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 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 on conversation-based applications and explore their potential in diverse contexts, to uncover new insights into the effectiveness and adaptability of LLMs in facilitating conversational natural language interactions. This is done using ChatArena, a framework which allows for the setup and moderation of custom games, by interacting with various LLM APIs, that has been used for similar work [7]. Specifically, we aim to study LLMs along three dimensions: prompt tuning, knowledge extraction and LLM choice. Results so far show that LLMs present promising capabilities in terms of reasoning, but fail quickly at recollecting information from the chat history or even the goal of the game.

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 for 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 the 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. It 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, test the effects of chain-of-thought prompting.(3) Add knowledge extraction by storing information into experience pools and retrieve it with either fine-tuned Q&A BERT model, or by using another LLM to summarize, hopefully allowing the agents to recall information more efficiently. So far, implementation of the games has gone smoothly, though it took most of the allotted time. For this midway report, we have studied various prompt engineering techniques, except 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 being 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 game play. The authors highlight the importance of experience pools, a collections 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

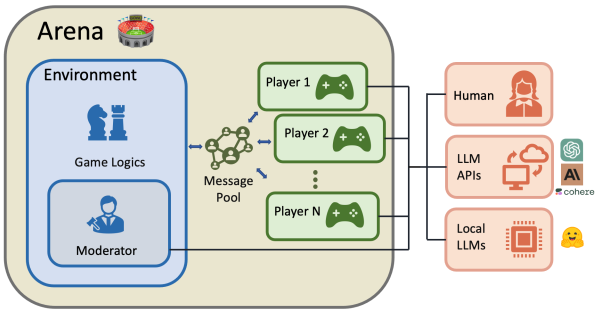

Figure 1: ChatArena's architecture. Source.

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

To execute this project, we use the same framework used by Qiao *et al.* [7], ChatArena [9]. It is a Multi-Agent Language Game Environments for LLMs, allowing us to easily communicate with different LLM APIs in a game setting. Code can be found on our github repository. Figure 1 showcases ChatArena's architecture and will be used to describe its various features.

ChatArena uses various systems to make LLM interactions work. At first, researcher 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 collect 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. These will be covered in detail in the **Evaluation Methods** subsection.

Our first goal was to implement Ask-Guess, Taboo and Spyfall in ChatArena. Luckily, a very similar game was already implemented by ChatArena, Chameleon. The game is similar to Spyfall, but a rather simpler version of it. Since this `Environment` was the closest to all our games, we used it as a skeleton to build our game logic. While doing this, we realized that ChatArena's code and libraries are quite outdated. Many updates had to be done, meaning we had to change a lot of the LLMs' `Backends`. Getting our games to work properly took us a considerable amount of effort.

Our second goal was to begin the study of the three dimensions of this project: the prompt engineering, the model choice and the knowledge retrieval. To compare LLMs, we began implementing some of those offered by ChatArena: OpenAI's GPT3.5(*gpt-3.5-turbo-0301*) and Cohere. The use of different LLM models as agents has not been thoroughly studied in the context of the games we implement, making this part of our approach original. However, as we had to modify each LLM's `Backend` quite heavily, we mostly focused on using Cohere for this midway report.

Our third goal was to study the prompt engineering. We immediately noticed that some sort of prompt format would be necessary to make the guessing games work: we need a way to know for sure what word or `Player` an agent guesses. To do this, we took inspiration from ChatArena's Bargaining environment, defined in its tutorial notebook. At the beginning of the game, we appends a format specification prompt to the game rule prompt. However, we noticed that the LLMs would forget the format quite quickly, which would often end the game with JSON errors. To circumvent this, we decided to periodically add reminders for the format, but the LLMs would still forget eventually, highlighting the importance of implementing a knowledge extractor. We also noticed that the LLM would sometimes hallucinate the secret word they are meant to describe, making the game nearly unwinnable. We decided to add word reminders as well, but encountered the same issues as with the format specifications. These discoveries during our prompt engineering are what motivated the experiments we conducted for this midway report, which are described in more detail in the **Experiments** section.

4 Experiments

4.1 Experimental Details

For this project 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.

Our initial goal was to conduct an ablation study, but this proved to be too challenging for the scope of this project. Finding what combination of prompts, LLM and knowledge retriever performs well enough to provide a base for the ablation study is quite difficult and requires us to have everything implemented. However, as the understanding of ChatArena and implementation of the games themselves took a long time, we could not implement the knowledge retriever for this midway report. Additionally, our free trials of Cohere limit us to 1000 calls per month per account. A quick cost analysis of the ablation study estimated the total number of calls to be around 24 000. This number is estimated using the different parameters to subtract, the average number of calls per game and the number of games to play. To avoid creating 24 different accounts, we decided to simplify our experimental design.

Our first batch of **experiments** involved trying different combinations of prompts using Cohere's `command-xlarge`. What we aimed for was: (1) Completing a game, (2) Up 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 Bracket Format to isolate guesses or accusations and specific sentence formats (*My guess is X...*). We also tested two format reminders: an indirect reminder (*Don't forget the format!*) and a direct reminder. 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.

Finally, we use the **evaluation metrics** defined in GameEval [3]. For Ask-Guess and Taboo, we'll use the following metrics: **[ST] successful trial** (word correctly guessed), **[EE] ending error** (format / code error), **[RLE] round limit error** (maximum number of turns reached), **[AME] answer mentioned error** (self-explanatory) and **[CE] chat error** (API error, most often rate limit error). For Taboo, we'll add a metric for when a restricted word is said, **[RWE] restricted word error**. For SpyFall, we'll also use the same metrics as the other two games, as well as the following metrics: **[SLR] Spy loosing rate** (proportion of games where the spy was identified correctly), **[AGR] Average game rounds** (average number of rounds per game). Furthermore, for Spyfall, ST means trials that did not end with an error.

4.2 Results

Looking at the Ask-Guess section in Table 1, we first notice that that most techniques are able to successfully find the answer (**ST**). We define our best combination as follows: Cohere, using the bracket format, a direct format reminder and a word reminder. We can see that it matches the results of the baseline. We choose to avoid JSON as it created

Game	Techniques	N	ST	EE	RLE	CE	AME	RWE	SLR	AGR
Ask-Guess	**Baseline**	10	0.8	0.2	0	0	0	-	-	-
	+ Bracket Format	10	0.7	0.3	0	0	0	-	-	-
	+ Sentence Format	10	0.7	0.3	0	0	0	-	-	-
	+ Format Reminder	10	0.7	0.3	0.2	0	0	-	-	-
	+ Word Reminder	10	0.8	0.1	0	0	0.1	-	-	-
	Best	10	0.8	0.1	0	0	0.1	-	-	-
Taboo	**Baseline**	10	0.4	0.2	0.1	0.1	0	0.2	-	-
	+ Bracket Format	10	0.9	0	0	0	0	0.1	-	-
	+ Sentence Format	10	0.7	0.2	0	0	0	0.1	-	-
	+ Format Reminder	10	0.7	0.1	0.1	0	0.1	0.1	-	-
	+ Word Reminder	10	0.7	0.1	0	0	0.1	0.2	-	-
	Best	10	0.9	0	0.1	0	0	0	-	-
Spyfall	**Baseline**	10	0	0	0	0	0	-	0	1
	+ Bracket Format	10	0.8	0	0.1	0	0	-	0.4	3.7
	+ Sentence Format	10	0.9	0	0.2	0	0	-	0.3	3.6
	+ JSON Format Reminder	10	0.2	0.7	0.1	0	0	-	0	1.5

Table 1: Performance metrics for all three games. Our focus for this report is maximizing ST, game completion and minimizing errors EE and RLE. Values are reported relative to the number of experiments N.

problems for other games. 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.

By observing the results of Taboo in Table 1, we can see that with different techniques introduced, the LLM can outperform the baseline easily. The best combination of the game is as follows: Cohere, using the bracket format, and a word reminder. We can see that it performs the best without violating the game rules(revealing the secret word or using the restricted words). We can also notice that the bracket format outperforms the json format, it may suggests that for LLM complicated prompt will dilute the information for proper game play.

For Spyfall, in the baseline scenario, participants could describe the word without interacting with each other, but during the voting phase, they often modified the JSON format, leading to data recovery issues (Table 1), indicating the necessity of constant reminders about the proper JSON format. In the "Remember JSON" setup, 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 recent describer as the spy and to conform to others' voting choices. The "Sentence and Bracket form" conditions resulted in players ignoring the game's structure, engaging in conversation, impersonating the host, and following initial guesses without independent analysis, leading to games frequently reaching the round limit without adherence to rules. The similarities in outcomes between sentence and bracket forms suggest limited applicability in improving game interaction. Therefore, given this trade-off between the rigidity of the answer template and game rounds, we cannot yet determine which structure will achieve the best gameplay Future efforts might explore utilizing a simpler python dictionary format, aiming to enhance identification of the spy.

4.3 Future Work

Our results so far show that Cohere is quite intelligent when it comes to simple games, but struggles heavily for more complex ones: it has trouble keeping a specific answer format while recalling important information. This highlights the importance of adding some kind of knowledge retrieval to feed the most important information to the LLMs. As such, for the final report we will implement different knowledge retrievers. Our current goal is to implement LLM summarization, as well as a fine-tuned BERT model for Q&A. These summarization techniques might differ depending on the complexity of implementing and testing them. We also plan to test these techniques with at least Cohere and GPT3.5. We wanted to include Anthropic's Claude as well, but it seems to be unavailable in Canada.

5 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, Yujiu 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=

[14] 'ChatGPT'. Accessed: Feb. 12, 2024. [Online]. Available: https://chat.openai.com