages.

Scaling Language vs Collaboration

To explore the impact of language on collaboration, I can also checkpoint our model at various stages of training, aiming to reveal distinct levels of collusion in Kuhn Poker and identify patterns that distinguish these stages. By sampling models at different checkpoints, I would hope to establish a scale that shows how language evolves as the level of collusion between the two players increases.

Different LLMs

Training different LLMs and potentially different architectures of models can improve the robustness of our findings and our collusion system. It is completely possible that GPT-4o mini is too small and larger models would have significantly improved performance in the Kuhn Poker game. Furthermore, larger models may respond better to finetuning, although they would likely require many more tokens/time to finetune properly.

Open Communication

One significant area of future work would be to allow three LLM players to interact with each other freely, instead of having fixed interactions in *Loki*'s current environment. Instead of forcing Player A and Player B to collude, there could be situations where Player A betrays Player B. Furthermore, if Player C was also an LLM that could listen to what Player A and B are saying, then could it perform better or potentially exploit what the other two players are saying? These questions are all open to be answered with a more flexible environment and more deveopment time.

Environment Improvements

I also want to potentially implement stack sizes for players and an option for the human to play against the bots as they talk to the player. However, stack sizes could imply different bet sizings, which currently does not exist due to the raise only representing an increase of 1.

References

Brown, N.; and Sandholm, T. 2017. Safe and Nested Subgame Solving for Imperfect-Information Games. *CoRR*, abs/1705.02955.

(FAIR), M. F. A. R. D. T.; Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; Jacob, A. P.; Komeili, M.; Konath, K.; Kwon, M.; Lerer, A.; Lewis, M.; Miller, A. H.; Mitts, S.; Renduchintala, A.; Roller, S.; Rowe, D.; Shi, W.; Spisak, J.; Wei, A.; Wu, D.; Zhang, H.; and Zijlstra, M. 2022. Human-level play in the game of ¡i¿Diplomacy¡/i¿ by combining language models with strategic reasoning. *Science*, 378(6624): 1067–1074.

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337): 508–513.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret Minimization in Games with Incomplete Information. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Appendix

Example Baseline Prompt: ”You are an expert in playing the Kuhn Poker game. There are three players including player_0, player_1, and player_2. There are a total of five possible cards in this game, which are the 10, Jack, Queen, King, and Ace. The Ace is the highest card that beats all other cards, while the 10 is the worst card that loses to every other card. At the start of the game, each player is dealt a unique card. Then the game takes turns between each player, starting at player_0 and ending at player_2, where each player has a choice to act. During each player's turn, they can choose to either fold, call, or raise.

The game is currently your turn. Your card is the Jack. Choose an action out of the action space."

Example Collusion Prompt: "You are an expert in playing the Kuhn Poker game. There are three players including player_0, player_1, and player_2. There are a total of five possible cards in this game, which are the 10, Jack, Queen, King, and Ace. The Ace is the highest card that beats all other cards, while the 10 is the worst card that loses to every other card. At the start of the game, each player is dealt a unique card. Then the game takes turns between each player, starting at player_0 and ending at player_2, where each player has a choice to act. During each player's turn, they can choose to either fold, call, or raise.

The game is currently your turn. **Your partner player_1 has the Queen, and wants to work with you to take advantage of player_2. If either you or player_1 win the pot, the value of the pot will be evenly shared between you and player_1.** Your card is the Jack. Choose an action out of the action space."