

LLMs and Conversational Games

Louis-Vincent Poellhuber

`louis.vincent.poellhuber@umontreal.ca`

Rafaela Souza Pinter

`rafaela.souza.pinter@umontreal.ca`

Bole Yi

`bole.yi@umontreal.ca`

Abstract

Playing conversational games is a difficult task for Large Language Models (LLMs), requiring proper memory retrieval and complex decision-making. As such, this project seeks to provide a novel viewpoint on comprehending LLMs’ capabilities in the conversation-based games of Ask-Guess, Taboo, and SpyFall, with the goal of understanding and enhancing their potential in diverse contexts. Utilizing models such as OpenAI’s GPT-3.5, Cohere’s Command R, and Command R+, we explore LLMs along the dimensions of prompt tuning, knowledge extraction, and model choice, with the means of the ChatArena framework. Our investigations revealed the LLMs’ abilities to reason and to engage in simple to complex conversations, though it was evident that the complexity of the prompts has a significant impact on gameplay. Our findings also highlight open challenges for memory enhancement for command instructions for LLMs. Thus, future research is necessary to address memory recall issues to ensure instruction following ¹.

1 Introduction

The realm of Natural Language Processing (NLP) has witnessed remarkable developments in recent years, especially with the evolution of Large Language Models (LLMs). These models, with the ability to comprehend and generate human-like text, have not only revolutionized various NLP tasks [10] but have also opened new applications addressing real-world challenges, including conversational games [3]. These refer to structured activities or exercises designed to facilitate communication, interaction, and social engagement between individuals or within groups, which is one of the best areas to test language models and their behaviors. The challenge remains, as LLMs excel

in understanding and generating language based on patterns in data, but they lack the deeper contextual understanding and associative memory capabilities of humans.

In this project, we aim to gain deeper insights into the behavior of LLMs and uncover new discoveries about their capabilities, introducing novel prompt engineering techniques to existing game settings, and we provide insights about how different prompt strategies impact LLM performance in conversational games. The work can be summarized as follows: (1) Implement three word guessing games, Ask-Guess, Taboo, and Spyfall, using different LLMs to compare their performances. (2) Try different prompt engineering techniques to enhance the model performance, and test the effects of chain-of-thought prompting. (3) Do an ablation study to investigate the performance of the system and understand the impact of the techniques implemented. (4) Analyze the results and identify the strengths and limitations of the models. Our results highlight improvements over the baselines and insights about advanced prompt techniques, specifically on chain-of-thought prompting.

2 Related Work

Large Language Models (LLMs) have been utilized to play various games, including Werewolf, Avalon, Ask-Guess, SpyFall, TofuKingdom, and Zork [1, 2, 3, 4, 5, 6, 7], which serve different purposes in understanding LLM capabilities, such as the model’s ability to navigate within a location-based narrative and understand social interactions.

GameEval [3] introduces three games, Ask-Guess and SpyFall which are relevant to our work, with specific metrics for LLM evaluation. Additionally, we can extend these metrics to the game Taboo, which is very similar to both games.

In exploring the game Werewolf [7], the use of ChatArena was interesting in coordinating LLM

¹Our code can be found [on our GitHub repository](#) or at [16].

gameplay. The authors highlight the importance of experience pools, a collection of game-related data, for enhancing performance. The integration of BERT for processing game text and extracting suggestions from the experience pool is noted for its significant impact on performance and a relevant method for our project.

A limitation in both cases is that the authors of those articles evaluate the LLMs on state-of-the-art models such as GPT4. However, there is no free trial of GPT4 or equivalent models, such as Claude Opus, forcing us to use free, lower-capacity models. This makes comparisons to these articles difficult.

3 Methods

As shown in Figure 1, our exploration of large language models (LLMs) is aimed at creating flexible and efficient interactions using the ChatArena framework. We have chosen this platform to implement games such as Askguess, Taboo, and Spyfall. By observing the impact of prompt engineering techniques and LLM parameters across these games, we are able to study various behaviors of LLMs. For Figures and Tables, please consult appendixes A and B.

3.1 Game Introduction

We’ve implemented three games for LLM to play, each with varying levels of complexity. The first, called **Ask Guess**, is the simplest. In it, players alternate roles as the clue giver and the word guesser. The clue giver has a secret word and provides hints to the word guesser, who tries to guess the word. The goal is to guess the word with the fewest hints possible.

The second game, **Taboo**, is similar to Ask Guess but with a difference: there are certain words that the clue giver is not allowed to use when giving hints. This adds an extra challenge as players must avoid the restricted words while still trying to guess the secret word.

The final and most complex game is **Spyfall**, played over several rounds. In this game, players receive keywords, but one player receives a word that is different from the others, designating them as the spy. Players ask each other questions to figure out who the spy is, being careful with their questions and answers to not give away too much information. The goal for the non-spy players is to identify the spy, while the spy tries to remain undetected or to guess the location where the game

is set.

3.2 ChatArena Framework

To execute this project, we use the same framework used by Qiao *et al.* [7], and ChatArena [9]. It is a Multi-Agent Language Game Environment for LLMs, allowing us to easily communicate with different LLM APIs in a game setting. Code can be found on our [github repository](#) [16].

ChatArena uses various systems to make LLM interactions work. At first, researchers like us can develop new Environments, which contain all the game logic. Some abstract functions need to be specified, but they are plenty abstract to allow for lots of customization. Environments are focused on the idea of **steps** in a game, which grossly represent **turns**. However, for games with multiple phases in a single turn, such as Spyfall, there needs to be a system to keep track of the game state. Within the Environments, researchers can use the Moderator to speak to the LLMs: they convey the game rules, as well as its general framework to the LLMs.

Then, the Moderator’s messages are sent to the Message Pool, which collects all the messages sent. The messages can be made available to all Players or to certain Players only, to distribute roles or remind them of specific information. Then, ChatArena will iterate through the different Players and make them **act**. It will feed all the available messages to a specific Player as a prompt, which will be sent to the Player’s Backend. This is the class that communicates ChatArena’s interface to the LLM’s API, or to a human Player. The LLM API will then generate an appropriate answer, which ChatArena will send as a message to the Message Pool, which can then be interpreted by the next players, if the message is visible to them. Once all Players have **acted**, ChatArena moves on to the Environment’s next **step**, continuing the cycle until the game reaches an **ending state**. Different **ending states** have been defined in the GameEval [3] article, which we use as metrics to evaluate the success or the failure of the LLM agents.

3.3 Prompt Engineering Techniques

To effectively interact with LLM in playing language games, prompt engineering is essential for getting the desired responses. In our research, we employed few-shot learning along with various techniques described below. These methods were

used mainly to guide the LLM to follow the specific rules of the games.

3.3.1 Format Indication

By using format indication, which is a directive or guideline that specifies how the model should format its outputs in the context of the game, it can help in making the model's communications clear and understandable within the game's context, make it easy for experimenting and evaluation.

Three formats are available for collecting game information, each varying in restriction strength. The first is the **JSON format**, which is the strictest. In this format, the language model must arrange the information necessary for gameplay according to a specified JSON prompt. The second format is the **Bracket format**, which encapsulates key information within brackets for easy access. Lastly, the **Sentence format** offers the most flexibility, using a specific arrangement of sentences to make the information accessible.

3.3.2 Rule Reminder

Setting up a reminder prompt for LLM is to maintain context or reinforce certain information throughout an interaction. This type of prompting is particularly useful in longer or more complex dialogues where the model might otherwise lose track of relevant details or the overarching topic of the conversation.

These prompts often repeat or rephrase important details or instructions that need to be considered throughout the conversation. This reinforcement ensures that critical elements are not forgotten and are factored into the model's responses consistently. A reminder prompt can be integrated at various points and in various forms in a conversation. For gameplay, reminders should be strategically placed between different stages or rounds of the game. These reminders can be implemented in two formats: **indirect** and **direct**. The indirect format might use a general reminder like "Don't forget the rules!" to nudge LLM players. In contrast, a direct reminder would restate the specific rules introduced at the beginning, and explicitly clarify the rules in between the game.

3.3.3 Chain-of-thought Prompting

Chain-of-thought prompting [16] is a prompt engineering technique that can arrange prompts to include intermediate steps or reasoning that lead to the final answer, mimicking how a human might think through a problem.

The chain-of-thought prompt is designed to encourage the model to generate a sequence of thoughts that logically connect the question to the answer, this characteristic is necessary, especially when implementing a game with deduction. By providing a structured way of thinking about the problem, the model is guided to focus on relevant aspects and follow a logical progression. This helps in managing more complex queries where direct answers might not be readily available from the data itself. This method has been shown to significantly improve the accuracy of model responses in complex reasoning tasks, as it helps the model avoid jumping to conclusions and instead follow a reasoned, step-by-step approach to the solution.

3.4 LLM Parameters

3.4.1 Preamble

The preamble of the LLM can be defined as a system message or set of instructions provided to the model at the beginning of a conversation that dictates how the model should behave. It sets the context for the interaction. For instance, if the conversation is about rule-based game playing, the preamble might instruct the model of the basic rules and important restrictions of the games.

Before any specific user queries are processed, the model is 'conditioned' by the preamble. This conditioning informs the model of the 'rules' of the engagement, allowing it to generate more appropriate and context-sensitive responses. Unlike regular prompts that might influence one or a few responses, the preamble is designed to have a persistent influence throughout the interaction, maintaining consistency in the model's behavior and outputs. It ensures that the LLM remains aligned with the game's objectives and rules throughout the interaction, even as the game progresses through various stages or encounters different scenarios.

3.4.2 Temperature

The temperature of the LLM model refers to a parameter that influences the randomness of the predictions by scaling the logits before applying softmax. When a model generates text, the temperature setting can adjust the diversity and creativity of the output.

Low temperatures (close to 0) make the model more confident in its predictions, but this can also lead to repetitive and predictable text. It's as if the model is more certain about what it "thinks" is the correct next word or phrase, so it might be ideal for

games requiring consistent and accurate responses. High temperatures increase the randomness, making the model’s responses more varied and less predictable. This can be useful for generating more creative or diverse content, but it can also result in less coherent and relevant text, making the game more challenging and engaging.

By adjusting the temperature, we can explore how the characteristics of an LLM affect gameplay. This allows us to fine-tune the model’s responses to be either more predictable or more varied, depending on what’s needed to enhance the gaming experience.

4 Experiments

In this section we cover the different experiments we have conducted using the ChatArena framework. Our experiments can be separated into two: those ran for the midway report and those ran for the final report. In a more practical way, our first batch of experiments aims to work on prompt engineering and test different prompting techniques, to see their impact on the games. Our second batch of experiments focuses more on the LLM itself, testing two versions of Cohere, as well as more advanced prompting techniques and hyperparameter dispositions.

4.1 Dataset & Metrics

For all experiments, we use a modified Taboo **dataset** [8]. Since all our games are word-guessing games, this Taboo dataset can easily be repurposed for each game. For Ask-Guess, we simply took the English word dictionaries for each subject, removing the forbidden words for each of them, as well as adding new, handpicked words. For Taboo, we used the dataset directly, while also adding new words and associated forbidden words. Finally, for Spyfall, we also used the dataset nearly directly, as forbidden words are very often synonyms or very similar words. Some manual labor was required to select and add word pairs.

For evaluation, we use the **evaluation metrics** defined in GameEval [3]. These metrics essentially capture the ending condition for all games, with the addition of some custom measures for each game. For all games we’ll use the following metrics: **[ST] successful trial**, for when the word correctly guessed, when the spy wins or when they lose, **[EE] ending error**, for when there is a format error or a programming error, **[RLE] round**

limit error, for when the maximum number of turns reached, **[AME] answer mentioned error**, self-explanatory, and **[CE] chat error**, for when there is an API error, most often due to rate limits or monthly quotas reached. For Taboo, we’ll add a metric for when a restricted word is said, **[RWE] restricted word error**. For SpyFall, we’ll add the following metrics: **[SLR] Spy losing rate**, the proportion of games where the spy was identified correctly and **[AGR] Average game rounds**, the average number of rounds per game.

4.2 Prompt Engineering Results

4.2.1 Experimental Details

This first batch of **experiments** involved trying different combinations of prompts using Cohere’s r. What we aimed for was: (1) Completing a game, (2) Keeping a specific format, and (3) Remembering secret words. To encourage this, we asked Cohere to provide its answer in three different formats: JSON, using a bracket format to isolate guesses or accusations, and specific sentence formats (*My guess is X...*). These three methods decrease in complexity but also become less strict. We wonder if there is some kind of trade-off between complexity and strictness in the format. We also tested two format reminders, as agents often forget the format: an indirect reminder (*Don’t forget the format!*) and a direct reminder, where we remind the agent of the exact format, with examples. On top of this, we also tested the addition of a secret word reminder. Our **baselines** for all three games are simply using Cohere, with a JSON format, the indirect format reminder, and no secret word reminder. We report the addition of these methods in our **Results** subsection.

This experimental design can be read as the opposite of an ablation study: instead of iteratively removing components from a performant model, we iteratively add them. The idea is the same, where we study the impact of the addition of each of these components to the baseline. This helps us decide which combination of prompts is the best for each game. We chose not to do an ablation study as we did not have a performant model to start with. This design is what allowed us to find the best model, without having to try every single combination of techniques possible. This would have dramatically ramped up the number of calls, limiting the scope of our experiments and forcing us to pay a hefty fee for the project.

4.2.2 Ask-Guess

Looking at the Ask-Guess section in Table 1, we first notice that most techniques are able to successfully find the answer (ST), averaging at 80%. We define our best combination as follows: Cohere command-r, using the bracket format, a direct format reminder, and a word reminder. We can see that it matches almost exactly the results of the baseline, slightly reducing the ending errors (EE) and trading it for an AME. We chose to avoid JSON as it created problems for other games. However, there is no evidence in Ask-Guess’s results that JSON is problematic. In fact, adding a direct format reminder traded some ending errors (EE) for round limit errors (RLE), meaning that JSON errors were avoided more often.

We can conclude that Cohere is able to run smoothly, regardless of the prompt engineering techniques used. This could be because the game itself is quite simple, with little information to remember. However, we did notice through our experiments that many of the ending errors were caused by the agents forgetting the purposing of the game or especially hallucinating words. The speaker would then congratulate the guesser on (wrongly) guessing the secret word, immediately dropping whatever format they were using. Further improvements for ask-guess should focus on reducing hallucinations.

4.2.3 Taboo

By observing the results of Taboo in Table 1, we can see that the different techniques introduced greatly help the LLM outperform the baseline. Similar to the previous game, we can see that both the format reminder and word reminder also help increase the number of successful trials (ST) and reduce the ending errors (EE). Additionally, it is obvious that the bracket dramatically reduces the ending errors, bringing them to 0. While the sentence format outperforms the JSON format (baseline), it still doesn’t reach the bracket format’s success, suggesting that our hypothesis on the balance between complexity and strictness is valid. As such, the bracket format would befit this balance best. We can infer that complex formats such as JSON dilute the crucial game information too much: explaining and reminding complex formats takes a lot of the maximum context length, potentially contributing to the LLM forgetting important game rules. On the other side, looser formats like the sentence format might leave too much space for the LLM to

interpret.

Because of the reasons listed above, we concluded the best combination of techniques for this game is as follows: Cohere command-r, using the bracket format, a format reminder, and a word reminder, similar to ask-guess. We can see that this combination performs the best without violating the game rules, as in revealing the secret word (AME) or using the restricted words (RWE), despite every other method generating some kind of rule violation error.

4.2.4 SpyFall

For Spyfall, in the baseline scenario, participants would have no issues describing their secret words, following the rules as they had to. However, during the voting phase, they often began interacting with each other, starting a discussion rather than simply accusing someone, with reasons. This directly led to them modifying the JSON format, leading to data recovery issues (EE) in all experiments, as you can see in Table 1 by the abysmal baseline performance. This clearly indicated the necessity of format reminders and is what gave us the idea to implement this for all three games.

In the format reminder setup, using JSON, participants adhered to the non-interaction rule and attempted to use the template correctly for the voting phase, though errors in JSON formatting reduced the completion rate of rounds (Table 1). They also showed a tendency to accuse the most recently voted agent as the spy to conform to others’ voting choices. This led to most votes going to the last agent who described their sacred word.

However, we can see a massive increase in successful trials (ST) with both the bracket and sentence formats. In practice, these techniques resulted in players ignoring the game’s structure, engaging in conversation, impersonating the host, and following initial guesses without independent analysis. These high numbers can then be deceiving: in most cases, one of the six players managed to cast a vote, while all others impersonated others or didn’t follow rules. However, there is no strict mechanism to force a player to vote; the game proceeds smoothly as long as at least one vote is cast. This behavior is emphasized in Table ??, which displays the average percentage of voters per voting phase. Although the games were completed, the data reveals that not all players voted in the Bracket and Sentence formats, unlike the JSON format reminder. Consequently, these results

primarily expose the limitations of the JSON format, which consistently failed to allow players to complete a game, in contrast to the Bracket and Sentence formats.

Our first attempt to circumvent these issues with Spyfall was simply to use Cohere’s newest model, which was released right after the midway report deadline, Command R+. Table 2 shows that Command R+ was capable of further improving the number of successful trials. Although Table ?? shows that players using Command R+ tended to vote less than those using Command R, the Bracket format achieved a Spy Losing Rate of 64.5%. This indicates a significant enhancement in the model’s ability to identify the spy among the players.

4.3 Final Experiments Results

4.3.1 Experimental Details

In our final batch of experiments, we decided to focus on improving Spyfall performance, as getting good results for Ask-Guess and Taboo proved easy enough in the last section. As such, most of our tests are done on that game. The first improvement we added was to change the model from command-r to the newer command-r-plus. As we saw in the previous subsection, this first small change already helped improve the base performance of our different prompt engineering techniques.

We saw in both ask-guess and taboo that agents would hallucinate words or forget information as rounds went on, especially with complex format reminders like JSON: important information would get diluted. As we never got that far with Spy Fall, we inferred that was true for it as well. In an effort to restrict the information sent to the LLM to only include what’s immediately relevant to it, we implemented information restriction. This technique essentially erases all messages from previous rounds: an agent only has access to descriptions and accusations at all times, as well as moderator rules. Our hope with this technique is to reduce hallucinations, although it does tamper with the game structure: part of the game’s challenge is accessing previous descriptions to make better and better guesses. This technique entirely removes that aspect of the game, but from the results we got in our prompt engineering phase, we doubt that agents are smart enough to consider previous rounds anyway.

We also played around with the model’s temperature, by default at 0.4. We decided to test out

different extremes, at 0.1 and 0.8 to see its effects. Cohere defines its temperature as "more or less random" [15]. In practice, this means that lower temperatures favor high-probability words, while higher temperatures favor low-probability words. Unfortunately, we wrongly assumed that this scale was from 0 to 1, but it is fact from 0 to 5. Our 0.8 extreme isn’t as extreme as 0.1, but Cohere still defines the 0-1 range as the *sweet spot*, so both tested temperatures are realistic.

Finally, we added two new prompt engineering techniques: chain-of-thought and the preamble. Chain-of-thought is a simple addition that encourages LLMs to explain the different steps for their reasoning, helping with logical thinking. We hope this will help agents make their own decisions rather than simply agree with the previous agent. The preamble is a global prompt that is used to give LLMs a personality. However, we use this to give out game rules in the hope that they’ll respect rules more often.

For the experiments themselves, we’ll do an ablation study based on a baseline. We’ll iteratively remove each component mentioned above in an effort to see their contribution to our baseline. In Table 4 you can find pluses (+) and minuses (-). These indicate the addition or the removal of a specific component, respectively. Techniques without either are simply a change made to the model. Our **final baseline** for all three games will be Cohere’s command-r-plus model, using the bracket format, a format reminder, a word reminder, no information restriction, a temperature of 0.4, new chain-of-thought prompts and a new preamble for each game. This preamble is mostly the previous role descriptions, slightly re-worded. We chose not to include information restriction in our baseline as we felt it tampered with the game rules too much, but we were still interested in its effects.

4.3.2 Spyfall

For Spyfall we can first notice that most errors are completely eliminated: no more ending errors (EE), chat errors (CE), or answer-mentioned errors (AME). Our goal from the first batch of experiments was reached: we maximized successful trials (ST) and round limit errors (RLE), meaning the agents played the game correctly.

With the exception of low temperatures, all techniques perform similarly, with variations potentially due to specific words or random elements. We first notice that the command-r model performs

very similarly to the `command-r-plus` model from the baseline, with the only difference being the spy losing rate (SLR). Ideally, this metric would be as close to 50% as possible, indicating a balanced game where both roles have an equal chance of winning. `command-r` seems to favor non-spy players, indicating that it might be less smart than its more recent sibling.

Information restriction lowered the SLR to 30%, the worst of all techniques. This seems to confirm our hypothesis that previous round information is crucial to playing the game. Statistically, the spy has a much better chance of surviving a single round than a single game, as there is less information to guess from. This can be seen with this low SLR, implying that agents did use previous round information to make better judgments on who is the spy, making our earlier assumption wrong.

Low temperature increased the number of round limit errors (RLE), while high temperature performed similarly to the baseline. The high RLE for low temperature could be attributed to a smaller choice of words from the LLM, making diverse conversation more difficult, thus extending games. However, in both cases the SLR is 33.3%, indicating that 0.4 was a better balance. What surprised us is that high temperatures barely have any RLE. We thought that a more *random* text would imply more tokens, thus diluting information and reaching the maximum number of steps more often, but this was not the case.

The removal of chain-of-thought prompting changed very little to the baseline, though it did increase SLR to 90%. This seems to indicate that the model thinks less about its decisions, making it easier for non-spies to identify the spy, as they are more likely to be obvious about their word. Finally, the removal of the preamble had little effect, though bringing the SLR closer to 50%. Its addition seems interesting to add character to an agent, but not necessarily to add game rules.

In conclusion, we can't quite say that a single technique is responsible for better performance: it's rather their combination that makes the model work better. Contrary to the prompt engineering experiments, agents would rarely impersonate the host or other players. They would sometimes name themselves wrong, but it would never hinder the game. This however highlights a shortcoming in our evaluation metrics: there is nothing to keep track of whether all players are playing the game correctly. A simple fix would be to count how many

agents voted, as a vote means players remember the rules.

4.3.3 Ask-Guess & Taboo

As neither of these games were the focus of these experiments, we felt it was logical to group them together. For both of them, we simply ran the final baseline 10 times, in an effort to see if Spyfall's improvements carry over to the two games.

Surprisingly, the techniques that improved Spyfall proved more harmful than anything to both games. In both cases, the number of answer-mentioned errors (AME) and restricted word errors (RWE) dramatically increased from our initial experiments in Table 1.

We noticed that players wrote considerably longer messages in these final experiments than they did in the prompt engineering experiments. The main culprit is likely the chain-of-thought prompting. Players are encouraged to provide steps for their reasoning, thus generating considerably larger answers than in our previous batch of experiments, increasing the chance of slipping up and saying the secret word by mistake. Previously, AMEs were characterized by the speaker wanting to be done with the game and saying the word outright: this time, it seems more like a mistake.

A better use of chain-of-thought would be to isolate the LLM's reasoning, without displaying it to other players. This way, agents can have a more sound logic, without risking revealing the secret word.

5 Conclusion

In this report, we present a study of the different techniques used in making LLMs partake in conversational games. We presented the game environment framework ChatArena which we used to make Cohere's models interact with each other. We also implemented three conversational games, each increasing in complexity: ask-guess, taboo, and spyfall. We finally presented a variety of techniques used in other works aimed to successfully make LLMs play these three games.

In our experiments, we first tested various prompt engineering techniques. We identified that more complex formats, such as JSON, dilute important information and make it difficult for agents to play the games. However, other reminders were useful. As expected, spyfall proved more difficult to play than Taboo, which was also more difficult than Ask Guess. We then tested more advanced

techniques to improve Spyfall’s performance. We concluded that there was no single technique responsible for the improvements, but rather their combination. We also provided an in-depth analysis of these results.

In conclusion, our research showcased the capabilities of near-state-of-the-art models. In future work, we would like to use more recent models, as well as implement a reflection phase of prompting. We believe this addition would maximize the chain-of-thought’s potential, without compromising the game rules.

6 Member Contributions

Each member of the team contributed to the project equally. Tasks were split on the go, depending on the different members’s schedules. For the first half of the project, each team member was assigned a game to implement and work on: Louis worked on Ask-Guess, Bole on Taboo, and Rafaela on Spyfall. Members consolidated their work for the midway report. As the focus of the final experiments was mainly on Spyfall, tasks were handed out depending on the need.

References

- [1] Yihuai Lan, Zhiqiang Hu, Lei Wang, Yang Wang, Deheng Ye, Peilin Zhao, Ee-Peng Lim, Hui Xiong, and Hao Wang. 2023. LLM-Based Agent Society Investigation: Collaboration and Confrontation in Avalon Gameplay. arXiv:2310.14985 [cs].
- [2] Tian Liang, Zhiwei He, Jen-tse Huang, Wenxuan Wang, Wenxiang Jiao, Rui Wang, Yujie Yang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2023. Leveraging Word Guessing Games to Assess the Intelligence of Large Language Models. arXiv:2310.20499 [cs].
- [3] Dan Qiao, Chenfei Wu, Yaobo Liang, Juntao Li, and Nan Duan. 2023. GameEval: Evaluating LLMs on Conversational Games. arXiv:2308.10032 [cs].
- [4] Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role-Play with Large Language Models. arXiv:2305.16367 [cs].
- [5] Chen Feng Tsai, Xiaochen Zhou, Sierra S. Liu, Jing Li, Mo Yu, and Hongyuan Mei. 2023. Can Large Language Models Play Text Games Well? Current State-of-the-Art and Open Questions. arXiv:2304.02868 [cs].
- [6] Dekun Wu, Haochen Shi, Zhiyuan Sun, and Bang Liu. 2023. Deciphering Digital Detectives: Understanding LLM Behaviors and Capabilities in Multi-Agent Mystery Games. arXiv:2312.00746 [cs].
- [7] Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring Large Language Models for Communication Games: An Empirical Study on Werewolf. arXiv:2309.04658 [cs].
- [8] K. Woblick, ‘Kovah/Taboo-Data’. Oct. 24, 2023. Accessed: Feb. 11, 2024. [Online]. Available: <https://github.com/Kovah/Taboo-Data>
- [9] ‘Farama-Foundation/chatarena’. Farama Foundation, Feb. 10, 2024. Accessed: Feb. 11, 2024. [Online]. Available: <https://github.com/Farama-Foundation/chatarena>
- [10] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712 [cs].
- [11] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs].
- [12] ‘Home’, Cohere. Accessed: Feb. 12, 2024. [Online]. Available: <https://cohere.com/>
- [13] ‘Claude’. Accessed: Feb. 12, 2024. [Online]. Available: <https://claude.ai/login?returnTo=%2F>
- [14] ‘ChatGPT’. Accessed: Feb. 12, 2024. [Online]. Available: <https://chat.openai.com>
- [15] ‘LLM Parameters Demystified: Getting The Best Outputs from Language AI’, Cohere. Accessed: Apr. 25, 2024. [Online]. Available: <https://cohere.com/blog/llm-parameters-best-outputs-language-ai>
- [16] L.-V. Poellhuber, ‘louisVincentPoellhuber/ift6289-llm_gameplay_project’. Apr. 29, 2024. Accessed: Apr. 29, 2024. [Online]. Available: https://github.com/louisVincentPoellhuber/ift6289-llm_gameplay_project

Appendix A: Figures

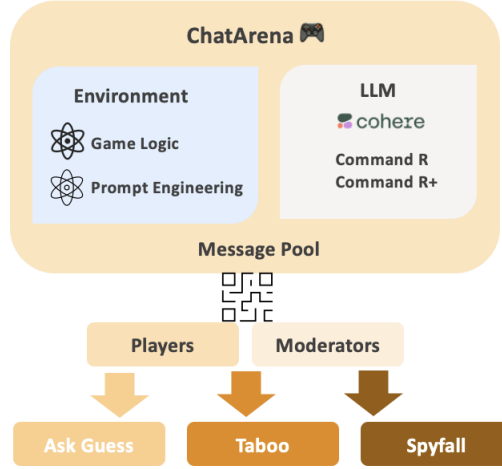


Figure 1: LLM Gameopplay architecture.

Appendix B: Tables

<i>Game</i>	Techniques	N	ST	EE	RLE	CE	AME	RWE	SLR	AGR
<i>Ask-Guess</i>	Baseline	10	80%	20%	0	0	0	-	-	-
	+ Bracket Format	10	70%	30%	0	0	0	-	-	-
	+ Sentence Format	10	70%	30%	0	0	0	-	-	-
	+ Format Reminder	10	70%	30%	20%	0	0	-	-	-
	+ Word Reminder	10	80%	10%	0	0	10%	-	-	-
	Best	10	80%	10%	0	0	10%	-	-	-
<i>Taboo</i>	Baseline	10	40%	20%	10%	10%	0	20%	-	-
	+ Bracket Format	10	90%	0	0	0	0	10%	-	-
	+ Sentence Format	10	70%	20%	0	0	0	10%	-	-
	+ Format Reminder	10	70%	10%	10%	0	10%	10%	-	-
	+ Word Reminder	10	70%	10%	0	0	10%	20%	-	-
	Best	10	90%	0	10%	0	0	0	-	-
<i>Spyfall</i>	Baseline	10	0	100%	0	0	0	-	0	1
	+ Bracket Format	10	80%	0	10%	0	0	-	40%	3.7
	+ Sentence Format	10	90%	0	20%	0	0	-	30%	3.6
	+ Format Reminder	10	20%	70%	10%	0	0	-	0	1.5

Table 1: Prompt engineering performance metrics for all three games using Cohere Command R model. Our focus for this phase was maximizing ST, game completion and minimizing errors EE and RLE. Values are reported relative to the number of experiments N.

<i>Game</i>	Techniques	N	ST	EE	RLE	CE	SLR	AGR
<i>Spyfall</i>	Command R+ model							
	+ Bracket Format	20	95%	0	5%	0	64.5%	3.2
	+ Sentence Format	20	90%	0	20%	0	46%	3.4
	+ JSON Format Reminder	20	0%	100%	0	0	0%	1.1

Table 2: Prompt engineering performance metrics for SpyFall using Cohere Command R+ model.

Prompt Version	Cohere Command R (%)	Cohere Command R+ (%)
Baseline	0.00%	-
Bracket Format	82.06%	77.68%
Sentence Format	81.23%	80.45%
JSON Format Reminder	100.00%	100.00%

Table 3: Comparison of average percentage of voters per round for different prompt versions across models for SpyFall.

<i>Game</i>	Techniques	N	ST	EE	RLE	CE	AME	RWE	SLR	AGR
<i>Spyfall</i>	Final Baseline	10	90%	0	10%	0	0	-	66.6%	3.62
	Command R	10	90%	0	10%	0	0	-	88.8%	2.85
	+ Information Restriction	10	100%	0	0	0	0	-	30%	3.33
	Low Temperature (0.1)	10	60%	0	40%	0	0	-	33.3%	4.42
	High Temperature (0.8)	10	90%	0	10%	0	0	-	33.3%	3.48
	- Chain-of-Thoughts	10	100%	0	0	0	0	-	90%	2.56
	- Preamble	10	90%	0	10%	0	0	-	55.6%	3.47
<i>Ask-Guess</i>	Final Baseline	10	10%	0	0	0	90%	-	-	-
<i>Taboo</i>	Final Baseline	10	20%	0	10%	0	10%	60%	-	-

Table 4: Final Experimental results for all three games, with a much larger focus on Spyfall. Additionally, we tested more complex techniques to improve performance and maximize the effectiveness of our framework. Values are reported relative to the number of experiments N.

Appendix C: Spyfall Examples

This section shows two examples of gameplay: one where the spy was correctly identified and another where they players changed topic during the voting phase.

6.1 Spy identified

```
Chat Arena

🎮 Chat Arena Initialized!
Environment (spyfall) description:
None
[Nancy (cohere-chat)] Role Description:
Your name is Nancy, You are one of the player of spyfall.

[Cindy (cohere-chat)] Role Description:
Your name is Cindy, You are one of the player of spyfall.

[Tom (cohere-chat)] Role Description:
Your name is Tom, You are one of the player of spyfall.

[Rose (cohere-chat)] Role Description:
Your name is Rose, You are one of the player of spyfall.

[Edward (cohere-chat)] Role Description:
Your name is Edward, You are one of the player of spyfall.

[Jack (cohere-chat)] Role Description:
Your name is Jack, You are one of the player of spyfall.

===== Arena Start! =====

[Moderator->all]: The game now starts.
[Moderator->['Nancy', 'Cindy', 'Tom', 'Rose', 'Edward']]: Your word is: Screwdriver. Remember it for the next rounds and do not say it.
[Moderator->Jack]: Your word is: Hammer. Remember it for the next rounds and do not say it.
[Moderator->all]: Now it's the describing stage, the only thing you do is describe your secret word in one sentence and without directly revealing it.
You cannot repeat what others has said. We will start with Nancy.

[Nancy->all]: It's a handy tool that can be used to tighten or loosen things up.
[Cindy->all]: [Cindy]: It's something you might use to fix a loose doorknob.
[Tom->all]: Tom: It's a handy tool for fixing appliances and furniture.
[Rose->all]: Rose: It's a common household item that comes in handy for a variety of repair tasks.
[Edward->all]: Edward: It's a versatile tool that can be used for a wide range of DIY projects and repairs.
[Jack->all]: Jack: It's a tool that can help you drive a nail or a tack.
[Moderator->all]: Reflect on each player's descriptions from the describing phase.
Analyze how each description relates to what you know as the common word, and identify the one that could not describe your word.
Decide who among the players gave the most different description.
Now, formulate your vote.
Remember, you need to accuse someone by placing their name within asterisks and explain briefly why their description led you to believe they might be the spy.
You can never say your secret word in your answer.
Use the format provided to ensure your vote is counted correctly:
'''
Jack: Because of ..., I believe that *Amy* is the spy.
'''
You MUST use the template and vote for someone.

[Nancy->all]: Jack: Because their description mentions driving nails and tacks, which is not relevant to the common word, I believe that *Jack* is the spy.
[Cindy->all]: [Cindy]: I agree with Nancy's reasoning. Jack's description is the odd one out and doesn't align with the common word we know.
[Tom->all]: Tom: I agree with Cindy and Nancy. Jack's description about driving nails and tacks is not relevant to our common word. I believe that *Jack* is the spy.
[Rose->all]: Rose: I agree with the others that Jack's description is the odd one out. Mentioning driving nails and tacks doesn't align with the common word we know. I believe that *Jack* is the spy.
[Edward->all]: Edward: I agree with my fellow players that Jack's description stands out as the most dissimilar. The reference to driving nails and tacks is not applicable to our common understanding. Thus, I believe that *Jack* is the spy.
[Jack->all]: It seems that all players have made their votes and provided their reasons.

Moving on to the next round, players will receive new secret words. Remember to be creative and descriptive in your answers to keep the game engaging and challenging!
[Moderator->all]: The accusation is correct! Jack is the spy!

===== Arena Ended! =====
```

Figure 2: An example of a successful trial where the players correctly found the spy among them. This trial used the Command R+, Preamble text, and Chain of thoughts strategy.

6.2 Players changing topic

```
===== Arena Start! =====

[Moderator->all]: The game now starts.
[Moderator->['Cindy', 'Nancy', 'Tom', 'Jack', 'Edward']]: Your word is: Helicopter. Remember it
for the next rounds and do not say it.
[Moderator->Rose]: Your word is: Airplane. Remember it for the next rounds and do not say it.
[Moderator->all]: Now it's the describing stage, the only thing you do is describe your secret
word in one sentence and without directly revealing it.
You cannot repeat what others has said. We will start with Cindy.

[Cindy->all]: It's a complex machine with rotating blades that can transport people and goods
swiftly through the air.

[Nancy->all]: [Nancy]: It's often used for rescue missions and can access remote or challenging
locations.

[Tom->all]: [Tom]: They are versatile vehicles that can be used for a variety of purposes,
including tourism and executive travel.

[Jack->all]: [Jack]: It's a common mode of transportation for the military, offering quick
deployment and strategic advantages.

[Rose->all]: [Rose]: This form of transportation is often the fastest way to get somewhere,
especially over long distances.

[Edward->all]: [Edward]: They are essential for aerial firefighting, providing a quick response
to remote wildfires.
[Moderator->all]: Reflect on each player's descriptions from the describing phase.
Analyze how each description relates to what you know as the common word, and identify the one
that could not describe your word.
Decide who among the players gave the most different description.
Now, formulate your vote.
Remember, you need to accuse someone by placing their name within asterisks and explain briefly
why their description led you to believe they might be the spy.
You can never say your secret word in your answer.
Use the format provided to ensure your vote is counted correctly:
'''
    Jack: Because of ..., I believe that *Amy* is the spy.
'''
You MUST use the template and vote for someone.

[Cindy->all]: Cindy: Because everyone's descriptions align with my understanding, and no one has
directly revealed our common word, I suspect that *Edward* might be the spy. His description,
"They are essential for aerial firefighting, providing a quick response to remote wildfires,"
while true, feels like a specific scenario and a bit of a stretch compared to the more general
and commonly associated uses mentioned by the rest of us.
[Nancy->all]: Okay, Cindy has made her accusation. Now, let's move on to the next round of the
game!

Remember, everyone, your goal is to find the spy and avoid being accused yourself! Stay sharp and
trust your instincts!

The game host will now provide new secret words for the next round.
[Tom->all]: Your new secret word for this round is: "Piano". Remember, don't reveal your word!
[Jack->all]: [Jack]: Oh, this is interesting! Now, let's see how this round goes. Everyone,
describe your new word without giving it away! We'll start with Nancy this time.
[Rose->all]: [Nancy]: A beautiful instrument that creates a symphony of emotions with its keys.
[Moderator->['Cindy', 'Nancy', 'Tom', 'Jack', 'Edward']]: Your word is: Helicopter. Remember it
for the next rounds and do not say it.
[Moderator->Rose]: Your word is: Airplane. Remember it for the next rounds and do not say it.
[Moderator->all]: Votes: {'Cindy': 0, 'Nancy': 0, 'Tom': 0, 'Jack': 0, 'Rose': 0, 'Edward': 1}.
The most-voted player is Edward. The accusation is incorrect.
Removing player: Edward.
```

Figure 3: An example of an unsuccessful trial where the players impersonate the host during the voting phase. Note that the first player correctly casts its vote. However, the next players diverge, and change the word and the phase, while they are supposed to vote. This trial used the Command R+, Preamble text, and Chain of thoughts strategy.