Deep Neural Extraction and Analysis of Linguistic Cues from Synchronous Multiplayer Social Deduction Gaming Streams

Master Thesis

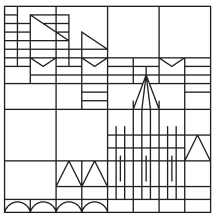
presented

by

Tim Kleinlein

at the

Universität Konstanz



Faculty of Sciences
Department of Computer and Information Science

1. Evaluated by	Jun.-Prof. Dr. Andreas Spitz
2. Evaluated by	Prof. Dr. David Garcia

Konstanz, 2024

Abstract

Social deduction games, with their deceptive and persuasive environments, provide an interesting field for social scientific analysis of human behavior both within and outside the game context. In this thesis, I examine player behavior in discussions of the multiplayer online social deduction game *AmongUs* by analyzing 363 synchronous video on demand streams (VODs) from 41 different game sessions recorded in 2022. To achieve this, I first extract relevant information from the VODs using a multi-step information extraction pipeline.

I begin by applying image recognition, speech extraction methods, and various heuristics to temporally synchronize the VODs and identify key moments in the streams, such as individual game lobbies and the discussions. I then transcribe the discussions and assign the utterances to individual speakers. One of the main challenges I face is speaker diarization - assigning transcribed utterances to specific speakers — due to the complex conversational conditions of the discussions, such as overlapping speech and the large number of speakers. I overcome this challenge by employing the Personal Voice Activity Detection model, a target speaker extraction method. In addition to their utterances, I also identify the respective game roles of the players through image recognition. By combining the extracted speech and identified secret game roles, I create a valuable data set for analyzing player behavior. The combination of deep neural methods and heuristics presents a successful and effective approach for extracting information from synchronous video streams, and can offer potential guidance for future research.

I then analyze the extracted data by fine-tuning a BERT model for text classification to predict the secret game roles of players based on their utterances in the discussions, achieving an accuracy far above random guessing. This demonstrates that large language models can expose players' deceptive behavior to some extent. Thus, I apply feature importance methods to explore the decision-making processes of the model, which yields interesting insights, particularly for individual utterances. While other popular text mining techniques such as sentiment analysis, part-of-speech tagging, word frequency analysis, or topic modeling contribute mainly to the descriptive analysis of the language used in the extracted utterances, they fail to identify patterns that reveal players' secret roles.

Contents

List of Figures

List of Tables

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
Grad-CAM	Gradient-based Class Activation Mapping
IDF	Inverse Document Frequency
LDA	Latent Dirichlet Allocation
LIME	Local Interpretable Model-agnostic Explanations
LLMs	Large Language Models
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
POS	Part of Speech
PVAD	Personal Voice Activity Detection
SET	Score and Embedding Conditioned Training
SHAP	SHapley Additive Explanations
TF-IDF	Term Frequency–Inverse Document Frequency
VAD	Voice Activity Detection
VODs	Video on demand streams

1 Introduction

So-called social deduction games describe games in which players try to find out the unknown roles and team memberships of their fellow players. The players try to deceive the other players through their communication and behavior and convince them of certain beliefs to win the game. In the social sciences, attempts are made to explain human behavior based on theory and empiricism. These type of games are therefore an interesting field for social science analysis: By analyzing the behavior of the players in the game, interesting conclusions can be drawn about human behavior in contexts also outside of the game situation.

1.1 Research Question

This thesis analyzes the behavior of players in the multiplayer online social deduction game AmongUs. In this game, players pursue different goals according to their secret role assigned in the game and interact with other players in discussion rounds. Synchronous multiplayer gaming streams of several streamers from a total of 41 game sessions from the year 2022, which could be downloaded as video on demand streams (VODs) from Twitch, are available for the analysis. The thesis focuses primarily on extracting the players‘ conversations from the discussion rounds in order to explain the players' behavior, in particular by trying to predict the players' secret roles based on the conversations. To extract the relevant conversations of the players, various technical challenges have to be solved. In several steps, a temporal synchronization of the analyzed streams is established, relevant time periods such as the individual games and the discussions within these games are identified, the players' utterances from the discussions are transcribed and assigned to the players, and the game roles of the individual players are identified. After bringing together all the extracted information, a data set is created that contains the utterances made by the individual players as well as their secret role within the game when making these utterances. Based on this data set, I examine three research questions:

- Are large language models (LLMs) able to explain player behavior, i.e., is it possible to train a classifier based on a LLM to predict the players' secret roles based on the language used by them?
- If LLMs are able to do so, how do they accomplish this, i.e., what patterns do they base their decisions on and what do these patterns tell us about human behavior?
- How well are popular text mining methods suited to investigate player behavior in social deduction games? Do they provide purely descriptive insights or can they also recognize patterns in the utterances that enable, for example, a successful role prediction?

To study whether LLMs are able to explain player behavior, this thesis will primarily attempt to predict the secret roles of the individual players based on their utterances. This attempt is based on the assumption that a LLM can identify certain behavioral patterns for the respective roles in the discussions. If it can predict a player's role successfully, the player behaves according to these identified behavioral patterns. The behavioral patterns identified by the LLM are therefore correct and it can explain the player's behavior to a certain extent. I therefore investigate to what extent a LLM can be trained to predict the player's role based on the utterances made by the player. To do this, I fine-tune a Bidirectional Encoder Representations from Transformers (BERT) model for text classification based on the extracted utterances of the players. Then I evaluate its performance.

Then, I use a feature importance method to analyze which patterns such a classification model can use to achieve success in predicting the role. In this way, I investigate the extent to which human behavior in discussions can be generalized and the extent to which people unconsciously orient themselves to certain patterns in discussions. On the one hand, I try to gain interesting insights into the decision-making of the model based on all utterances of the data set. On the other hand, I also examine individual utterances in order to investigate the importance of individual tokens for the exposure of a player.

In a further analysis thread, I examine the utterances using popular text mining methods. By applying part of speech (POS) tagging, a word frequency analysis, a sentiment analysis, or a topic analysis, I carry out a descriptive analysis of the language used. In addition, I investigate whether these methods can be used to identify certain regularities for certain player roles that make it possible to predict them.

1.2 Contribution

This work demonstrates that it is possible to extract high-quality information from synchronous multiplayer gaming streams, enabling a subsequent behavioral analysis of players. By applying several steps in an information extraction pipeline, I successfully extract relevant data from 363 VODs from 41 sessions of the online game Among Us. The dataset contains utterances made by 8 different streamers during in-game discussions, where players vote to eliminate suspicious participants. Additionally, the dataset includes the secret game roles of the players at the time of each utterance, crucial for understanding their objectives and behaviors. Beyond the dataset itself, several sub-steps in the pipeline provide valuable insights into the extraction process, laying the groundwork for future research. Below, I outline key contributions across the major stages of the pipeline.

Temporal Synchronization of VODs In the first step, I apply image recognition and heuristics to temporally synchronize multiple VODs of a session. This synchronization is crucial, as it allows for the meaningful interpretation of player interactions by aligning the behavior and utterances of different streamers of the same game session.

The process relies on identifying game lobbies as temporal anchors. Image recognition is used to detect possible lobby times in individual VODs, and then, with the help of heuristics, I determine session-wide lobby times by matching timestamps of these possible lobby times across streams. From there, I calculate the average time intervals between streamers to synchronize their

VODs. This synchronization is essential for subsequent analysis and serves as a strong foundation for extracting key moments in the streams, such as discussion rounds. The methods applied here can inspire future approaches to temporal synchronization and key moment extraction in other synchronous video or audio streams.

Text Extraction from Discussions The second phase of the pipeline involves extracting text data from discussions within the game. Using speech extraction techniques, I identify time periods with multiple speakers, which indicate discussion rounds. Heuristics are then applied to refine these time periods across different VODs in the same session. This combined approach offers an efficient way of identifying relevant discussions based on linguistic cues.

Once the relevant discussion rounds are identified, I transcribe the utterances. Although transcription itself is not novel, the challenge lies in speaker diarization, especially given the nature of the data: multiple speakers, overlapping conversations, interruptions, and interjections. Traditional speaker diarization methods struggle in this environment, so I implement a target speaker extraction model to handle this complexity. Using the Personal Voice Activity Detection (PVAD) model with a Score and Embedding Conditioned Training (SET) architecture, I classify audio into "Target Speaker Speech," "Non-Target Speaker Speech," and "No Speech." The model performs well even in the challenging conditions of the discussions, thanks to its ability to utilize known speaker profiles as input. This recommends the use of such a model architecture for audio sequences with a comparable profile in the future.

Artificial Creation of Training Data for Speaker Diarization To achieve effective speaker diarization, I generate artificial training data from the VODs. The process involves two key steps:

Identifying and extracting individual speaker utterances outside discussion rounds as unique audio snippets for each speaker. Combining these snippets to generate artificial, labeled conversations for training the PVAD model. These conversations, paired with speaker-specific audio vectors that can be calculated for each streamer using his unique audio snippets, enable the model to recognize and distinguish between speakers during the complex in-game discussions. This method of creating artificial training data serves as a generalizable approach for how training data for target speaker extraction models can be created to allow the use of such a model architecture for speaker diarization tasks: this process can generally be applied to all streams in which there are passages in which the streamer talks to other people as well as passages in which they hold monologues.

Image Recognition for Role Identificatio Another contribution is the extraction of players' secret roles using image recognition. The relevant sections of the VODs, typically at the start of a new game lobby, were identified earlier in the pipeline. The secret roles are used later in the analysis to understand player behavior and serve as labels in classification tasks. This demonstrates how simple information extraction from a video stream by image recognition can be used as part of a larger extraction and analysis pipeline.

Application of LLMs for Behavioral Analysis Once the dataset is prepared, I apply NLP techniques to analyze player behavior. A significant contribution here is the successful prediction of players' secret roles using a fine-tuned encoder model. By analyzing the utterances of the

respective player, the model achieves an accuracy significantly above random guessing, showing that LLMs can detect behavioral patterns in players' language.

Further investigation with feature importance methods, such as Integrated Gradient, provides insights into the model's decision-making process. While general patterns are difficult to detect, interesting findings emerge from analyzing individual utterances, giving an idea of how deceptive behavior is revealed in player's language. These findings suggest that LLMs have potential for analyzing both in-game behavior of the players, but also about human behavior outside the game context.

2 Related Work

Analyzing player behavior in social deduction games involves understanding how individuals interact, communicate, and strategize within the context of these games. Several studies delve into different aspects of player behavior in social deduction games. Brandizzi et al. [2021] focus on how communication is used to support cooperation in social deduction games, highlighting how players communicate to deduce each other's hidden intentions. Wiseman and Lewis [2019] investigate the data sources that players rely on in social deduction games, shedding light on the information players use during gameplay. To do this, they have players engage in an iterative version of the Prisoner's Dilemma and analyze the player-reported data regarding the information they use.

Much research on social deduction games attempts to develop the most successful AI agent for the social deduction game being analyzed. For example, Chuchro [2022] trains an AI-based agent for the game Avalon. In this game, the agent has to take part in votes in various rounds of the game and propose certain missions. Based on game data on the voting behavior and mission proposals of human players, the agent trained with a linear support vector classifier achieves a performance far above that of human players in some game situations. Serrino et al. [2019] also develop an agent for Avalon that exceeds human performance. They train the agent using reinforcement learning.

In order to be successful in social deduction games, it is important to understand the persuasive behavior of the players. This is why there is also several research that does not focus exclusively on finding the winning strategy, but instead sets itself the task of investigating the (decision-making) behavior of the players. Fang et al. [2022] explore the role of biosignals in gameplay strategies and players' physiological synchrony in social deception games by examining the heart rate of players in the social deduction game Mafia to explain their decision-making behavior. Chittaranjan and Hung [2010] develop a model to predict outcomes of the Werewolf social deduction game based on players' speaking and interrupting behavior. They analyze non-verbal audio cues and speaker turns to identify lying, for example, but do not examine the language used. In the social deduction game Hidden Agenda, Kopparapu et al. [2022] investigate how individuals might learn to synthesize potentially unreliable information about others, and elucidate their true motivations. To do this, they use reinforcement learning to train different agents, which learn a variety of behaviors, including partnering with other agents and correct voting without need for communication in natural language.

Most of the persuasion and deception strategies used in social deduction games are based on verbal communication between the players. Prior work has shown LLMs can exhibit cognitive thinking such as formal reasoning, world knowledge comprehension, and deception navigation in group settings [Mahowald et al., 2024]. This is also confirmed in Wang et al. [2023], where various agents based on LLMs are trained in a recursive contemplation framework for the game Avalon, achieving strong performance in the game's deceptive environment. Therefore, there is

much research on the use of LLM agents in social deduction games. Bakhtin et al. [2022] develop an agent for the social deduction game Diplomacy, a strategy game involving both cooperation and competition that emphasizes natural language negotiation and tactical coordination between seven players. Their agent integrates a LLM with planning and reinforcement learning algorithms by inferring players' beliefs and intentions from its conversations and generating dialogue in pursuit of its plans. Xu et al. [2024a] let different LLM-based agents play against each other in the game Werewolf and demonstrate that they show strategic behavior, which suggests an understanding of LLMs of deceptive behavior. Xu et al. [2024b] also have an agent based on an LLM play Werewolf with additional training for its decision-making with reinforcement learning. This allows the agent to achieve an above-human performance, for example in the task of predicting the roles of the other players. Wu et al. [2024] also achieve impressive performance in the game Werewolf, by enhancing the reasoning capabilities of their LLM-based agent with an external Thinker module. The Thinker module thereby has domain-specific knowledge, using data from human sessions and being trained with reinforcement learning.

Agents based on LLMs are also developed for the social deduction game AmongUs analyzed in this thesis. Chi et al. [2024] developed the framework AMONGAGENTS, a fully text-based game environment, that mirrors the dynamics of AmongUs. In this environment, they have different agents based on generative LLMs to compete against each other. The agents carry out actions in the game (such as completing various tasks given by the game) and communicate with each other in the discussion rounds. The agents base their game decisions not only on conversations with other players, but also on interactions with the game environment itself, for example by checking surveillance cameras. Thus, the agents must develop skills to remember, reflect, interact with other agents, and plan through evolving complex circumstances. The results show that state-of-the-art LLMs can understand the rules of AmongUs and make decisions based on the game context.

The previously presented works analyze the application of LLMs in social deduction games by having agents based on LLMs compete against each other in simulated game rounds. A few studies analyze the application of LLMs to utterances made by human players in social deduction games. The above-described agent trained by Xu et al. [2024b] for the game Werewolf competes against human players in an evaluation step. The agent achieves an above-average performance compared to the human opponents, meaning that it appears to be able to extract valuable information from the human utterances. Lai et al. [2022] examine a large data set of dialogue transcriptions and videos for the game Werewolf. They use a multimodal approach, i.e. in addition to LLMs, they also use the dialogue context and visual signals to predict the persuasion strategies used by the players. Based on these applied persuasion strategies, in a further step they make an attempt to derive the outcomes of various games. To the best of my knowledge, the behavior and language of human players for the game AmongUs have not yet been investigated. This thesis therefore makes a further contribution to the application of LLMs for the behavioral analysis of human players in the setting of a social deduction game.

3 Background

3.1 Image Recognition

I carry out the extraction of lobby times and the identification of game roles from the VODs using image recognition. I use several existing methods, which are briefly presented below.

Frame Extraction Using *FFmpeg* The *FFmpeg* tool, a free and open-source suite of libraries and programs for handling and processing multimedia files, is used to extract the images [ffmpeg, 2024]. *FFmpeg* is widely used in both industry and research for its versatility and efficiency in handling various multimedia tasks, therefore making it a good choice for an efficient frame extraction.

Recognizing Text From Images Using OCR For recognizing text from the extracted images, I use Optical Character Recognition (OCR). OCR is a technology that recognizes text from media. Traditionally, such media are scanned documents or photographs. However, virtually any kind of image containing written text, be it typed, handwritten, or printed, can be converted into machine-readable text. Before applying OCR to the extracted frames, a systematic approach involving several preprocessing steps can significantly improve the accuracy and efficiency of text extraction from images. The steps involved in this approach include grayscale conversion, dark frame identification, cropping, binary thresholding, and morphological operations. Grayscale conversion simplifies the detection process by reducing the complexity of the image data [Patel et al., 2012]. Dark frame identification helps filter out non-relevant frames with black backgrounds, focusing the OCR on the relevant text regions [Stubberud et al., 1996]. Cropping the images to specific trigger text locations, such as "your role is" ensures that the OCR engine targets the critical areas for text extraction [Patel et al., 2012]. Binary thresholding aids in creating binary images that distinguish text from the background, facilitating better text detection. Additionally, morphological operations, such as opening, can remove noise and enhance text visibility, further improving the OCR results [Stubberud et al., 1996]. After the preprocessing I utilize *Tesseract*, an open-source OCR engine, to extract text from images. *Tesseract* is known for its capability to recognize text characters within digital images, making it a valuable tool for document analysis [Nugroho et al., 2024]. In order to find lobby start and end times or game roles for the extracted images based on the recognized text, I use the Levenshtein distance as a measure. The Levenshtein distance, as defined by Damerau, quantifies the minimum number of single-character edits required to transform one sequence into another, encompassing operations like insertion, deletion, and permutation [Hamza et al., 2014]. This metric serves as a fundamental tool for comparing and evaluating the similarity between text strings, providing a measure of the difference or similarity between sequences based on the minimal number of operations needed for transformation

[Lawrence et al., 2021]. In both use cases, I always assign a frame to a specific event (lobby start or end) or a specific role if the Levensthein Distance does not exceed a certain threshold.

3.2 Transcription

The Open AI Whisper model in size large is used to transcribe the discussion rounds and the individual utterances of the streamers. The OpenAI Whisper model is an automatic speech recognition system that uses a transformer-based architecture trained on 680,000 hours of multilingual and multitask data, achieving high robustness and accuracy across various languages and challenging audio conditions. It supports tasks such as language identification, phrase-level timestamps, translation, and as used in my case, transcription [Radford et al., 2022].

3.3 Speaker Diarization

Task of Speaker Diarization In two steps of the information extraction pipeline, I must solve the task of speaker diarization. Speaker diarization describes a multistage process involving speech detection, speaker audio segmentation, speaker clustering, and speaker identification to label audio or video recordings with classes corresponding to speaker identity [Žibert and Mihelic, 2008]. This process aims to identify "who spoke when" by segmenting and clustering audio data based on speaker characteristics, enabling applications such as speaker recognition, speech processing, and audio analysis [Pande and Kale, 2023, Bai and Zhang, 2021].

Speaker Diarization Using *Pyannote* In the first step where I perform speaker diarization in this thesis, I make an attempt to find the discussion and movement phases in the individual games. For this purpose, I identify phases in which several speakers speak and phases in which only one person speaks within the analyzed streams. I use the popular speaker diarization model from *Pyannote* for this. The *Pyannote* speaker diarization model is an open-source toolkit designed for segmenting audio recordings by speaker, leveraging end-to-end neural building blocks within a *PyTorch* framework. This model includes modules for tasks such as voice activity detection, speaker embedding, and clustering, which are optimized jointly to enhance diarization performance and accuracy across diverse data sets [Bredin, 2023].

Target Speaker Extraction Methods for Speaker Diarization In the second step, speaker diarization of the discussion rounds, I can not use a conventional speaker diarization method. The reason is that many pre-trained models, even the best ones, such as the *Pyannote* speaker diarization model, achieve results with detection error rates in the range of 90% when applied to test audios of the discussion rounds, which is far too high for a subsequent analysis of the text data to make sense. This is firstly due to the fact that most of the models are trained to perform speaker diarization on audio data with few speakers (usually two to four), which is significantly exceeded in the present use case with on average 11.4 speakers in a lobby. Secondly, the models are usually specialized in non-overlapping conversations, which is not the case here, as the streamers often interrupt each other in the discussions or speak at the same time. However, the use case also has two advantages over conventional audio data: Firstly, the quality of the audio

signals is relatively good due to the audio setup optimized by the streamers. Secondly, the speakers involved in the conversation are known, which is why their speaker profile can be provided to the speaker diarization model as additional input. This approach has already been applied in other works and has always contributed to an immense reduction of the detection error rate [Medennikov et al., 2020]. One of the most promising methods using speaker profiles is end-to-end neural diarization, where the diarization is performed in a single stage and the frame-level acticity probabilities of each speaker are output separately [Fujita et al., 2019]. In another approach, the speaker profiles are given as input in order to direct the focus of the developed system to the characteristics of this speaker and to extract only the speech passages of this target speaker. This direction is represented by such approaches as Target-Speaker ASR [Kanda et al., 2019], Speaker Beam [Žmolíková et al., 2017, Delcroix et al., 2018], and Voice Filter [Wang et al., 2019] aimed at the target speaker speech extraction. A specific representative of this target speaker recognition is the PVAD approach [Ding et al., 2020]. With an approach based on this method promising results were achieved in the CHiME-6 Challenge [Medennikov et al., 2020]. The CHiME-6 Challenge focuses on speech recognition in complex, real-life environments using multi-channel recordings from dinner party scenarios captured by Kinect arrays across three rooms [Watanabe et al., 2020]. Just like the recordings from the discussion rounds, these recordings have a large proportion of overlapping speech. In track 1 of this challenge, which comes closest to the present use case due to the permission of manual segmentation of the audio sequences of the participating streamers to form the speaker profiles model input, an impressive detection error rate of 37.4% was achieved with a Target-Speaker VAD model, which is based on a PVAD model. Thus, I choose a PVAD model as the model architecture for the speaker diarization of the discussion rounds. The selected PVAD model will be briefly described in the following.

PVAD Model Architecture The PVAD is a system for detecting the voice activity of a target speaker at frame level. In the setup of this thesis the frames have a length of 10ms. For each frame, the model calculates the probabilities for three classes: No Speech, Target-Speaker-Speech, Non-Target-Speaker-Speech. To do this, the model receives two vectors as input: First is the audio sequence for which the target speaker extraction is performed. To convert the audio sequence to a vector that can be input into the PVAD model, the audio is first divided into overlapping frames of 25ms width with a 10ms step. Each frame is then transformed into a 40-dimensional log Mel-filterbank energy vector. This vector representation captures the spectral properties of the audio signal, which are essential for distinguishing between speech and no-speech segments as well as between different speakers. I use the open-source Python package *librosa.feature.melspectogram* for this step [McFee et al., 2015]. As second input the model takes the speaker profile of the target speaker encoded in a so-called *d*-vector [Wan et al., 2020]. A *d*-vector is an embedding that represents the voice characteristics of a target speaker. It is determined using multiple utterances of a target speaker. The calculation of *d*-vectors also involves processing Mel-filterbank features that are extracted from several neighboring frames of an audio sequence. These frames are consecutive segments of audio, typically overlapping, that provide a short window of sound data, usually 10ms in length each. By combining features from multiple neighboring frames (i.e., frames that are close together in time within the same utterance), one captures both the present and adjacent temporal contexts, allowing for a more comprehensive representation of the acoustic characteristics at that moment. These combined feature vectors are then passed through a retrained 3-layer Long Short-

Term Memory (LSTM) speaker verification model in a next step. Importantly, *d*-vectors are not derived from a single utterance but are instead calculated using multiple utterances from the target speaker. For each utterance, at each time step, the activations from the last hidden layer of the LSTM are extracted, L2-normalized, and averaged over the entire utterance. The final *d*-vector is then computed by averaging these utterance-level embeddings across all the provided utterances of the target speaker, thereby capturing a comprehensive and robust representation of the speaker's unique voice characteristics. The *d*-vector extractor implementation used in this work is called *Resemblyzer3*. It is a freely available community implementation of the text-independent speaker verification method proposed in Wan et al. [2020].

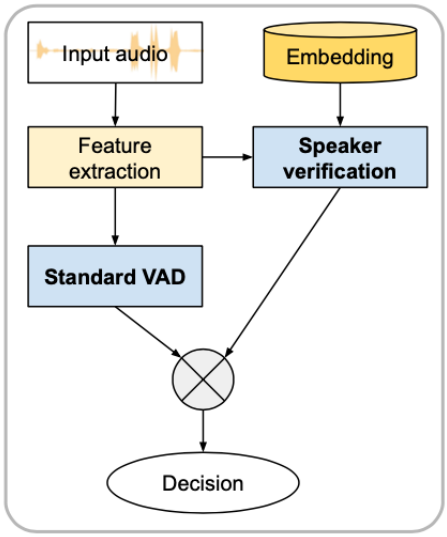


Figure 3.1: Score Combination Architecture to Implement PVAD.

In the original work on the PVAD model from Ding et al. [2020], four different variants of the model are presented. For didactic reasons I present the simplest variant, which is displayed in Figure 3.1, first, before I describe the more advanced variant which I use in the thesis. In the simplest approach a standard voice activity detection (VAD) and a standard pre-trained speaker verification system are combined, meaning the vector with the acoustic features of the audio sequence is fed into both a classic VAD model and a speaker verification model independently for each frame. In the following I denote the frame of the input acoustic features at time t as $x_t \in \mathbb{R}^D$, where D is the dimensionality of the acoustic features. In the application of this work it is 40, as I use 40-dimensional log Mel-filterbank energies as the features. I use subscript $[t]$ to denote the subsequence ending at time t, i.e. $x_{[t]} = (x_1, ..., x_t)$. As described, the vector with the acoustic features of the audio sequence is fed into a standard VAD model, $f_{\text{VAD}}(\cdot)$, for each frame, which outputs unnormalized probabilities for speech (s) and no-speech (ns) for each frame:

$$z_t = f_{\text{VAD}}(x_{[t]}), \tag{3.1}$$

where $z_t = [z_t^s, z_t^{ns}]$. In addition, the vector with the acoustic features is also fed into a speaker verification model, $f_{\text{SV}}(\cdot)$, which produces an embedding of the analyzed utterance for each frame:

$$e_t = f_{\text{SV}}(x_{[t]}). \tag{3.2}$$

This embedding captures the acoustic characteristics of the analyzed audio sequence, which is why its cosine similarity with the *d*-vector of the target speaker, e^{target}, is taken as a measure of similarity with the speaker profile of the target speaker:

$$s_t = cos(e_t, e^{target}). \tag{3.3}$$

At this point the model has calculated for each frame the probability of speech, z_t^s, and the acoustic similarity with the target speaker, s_t. Coming from these values for each frame the PVAD probabilities of the three classes Target Speaker Speech, Non-Target Speaker Speech, and No Speech can be calculated in the following way:

$$z_t^k = \begin{cases} s_t \cdot z^s, & \text{if } k = tss, \\ (1 - s_t) \cdot z^s, & \text{if } k = ntss, \\ z_t^{ns}, & \text{if } k = ns. \end{cases} \tag{3.4}$$

Figure 3.2 displays the architecture of the PVAD model that performs best, and which is also used in this thesis. In this architecture the acoustic feature vector x_t is first concatenated with the frame-level speaker verification score s_t and the target speaker embedding e_{target} before being given as input $\hat{x}_t$ to a PVAD model to calculate the PVAD probabilities. Since the acoustic feature vector is 40-dimensional, the embedding is 256-dimensional, and the frame-level speaker verification score is 1-dimensional, the concatenated feature vector in this approach is 297-dimensional:

$$\hat{x}_t = [x_t, e^{target}, s_t]. \tag{3.5}$$

The PVAD model is a 2-layer LSTM network with 64 cells, followed by a fully-connected layer with 64 neurons.

Creation of Training Data for PVAD To create training data for the described model using the PVAD architecture, I follow a method based on the approach from Ding et al. [2020]. First, I extract individual utterances from different streamers / speakers using their VODs. Then, to mimic real conversational speech, I concatenate utterances from several speakers to generate

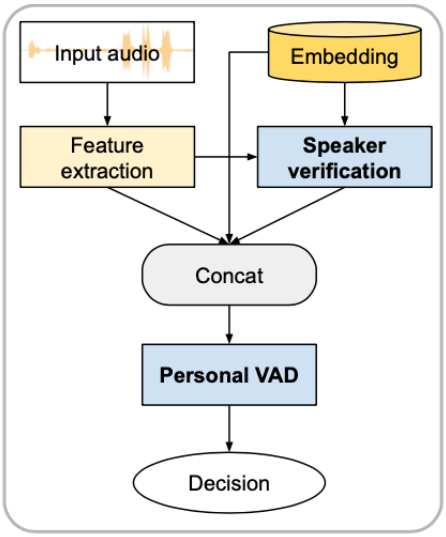


Figure 3.2: SET Architecture to Implement PVAD.

single, longer utterances. For each of these concatenated utterances, I choose a random number n representing the number of utterances to concatenate from a uniform distribution:

$$n \sim Uniform(a, b), \tag{3.6}$$

where a and b are the minimal and maximal numbers of utterances used for concatenation. For these concatenated utterances, I randomly pick one speaker as the target speaker. Then I label each frame in the concatenated utterance with one of three labels: "No Speech", "Target Speaker Speech", or "Non-Target Speaker Speech". I assign these labels based on whether the speech came from the target speaker and if any speech was present in that particular frame (in contrast to the original method proposed, to know this I had to apply a transcription method to the individual utterances beforehand).

3.4 Used LLMs and Text Mining Methods

Fine-Tuning a BERT Model for Player Role Prediction To test the extent to which LLMs can predict player behavior, I fine-tune a BERT model for the task of predicting a player's role based on their utterances. BERT is a language representation model developed by Google. It is designed to pre-train deep bidirectional representations by jointly conditioning on both the left and right context across all layers. During pre-training, BERT predicts words that are masked out from the input text, and it can be subsequently fine-tuned for specific tasks, such as text classification as in this study, with minimal additional parameters. BERT's architecture is based on the Transformer model, which uses self-attention mechanisms to capture complex patterns in lan-

guage data. This approach has significantly improved performance benchmarks across various natural language processing (NLP) tasks, including text classification [Devlin et al., 2019]. Fine-tuning BERT for text classification tasks has become so common that research has been conducted to optimize this process. For example, a paper by Sun et al. [2020] explores various fine-tuning methods for BERT in text classification tasks. The study conducts extensive experiments to evaluate the effectiveness of different fine-tuning approaches and proposes a comprehensive solution for optimizing BERT fine-tuning.

Integrated Gradient as Explainable AI Method For the fine-tuned model I apply feature importance methods to understand how the model derives its classifications. For this I use Integrated Gradients, which is a feature importance method used in text classification models to understand the contribution of each word or token to the model's prediction. This method assigns importance scores to individual words in a text input by calculating the gradient of the model's prediction with respect to the input text. By analyzing these importance scores, Integrated Gradients helps reveal which words or tokens have the most significant impact on the model's decision-making process, providing insights into the text features that influence the classification outcome. The calculation of the feature importance scores can roughly be summarized as follows: Formally, suppose we have a function $F : \mathbb{R}^n \rightarrow [0, 1]$ that represents a deep network. Specifically, let $x \in \mathbb{R}^n$ be the input at hand, and $x' \in \mathbb{R}^n$ be the baseline input. In my case, the baseline is an all-zero token embedding. Next, one considers a straight-line path from the baseline, x', to the actual input, x, and computes the gradients at all points along the path. The integrated gradient along the i^{th} dimension for an input x and baseline x' is defined as follows. Here, $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i^{th} dimension.

$$\text{Integrated Gradients}_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha, \tag{3.7}$$

The result is a set of importance scores for each token in the text input, indicating how much each token contributed to the final classification prediction. Key concepts include the baseline input, which serves as the reference point for comparison, and interpolation, which creates intermediate inputs between the baseline and the actual input to analyze how the model's output changes. Gradient calculation involves finding the derivative of the model's output with respect to its input, providing a local sensitivity measure. Averaging the gradients along the path from baseline to input ensures smooth and comprehensive importance score [Sundararajan et al., 2017].

Various Text Mining Methods Further analyses carried out in the thesis include various methods from text mining, also drawing on existing methods. In the sentiment analysis of the extracted streamer utterances I use "cardiffnlp/twitter-roberta-base-sentiment-latest," a RoBERTa-based model trained on approximately 124 million tweets from January 2018 to December 2021, and fine-tuned for sentiment analysis using the TweetEval benchmark. This model leverages the robust architecture of RoBERTa to classify text into sentiment categories, making it highly effective for analyzing sentiment in social media content. This choice seems reasonable in the context

of online video game streaming, as these utterances often share informal, spontaneous, and dynamic characteristics typical also of social media interactions [Loureiro et al., 2022].

For the word frequency analysis I use the Brown corpus from the Natural Language Toolkit (NLTK), which is a standard corpus of American English used for linguistic analysis, to calculate the Inverse Document Frequency (IDF) of words [Bird et al., 2009].

I also try to identify different topics within the streamers utterances. For this I use Latent Dirichlet Allocation (LDA), a generative statistical model, which discovers underlying topics in a collection of documents by assuming that each document is a mixture of topics and each topic is a distribution of words. In the present use case instead of a set of documents one has a set of single utterances of the streamers. Using LDA on individual sentences can present challenges due to the limited context and word co-occurrence in shorter text segments [Yan et al., 2013], which is probably also why the final results of my topic analysis are not meaningful. I use the implementation of the *LdaModel* from the *Gensim* library which is iterating over the corpus and adjusting the distribution of topics within utterances and words within topics to maximize the likelihood of the observed data. This method is originally proposed by Blei et al. [2003].

In the thesis I also perform POS tagging using the *NLTK* library. The process involves tokenizing sentences and words in each utterance and then tagging each word with its corresponding part of speech (e.g., noun, verb, adjective). The POS tagging helps in identifying and counting the frequency of different parts of speech in utterances for the different roles Crewmates, Neutrals, and Impostors. The *NLTK* POS tagger uses the Penn Treebank tagset, a widely-used resource for linguistic annotation that was developed as part of the Penn Treebank Project [Bird et al., 2009, Marcus et al., 1993].

4 Description of AmongUs and Data

In the thesis, VODs of AmongUs, a popular social deduction game in the online streaming community, are analyzed, which is why the game is briefly presented below. Subsequently, the available data is described.

AmongUs AmongUs is an online multiplayer game developed by Innersloth in which players have to work together as astronauts on a spaceship, base or planetary station to complete various tasks. The game can be played with four to fifteen players. In the base game, most players take on the role of Crewmates, and there are also a certain number of players who act as Impostors. The Impostors have the task of secretly sabotaging and eliminating the Crewmates without being discovered. Crewmates win by either completing all the tasks or identifying the Impostors and voting them out of the game. Crewmates complete the tasks by running around the map and solving simple tasks at the appropriate locations. Impostors can kill players if they get within a minimum distance of them. The corpse of a killed Crewmate remains lying around after elimination and is therefore visible to other players. If other players find a corpse, they can call a discussion round by pressing a button. In this discussion, the players can communicate with each other and try to find out who the Impostors are. Then, suspicious players can be eliminated by voting. The game is therefore basically divided into two phases: The movement phase, in which players move around the map and try to fulfill tasks or eliminate other players according to their role, and the discussion phase, in which players try to vote suspicious players out of the game based on their discussions. Figure 4.1 and Figure 4.2 show screenshots for these two phases of the game. With its mix of teamwork, deception, and strategic thinking, AmongUs has gained worldwide popularity and has become a cultural phenomenon, especially during the COVID-19 pandemic.

In this thesis, VODs of two modifications of the base game were analyzed: Town of Us[1] and The Other Roles[2]. A table showing in which session which modification is played is provided in Appendix A.1. In these modifications, the players have more specific roles with their own abilities in addition to their main role as Impostor or Crewmate. For example, there is the role of the Detective, who as a Crewmate has the ability to see the footprints of other players, or the Cleaner, who as an Imposter has the ability to clean corpses so that they are no longer visible. There is also a third main role, the Neutrals. In addition to specific abilities, these players also have their own goals, such as the role of the Jester, who wins the game if he is voted out in a discussion round. As the two modifications are fundamentally no different from the base game AmongUs, we will continue to refer to AmongUs in the rest of the thesis, even if both modifications are meant.

Overall, AmongUs is well suited for the analysis of player behavior for three reasons: Firstly, there is a clear separation within the game between the movement and discussion phases. This

[1] https://github.com/eDonnes124/Town-Of-Us-R Town Of Us: Reactivated is a modified version of Among Us.
[2] https://github.com/TheOtherRolesAU/TheOtherRoles The Other Roles is a modified version of Among Us.

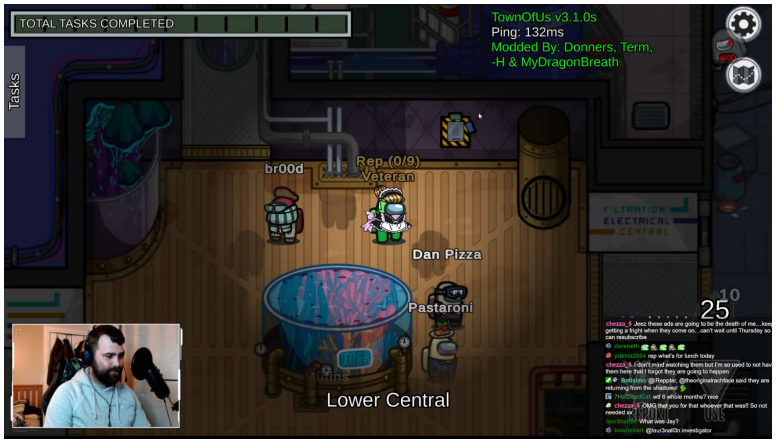


Figure 4.1: In-Game-Screenshot of the Movement Phase.

Description: This figure illustrates the in-game-perspective of a Crewmate. One can see the different players displayed with their names who walk around the spaceship to solve tasks. The player character is in the middle of the screen with the name "Rep" and has the Crewmate role "Veteran". On the top left you can see a progress bar for the completed missions.

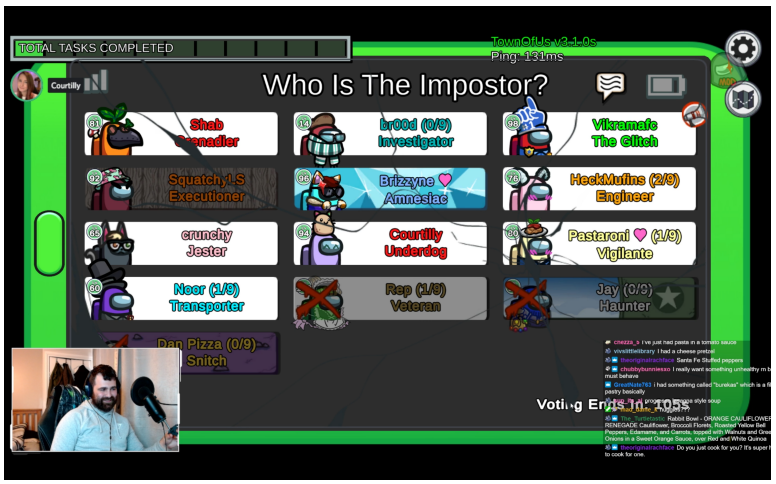


Figure 4.2: In-Game-Screenshot of the Discussion Phase.

Description: The lobby is meeting for a discussion round. Of the 13 players in the lobby, 2 have died already. The remaining players can now vote which player will be ejected. Once the discussion round ends (there are 105 seconds remaining), the player with the most votes will be ejected.

means that all times in which the players communicate verbally with each other are clearly defined

and can be included in the analysis, so there is no risk that verbal communication between the players will not be considered. Secondly, the players involved act with a clear objective function. If you look at the base game, the two main roles Crewmate and Impostor have clear goals: completing all tasks and eliminating all Crewmates. The modifications of the game analyzed in the thesis can also be broken down to the base game by looking exclusively at the game rounds of the streamers in which they were either Crewmate or Impostor. The more specific roles assigned to them give the players additional abilities, but do not change the basic winning condition. The relatively low complexity of the winning conditions achieved in this way makes it possible to achieve meaningful results in the player behavior analysis. Thirdly, AmongUs and its modifications enjoyed great popularity in the online streaming community, which is why there are a large number of available streams that serve as a data basis for the analysis. In particular, it is positive to mention that for all sessions there are recorded streams from several streamers. This is a great advantage, as the synchronization of the different streams of a game session made many steps of the information extraction pipeline possible in the first place, such as the identification of lobby and discussion times.

Data The analyzed streams were originally streamed on Twitch and were available for download as a VOD stream for some time after the original broadcast. In total 394 VODs were originally downloaded, but only 363 of these VODs are used, as the gaming session is only known for these. The VODs have an average duration of 3 hours and 5 minutes. The VODs come from 41 different sessions from games played in the period from 25.01.22 - 24.05.22. Several streamers play together in a session (on average there are 11.4 streamers per session, although only 8.9 streamers have made their streams available for download). A VOD is the recording of one session of one streamer. Several lobbies are played within a session. A lobby is an independent, completed, individual game round. At the start of the thesis for each of these VODs there exists a mkv file (video + audio) and an entry in a VOD database containing metadata about the stream (Publishing date, Game session, Title, …). In a prior step, Dang [2022] made an attempt to identify different lobbies for each of these VODs using image recognition: Individual frames were extracted for each VOD (at a rate of 3 frames per second), which were then converted to a 1280 x 720-pixel resolution. Since the lobby start and lobby end events in AmongUs each have a specific splash screen with a specific text, the extracted frames were examined for this pattern in order to identify the correct timestamp of these events. For this purpose, the frames were first prepared accordingly (identification of dark frames, zooming to the trigger locations, application of binary thresholding and morphological operations) before the text was analyzed with OCR and, if applicable, the frame was assigned to one of the two events. The events identified in this way were saved for each VOD in a separate srt file. The srt file contains the start and end time for each identified event and thus a duration, as well as a text describing whether it is an identified lobby start or an identified lobby end.

5 Methodology

The following section describes the pipeline for extracting information from the streamed VODs in order to carry out an analysis of player behavior. To provide an overview, the pipeline is first described superficially before the individual steps are discussed in more detail in the following chapters. The pipeline is displayed in Figure 5.1. The implementation of the pipeline is mainly carried out using *Python*[1].

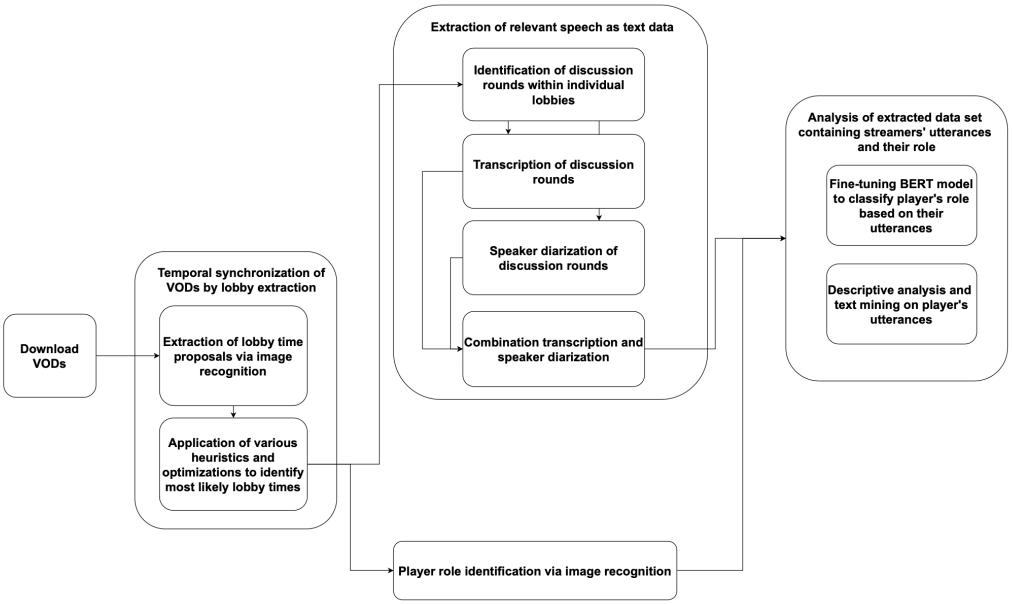


Figure 5.1: Information Extraction and Analysis Pipeline.

Download VODs The information extraction process begins with the download of the VODs from Twitch. In addition to the mkv file, I also download metadata for each VOD to allow the individual VODs to be assigned to a game session.

Temporal Synchronization of VODs by Lobby Extraction In a second step, a temporal synchronization of all VODs of a session is established. This is achieved by identifying all

[1] https://gitlab.inf.uni-konstanz.de/tim.kleinlein/among-us-analysis GitLab Repository among-us-analysis.

the lobby times for each VOD in the session. To identify potential lobby times for each VOD in a session, I apply image extraction followed by OCR. Based on these potential lobby times of the individual streamers, I determine the most probable lobby times of the entire session. For this purpose, the potential lobby times of the individual VODs are combined with various heuristics and optimization techniques. In this way I can determine robust and correct session-wide lobby times despite the identified potential lobby times of the individual VODs being incorrect in some cases due to inaccurate image recognition. In the end, I identify the individual lobbies with their respective times for each VOD and thus for each streamer for each session, making it possible to extract them. On the one hand, this is helpful as only in these sections of the VODs the game is played and therefore only these are relevant for the analysis. On the other hand, the corresponding times in the respective VOD is now known for each lobby of each session for each streamer, which creates a temporal synchronization of the different VODs of a session.

Extraction of Relevant Speech as Text Data The most complex part of the extraction pipeline is extracting the relevant text data from the players' discussions. To do this, the first step is to identify the time periods of the discussion rounds for each VOD for each lobby. For this step, the fact that the players in AmongUs can only communicate with each other during the discussion phases of the game can be used. In the movement phases, on the other hand, the other players are muted, which is why the streamers only talk to their chat during this phase. It can therefore be assumed that several speakers speak during the discussion phases, while only one speaker speaks during the movement phase. In order to identify the respective phases on this basis, I apply a conventional *Pyannote* speaker diarization model to the audio of each lobby. This model identifies all the speakers involved in an audio sequence and outputs the time windows in which they speak. In this way, candidates for discussion rounds can be created for each VOD for each lobby by identifying time periods in which several speakers are detected. By comparing the identified candidates for discussion rounds of all VODs of a lobby, the probable discussion rounds of a lobby can be determined by applying some heuristics.

In a next step, I transcribe the discussion rounds identified in this way using the Open AI Whisper Model. It should be mentioned that I transcribe a discussion round not just once, but for each streamer involved in the discussion round. This is due to the fact that the streamers often interrupt each other in the discussions and talk at the same time, which is why the transcription does not always lead to exactly the same results. In future extraction and analysis steps, for each streamer the transcription from their VOD will be used, as it can be assumed that the streamer's utterances are best understood in their own stream.

In order to assign the transcribed statements to the individual streamers, the task of speaker diarization must be solved, which identifies which speaker is speaking when for a given audio sequence. Due to the special characteristics of the discussion rounds (many overlapping conversations and many different speakers who are known in advance), this task is solved with a PVAD model, which as a target speaker extraction model filters out the times at which the target speaker speaks from audio sequences. In addition to the audio sequence to be classified, the model also takes an embedding of the target speaker's voice characteristics as input.

In order to train such a PVAD model, corresponding training data must be created. For this purpose, I first extract utterances in which only the streamer speaks as audio snippets from each streamer's streams. I find these utterances by using the *Pyannote* speaker diarization model to

identify the times in the identified movement phases when the streamer is holding a monologue or talking to his chat. Based on these audio snippets of the individual streamers, I can create artificial conversations with several speakers by randomly concatenating them. Since the resulting conversations are labeled (it is known which speaker is speaking at any given time), I can use them as training input for the PVAD model. In a further step, I extract the relevant features from them in order to pass the sequence to the PVAD model as vector input. The second model input, a vector that defines the audio characteristics of the target speaker, can also be calculated based on the audio snippets of each streamer. For this purpose, I use 100 utterances of each streamer to determine his so-called *d*-vector. Now that the training data has been artificially created, the model training is carried out. To do this, for each of the artificially created conversations, I randomly select a target speaker with his respective *d*-vector embedding; the exact configurations of the training are described in the chapter on speaker diarization.

After successful model training, I apply the model to the discussion phases. As output the model classification (No Speech, Target Speaker Speech, Non-Target Speaker Speech) and the associated probabilities are available for each discussion phase for each of the 10ms frames. In a final step, I bring the transcription together with this speaker diarization model classification so that each speaker can be assigned with his utterances. Here again, I use some heuristics such that the utterances that the transcription method outputs as coherent, are always assigned to a specific speaker if the number of frames of this utterance assigned to this speaker with a probability above a certain threshold exceeds a certain ratio. At the end of the speech extraction process, a data set is created that shows the utterances made by each speaker in each discussion round.

Player Role Identification In order to analyze player behavior, it is important to know the objectives pursued by the players. As these vary in AmongUs for each lobby, depending on the assigned role, this role must be extracted for each streamer for each lobby. This is another step in the pipeline. The chosen method for role extraction is image recognition. This is suitable for AmongUs, as the roles are always assigned in a splash screen at the start of each lobby. This splash screen is a black screen with the role assignment at the top. This makes it very easy to extract the text and therefore the role using OCR. I extract images for each lobby around the identified lobby start time, then I pre-process these before using OCR to try to detect an assigned role. I do this for all extracted images of each lobby to then determine the most likely role for the lobby by making a majority vote on all roles determined in this way.

Analysis of Extracted Data Set By combining the extracted text data and the extracted game roles, the final result is a data set that lists all the utterances made by each streamer for each discussion round in which they took part, as well as the game role they had at that moment. I analyze this data set in the final step of the pipeline. Among other things, I examine whether it is possible to train a model that can classify which game role a streamer has based on the utterances he has made. For this purpose, I fine-tune a pre-trained BERT model for text classification using the created data set. Due to the complex and sometimes very different objective functions of roles from the Neutral main role, these are excluded from the classification. The model trained in this way can prove in the evaluation that it achieves a performance in the role classification that significantly exceeds random guessing. A classification at the discussion level, in which a majority

vote is made using the role classification of all individual utterances made by a streamer in a discussion round, achieves an even better performance. Due to the good model performance, I use the Explainable AI method Integrated Gradient in a further step to gain insights regarding feature importance of the model. In addition to training and evaluating the classification model, I also apply descriptive methods and text mining to the data. I carry out a sentiment analysis, a topic analysis, a word frequency analysis and an analysis of the part of speech used - also differentiated by game role. The results obtained are presented in the Chapter 6.

5.1 Temporal Synchronization of VODs by Lobby Extraction

5.1.1 Methodology

Creation of Initial Proposals for Lobby Times on VOD Level In the initial stages of the extraction pipeline, I identify and extract individual lobbies from the VODs for each streamer. This step is essential as it assigns streamers to the lobbies they participated in and isolates the segments of the VODs where the game is actively played, as only these parts are relevant for subsequent analyzes. Moreover, extracting these lobbies enables the temporal synchronization of VODs from different streamers within the same session. The exact procedure is described in the following: Initial preprocessing involves using image recognition techniques to identify key events in the VODs, specifically the start and end of game lobbies. This is achieved by extracting frames at a rate of three frames per second, preparing them accordingly (identification of dark frames, zooming to the trigger locations, application of binary thresholding and morphological operations), and then applying OCR to detect specific text patterns indicating the lobby events lobby start and lobby end. The identified lobby events are recorded in srt files, which contain timestamps and descriptions of each detected event. After applying some heuristics to filter out the most realistic timestamps, using them I create initial proposals for individual lobbies by forming arrays with potential lobby start and end times for each streamers VOD separately. I filter these arrays - called lobby times - to remove for example unrealistic durations, such as lobby times lasting less than a minute ensuring only plausible lobbies are considered. As each VOD is associated with metadata such as its publication date, I can use this publication date to convert the timestamps of the identified lobby times to UTC to facilitate the comparison and synchronization across different streamers' VODs.

Creation of Proposals for Session-Wide Lobbies Coming from these lobby times of the individual streamers of the session, the next step involves finding proposals for session-wide lobbies. I do this by sorting all identified lobby times of a session chronologically and then creating session wide lobbies by grouping all the lobby times with a start time within two minutes of each other. I calculate median start and end times for each of the created session-wide lobby to assign lobby times containing None values (some lobby times have no start or end timestamp as the image recognition does not detect it correctly). Subsequently, I refine the in the described way created session-wide lobby candidates further by merging lobbies with similar end times and correcting temporal inconsistencies between the lobbies (in some cases it can happen that lobby

n has a later lobby end time than lobby $n + 1$ due to the lobby creation process described earlier). Through the assignment of individual streamers lobby times to session-wide lobbies I also extract for each streamer in which lobby he participates. To enhance the assignment accuracy of streamers to lobbies, I apply further heuristics. This involves removing unassigned lobby times of streamers that fall between two lobby times assigned to consecutive lobbies and merging two lobby times assigned to the same lobby if one lacks a start or end time. These steps ensure a more reliable mapping of streamers to lobbies.

Manual Examination of Streamers and Lobbies With Missing Information Despite already achieving meaningful results after the extensive programmatic lobby extraction and synchronization, manual examination is necessary as some information is still missing for certain streamers and lobbies to make a final assignment. As preparation for the manual examination, for each session all the trustworthy lobby times are defined. I consider a lobby time trustworthy if at least one other lobby time is assigned to the same lobby and has a similar start and end time (maximum 10 seconds apart) and a similar duration (maximum 5 seconds difference). Log files are generated for each session, stating streamers with non-consecutive lobby assignments, streamers without at least one trustworthy lobby times, and lobbies without at least one trustworthy lobby time. For these streamers and lobbies I manually review the respective VODs to record for each streamer with non-consecutive lobby assignments if they indeed skipped a lobby. Also, for each streamer and lobby without a trustworthy lobby time I record such a trustworthy lobby time. For the manual examination, it is necessary to assign player names to specific streamers. I have created a non-exhaustive list to the best of my knowledge for this assignment, which is attached in the Appendix A.2. This list is also used in further steps of the pipeline whenever necessary.

Merging Programmatic and Manual Examination The final step involves merging the results from the manual examination with the programmatic assignments. This includes assigning trustworthy lobby times to all streamers and lobbies without such a trustworthy lobby time. Additionally, I finalize the assignment of streamers to lobbies they participated in: Streamers are assumed to have participated in all lobbies between their first and last assigned lobbies unless I explicitly identified they have skipped some lobbies in between.

Calculation of a Dictionary Storing Temporal Relationships Between Streamers At this point, you know which lobby each streamer has participated in. In addition, at least one trustworthy lobby time is known for each streamer and for each lobby. If the time offset between the start and end times of the different streamers is known, a start and end time can be calculated for each streamer for all lobbies in which they have participated. I do this by taking the known trustworthy lobby time for the lobby in question and calculating the start and end time, taking into account the time offset between the streamer and the streamer belonging to the trustworthy lobby time. How such a dictionary, which represents the time differences between the streamers, is calculated, is explained in the following. In a first step, I store all known temporal relationships between two streamers of a session, with known temporal relationships being determined based on the trustworthy lobbies of the streamers. In a next step, based on these temporal relationships, I can create in all sessions a network of temporal relationships in which a temporal relationship can be derived between all streamer pairs of the session: either directly based on a known temporal

relationship to each other or indirectly via a chain of known temporal relationships to intermediate streamers that connect the two streamers with each other. However, this type of network can only be created if the known temporal relationships are completely consistent. Due to possible inconsistencies in the image recognition or in the timestamps of the stream, this is not given in the present case. Thus an attempt must be made to create a network of temporal relationships that is the most likely possible, taking into account the known temporal relationships. In order to achieve this, I treat the problem at hand as a linear system of equations of the form

$$Ax = b. \tag{5.1}$$

The matrix $A \in \mathbb{R}^{n \times s}$, with n being the number of known temporal relations, s being the number of streamers in the session, consists exclusively of the values $\{1, -1, 0\}$ and acts as a selection matrix: it represents a known temporal relation in each of its rows by selecting the two affected streamers by the values $\{-1, 1\}$, while the remaining streamers are ignored by $\{0\}$. The vector $x \in \mathbb{R}^s$ contains all streamers, with the streamers involved in the known temporal relationship being selected by multiplication with the selection matrix. The vector $b \in \mathbb{R}^n$ represents the temporal differences between the two selected streamers of the known temporal relationship. In this way, all known temporal relationships are displayed. In the next step, I calculate the most likely temporal relationships between the streamers on this basis. I calculate the least square solution to solve the system of equations $Ax = b$, by deriving a vector x that minimizes the 2-norm $|b - Ax|$. By subtracting the timestamps obtained for the streamers by the optimization from the vector x, I can now calculate the most likely temporal relationship to each other for each pair of streamers.

Calculation of Final Lobby Times for Each Streamer After creating the streamer distance dictionary, I can now calulate the final lobby times for each streamer as mentioned before. In addition to the streamer distance dictionary, the information about which lobbies the streamers have participated in and the trustworthy times of the individual lobbies are used as mentioned: I iterate through all lobbies of the session and calculate the lobby time for each streamer who has participated in the lobby by adding the time difference between the streamer and the streamer from whom the trustworthy lobby time originates to the trustworthy lobby time of the lobby. At this point it should also be noted that for eleven sessions the programmatic lobby extraction can hardly produce any usable results which is why they are excluded from the further analysis. Mostly the reason of the failure of the programmatic extraction is that for a majority of the streamers the srt file containing the results from the image classification is missing and thus no trustworthy lobbies can be build.

5.1.2 Evaluation

The final lobby times assigned to the streamers are evaluated. This involves randomly selecting 10 lobbies from all sessions and streamers and manually checking their correctness. The results, which are shown in Table 5.1, indicate a successful extraction, as nine of the ten selected lobby times actually belong to the correct lobby number. In one case, a lobby was not recognized in the

Streamer	Session	Lobby	Start	End	Correct Start	Correct End
jvckk	2022-03-01_S1	12	01:57:08	02:01:45	01:57:07	02:01:45
aribunnie	2022-02-24_S1	6	00:50:21	00:54:40	00:50:23	00:54:40
karacorvus	2022-01-25_S1	15	02:42:02	02:47:13	02:33:19	02:41:20
kaywordley	2022-02-22_S1	11	01:31:55	01:39:41	01:31:55	01:39:41
reenyy	2022-03-10_S1	10	01:42:33	01:52:15	01:42:34	01:52:15
vikramafc	2022-01-27_S1	13	01:47:05	02:02:03	01:47:02	02:02:04
skadj	2022-03-03_S1	8	01:07:28	01:14:07	01:07:30	01:14:07
irepptar	2022-01-26_S1	6	00:55:52	01:02:45	00:55:50	01:02:45
vikramafc	2022-01-27_S1	4	00:26:10	00:33:50	00:26:13	00:33:50
vikramafc	2022-03-09_S1	3	00:23:50	00:29:14	00:23:50	00:29:14

Table 5.1: Evaluation Results of Lobby Extraction.

extraction process, which is why the times of lobby 15 actually correspond to lobby 16. When evaluating the nine successfully assigned lobby times, the identified start point of the lobby is on average 1.5 seconds away from the actual start point, while the identified end is on average even only 0.1 seconds away.

5.2 Extraction of Relevant Text Data

To be able to analyze the audio data from the conversations in the streamers' discussion rounds using text mining, it must first be converted into text data. This involves three tasks: First, it must be identified which points in time of the audio stream are relevant, i.e., in which time periods the discussion rounds of the respective lobbies are, as only these conversations are to be analyzed. In the next step, the audio data from the relevant time periods must be transcribed with an error rate that is as small as possible. In the final step - speaker diarization - the transcribed text must then be assigned to the various speakers in the conversation. The three individual steps are described below.

5.2.1 Identification of Discussion Rounds

Methodology

AmongUs Game Dynamics as Foundation for Identification of Discussion Rounds To identify the discussion rounds within a lobby, I can use the fact that the streamers in their VODs only activate the audio of the other streamers involved in the session during the discussion rounds. In the movement phase, on the other hand, only the streamer can be heard commenting on their gameplay or communicating with their Twitch chat. In order to identify the times of the discussion rounds, it is therefore necessary to find the phases in which several speakers can be heard. In three of the thirty sessions analyzed, the streamers also talk to each other during the movement phase. As it is therefore not possible to extract the discussion rounds for these sessions using the methodology described below, I must exclude them from the analysis[2].

[2] The following sessions are excluded due to streamers communicating with each other in the movement phase: 2022-01-25_S1, 2022-05-19_S1, 2022-05-23_S1.

Using Pyannote Speaker Diarization Model for Individual Streamer's Discussion Round Proposals To find the phases within a lobby with multiple speakers, I use the Speaker Diarization Model 3.1 from the open-source speech extraction tool *Pyannote*. This popular model finds all segments in the analyzed audio file in which different speakers are speaking. For each identified segment, there is a start and end time as well as a speaker ID assigned by the model. After applying the model to the audio file of an entire game lobby, I can use these segments to create initial suggestions for discussion rounds: This involves iterating over the segments sorted chronologically in ascending order and checking in each case whether the last three segments have different speakers. If this is the case, I check whether the start time of the first segment (and speaker) is less than 10 seconds away from the start time of the third segment (and speaker). If this is also the case, a candidate for a discussion round is found, with the start time of the first segment and the end time of the last segment. In a further step, I iterate over all candidates and always combine successive candidates for discussion rounds if their start times are less than 30 seconds apart. Then I check the resulting candidates for discussion rounds again based on their duration. Since a discussion round in the games analyzed has a maximum duration of 90 or 135 seconds depending on the session and is only rarely shortened by all players voting early, I consider a duration of less than 60 seconds for a discussion round unrealistic. For this reason, I check all candidates for discussion rounds that are shorter than 60 seconds once again to see whether they can be connected. If there is less than 60 seconds between the end of such a candidate for discussion rounds and the start of the following candidate for discussion rounds, I connect them. In this way, I find suggestions for discussion rounds for each streamer in the lobby.

Creation of Lobby-Wide Discussion Round Proposals The next step is to use these suggestions from the individual streamers of the lobby to find the most likely discussion rounds of the lobby. First, I collect all start times and all end times of the discussion round candidates of the individual streamers in the lobby separately. Then I form so-called start and end clusters: For this purpose, I iterate through the chronologically ascending start respectively end times, and always assign them to a cluster if at least one other start respectively end time in that cluster is less than 15 seconds away from the considered start respectively end time. If no cluster is found to which I can assign a time, the time opens a new cluster. In order to identify the most likely correct clusters, I filter the start and end clusters created in this way in the next step: Only those clusters whose number of start or end times is greater than half the number of streamers in the lobby are considered further. All "likely start and end times for the discussion rounds" are now derived from these "likely clusters". For this purpose, I extract the 25% quantile of times from the likely start clusters and the 75% quantile of times from the likely end clusters. The likely start and end times obtained in this way I then merged again in the next step: I sort the likely times chronologically in ascending order, then I iterate through the times and build arrays of the form [likely start time, likely end time]. If two start and two end times follow each other, I add a None value. In a final step of the programmatic extraction, I make an attempt to replace the remaining None values with realistic times. To do this, I calculate a likely value for each None value based on the known corresponding start or end time by adding or subtracting the maximum duration of the discussion rounds. As already described, the maximum duration is exploited by the players in most discussion rounds. The maximum discussion round duration varies between 108 and 152 seconds depending on the session and is determined manually for each session in a previous

step. The most likely value for the None value calculated in this way I then check for plausibility: If the calculated value for a start time of a discussion round is earlier than the end time of the previous discussion round, or if the calculated value for an end time of a discussion round is later than the start time of the subsequent discussion round, I discard the proposal. For all calculated values that have passed this exclusion criterion, I check whether there is a cluster with at least three assigned start or end times among the originally identified clusters for start or end times (not the likely clusters), which has at least one start or end time that is less than 5 seconds away from the calculated value. If this is the case, I assume the calculated value to be plausible and replace the None value.

Manual Examination of Remaining Discussion Rounds With None Values At this point, a good identification of discussion rounds and their likely start and end times is already achieved. However, since 174 of the 897 lobbies analyzed still have at least one discussion round with a None value, I carry out a manual examination of these lobbies in the next step. I extract the correct start and end time for each discussion round in these lobbies. I then combine these lobbies with the extracted times with the remaining lobbies and their identified discussion times. In this way, I identify all discussion rounds contained in all lobbies with their respective start and end times.

Evaluation

Streamer	Session	Lobby	DR	Identified Start	Identified End	Correct Start	Correct End
ozzaworld	2022-01-27_S1	3	1	00:20:21	00:22:53	00:20:20	00:22:53
jvckk	2022-05-24_S2	6	3	00:55:16	00:56:19	00:55:16	00:56:19
ozzaworld	2022-02-09_S1	7	1	01:11:20	01:12:51	01:11:22	01:12:56
aribunnie	2022-02-17_S1	13	5	02:24:27	02:26:12	02:24:24	02:26:10
karacorvus	2022-02-23_S1	15	2	02:17:26	02:18:04	02:17:27	02:18:04
jvckk	2022-01-26_S1	12	2	01:54:06	01:55:52	01:54:06	01:55:54
zeroyalviking	2022-02-16_S1	15	3	02:13:39	02:14:46	02:13:38	02:14:39
jvckk	2022-02-23_S1	6	1	00:43:08	00:45:40	00:43:07	00:45:39
uneasypeasy	2022-01-26_S1	6	2	00:58:25	01:00:13	00:58:25	01:00:13
tenmamaemi	2022-02-02_S1	17	2	02:44:30	02:47:44	02:44:32	02:46:33

Table 5.2: Evaluation Results of Discussion Rounds Identification.

The results of the evaluation of the final extracted discussion rounds are displayed in Table 5.2. For this, I select ten random discussion rounds from all discussion rounds and check for the identified start and end times to see how far they are from the actual correct times. All identified discussion rounds do indeed exist, and the start and end times are also very close to the actual time. The only exception is discussion round 2 from lobby 17 of session 2022-02-02_S1: Here, the discussion round is actually correctly identified as well, but discussion round 2 actually ends a little earlier. Immediately after the end of this discussion round 2 (1 second later), however, discussion round 3 starts, as another meeting is convened immediately. This also explains why the designed procedure does not detect an end of discussion round 2 and a subsequent start of discussion round 3, as it connected the two. However, since both discussion rounds are still included in the final discussion rounds (as one long discussion round), the error is actually negligible.

For the remaining 9 discussion rounds, it can be seen that the extraction process is very successful: The identified start time is on average 1.11 second away from the correct start time of the discussion round, the identified end time on average 1.88 seconds.

5.2.2 Transcription of Discussion Rounds

Methodology

Extraction of Relevant Audio Sections for Transcription I use the pre-trained Whisper model from OpenAI to transcribe the identified discussion rounds. I use the model in the size large. For this purpose, I first extract the relevant time period as an audio file from the corresponding VOD for each discussion round. Two comments should be made about this extraction: Firstly, the start time of the extracted section is always defined as 2 seconds before the identified discussion round start, and the end time of the extracted section is always defined as 2 seconds after the identified discussion round end. This is only not done if the new start or end time calculated in this way is before or after the start or end time of the corresponding lobby. The decision for the small extension of the discussion period is motivated by the intention not to miss any utterances made at the start and end of the discussion round. Secondly, I use the open source tool *Ffmpeg* for the extraction. As the version used does not re-encode the extracted sub clip, the extraction is not accurate at frame level. As a result, there may be marginal differences in length between two extracted sections of the same discussion round from different VODs (from different streamers). This is negligible for the use case here. However, if, for example, synchronization of several transcriptions of the same discussion round is desired in other use cases, a variant of the extraction tool must be used that enables exact extraction at frame level by re-encoding the extracted section, which, however, takes considerably longer.

Transcription of Extracted Audio Sections After extraction, I transcribe the relevant audio file. The resulting transcribed text is divided into segments by the model. These segments contain one or more words that are recognized by the model as a coherent utterance. The chosen Python implementation is the *stable-ts* package with the class *stable-whisper* [3]. This implementation is built based on the OpenAI Python implementation of the whisper model [4], but modifies it to extend its functionality and produce more reliable timestamps. In particular, it allows not only the start and end times to be extracted at the segment level, but also the times of the individual words of the segments. This enables an even more accurate temporal placement of the transcribed words. I save the results of the transcription in srt file format.

It should be noted that I carry out the transcription of a discussion round not only once, but for each streamer involved in the discussion round. This is due to the fact that the resulting transcriptions differ slightly depending on the VOD used. The reason for this is probably that the audio quality and volume of the individual streamers differs depending on the VOD in consideration: the streamer whose VOD is analyzed is usually the loudest and best to understand, which is why the transcription preferentially includes the utterances of this main streamer, especially in conversation phases with several simultaneous, overlapping speakers. To ensure that no utter-

[3] https://github.com/jianfch/stable-ts?tab=readme-ov-file#usage Python package stable-ts.
[4] https://github.com/openai/whisper Python Implementation of the OpenAI Whisper model.

ances are lost, I carry out a transcription of the discussion round for each streamer involved in the discussion round using his respective VOD.

Evaluation

Session	Lobby	Discussion Round	Utterance	Exists	All words correct
2022-02-08_S1	8	2	Five seconds ago, Kay, someone attacked you.	1	1
2022-02-09_S1	14	3	I just can't care that there's more Futurama coming because it's like been cancelled and revived so many times. It's a fucking walking zombie at this point. I don't know.	1	0
2022-02-24_S1	15	1	Nice.	1	1
2022-02-16_S1	8	4	Alright,	1	1
2022-05-24_S2	4	0	hey	1	1
2022-02-01_S1	12	0	yeah who was either oh then it wasn't ari then it was just casper wait who was i there with	1	1
2022-01-27_S1	17	0	i	1	1
2022-02-15_S1	10	0	that makes so much more sense.	1	1
2022-02-12_S1	16	0	Yo, okay. We're not going to hide.	1	1
2022-02-15_S1	10	4	you got the kill!	1	1
2022-02-04_S1	3	3	the hint that he gave me is that his role was just neutral. know, looking at all the neutral roles, I figure he's most likely Arsonist. If he were killer,	1	0
2022-03-03_S1	6	0	I feel sorry for his boss, Rupana.	1	1
2022-02-21_S1	18	0	But chat, chat, we're eight away from hitting 90 subs in a day. That would be insane for me coming back to my, If anyone hasn't already hit the sub button, I really would appreciate it. If we can hit 90 for the day, that would be actually amazing.	1	0
2022-03-03_S1	13	2	I'm wearing my boba shirt from Kale and I'm drinking boba.	1	1
2022-02-15_S1	13	1	Where is that?	1	1
2022-03-01_S1	1	3	I did one kill.	1	1
2022-03-10_S1	11	2	you know what? That proves	1	1
2022-03-09_S1	12	2	Do I find dripstone? Do I mine it? I think I find it, right?	1	1
2022-05-24_S2	3	1	We're one away from the big 10. There was a...	1	1
2022-02-26_S1	7	1	Okay, we'll fix Skog.	1	0
2022-02-08_S1	3	0	From now. You tell him.	1	1
2022-03-03_S1	11	1	what do you think? You still voted me?	1	1
2022-02-08_S1	10	1	sorry guys sorry guys i'm a bit of a thing	1	1
2022-02-23_S1	15	0	That's awesome.	1	1
2022-02-01_S1	7	0	Literally	1	1
2022-05-24_S1	13	1	I actually, I'm going to shield Jay.	1	1
2022-02-22_S1	13	2	Hi. Welcome. Oh my lord.	1	1
2022-02-01_S1	3	4	See you later, man. Can someone explain?	1	1
2022-02-15_S1	12	0	this happened,	1	1
2022-02-08_S1	9	2	you could have won this whole thing.	1	1

Table 5.3: Evaluation Results of Transcribed Utterances.

The evaluation of the transcription displayed in Table 5.3 shows a very good performance of the transcription model. For thirty randomly selected utterances, I analyze whether they are actually made in the corresponding discussion round. All of these thirty statements are actually made, sometimes with small differences in individual words. Naturally, no statement can be made about the utterances not captured by the model. However, it can generally be said that this number was reduced as much as possible by transcribing a discussion rounds not just once, but for all participating streamers.

5.2.3 Speaker Diarization of Discussion Rounds

I train a PVAD model for speaker diarization. This is a model for target speaker extraction, i.e. it attempts to identify the frames of a given audio sequence in which a target speaker speaks. In addition to the audio sequence being classified, the model takes as input an embedding that represents the typical audio characteristics of the target speaker. The model outputs a classification for all frames of the input sequence into the three classes Target Speaker Speech, Non-Target Speaker Speech, and No Speech. In order to apply the described model in the present use case, I must carry out a few steps, which are shown in Figure 5.2: First, I generate artificial training data for

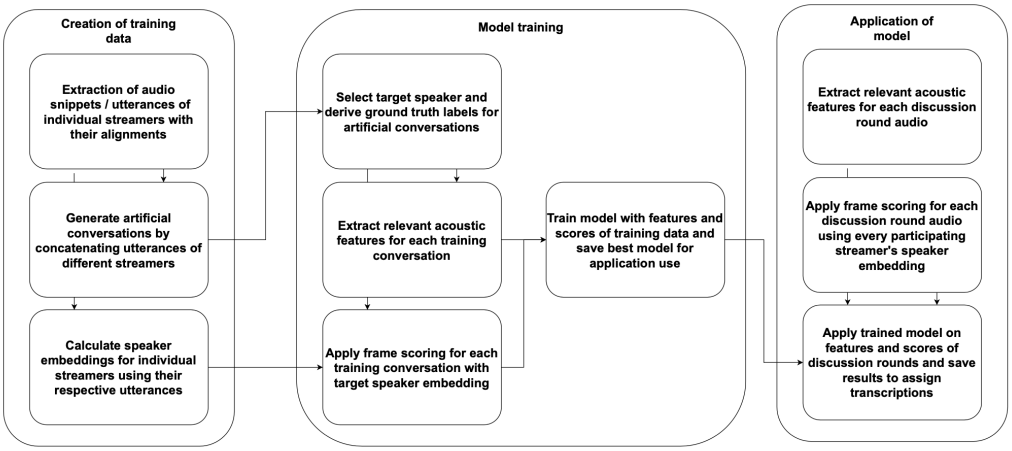


Figure 5.2: Speaker Diarization Pipeline.

the model from the available VODs. In the next step, I train the model. In a final step, I apply the trained model to the audio sequences of the discussion rounds to create a speaker diarization of the transcribed conversations. The individual steps of this speaker diarization pipeline are described in the following. For the implementation, I am mainly following Sedláček [2021]. In his thesis, he reproduces the code from Ding et al. [2020]'s original PVAD paper. He has made his code available in a public repository[5]. Despite having to make quite some adjustments to the code for my use case, his work laid an important foundation.

Creation of Training Data

Idea of Extracting Auido Snippets to Generate Training Data To perform classifications with a PVAD model, the input required is both the audio sequence to be classified and an embedding of the voice characteristics of the target speaker. Both are therefore required for model training: Firstly, conversations with several speakers that come as close as possible to the audio setting of the discussion rounds, and for which labels exist that indicate in which frame which speaker is speaking. And secondly, an embedding with the voice characteristics for all streamers involved in the discussion rounds. In order to generate both required training inputs, I extract audio snippets from the VODs of the individual streamers, in which only the individual streamers speak. In a next step, I concatenate these audio snippets with each other to artificially generate conversations that are very similar to the setting of the discussion rounds in terms of audio quality and for which it is known at which time which of the speakers is speaking. In addition, I can combine several audio snippets of a streamer in order to calculate an embedding of the voice characteristics of the respective streamer. In the following, I will therefore first describe how these individual audio snippets are extracted from the VODs of the individual streamers, before going

[5] https://github.com/pirxus/personalVAD/tree/master Unofficial implementation of the PVAD method

on to explain how I calculate both the artificial concatenated conversations and the embeddings of the individual streamers based on them.

Filtering of Considered Movement Phases As mentioned earlier, the analyzed game is divided into two phases: The discussion phase, in which the players are discussing with one another, and the movement phase, in which the players complete tasks while commenting on their game decisions or talking to their own chat. If there is speech during the movement phases, it can be assumed that the speaker is the streamer of the VOD. The times of the movement phases must therefore be identified so that utterances made within these phases can subsequently be extracted as audio snippets. The times of the movement phases are already known, as the times of the discussion phases have already been identified in a previous step. All remaining time periods within a lobby are movement phases. As players who have already died are allowed to talk to other players who have also already died during the movement phases, it is important to avoid including such movement phases of players who have died. For this reason, I only take the first two movement phases into account for each lobby, as the probability that the player has already died is too high in later movement phases. In addition, I only consider movement phases that are at least 60 seconds long. This is due to the fact that I cut off a buffer of 15 seconds at the beginning and end of the identified time period of the movement phase in order to exclude the possibility that slight errors in the identified start and end times would lead to parts of a discussion phase being extracted. By filtering for longer movement phases, the latter are still long enough to extract relevant audio snippets even after cutting off the 30-second safety buffer. I extract all movement phases that fulfill the criteria described as audio sequences for all participating streamers.

Extraction of Individual Utterances of Streamers Individual utterances of the streamers must now be extracted from these audio sequences. Once again, I use *Pyannote*'s Speaker Diarization model for this. This model has two key features that recommend its use: Firstly, it is very good at recognizing the sections of each audio sequence in which people speak. Secondly, it assigns a speaker ID to each of these identified utterances. Since the audio sequences of the movement phases are a setting with few speakers (ideally only 1 speaker) and few overlapping conversations, the assignment of the individual utterances to speaker IDs achieved in this way also has very good accuracy. I can thus use this speaker assignment to check the identified audio snippets once again to ensure that only the streamer of the VOD is speaking. Therefore I apply the *Pyannote* Speaker Diarization model to the audio sequences to check whether there is more than one speaker in the extracted section of the movement phase. If this is the case, I no longer consider the movement phase. If this is not the case, I extract all utterances identified by the model as audio snippets. At this point, for each streamer, I now have a collection of audio snippets that can be used to create artificial concatenated conversations or to calculate the speech characteristics of the individual streamer. As can be seen in Table 5.4, for each streamer there exist different numbers of audio snippets, depending on how many VODs of the respective streamer are available and how fruitful these VODs are in terms of audio snippets. In order to guarantee sufficient variation in the artificial conversations created and in the calculation of the voice characteristics of the individual streamers, I consider only streamers for which at least 100 audio snippets can be extracted for further analysis. The number of streamers analyzed is thus reduced to 12 streamers. It is worth mentioning that this decision is made because this work is a proof of concept. The choice of the

threshold of 100 utterances is arbitrary and, if necessary, the model would probably have been able to achieve successful results for a few more streamers with fewer audio snippets as well.

Streamer	Audio Snippets	Streamer	Audio Snippets
aribunnie	997	pwuppygf	45
ozzaworld	497	willyutv	43
jvckk	466	itsdanpizza	42
zeroyalviking	397	kaywordley	41
courtilly	379	junkyard129	35
irepptar	375	falcone	27
uneasypeasy	324	heckmuffins	26
karacorvus	257	x33n	22
skadj	168	reenyy	21
vikramafc	140	cheesybluenips	18
ayanehylo	129	hcjustin	17
jayfletcher88	103	chey	12
pastaroniravioli	93	pjonk	11
dooleynotedgaming	90	brizzynewindsong	7
therealshab	77	tenmamaemi	0
br00d	68	sidearms4reason	0
thecasperuk	64	kyr	0
taydertot	63	kruzadar	0
chilledchaos	59	dumbdog	0
jojosolos	57	atla5	0
paulleewhirl	55	aplatypuss	0
ressnie	47		

Table 5.4: Number of Extracted Audio Snippets for Each Streamer.

Creation of Labeled Artificial Conversations The next step is to generate labeled artificial conversations from the extracted audio snippets. To create an artificial conversation, I combine a random number of between 50 - 70 audio snippets. The reason for choosing 50 - 70 audio snippets is that, due to the average length of the audio snippets, this number leads to a length of the concatenated conversations that is similar to the length of the discussion rounds that will be analyzed later. In the concatenation process, I choose the first audio snippet at random by selecting a random streamer in a first step and then selecting a random audio snippet from this streamer in a next step. I repeat this process to randomly select the next audio snippet, which is then concatenated to the previous audio snippet. I repeat this until the previously determined random number of between 50 - 70 audio snippets is reached. I place back drawn audio snippets, i.e. they can occur several times within a concatenated conversation. In this way, I generate a total of 100,000 concatenated conversations, which are later used as training data. Technically, the audio snippets are concatenated by concatenating the waveform data of the individual audios in an array. Thus, to successfully generate a concatenated audio in this way, the audio snippets

must first be pre-processed: I need to convert the extracted audio snippets need from stereo to mono audio, as the PVAD model, like many other speech processing models, is designed to work with mono audio. This is because mono audio has only one channel instead of two, which reduces computational complexity. For the conversion, the right and left channels are averaged. In addition, I resample the audio snippets to a standardized sampling rate of 16,000. This sampling rate is also widely used in speech processing tasks as it maintains the balance between quality and computational complexity. In a final step, I trim the end of the audio snippets so that the length is divisible by 10ms, which is the length of the frames for which the PVAD Model performs its classifications. This ensures consistency in the alignments of the audio snippets, which will be discussed below.

For the resulting concatenated audio file, a label has to be saved for each frame that defines whether in this frame there is speech, and if so, by which speaker. To be able to do this, this information first has to be known for the individual audio snippets, which is why I first create a text alignment file for each of the individual audio snippets. Such an alignment file contains information about the spoken words in an audio snippet and their corresponding time stamps. It has a specific format from which the start and end times can be derived for each word. In this way, the periods of the audio snippet in which there is no speech can be extracted. I create the required alignment files by transcribing the audio snippets in a first step using the OpenAI Whisper model. I then write this transcription to the alignment file together with the timestamps of the individual transcribed words in the corresponding format. I also add the phases of silence with the corresponding time stamps. In this way, I create a large alignment file for each streamer, which contains the alignments of all its audio snippets. If I then concatenate several audio snippets together, I also concatenate their alignments in order to create an alignment for the concatenated conversation. This allows me not only to record which frames are spoken in and which are not, but also which of the speakers is speaking in the frames in which it is spoken.

To store the concatenated conversations created and other information relevant to the training (which will be discussed later), I use the *Kaldi* system [Povey et al., 2011]. *Kaldi* utilizes pairs of scp and ark files to efficiently store and describe data. The scp files usually hold information about how to obtain a particular resource, which is identified by a key (for example the utterance / audio snippet id). Each key is then associated with a recipe, which describes how the resource can be obtained. This could for example be the path to the source file, or an address referring to a specific position in an ark file. The ark files are essentially archive files designed for efficient data storing, typically used to store extracted features.

At this point, it should be mentioned that the artificially generated conversations do not have any overlapping conversations due to the selected method of concatenation. However, since the discussion rounds to be analyzed later with the model do have this, the performance of the model could probably have been improved even further by including sequences with overlapping conversations in the training data. However, as the performance is good enough as it is, this is not done in this case. The same applies to the possibility of adding music or other sounds typical for gaming streams to the artificial conversations created in the background, which could possibly have led to a slight improvement in model performance.

Creation of Speaker Embeddings for Streamers Using the Audio Snippets Now that concatenated conversations are created from the audio snippets, I also use these to create

embeddings for the individual streamers, which depict their voice characteristics. I create these embeddings in the present work using *d*-vectors as already described. For the purpose of this work, I modify the actual model class from the used *Resemblyzer3* implementation to support frame-level embedding extraction in addition to the default method, which extracts one *d*-vector for a sliding window of 160 frames. The extracted *d*-vectors have a dimensionality of 256. In total, I calculate *d*-vectors for all 12 relevant streamers. The basis for this are 100 randomly selected audio snippets of the corresponding streamer. After successful extraction, I save the *d*-vectors for each speaker in *Kaldi* format.

Training of the PVAD Model

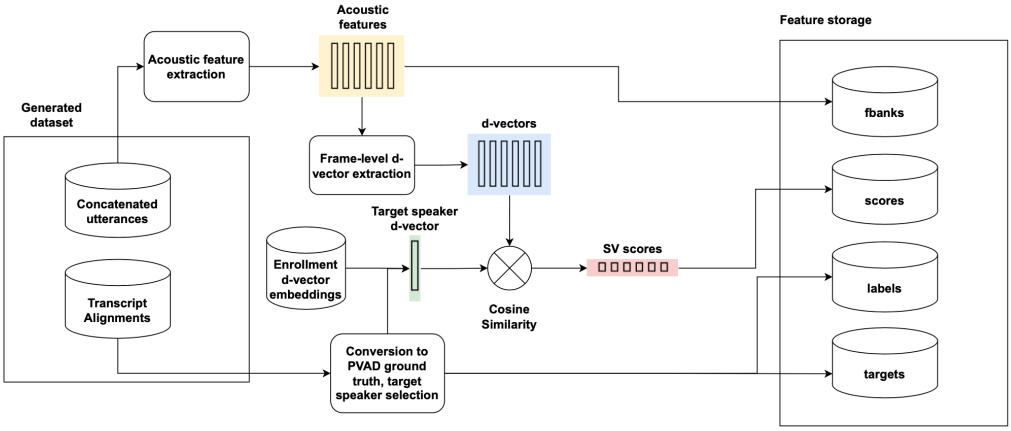


Figure 5.3: Feature Extraction Pipeline.

In the next step, I use the generated conversations and the speaker embeddings of the streamers to generate the input for the model training. For each conversation generated from the training data, I load the waveform, extract the acoustic features, select a target speaker and generate ground truth labels for the entire conversation. Then I carry out frame scoring using the corresponding target speaker embedding. The relevant features, scores, labels and target speakers are then saved in the *Kaldi* format. The individual steps, which are shown in the pipeline in Figure 5.3[6], are briefly described below before the training process is described.

Extraction of Acoustic Features and Target Speaker Selection for Generated Conversations In a first step, I randomly select one of the speakers involved as the target speaker for each artificially generated conversation. Then I extract the acoustic features by first normalizing the audio file of the respective conversation and then calculating the Mel filterbank features for it. The open-source Python package *librosa.feature.melspectogram* is used for this step.

[6] For this Figure, inspiration is taken from Sedláček [2021].

The features generated in this way represent the detailed acoustic features of each individual frame of the concatenated conversation and are later used directly as model input.

Calculation of Target Speaker Similarity Scores for Generated Conversations I also use these features to calculate *d*-vectors at frame level based on them. To additionally calculate *d*-vectors in a windowed manner, I also extract windows of these filter bank features. These window-level *d*-vectors provide embeddings that capture the speaker characteristics over short segments of the utterance. I extract them in addition to the frame-level embeddings because they can provide a continuous and consistent representation of the similarity between the target speaker and the individual segments of the utterance. This allows the model to later make decisions based on the entire duration of the conversation rather than just certain short segments. I then use the calculated *d*-vector embeddings, at frame and window level, to calculate similarity scores with the embedding of the target speaker using cosine similarity. The resulting similarity scores of the different *d*-vector embeddings (frame-level and window-level) I then combine in an array so that they can be used as further model input for classifying the conversation under consideration: The frame-level scores are taken directly, while the window-level scores are stretched over the entire length of the conversation using both the Kronecker product and linear interpolation. With the Kronecker product, each score value is duplicated for a fixed number of frames. With linear interpolation, which also has a fixed number of interpolated frames, the scores are evenly distributed between two points to enable a smooth transition between the scores. The similarity scores obtained in this way I later use as model input alongside the acoustic features of the audio file.

Generation of Labels for Generated Conversations To create the ground truth labels, I use the alignment files of the individual audio snippets of the concatenated conversation. Using these, it is possible to determine the times when there is no speech (label No Speech) and when there is speech. For the times of speech, I then checks whether the speaker is the target speaker (label Target Speaker Speech) or another speaker (label Non-Target Speaker Speech). I save all information required for training (features, scores, labels, target speakers) in *Kaldi* format.

Training of PVAD Using Created Training Input I use the SET variant of the PVAD Model as the model architecture, as experience has shown that it has the best performance. Thus, the input is a 297-dimensional feature vector combining the 40-dimensional log Mel-filterbank energies, the 256-dimensional target speaker *d*-vector representation and the array of speaker verification scores for each frame. I create a total of 100,000 artificial conversations as training data. In addition, I create 10,000 further artificial conversations using the same methodology, which I use as test data to determine how many epochs the model is trained. The final PVAD model consists of a 2-layer LSTM network of 64 cells each, followed by a hidden layer of 64 neurons. The activation function I assign to this hidden layer is the hyperbolic tangent activation function. In this configuration, the model only has 130 thousand parameters. During training, I use the Adam optimizer (Kingma and Ba [2017]) with a variable learning rate set to 1×10^{-3} for the first epoch, progressing down to 1×10^{-5} using learning rate scheduling. The batch size I use is 1 (note that this unusual batch size is chosen since the model for larger batch sizes always ended up predicting

just one of the three classes, probably reasoned in an issue with padding of the different-sized utterances in the training batch). I train the model for 10 epochs, but after each epoch I evaluate the trained model on the test data. As one can see from Table 5.5 displaying the overall loss as well as the precision, accuracy, and recall for the Target Speaker Speech label 2 for the model after epoch 6 and epoch 10, the model's performance did not increase significantly anymore after the sixth epoch. Thus, to avoid (further) overfitting, I use the state of the model after 6 training epochs as final model.

Train epoch	Overall Loss	Accuracy (L2)	Recall (L2)	Precision (L2)
6	0.1173	0.9911	0.9075	0.9539
10	0.1077	0.9919	0.9142	0.9598

Table 5.5: Train Logs for Different Epochs.

Application of the Trained Model on the Discussion Rounds

I now apply the trained model to the audio of all identified discussion rounds of the 12 relevant streamers. To guarantee the best possible audio quality for the respective target speaker, I extract the audio of the discussion rounds from the VODs of the respective target speaker. After extraction, I convert these audio files into the required format by resampling the sampling rate and converting from stereo to mono audio. I then determine the acoustic features and the similarity scores for the audio files using the method described above in order to give them to the model as input together with the *d*-vector embedding of the respective target speaker. In total, I apply the model to 6,829 different discussion rounds.

As can be seen in Figure 5.4, the length of the discussion rounds is close to the average value of 144.9 seconds for the majority of them. For each discussion round, I classify each 10ms window. This results in a total of 99,014,803 classified frames. Figure 5.5 shows that most of the frames are classified as Non-Target Speaker Speech, the second largest number is classified as No Speech, and the smallest number is classified as Target Speaker Speech. This result is to be expected, as the target speaker is only one of many discussion partners in the discussion rounds.

Combination of Speaker Diarization and Transcription.

Demand for Precision When Combining Speaker Diarization With Transcription In a final step, I combine the results of the speaker diarization with the transcription in order to extract the utterances made by each streamer in the discussion rounds. Since I then use the resulting data to explain player behavior, this step attempts to achieve a high level of precision in order to provide the foundation for a successful analysis. Thus, the idea is to only assign utterances to streamers if they are actually made by them with sufficient certainty. For this purpose, I use the fact that the PVAD Model has calculated probabilities for each frame for each of the three classes, which can be determined by applying softmax to the model output. Based on these probabilities, I perform some descriptive analyses to develop heuristics on how to maximize the precision of the final utterances assigned without losing too much data by using a threshold for considering a frame classification as target speaker speech.

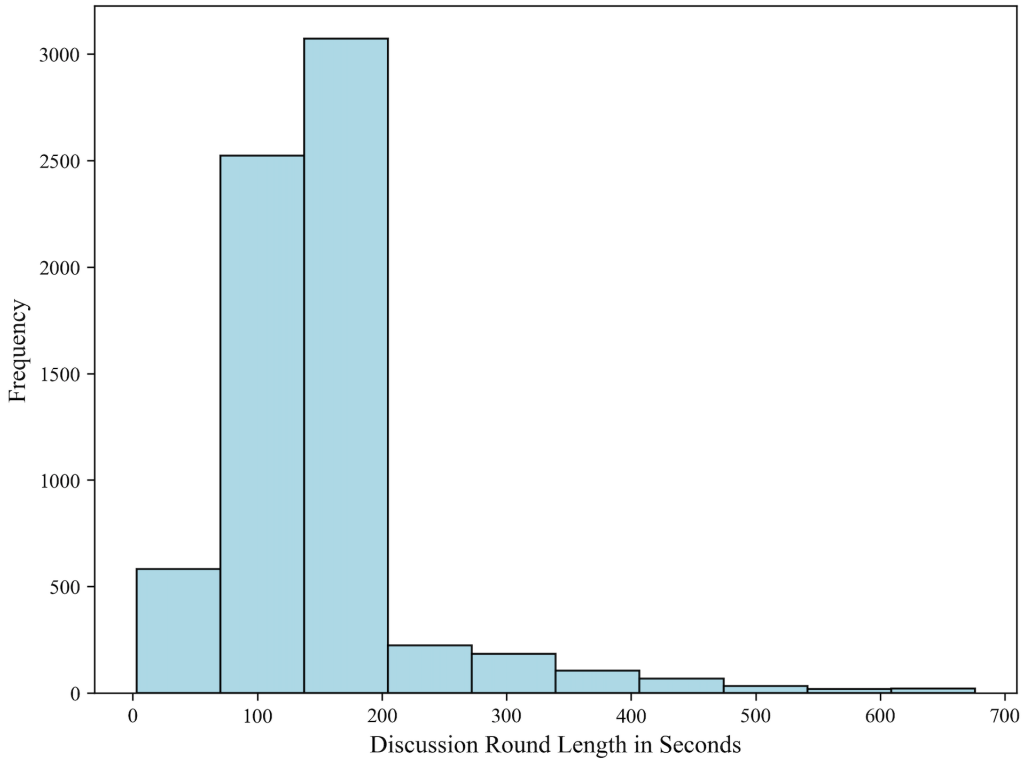


Figure 5.4: Distribution of Length of Analyzed Discussion Rounds.

Figure 5.6 shows the mean ratio of frames of a discussion round for which target speaker speech was classified that would be retained if a certain probability threshold of frames classified as target speaker speech had to be exceeded. The final procedure for assigning individual transcribed utterances to individual streamers, which I choose based on this descriptive analysis is outlined in the following: For each streamer, for all discussion rounds in which he is involved, I take the transcription of this discussion round with the audio file from the streamer's VOD as a basis. Secondly, I use the output of the PVAD Speaker diarization model applied to the audio file of this discussion round with the respective streamer as target speaker. The *ts-Whisper* model I use for the transcription not only provides the transcribed words with their time points, but also suggestions as to which of the words belong together, i.e. which words are part of a coherent transcribed utterance. These coherent utterances often comprise several words, but can also consist of just a single word, such as an exclamation. In order to assign what has been said to the individual streamers based on the output of the speaker diarization model, I consider these coherent utterances instead of individual words. I therefore always assign a streamer such a coherent utterance, but never just a part of it. To check whether an utterance can be assigned to a streamer, I first identify the exact timestamp of the first and last word of the utterance using the timestamps noted in the srt file of the corresponding transcription. All frames contained in the utterance I then use to

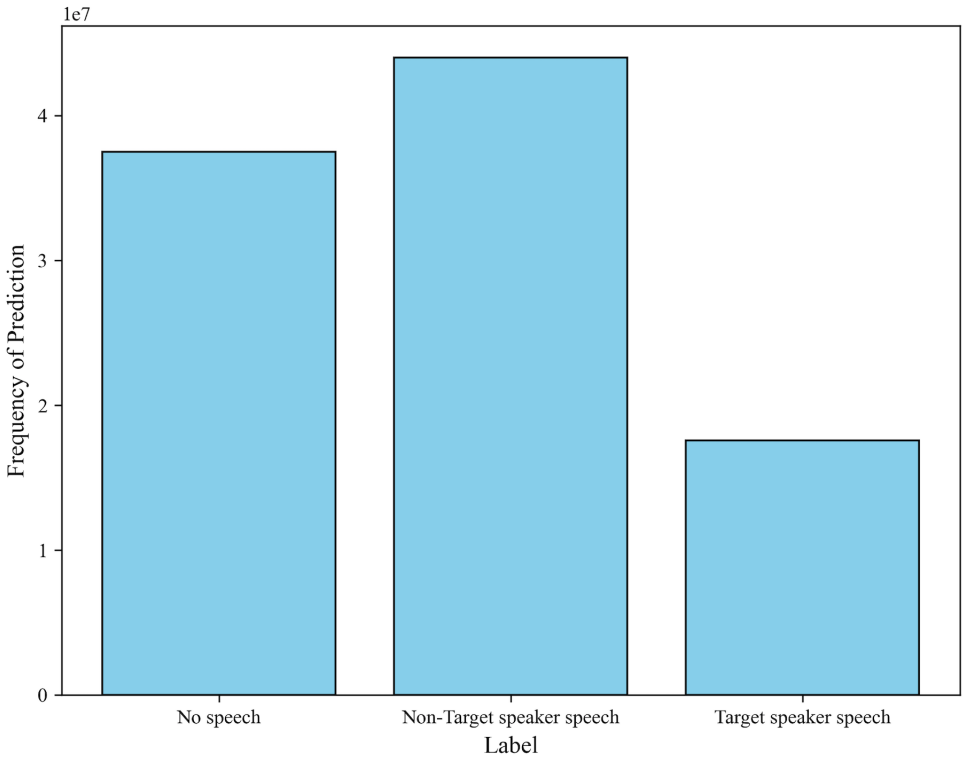


Figure 5.5: Distribution of PVAD Classifications Over all Classified Frames.

check whether the utterance can be assigned to the respective streamer (by using the corresponding model output in which the streamer is defined as a target speaker): For this, a certain ratio of frames of the utterance must be assigned to the streamer. To be assigned to the streamer as a frame, it is not sufficient to be classified with Target Speaker Speech. In order to exclude uncertain classifications as target speaker speech, I only consider classifications with a probability for the target speaker speech class above a certain threshold. The choice of these two thresholds - the minimum probability for a frame classification to be considered as target speaker speech and the minimum ratio of such frames required for an utterance to be assigned to a streamer - defines the precision of the extracted utterances. I define a total of three precision levels, with utterances being extracted for each of these for all streamers: The low precision level with a minimum probability of 0.5 for the consideration of frames classified as target speaker speech and a minimum ratio of 0.6 of such frames for an utterance assignment, the medium precision level with corresponding thresholds of 0.8 and 0.6, and the high precision level with thresholds of 0.8 and 0.8.

Other Approaches to Ensure Precision of Utterance Assignment It should be mentioned here that another way to increase the precision of the extracted utterances would be to look at the model output of the same discussion round for different streamers. In this case, only

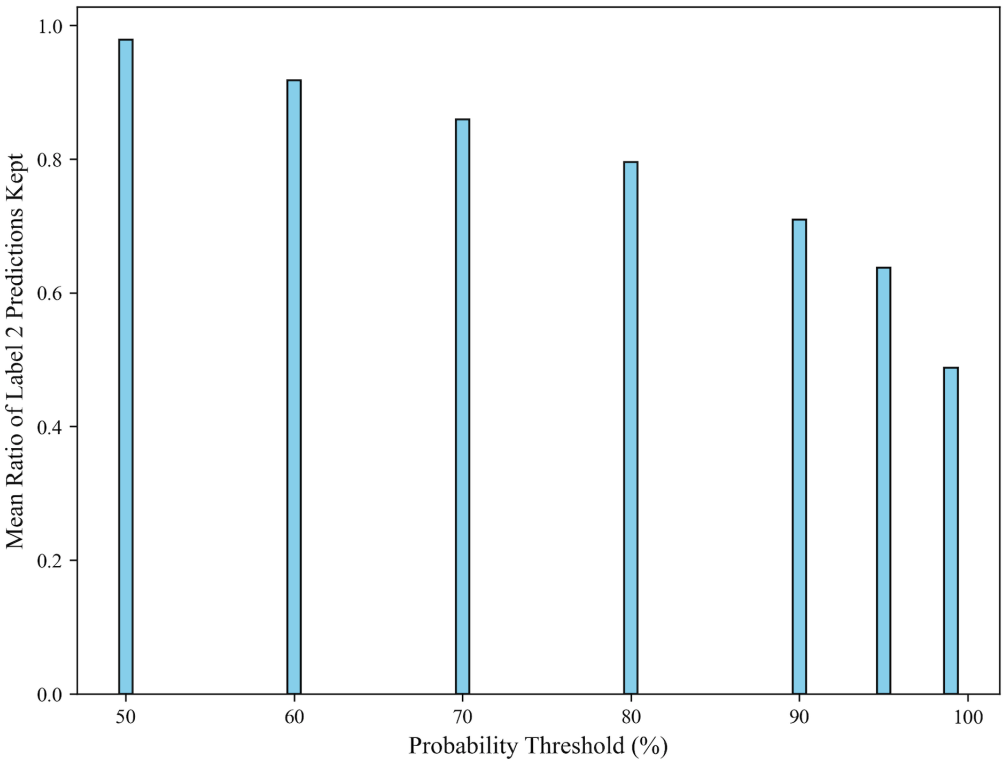


Figure 5.6: Mean Ratio of Frames Classified as Target Speaker Speech of a Discussion Round Kept for Different Minimum Probability Thresholds.

those frames that are classified consistently across all outputs could be considered further, i.e. only those that are classified as target speaker speech for a maximum of one streamer. However, this option is not applicable in the present case, as the extracted audio excerpts from the different VODs of the participating streamers show marginal differences in length for the same discussion round. As already described, this is due to the video extraction tool I use. Due to these marginal differences, the number of frames of the discussion round is not the same for the different streamers, which is why a comparison is not possible.

Combination of Coherent Utterances Regardless of the precision level, I make an attempt to combine coherent utterances for each streamer after extracting the utterances they have made. The reason for this is that coherent utterances often only make semantic sense in their entirety. This semantic information of an utterance would be lost by splitting it into shorter partial utterances, or the partial utterances might have a different, originally unintended semantic meaning. Therefore, I combine all utterances assigned to a streamer from a discussion round that are less than 2 seconds apart into one coherent utterance. At the end of the combination of the transcription of the individual discussion rounds with the results of the speaker diarization, there is

now a dictionary that lists all the utterances assigned to each streamer for each discussion round in which he was involved. There are three different lists of utterances for each discussion round depending on the precision level of the extracted utterances.

Evaluation

In order to evaluate the extracted utterances for their precision, I select ten random utterances assigned to a streamer for each level of precision. For the individual utterances, I then use the VOD of the corresponding streamer to manually verify whether the utterance was actually made by the specific streamer in the specific discussion round. To measure the success of the assignment of the utterances to streamers, I score an incorrect assignment with 0 and a correct assignment with 1. In some cases, one part of an utterance is spoken by the postulated streamer, while the second part is spoken by another streamer (in these cases, there is an error in the generation of the coherent utterances). These cases I score with 0.5.

Streamer	Session	Lobby	DR	Accuracy	Utterance	Correct Speaker Assignment	Player Dead
zeroyalviking	2022-02-08_S1	8	2	low_acc	Five seconds ago, Kay, someone attacked you.	1	0
skadj	2022-02-09_S1	14	3	low_acc	I just can't care that there's more Futurama coming because it's like been cancelled and revived so many times. It's a fucking walking zombie at this point. I don't know.	1	1
ayanehylo	2022-02-24_S1	15	1	low_acc	Nice.	1	1
irepptar	2022-02-16_S1	8	4	low_acc	Alright,	0	0
zeroyalviking	2022-05-24_S2	4	0	low_acc	hey	0	0
aribunnie	2022-02-01_S1	12	0	low_acc	yeah who was either oh then it wasn't ari then it was just casper wait who was i there with	0.5	0
aribunnie	2022-01-27_S1	17	0	low_acc	i	1	0
jvckk	2022-02-15_S1	10	0	low_acc	that makes so much more sense.	1	0
irepptar	2022-02-12_S1	16	0	low_acc	Yo, okay. We're not going to hide.	1	0
ozzaworld	2022-02-15_S1	10	4	low_acc	you got the kill!	1	0
skadj	2022-02-04_S1	3	3	mid_acc	the hint that he gave me is that his role was just neutral. know, looking at all the neutral roles, I figure he's most likely Arsonist. If he were killer,	1	1
ozzaworld	2022-03-03_S1	6	0	mid_acc	I feel sorry for his boss, Rupana.	0	1
vikramafc	2022-02-21_S1	18	0	mid_acc	But chat, chat, we're eight away from hitting 90 subs in a day. That would be insane for me coming back to my, If anyone hasn't already hit the sub button, I really would appreciate it. If we can hit 90 for the day, that would be actually amazing.	1	0
courtilly	2022-03-03_S1	13	2	mid_acc	I'm wearing my boba shirt from Kale and I'm drinking boba.	1	0
jvckk	2022-02-15_S1	13	1	mid_acc	Where is that?	1	0
zeroyalviking	2022-03-01_S1	1	3	mid_acc	I did one kill.	1	1
aribunnie	2022-03-10_S1	11	2	mid_acc	you know what? That proves	1	1
jvckk	2022-03-09_S1	12	2	mid_acc	Do I find dripstone? Do I mine it? I think I find it, right?	1	0
vikramafc	2022-05-24_S2	3	1	mid_acc	We're one away from the big 10. There was a...	1	1
courtilly	2022-02-26_S1	7	1	mid_acc	Okay, we'll fix Skog.	1	0
irepptar	2022-02-08_S1	3	0	high_acc	From now. You tell him.	1	0
karacorvus	2022-03-03_S1	11	1	high_acc	what do you think? You still voted me?	0.5	0
irepptar	2022-02-08_S1	10	1	high_acc	sorry guys sorry guys i'm a bit of a thing	1	1
skadj	2022-02-23_S1	15	0	high_acc	That's awesome.	1	1
jvckk	2022-02-01_S1	7	0	high_acc	Literally	0	0
irepptar	2022-05-24_S1	13	1	high_acc	I actually, I'm going to shield Jay.	1	0
courtilly	2022-02-22_S1	13	2	high_acc	Hi. Welcome. Oh my lord.	0.5	1
jayfletcher88	2022-02-01_S1	3	4	high_acc	See you later, man. Can someone explain?	0.5	1
ozzaworld	2022-02-15_S1	12	0	high_acc	this happened,	0	1
irepptar	2022-02-08_S1	9	2	high_acc	you could have won this whole thing.	0	0

Table 5.6: Evaluation Results of the Assignment of Utterances to Streamers.

As can be seen from Table 5.6, the assigned utterances achieve an average precision score of 0.73. This score suggests that about three quarters of the utterances assigned to a streamer actually be-

long to him, which is a promising result for the subsequent analysis of the data: since the majority of the utterances assigned to a streamer actually come from him, I can use them for a meaningful analysis of his behavior. The differences in assignment quality for the levels of precision do not appear to be significant; the average precision score in the randomly selected utterances is even higher for the low (0.75) and medium (0.9) levels of precision than for the high (0.55) level. In particular, the assignment seems to be difficult for short utterances, as the average precision score for the five utterances with only one word is 0.4, while it is 0.8 for the remaining longer utterances. Due to the structure of the game, it often happens that players are already dead at the time of discussion rounds, which was twelve times the case here. However, the performance of the presented methodology does not seem to depend on whether the players are still alive, as the difference between the average precision score of living (0.72) and dead players (0.75) is marginal.

The evaluation method presented here focuses on evaluating the precision of the assignments, as this is crucial for the use case at hand: For the analysis of player behavior, it is elementary that the data used actually originates to a large extent from this player. For other use cases in which, for example, recall is crucial, a different evaluation strategy would have to be selected: In these cases, one would have to manually extract all utterances made by a random streamer in randomly selected discussion rounds in order to subsequently see how many of these utterances were actually assigned to the streamer. However, since this is not crucial in the present use case, I do not do it here.

5.3 Player Role identification via Image Recognition

5.3.1 Methodology

Figure 5.7: Splash Screen Assigning Player a Role at Lobby Start.

Crewmate	Neutral	Impostor	Crewmate	Neutral	Impostor
Crewmate	Neutral	Impostor	Crewmate	Neutral	Impostor
Mayor	Jester	Godfather	Detective	Amnesiac	Grenadier
Medium	Arsonist	Mafioso	Haunter	Guardianangel	Morphling
Swapper	Jackal	Janitor	Investigator	Survivor	Swooper
Timemaster	Sidekick	Morphling	Mystic	Executioner	Traitor
Engineer	Vulture	Camouflager	Seer	Jester	Blackmailer
Sheriff	Lawyer	Vampire	Snitch	Phantom	Janitor
Deputy	Prosecutor	Eraser	Spy	Arsonist	Miner
Lighter	Pursuer	Trickster	Tracker	Plaguebearer	Undertaker
Detective	Thief	Cleaner	Trapper	The Glitch	Bomber
Medic		Warlock	Sheriff	Werewolf	Escapist
Seer		Bounty	Veteran	Doomsayer	Venerer
Hacker		Hunter	Vigilante	Juggernaut	Warlock
Tracker		Witch	Altruist	Vampire	Poisoner
Snitch		Ninja	Medic		Underdog
Spy		Bomber	Engineer		
Portalmaker		Yo-yo	Mayor		
Security Guard		Evil	Medium		
Medium		Guesser	Swapper		
Trapper			Transporter		
nice guesser			Aurial		
Bait			Hunter		
Shifter			Imitator		
			Oracle		
			Prosecutor		
			Vampire Hunter		
			Timelord		

Table 5.7: Available Player Roles in The Other Roles (Left) and Town of Us (Right).

Extraction of Relevant Images In each new lobby, a player is assigned a new role. Since the role is crucial for the objective of the respective player and thus also influences his behavior, information about the role is very valuable for the subsequent analysis. For this reason, I make an attempt to extract the role assigned to each streamer for each lobby. Since the roles are always assigned at the beginning of a lobby in a consistent splash screen, which can be seen in Figure 5.7, I use image recognition for this purpose: First, I extract relevant images, then I extract text from them by applying OCR in order to then check it for similarity with actually existing roles using the Levensthein distance. Since the role assignment always takes place at the beginning of a lobby, I always define the extracted images by the time of the identified lobby start of the corresponding lobby. I select a total of 16 frames for each streamer: The first frame is recorded 5 seconds before the lobby start, the last 10 seconds after, with one frame extracted per second.

Application of OCR on Extracted Images In order to prepare the extracted images optimally for the OCR application, I carry out several preprocessing measures: First, I crop the images to the relevant area containing the role to be recognized. Then I convert the image into a binary image by thresholding to enhance contrast and remove unwanted details. Then I apply morphological opening to remove small distortions and make the text areas clearer, before inverting the image to make the text easier to recognize in front of a dark background using OCR. Then I perform the OCR using the *Tesseract* tool, which is integrated into Python via the *pytesseract* library. I transfer the prepared image to *Tesseract* which returns the recognized text and confidence values. I only extract text with a confidence value above the threshold of 0.5. In order to derive the role of the streamer from this text, I examine it for similarity with existing roles. This requires a dictionary that lists all existing roles for the two modifications TownOfUs and TheOtherRoles analyzed in this thesis. Such a dictionary is created manually and is shown in Table 5.7.

Role Assignment Based on Extracted Text To assign a role to a streamer's lobby, I extract text for all 16 preprocessed images. In this text - whenever it is included - I remove the expression "Your role is", which always appears on the split screen when roles are assigned. Then I check the text of each image for similarity with all existing roles of the respective game using the Levensthein distance. For each picture, I record the role with the smallest Levensthein distance as the detected role if the calculated Levensthein distance was less than 4. In order to find a role for the lobby, I make a majority vote over all the extracted roles from all the images: the role that I identified in the most images is used as the lobby role. The role "Spy" is an exception, as it often has only a small Levensthein Distance with the extracted text due to the brevity of the word and is therefore incorrectly identified as a role for some images. Whenever I detect "Spy" as the role for the most images in a lobby, I assume the second most frequently detected role. If there is no second detected role, I assume "Spy" to be the role for now.

Manual Examination of Remaining Unidentified Lobbies A total of 2,131 out of 2,838 lobbies can be successfully identified using this methodology. For the remaining lobbies, either no role can be identified, or the identified role is "Spy" or "Seer" (which was partly correct, but partly also an incorrect assignment due to the short word length). I carry out a manual examination for all these lobbies using the extracted images. Thereby I discover that role identification is not possible for 44 lobbies, either because the streamer joined this lobby late or because the identified lobby start time is incorrect. I remove these special cases, so the lobbies are no longer included in the further analysis for the respective streamers. For the remaining lobbies, I identify the correct roles manually and then combine the results of the manual and programmatic role identification. The final result is a dictionary that lists the assigned roles for 2,794 lobbies of different streamers.

5.3.2 Evaluation

In order to evaluate the methodology for identifying the roles, I select 10 lobbies randomly and compare their identified role with the actual role. As can be seen in Table 5.8, the identified role matches the correct role every time. It can therefore be assumed that the identified roles are correct for the respective streamers, which is elementary for analyzing the game behavior of the players later on the basis of these roles.

Streamer	Session	Lobby	Identified role	Correct role
aribunnie	2022-01-27_S1	16	the glitch	the glitch
zeroyalviking	2022-02-15_S1	5	amnesiac	amnesiac
courtilly	2022-05-24_S1	1	investigator	investigator
skadj	2022-02-09_S1	11	jester	jester
zeroyalviking	2022-01-27_S1	5	the glitch	the glitch
zeroyalviking	2022-03-01_S1	1	medium	medium
jvckk	2022-03-09_S1	17	arsonist	arsonist
skadj	2022-02-23_S1	3	transporter	transporter
courtilly	2022-01-27_S1	2	veteran	veteran
karacorvus	2022-02-19_S1	19	underdog	underdog

Table 5.8: Evaluation Results of Image Recognition for Role Identification.

5.4 Combination of Extracted Text Data With Extracted Player Roles

In the final step of the information extraction pipeline, I combine the results from the text extraction and the role extraction. The text extraction takes place at the level of the discussion rounds, while the role extraction is carried out at the level of the lobbies. Therefore, I assign the role extracted for a lobby to all the discussion rounds contained in it. This results in a dictionary with the following form as the final data source for the analysis of player behavior: For each streamer, there is an entry for each discussion round in which he participated, with the utterances he made in the discussion round (for three different levels of precision) as well as his role while he made these utterances.

In total, the final data comprises 6,777 discussion rounds of individual streamers. If you look at the frequencies of the roles, shown in Figure 5.8, you can see that the distribution is not even. While there are only two discussion rounds for some roles, players were assigned the two most frequent roles "Sheriff" and "Jester" in a total of 594 and 559 discussion rounds respectively. If the roles are grouped into the three main roles of Crewmate, Neutral and Impostor, as can be seen in Figure 5.9, a picture emerges which is expected due to the dynamics of the game: the Crewmate role has the most data with 3,262 discussion rounds, followed by Neutral (2,234) and Impostor (1,281) at a considerable distance.

If one looks at the number of extracted utterances, a distinction must be made according to the level of precision. For example, the low degree of precision has the highest average of 5.86 utterances per discussion round of a streamer, while slightly fewer utterances were extracted for the medium (5.04) and high degree of precision (3.39). The average length of an utterance also decreases with increasing precision level: While the utterances are on average 9.25 seconds long for the low precision level, the average duration decreases to 8.59 seconds (medium) and 5.96 seconds (high) for a higher extraction precision. As can be seen from Figures 5.10 and 5.11, which show the average length of utterances and the average number of utterances separated by role for the low precision level, the length and number are evenly distributed across the roles. The presentation

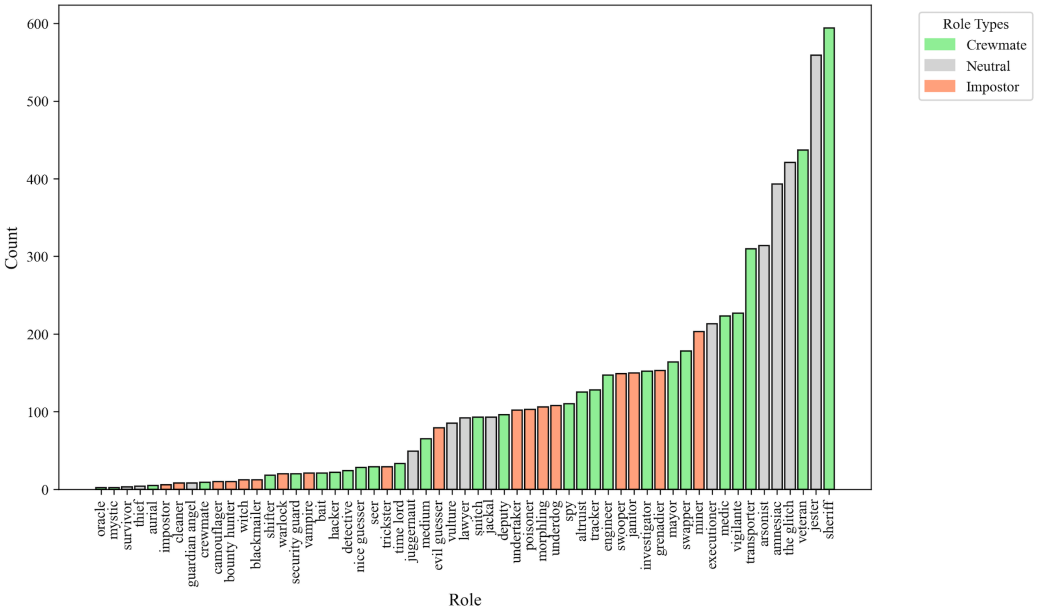


Figure 5.8: Number of Discussion Rounds of Individual Roles.

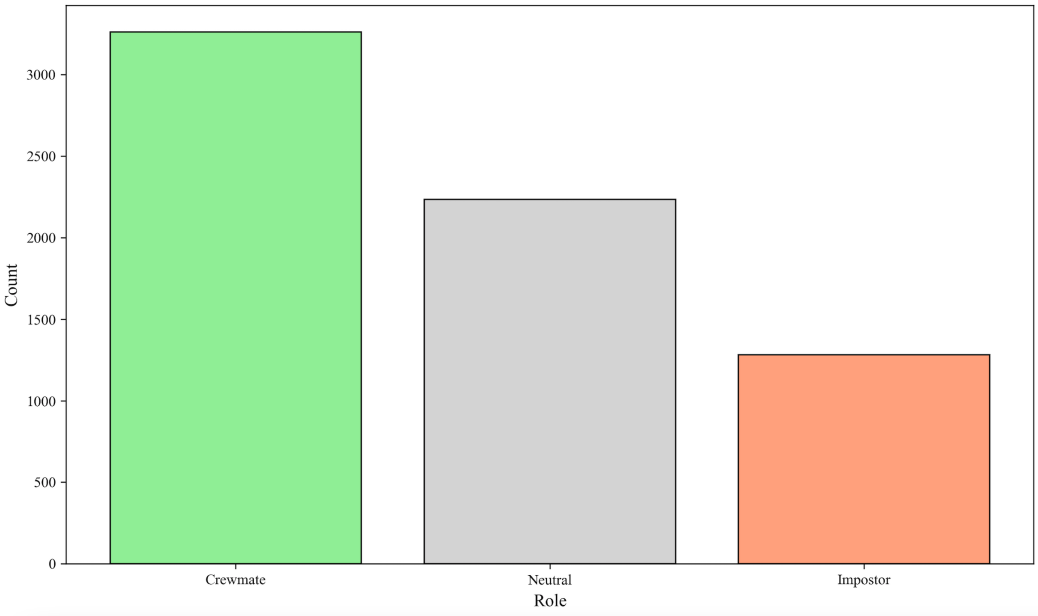


Figure 5.9: Number of Discussion Rounds of Main Roles.

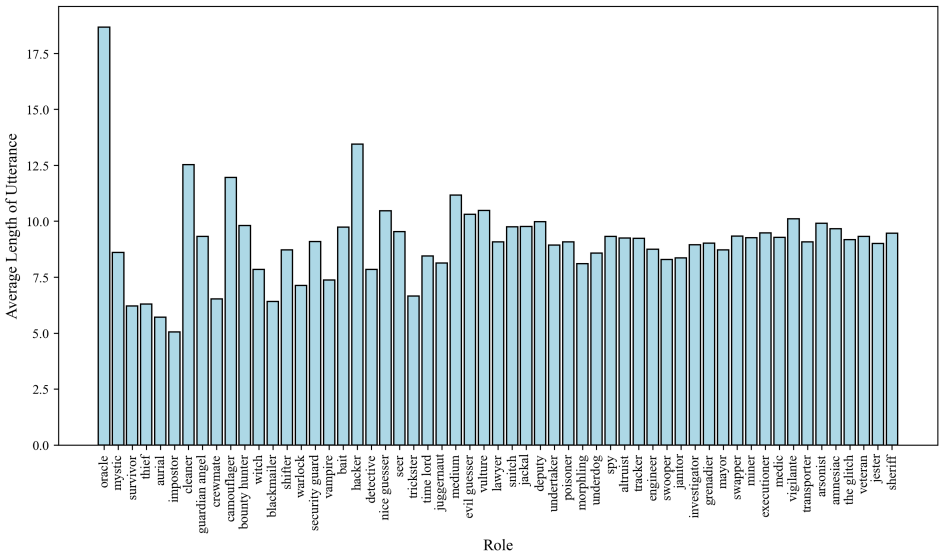


Figure 5.10: Average Length in Seconds of Extracted Utterances by Role (Low Precision).

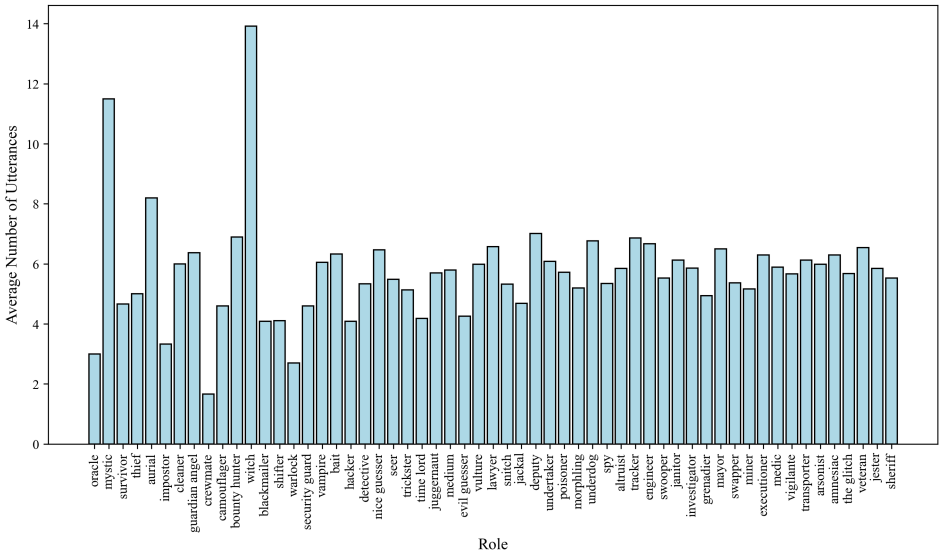


Figure 5.11: Average Number of Extracted Utterances by Role (Low Precision).

of the average length of utterances and the average number of the utterances separated by role for the other two precision levels is shown in the Appendix A.1, A.2, A.3 and A.4.

6 Language Analysis of Players' Utterances

When analyzing player behavior, I primarily attempt to predict the role of a player. The underlying argumentation structure is that a successful prediction of the role means that patterns are recognized in the player's behaviour (his utterances) that enable a successful prediction. These recognized patterns then provide information about a certain behaviour. In order to analyze the players' behavior with NLP methods, I follow two basic lines of analysis. Firstly, I analyze the extracted data descriptively using popular standard methods of text mining. In addition to descriptive analysis of the data, I examine the extent to which the role of the player can be predicted using these methods. The methods I use include a sentiment analysis, an analysis of the POS used, a word frequency analysis and a LDA topic analysis. Secondly, I fine-tune a pre-trained BERT model for text classification to the task of predicting the role of the streamers based on their utterances. For this model, I then make an attempt to determine the reasons for its good classification performance using feature importance methods. In other words, I examine the patterns discovered by the model in the players' behavior. In the following, the two lines of analysis are discussed separately in more detail by always presenting the methodology and the results.

6.1 Descriptive Analysis via Popular Text Mining Methods

6.1.1 Sentiment Analysis

I use the *cardiffnlp/twitter-roberta-base-sentiment-latest*, a RoBERTa-based model, for the sentiment analysis. The model is defined in a way that it assigns a sentiment score between -1 (negative) and 1 (positive) for a given utterance. In order to analyze the sentiment in the streamers' utterances from the discussion rounds, I divide the utterances into the three main roles Crewmate, Neutral, and Impostor. For each group, I classify all utterances for their sentiment. The sentiments of the different roles hardly differ. Looking at the proportion of utterances classified as negative, neutral and positive for the three groups, only marginal differences can be seen. The average sentiment score of all utterances of a role is also very similar. These findings can be seen in Figure 6.1 and 6.2. The two Figures shown depict the results for utterances with a high level of precision; the results for the other two levels of precision draw the same picture.

To summarize, the sentiment analysis provides two results: First, most utterances have a neutral sentiment. This is not surprising, as the utterances come from discussions in which the players largely report objectively on events or describe theories. Secondly, the sentiment of the utterances does not differ for different roles. In the game setting analyzed, it is therefore not possible to draw conclusions about a player's intentions and goals based on the sentiment of an utterance.

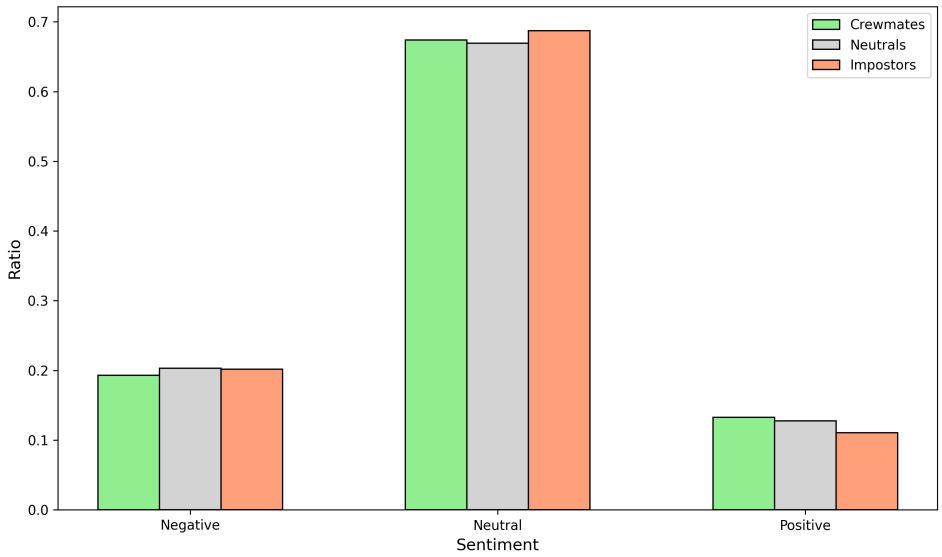


Figure 6.1: Distribution of Sentiments of Utterances by Role.

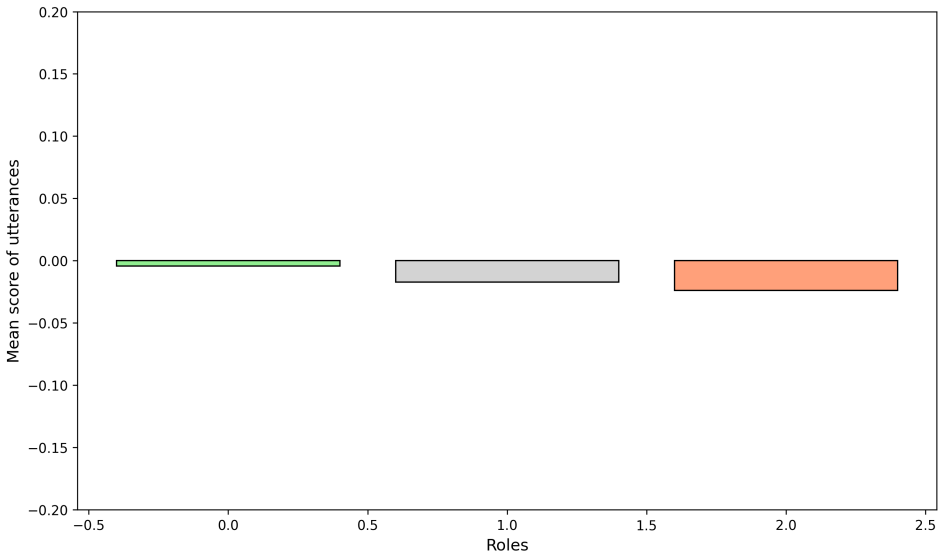


Figure 6.2: Mean Sentiment Score of Utterances by Role.

6.1.2 POS Tag Analysis

The *NLTK* POS Tagger with the associated *Penn Treebank* Tagset is used to tag the utterances with POS tags. The analysis examines the question whether streamers in certain roles tend to use certain POS tags more frequently. For this purpose, I separate the utterances according to the main roles again, then I tag each token for all utterances. For each main role, I calculate the share of each POS tag of all POS tags assigned within the role. Then I compare these shares of the various POS tags between the base roles. Figures 6.3, 6.4, and 6.5 for example, shows the differences in the shares of all POS tags between the three base roles. It should be noted here that all POS tags with a share of less than one percent are summarized under Rest. The Figures show utterances of the high precision level, but the same picture emerges for the other precision levels. The differences between the various roles in terms of the proportion of POS tags are marginal. It is therefore not possible to derive insights from the aggregated utterances of all streamers on how the use of certain POS tokens can provide information about a player's role.

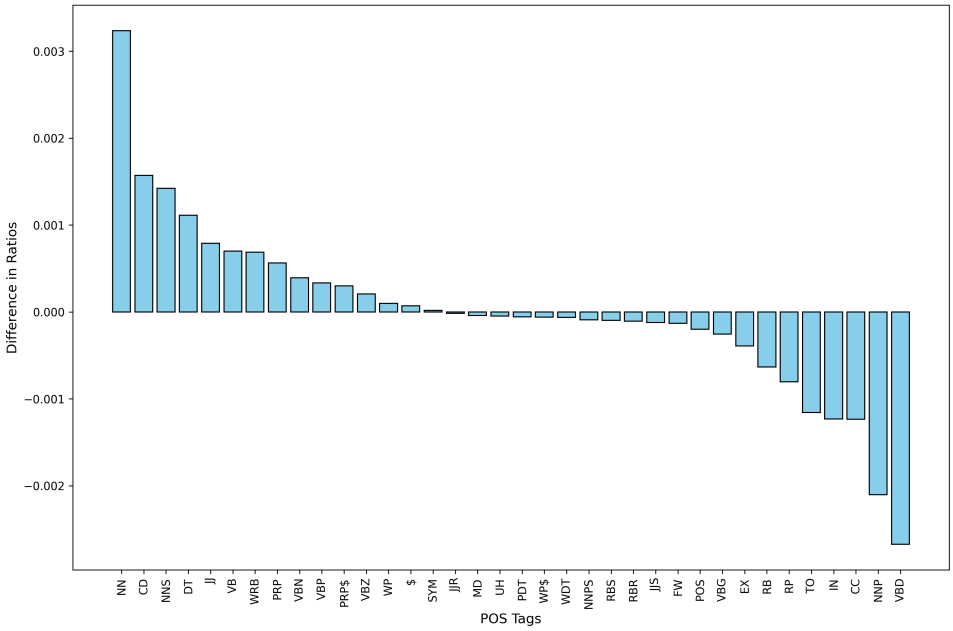


Figure 6.3: Difference of POS Tags Between Crewmate and Neutral.

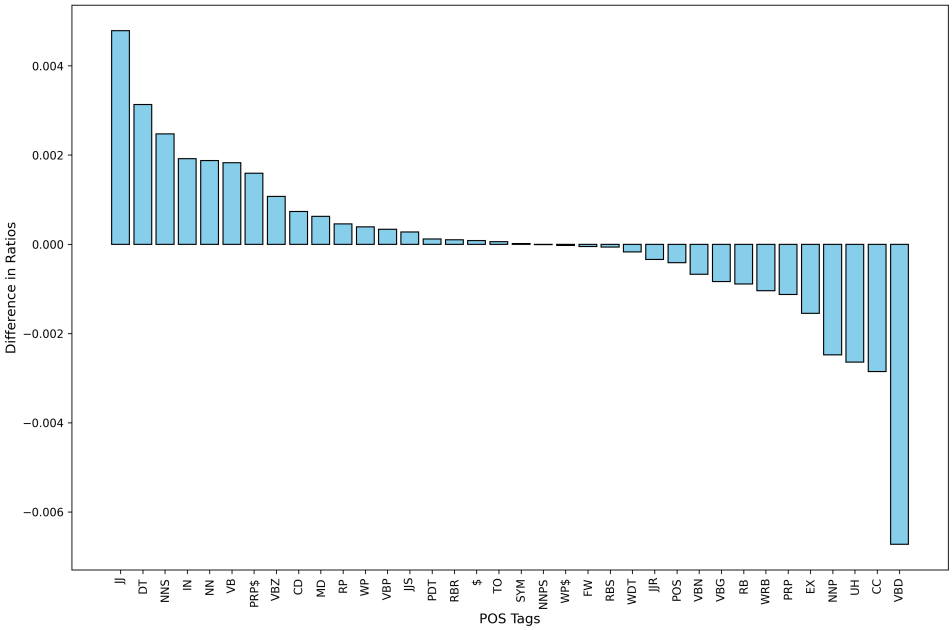


Figure 6.4: Difference of POS Tags Between Crewmate and Impostor.

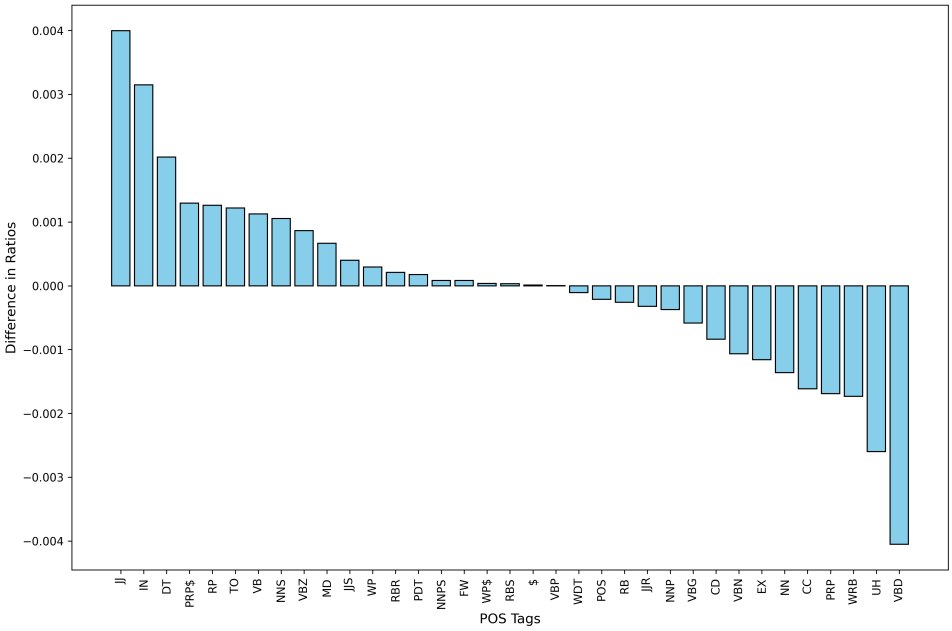


Figure 6.5: Difference of POS Tags Between Neutral and Impostor.

Driven by the idea that such patterns may exist rather on an individual level, as individual players have their own speaking habits, I carry out the same analysis at player level. For this, I perform the same analysis for the two streamers for which the most utterances are available, namely *Ozzaworld* with 5,485 utterances and *Zeroyalviking* with 2,801 utterances. However, as can be seen in Figures 6.6, 6.7, and 6.8, which show the differences in the proportion of POS tags between the various roles for *Ozzaworld*, the differences are marginal even when a filtered view on a single streamer is taken. This is also the case for the other precision levels and the streamer *Zeroyalviking* (see Appendix A.5, A.6, and A.7), which is why the conclusion remains that a role prediction based solely on the POS tags used is not possible.

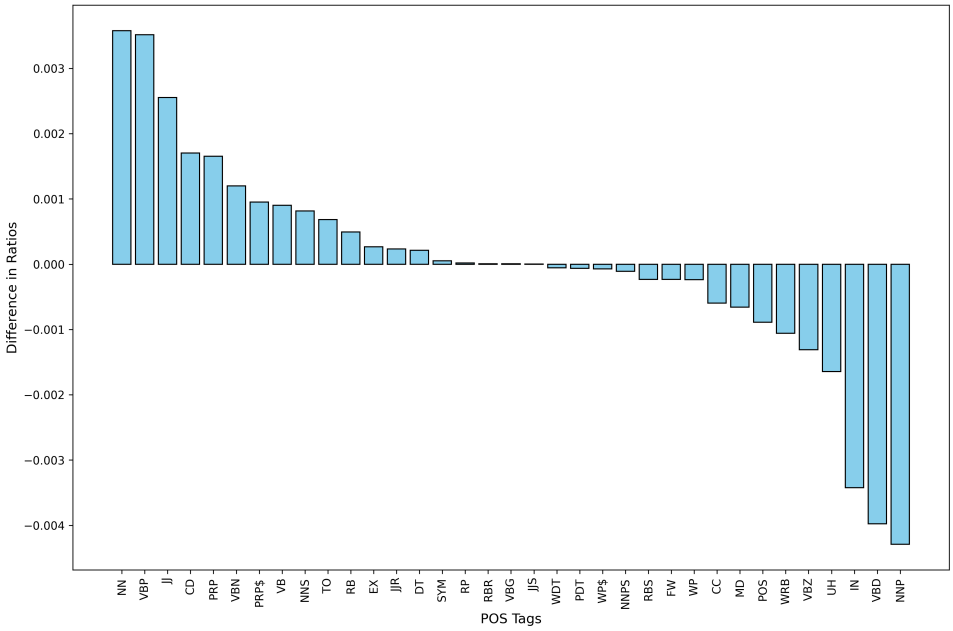


Figure 6.6: Difference of POS Tags Between Crewmate and Neutral for Ozzaworld.

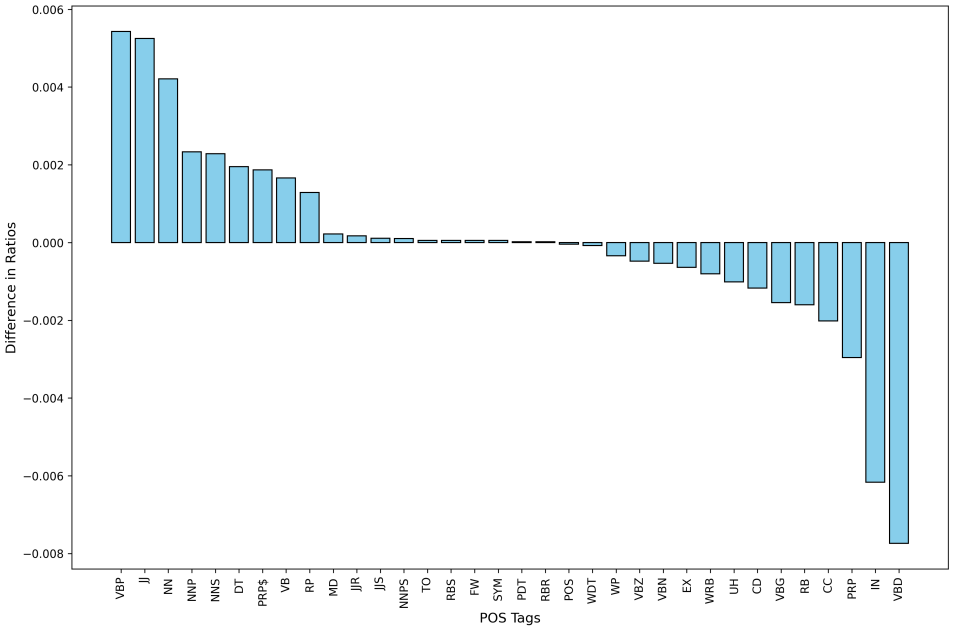


Figure 6.7: Difference of POS Tags Between Crewmate and Impostor for Ozzaworld.

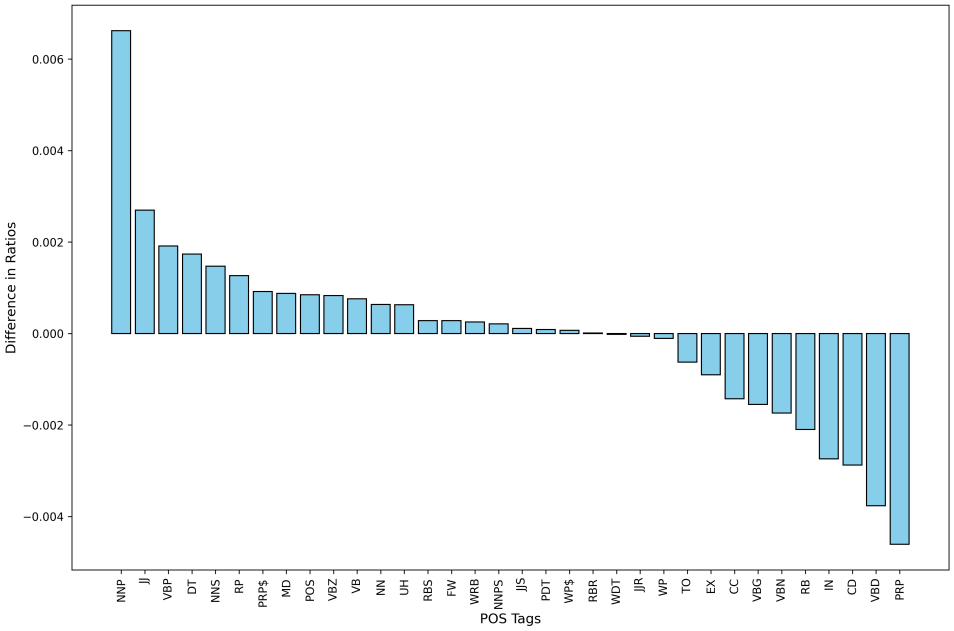


Figure 6.8: Difference of POS Tags Between Neutral and Impostor for Ozzaworld.

6.1.3 Word Frequency Analysis

In a further analysis step, I examine the words used in the discussions. The objective here is twofold: on the one hand, a descriptive representation of the words used is to provide interesting insights into the language used in discussions in an online gaming stream. On the other hand, it is to investigate whether there are certain words or word families that indicate a certain role. In order to base the analysis on a data set of words that is as large as possible, I carry out the analysis presented below with utterances of the low precision level. A total of 470,052 words are available for the analysis with the low level of precision, which originate from the utterances of all streamers. After removing punctuation and stop words, the number is reduced to 97,799. I use two lists to remove stop words, the *NLTK* English stop word list and a stop word list from *Countwordsfree*[1], which contains many interjections that occur frequently in the utterances presented here. Figure 6.9 shows the 20 most frequently used words, Figure 6.10 shows a word cloud of the 100 most frequent words.

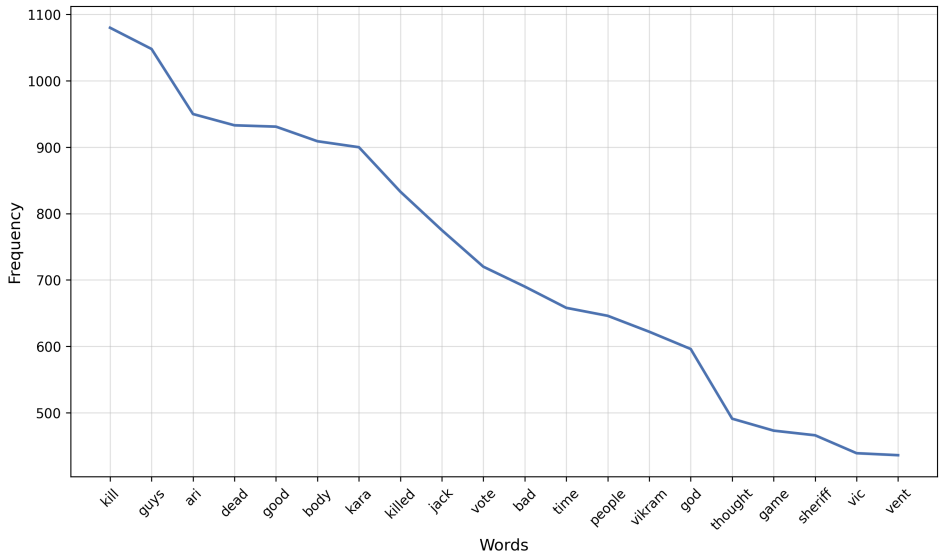


Figure 6.9: Top 20 Words Used in Utterances.

Since the top words contain many words from everyday usage, the next step is to find words that are explicitly used a lot in this game. To do this, I calculate the Term Frequency–Inverse Document Frequency (TF-IDF) score for each word before filtering for the words with the top score. The TF score I thereby calculate via

$$TF(t) = \text{Number of times term t appears in all utterances.} \tag{6.1}$$

[1] https://countwordsfree.com/stopwords

Figure 6.10: Wordcloud With Most Used Words.

I use the *Brown Corpus*, a standard corpus of American English used for linguistic analysis, to calculate the IDF of each word. For each word contained in the streamer utterances, I analyze the number of documents in the corpus in which the word occurs. I then calculate the IDF using the usual formula:

$$IDF(t, D) = log(\frac{|D| + 1}{|\{d \in D : t \in d\}| + 1}), \tag{6.2}$$

where $|D|$ is the total number of documents in the corpus and $|\{d \in D : t \in d\}|$ is the number of documents in which the term t appears. Based on the TF and the IDF score, the TF-IDF score can then be calculated for each word:

$$TF - IDF(t, D) = TF(t) \times IDF(t, D). \tag{6.3}$$

Figures 6.11 and 6.12 display the 20 words with the highest TF-IDF score and a word cloud with the top 100 words. The words with the high TF-IDF scores are mainly names of the streamers (e.g. *Ari* and *Kara*), roles of the game (e.g. Jester and Glitch), or verbs that describe an action that frequently occurs in the course of the game (e.g. kill and vote).

In order to analyze the extent to which predictions about the game role can be made based on the frequency of certain words, I focus the analysis again on the words of an individual player. The idea is again that individual players have different language habits with different vocabularies, which is why differences between different roles can only be revealed on an individual level. Once again, I analyze the streamer *Ozzaworld*, as most utterances are available for him. I separate his utterances by the main role before I carry out the word frequency analysis separately for both main roles. I limit the analysis to the two main roles Impostor and Crewmate, as the Neutral role com-

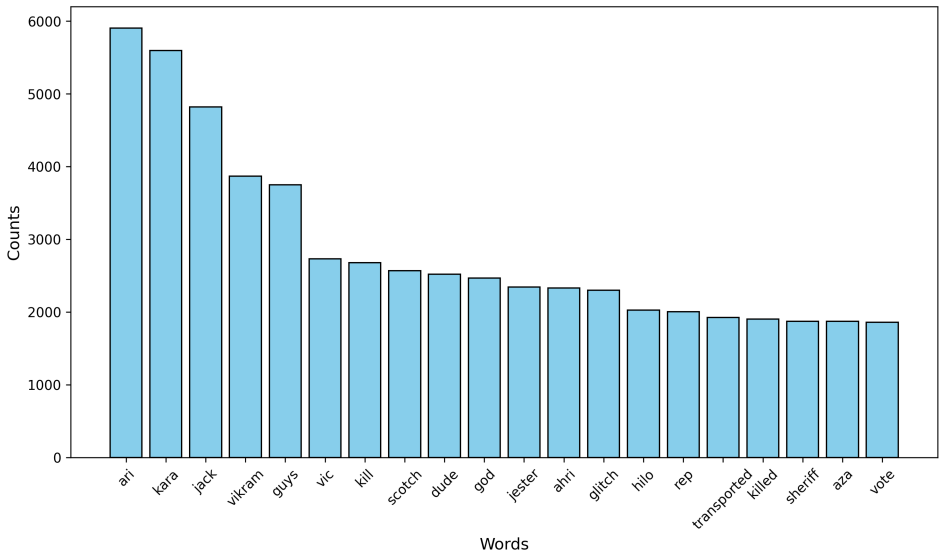


Figure 6.11: Top 20 Words Used in Utterances According to TF-IDF Score.

Figure 6.12: Wordcloud With Most Used Words According to TF-IDF Score.

pared to the other two roles often does not have such a clear objective function. In order to check whether statements about role membership can be made on the basis of word frequencies, I thus first test the simpler distinction between Crewmates and Impostors. For the utterances of both basic roles, I determine the most frequent words simultaneously to the methodology described above, once by pure TF score and once by TF-IDF score.

In order to draw a comparison between the results of the two basic roles, I make a comparison for each word said as to in which role it is said more frequently. Since for *Ozzaworld* the number of words as Crewmate is far greater than the number of words as Impostor, the TF and TF-IDF

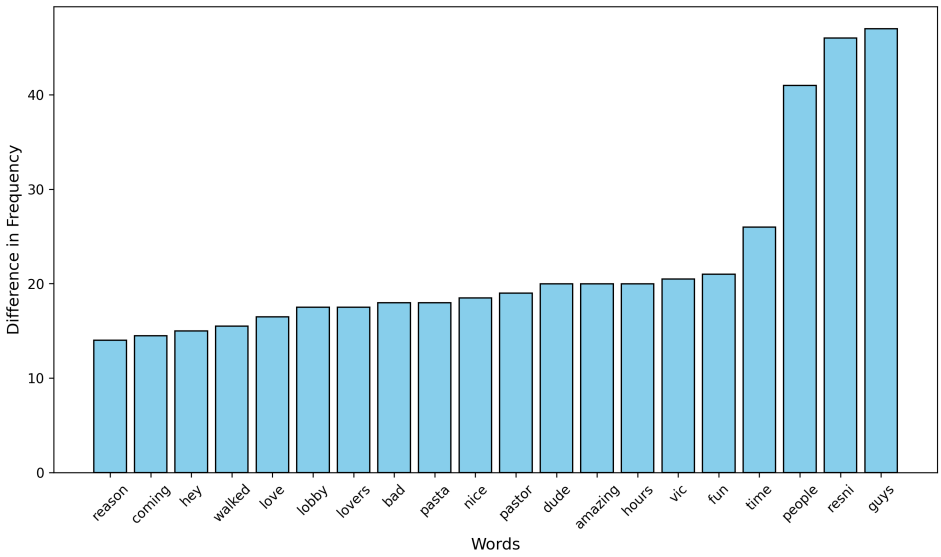


Figure 6.13: Top 20 Words Used More Frequently as Crewmate Than as Impostor for Ozzaworld Based on TF Score.

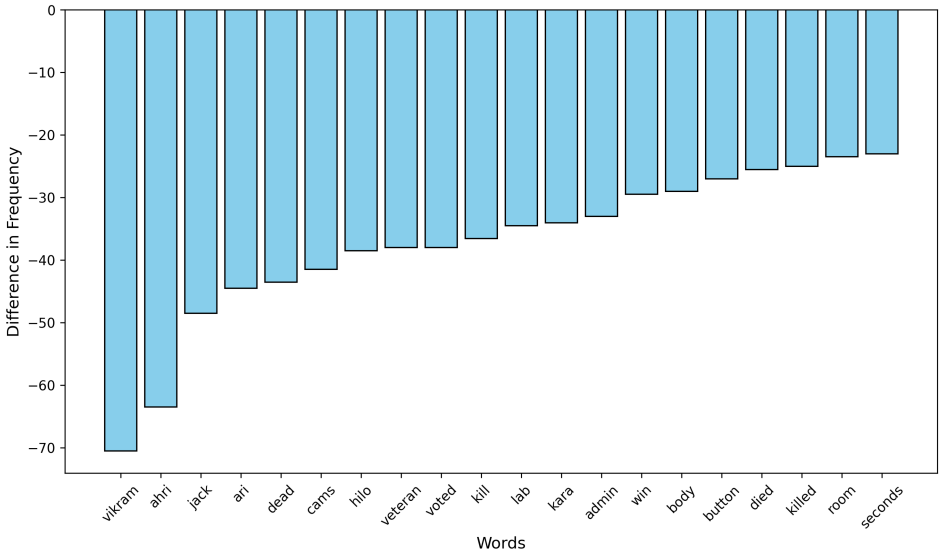


Figure 6.14: Top 20 Words Used More Frequently as Impostor Than as Crewmate for Ozzaworld Based on TF Score.

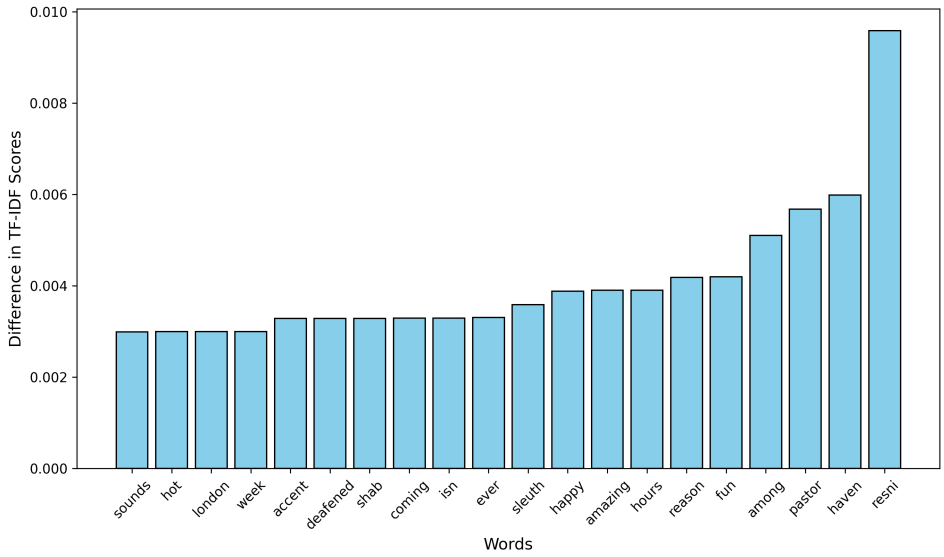


Figure 6.15: Top 20 Words Used More Frequently as Crewmate Than as Impostor for Ozzaworld Based on TF-IDF Score.

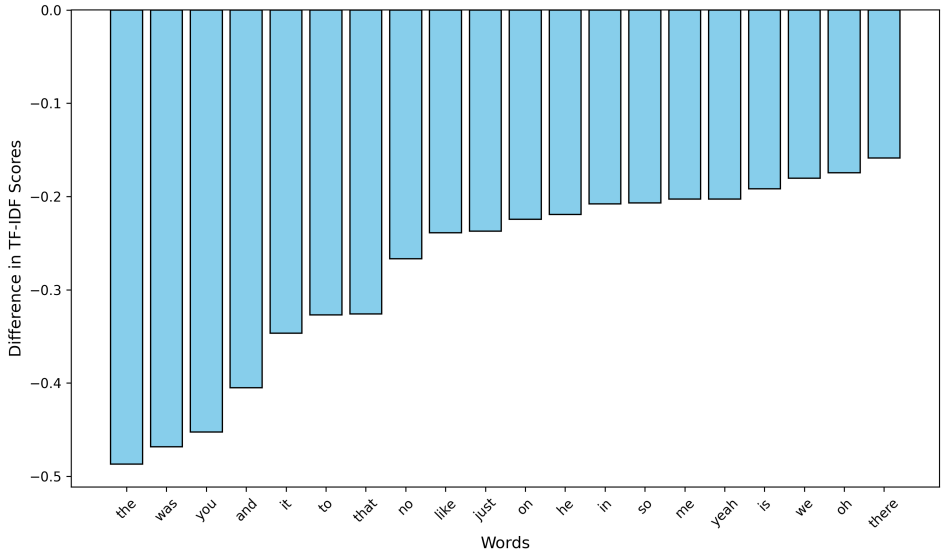


Figure 6.16: Top 20 Words Used More Frequently as Impostor Than as Crewmate for Ozzaworld Based on TF-IDF Score.

scores of the words said as Impostor are corrected upwards by the corresponding factor. This makes it possible to calculate the difference between the number of words said as Crewmate and Impostor for each word, and thus to derive how much more often the streamer would have said a word if he had been Impostor as often as Crewmate. Figures 6.13 and 6.14 show the 20 words with the greatest difference in favor of one of the two roles for the TF scores, Figures 6.15 and 6.16 for the TF-IDF scores. The difference represents how much more frequently a word is used as a Crewmate; thus, a negative difference means that the word is used more frequently as an Impostor. From the results, I can not recognize any clear patterns: Some of the words are again streamers and role names, but the remaining words do not draw a clear picture either. Thus, the conclusion is that the simple frequency of individual words does not provide any indication of the player's role. An explanation of behavior based solely on word frequencies therefore does not appear to be possible.

6.1.4 LDA Topic Analysis

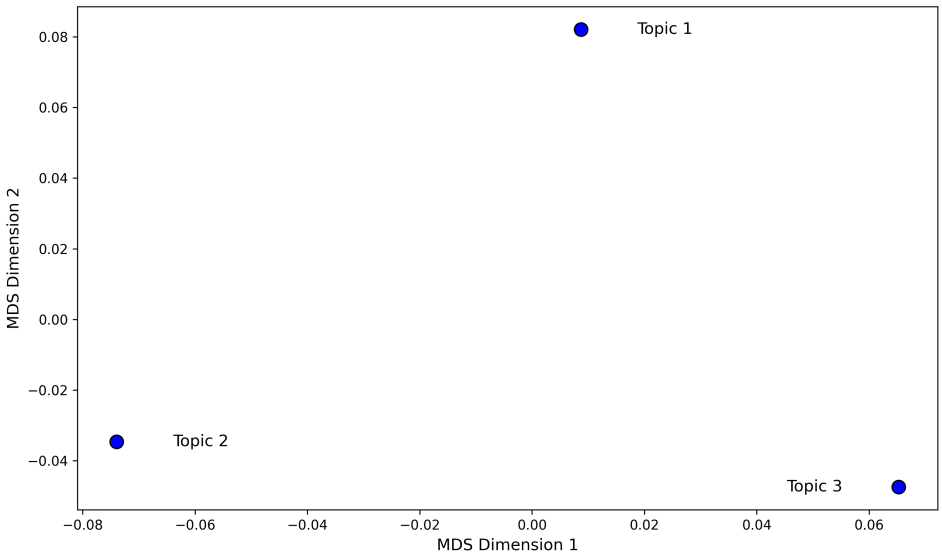


Figure 6.17: Distances of the Three Identified Topics Using LDA.

In a final descriptive analysis step, I examine whether it is possible to find recurring and dominant topics in the streamers' discussions using LDA. Before the analysis, the text corpus is preprocessed accordingly by removing the stopwords. I set the number of topics to be extracted by LDA to 3. Figure 6.17 shows the distances of the three identified topics with their respective 10 keywords being displayed in Figure 6.18. One can see that it is not possible to identify meaningfully distinct topics. The low coherence score of 0.48 also indicates that a topic analysis in this use case does not produce any meaningful results.

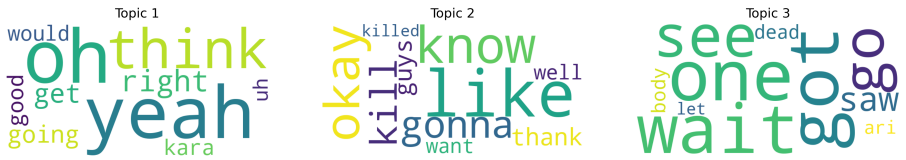


Figure 6.18: Wordclouds for the Three Identified Topics.

6.2 Classifier for Role Identification Based on Players' Utterances

6.2.1 Training and Evaluation of Classifier on Single Utterances

In the second strand of analysis, I train a classification model to predict the role of the streamers based on their utterances in the discussion rounds. Once again, I limit the model to the simple case of distinguishing between the two main roles Crewmate and Impostor. I take the individual utterances as input, which is why a total of 26,461 data points are available for the low precision level (medium precision level: 22,753 and high precision level: 15,292), of which 19,359 (16,687; 11,283) utterances come from Crewmates and 7,102 (6,606; 4,009) from Impostors. For the classification task, I do not train a model from scratch, but rather fine-tune the pre-trained model for text classification *Bert-Base-Cased*. For evaluation, I perform 5-fold cross validation. Due to the imbalance of the two classes in the training data, I oversample the minority class Impostor by duplicating randomly selected utterances that are made as Impostor within the individual folds until the number of Impostor and Crewmate utterances is equal. It should be noted that when oversampling, I always ensure that I do not duplicate any utterances from the fold that is used for evaluation. This prevents the risk of data leakage. I train the model for 3 epochs each with a learning rate of $2e^{-5}$, the batch size is 8. Each of the formed folds I use once as a test set to check the performance of the corresponding model on unseen data. I perform the described training on all three existing data sets (low, medium and high precision utterances). The accuracy scores for all three precision levels are shown in Table 6.1.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Low Precision Utterances	60.15	58.57	59.76	60.82	62.52
Medium Precision Utterances	62.51	55.83	61.17	58.72	60.57
High Precision Utterances	60.24	59.59	61.47	59.05	61.41

Table 6.1: Accuracy Scores of 5-Fold-Cross Validation of Trained Classifier.

One can see that the model is able to achieve an accuracy of over 50%. This ability is independent of the level of precision of the data, the accuracies hardly differ here.

If one looks at the confusion matrix of the models, i.e. the confusion matrix of the high precision level of fold 5 shown in Figure 6.19, the following insight emerges: Since the true label in the real game setting is Crewmate far more frequently due to the high imbalance in the data, one

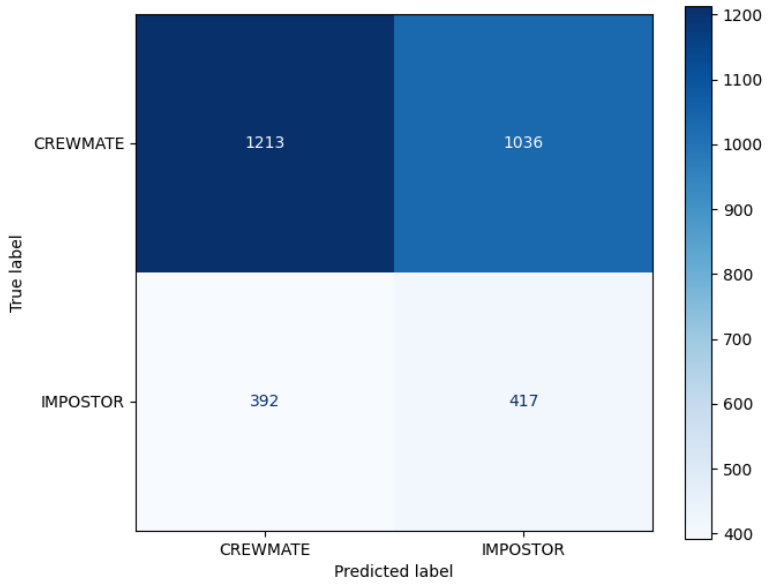


Figure 6.19: Confusion Matrix of Fold 5 With High Precision Utterances.

would perform better than the model in a real game if one always predicts Crewmate. This is because the model predicts both classes very evenly due to the oversampling of the minority class in the training data. To use the model profitably in a game, one would have to correct for this imbalance in the data. However, this is not the objective here, but to answer the question of whether the model recognizes dynamics in the text data that enable it to predict the role of the player better than random guessing. Since the model consistently achieves accuracies of well over 50%, this question can be answered positively. To explore the recognized patterns and dynamics, I apply an Explainable AI method for Feature Importance to the trained model in Chapter 6.2.3.

6.2.2 Application of Classifier on all Utterances of Discussion Rounds

Even if this is not the main focus of the work, I analyze how well the model would perform in its current form (without correcting for the oversampling in the training data) in the game setting. In the game, players must always vote at the end of each discussion round to eliminate suspicious players. They therefore have not just one, but all utterances made by a player in the discussion round available when making their decision about the player's role. To analyze this situation, I classify players with the model for their role by classifying all their utterances made in a discussion round and then making a majority vote on the roles predicted for the individual utterances. This procedure can be motivated in particular by the fact that many utterances made are unlikely to have any useful information content regarding the player role. However, if all utterances of a

discussion round are taken together, it is more likely that some of the utterances contain information about the player's role. For the classification of the individual utterances, I use the 5 trained models from the 5-fold cross-validation. It should be noted that when creating the 5 folds, I make sure that utterances of a streamer from the same discussion round are always in the same fold. This means that all utterances from the discussion rounds contained in the individual folds can now be used for evaluation without running the risk of data leakage. I carry out the evaluation of the majority vote method for all three degrees of precision; the accuracies achieved are shown in Table 6.2.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Low Accuracy Utterances	59.87	63.47	62.01	64.72	69.45
Low Accuracy Utterances (>3)	59.48	64.34	61.32	66.28	70.82
Medium Accuracy Utterances	56.10	52.70	50.81	58.94	66.03
Medium Accuracy Utterances (>3)	56.04	52.31	49.02	59.85	66.72
High Accuracy Utterances	66.24	70.23	64.12	66.24	57.34
High Accuracy Utterances (>3)	65.35	73.46	67.32	68.05	55.77

Table 6.2: Accuracy Scores of 5-Fold-Cross Validation of Trained Classifier Using the Majority Vote Method.

The method can achieve an improvement in accuracy for the low (on average 5.8%) and the high level of precision (on average 7.3%), while for the medium precision level the accuracy decreases on average by 4.7%. Based on the consideration that players can be classified particularly well if they have frequently said something in the discussions, I carry out the same analysis with only discussion rounds in which streamers have more than 3 utterances. The results are shown in Table 6.2 as well, but do not show any notable improvement over the majority vote method without filtering the discussion rounds.

As already mentioned, the methodology can be further optimized to enable a practical use of the classifier in the game setting. Three main approaches can be pursued for this: Firstly, the imbalance in the distribution of roles (far more Crewmates than Impostor) must be corrected for by weighting the model output accordingly. Secondly, the majority vote method can be used to place a special focus on important utterances in a discussion round. As already mentioned, many of the utterances made by streamers in the discussion round have little informative content about their role, while a few reveal information. These utterances could be identified using the model output, which also provides a certainty for each classification, and then weighted more heavily in the majority vote. If enough data is available, it is of course also feasible to train separate models for individual streamers. These would then be trained exclusively on the utterances of a single streamer and would probably be even better at capturing the individual language habits of the streamer and thus predicting their role. However, since this work is more concerned with the question of whether it is possible for the model to recognize behavioral patterns of the players than to optimize the model for usability as support for a successful AmongUs game, I omit these optimization steps here.

6.2.3 Feature Importance Analysis

As previously demonstrated, the performance of the classification model shows that the model recognizes patterns in the streamers' utterances that allow it to predict their role better than random guessing. By applying a feature importance method, I analyze how the model makes its decisions. The method I use is Integrated Gradients, which is a feature importance method used in text classification models to understand the contribution of each word or token to the model's prediction. This method assigns importance scores to individual words in a text input by calculating the gradient of the model's prediction with respect to the input text. By analyzing these importance scores, Integrated Gradients helps reveal which words or tokens have the most significant impact on the model's decision-making process, providing insights into the text features that influence the classification outcome. As a classification model, I analyze the model from fold 1 trained on utterances with a low precision level. As text input, I use all 26,461 extracted utterances with a low degree of precision. For each text input, a token attribution is calculated for each token for the respective prediction.

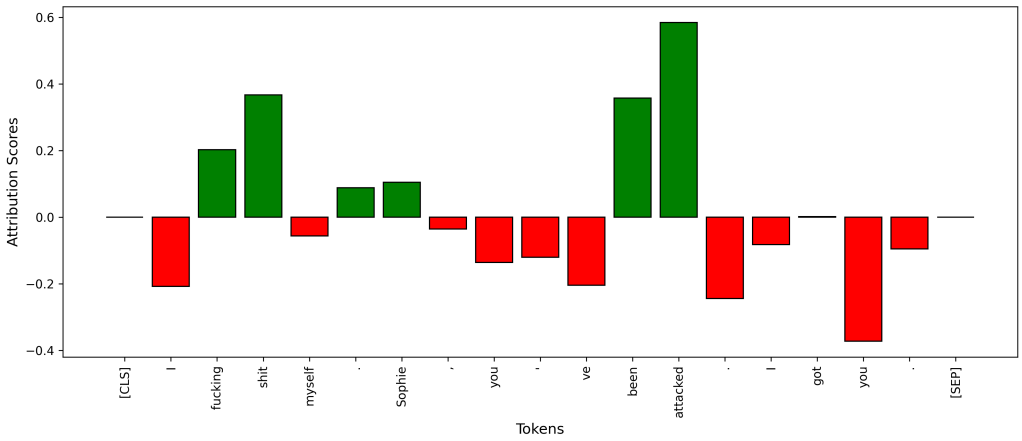


Figure 6.20: Feature Importance Scores for Example Utterance Classified as Impostor.

Figure 6.20 shows the result of such a feature importance analysis for an example utterance, which is classified as an Impostor. A positive Feature Importance Score means that the token contributes positively to the final classification of the model. The higher the feature importance score, the greater the influence of the token on the final decision of the model. In the utterance shown, for example, the token "attacked" in particular has a crucial influence on predicting the Impostor class. Talking about attacks on other player seems to convince the model that the speaker is an Impostor. The fact that the speaker reassures the attacked player of his support would have made the model more likely to predict Crewmate, but overall the features that cause a classification as Impostor dominate.

Across all utterances, I calculate the average feature attribution score for each token for both classes (Crewmate and Impostor). Figure 6.21 and 6.22 show the 20 tokens with the most positive and with the most negative average feature attribution score for the model classification of an

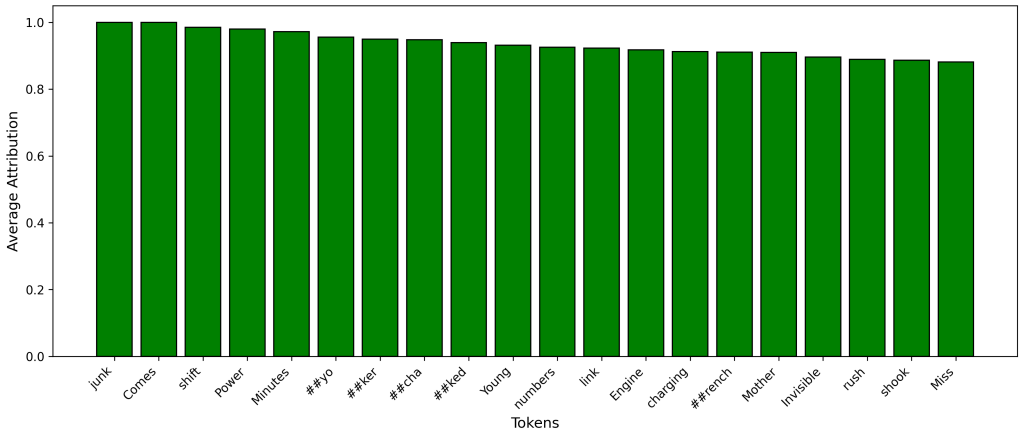


Figure 6.21: Top Tokens According to Average Feature Importance Score for Impostor Classification.

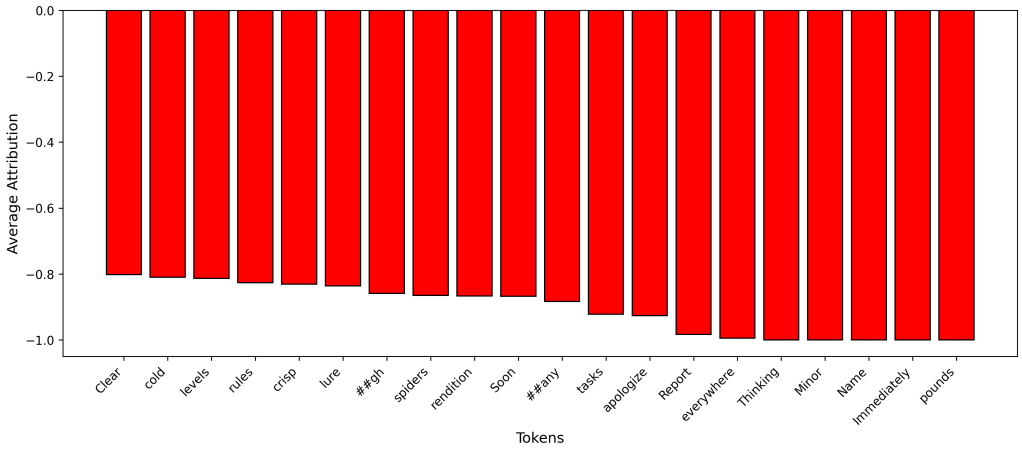


Figure 6.22: Bottom Tokens According to Average Feature Importance Score for Impostor Classification.

utterance as Impostor, the top and bottom 20 tokens for the classification as Crewmate are shown in the Appendix A.8 and A.9. As can be seen, the majority of these tokens seem rather arbitrary and do not allow any direct interpretation without context.

For some tokens, however, looking at the average feature attribution provides interesting insights even without context. Looking at the average feature attribution of the question mark token, for example, provides an indication of which role, according to the model, the asking of questions indicates. The fact that the average attribution of the question mark token to the prediction of a crewmate is negative at -0.11 could be explained by the fact that Impostor try to distract from

themselves by asking questions. According to the strategy "attack is the best defense", they try to suspect other players and put them in a position of having to explain themselves.

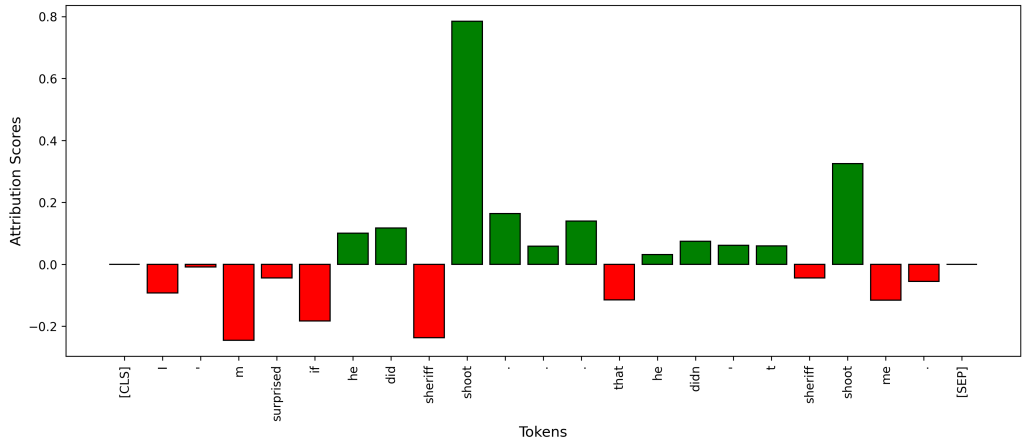


Figure 6.23: Meaningful Feature Importance Scores for Example Utterance Classified as Impostor.

However, it must be said that in the present case, it is primarily the analysis of individual utterances by Integrated Gradient that can lead to meaningful insights into the model decisions. For example, in the utterance shown in Figure 6.23, the method clearly shows that talking about a "shooting" moves the model to an Impostor classification.

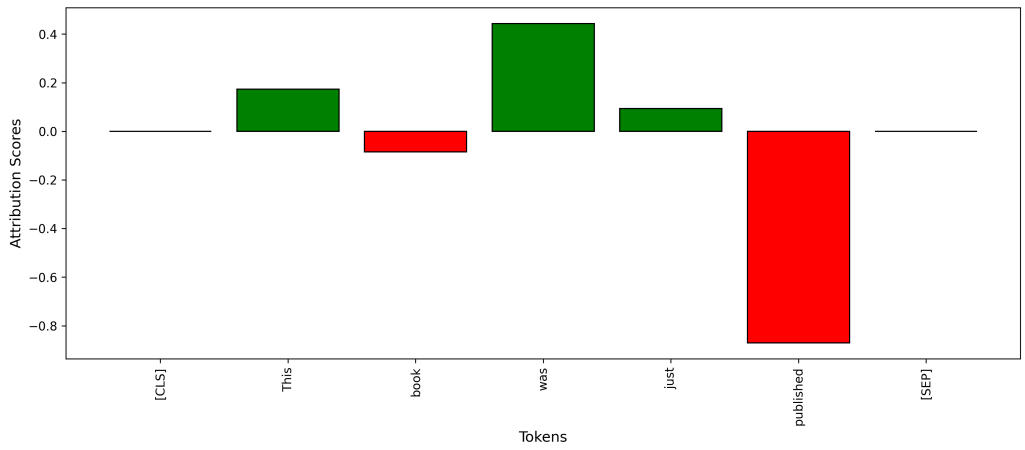


Figure 6.24: Feature Importance Scores for Example Utterance Without Information About Role.

In other cases, however, as in Figure 6.24, there are utterances that simply do not contain any real information about the player role, which is why an interpretation of the Feature Importance Score is meaningless.

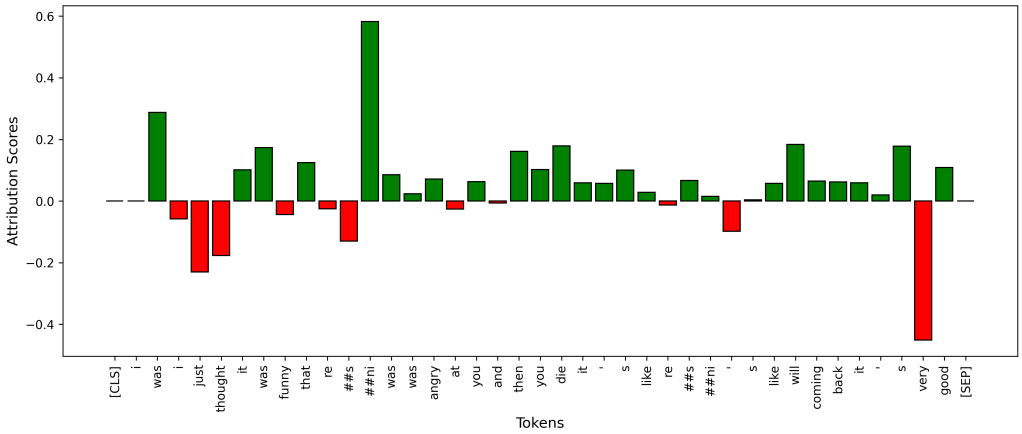


Figure 6.25: Surprising Feature Importance Scores for Example Utterance Classified as Crewmate.

In yet other cases, such as in Figure 6.25, which is classified as a Crewmate, the utterance seems to contain information about the role, but the model's focus on certain tokens such as "##ni" (part of the player's name Ressni) cannot be explained intuitively. In summary, one can say that no explanation for the model decisions on a general level can be found using feature importance methods that refer to tokens. However, interesting insights into the decision-making process of the model can be gained for individual utterances.

6.2.4 Exclusion of Possibility of Better Classifier Performance due to Dead Player Bias

Players can die in AmongUs in two ways: Either they are killed by an Imposter during the movement phase, or they are voted to death by other players at the end of a discussion round. This is why it can happen, especially in later discussion rounds within a lobby, that players who have already died take part in the discussion rounds. In this case, the dead players are generally not allowed to talk to the other players, but sometimes they do. In addition, they often talk to their chat in these situations and have muted the other players. In these two cases, the presented information extraction process extracts the utterances of these streamers, although the desired setting - the players talking to the other players while still having a secret role - is not given. This degrades the quality of the training data, which could probably lead to a deterioration in model performance when predicting roles based on utterances of alive players. However, this does not seem to pose a problem since the model is still able to recognize decisive patterns in the players' utterances to perform a role classification with a good performance. However, it is important to rule out the possibility that this good model performance is only driven by the utterances of dead players, and that the model cannot outperform random guessing when classifying utterances in the realistic setting - the utterance is addressed to other players and the true role of the speaker is not known yet. For this reason, I perform an evaluation in which I classify random utterances of already dead

players with the model and then compare the achieved performance with the known performance scores from the evaluation with all utterances.

Discussion Round	Player Dead
2022-02-16_S1_l10_5_ozzaworld	1
2022-05-24_S1_l3_5_courtilly	1
2022-03-03_S1_l13_4_irepptar	0
2022-05-24_S1_l11_4_irepptar	1
2022-05-24_S1_l11_4_jayfletcher88	1
2022-02-22_S1_l13_4_aribunnie	0
2022-02-08_S1_l13_4_zeroyalviking	0
2022-02-12_S1_l20_4_courtilly	1
2022-02-04_S1_l3_4_irepptar	0
2022-02-17_S1_l13_4_irepptar	0
2022-02-22_S1_l16_4_ayanehylo	0
2022-02-02_S1_l7_4_ozzaworld	0
2022-02-16_S1_l8_4_zeroyalviking	0
2022-05-24_S1_l3_4_irepptar	1
2022-02-15_S1_l10_4_jvckk	0
2022-02-16_S1_l8_4_irepptar	0
2022-05-24_S1_l3_4_jayfletcher88	0
2022-02-12_S1_l22_3_ozzaworld	
2022-05-24_S1_l2_3_irepptar	1
2022-01-28_S1_l7_3_ozzaworld	0
2022-03-10_S1_l5_3_skadj	
2022-02-08_S1_l12_3_skadj	0
2022-03-03_S1_l2_3_ozzaworld	1
2022-02-04_S1_l12_3_vikramafc	0
2022-02-05_S1_l9_3_ozzaworld	0
2022-02-09_S1_l14_3_ozzaworld	1
2022-02-04_S1_l3_3_irepptar	0
2022-02-16_S1_l8_3_ayanehylo	0
2022-02-22_S1_l8_3_vikramafc	0
2022-03-02_S1_l10_3_ozzaworld	0

Table 6.3: Evaluating the Liveliness of a Player for 30 Randomly Selected Discussion Rounds.

For this, I use the model from fold 1 trained for utterances with a low precision level. Therefore, only utterances from discussion rounds that were in the test set when this model was trained are used as test utterances. As the probability of being dead as a player increases with the duration of the game, I select the highest numbered discussion rounds. I select a total of 30 discussion rounds; these are always the third, fourth or even fifth discussion round in a lobby. For these discussion rounds, I then manually check whether the corresponding player is already dead at the time of the discussion round. The results of this evaluation are shown in Table 6.3. As one can

see, the streamer is already dead in 36% of the discussion rounds. This share is much lower in the complete data used for the model training, since here I explicitly filter on the highest numbered and thus latest discussion rounds of each lobby. This significantly increases the probability of a dead streamer, and in addition, the discussion rounds identified in this way come from lobbies that go on for an exceptionally long time and therefore have an above-average number of dead players. The proportion of utterances originating from dead players in the full analysis is therefore probably much smaller, which is why they are unlikely to have too drastic an impact on model performance.

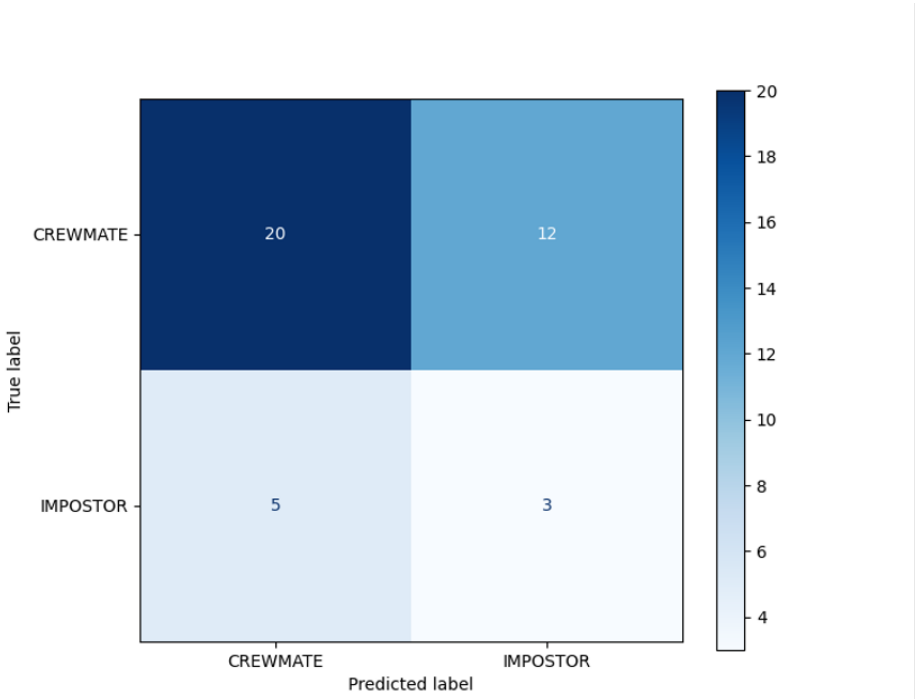


Figure 6.26: Confusion Matrix of all Utterances from 11 Discussion Rounds of Dead Players.

Nevertheless, I still test this by classifying all utterances of the 11 identified discussion rounds with players who have already died. I then look at the model performance, a confusion matrix is shown in Figure 6.26. Overall, an accuracy of 57.5% is achieved for the classifications, which is worse than the accuracy of all models of all folds and precision levels. Due to the small number of data points for this evaluation, it is not possible to generalize that the model performs worse for utterances of dead players. However, the results definitely do not indicate that the general model performance is significantly improved by such utterances, such that the model's ability to predict roles should be questioned.

7 Discussion and Future Work

In summary, the extraction of relevant data from the videos proves highly effective. By integrating multiple methods and applying heuristics, I successfully extract the necessary information for behavioral analysis from online gaming streams. The pipeline I present thus serves as a reliable guide for effectively extracting data from synchronous video streams. A key task in this process is identifying relevant timestamps in the streams, such as the time periods of lobbies and discussion rounds. For both, I combine various techniques: image recognition is used for lobby identification, and speaker diarization is employed for discussion rounds. The results are further refined using a set of heuristics. As can be seen by the evaluations, this approach yields excellent results, with identified lobbies and discussion rounds both being almost exclusively accurate and their identified timestamps almost always matching the actual timestamp of the event.

However, in both cases I still have to perform a manual examination of some special cases that cannot be identified by the programmatic extraction process. Fortunately, this manual examination does not represent a significant amount of work in the current use case. If the described procedure is to be applied in some form to a similar use case with far more data, the manual review process could become more time-consuming. In such scenarios, further optimization through additional heuristics could be explored to reduce the manual workload. That said, the programmatic approach already addresses a significant portion of the special cases, and many of these are simply not possible to cover programmaticaly (for example, cases in which streamers play a different game on their VODs while leaving AmongUs running in the background). It is therefore necessary to assess the extent to which further optimization of the programmatic extraction process is actually promising.

Despite the difficulties posed by the data, I achieve very good results for speech extraction. By leveraging a target speaker extraction model for speaker diarization, in the difficult setting of multi-speaker discussions, I achieve a level of performance that ensures data extraction of sufficient quality. The procedure for generating the training data and performing model training for the speaker diarization can be adopted for similar use cases for speech extraction from discussions with known speakers. If even better performance is required for these use cases, there are several starting points. One approach would be to better align the artificially generated training data with real-world conditions. In the current implementation, audio snippets from individual streamers are simply concatenated sequentially. To simulate more realistic interactions — such as speakers interrupting each other or talking simultaneously — these snippets could be overlapped. Additionally, the artificial conversations could be enhanced by introducing typical background noise. In the case of gaming streams, this could involve adding music or in-game sound effects to more closely resemble the actual environment.

However, not only speaker diarization alone, but also the combination of the results of speaker diarization with the transcriptions still has potential for improvement. The most obvious ap-

proach would be to use the synchronicity of the streams, i.e. the fact that a single discussion round was transcribed and segmented by speaker diarization several times for each participating streamer. When combining speaker diarization and transcription, the results of the individual streamers could be combined for a discussion round in order to find the most likely correct assignment using various heuristics. In addition, the consistency of the data could be further increased in this way. As already mentioned, this approach would require a more precise, but also much more time-consuming program to extract the audio streams of the individual discussion rounds for each streamer to ensure exact temporal synchronicity between the individual discussion rounds.

Another approach to improve the combination of speaker diarization and transcription would be to apply a semantic analysis of the transcription. In this way, related utterances that are likely to come from one speaker could be identified on a semantic basis. This could be used as additional information when assigning related utterances to individual speakers based on the output of the speaker diarization model.

A final, probably very promising approach to further improve speech extraction would be a multi-modal approach in which visual data is used in addition to audio data. Since the streamers can usually also be seen in a small window with their face in their streams, a more complex model could also evaluate this information in order to analyze whether a streamer is currently speaking based on the movement of their mouth, for example. In some online gaming streams, there is also a display that shows which other streamer is currently speaking by name (unfortunately, only very few streamers use this function in the analyzed AmongUs streams, which is why this was not possible here). The volume of an audio signal could also be used as an additional feature, as the streamers can usually be heard loudest on their own streams. All of these approaches presented would lead to further optimization of the performance of text extraction from the video data. However, this was not necessary in the present use case, as text extraction had already produced very good results that were of sufficient quality for further analysis.

Analyzing the data using popular text mining methods provides interesting descriptive insights into the language used in the discussions. For example, the analysis of POS tags in the utterances shows which word types are used in particular in the discussions. In a further step, these results could be compared with distributions of POS tags in other discussions, such as political debates. This could possibly reveal interesting differences in the sentence structure of the arguments. A further analysis could also examine the increased use of which POS tags leads to success in a discussion. In the present case, this would require identifying the winners for the individual lobbies or the players who are eliminated by votes after the individual discussion rounds in order to then examine the POS tags they use.

The descriptive analysis of word frequencies can also serve as a basis for interesting further investigations. Since the streamers analyzed here are mostly young people, the analysis provides a good picture of the language used by young people in the internet, especially in spoken discussions. The language used can, for example, be compared with language used in older videos (e.g. from other gaming streams) in order to discover potential trends and developments in the language used by young people. Once you have identified winners for the individual lobbies, you can also analyze the word frequencies to see whether there are certain "winning words" that contribute in particular to a successful outcome of discussions. Another starting point is to analyze in more detail why the results show that some of the streamers are mentioned very frequently and some are mentioned and probably suspected rather rarely in the discussions. The behavior of these stream-

ers could be analyzed in more detail in order to uncover which behaviors are more likely to lead to suspicion.

All three applied text mining methods - POS tagging, word frequency analysis, and sentiment analysis - are not able to contribute to a decent role classification based on their results alone. One of the likely explanations is that most of the streamers' utterances contain little information about their role. Many utterances are descriptive descriptions of gameplay and therefore do not differ significantly between roles. Based on a purely descriptive analysis of these utterances, a classification is therefore not possible. This result could also be interpreted to mean that the streamers of AmongUs analyzed here are now so skilled at hiding their role due to their enormous gaming experience that a simple frequency analysis of words, POS tags, or sentiment is no longer sufficient and more complex methods must be used.

The trained classification model is able to perform the classification between the roles with an accuracy that is far better than random guessing. This result suggests that the model recognizes certain dynamics in the streamers' utterances that reveal their identity. This result provides evidence that LLMs are able to extract hidden information from players' utterances. This motivates further analysis of the extracted data: For example, it could be investigated whether the trained model or other LLMs are able to solve more complex tasks with the data. It would be conceivable, for example, to expand the binary classification task and use not only the two main roles but also the more specific roles as labels. However, it would be necessary to check whether the amount of training data is then sufficient for a good model performance. One could also try to expand the model input to include additional components. For example, the model could also receive visual features that represent the facial expressions and gestures of the person speaking as input for its classification. These features would have to be extracted in a previous step, which is probably possible due to the camera settings of most of the streamers analyzed. Whether the quality of the data obtained in this way is actually good enough to capture the facial expressions and gestures of the person speaking in a meaningful way remains to be tested. Another conceivable feature would be the audio characteristics of the utterances. For example, a model can certainly recognize patterns in the utterances based on volume or voice pitch for better classification.

In addition to the currently implemented Integrated Gradient method, many other methods such as Local Interpretable Model-agnostic Explanations (LIME), SHapley Additive Explanations (SHAP), Attention Rollout and Gradient-based Class Activation Mapping (Grad-CAM) can be used in the Feature Importance analysis to understand how the model makes its decisions. In this way, it may be possible to better understand how the model makes its decisions and what patterns are recognized in the utterances. These recognized patterns can then be analyzed in more detail, as they may also provide fundamental insights into people's behavior in discussions, which can then be compared with findings from other social science disciplines. For example, if we assume that Impostors have to lie more often in their utterances because they cannot honestly report what they did in the movement phase without exposing themselves, we can look at which features in particular lead the model to predict a lie. The findings obtained in this way can then be compared, for example, with findings from psychology on classic human behavior when lying.

8 Conclusion

In this work, I analyze the behavior of players engaging in modified versions of the online social deduction game Among Us using LLMs. To perform the analysis, I extract relevant data from 363 VODs, each with an average duration of 3 hours and 5 minutes, using deep neural methods. To perform the analysis, I first identify crucial timestamps, such as lobby and discussion phases, using image recognition and speech extraction methods. Based on these timestamps, I am able to extract important information, including player roles and transcribed utterances. As part of this process, I also train a target speaker extraction model which allows for accurate speaker diarization, even in the complex acoustic environment of multiplayer streams.

Despite the challenges of working with data from online multiplayer gaming streams, the information extraction pipeline proves highly effective in generating high-quality data for subsequent analysis of the player behavior. This framework, combining image recognition and advanced speech extraction methods, can thus serve as a guide for future research on synchronous video streams.

The resulting data set includes transcribed utterances for each player and their corresponding secret game roles. Through NLP methods, I conduct descriptive analyses and apply classification models to explore the language patterns that could reveal player roles. While simpler methods fail to achieve high accuracy, a fine-tuned BERT model for text classification shows promising results, identifying patterns in the utterances with an accuracy of up to 60%, which improves when multiple utterances are combined.

The application of Explainable AI, particularly Integrated Gradient, provides valuable insights into the model's decision-making process, highlighting the importance of certain input tokens. These findings motivate to further explore the potential of LLMs in analyzing player behavior, as well as understanding human behavior in broader contexts through explainable models.

Bibliography

Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview, 2021. URL https://arxiv.org/abs/2012.00931.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, and Markus Zijlstra. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378, 11 2022. doi: 10.1126/science.ade9097.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 01 2009. ISBN 978-0-596-51649-9.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. 3(null): 993–1022, mar 2003. ISSN 1532-4435.

Nicolo' Brandizzi, Davide Grossi, and Luca Iocchi. Rlupus: Cooperation through emergent communication in the werewolf social deduction game, 2021. URL https://arxiv.org/abs/2106.05018.

Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *Proc. INTERSPEECH 2023*, 2023.

Yizhou Chi, Lingjun Mao, and Zineng Tang. Amongagents: Evaluating large language models in the interactive text-based social deduction game, 2024. URL https://arxiv.org/abs/2407.16521.

Gokul Chittaranjan and Hayley Hung. Are you a werewolf? detecting deceptive roles and outcomes in a conversational role-playing game. page 5334–5337, 2010. doi: 10.1109/ICASSP.2010.5494961. URL https://infoscience.epfl.ch/handle/20.500.14299/55964.

Robert Chuchro. Training an assassin ai for the resistance: Avalon, 2022. URL https://arxiv.org/abs/2209.09331.

Ying-Kai Dang. Information extraction from synchronous multiplayer gaming streams. Bachelor's thesis, University of Konstanz, 2022.

Marc Delcroix, Katerina Zmolikova, Keisuke Kinoshita, Atsunori Ogawa, and Tomohiro Nakatani. Single channel target speaker extraction and recognition with speaker beam. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5554–5558, 2018. doi: 10.1109/ICASSP.2018.8462661.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Shaojin Ding, Quan Wang, Shuo yiin Chang, Li Wan, and Ignacio Lopez Moreno. Personal vad: Speaker-conditioned voice activity detection, 2020.

Cathy Mengying Fang, G. R. Marvez, Neska ElHaouij, and Rosalind Picard. Cardiac arrest: Evaluating the role of biosignals in gameplay strategies and players' physiological synchrony in social deception games. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391566. doi: 10.1145/3491101.3519670. URL https://doi.org/10.1145/3491101.3519670.

ffmpeg. FFmpeg. https://ffmpeg.org, 2024. Accessed: 2024-09-09.

Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-end neural speaker diarization with self-attention, 2019.

Bakkali Hamza, Yousfi Abdellah, Gueddah Hicham, and Belkasmi Mostafa. For an independent spell-checking system from the arabic language vocabulary. *International Journal of Advanced Computer Science and Applications*, 5(1), 2014. doi: 10.14569/IJACSA.2014.050115. URL http://dx.doi.org/10.14569/IJACSA.2014.050115.

Naoyuki Kanda, Shota Horiguchi, Ryoichi Takashima, Yusuke Fujita, Kenji Nagamatsu, and Shinji Watanabe. Auxiliary interference speaker loss for target-speaker speech recognition, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Kavya Kopparapu, Edgar A. Duéñez-Guzmán, Jayd Matyas, Alexander Sasha Vezhnevets, John P. Agapiou, Kevin R. McKee, Richard Everett, Janusz Marecki, Joel Z. Leibo, and Thore Graepel. Hidden agenda: a social deduction game with diverse learned equilibria, 2022. URL https://arxiv.org/abs/2201.01816.

Bolin Lai, Hongxin Zhang, Miao Liu, Aryan Pariani, Fiona Ryan, Wenqi Jia, Shirley Anugrah Hayati, James M. Rehg, and Diyi Yang. Werewolf among us: A multimodal dataset for modeling persuasion behaviors in social deduction games, 2022. URL https://arxiv.org/abs/2212.08279.

Joshua Lawrence, Rebecca Knoph, Autumn McIlraith, Paulina A Kulesz, and David Francis. Reading comprehension and academic vocabulary: Exploring relations of item features and reading proficiency. *Reading Research Quarterly*, 57, 07 2021. doi: 10.1002/rrq.434.

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. Timelms: Diachronic language models from twitter, 2022. URL https://arxiv.org/abs/2202.03829.

Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models, 2024. URL https://arxiv.org/abs/2301.06627.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004.

Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015. doi: 10.25080/Majora-7b98e3ed-003.

Ivan Medennikov, Maxim Korenevsky, Tatiana Prisyach, Yuri Khokhlov, Mariya Korenevskaya, Ivan Sorokin, Tatiana Timofeeva, Anton Mitrofanov, Andrei Andrusenko, Ivan Podluzhny, Aleksandr Laptev, and Aleksei Romanenko. Target-speaker voice activity detection: A novel approach for multi-speaker diarization in a dinner party scenario. In *Interspeech 2020*, interspeech 2020. ISCA, October 2020. doi: 10.21437/interspeech.2020-1602. URL http://dx.doi.org/10.21437/Interspeech.2020-1602.

Ircham Aji Nugroho, B. H. Susanti, Mareta Wahyu Ardyani, and Nadia Paramita R.A. The design of a c1 document data extraction application using a tesseract-optical character recognition engine. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 2024. URL https://api.semanticscholar.org/CorpusID:269285308.

Vinod Pande and Vijay Kale. *Speakers Identification Using Diarization Techniques*, pages 905–915. 05 2023. ISBN 978-94-6463-135-7. doi: 10.2991/978-94-6463-136-4_80.

Chirag Patel, Atul Patel, and Dharmendra Patel. Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, 55:50–56, 10 2012. doi: 10.5120/8794-2784.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Vesel. The kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 01 2011.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

Simon Sedláček. Personal voice activity detection. Bachelor's thesis, Brno University of Technology, 2021.

Jack Serrino, Max Kleiman-Weiner, David C. Parkes, and Joshua B. Tenenbaum. Finding friend and foe in multi-agent games, 2019. URL https://arxiv.org/abs/1906.02330.

Peter Allen Stubberud, Junichi Kanai, and Venugopal Kalluri. Improving optical character recognition accuracy using adaptive image restoration. *J. Electronic Imaging*, 5:379–387, 1996. URL https://api.semanticscholar.org/CorpusID:40100090.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020. URL https://arxiv.org/abs/1905.05583.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL https://arxiv.org/abs/1703.01365.

Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification, 2020.

Quan Wang, Hannah Muckenhirn, Kevin Wilson, Prashant Sridhar, Zelin Wu, John Hershey, Rif A. Saurous, Ron J. Weiss, Ye Jia, and Ignacio Lopez Moreno. Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking, 2019.

Shenzhi Wang, Chang Liu, Zilong Zheng, Siyuan Qi, Shuo Chen, Qisen Yang, Andrew Zhao, Chaofei Wang, Shiji Song, and Gao Huang. Avalon's game of thoughts: Battle against deception through recursive contemplation, 2023. URL https://arxiv.org/abs/2310.01320.

Shinji Watanabe, Michael Mandel, Jon Barker, Emmanuel Vincent, Ashish Arora, Xuankai Chang, Sanjeev Khudanpur, Vimal Manohar, Daniel Povey, Desh Raj, David Snyder, Aswin Shanmugam Subramanian, Jan Trmal, Bar Ben Yair, Christoph Boeddeker, Zhaoheng Ni, Yusuke Fujita, Shota Horiguchi, Naoyuki Kanda, Takuya Yoshioka, and Neville Ryant. Chime-6 challenge:tackling multispeaker speech recognition for unsegmented recordings, 2020.

Sarah Wiseman and Kevin Lewis. What data do players rely on in social deduction games? In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, CHI PLAY '19 Extended Abstracts, page 781–787, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368711. doi: 10.1145/3341215.3356272. URL https://doi.org/10.1145/3341215.3356272.

Shuang Wu, Liwen Zhu, Tao Yang, Shiwei Xu, Qiang Fu, Yang Wei, and Haobo Fu. Enhance reasoning for large language models in the game werewolf, 2024. URL https://arxiv.org/abs/2402.02330.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf, 2024a. URL https://arxiv.org/abs/2309.04658.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. Language agents with reinforcement learning for strategic play in the werewolf game, 2024b. URL https://arxiv.org/abs/2310.18940.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 1445–1456, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351. doi: 10.1145/2488388.2488514. URL https://doi.org/10.1145/2488388.2488514.

Janez Žibert and France Mihelic. *Novel Approaches to Speaker Clustering for Speaker Diarization in Audio Broadcast News Data*. 11 2008. ISBN 978-953-7619-29-9. doi: 10.5772/6386.

Kateřina Žmolíková, Marc Delcroix, Keisuke Kinoshita, Takuya Higuchi, Atsunori Ogawa, and Tomohiro Nakatani. Speaker-aware neural network based beamformer for speaker extraction in speech mixtures. pages 2655–2659, 08 2017. doi: 10.21437/Interspeech.2017-667.

A Appendix

A.1 Description of AmongUs and Data

Game	Count
TheOtherRoles v3.3.3	3
TheOtherRoles v3.4.0	2
TownOfUs v2.5.0 dev6	2
TheOtherRoles v3.4.2	3
TownOfUs v2.5.0 dev12	2
TownOfUs v2.5.0 dev14	4
TownOfUs v2.5.1s	4
TheOtherRoles v3.4.3	1
TownOfUs v2.6.0s	4
TownOfUs v2.6.1s	5
TheOtherRoles v3.4.4	1
TownOfUs v2.6.4s	5
TheOtherRoles v4.1.3	1
TownOfUs v3.0.1s	2
TownOfUs v3.1.0s	2
TownOfUs	**30**
TheOtherRoles	**11**

Table A.1: Played Modifications of AmongUs in Different Game Sessions.

A.2 Methodology

Streamer Name	In Game Name
uneasypeasy	Uneasy, Unevsy
irepptar	Rep
br00d	br00d, Ovalbrood, Cvalbrood, Ovalbed
aribunnie	AriBunnie
skadj	Skadj, SKvdj
vikramafc	VikramAFC
	PJONK
karacorvus	Kara, Kvrv
x33n	X33N
ozzaworld	Ozza, Ozzv
cheesybluenips	Cheesy
jvckk	jvckk
zeroyalviking	Ze
willyutv	Will
junkyard	Junk
pwuppygf	pwuppygf
paulleewhirl	
	Casper
reenyy	Reenyy
	SquatchyLS
	Dunrunnin
	Atlas
	NerdOut
	Chilled
kaywordley	Kay
tenmamaemi	TenmaMaemi
dooleynotedgaming	Jeremy
pastaroniravioli	Pasta, Pastaroni
falcone	Falcone
	Sparkly
therealshab	Shab
	Karasper
atla5_w1nz	ATLA5
ayanehylo	Hylo
	Dun
brizzynewindsong	Brizzyne
	sophietx
ressnie	Ressnie, Rvssniv
	Elswick
	WOLFY
	Rawbuhbuh
	Nortski
	ZERO
jojosolos	jvjvsvlvs
	sophimane
chey	Chey
taydertot	TayderTot
aplatypuss	APIatypus
kruzadar	Kruzadar
dumbdog	DumbDog
kyr_sp33dy	Speedy
sidearms4reason	SideArms
itsdanpizza	Dan Pizza
heckmuffins	HeckMufins
jayfletcher88	Jay
	Noor
	crunchy
hcjustin	HCJustin
paulleewhirl	Paullee, PaulLee
courtilly	Covrtilly
	Juggernaut

Table A.2: List of In-Game-Names Assignment to Streamers

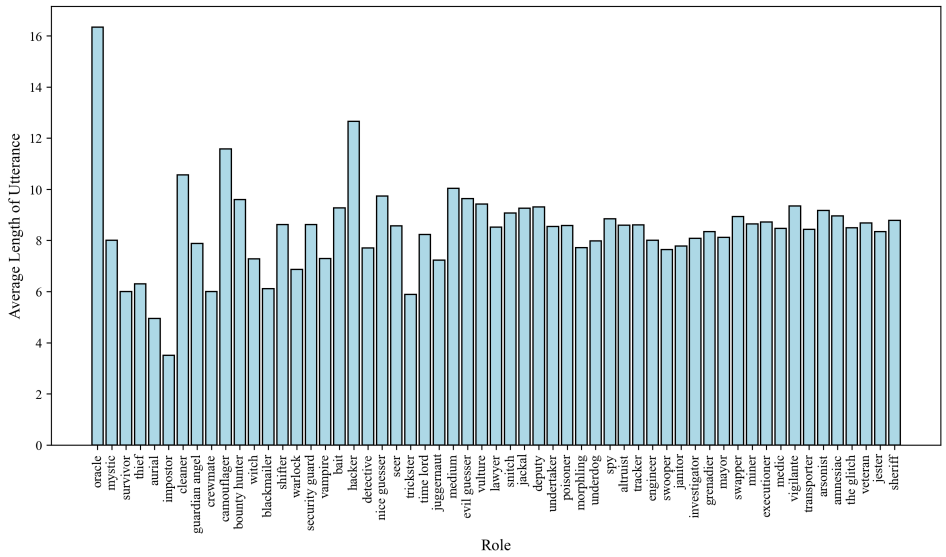


Figure A.1: Average Length in Seconds of Extracted Utterances by Role (Medium Precision).

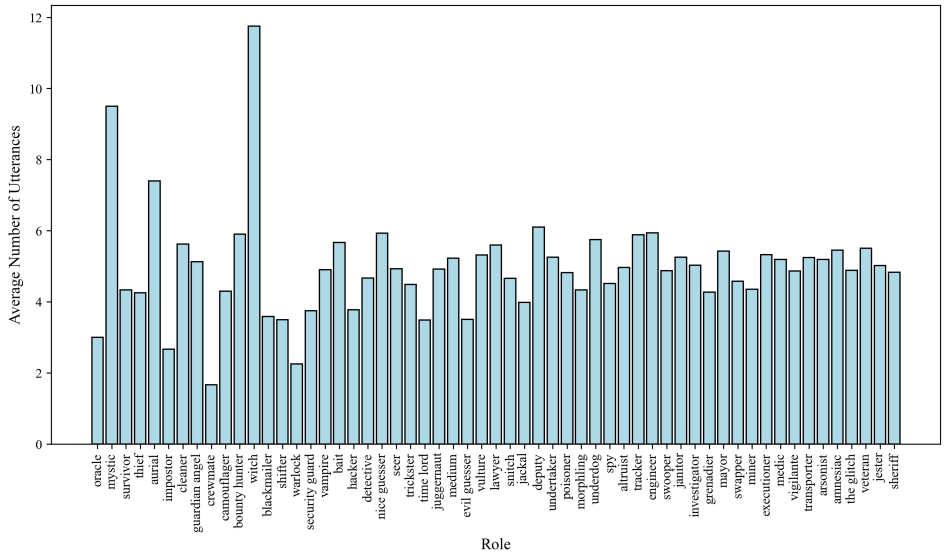


Figure A.2: Average Number of Extracted Utterances by Role (Medium Precision).

A.3 Language Analysis of Players' Utterances

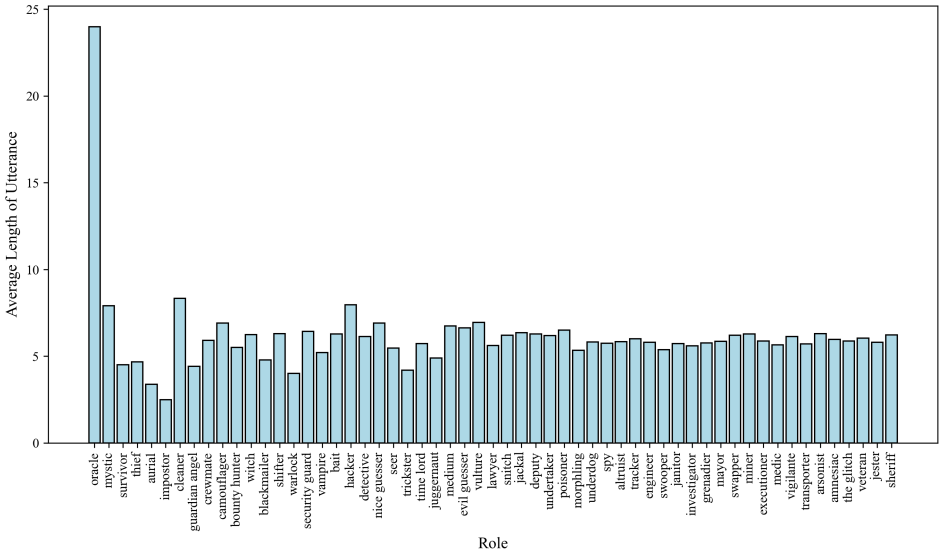


Figure A.3: Average Length in Seconds of Extracted Utterances by Role (High Precision).

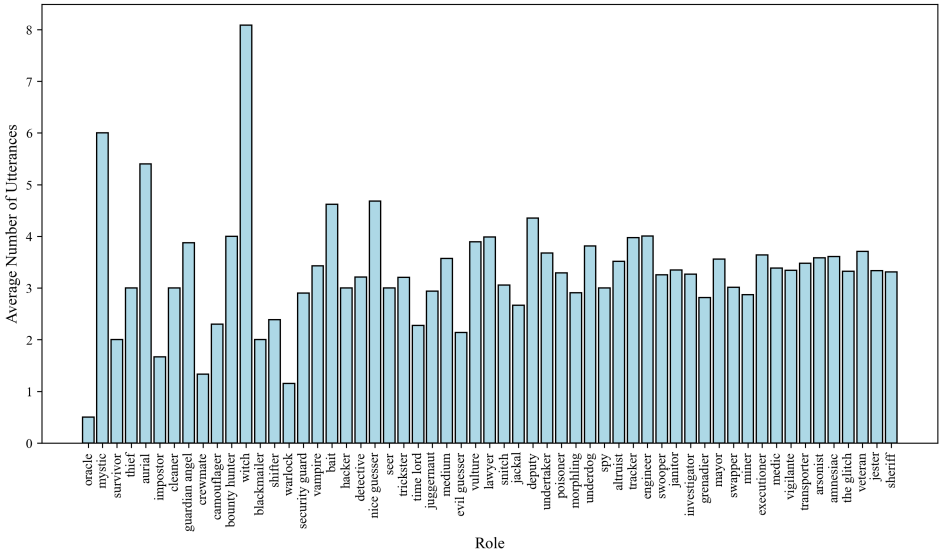


Figure A.4: Average Number of Extracted Utterances by Role (High Precision).

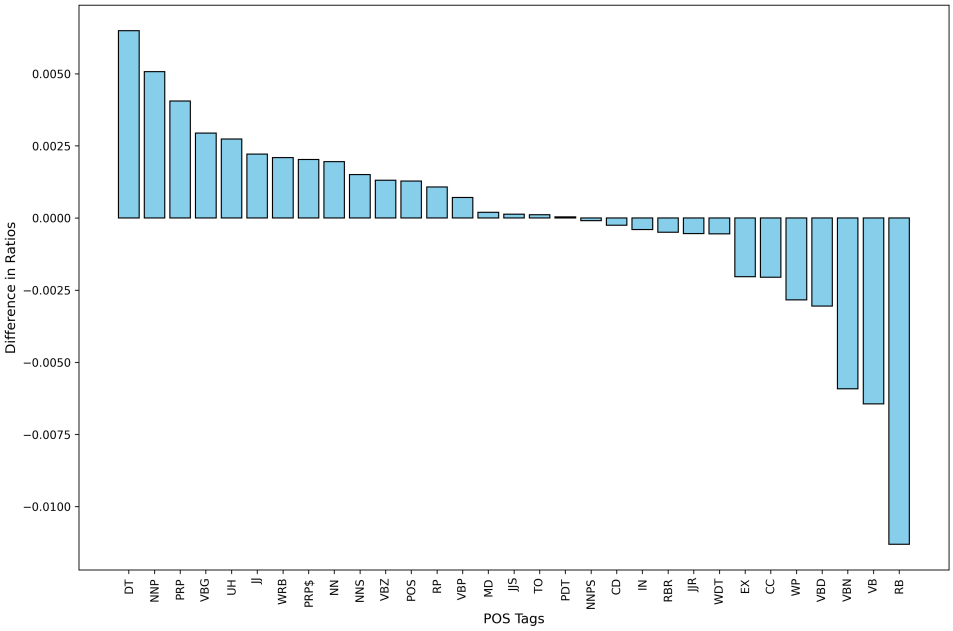


Figure A.5: Difference of POS Tags Between Crewmate and Neutral for Zeroyalviking.

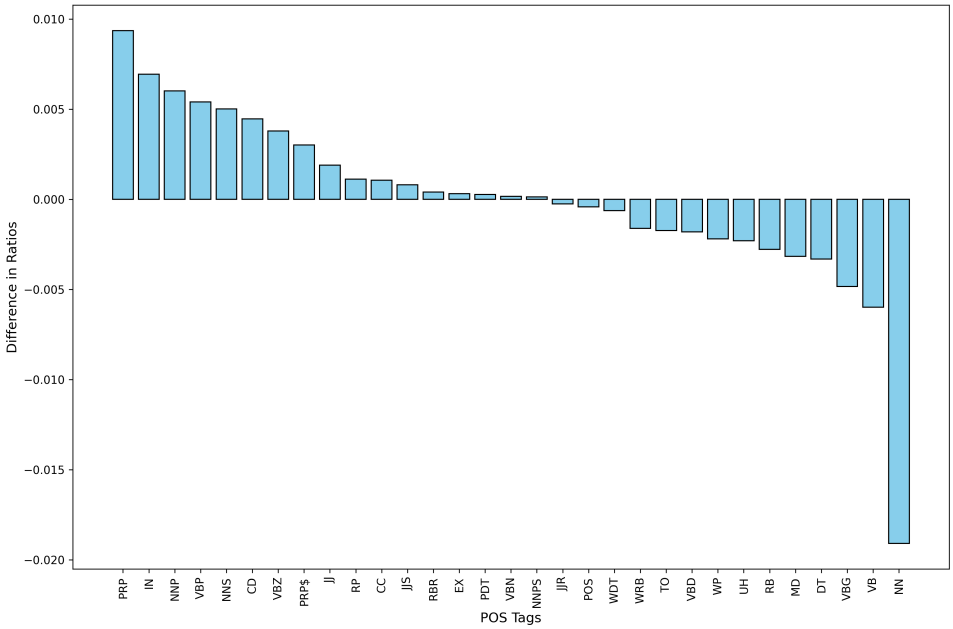


Figure A.6: Difference of POS Tags Between Crewmate and Impostor for Zeroyalviking.

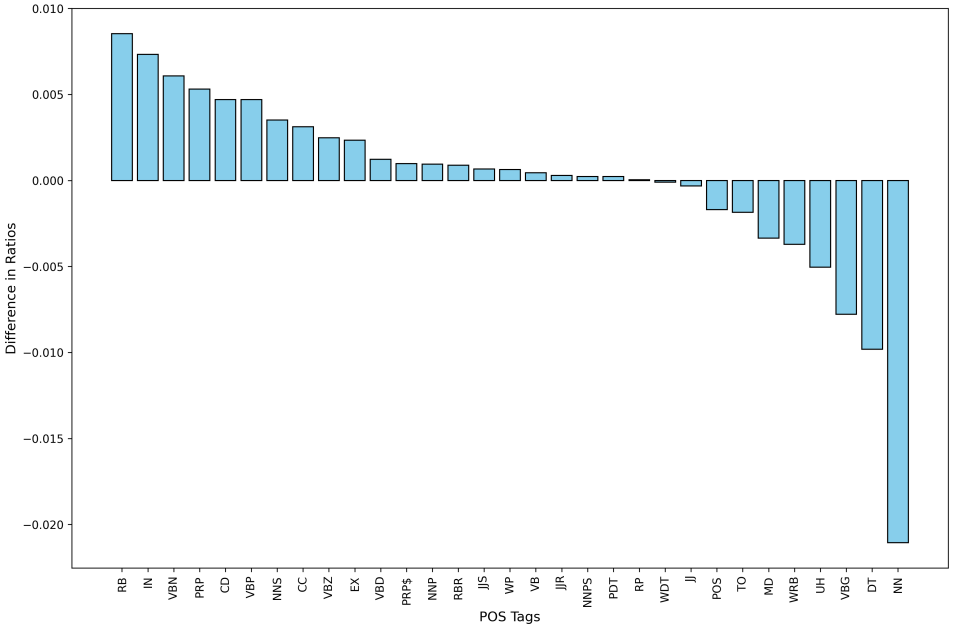


Figure A.7: Difference of POS Tags Between Neutral and Impostor for Zeroyalviking.

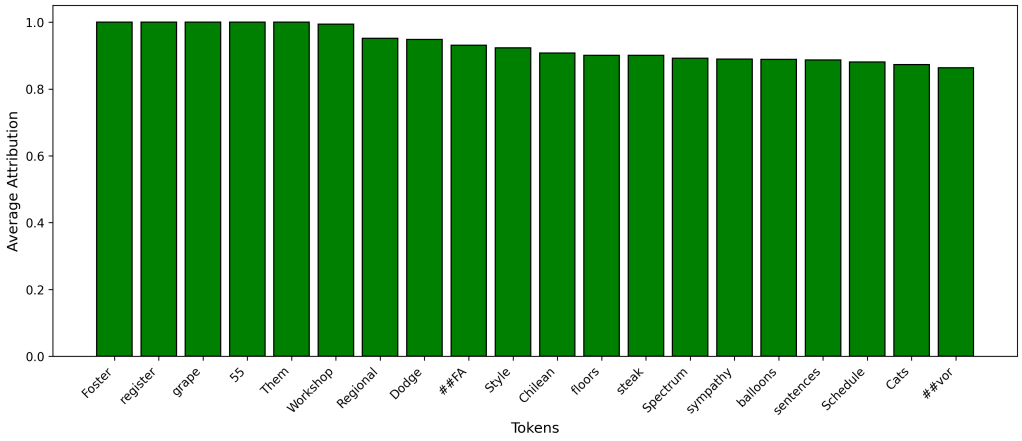


Figure A.8: Top Tokens According to Average Feature Importance Score for Crewmate Classification.

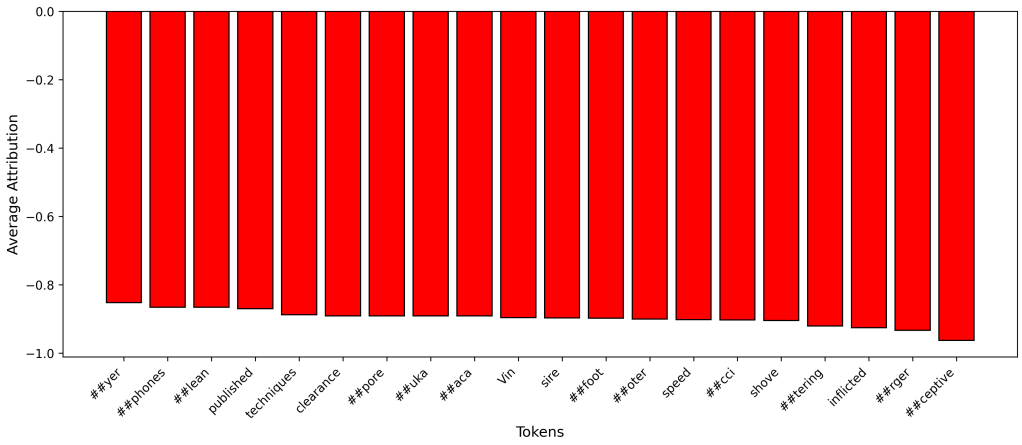


Figure A.9: Bottom Tokens According to Average Feature Importance Score for Crewmate Classification.

Please prepare this declaration in duplicate. Submit a copy of this declaration with your final thesis (but do not attach).

Universität Konstanz

Department of Politics and Public Administration

Declaration of Authorship

Last name, first name	Student ID number
Kleinlein Tim	01/11512 77

Title of master's thesis:
Deep Neural Extraction and Analysis of Linguistic Cues from Synchronous Multiplayer Social Deduction Gaming Streams

First reviewer:
Jun.-Prof. Dr. Andreas Spitz

I. I hereby declare that the above-mentioned master's thesis has been composed entirely by myself and that I have used the stated resources[1] only. This means especially that **any content** (such as texts, illustrations, data, models, ideas, hypotheses, arguments; even if these are single sentences or partial sentences) **contributed by other persons and which I have used is individually and explicitly referenced. To that end, I have included comprehensive and precise citations identifying the actual source used. Extracts from other texts are referenced immediately after they appear in the text** (please note that a reference, e.g. at the end of a paragraph that also includes your own text or passages taken from other sources, does not suffice!). I have clearly identified the use of all sources, irrespective of type (such as journal essay, internet page, book, conference paper, primary source, secondary literature) and extent, i.e. regardless of whether I have used the content in question in whole or in part, edited or unedited, verbatim, in summary or in translation. If I have used generative AI tools as aids, I understand that I am solely responsible for the accuracy of the content of the AI-generated passages, as well as for referencing other people's wording and ideas in accordance with the principles of good scientific practice. In the case of an adoption of AI-generated passages that go beyond purely formal, stylistic corrections and/or translations, I have marked them and, in addition to the AI tool used, I have indicated the prompts I entered. This serves to clearly identify my own work vis-à-vis that of others.
I further declare that the enclosed thesis has not been submitted before as a bachelor's and/or master's thesis or as a record of academic achievement for any other course either in whole or in part. If parts of my thesis were submitted before, I hereby declare that I have identified the passages/illustrations/data etc. in question as self-quotations by including a comprehensive and precise citation of the exact record of academic achievement as described above.
I am aware of the following:

- I am guilty of **plagiarism** if, contrary to the requirements set out above, I use the work of others in my thesis without acknowledging the precise source. This would be to lay claim to an achievement that is not my own.
- If a candidate intentionally violates the University of Konstanz's Statutes to Ensure Good Scientific Practice when composing his/her thesis, it will be, in all probability, judged as an attempt at cheating (**plagiarism**). This means that the thesis will be graded "nicht ausreichend/fail" (5,0), placed on record in the student's examination file, and put before the Examination Board (StPA).
- In repeated or particularly serious cases, the StPA may decide to exclude the candidate from further performance assessments, resulting in a complete loss of right to take any further examinations in this study programme.
- The legal basis for this is laid down in the applicable examination regulations for the study programme in question.

I have taken note of these guidelines regarding plagiarism.

II. I also submit an unprotected **electronic version** of my final thesis via e-mail.

III. After the examination process has been completed, the work will be submitted to the University of Konstanz and catalogued. It will thus be available to the public through viewing and lending. The collected descriptive data such as author, title, etc. will be publicly available.
As author of the respective work:

☒ I consent to this procedure.
☐ I do not consent to this procedure.

Konstanz, 17.09.24
Place, date

Tim Kleinlein
Student's signature

[1] Including the use of artificial intelligence (AI) tools.