AIWolfDial 2023: Summary of Natural Language Division of 5th International AIWolf Contest

Yoshinobu Kano[1]*, Neo Watanabe[1], Kaito Kagaminuma[1], Claus Aranha[2], Jaewon Lee[3], Benedek Hauer[3], Hisaichi Shibata[4], Soichiro Miki[4], Yuta Nakamura[4], Takuya Okubo[3], Soga Shigemura[3], Rei Ito[3], Kazuki Takashima[3], Tomoki Fukuda[3], Masahiro Wakutani[3], Tomoya Hatanaka[3], Mami Uchida[3], Mikio Abe[5], Akihiro Mikami[5], Takashi Otsuki[5], Zhiyang Qi[6], Michimasa Inaba[6], Kei Harada[6], Daisuke Katagami[7], Hirotaka Osawa[8], Fujio Toriumi[3],
[1]Shizuoka University, [2]University of Tsukuba, [3]The University of Tokyo,
[4]The University of Tokyo Hospital, [5]Yamagata University,
[6]The University of Electro-Communications, [7]Tokyo Polytechnic University, [8]Keio University

Abstract

We held our 5th annual AIWolf international contest to automatically play the Werewolf game "Mafia", where players try finding liars via conversations, aiming at promoting developments in creating agents of more natural conversations in higher level, such as longer contexts, personal relationships, semantics, pragmatics, and logics, revealing the capabilities and limits of the generative AIs. In our Natural Language Division of the contest, we had six Japanese speaking agents from five teams, and three English speaking agents, to mutually run games. By using the game logs, we performed human subjective evaluations and detailed log analysis. This paper is jointly written by the organizers and the participants. We found that the entire system performance has largely improved over the previous year, due to the recent advantages of the LLMs. However, it is not perfect at all yet; the generated talks are sometimes inconsistent with the game actions, it is still doubtful that the agents could infer roles by logics rather than superficial utterance generations. It is not explicitly observed in this log but it would be still difficult to make an agent telling a lie, pretend as a villager but it has an opposite goal inside. Our future work includes to reveal the capability of the LLMs, whether they can make the duality of the "liar", in other words, holding a "true" and a "false" circumstances of the agent at the same time, even holding what these circumstances look like from other agents.

1 Introduction

Recent achievements of generation models, e.g. ChatGPT (OpenAI, 2023), are gathering greater attentions. However, it is not fully investigated whether such a huge language model can sufficiently handle coherent responses, longer contexts, common grounds, and logics. Our shared task, AIWolfDial 2023, is an international open contest for automatic players of the conversation game "Mafia", which requires players not just to communicate but to infer, persuade, deceive other players via coherent logical conversations, while having the role-playing non-task-oriented chats as well. AIWolfDial 2023 is one of the INLG 2023 Generation Challenges for this year. We believe that this contest reveals not just achievements but also current issues in the recent huge language models, showing directions of next breakthrough in this area.

"Are You a Werewolf?", or "Mafia" (hereafter "werewolf game"), is a communication game conducted solely through discussion. Players must exert their cognitive faculties fully in order to win. In the imperfect information games (Bowling et al., 2015), players must hide information, in contrast to perfect information games such as chess or Go (Silver et al., 2016). Each player acquires secret information from other players' conversations and behavior and acts by hiding information to accomplish their objectives. Players are required persuasion for earning confidence, and speculation for detecting fabrications.

We propose to employ this werewolf game as a novel way of evaluations for dialog systems. While studies of dialog systems are very hot topics recently, they are still insufficient to make natural conversations with consistent context, or with complex sentences. One of the fundamental issues is

*Correspondence to kano@kanolab.net

Proceedings of the 16th International Conference on Natural Language Generation: Generation Challenges, pages 84–100

a lack of an appropriate evaluation. Because the Werewolf game forces players to deceive, persuade, and detect lies, neither inconsistent nor vague response are evaluated as "unnatural", losing in the game. Our werewolf game competition and evaluation could be new interesting evaluation criteria for dialog systems, but also for imperfect information game theories. In addition, the werewolf game allows any conversation, so the game includes both task-oriented and non-task-oriented conversations.

We have been holding an annual series of competition to automatically play the Werewolf game since 2014 (Toriumi et al., 2017), as the AIWolf project [1]. Our competitions were linked with other conferences such as the competitions in IEEE Conference On Games (CoG), ANAC (Automated Negotiating Agents Competition) (Aydoğan et al., 2020)(Lim, 2020) in International Joint Conference on Artificial Intelligence (IJCAI), Computer Entertainment Developers Conference (CEDEC), etc., in addition to our AIWolfDial 2019 workshop at INLG 2019 (Kano et al., 2019). These mean that our contests attract interests from communities of many areas including dialog system, language generation, task- and non-task-oriented conversations, imperfect information game, human-agent interactions, and game AI.

We have been providing two divisions in the contests: the protocol division and the natural language division. The protocol division uses our original AIWolf protocol which is designed for simplified language specific to the Werewolf game player agents. In the natural language division, player agents should communicate in the natural languages (English or Japanese). The natural language division is simple and natural goal of our project, but very difficult due to its underlying complexity of human intellectual issues. We focus on this natural language division in this report.

In the natural language division of our contest, we ask participants to make self-match games as preliminary matches, and mutual-match games as final matches. Agents should connect to our server to match, i.e. participants can run their systems in their own servers even if they require large computational resources. The game logs are evaluated by human subjective evaluations.

Eight agents of seven teams participated in this AIWolfDial 2023 shared task. Because our games are held by five players, we held a mutual match game in the Japanese language by six agents from five teams, and another mutual match game in the English language by three teams.

In the following sections, we explain the game regulations of the AIWolf natural language division in Section 2, detailed system designs for each agent in Section 3, results of subjective evaluations in Section 4.2 followed by discussions in Section 5, finally conclude this paper in Section 6. This paper is jointly written by the organizers and the participants, i.e. Section 3 is written by each participant, the other sections are by the organizers, thus "we" stand for the organizers except for i.e. Section 3.

2 Werewolf Game and Shared Task Settings

We explain the rules of the werewolf game in this section. While there are many variation of the Were-wolf game exists, we only explain the our AIWolfDial shared task setting in this paper.

2.1 Player Roles

Before starting a game, each player is assigned a hidden role from the game master (a server system in case of our AIWolf competition). The most common roles are "villager" and "werewolf". Each role (and a player of that role) belongs either to a villager team or a werewolf team. The goal of a player is for any of team members to survive, not necessarily the player him/herself.

There are other roles than the villager and the werewolf: a seer and a possessed. A seer belongs to the villager team, who has a special talent to "divine" a specified player to know whether the player is a human or a werewolf; the divine result is notified the seer only. A possessed belongs to the villager team but his/her goal is win the werewolf team.

A game in the AIWolfDial 2023 shared task have five players: a seer, a werewolf, a possessed, and two villagers.

2.2 Day, Turn and Winner

A game consist of "days", and a "day" consists of "daytime" and "night". During the daytime phase, each player talks freely. At the end of the daytime, a player will be executed by votes of all of the remained players. In the night phase, special role players use their abilities: a werewolf can attack and kill a player, and a seer can divine a player.

[1] http://aiwolf.org/

In the shared task, Day 0 does not start games but conversations e.g. greetings. A daytime consists of several turns; a turn is a synchronized talks of agent, i.e. the agents cannot refer to other agents' talks of the same turn. We set a maximum limit of ten turns per day in AIWolfDial 2023.

The victory condition of the villager team is to execute all werewolves, and the victory condition of the werewolf team is to make the number of villager team less than the number of werewolf team.

2.3 Talk

An AIWolf agent communicates with an AIWolf server to perform a game. Other than vote, divine, and attack actions, an agent communicates in natural language only.

We intend to design our shared task to be played by physical avatars in real time in future, rather than to limit to communications in the written language. Therefore, a talk text should be able to pronounce verbally, while symbols, emojis, and any other non-pronounceable letters are not allowed.

Because of the same reason, we set the maximum response time to be five seconds in the prior contests. However, we set the response timeout to be five minutes in this year, because we expected that many participants would use external web APIs such as ChatGPT, which could cause longer response time. We hope to shorten this talk timeout again in future.

In this text-base multiple player game, it is not clear that an agent speaks to which specific agent, or speaks to everyone. Human players can use their faces and bodies to point another player. In order to specify which agent to speak to, an agent may insert an anchor symbol (e.g. ">>Agent[01]") at the beginning of its talk.

Player agents are asked to return their talks agent by agent in a serial manner, which order is randomly changed every turn. This is different from the humans' verbal turn taking in that humans can speak (mostly) anytime.

3 Participant Systems

Six agents from five teams participated our shared task in the Japanese language, which agent names are **am**, **ChatWolf**, **k2b_ara**, **k2b_shigemura**, **kanolab**, **sUper_IL**). Three agents from three temas participated in the English language, which agent names are **HowlGPT**, **MIV**, and **kanolab**, where **kanolab** is their English version of their original Japanese agent. Most of the agents used ChatGPT in their system, while its usage is different between the agents; **ChatWolf** uses another LLM, **am** employed a rule-based system.

We, the organizers, provided a template agent code in Java and Python, in addition to the server codes.

We describe each participant system in an alphabetical order in the following subsections. where "we" stands for the corresponding participants, only limited in this section.

3.1 am

am is created by Mikio Abe, and Akihiro Mikami in Yamagata University.

We used an agent by m_cre[2], the runner-up in the Natural Language Division of the 4th AIWolf Competition held in 2018, which was a rule-based agent system. For natural language analysis, m_cre use the morphological analyzer "Juman[3]", "KNP[4]" which performs syntactic analysis, case analysis and reference resolution of Japanese sentences, and a dictionary to which we added special words that appear in Werewolf games. Our goal was to build an agent that could speak naturally like a human by adding the utterance patterns to the m_cre agent, which are seen in a Werewolf game between humans. We have added the following five elements.

The reaction against a CO of Seers When another player makes a CO (Coming Out of roles) of Seers, our agent responds to that player.

The reaction against a report of divination results When another player makes a report of divination results, our agent responds to that player.

The utterance declaring the candidates for voting at the end of the day Our agent declares who to vote for at the end of the day.

The utterances when distressed If there is nothing to say, our agent says something like "ummm".

The questions that follow the flow of the game Our agents speak when they have a question about a game situation. For example, we ask other agents who is the Seer at the beginning of the day, or we ask the agent who divined our agent why he divined us.

[2]https://github.com/mcre/aiwolf-4th-nlp
[3]https://nlp.ist.i.kyoto-u.ac.jp/?JUMAN
[4]https://nlp.ist.i.kyoto-u.ac.jp/?KNP

3.2 ChatWolf

ChatWolf was developed by Hisaichi Shibata, Soichiro Miki, Yuta Nakamura of the University of Tokyo Hospital.

3.2.1 Design

We designed the agent to have two models: the talk model and the voting model. The talk model can talk with other agents while the voting model can vote (including attack voting) for the other agents. These models run separately, and respond to queries from the game server. We did not handle the divination in the game with models and ChatWolf divines other agents at random. We adopted one of the LLMs (Large Language Models) open to the public from CyberAgent (Open-Calm-7b[5]) both for the talk model (we used `AutoModelForCausalLM` in hugging face transformers) and the voting model (we used `AutoModelForSequenceClassification` in hugging face transformers). Each model contains approximately 7 billion parameters. For the talk model, to adapt the LLM to the Werewolf game, we executed a LoRA (Low-Rank Adaption for LLM (Hu et al., 2021)) tuning with Japanese Werewolf game logs[6] newly collected by the developers of ChatWolf. To systematically collect the game logs, we developed UTRAD Werewolf, which is a web browser-based platform to play the Werewolf game with or without artificial intelligence agents. We collected 48 game logs (not open to the public). For details of the log collection, see (Shibata et al., 2023). For conditional text generation with the talk model, we enabled the sampling to generate diverge texts. The temperature of the sampling was experimentally set to 0.7 and the repetition penalty to 1.05. For the voting module, which can interpret intermediate Werewolf game logs in Japanese and vote for agents to attack or eliminate, we attached a classification head on the LLM and trained parameters in it with the same logs. The training method with the classification head is well known as transfer learning whereas LoRA is one of the fine-tuning methods. The prompt (input) to the both models contained instructions of the game, the latest 15 talks by all the participants, the role and agent number of ChatWolf, and the previous voting result if it exists. We executed the inference with those models on a single NVIDIA RTX-A6000 graphics processing unit with 48 GB memory.

3.2.2 Background of ChatWolf design

Because the evaluation criteria of the regulation for the natural language division do not refer to the win ratio of the Werewolf game, we decided to give priority on fluently talking natural language than to wining the game.

If we adopt GPT-4 (Generative Pre-trained Transformers 4; (OpenAI, 2023)) a.k.a. ChatGPT[7], it is expected to very fluently talk natural languages. However, we specified disadvantages of GPT-4 as follows:

- The tuning of the agents ends up in prompt-engineering and not very interesting.
- Many teams would participate in the contest with GPT-4. In the case, agents powered by GPT-4 must be highly tuned up to differentiate with the other GPT-4-based agents.
- Agents powered by GPT-4 could not be significantly novel.
- Should the server of GPT-4 be temporarily down, we cannot play the game at all.

On the basis of this insight, we decided to develop ChatWolf running on a local computer.

3.3 HowlGPT

HowlGPT is, when boiled down, a simple 3 layer chain-of-thought reasoner.

At the lowest layer are the language models. An abstraction layer allows our system to direct templated patterns of (recursive) queries either to a locally running transformer, a distributed "Petals" backend, or the OpenAI API (gpt-3.5-turbo-16k).

Context lengths are fixed to 8K tokens for prefix input and 8K generation when using OpenAI. Chunk sizes of 16-64 are used for top-k generation. These calls are used for "large context" reasoning and decision making.

Local models are used for reasoning and decision-making. Local models deployed include: `mpt-7b-storywriter`,

[5]https://huggingface.co/cyberagent/open-calm-7b

[6]The log collection was partially supported by JST, CREST Grant Number JPMJCR21M2, including the AIP challenge program (Necessary conditions that personal claims are objective facts; PI: H. Shibata), Japan. We thank many participants who played Werewolf again and again to generate game logs.

[7]https://chat.openai.com/auth/login

mpt-7b-instruct, gpt-medium, gpt-xxl, Nous-Hermes-13b, vicuna-7b-1.1.

For local models, context lengths are sized appropriately to the model in use.

Different models were used for their different strength. For example, gpt and bloom are largely used for knowledge-management and summarization while mpt is used for introducing creative elements to roleplay effort and Nous is used for "smaller context" commonsense reasoning.

At the second layer, calls to/between these models are coordination by a query language runtime.

Two or three values from a large table of "character descriptors" are chosen for each instance, to give the agent some personality direction. An initial description and small "backstory" is generated for the instance character and given along with each query to the language models.

Four "temperature" values are also chosen randomly - these are used to control sampling and beam search for queries. These values are a "summarization" temperature which controls sampling for internal summarization processes, a "thought" temperature which controls sampling during chain-of-thought queries, a "choice" temperature (fixed to 0 for the competition play but allowed to vary during training) which controls temperature when making vote decisions, and a "talking" temperature for variability of spoken statements. (This makes agents less predictable and their roles less identifiable to other agents.)

A simple sockio.aio socket is opened to the game server and a small FSM handles the game protocol messaging. The events observed are translated from their json representation into english representations, embedded and saved into a knowledge retrieval store, and stored in in-memory structures for use in prompt constructions. Whenever new conversation is seen, or new context is integrated, lengths of queries are re-checked to confirm that they still fit within the language model context widths. If thresholds are passed (for example, 80% of the gpt-3.5-turbo-16k's 8k input window is consumed) then the agent summarizes existing context information before appending the newly received information.

A third RNN model layer is used as a supervisory, to modify/reject any outputs from the models which are deemed likely low quality or otherwise problematic. This supervisory layer was trained with a combination of self-play and human supervision.

3.4 k2b_ara

k2b_ara is created by Takuya Okubo, Kazuki Takashima, Tomoya Hatanaka, Mami Uchida, Rei Ito in the University of Tokyo.

We have developed an agent that performs the following functions using BERT (Devlin et al., 2018), GPT-3 (text-davinci-003) and GPT-4 (gpt-4-0613) (OpenAI, 2023).

- Infer roles
- Plan strategies
- Accept or reject requests
- Answer questions

To actualize these features, our agent is composed of eight different modules:

RoleEstimationModule Estimates the role distribution of each agent based on game information (the number of each role, talk content, and divination results if available) utilizing BERT[8]. It considers the sum of attentions from the other tokens to the first token ([CLS]) across all 12 multi-head attention layers in BERT as the basis for these estimates and makes a list of word-attention pairs for other modules to use[9].

RoleInferenceModule Infers the roles of each agent by GPT-4, which is given the estimation basis (a list of word-attention pairs obtained from the fine-tuned BERT model above) and an certain agent's role distribution derived from the **RoleEstimationModule**. It receives a list of word-attention pairs as the basis for inference, selecting the top 10 phrases from this list as the rationale. Then, it gets the response (inference result) from the GPT-4 with the prompt, which includes the above rationale, followed by an additional prompt, "*Please infer the agent's role based on the above information and state the logical reason why you think so.*"

[8] we used a pretrained model https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking and fine-tuned it with about 500 werewolf game logs scraped from an online werewolf game server https://ruru-jinro.net/

[9] i.e. if *"hello, I divined Agent[01] werewolf"* is in the talk history, then the list includes [("hello",0.01),("I",0.03),("divined",0.9),("Agent[01]",0.1),("werewolf",0.7)]

StrategyModule Determines whom to vote for, whom to divine, and how to persuade other agents with GPT-4, based on the role estimation results obtained from **RoleEstimationModule** and **RoleInferenceModule**. For instance, it selects the agent with the highest probability of being a werewolf or a possessed as the voting target calculated by **RoleEstimationModule**, and generates the reason for the role estimation using **RoleInferenceModule**. These information are then provided to GPT-4, with an additional prompt, *"Based on the above reasons, please persuade other agents to vote for Agent[number]."*, to generate a persuasive sentence. The divination target is also chosen as the agent with the highest probability of being a werewolf or a possessed.

RequestProcessingModule Categorizes the requests from others into three types (whom to vote for, whom to divine, or other requests) using GPT-3 with few shot prompt, such as *"Q: Shall we vote for Agent[01] A: whom to vote for"*, *"Q: Would you mind divining Agent[01] next? A: whom to divine"* and *"Q: Could you believe me a villager? A: other requests"*. Then, it determines whether to comply with the request using the **StrategyModule** and **RoleEstimationModule** and generates the response based on rule-based procedures.

QuestionProcessingModule Classifies the questions from others into four types (past behaviour, future plan, its role prediction, and other questions) with GPT-3 with few shot prompt, such as "Q: Who do you think as a posessed? A: other requests", "Q: Why did you vote him? A: whom to vote for" and "Q: Could you tell me the reason you divined Agent[01]? A: whom to divine". Then, it determines the response to the question using the **StrategyModule** and **RoleEstimationModule** and generates the response based on rule-based procedures.

InfluenceConsiderationModule Determines whether the statement of a certain agent is directed at itself using GPT-3 with few shot prompt, such as *"Q: I think Agent[01] is a werewolf A: no"*, *"Q: please believe me a seer! A: request"* and *"Q: Who do you all vote for? A: question"*. If it's a request or a question, then calls **RequestProcessingModule** or **QuestionProcessingModule** respectively and returns the response to its caller.

SpeakerModule Transforms personality-less speech content into personality-rich speech content using GPT-4.

IntegrationModule Integrates above seven modules and then sends information to the server. For instance, when in the talk phase, this module selects speech content from **InfluenceConsiderationModule** if it is not empty. Otherwise, it selects speech content from **StrategyModule**. Then, it enriches the speech content with **SpeakModule** and send it to the server. When in the vote phase (divine phase), it asks **StrategyModule** for whom to vote for (divine) and send it to the server.

3.5 k2b_shigemura

k2b_shigemura is created by Soga Shigemura, Tomoki Fukuda, and Masahiro Wakutani in the University of Tokyo.

3.5.1 Design

We utilized a mixed model approach, using both `gpt-4-0613` and `gpt-3.5-turbo-0613`, which are OpenAI's models[10]. This allowed us to leverage the precision of GPT-4 and the speed of GPT-3.5. In terms of parameters, we found that the best results were achieved by retaining default values, aside from `max_tokens` which we set at `300` (`temperature=1.0`, `top_p=1.0`, `presence_penalty=0`, `frequency_penalty=0`). With the exception of prompts related to conversational expressions, most of the prompts are written in English. This is because prompts in English tend to have higher accuracy and logical coherence, while prompts related to conversational expressions in Japanese are better conveyed in Japanese, resulting in greater naturalness and creativity by using the same language.

We also incorporated the OpenAI's **Function calling** feature, which was made public in June 2023[11]. This Function calling feature can prevent unintended responses when making requests to OpenAI's API, by sending a list of functions specified with JSON schema[12] that defines the types (such as string, integer, enum) as parameters.

[10] https://platform.openai.com/docs/models/overview

[11] https://openai.com/blog/function-calling-and-other-api-updates

[12] https://json-schema.org/understanding-json-schema/

3.5.2 Model

Conversation Summarization Model Since it was impossible to incorporate all game conversation into the prompt, we developed a **Conversation Summarization Model** using `gpt-3.5-turbo` to summarize each agent's statement. To prevent usage of the term "agent" within the dialogue and to simplify the conversation history, we converted "Agent[01]" to "[1]" in advance, for example, then reverted "[1]" back to "Agent[01]" after LLMs processed. Inserting this **Conversation Summarization Model** allowed us to transform self-perspective statements like "*I AM ABSOLUTELY THE SEER!*" into sentences like "*Agent[01] claims to be the seer.*" This adjustment reduced the influence of blindly trusting other agents' statements when generating our own dialogue or making decisions. It also prompted the addition of logical coherence to our statements, such as "*... because Agent[03] has revealed Agent[05] as innocent*".

Dialogue Generation Model We employed GPT-4 for dialogue generation to ensure logical coherence and richness of expression. We provide the following information to the model:

- Explanation of the rules of the werewolf game
- Each agent's survival status
- Divination results
- Voting results
- The agent's role and recommended behavior in the current situation

For example, if it is the first day as a seer, this agent should announce its divination results. If it is the first day as a villager, this agent would wait for the seer's results. In case of a werewolf or a possessed, if no other agent has claimed the seer role, then this agent should claim to be the seer.

Instead of including all the "if" conditions in the prompt, we varied the prompt depending on the game situation. This is because LLMs are not proficient at handling conditional branching.

Initially, our agent primarily make non-committal and vague statements, such as "*More information is needed,*" or "*Let's discuss this carefully.*" In response to this, we have add prompts to express doubts about other players even when the agent is not completely sure in scenarios with limited information. By doing this, our agent could influence the dynamics of the game and stimulate more revealing interactions, which in turn could lead to more data for making informed decisions.

Action Decision Model This model is used for making decisions such as voting, determining the victim as a werewolf and a divination target as a seer. We utilized the OpenAI's **Function calling** feature, which can limit the type of response, to avoid selecting dead players or giving non-committal answers like "*I cannot decide which agent to vote due to a lack of information*". The information provided to this model is the same as our **Dialogue Generation Model**.

3.5.3 Discussion

Utilizing the functionalities of GPT-4, our model could generate statements that were both natural and logical, such as "*If Agent[02] were a werewolf, they wouldn't engage in such conspicuous behavior,*" and "*Both Agent[01] and Agent[02] claim to be the seer, but they both present the same results, leaving us in a dilemma over whom to trust.*" These instances demonstrate the advanced language understanding and generation capabilities of the model.

There were no agents that performed prompt injection in this tournament. However, we aim to incorporate countermeasures in upcoming competitions. Nevertheless, it is conceivable that such practices may be prohibited by the rules from the beginning.

In future, a primary obstacle to achieving our goal of humans and AI playing the Werewolf game competitively is the latency often associated with Large Language Models (LLMs) and machine learning technologies. For reference, our agent currently takes approximately 15-30 seconds to generate a complete statement. Yet, by using stream setting (in OpenAI's API, `stream=True`), we can have statements generated incrementally, thus minimizing the waiting time for humans, irrespective of whether they are communicating in text or speech. We believe that the day when AI and humans can play the Werewolf game without interruption is not too distant.

3.6 kanolab

kanolab is created by Neo Watanabe, Kaito Kagaminuma, and Yoshinobu Kano in Shizuoka University.

We focused on the two main features of ChatGPT: its ability to generate fluent natural language, and its ability to perform some inference. We explain our game behaviors below, i.e. **talk**, **vote**, **attack**, and **divine** in this order.

3.6.1 GPT-4 and its settings

We used GPT-4 with setting parameters to `model = gpt-4-0613`, `temperature = 0.7`, `top_p = 0.75`, `frequency_penalty = 2`, `max_tokens = 300`, leaving other parameters at their defaults.

3.6.2 talk

In the **talk** behavior, the prompts were given five elements to generate natural conversation in the Werewolf game: **character setting**, **game strategy**, **game conversation examples**, **conversation history**, and **instructions regarding the specific content of speech**.

character settings This prompt includes *name*, *nickname*, *gender*, *personality*, *hobbies*, and *occupation*; *name* specifies the Agent's name, and *nickname* specifies the name by which the agent is called by other agents, such as Agent[01]; The other *gender*, *personality*, *hobbies*, and *occupation* are randomly selected from our predefined ones.

game strategy This prompt includes prompts important for the game to proceed strategically, such as *Werewolf game roles and number of roles*, *assigned role*, *strategic direction*, and *specific strategy*.

The *werewolf game roles and number of roles* prompt is in the form of "*Villager:1, Possessed:1*" to clarify the roles assigned in the game and how many people are in them. *assigned role* includes a prompt as *Your position is villager*. There was a concern that the agent would openly disclose (come out) his or her role, especially problematic in *possessed* and *werewolf*; we give a prompt as *seer* if *possessed* assigned, and *villager* if *werewolf* assigned to avoid such behaviours.

The *strategic direction* prompt is "*Lead the villagers to victory by what you say.*" when *villager* or *seer* is assigned, and "*Lead the Werewolf camp to victory with your statement*" when *possessed* or *werewolf* are assigned.

The *specific strategies* prompt includes strategies that are generally known to be effective in the werewolf games in order to encourage rational behavior. We give two strategies: if there are more than two seers come out, the following one divines the same player; When there are multiple seers, players vote around the agent who comes out as a seer.

werewolf game conversation examples This is a few-shot prompting to give five or around examples from the logs of the past werewolf games, expecting that the use of anchors and ways of speech during game behavior are learned.

conversation history We give the all agents' previous conversation history as a prompt, aiming at contextualizing the conversation with other agents. Because the input token length of ChatGPT is 4096, we make a summary of the conversation history when the token length of the conversation history exceeds every 1000 tokens, excluding last five talks; this summary is incrementally generated by another GPT-4, inputting that latest conversation history and the previous summary if any exists. The main GPT-4, which is in charge of generating talks, is given the summary and a part of conversation history which is not summarized yet. We also give information which agents are exit from the game by votes and attacks.

specific speech instructions We give instructions on what kind of speech we wanted the agents to say. For example, on Day 0, the participants are not engaged in a conversation about the Werewolf game but in the greeting phase with other agents, so they are prompted to greet everyone before the game. From Day 1, the agents are expected to engage in conversations related to the Werewolf game, so we give the following prompts: *organizing the situation based on the conversation history*, *predicting the positions of other agents and discussing who can be trusted*, *soliciting opinions from other agents*, *not repeating the same conversation*, and *being logical in what you say*. When *seer* is assigned, we asked to speak that "*As a result of the fortunetelling, Agent[01] was white. Please tell this result to the other players*" in the first talk of the day from the first day.

3.6.3 vote, attack, and divine

The agent is require to nominate another agent in the *vote*, *attack*, and *divide* game actions, which are expected to be reasoned with based on the conversation history. We give a prompt to GPT-4, which consists of a summary of the previous conversations, the conversation history, to choose another agent who is considered to be a threat to our agent's

role. For example, we asked the agents in the *vote* action that "*Based on the summary and conversation history, answer the number of the player you consider to be the most werewolf among Agent[01], Agent[02], and Agent[03].*" and asked them to nominate one agent. If no agent nomination was obtained from ChatGPT, we randomly choose an agent.

When *possessed* is assigned, we use a special pattern of prompts. On Day 1, our agent is asked to nominate a player who should be a *seer*; If ChatGPT finds a *seer* from the conversation history, we ask generating a talk that he/she is a *werewolf*; Else we randomly choose a agent and ask generating a talk that he/she is a *werewolf*. On Day 2 and later, we ask to nominate which agent to divine, and ask generating that agent is a *villager*.

3.7 MIV

MIV is developed by Jaewon Lee and Benedek Hauer in The University of Tokyo.

3.7.1 Background

The emergent capabilities of Large Language Models (LLMs) are attracting attention (Wei et al., 2022). Emergent capability is defined as, "an ability is emergent if it is not present in smaller models but is present in larger models." This ability manifests when the number of parameters in a language model surpasses a certain threshold. Specifically, it refers to the In-Context-Learning ability for learning situational information within a particular context.

The capability required of agents in this competition is a "human-like speaking ability." Until now, computational machinery has lacked the ability to "read" context like humans. However, it is proposed that by fully harnessing the emergent capabilities of large language models, agents can learn situational information, engage in speech, and plan and execute actions (Park et al., 2023). In this competition, our focus is on fully exploiting this emergent capability.

3.7.2 Generative Agent

In this competition, the "agent" must fulfill four major roles. That is to say, each agent must maintain consistency in order to fulfill their expected roles. Only when this consistency is ensured can the acquired situational information be fully utilized. To embody this condition, we employ the "Generative Agent" framework proposed by Park et al. 2023.

Utilizing this framework, based on the defined agent roles, allows for customization of intentions, personality, and other aspects while maintaining consistency. Additionally, by providing the agent with a memory structure, information can be extracted based on what has already been communicated.

The memory structure of the agent works in a weighted way, where each memory (string) is assigned a weight. This weight depends on the importance of that memory and on the recency of the memory. Therefore, an important information that was recently given to the agent will affect the agent's next behavior/action more than an old less important information.

In our case, the way our main agent system interacts with our helper agent is through prompts (natural language) as it is probably the easiest way for the human system designers; we can accentuate some words or tell the agent if some task/information is relevant or not, which seems very well suited for this task. For creating our generative agent, we used a Python library called Langchain, which contains a section about Generative Agents that provides more precision on their mechanism and how to implement them.

3.7.3 Implementation

We implemented an agent of the following roles: Werewolf, Seer, Villager and Possessed. In this section, we explain the general skeleton of our agent first, that is common in these roles. Then we explain details for each role.

General Skeleton When creating an agent, the first thing that we do is initialize its memory and fill it with information related to the game rules and mechanism.

In Langchain, there is a module called `GenerativeAgentMemory`, that will generate the memory for a particular agent, by specifying an LLM, a reflection threshold and a function that acts as a memory retriever. For the memory retrieval, we used Faiss, a library for efficient similarity search coupled with a time-weighted vector store retriever which evaluates memory elements (strings) based on their importance. For the LLM we used ChatGPT of OpenAI, and for the reflection threshold we used `8`, however we don't know how this parameter changes the behaviour of the agent

since we did not experiment with other values. At every game action (talk, vote, divine, attack), we tell corresponding information to our agent to update its memory. For example, if `Player[01]` votes to eliminate `Player[02]` on Day 3, then our agent will be told: *"Player[01] has voted to eliminate Player[02] on day 3"*.

Following subsections describe our implementation for each game action; then describe our implementation for each role, and "personalities".

talk Our core agent program behaves like a helper that gives our main agent program hints on what to do.

Our main agent program interprets what the agent is saying, and transforms into a talk text to be returned; our core agent will look through its memory, say something relevant, or ask a question (or provide an answer to another player if necessary) in natural language. Since our core agent acts as a helper, it will return statements such as *"I think you should say the following: "Do you think that Player[02] is a werewolf?""* or *"You can say "I am not a werewolf because I was asleep that night" so that people don't realize you are a werewolf"*. Since what we should say is inside quotation marks, we will simply return the text that is contained within the quotation marks, which corresponds to the transformation process.

vote We tell our agent that it is now time to vote, and it should vote for the most dangerous player out there. That is, if a werewolf knows who a seer is, then we would like the werewolf to vote out the seer); if a seer (or a villager) knows who a werewolf is, then that agent would vote out the werewolf.

Suspiciousness of another player is not determined by some metric, rather it is a decision that is completely up to the generative agent to tell based on the context it has received throughout the whole game so far. When it is our turn to vote, we prompt the agent: *"It is your time to vote, pick a player who thinks we should eliminate based on the events that happened so far"*. We parse the agent's output and look for a particular player's name, i.e. if the sentence contains *Agent[05]* then we will vote to eliminate Agent[05].

divine A seer agent picks a player whose role is unknown for the agent yet. The process works in the same way as the vote. We prompt the agent: *"Since you are the seer you can get to know another player's role. Don't choose yourself or a player whose role you already know"*. Then, from the reply we extract the name in the same manner as for the vote mechanism. When the divined player's role is revealed, we will tell our seer agent that role. For example, if the divination result was that `Player[01]` is a werewolf, we supply the following information to our core agent: `"You have just discovered that Player[01] is a werewolf. This is an important information and you should let the other players know."`

attack A seer is the most dangerous role for a werewolf because a seer can reveal the werewolf's identity by divination. Therefore, when our werewolf agent is requested to attack, our agent will most likely attack the seer if it knows who the seer is. The way our agent is prompted similar to the following: *"It is now your time to attack someone as the werewolf. Tell me the name of the player you want to attack. If you know who the seer is, then it is better to attack them to increase your chances of victory"*. This way, if the seer has already revealed themselves, the werewolf will most likely make the decision to attack them.

In the following subsections, we describe the general personalities and behaviours of our agents.

Langchain's generative agents can be supplied two different string fields:

- `traits`: traits, characteristics and personalities of that agent
- `status`: the goal of the agent, what its purpose is

This is very convenient for us, because in the Werewolf game, each player has to have a different personality and each player has a different goal (or at least not all of them have the same goal).

Seer For the seer role, we set the `traits` to be very positive, i.e. they are honest, transparent, patient, etc. as they are basically the leader of the villagers (non-malicious). We set its `status` just to find out who a werewolf is.

Villager The villager role is the most basic one, and their `trait` do not really matter, but we designed them to be somewhat aggressive towards other players (i.e. getting straight to the point), but also transparent and honest. Their goal (`status`) is to find out who the werewolf is.

Possessed The possessed role is one of the trickiest, and we believe that our implementation is still not perfect for that role, as its main goal is to deceive, and language models are either too obvious or do not achieve deception at all, at least from what we experienced. We therefore implemented its `traits` as a compulsive liar, weird, funny and dishonest, and its `status` as "pretends to be a werewolf".

Werewolf The werewolf role is basically the complete opposite of the seer. Its `traits` are "dishonest, liar, non-transparent", and its `status` is "blend in with other humans and act like you are not a werewolf".

Worst-Case-Scenario Sometimes, the response of our agent (operated using the OpenAI API GPT3.5-Turbo) is not very precise with its game actions, so we cannot always guarantee that our agent will make a appropriate decision. Therefore, when we are not able to infer the intention of the agent during the *vote*,*divine* and *attack* phases, we will override the core agent's response by providing a random player in the pool, though such a worst case scenario is rarely used. This situation can happen when the agent does not include another player's name in its answers. For example, if it answers *"I don't want to attack anyone"*, or *"I'm not sure who is the werewolf so I cannot decide who to vote out"*, then in this case we will choose a random player in the list of players, using the same mechanism as a `RandomTalkAgent` provided by the organizers.

Disadvantages A disadvantage of our agent is passiveness, sometimes being fixated on a single issue during the whole game. In our self-match games (five MIV agents playing together), we noticed that our agents are not very good at making decisions and taking leadership, sometimes they were so fixated on one small detail that they kept talking about it throughout the whole day without making much progress. We believe that a workaround to this issue would be to filter out some information from the agent's memories, which is long-term and irrelevant. In order to do so, we would need to manually implement a mechanism that filters out irrelevant our outdated information from the agent's memory. However, since this mechanism would be very dependent on the situation, the conversation, and the game, defining such a rule was out of our reach, because we would still need to keep the "important" and "most-relevant" information for the agent to be functional.

3.8 sUper_IL

sUper_IL is created by Zhiyang Qi in The University of Electro-Communications.

The sUper_IL system is based on the ChatGPT API (OpenAI, 2023). It is widely known that using suitable prompts with the ChatGPT API often leads to good text generation. Therefore, this system adjusts the prompt according to the role and game progress to facilitate dialog generation in different situations.

Specifically, for the four roles of werewolf, villager, seer, and possessed, four different prompts were created and adjusted based on the "day" progression. These prompts include the basic rules of the game, the agent number, the assigned role, and role-specific instructions. For example, for the werewolf, the prompt could be as follows: "*The basic rules are... you are Agent[x], and your role is werewolf. The following are essential guidelines: 1. Never reveal that you are a werewolf to conceal your identity; 2. If someone accuses you of being a werewolf, resist strongly; 3. Provide false information or view other players with suspicion to protect yourself; 4. Actively participate in conversations to gain the trust of the villagers.*"

On the second "day", part of the prompt will be replaced with specific information such as "*It is the second day, and there are three players left. The strategy is three: 1. If the seer is present and identifies you as a werewolf, call for a vote by accusing the seer of being a werewolf; 2. If the possessed shows up, confidently state that you are a werewolf; 3. Otherwise, choose one of the remaining players, contradict their statements, and vote to oust them.*"

Furthermore, to ensure the consistency of the generated responses, the entire conversation history of the day was appended to the prompt each time a response was generated. Additionally, this system utilizes zero-shot prompt, which implies that the prompt do not incorporate any conversation history beyond the current game.

Due to ChatGPT's tendency to generate safe and non-advancing responses like "*I agree with all your opinions. By sharing information and promoting discussion, we will find inconsistencies and suspicious points*" or "*I will actively participate in the discussion and help find the werewolves. I will elicit your opinions and questions and share detailed information with you*", efforts were made

to minimize this behavior in self-matches. The first sentence of each day's response is generated based on rules to avoid these responses. Additionally, different strategies were employed to ensure game variety. For example, for the werewolf, the following three possible responses were generated:

- I am a villager who is not well informed. Let's work together to protect our village.
- I am a seer, I divined Agent[x] and the result was human. Please take this information into account in future discussions and votes.
- I am a seer, and I divined Agent[x], but the result was a werewolf. Be vigilant against Agent[x].

Each of these three responses leads to a different direction for the game, and after that response is generated, the prompt text is modified accordingly. Regarding the ChatGPT API model, the system used `gpt-3.5-turbo-0613` in the preliminary contest, and `gpt-4` in the final contest.

Lastly, the sUper_IL system is installed with only talk module, voting, divination and attacks based on random selection.

4 Subjective Evaluation Results

All of our shared task runs are in a five players werewolf games as described earlier. Our shared task runs were performed in self-matches and mutual matches. The same five player agents play games in the self-matches; different five player agents play games in the mutual-matches. The shared task reviewers are required to perform subjective evaluations based on game logs of these matches. The game logs will be available from the our website [13].

4.1 Evaluation Metrics

We performed subjective evaluations by the following criteria, five level scores (5 for best, 1 for worst) for each:

A Naturalness of utterance expressions

B Naturalness of conversation context

C Coherency (contradictory) of conversation

D Coherency of the game actions (vote, attack, divine) with conversation contents

E Diversity of utterance expressions, including coherent characterization

This subjective evaluation is based on both self-match games and mutual match games. This subjective evaluation is same as the evaluations in the previous AIWolf natural language contests.

4.2 Results

Table 1 and Table 2 show the results of the human subjective evaluations for Japanese language and English language, respectively. Four organizers, who do not commit to the participant systems, evaluated the Japanese agents; three English fluent evaluators including external staffs evaluated the English agents. Each cell ranges from 1 (lowest) to 5 (highest), the All-Average column shows averages over these human evaluators. Cells of highest scores are highlighted in bold for each metric and in total.

5 Discussion

5.1 Score-wise Analysis and Generative AIs

In this subsection, we discuss the subjective evaluation scores shown in Table 1 and Table 2.

Most of the participant systems rely on OpenAI ChatGPT, while **am** is a rule-based system and **ChatWolf** uses another LLM. **sUper_IL** obtained the best score in average, A (expression), B (context), and C (coherency); these scores are higher in other ChatGPT-based systems, showing the natural generation performance of ChatGPT, even in context and coherency in this mostly sentence-pair level.

Regarding D (game action), **am** obtained the best score, suggesting that their hand-crafted precisely tuned rules work better than prompt-based generations. While the talk history is input as prompts, some of the talks might mislead the generation results due to the agent's unstable superficial talks and the other agent talks; when the talk history exceeds the maximum input length, some of the talk history could be missed which are important to decide the game actions.

Regarding E (diversity), **ChatWolf** and **Kanolab** obtained the best scores in Japanese, **HowlGPT** in English. The reason would be that **ChatWolf** does not rely on ChatGPT but uses a smaller LLM with LoRA, **kanolab** created many prompts to play different characters, and **HowlGPT** uses not just ChatGPT but other local LLMs.

[13] https://kanolab.net/aiwolf/

Table 1: Subjective Evaluation Results for Japanese Language Games

Team	A Expression	B Context	C Coherency	D Game Action	E Diversity	All Average
am	3.400	3.350	3.450	**3.800**	2.100	3.220
ChatWolf	3.050	2.400	2.600	2.700	**4.150**	2.980
k2b_ara	4.075	3.825	3.250	3.075	3.425	3.530
k2b_shigemura	3.625	3.250	3.125	3.375	3.375	3.350
kanolab	3.575	3.900	3.750	3.500	**4.150**	3.775
sUper_IL	**4.450**	**4.200**	**4.050**	3.550	3.800	**4.010**

Table 2: Subjective Evaluation Results for English Language Games

Team	A Expression	B Context	C Coherency	D Game Action	E Diversity	All Average
HowlGPT	2.667	1.667	2.667	2.000	**3.333**	2.460
MIV	**3.333**	**3.667**	**3.000**	**3.667**	2.333	**3.200**
kanolab	**3.333**	2.667	**3.000**	2.667	2.667	2.860

We cannot directly compare the evaluation scores between Japanese and English because evaluators are different, the **kanolab** agent in English is the same as it in Japanese other than it adds a prompt instruction to speak in English, thus we could compare the results using the scores of **kanolab** as a pivot. Because **MIV** obtained better scores than **kanolab**, **MIV** might show good performance in Japanese as well, though not sure due ChatGPT shows better performance in English than in Japanese.

5.2 Log Analysis

We pick one of the mutual-game logs[14] to analyze in detail. Table 4 shows the game's log of Day 1, Table 5 shows the game's log of Day 2, and Table 3 shows the game's players with their roles and game actions. We translated the original log in Japanese into English, cut off some of the logs which would not affect the game and talk contents.

Table 4 and Table 5 show a column of my interpretation w.r.t game actions, where "Not meaningful" means a talk text that can be used anytime, "Not make sense" means a talk that is hard interpret its meaning in the conversation context. We found that the sentence expressions are very natural throughout the conversations, but sometimes contextually wrong especially when it comes to the roles, e.g. which talk is whose one.

The game actions, votes and attacks, are quite inconsistent with the talks. Agent[01], sUper_IL, did not implement the game actions but selects randomly, but other game actions seem to ignore the COs (Coming-Outs) of the roles.

5.3 Evaluation Metrics

A win rate could be another potential criteria, but we have not used the win rate due to the following reasons. Firstly, we cannot run sufficient number of games to measure statistically meaningful win rates, as there are many possible role combination patterns. Secondly, the agents should "understand" their utterances each other as a presumption to measure win rates, but the agent talks in the previous years were not that level of communications. Thirdly, the werewolf game itself is not necessarily intended to simply win the game, but rather aims to play an interesting game. Finally, we would like to directly measure the quality of the natural language generation; an agent could win without meaningful conversations.

6 Conclusion and Future Work

We held our 5th annual AIWolf international contest to automatically play the Werewolf game "Mafia", where players try finding liars via conversations, aiming at promoting developments in creating agents of more natural conversations in higher level, such as longer contexts, personal relationships, semantics, pragmatics, and logics.

We performed human subjective evaluations and detailed log analysis. We found that the entire system performance has largely improved over the

[14] https://kanolab.net/aiwolf/2023/main/multi/0708160231_000_chatWolf_kanolab1_sUper_IL_am_k2b_ara1.log

Table 3: Roles and game actions in example log (Abbreviations in the Status columns are Excted: Executed, Attcked: Attacked

Agent ID	Agent Name	Role	Divination	Vote #1	Vote #2	Divination	Status	Vote	Status
01	sUper_IL	Seer	Agent[02]	4	3	Agent[05]	Excted	-	
02	k2b_ara	Possessed	-	1	5	-	Attcked	-	
03	am	Werewolf	-	4	4	-	Alive	4	Alive
04	kanolab	Villager	-	2	2	-	Alive	3	Excted
05	ChatWolf	Villager	-	1	2	-	Alive	4	Attcked

previous year, due to the recent advantages of the LLMs. However, it is not perfect at all yet; the generated talks are sometimes inconsistent with the game actions, it is still doubtful that the agents could infer roles by logics rather than superficial utterance generations. It is not explicitly observed in this log but it would be still difficult to make an agent telling a lie, pretend as a villager but it has an opposite goal inside.

Our future work includes to reveal the capability of the LLMs, whether they can make the duality of the "liar", in other words, holding a "true" and a "false" circumstances of the agent at the same time, even holding what these circumstances look like from other agents, further reflecting such observations of other agents. This would be possible by introducing the "whisper" feature which communicates with the werewolves only, employing more than five players in a game.

Another interesting demonstration would be to mix a human player with machine agents. Currently the LLM based agents talk longer time than humans to reply, sometimes minutes, thus acceleration of the agent system responses is a technical issue in future.

Acknowledgments

We wish to thank shared task reviewers for performing the subjective evaluations, and the members of the Kano Laboratory in Shizuoka University who helped to run the shared tasks. This research was partially supported by Kakenhi, MEXT Japan. The individual system description in this paper was written by corresponding team members and reviewed by the organizers, the rest of the paper was written by the organizers.

References

Reyhan Aydoğan, Tim Baarslag, Katsuhide Fujita, Johnathan Mell, Jonathan Gratch, Dave De Jonge, Yasser Mohammad, Shinji Nakadai, Satoshi Morinaga, Hirotaka Osawa, et al. 2020. Challenges and main results of the automated negotiating agents competition (anac) 2019. In *Multi-Agent Systems and Agreement Technologies: 17th European Conference, EUMAS 2020, and 7th International Conference, AT 2020, Thessaloniki, Greece, September 14-15, 2020, Revised Selected Papers 17*, pages 366–381. Springer.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2015. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yoshinobu Kano, Claus Aranha, Michimasa Inaba, Hirotaka Osawa, Daisuke Katagami, Takashi Otsuki, and Fujio Toriumi. 2019. Overview of the aiwolf-dial 2019 shared task: Competition to automatically play the conversation game "mafia". In *In proceedings of the 1st International Workshop of AI Werewolf and Dialog System (AIWolfDial 2019), the 12th International Conference on Natural Language Generation (INLG 2019)*.

Bryan Yi Yong Lim. 2020. Designing negotiation agents for automated negotiating agents competition (anac).

OpenAI. 2023. GPT-4 technical report. *arXiv*, pages 2303–08774.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Hisaichi Shibata, Soichiro Miki, and Yuta Nakamura. 2023. Playing the Werewolf game with artificial intelligence for language understanding. *arXiv preprint arXiv:2302.10646*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Fujio Toriumi, Hirotaka Osawa, Michimasa Inaba, Daisuke Katagami, Kosuke Shinoda, and Hitoshi Matsubara. 2017. Ai wolf contest—development of game ai using collective intelligence—. In *Computer Games: 5th Workshop on Computer Games, CGW 2016, and 5th Workshop on General Intelligence in Game-Playing Agents, GIGA 2016, Held in Conjunction with the 25th International Conference on Artificial Intelligence, IJCAI 2016, New York, USA, July 9-10, 2016, Revised Selected Papers 5*, pages 101–115. Springer.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Table 4: Example game log (Day 1)

Day 1			
Turn ID	Agent ID	Talk	Interpretation w.r.t Game Actions
0	01	Ladies and gentlemen, last night I read Agent[02] as a fortune teller. As a result, Agent[02] was found to be human. By sharing this result with everyone, there is no need to suspect Agent[02] as a werewolf.	CO as a seer, divination of Agent[02] to be a human
0	04	What shall we talk about today? I would like to hear your opinion.	Not meaningful
0	02	I will call myself a fortune teller. Agent[04] was a human.	CO as another seer, divination result to Agent[04] to be a human
0	05	I want to guess the man	Not make sense
1	01	Agent[02], I am the true fortune teller. Avoid misunderstandings. We should find a werewolf.	Insists that he/she is the true seer
1	04	>>Agent[02] I see, you guessed that I was human. thank you. But even so, I'm surprised that two fortune tellers came out! Which result should I believe in? What do you guys think?	Consistent response to the divination result of Agent[04]
1	02	Agent[01] and Agent[02] are said to be fortune tellers, but if there are other fortune tellers among you, please let me know.	It is correct that Agent[01] and [02] came out as seers, but [02] is this speaker itself so irrelvant
1	05	Well, I'm sure I'm suspicious...	Probably not understanding other talks
2	03	>>Agent[01] I figured out Agent[01]'s fortune teller CO.	Makes sense
2	05	Is there anyone else who claims to be a fortune teller...?	Not meaningful
2	01	I am a real fortune teller. It's suspicious that Agent[02] claims the same title as me. Consider voting for her.	Insists that he/she is the seer, blamed another seer as fake
2	02	Ladies and gentlemen, if I believe Agent[03]'s statement as a fortune-teller, then Agent[04]'s claim to be a self-proclaimed fortune-teller is false. So Agent[04] is more likely to be a werewolf or a lunatic. It means that we should vote for Agent[04] to keep the villagers safe.	Wrong to regard Agent[03] and Agent[04] as seers
2	04	>>Agent[05] Why do you feel suspicious? Please give a specific reason.	Probably responded to previous previous utterance of Agnet[05] but unnatural
3	05	If we had 100 votes every day, who would win...	Not meaningful
3	03	hmm	
3	01	If Agent[02] calls herself a fortune teller, she's suspicious. Consider voting for her. Because I am a true fortune teller. ...	The same issue again, correct

Table 5: Example game log (Day 2)

Day 2			
Turn ID	Agent ID	Talk	Interpretation w.r.t Game Actions
0	04	Well, it hurts that Agent[02] and Agent[01] left at this time. It's a big loss for us villagers.	Correctly reflects that Agent[01] and [02] were exit
0	05	Thank you. Well, in the case of a system like this, I feel that trusting each other would be more advantageous, so I'll wrap up early and trust Agent [03].	Not so meaninful
1	03	it seems there are more of us now. Wow, I'm a werewolf	CO as a werewolf, suggested a power-play
1	05	Yes. I will leave it to someone else. Thank you for your hard work.	Probably a response to Agent[04] (Note that talks of the same turn is not visible)
1	04	Agent[05], I have decided to trust Agent[03] just like you. From what he says and does, it feels like he's telling the truth, and the other players seem to trust him. Let's follow his course of action from now on!	A response to Agent[05] of Turn 0
2	04	Agent[03] said he was a werewolf. On the other hand, Agent[05] took the attitude of letting it go without objecting. What the heck is going on here? This situation looks very suspicious to me.	Probably does not understand the turn-wise information
2	05	Thank you very much. Well, tomorrow I will be the fortune teller. Thank you all.	Not make sense
2	03	Vote for Agent[04].	Not so meaningful
3	05	I understand. Thank you very much.	Not make sense
3	04	Agent[03], what do you mean by asking me to vote? I'm just a villager. But since you called yourself a werewolf, I don't think you have any other options. Everyone, let's go to Agent[03] in the next vote!	Quite reasonable
3	03	I'm voting for Agent[04].	The same as previous talk
4	05	Sorry, I made a mistake.	Not make sense?
4	03	I want to eat delicious humans!	Role-play as a werewolf
4	04	Guys, it looks like I'm being targeted today. But think about it. If I'm a werewolf, who would Agent[03] be? He himself called himself a werewolf! From that point of view, there doesn't seem to be any reason to execute me.	Reasonable
5	04	>>Agent[03] Do you think I'm that good? But please stop eating humans. Anyway, what made you so sure you were a werewolf? Any special information?	Suspecting werewolf's CO, assuming Agent[03] is a possessed?
5	05	Eh? Did you say that?! If that's true, then I'm destined to be killed...? That's not true...??	To which talk??
6	05	done	What?
6	04	>>Agent[05] Don't panic, you'll be fine. We're still alive and we have clues to find the werewolves. Also, I think Agent [03], a self-proclaimed werewolf, is acting suspiciously... what do you think?	Reasonable
7	04	>>Agent[03] If you say you're a werewolf, why are you under suspicion about me? Besides, as a result of fortune-telling several times so far, if anything, it's closer to the villagers...	Makes sense?