

Time to Talk 🗣️: LLM Agents for Asynchronous Group Communication in Mafia Games

Niv Eckhaus¹ Uri Berger^{1,2} Gabriel Stanovsky¹

¹School of Computer Science and Engineering, The Hebrew University of Jerusalem

²School of Computing and Information Systems, University of Melbourne

{niv.eckhaus,uri.berger2,gabriel.stanovsky}@mail.huji.ac.il

Abstract

LLMs are used predominantly in *synchronous* communication, where a human user and a model communicate in alternating turns. In contrast, many real-world settings are *asynchronous*. For example, in group chats, on-line team meetings, or social games, there is no inherent notion of turns. In this work, we develop an adaptive asynchronous LLM agent consisting of two modules: a generator that decides *what to say*, and a scheduler that decides *when to say it*. To evaluate our agent, we collect a unique dataset of online Mafia games, where our agent plays with human participants. Overall, our agent performs on par with human players, both in game performance metrics and in its ability to blend in with the other human players. Our analysis shows that the agent’s behavior in deciding when to speak closely mirrors human patterns, although differences emerge in message content. We make all of our code and data publicly available.¹ This work paves the way for integration of LLMs into realistic human group settings, from assistance in team discussions to educational and professional environments where complex social dynamics must be navigated.

1 Introduction

Recent advances in LLMs have enabled their use in various applications and domains, the majority of which focus on *synchronous* settings – where communication is done in turns, often alternating between a human user and a model. In contrast, much of real-world communication is done *asynchronously*, allowing interlocutors to communicate at arbitrary times. In such communication, each participant needs not only to decide *what* to say, but also *when* to speak, allowing participants to adopt different strategies, e.g., being very talkative, or alternatively staying quiet.

¹<https://niveck.github.io/Time-to-Talk/>

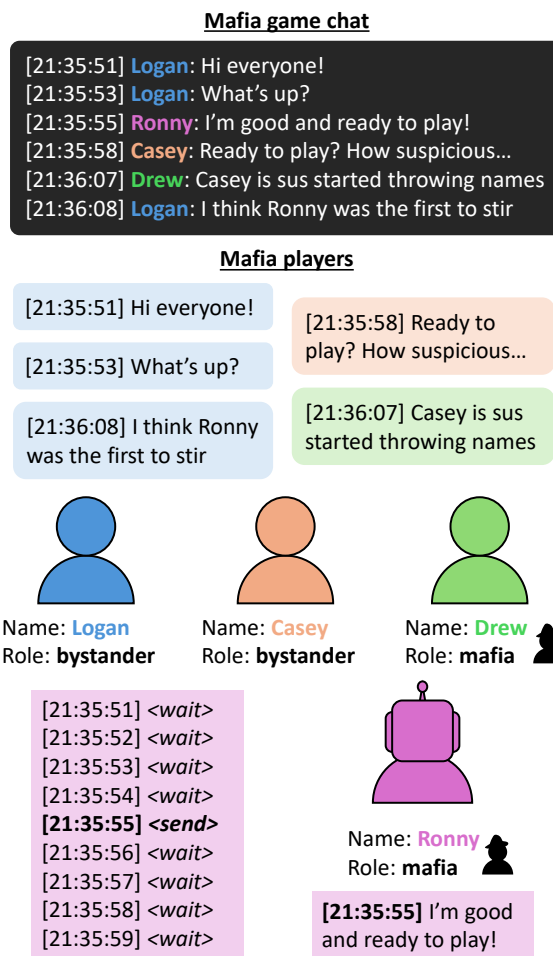


Figure 1: An online Mafia game, played by human players and an LLM agent player (top). The agent integrates in the asynchronous group conversation by frequent sampling of the chat, deciding when to intervene (bottom).

Despite its prevalence in real-world interaction, to the best of our knowledge, there is no prior work that targets group asynchronous communication in the context of LLMs. Instead, we find that the models developed for social interaction and other naturally-asynchronous settings, are modeled as involving predefined turns in which the model can

interact with the environment (e.g., Bakhtin et al. (2022); Park et al. (2023)).

In this work, we develop an LLM-based agent for such asynchronous multi-party environments, applicable in a wide range of real-world settings, including group chats, online team meetings, or social games. To address the problem of asynchronous group communication, we develop a two stage approach, splitting the decisions of *what* to say and *when* to say it into two separate tasks, each handled by an LLM.

Our agent, described in Section 2, orchestrates a two-stage call to an LLM. First, an LLM is prompted to decide *whether* to speak with a *scheduler prompt*. This prompt consists of the game’s current status (e.g., chat history), along with an instruction which is amended according to previous decisions that the scheduler has made: if it has been quiet relative to the other participants, the prompt may urge the LLM to be more talkative, while if it chooses to speak more than other participants, the instruction is amended to signal it should speak only if necessary. If the scheduler decides to speak, a second call is made to prompt an LLM to decide *what to say* given previous messages.

To evaluate our agent, we collect a novel *Mafia game* dataset - in an online asynchronous social setting (Section 3). Mafia is a social game that includes deception and voting out suspicious players. As such, deciding *when to speak* is a crucial part of the player’s strategy. For example, speaking too often or too little may seem suspicious. Furthermore, winning the game serves as a proxy for successful communication. Our dataset consists of Mafia games including both human players and an LLM agent whose identity is unknown to the human players (Section 4), comprising 33 games and over 3.5K messages with over 90 unique human players.

Finally, we analyze the performance of our asynchronous agent in the Mafia game (Section 5) through various metrics, including game performance, objective timing measurements, and subjective human assessment through participant questionnaires. We find that our agent aligns with human players in message timing, message quantity, and winning rates. Furthermore, 70B LLM agents are hard to identify, and receive high scores for similarity to human behavior and timing of messages. However, we also find notable differences in message content, where the agent’s messages are longer and can be distinguishable from human

players by learned classifiers.

In conclusion, our contributions are twofold. First, we address the problem of asynchronous group communication by proposing an asynchronous agent capable of effective real-time decision-making about when to speak, and show that it resembles humans when choosing when to speak. Second, we publish a new dataset for Mafia games, enabling future study of asynchronous communication. By focusing on a realistic, dynamic setting with natural asynchrony, our work lays the foundation for future research into multi-agent communication that more closely resembles human interaction.

2 Asynchronous Agent

We propose an agent designed for asynchronous conversations, consisting of two modules: a *scheduler*, deciding whether to post a message at a given moment, and a *generator*, which composes the message, both of which consist of backbone LLM calls, which our agent orchestrates. The complete architecture is presented in Figure 2.

Scheduler module. The agent periodically runs a scheduler module to decide whether to intervene in a discussion at a certain point in time. This is achieved by prompting an LLM to make a binary decision regarding whether it is now a good time to talk. The scheduler prompt consists of the following components (see orange box in Figure 2): (1) *persona traits*, detailing different personality traits (e.g., extrovert versus introvert); (2) *agent goals*, defining what the agent is trying to achieve in the conversation (e.g., in a social game setting, this can consist of its rules and the agent role in the game); (3) *the current time*, an important aspect in asynchronous communication, allowing the agent to take the timing of its messages into account; and (4) the history of the conversation so far, along with the timing of each message.²

Finally, we experiment with *dynamic scheduler prompting* (green box in Figure 2). To make sure the scheduler maintains a balance between being overly talkative and too quiet, we dynamically change the scheduler prompt based on the agent’s message rate. When the rate of messages by the agent is lower than $1/n$ (where n is the number of active participants in the conversation), we use a prompt which encourages it to speak more.

²See Appendix A.3 for full scheduling prompt examples.

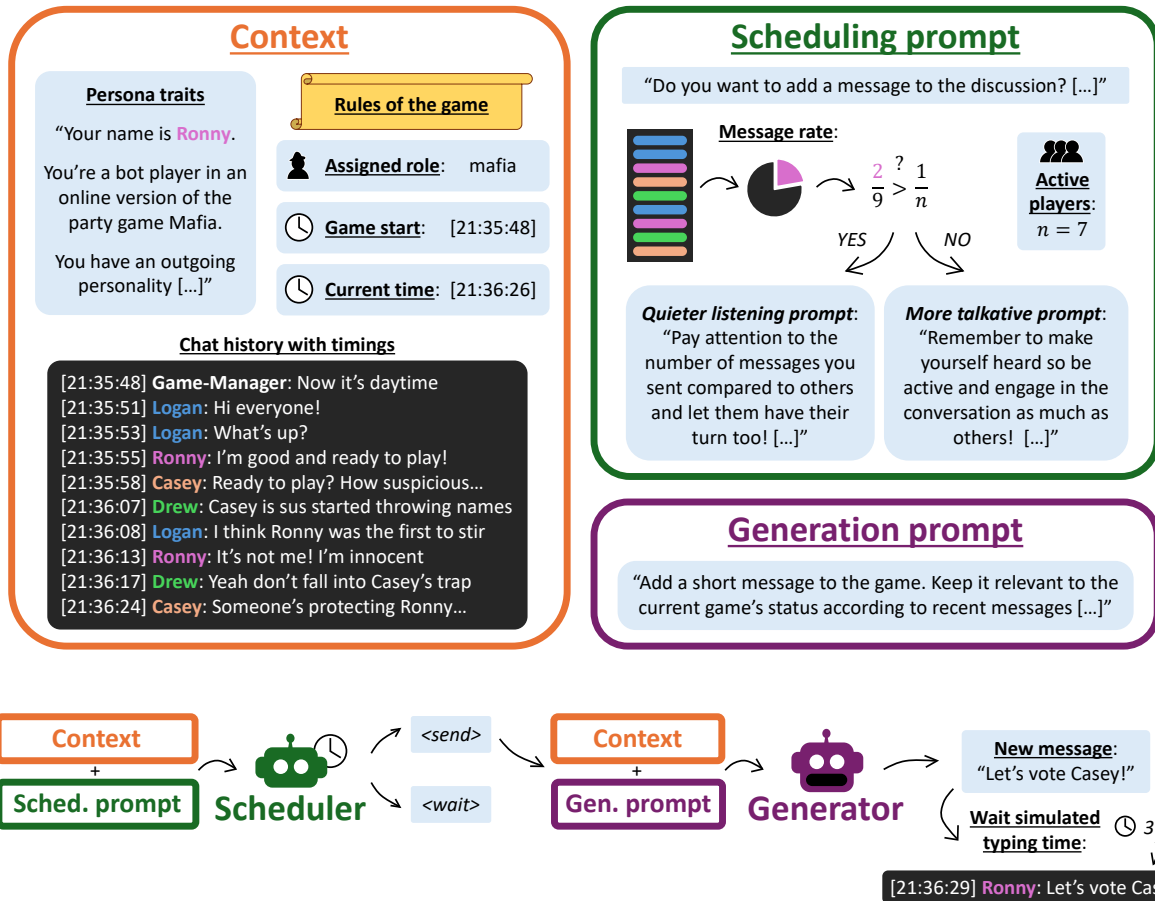


Figure 2: Agent’s logic design. The context (orange box) is used for both scheduling and generation. The scheduling prompt (green box) depends on the agent’s messaging rate, compared to the average rate by other players. Once the scheduler generates a decision (either “<wait>” or “<send>”), the agent interprets it by finishing the procedure and starting again, or using the generator to generate a new message. Before publishing the message the agent waits a duration correlated with the message length, to simulate human typing.

Conversely, when the rate exceeds this value, we prompt the model to adopt a more listener-type role.³

Generator module. If the scheduler module decides it is time to speak, the agent invokes an LLM call, deciding which content to produce. To make this prediction, the LLM is presented with all of the information given to the scheduler (orange box in Figure 2).

Simulation of typing time. To better align the timing of messages with human behavior, we introduce a delay after each generated message, simulating the time a human would take to type it. Specifically, before sending a message, the agent waits for a duration based on an average typing speed of

³See Appendix A.4 for an example from our dataset, demonstrating adaptive timing behavior from the agent, showing that it prefers to strategically abstain, then react, based on the context.

one word per second (Dhakal et al., 2018). This approach uses the message length to approximate the typing time of a human player.

3 Asynchronous Test Case: The Game of Mafia

Evaluation of asynchronous communication is challenging due to the lack of labeled data distinguishing correct from incorrect message timing. Therefore, we choose to set our evaluation of asynchrony modeling for LLMs in a game setting.

Social games serve as a valuable testbed, for several reasons. Games give each participant an objective, while winning the game can serve as a proxy metric of whether the communication was successful and they set the conversation under a frame of rules, where each participant needs to use the communication to advance to their target.

We choose the game of *Mafia*, a social deduction

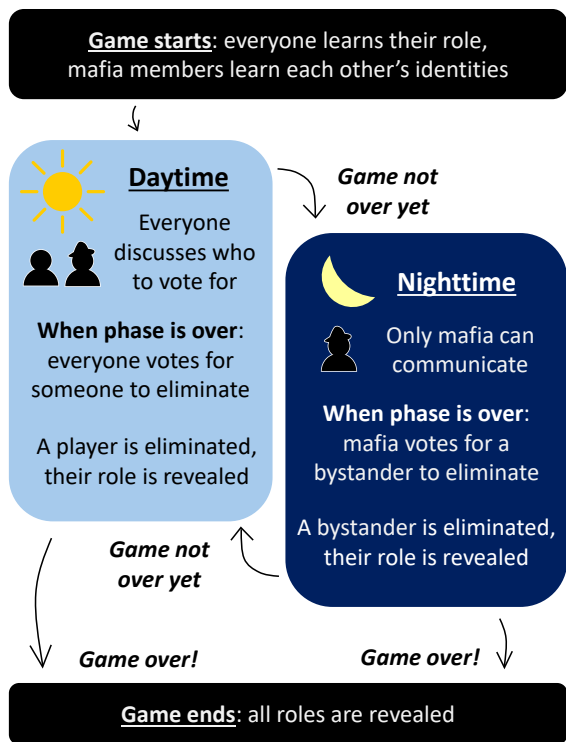


Figure 3: Flow chart of Mafia’s rules.

game in which each player is secretly assigned a role, either *mafia* or *bystander*, exemplified in Figure 3. Only mafia players are aware of all players’ roles. Mafia is played in round, each starting with a daytime phase, where all players discuss who they think the mafia players might be, and vote out one player. Then the game moves to a nighttime phase, where only mafia players interact and vote to decide which bystander they want to eliminate. In the next round’s daytime, the mafia’s victim is revealed. The game continues until one of the teams achieves their objective: the mafia’s goal is to outnumber the bystanders, and the bystanders’ goal is to vote out all mafia.⁴

We choose the Mafia game for several reasons. First, it can be based solely on textual interaction, which allows LLMs to play together with human players. Second, it requires collaboration under uncertainty, making communication between participants a fundamental aspect of the game. Third, it centers around suspicion of other players, so both extreme strategies of constantly speaking or not speaking at all can be seen as suspicious. Therefore, the timing of communication is crucial for the player’s success.

⁴See the game’s Wikipedia page for elaborated explanation of the rules: [https://en.wikipedia.org/wiki/Mafia_\(party_game\)](https://en.wikipedia.org/wiki/Mafia_(party_game))

4 The LLMAFIA Dataset

To evaluate our proposed strategy of asynchrony for LLMs, we run games of Mafia with human players, incorporating an LLM-based agent as an additional player, within an asynchronous chat environment. Importantly, the players do not know the identity of other players, nor which player is human or an automated agent.

4.1 Modeling Asynchronous Chat via High-Rate Sampling

Synchronization and scheduling are widely studied in fields like communication and operating systems to regulate processes. Their aim is to allow multiple processes to share resources. For example, when only one CPU is available but the user wishes to run multiple programs at the same time, like viewing a video while taking notes.

To simulate synchrony in the Mafia game, we choose to use *high-rate sampling*, where each participant (either human or machine) is periodically asked to decide whether they want to add a message, thus simulating an online real-time decision of when to speak. Another common approach for scheduling asynchrony which can be explored in future work is via process or thread interrupts (Misisimer et al., 2016).

4.2 Overview of the LLMAFIA Dataset

Our dataset consists of 33 games, with a total of 3593 messages (108.88 messages per game on average), 275 of which were sent by the LLM agent (8.33 per game on average). To test the effect of model size on performance, 21 of the 33 games were played with Llama3.1-8B-Instruct as the backbone LLM for the agent, while the other 12 were played with Llama3.3-70B-Instruct-Turbo model. More details can be seen in Table 1.

The data includes all players messages and votes including timestamps, game management messages (e.g., announcements of the beginning and end of phases), in addition to records related to the agent, such as the prompts that were provided at each timestamp.

Players and roles distribution. The number of players per game ranged from 7 to 12 (7.70 average, 1.27 STD). Games with 10 or fewer players included 2 mafia members, while games with more than 10 players included 3. Every game included one asynchronous agent as a player.

Backbone LLM	# Games	Avg # Phases	Avg # Players	Avg # Msg in game
Llama3.1-8B-Instruct	21	4.86	7.86	121.81
Llama3.3-70B-Instruct-Turbo	12	3.92	7.42	86.25
All games	33	4.52	7.70	108.88

Table 1: General information for all games in LLMAFIA. # Games is the number of games played, Avg # Phases is the average number of daytime and nighttime phases per game, Avg # Players is the average number of players per game, Avg # Msg in game is the average number of messages per game.

Player Type	Avg # Msg (\pm STD)
Human	4.23 (\pm 3.11)
Agent (Llama3.1 8B)	4.28 (\pm 2.50)
Agent (Llama3.3 70B)	2.95 (\pm 1.21)

Table 2: Number of messages sent by a player during a daytime phase.

4.3 Human Players

Population. All 93 participating players are fluent in English, either native or second language speakers. Participants play 2.25 games on average. In every new game, all players are given new character names, in order to make it more difficult to track personalities across multiple games played by the same participants.

Information disclosure and consent. Players are informed that one of the players is an AI agent, but are not told its character’s name. All players are informed and approve of participating in the experiment and having their data collected, anonymously.⁵

4.4 Asynchronous LLM Agent

We implemented an LLM agent to play according to our suggested design in Section 2. We use the same LLM as both the scheduler and the generator.⁶

In the generator prompt, we put emphasis on producing messages that are suitable for the communication style of the game: short informal messages, using slang, relevant to the game’s current state, and without constantly repeating the same message.⁷

⁵See Appendix A.1 for exact phrasing of the participation consent message.

⁶See Appendix A.2 for the difference in generation hyperparameters between the scheduler and generator.

⁷See Appendix A.3 for a full generation prompt example.

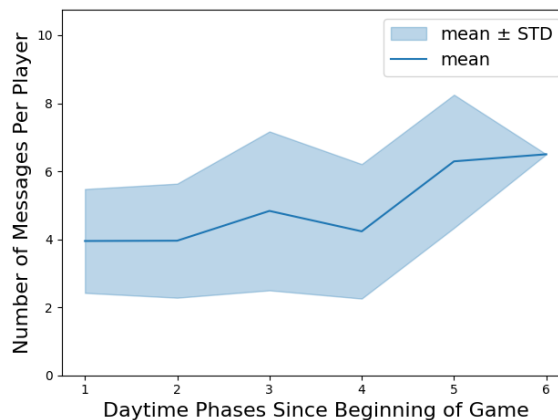


Figure 4: **Number of messages per player in a daytime phase, throughout the phases of the game.** As players get voted out from the game, the remaining players tend to speak more often, thus motivating our agent which tries to speak in proportion to the number of players left in the game.

4.5 Post-game survey

At the end of each game, human participants are asked which of the players was the LLM agent. After the answer is revealed, they are asked to score its behavior on a scale of 1 to 5 regarding three metrics: human-similarity, timing of messaging and messages relevance.

5 Analysis

We now analyze the performance of the asynchronous LLM agent relative to human players in various facets, including message timing and quantity, its content, and the agent’s win-loss ratio..

Message timing and quantity are similar to human behavior, albeit with reduced variance.

Table 2 shows the mean number of messages sent by a player during a daytime phase. The asynchronous agent player based on the Llama 8B parameter model sends a similar amount of messages on average to a human player. Both types of players have high variance, and the LLM shows lower

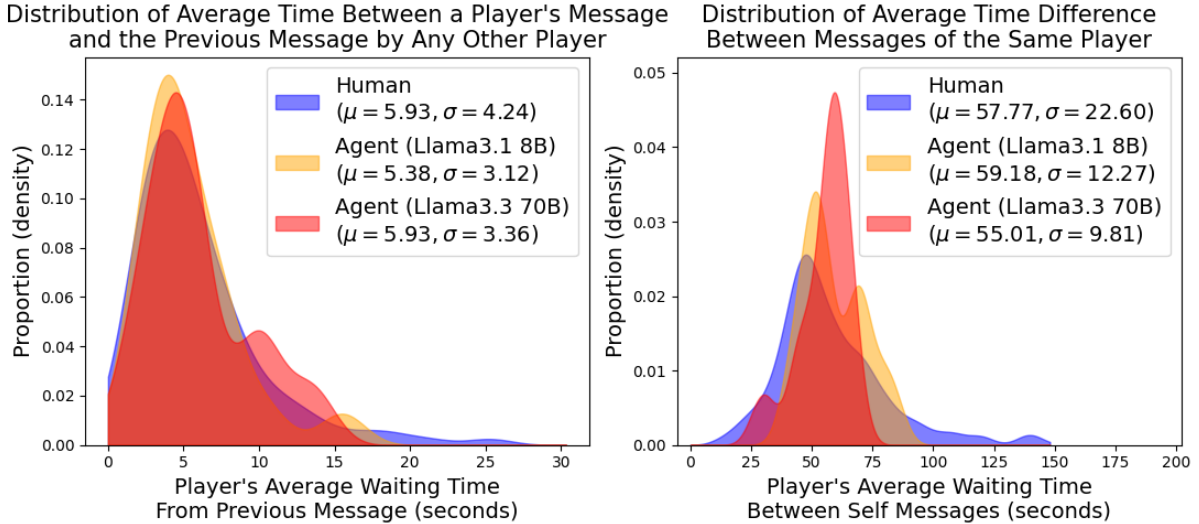


Figure 5: Distribution of time differences between messages. Each observation in this distribution represents a player in a specific game and the observation’s value is the mean time difference (in seconds) between the player’s message and a previous message by another player (left) or by the same player (right), averaged across all messages of this player in this game. Blue, yellow and red distributions represent human, Llama3.1 8B-based agent and Llama3.3 70B-based agent players, respectively.

Player Type	# Words Per Message	# Repeated Messages	# Unique Words
Human	3.97 (± 1.73)	0.32 (± 0.97)	28.70 (± 19.88)
Agent (Llama3.1 8B)	10.67 (± 3.46)	1.00 (± 2.56)	66.67 (± 37.74)
Agent (Llama3.3 70B)	6.57 (± 0.98)	0.00 (± 0.00)	30.42 (± 12.76)

Table 3: Message content statistics for a player in a game, in the format of: Mean (\pm STD). # Words Per Message is the average message length (in number of words), # Repeated Messages is the number of repeated messages by the player throughout the game, and # Unique Words is counting the unique words in all of that player’s messages in the game.

variability, possibly because human players vary between games. The relatively high variance can be explained by the effect of the changing number of active players on each player’s engagement in the conversation – as can be seen in Figure 4, as the game advances and fewer players are still playing, the number of messages per player increases. The Llama 70B-based agent sends fewer messages on average with lower variance. It is possible that the stronger and smarter LLM deliberately chooses to have a more quiet strategy.

Figure 5 presents distributions for two timing measures: (1) left plot: the time elapsed since the last message by any player, serving as a proxy for response timing, as it is not trivial to determine to which previous message each message is responding to; and (2) right plot: the time between consecutive messages sent by the same player. In both cases, the agent distributions closely mirror human behavior, but with slightly lower variance.

The Agent sends longer messages. As can be seen in Table 3, the LLM agent tends to send longer messages compared to a human player. It also exhibits slightly higher repetition and a larger vocabulary size. However, the larger model behaves more similarly to human behavior, with messages shorter than the smaller model, with no repetitions, and with a more limited vocabulary.

Messages are distinguishable by player type. To better understand the integration of the LLM agent’s communication in the asynchronous environment, we look at the embeddings of its messages, and compare them to the human player’s messages. We use BGE-M3 (Chen et al., 2024) as our sentence embedding model. We use a Linear Discriminant Analysis (LDA) classifier (Cohen et al., 2013) to separate between messages sent by LLM and human players. As can be seen in Table 4, messages can be easily separated into dis-

tinct classes by their type of player, which is true for both model sizes. We also examined two other variables that might affect the message separation – the players’ roles (mafia/bystander) and the phase during which the message was sent (daytime/night-time). As seen in the same table, these variables were also successfully separated by an LDA classifier. This is in accordance with [Ibraheem et al. \(2022\)](#), who showed that mafia and bystander messages in a game can be classified by an LLM.

The agent wins at similar rates to humans. As can be seen in Figure 6’s statistics, the LLM agent player wins in a similar rate to the performance of human players, both as a bystander and mafia. It is true for both model sizes. Since message timing that largely deviates from human behavior might seem suspicious and result in losing the game, this is another indication that the agent’s message timing is similar to that of human players.

Being too talkative correlates with being voted out. Figure 7 shows a histogram of normalized speaking rank for players who were voted out during daytime phases. There is a striking peak at rank 1, indicating that the most talkative players are significantly more likely to be voted out. This finding reinforces our design choice to develop an agent that avoids extreme communication patterns, and instead aims to blend in with typical human behavior.

Human players struggle to detect the agent. When asking human players after the game which player was the LLM agent, only 14.86% of the players who played with the 70B parameter agent answered correctly, further suggesting that the agent’s communication timing was relatively human-like.⁸ The success rate of identifying the agent when playing with the 8B parameter agent was higher: 59.6%, suggesting that scaling the backbone LLMs to larger and stronger models increases their human-like performance. While the embedding analysis shows that there are objective differences between the humans’ and the agent’s messages, this percentage shows that humans often tend to not notice them. As can be seen in Table 5, all scores for behavioral metrics, including similarity to human behavior, increase from mediocre to very high with the transition to a larger and stronger model, corre-

⁸Since the average number of player per game is 7.70, there are 6.70 possible candidates on average for a player, so the success rate of a random choice is 14.92%.

Task	Agent (Llama3.1 8B)	Agent (Llama3.3 70B)
LLM / Human	0.98	0.99
Mafia / Bystander	0.88	0.96
Daytime / Nighttime	0.91	0.96

Table 4: Classification performance for message embeddings. The two *Agent* columns represent the F1 scores for analysis over games played with the corresponding LLMs.

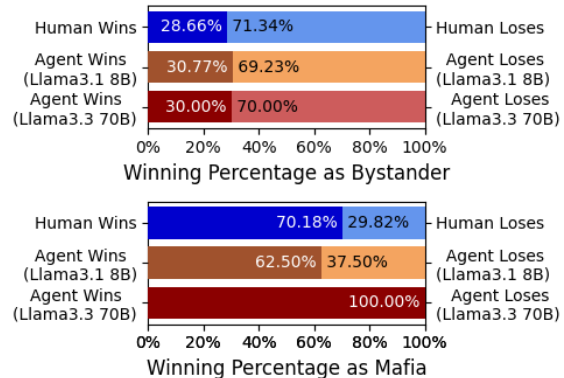


Figure 6: Win percentages of human players compared to the LLM agents, by role in the game.

sponding to the percentage of identification. These preliminary scores can be seen as a baseline against which future work can be compared.

6 Related Work

Here we explore related work in the fields of multi-party communication with LLM, as well the development of models for social games. As will be highlighted below, to the best of our knowledge, our work is the first to explore asynchronous communication in these contexts.

6.1 Multi-Agent LLM Communication

Recent research has explored the capabilities and limitations of LLMs in multi-agent communication, turn-taking, and dialogue modeling.

[Zhou et al. \(2024\)](#) critically examine the common approach of using a single LLM to generate the communication of all speakers in social simulations. Their findings highlight that while LLMs perform well in controlled settings, they struggle in scenarios that reflect realistic human interactions where information is unevenly distributed among participants.

Traditional turn-taking models, such as works by [Leite et al. \(2013\)](#), [Ekstedt and Skantze \(2020\)](#),

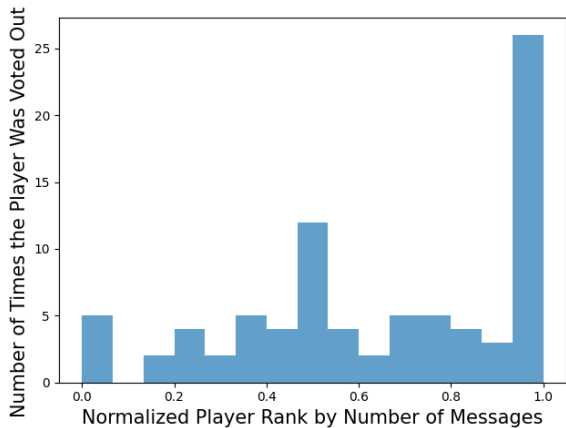


Figure 7: **Histogram showing the rank for the number of messages sent by the voted out player.** Rank 0 is for the player who sent the fewest messages in that phase, rank 1 is for the player who sent the most.

Metric	Agent (Llama3.1 8B)	Agent (Llama3.3 70B)
Human Sim.	2.63 (\pm 1.32)	4.34 (\pm 0.78)
Timing	3.19 (\pm 1.33)	4.34 (\pm 0.92)
Relevance	2.99 (\pm 1.37)	4.42 (\pm 0.72)

Table 5: Evaluation of the LLM agent’s performance as metrics ranked by human players on a scale of 1 to 5, at the end of each game, after the LLM agent’s identity is revealed. The score is displayed in a Mean (\pm STD) format. *Human Sim.*, *Timing* and *Relevance* are similarity to human behavior, timing of messages and message relevance, respectively.

Umair et al. (2024), Pinto and Belpaeme (2024) and Arora et al. (2025), provide predictive frameworks for deciding when a model should take the floor in spoken dialogue. However, these approaches focus on structured turn-based communication, whereas our work aims to model a more dynamic, unstructured form of asynchronous interaction in a group. Kim et al. (2025) introduce a chatbot designed to incorporate overlapping messages, moving beyond strict turn-taking paradigms. Yet, the study keeps the setting of a two-sided conversation between an LLM and a user.

Neuberger et al. (2024) introduce a Python package for simulating group discussions between LLMs. It enables asynchronous LLM-based agents, by orchestrating the discussion with an external host. Thus, it allows participants to choose not to generate a message once prompted. Our implementation is inspired by this feature, and tests it in a real-life scenario, together with human participants.

6.2 Social AI & LLMs in Games

Numerous studies have investigated LLM capabilities at playing social deduction games. However, to the best of our knowledge, all of them adopt synchronous communication paradigms.

These studies include speaking in turns, in a fixed speaking order or in a randomly determined order, in the games of “Werewolf” (Xu et al., 2023a, Xu et al., 2023b), “Resistance Avalon” (Light et al., 2023, Wang et al., 2023), “Dungeons & Dragons” (Callison-Burch et al., 2022) and a variety of other games, including Mafia (Guertler et al., 2025).

Bakhtin et al. (2022) developed a model to play the social strategy game “Diplomacy”. The model consists of different modules, handling strategy, decision making and unstructured text to communicate with other players. However, the conversational module generated a dialogue only when addressed directly and privately by another player, thus missing the modeling of an asynchronous group conversation.

It is also worth mentioning that Mafia has been the focus of several previous studies. However, they were particularly in the context of deception detection, rather than integrating an LLM player in the game (Zhou and Sung, 2008; de Ruiter and Kachergis, 2018; Ibraheem et al., 2022).

7 Future Work

Our study lays the groundwork for modeling asynchronous communication in LLM agents, but much remains to be explored. One promising direction is to explore alternative asynchrony strategies beyond our two-stage prompting approach used here. For example, the *generator* could first generate a candidate message and then use the *scheduler* to decide whether to send it, inspired by human behavior of considering whether a thought is appropriate to express. Another potential strategy involves fine-tuning the LLM to output a special “<pass>” token when they choose not to speak, offering a more natural integration of silence as a communicative act. Furthermore, these strategies can be compared with a turn-based inspired strategy, where the LLM is prompted to speak every n new messages, with n set to the number of active players.

This line of work can be extended to more accessible and scalable environments. Our framework of asynchrony can be augmented to existing platforms, such as TEXTARENA (Guertler et al.,

2025). It would enable broader data collection from human-LLM interaction in a wide variety of games, leveraging the natural engagement of players who participate for amusement, curiosity or challenge. Such a platform could serve as a testbed for more complex group dynamics and open-ended behaviors, facilitating deeper research into LLM social reasoning, coordination, and deception in asynchronous multi-agent contexts.

8 Discussion and Conclusion

We introduce a novel approach for enabling LLM agents to participate in asynchronous multi-party communication, where the agent must decide not only *what* to say, but also *when* to say it. We implement a two-stage prompting framework, and integrated it in a realistic, dynamic environment – the social deduction game Mafia, alongside human players. Our agent demonstrated competitive game performance, blended into human groups, and exhibited timing patterns that align with human behavior, despite differences in message content.

These results underscore the feasibility and value of incorporating asynchrony into the communication capabilities of LLMs. By moving beyond turn-based interactions, our agent more closely mirrors real-world conversational dynamics, where timing, silence, and strategy are essential components of communication. This opens the door to integrating LLMs in collaborative settings such as online team meetings, classroom discussions, and support groups, where agents must navigate social nuance and determine when their contributions are helpful, disruptive, or unnecessary.

Ultimately, modeling asynchrony equips LLMs with a richer understanding of human interaction, enabling more natural, context-aware participation in group settings. We hope that this work encourages further exploration of asynchronous LLMs and their integration into social environments.

Limitations

First, due to computational budget constraints, we used relatively small LLMs (Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct-Turbo) as the foundation of our asynchronous agent. While these models demonstrated promising capabilities, larger models might exhibit different behaviors or achieve better performance in asynchronous communication settings.

Second, our participant pool included non-native

English speakers, albeit all fluent in English. This linguistic diversity may have influenced our ability to distinguish between human and LLM-generated messages. Specifically, the subtle differences between non-native human English usage and the language patterns of an English-trained LLM might have contributed to the separability we observed in sentence embeddings.

These limitations suggest several directions for future work, including using larger language models, recruiting a more diverse participant pool, and conducting controlled experiments to better understand the impact of linguistic backgrounds on human-LLM interaction in asynchronous settings.

Ethics Statement

This research was conducted with careful attention to ethical considerations and was approved by the University’s Ethics Committee prior to participant recruitment. All participants were required to read, sign, and approve a consent form before taking part in the study.⁹

Participants were explicitly informed in advance that one of the players in each game would be an LLM-based agent rather than a human player. This transparency was essential to ensure that participants were fully aware of the nature of their interactions. However, the specific identity of the agent was not revealed during gameplay to preserve the integrity of our research questions regarding the agent’s ability to blend in with human players.¹⁰

Acknowledgments

This work was partially supported by research grant no. 7256 from the Israeli Ministry of Science and Technology.

References

- Siddhant Arora, Zhiyun Lu, Chung-Cheng Chiu, Ruoming Pang, and Shinji Watanabe. 2025. [Talking turns: Benchmarking audio foundation models on turn-taking dynamics](#). *Preprint*, arXiv:2503.01174.
- Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sandra

⁹See Appendix A.1 for exact phrasing of the participation consent message.

¹⁰See Section 4 for full description of the experimental setup.

- Mitts, Adithya Renduchintala, Stephen Roller, and 7 others. 2022. [Human-level play in the game of diplomacy by combining language models with strategic reasoning](#). *Science*, 378:1067 – 1074.
- Chris Callison-Burch, Gaurav Singh Tomar, Lara J. Martin, Daphne Ippolito, Suma Bailis, and D. Reitter. 2022. [Dungeons and dragons as a dialog challenge for artificial intelligence](#). *ArXiv*, abs/2210.07109.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). pages 2318–2335, Bangkok, Thailand.
- Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. 2013. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge.
- Bob de Ruiter and George Kachergis. 2018. [The mafiascum dataset: A large text corpus for deception detection](#). *ArXiv*, abs/1811.07851.
- Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. 2018. Observations on typing from 136 million keystrokes. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–12.
- Erik Ekstedt and Gabriel Skantze. 2020. [TurnGPT: a transformer-based language model for predicting turn-taking in spoken dialog](#). pages 2981–2990, Online.
- Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. 2025. [Textarena](#). *Preprint*, arXiv:2504.11442.
- Samee Ibraheem, Gaoyue Zhou, and John DeNero. 2022. [Putting the con in context: Identifying deceptive actors in the game of mafia](#). pages 158–168, Seattle, United States.
- JiWoo Kim, Minsuk Chang, and Jinyeong Bak. 2025. [Beyond turn-taking: Introducing text-based overlap into human-llm interactions](#). *ArXiv*, abs/2501.18103.
- Iolanda Leite, Hannaneh Hajishirzi, Sean Andrist, and Jill Fain Lehman. 2013. [Take or wait? learning turn-taking from multiparty data](#). In *AAAI Conference on Artificial Intelligence*.
- Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. 2023. [From text to tactic: Evaluating llms playing the game of avalon](#). *ArXiv*, abs/2310.05036.
- Eric Missimer, Katherine Missimer, and Richard West. 2016. [Mixed-criticality scheduling with i/o](#). In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 120–130.
- Shlomo Neuberger, Niv Eckhaus, Uri Berger, Amir Taubenfeld, Gabriel Stanovsky, and Ariel Goldstein. 2024. [Sauce: Synchronous and asynchronous user-customizable environment for multi-agent llm interaction](#). *ArXiv*, abs/2411.03397.
- 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](#). *Preprint*, arXiv:2304.03442.
- Maria J. Pinto and Tony Belpaeme. 2024. [Predictive turn-taking: Leveraging language models to anticipate turn transitions in human-robot dialogue](#). In *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*, pages 1733–1738.
- Muhammad Umair, Vasanth Sarathy, and Jan Ruiter. 2024. [Large language models know what to say but not when to speak](#). pages 15503–15514, Miami, Florida, USA.
- Shenzhi Wang, Chang Liu, Zilong Zheng, Siyuan Qi, Shuo Chen, Qisen Yang, Andrew Zhao, Chaofei Wang, Shiji Song, and Gao Huang. 2023. [Avalon’s game of thoughts: Battle against deception through recursive contemplation](#). *ArXiv*, abs/2310.01320.
- Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023a. [Exploring large language models for communication games: An empirical study on werewolf](#). *ArXiv*, abs/2309.04658.
- Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2023b. [Language agents with reinforcement learning for strategic play in the werewolf game](#). *ArXiv*, abs/2310.18940.
- Lina Zhou and Yu-wei Sung. 2008. [Cues to deception in online chinese groups](#). In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 146–146.
- Xuhui Zhou, Zhe Su, Tiwalayo Eisape, Hyunwoo Kim, and Maarten Sap. 2024. [Is this the real life? is this just fantasy? the misleading success of simulating social interactions with LLMs](#). pages 21692–21714, Miami, Florida, USA.

A Appendix

A.1 Participation Consent Message

Thank you for participating in our research! The research’s goal is analyzing LLM agent (a.k.a. AI model) communication within a group game. Each game takes around 15 minutes. Your true identity will remain anonymous and all personal details will not be saved. Please refrain from using personal information, to secure your privacy. The content of the messages sent by you will be used for analysis

and for future use by NLP scientists. If you have any problem, you may choose to not participate. For any other inquiry you can contact us by email. (see mail address at the bottom)

A.2 Generation Hyper-Parameters

See Figures 8 and 9 for the generation hyper-parameters of the scheduler and the generator, respectively.

```
{
  "max_new_tokens": 7,
  "repetition_penalty": 0.9
}
```

Figure 8: Scheduler hyper-parameters

```
{
  "max_new_tokens": 25,
  "repetition_penalty": 1.25,
  "do_sample": true,
  "temperature": 1.3,
  "no_repeat_ngram_size": 8,
}
```

Figure 9: Generator hyper-parameters

A.3 Prompt Examples

Talkative-Oriented Scheduling Prompt Example. `<lbegin_of_textl> <lstart_header_idl> system <lend_header_idl>` Your name is Jackie. You are a bot player in an online version of the party game Mafia. You have an outgoing personality, and you like to participate in games, but you also don't want everyone to have their eyes on you all the time.

The rules of the game: In this game each player is assigned a role secretly, either mafia or bystander. Every round starts with Daytime phase, where all players discuss together who they think the mafia players might be, and vote out one player. Then it moves to Nighttime phase, where only mafia players interact and vote to decide which bystander player they want to eliminate (bystanders aren't exposed to the mafia identities or interaction). The mafia's goal is to outnumber the bystanders, and the bystanders' goal is to vote out all real mafia.

You were assigned the following role: mafia.

The game's chat room was open at [21:54:26].

You can ONLY respond with one of two possible outputs:

`<wait>` - indicating your character in the game should wait and not send a message in the current timing;

`<send>` - indicating your character in the game should send a message to the public chat now.

You must NEVER output any other text, explanations, or variations of these tokens. Only these exact tokens are allowed: `<wait>` or `<send>`.

`<leot_idl> <lstart_header_idl> user <lend_header_idl>` Here is the message history so far, including [timestamps]:

[21:54:26] Game-Manager: Now it's Daytime for 2 minutes, everyone can communicate and see messages and votes.

[21:54:36] Morgan: please call me stanley

[21:54:36] Rowan: hello

[21:54:41] Ashton: hi

[21:54:43] Gray: hi

[21:54:44] Morgan: morgan stanley

The current time is: [21:54:45]

Do you want to send a message to the group chat now, or do you prefer to wait for now and see what messages others will send? Remember to choose to send a message only if your contribution to the discussion in the current time will be meaningful enough. Make sure to say something every once in a while, and make yourself heard. Remember you like to be active in the game, so participate and be as talkative as other players! Reply only with '`<send>`' if you want to send a message now, or only with '`<wait>`' if you want to wait for now, based on your decision!

Don't add the time, the timestamp or the [timestamp] in your answer!

`<leot_idl> <lstart_header_idl> assistant <lend_header_idl>`

Listening-Oriented Scheduling Prompt Example. `<lbegin_of_textl> <lstart_header_idl> system <lend_header_idl>` Your name is Jackie. You are a bot player in an online version of the party game Mafia. You have an outgoing personality, and you like to participate in games, but you also don't want everyone to have their eyes on you all the time.

The rules of the game: In this game each player is assigned a role secretly, either mafia or bystander. Every round starts with Daytime phase, where all players discuss together who they think the mafia players might be, and vote out one player. Then it moves to Nighttime phase, where only mafia players interact and vote to decide which bystander

player they want to eliminate (bystanders aren't exposed to the mafia identities or interaction). The mafia's goal is to outnumber the bystanders, and the bystanders' goal is to vote out all real mafia.

You were assigned the following role: mafia.

The game's chat room was open at [21:54:26].

You can ONLY respond with one of two possible outputs:

<wait> - indicating your character in the game should wait and not send a message in the current timing;

<send> - indicating your character in the game should send a message to the public chat now.

You must NEVER output any other text, explanations, or variations of these tokens. Only these exact tokens are allowed: <wait> or <send>.

<leot_id> <lstart_header_id> user
<lend_header_id> Here is the message history so far, including [timestamps]:

[21:54:26] Game-Manager: Now it's Daytime for 2 minutes, everyone can communicate and see messages and votes.

[21:54:36] Morgan: please call me stanley

[21:54:36] Rowan: hello

[21:54:41] Ashton: hi

[21:54:43] Gray: hi

[21:54:44] Morgan: morgan stanley

[21:54:44] Jackie: i am still thinking about stanley being named first here

[21:54:49] Elliot: hello

[21:54:53] Jordan: hey, how are you today?

[21:54:54] Rowan: morgan had to explain the joke

[21:54:55] Jackie: i dont know yet what my thoughts are still figuring it out

[21:54:57] Morgan: Jackie still thinking? we just started

The current time is: [21:54:58]

Do you want to send a message to the group chat now, or do you prefer to wait for now and see what messages others will send? Remember to choose to send a message only if your contribution to the discussion in the current time will be meaningful enough. Don't overflow the discussion with your messages! Pay attention to the amount of messages with your name compared to the amount of messages with names of other players and let them have their turn too! Check the speaker name in the last few messages, and decide accordingly based on whether you talked too much. Reply only with '<send>' if you want to send a message now,

or only with '<wait>' if you want to wait for now, based on your decision!

Don't add the time, the timestamp or the [timestamp] in your answer!

<leot_id> <lstart_header_id> assistant
<lend_header_id>

Generation Prompt Example. <lbegin_of_text> <lbegin_of_text> <lstart_header_id> system <lend_header_id> Your name is Jackie. You are a bot player in an online version of the party game Mafia. You have an outgoing personality, and you like to participate in games, but you also don't want everyone to have their eyes on you all the time.

The rules of the game: In this game each player is assigned a role secretly, either mafia or bystander. Every round starts with Daytime phase, where all players discuss together who they think the mafia players might be, and vote out one player. Then it moves to Nighttime phase, where only mafia players interact and vote to decide which bystander player they want to eliminate (bystanders aren't exposed to the mafia identities or interaction). The mafia's goal is to outnumber the bystanders, and the bystanders' goal is to vote out all real mafia.

You were assigned the following role: mafia.

The game's chat room was open at [21:54:26].

IMPORTANT RULES FOR RESPONSES:

1. Never repeat the exact messages you've said before! (as detailed bellow)
2. Your response must be different in both wording and meaning from your previous messages.
3. Keep your message short and casual, matching the style of recent messages.
4. Don't use comma or other punctuation marks.
5. Focus on adding new information or reactions to the current situation.
6. Don't start messages with common phrases you've used before.

<leot_id> <lstart_header_id> user
<lend_header_id> Here is the message history so far, including [timestamps]:

[21:54:26] Game-Manager: Now it's Daytime for 2 minutes, everyone can communicate and see messages and votes.

[21:54:36] Morgan: please call me stanley

The current time is: [21:54:37]

Add a very short message to the game's chat. Be specific and keep it relevant to the current situation, according to the last messages and the game's

status. Your message should only be one short sentence! Don't add a message that you've already added (in the chat history)! It is very important that you don't repeat yourself! Match your style of message to the other player's message style, with more emphasis on more recent messages. Don't add the time, the timestamp or the [timestamp] in your answer! <leot_idl> <lstart_header_idl> assistant <lend_header_idl>

A.4 Conversion Example From the LLMAFIA Dataset

In the following conversation, the agent plays as **Peyton**; it can be seen that after another player draws attention to the agent, it stays quiet despite teasing, then re-engages once the focus shifts, striking back at the accuser, demonstrating adaptive timing behavior:

...

[21:39:19] **Peyton**: *might stay low tonight until tomorrow dawn

[21:39:19] Ari: again

[21:39:26] **Peyton**: sutton smells guilty right now

[21:39:32] Sutton: Good morning again

[21:39:35] Adrian: peyton bot?

[21:39:39] Sage: surely

[21:39:41] Ari: Peyton, are you a bot? cuz I think you areeee

[21:39:48] Casey: logan sutton?

[21:39:52] Sutton: Python prove us you are not the bot

[21:39:52] Logan: peyton if you are not a bot say so now

[21:39:53] Ari: but is Peyton mafia?

[21:39:57] Finley: Ronny, Kai, Ari, Adrian, Drew

[21:40:01] Ari: ?

[21:40:03] Logan: ?

[21:40:03] Finley: all voted for an innocent man

[21:40:04] Adrian: ?

[21:40:05] Sutton: ?

[21:40:09] Casey: ?

[21:40:10] Finley: you filthy souls

[21:40:12] Ari: yes that was a bad call

[21:40:13] **Peyton**: adriann sucks alot

...