UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE
POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Pyannote, Whisper și LM: Cavaleri ai Speaker Diarization

Deonise Costin-Alexandru

Coordonator științific:

Prof. dr. ing. Florin Pop

BUCUREȘTI
2024

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY
POLITEHNICA BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

Computer Science & Engineering Department

DIPLOMA PROJECT

Pyannote, Whisper and LM: Knights of Speaker Diarization

Deonise Costin-Alexandru

Thesis advisor:

Prof. dr. ing. Florin Pop

BUCHAREST
2024

CONTENTS

ABSTRACT

This paper presents a pipeline for enhancing Speaker Diarization through integrated state-of-the-art models: Pyannote for Speaker Segmentations, Whisper for Speech-to-Text (STT), and a Large Language Model (LLM) for improving both word-level accuracy and speaker-level segmentation. The pipeline begins with Pyannote that segments the audio into chunks, which are then processed by Whisper to generate text for each speaker segment. These transcriptions undergo refinement using an LLM, aimed at improving diarization accuracy and transcript readability. There were a couple of LLMs that were interrogating, both of them are used frequently by humans. The model should be used in meetings or calls, just like the dataset that was used, CALLHOME American English Speech. GPT-4 was the best contestant for reducing the cpWER (0.72%), but overall GPT-3.5 Turbo managed to get better results in diarization tasks.

Acest articol prezintă o metodă pentru îmbunătățirea Speaker Diarization prin integrarea unor modele de ultimă generație: Pyannote pentru Segmentarea Vorbitorilor, Whisper pentru Conversia Vorbirii în Text (STT), și un Model de Limbă (LLM) pentru îmbunătățirea preciziei, la nivel de cuvânt, și segmentare, la nivel de vorbitor. Pipeline-ul începe cu Pyannote, care segmentează fișierul audio în bucăți, acestea fiind apoi procesate de Whisper, pentru generarea de text aferent fiecărui speaker. Transcrierile obținute sunt rafinate folosind un LLM, cu scopul de a îmbunătăți precizia diarizării și lizibilitatea transcrierilor. Au fost utilizate două LLM-uri frecvent interogate de oameni. Modelul este destinat utilizării în întâlniri sau apeluri, similar cu setul de date utilizat, CALLHOME American English Speech. GPT-4 a fost cel mai bun pentru reducerea cpWER (0.72%), dar, per total, GPT-3.5 Turbo a reușit să obțină rezultate mai bune în cerințele de diarizare.

1 INTRODUCTION

This thesis shows that Speaker Diarization - the process of segmenting an audio recording into speaker-specific segments, which is one of the most complex tasks for the transcription task, is based on combining state-of-the-art processes that have already been proven to perform well individually: Pyannote for Speaker Segmentation [1], [2], Whisper for Automatic Speech Recognition (ASR) [3] and different Language Models (LM) [4], [5] for interacting with text. There are several metrics that are used in order to prove that this method can perform very well in different scenarios. For the background, there were some serious engineering challenges along the way: the output of Pyannote was used to split the audio file into chunks, where Whisper performed ASR, and in the end, ChatGPT-3.5 Turbo and ChatGPT-4 closed the experiment. For the dataset, the CALLHOME English dataset [6] is used, which consists of almost 12 hours of 140 conversations.

1.1 Context and Motivation

Speaker Diarization is commonly known for "who spoke when" [7], [8] and has a lot of usage in some important tasks: military operations, like when you have captured sound from a discussion of two or more people and you do not have the time to listen exactly to what a specific person said; it can serve as assistance to psychologists after a meeting to extract insights about the person, or can be used in situations for sales departments or marketing teams for one-to-one phone calls. There are various applications like meeting transcription, audio indexing, and surveillance. There are already some state-of-the-art tasks like ASR for extracting the text or Speaker Segmentation for giving timestamps for each speaker, but these have to be combined in order to extract meaningful information from meetings or calls. This can also be improved with the capabilities of an LM because the trimming of the speakers' labels cannot be precise or the ASR model can be generative and can attribute some words to the wrong speaker.

There are several models out there that are not so consistent on different datasets, so this thesis managed to combine state-of-the-art models and on top of those to post-process the results with two different LMs. Pyannote is used with version 3.1 (last update: 20th June 2024) and Whisper with model large-v3 (last update: 17th November 2023). There is not enough information to be known only for "who spoke when" without a transcript, so the text should rise to the performance. The post-processing method is based on two of the most used LMs at this moment: ChatGPT-3.5 Turbo and ChatGPT-4.

1.2 Problem Definition

Text extraction and speaker diarization are mostly hampered by the datasets and corpora that the system has been built with. This paper [9] discusses the Broadcast News (BN) domain, where speech is recorded in two locations: a quiet studio and a noisy field. The recordings are done using boom or lapel microphones. On the other hand, desktop, telephone, or single microphones are typically used to record meetings since they are more user-friendly than head-mounted or lapel microphones. Consequently, BN data often have a higher signal-to-noise ratio than meeting recordings. Furthermore, variations in recording quality might result from variations in meeting room layouts and microphone locations, such as background noise, reverberation, and varied speech volumes (depending on the distance between speakers and microphones) or speakers overlapping.

One of the trickiest parts of speaker diarization is managing overlapping speakers. Several recent studies produced positive findings for this task. By employing a novel Two-stage OverLap-aware Diarization framework (TOLD) that involves a speaker overlap-aware post-processing (SOAP) model to iteratively refine the diarization results of EEND-OLA [10] , this one was able to achieve a 14.39% relative improvement in terms of Diarization Error Rates (DER). Thus far, the most successful method has been end-to-end neural diarization using an attractor based on encoder-decoder [11], which is also effective for an unspecified number of speakers.

Handling the number of speakers and the language that is present in a discussion are some of the most challenging problems in research.

1.3 Thesis Objectives

The primary objective of this thesis is to enhance the performance of Speaker Diarization and text extraction by improving the following metrics:

- Diarization Purity.
- Diarization Coverage (for the text).
- Word Diarization Error Rate (WDER).
- Concatenated Minimum-Permutation Word Error Rate (cpWER).

To achieve these goals, several challenges needed to be addressed:

- **Audio Reconstruction**: The CALLHOME English dataset provided audio data split into numerous segments. This necessitated an engineering solution to reconstruct the audio into a coherent "wav" format.
- **Audio Chunking**: Trimming the audio files into appropriate chunks for ASR was crucial for accurate processing.

- **Prompt Generation for Large Language Model (LLM)**: Ensuring high-quality prompts for the LLM was essential for contextually accurate diarization and transcription.

Using Whisper, a generative model, presents a significant problem because of its propensity to have hallucinations when speaker segmentation is not ideal. As a result, the words of the next speaker may be mistakenly identified with the present one.

Processing a single audio file also takes a long time since it involves a lot of calculations. The LLM and Whisper require the most time for processing. The processing time for a 5-minute discussion still averages about 3.5 minutes, even with the notably accelerated NVIDIA GeForce RTX 4090.

Despite Pyannote's segmentation, Figure 1 shows how Whisper sometimes generates context on its own because of brief discussion pauses. The suggested approach makes use of contextual data from the LLM in an attempt to remedy this.

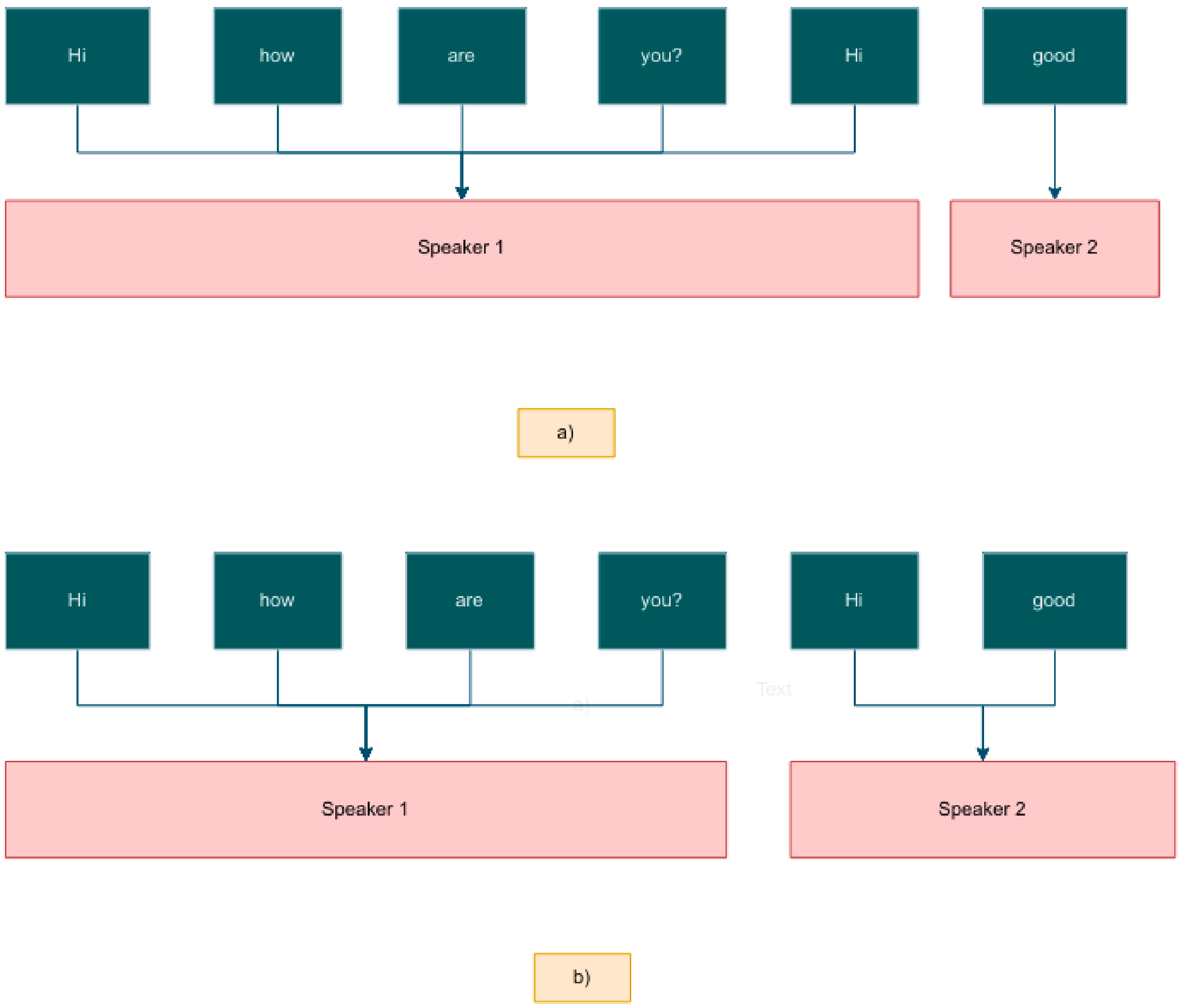


Figure 1: Illustration of Whisper's tendency to generate context independently (a) and the proposed method's correction using LLM context (b).

2 BACKGROUND AND PREVIOUS SOLUTIONS

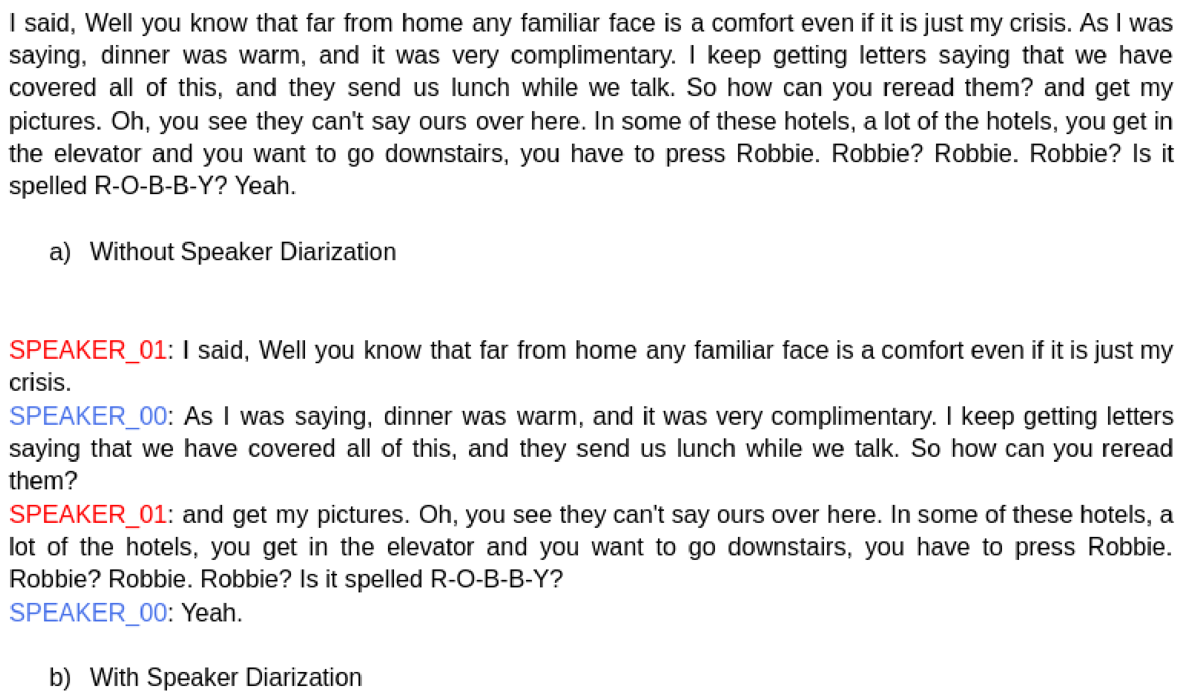
2.1 Speaker Diarization: An Overview

The first Speaker Diarization studies were published in the 1990s. These early attempts were focused on applications like radio broadcast news and communications, with the main goal being to improve ASR performance. Consequently, ASR results directly influenced many of these initiatives. For instance, they implemented a two-pass decoding, identifying speaker changes in the first pass. Siu, who also concentrated on ASR, can be considered one of the pioneers of Speech Separation Guided Diarization (SSGD), employing blind speech separation for the diarization process. The clustering-based diarization paradigm took root during these early phases. These studies identified diarization as a clustering problem. After Voice Activity Detection (VAD), each speech segment undergoes extraction of speaker discriminative features. These features are then aggregated to determine the total number of speakers in the recording and to establish segmentation for each speaker by assigning characteristics to each associated sub-segment.

In recent years, Speaker Diarization has significantly evolved with the adoption of machine learning techniques. Modern approaches incorporate advanced methods such as:

- **Feature Embeddings**: Using Deep Learning (DL) models to extract speaker-specific features from audio signals.
- **Clustering Algorithms**: Employing agglomerative clustering or spectral clustering to group speech segments based on their features.
- **Speaker Change Detection**: Developing robust algorithms to automatically detect speaker changes within continuous speech streams, improving segmentation accuracy.
- **Integration with Speech-to-Text (STT)**: Extracting text for each speaker to provide insights.

Despite advancements, Speaker Diarization remains a challenging task due to variability in speech patterns, overlapping speech, and noisy environments. Researchers continue to explore novel approaches to improve accuracy and efficiency, leading to ongoing developments in the field. In Figure 2 can be seen how Speaker Diarization works.


I said, Well you know that far from home any familiar face is a comfort even if it is just my crisis. As I was saying, dinner was warm, and it was very complimentary. I keep getting letters saying that we have covered all of this, and they send us lunch while we talk. So how can you reread them? and get my pictures. Oh, you see they can't say ours over here. In some of these hotels, a lot of the hotels, you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? Is it spelled R-O-B-B-Y? Yeah.

a) Without Speaker Diarization

SPEAKER_01: I said, Well you know that far from home any familiar face is a comfort even if it is just my crisis.
SPEAKER_00: As I was saying, dinner was warm, and it was very complimentary. I keep getting letters saying that we have covered all of this, and they send us lunch while we talk. So how can you reread them?
SPEAKER_01: and get my pictures. Oh, you see they can't say ours over here. In some of these hotels, a lot of the hotels, you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? Is it spelled R-O-B-B-Y?
SPEAKER_00: Yeah.

b) With Speaker Diarization

Figure 2: Illustration of the importance of Speaker Diarization with ASR.

2.2 Techniques in Speaker Diarization

There are only two types of Speaker Diarization: offline and live. The offline one is useful when there are hours of recording and the main objective is to get the transcript from those conversations. In contrast, live diarization is used when there is no opportunity to record or when real-time results are needed.

1. **Offline Diarization**
   - **Feature Extraction**:
     - From the input audio files, extract features such as Mel-Frequency Cepstral Coefficients (MFCCs), log-mel spectrograms, or other spectral features.
     - Utilize libraries such as **librosa** [12] or built-in methods.
     - Normalize and window the features appropriately to capture relevant information while minimizing noise and irrelevant variations.
   - **Voice Activity Detection**:
     - Use VAD [13], [14], [15] to detect and segment the portions of the audio where human speech is present.
     - Employ VAD models like ‘speechbrain‘ or other pre-trained VAD systems to accurately identify speech segments.
     - Filter out non-speech segments (silence, noise, etc.).

- **Embeddings**:
    - Compute speaker embeddings for each speech segment identified in the previous step. Speaker embeddings are high-dimensional vectors that capture unique characteristics of each speaker's voice.
    - Use state-of-the-art speaker embedding techniques.
- **Clustering**:
    - Perform clustering on the speaker embeddings to group segments by speaker identity.
    - Apply clustering algorithms such as KMeans, agglomerative hierarchical clustering, or spectral clustering.
    - Fine-tune clustering parameters and techniques to handle variations in the number of speakers, ensuring optimal grouping.
- **Resegmentation**:
    - Refine the initial speaker segmentation using resegmentation techniques. This involves adjusting segment boundaries and reassigning segments to improve overall accuracy.
    - Utilize models that re-evaluate speaker boundaries considering context and acoustic continuity, such as Hidden Markov Models (HMMs).
    - Iteratively validate and adjust results to ensure the final segmentation is as accurate as possible.

2. **Online Diarization**
    - **Frame-wise Embedding Extraction**:
        - Extract frame-wise embeddings from the audio input using a neural network, such as Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN).
        - Ensure embeddings provide a detailed representation of audio features for each time frame, capturing nuances of speech signals.
    - **Encoder-Decoder-based Attractor Calculation (EDA)**:
        - Implement an encoder-decoder architecture using Long Short-Term Memory (LSTM) networks to sequentially calculate speaker-wise attractors.
        - The encoder processes frame-wise embeddings to generate a context vector summarizing audio features over time.
        - The decoder uses this context vector to generate attractors for each speaker until a stopping condition is met.
    - **Flexible Number of Speakers**:
        - Continue attractor generation until the decoder determines no more attractors are necessary.

- **Diarization Results Estimation**:
  - Calculate diarization results by dot product of attractors and frame-wise embeddings.
  - This operation determines similarity between each frame and each attractor, assigning frames to appropriate speakers.
  - Embeddings from overlapping speech frames yield higher dot product values with multiple attractors, effectively handling speaker overlaps.
- **Iterative Inference for Speaker Limit Removal**:
  - Implement iterative inference to remove restriction on maximum number of output speakers.
  - This approach adjusts attractors and diarization results iteratively until accurately identifying all speakers in the audio.
- **Alignment with External Speech Activity Detector**:
  - Align estimated diarization results with output from external speech activity detector.
  - This alignment ensures fair comparison with conventional cascaded approaches by standardizing detected speech segments.
- **Evaluation and Comparison**:
  - Conduct extensive evaluations on simulated and real datasets to assess performance of proposed EEND-EDA method.
  - Highlight superior performance of EEND-EDA, demonstrating advantages in handling overlapping speech and adapting to unknown number of speakers.

2.3 Existing Technologies and Tools

For live and offline applications there are two state-of-the-art competitors: Pyannote and Nemo [16]. They are presented in Table 1.

Model	Pyannote	NeMo
VAD	Pyannet derived from Syncnet	MarbleNet
Audio embedding	ECAPA-TDNN	TitaNet
Clustering	Hidden Markov Model clustering	Multi-scale clustering (MSDD)

Table 1: Comparison of state-of-the-art Speaker Diarization.

Both of them take the audio file as input and generate a Rich Transcription Time Marked (RTTM) file as output. They have public pre-trained models available, which this thesis leverages. Here is a list of advantages and disadvantages for each:

- Only **Pyannote** has **multilabel segmentation**.
- **NeMo** can be easily integrated with **ASR** and Natural Language Processing **(NLP)** tasks.
- Both of them have the ability to **specify the number of speakers** for inference, but they can also **determine it** automatically.
- Pyannote is not as customizable for pipelines.

In this thesis, Pyannote is used for Speaker Segmentation, which produces an RTTM with various fields: Type, File ID, Channel, Start Time, Duration, Orthography, Confidence, Speaker, x, y. The fields that are particularly useful are File ID, Start Time, Duration, and Speaker, as these allow for continuous segments per speaker without consecutive repetitions.

Additionally, there is WhisperX [17], which timestamps words. It refines timestamps from Whisper transcriptions using forced alignment with a phoneme-based ASR model (e.g., wav2vec 2.0). This provides word-level timestamps and improves segment accuracy. The authors demonstrated that this preprocessing reduces hallucination and repetition, enabling batched transcription within the audio. It resulted in a twelve-fold speed increase without sacrificing transcription quality. WhisperX provides accurate word-level segmentations with minimal inference overhead, resulting in time-accurate transcriptions suitable for various applications.

2.4 Challenges in Speaker Diarization

The meeting scenario for Speaker Diarization, known as "who speaks what at when," is one of the most valuable and challenging scenarios for the deployment of speech technologies. Specifically, two typical tasks, Speaker Diarization and multi-speaker automatic speech recognition, have attracted attention. The lack of large public meeting data has been an obstacle for the advancement of the field. This paper [18] introduced a solution for this: a corpus, AliMeeting, with 120 hours of Mandarin data, which includes 8-channel microphone. It is a significant dataset as it involves meetings with 2-4 speakers.

In Table 2 are their results.

Table 2: Multi-speaker ASR results on Eval set (CER%).

Conformer	Training data	Eval-Ali-far	Eval-Ali-far-bf
A	Train-Ali-near	49.0	45.6
SOTA	Train-Ali-far	34.3	38.2
SOTB	+Train-Ali-near	30.8	33.2
SOTA_bf	Train-Ali-far-bf	41.2	33.6
SOTB_bf	+Train-Ali-near	34.3	29.7

In this case, the Train-Ali-far set was used to train Conformer SOTA, and the Ali-far-bf data was used to train Conformer SOTA_bf. The training set of Train-Ali-near was introduced to model B in order to enhance the model's capacity for acoustic modeling, as demonstrated by the improvement in the Eval sets. The SOT multi-speaker model significantly outperforms the single-speaker model on our multi-speaker evaluation set. Additionally, the use of Beamformer (bf) in conjunction with SOT produces positive results on the Ali-far-bf evaluation set.

There are also the problems with live diarization and overlapping speakers, but we discussed them earlier in the thesis. See chapter 2.2 for the live pipeline.

Another problem that can arise is the lack of data. More and more data is not free or collaborative. This can be an obstacle for most new researchers. In Table 3 is a list of some datasets and their pricing for further usage.

Audio	**Diarization ground truth**	**Language**	**Pricing**
2000 NIST Speaker Recognition Evaluation	Disk-6 (Switchboard), Disk-8 (CALLHOME)	Multiple	$2400.00
2003 NIST Rich Transcription Evaluation Data	Together with audios	en, ar, zh	$2000.00
CALLHOME American English Speech	CALLHOME American English Transcripts	en	$2500.00
The ICSI Meeting Corpus	Together with audios	en	Free License
The AMI Meeting Corpus	Together with audios (need to be processed)	Multiple	Free
Fisher English Training Speech Part 1	Fisher English Training Speech Part 1 Transcripts	en	$8000.00
Fisher English Training Part 2 Speech	Fisher English Training Part 2 Transcripts	en	$8000.00

Table 3: List of audio datasets with their details.

2.5 Innovations in the Field

There have been two innovations in recent years: incorporating LLMs to correct Word Error Rate (WER) and implementing End-to-End Neural Networks for live diarization.

The LLM interprets text and assigns words to the correct speaker, addressing issues that arise from generative ASR models. In this thesis, it is structured the pipeline to integrate the LLM at the final stage. It can be seen in Figure 3.

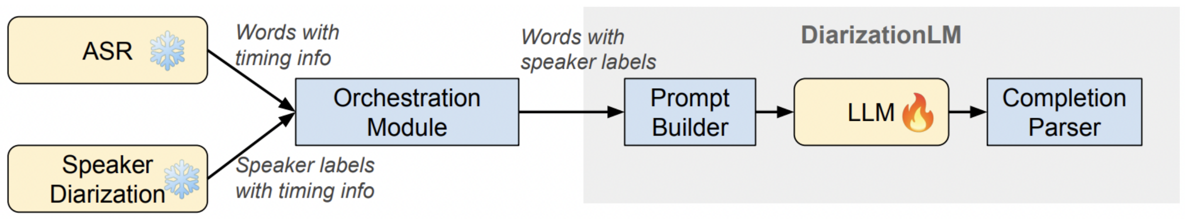


Figure 3: DiarizationLM. [19]

The Encoder-Decoder Based Attractors for End-to-End Neural Diarization are widely regarded as state-of-the-art for live diarization. This approach effectively handles scenarios with an unknown number of speakers. This method is outlined in Figure 4.

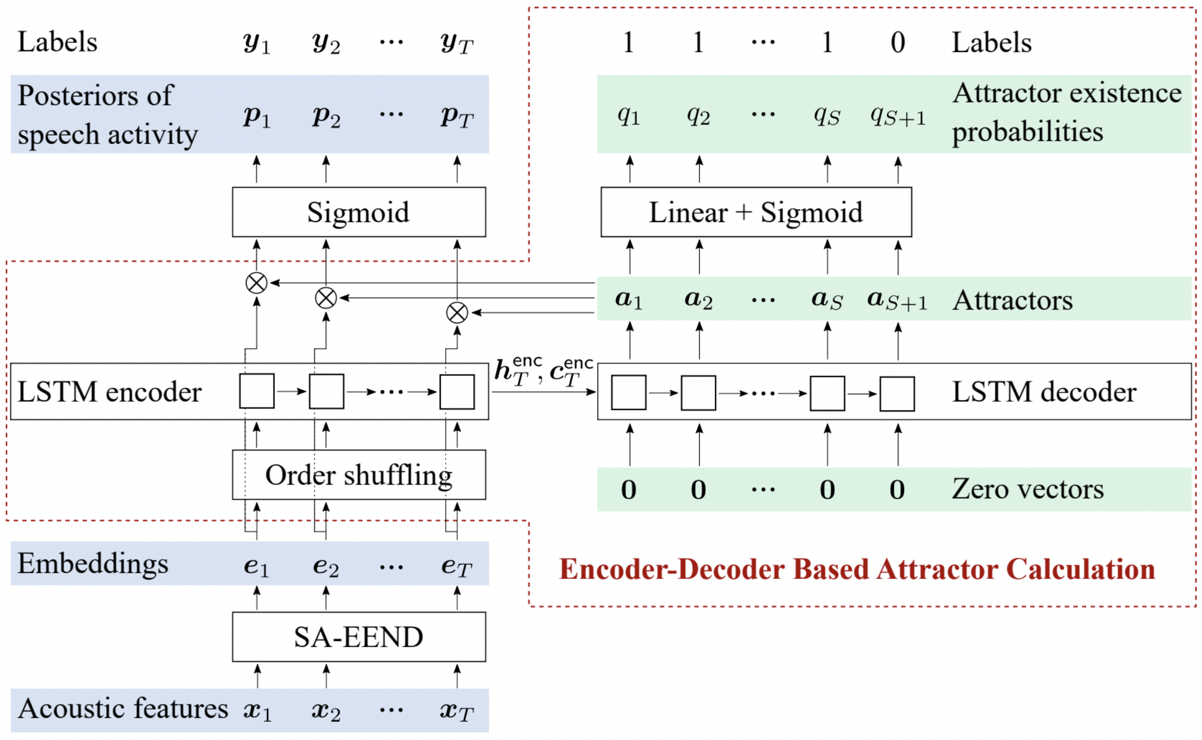

Labels y_1 y_2 $\cdots$ y_T
Posteriors of speech activity p_1 p_2 $\cdots$ p_T
Sigmoid
LSTM encoder
h_T^{enc}, c_T^{enc}
Order shuffling
Embeddings e_1 e_2 $\cdots$ e_T
SA-EEND
Acoustic features x_1 x_2 $\cdots$ x_T
1 1 $\cdots$ 1 0 Labels
q_1 q_2 $\cdots$ q_S q_{S+1} Attractor existence probabilities
Linear + Sigmoid
a_1 a_2 $\cdots$ a_S a_{S+1} Attractors
LSTM decoder
0 0 $\cdots$ 0 0 Zero vectors
Encoder-Decoder Based Attractor Calculation

Figure 4: EEND with Encoder-Decoder based Attractor calculation (EENDEDA). [20]

3 PROPOSED PIPELINE FOR SPEAKER DIARIZATION

3.1 Overview of the Pipeline

The architecture is straightforward and can be seen in Figure 5.

1. Process the dataset from the CALLHOME American English data to convert the given data into audio.
2. Use Pyannote to extract segments (start-end) for each speaker.
3. Divide the audio file based on RTTM and run Whisper on each new audio segment.
4. Compile the data into a single script for diarization.
5. Utilize GPT-3.5 Turbo and GPT-4 with zero-shot or one-shot prompts, ensuring consistency across prompts for each instance.

It is important to note that the entire pipeline operates at 0 temperature to maximize accuracy.

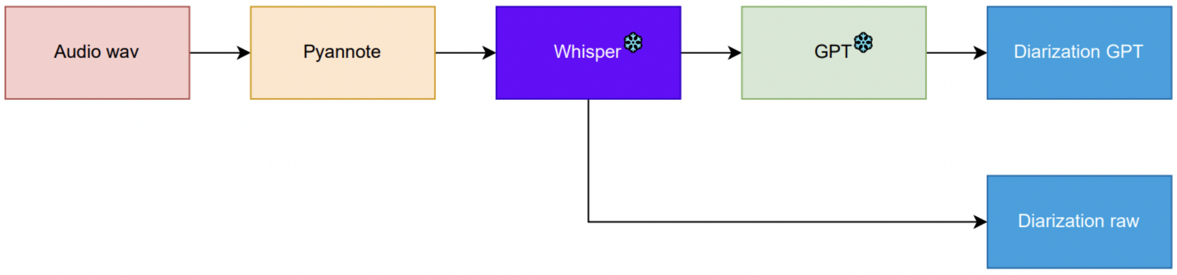


Figure 5: Pipeline of the model.

Pyannote version 3.1, the latest, outputs the list of speakers along with their respective start and end times. Whisper utilizes large-v3 twice: once to transcribe the entire audio and another time for each segment provided by Pyannote. Due to limited contextual understanding, GPT-3.5 Turbo and GPT-4 are employed. They receive identical prompts, whether zero-shot or one-shot.

3.2 Pyannote for Speaker Diarization

Pyannote is a Python library specifically designed for audio analysis tasks, including Speaker Diarization. It provides tools and pre-trained models that enable accurate identification and segmentation of speakers within audio data. Pyannote operates effectively on mono audio samples at 16kHz and generates segments with annotations.

To integrate Pyannote into our Speaker Diarization pipeline, the following setup and configuration steps were undertaken:

- **Installation**: Pyannote was installed using standard Python package management tools (pip install Pyannote). Dependencies such as PyTorch were also installed to leverage GPU acceleration for improved processing speed.
- **Model Initialization**: The pre-trained diarization pipeline (Pyannote/speaker-diarization-3.1) was initialized with an authentication token (use_auth_token) for secure access to the model repository.

To implement Pyannote, we started with this example code snippet [21]. It demonstrates how to integrate Pyannote and how to configure it to run on GPU. The library allows control over the number of speakers from the outset, including options to set min_speakers and max_speakers. Pyannote can also load WAV files directly into memory for faster processing.

Listing 3.1: Using pre-trained Pyannote model.

```
pipeline = Pipeline.from_pretrained(
"Pyannote/speaker-diarization-3.1",
use_auth_token="HUGGINGFACE_ACCESS_TOKEN_GOES_HERE")

diarization = pipeline("audio.wav")
```

The output format is an RTTM file named "sample.rttm," which is suitable for further analysis and evaluation tasks such as calculating Diarization Error Rate (DER). It provides a comprehensive listing of speaker enumerations as well.

3.3 Whisper for Text Extraction

Whisper is a pre-trained model designed for Automatic Speech Recognition (ASR) and speech translation. Trained on 680k hours of labeled data, Whisper models demonstrate robust generalization across various datasets and domains without the need for fine-tuning.

Whisper offers several model variants tailored to different tasks: tiny, base, small, medium, large, large-v2, and the latest, large-v3, which is used in this paper. In Table 4 is a comparison of their performance:

Size	Parameters	English-only	Multilingual
tiny	39M	x	x
base	74M	x	x
small	244M	x	x
medium	769M	x	x
large	1550M	NaN	x
large-v2	1550M	NaN	x
large-v3	1550M	NaN	x

Table 4: Comparison of Whisper models.

To integrate Whisper into our Speaker Diarization pipeline, the following setup and configuration steps were undertaken:

- **Installation**: Whisper was installed using the Transformers library. First, install the Transformers library from the GitHub repository:
  $ pip install –upgrade git+https://github.com/huggingface/transformers.git
- **Model Initialization**: The pre-trained model pipeline (model_id = "openai/whisper-large-v3") was initialized.

For implementing Whisper, we used this code example [22], which demonstrates how to integrate and run Whisper on GPU for optimized performance. Below is an example of how to call the pre-trained pipeline. You can adjust parameters like temperature and batch size for better GPU utilization.

The output is a dictionary containing the resulting text ("text"), segment-level details ("segments"), and detected language ("language"), which is automatically identified unless specified otherwise (decode_options["language"]). In our case, English is explicitly set as the language since the dataset is entirely in English. Whisper's capability to detect language dynamically enhances its versatility.

Listing 3.2: Usage of pre-trained Whisper.

```
pipe = pipeline(
"automatic-speech-recognition",
model=model,
tokenizer=processor.tokenizer,
feature_extractor=processor.feature_extractor,
max_new_tokens=128,
chunk_length_s=30,
batch_size=16,
return_timestamps=True,
torch_dtype=torch_dtype,
device=device,
)

result = pipe("audio.mp3")
print(result["text"])
```

3.4 Integration with Language Models

Having consulted the documentation [23], we can interact with different models of GPT.

Listing 3.3: Send prompt to GPT.

```
completion = client.chat.completions.create(
  model=YOUR_MODEL,
  messages=[
    {"role": "system", "content":
                                "You␣are␣a␣helpful␣assistant."},
    {"role": "user", "content": prompt}
    ]
```

You can provide a prompt for whatever you need. You will require certain keys to access it:

Listing 3.4: Usage of GPT.

```
client_GPT = AzureOpenAI(
    azure_endpoint = YOUR_ENDPOINT,
    api_version = YOUR_VERSION,
    api_key = YOUR_KEY,
    )

model_GPT4 = YOUR_MODEL_NAME
```

After this, the result can be extracted using **completion.choices[0].message.content**.

4 DETAILED METHODOLOGY

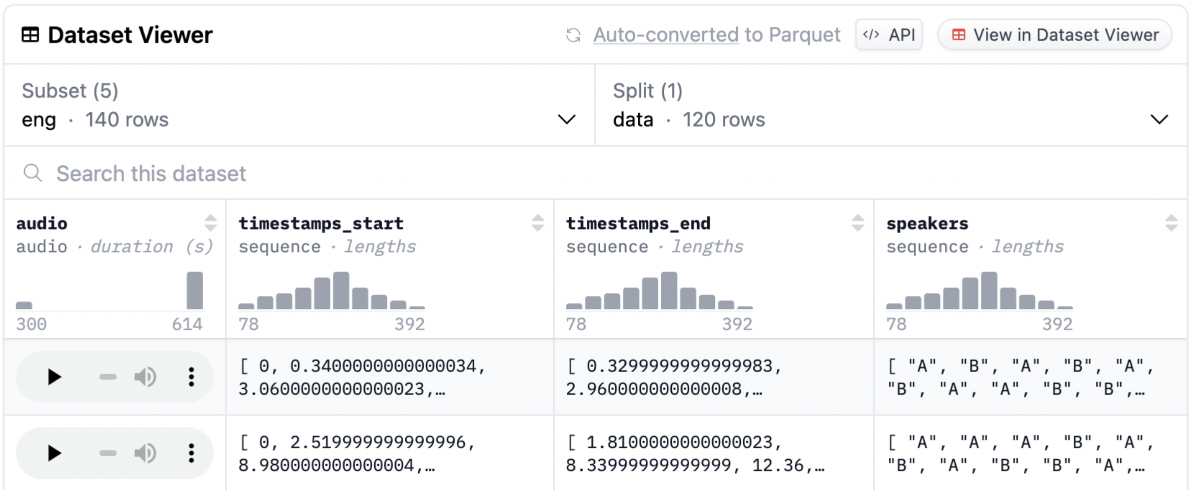
4.1 Dataset Description: CALLHOME English Dataset

The CALLHOME Corpus is a collection of unscripted telephone conversations between native speakers in Chinese, English, German, Japanese, and Spanish.

This processed version of the original CALLHOME dataset is sourced from the TalkBank corpora, available here. It includes subsets in Chinese, English, German, Japanese, and Spanish. The dataset is compatible with the diarizers library for fine-tuning Pyannote segmentation models, as it has been pre-processed using diarizers.

The choice of this dataset was straightforward: it consists of conversations between only two speakers and is not overly large. Given the time-consuming nature of tasks such as Pyannote segmentation, Whisper ASR, GPT interrogation, audio trimming, etc., limiting the dataset to 140 entries was a practical decision. Furthermore, many research papers on Speaker Diarization and other audio tasks use this dataset (e.g., EXAMPLE).

The dataset structure is as follows in Figure 6.



eng · 140 rows

data · 120 rows

audio audio · *duration (s)* 300–614	timestamps_start sequence · *lengths* 78–392	timestamps_end sequence · *lengths* 78–392	speakers sequence · *lengths* 78–392
	[0, 0.3400000000000034, 3.0600000000000023,…	[0.3299999999999983, 2.960000000000008,…	["A", "B", "A", "B", "A", "B", "A", "A", "B", "B",…
	[0, 2.519999999999996, 8.980000000000004,…	[1.8100000000000023, 8.33999999999999, 12.36,…	["A", "A", "A", "B", "A", "B", "A", "B", "B", "A",…

Figure 6: CALLHOME dataset from Hugging Face.

4.2 Preprocessing Steps

For processing the data, we managed to gain these insights:

Listing 4.1: Dataset manipulation.

```python
from datasets import load_dataset

ds = load_dataset("talkbank/callhome", "eng")
dataset = ds['data']

for audio_number, row in enumerate(dataset):
    audio_data = row['audio']
    speakers_GT = row['speakers']

    audio_array = audio_data['array']
    sampling_rate = audio_data['sampling_rate']

    print(type(audio_array))
    print(type(speakers_GT))
    break
```

For converting the audio_array to a WAV file, which is of type <**class 'numpy.ndarray'**>, and obtaining the sampling_rate, we used the soundfile library (**import soundfile as sf**). The code snippet below normalizes the audio data (audio_array) to 16-bit integer format suitable for saving as a WAV file.

Listing 4.2: Converting array data to WAV file.

```python
audio_array_normalized = np.int16(audio_array * 32767)
output_wav_file = os.path.join(os.path.dirname(output_csv_path),
                  os.path.splitext(output_csv_path)[0] + '_audio.wav')
sf.write(output_wav_file, audio_array_normalized,
        sampling_rate, subtype='PCM_16')
```

4.3 Evaluation Metrics

While the Diarization Error Rate (DER) provides a convenient method for comparing different diarization approaches, it often falls short in fully capturing the types of errors committed by the system. Our objective is to develop a robust pipeline capable of achieving speaker diarization and text alignment for each speaker at the word level.

Purity and Coverage [24] are two complementary evaluation metrics that offer deeper insights into system behavior, particularly in correctly identifying speaker clusters.

Word Diarization Error Rate (WDER) [25] and concatenated minimum-permutation Word Error Rate (cpWER) [26] are micro-level metrics used to gauge the accuracy of Speech-to-Text (STT) systems.

4.3.1 Diarization Purity

If every label in a hypothesized annotation overlaps exclusively with segments belonging to a single reference label, then the annotation is considered perfectly pure.

The purity metric is defined as:

$$\text{Purity} = \frac{\sum_{\text{cluster}} \max_{\text{speaker}} |\text{cluster} \cap \text{speaker}|}{\sum_{\text{cluster}} |\text{cluster}|} \tag{1}$$

Here, $|\text{cluster} \cap \text{speaker}|$ represents the duration of their intersection, where speaker denotes the total speech length of a specific reference speaker.

Under-segmented results, such as when two speakers are merged into one large cluster, typically yield lower purity and higher coverage. Conversely, over-segmented results, where there are numerous speaker clusters, tend to yield higher purity and lower coverage.

4.3.2 Diarization Coverage

When all segments from a particular reference label are clustered into the same cluster, a hypothesised annotation is said to have perfect coverage.

$$\text{Coverage} = \frac{\sum_{\text{speaker}} \max_{\text{cluster}} |\text{speaker} \cap \text{cluster}|}{\sum_{\text{speaker}} |\text{speaker}|} \tag{2}$$

Here, $|\text{cluster} \cap \text{speaker}|$ is the duration of their intersection, and cluster is the hypothesized cluster.

4.3.3 Word Diarization Error Rate (WDER)

The word diarization error rate, or WDER, quantifies the frequency with which words are incorrectly attributed to the wrong speaker relative to the actual data. WDER is a key metric used to assess the quality of diarization, where lower values indicate better diarization quality.

The formula for WDER is:

$$\text{WDER} = \frac{S_{IS} + C_{IS}}{\text{S} + \text{C}} \tag{3}$$

Where:

1. S_{IS} is the quantity of ASR substitutions with incorrect speaker tokens.
2. C_{IS} is the amount of Correct ASR words with Incorrect Speaker tokens.
3. S is the total amount of ASR replacements.
4. C is the number of Correct ASR words.

4.3.4 Concatenated Minimum-Permutation Word Error Rate (cp-WER)

The concatenated minimum-permutation word error rate (cpWER) assesses the alignment of recognized words with ground truth transcripts considering all possible permutations of the recognized words. It is defined as follows:

1. Combine every speaker's transcript for reference and research purposes.
2. Determine the WER between the hypothesis's reference and every speaker permutation that might exist.
3. Select the permutation with the lowest WER, which is thought to be the best one, out of all of them.

5 IMPLEMENTATION DETAILS

5.1 Software Requirements and Environment Setup

For this project, we utilized Anaconda with an image of `devel cuda 11.6.0 cudnn miniconda`, ensuring compatibility with the required CUDA and cuDNN versions for GPU acceleration. The following outlines the key software and versions used:

- Python Version: 3.10.12
- Anaconda: Managed environments and dependencies.
- Torch: Required for PyTorch-based applications.
- Environment: Configured with the latest libraries and dependencies to support machine learning tasks in speech processing.

Execution Environment

- Virtual Machine: All computations and processes were conducted on a virtual machine environment, ensuring isolated and controlled execution.
- Continuous Inference: The pipeline operated continuously with a consistent temperature setting, ensuring stable performance without manual intervention.
- Checkpoint Management: Checkpoints were implemented to safeguard against unforeseen errors, enabling recovery without restarting from the beginning.

For a comprehensive understanding of the implementation details and metric evaluations, please refer to Appendix A, where the complete code, including metric computations such as Diarization Purity, Diarization Coverage, WDER, and cpWER, is provided.

5.2 Implementation of Pyannote

The implementation of Pyannote for Speaker Diarization involves several critical steps, from initializing the diarization pipeline to processing and converting the diarization results. Below, we detail the implementation process.

Initialization of Pyannote The implementation of Pyannote for Speaker Diarization involves several critical steps, from initializing the diarization pipeline to processing and converting the diarization results. Below, we detail the implementation process.

Initialization of Pyannote The first step is to initialize the Pyannote pipeline. This is done by loading a pre-trained diarization model from the Hugging Face model hub. The initialization also includes setting up authentication and preparing the environment to leverage GPU acceleration for efficient processing.

Listing 5.1: Usage of pre-trained Pyannote.

```
class PyannoteProcessor:
    """
    Class to perform Speaker Diarization using the Pyannote library.
    """

    def __init__(self):
        self.pipeline = Pipeline.from_pretrained(
            "Pyannote/speaker-diarization-3.1",
             use_auth_token="your_huggingface_token",
        )
```

In the **__init__** method, the **Pipeline.from_pretrained** function loads the pre-trained diarization model. The **use_auth_token** parameter is required to access the model on the Hugging Face model hub. Make sure to replace **"your_huggingface_token"** with your actual Hugging Face authentication token.

Performing Diarization Once the pipeline is initialized, the next step is to perform diarization on the given audio file. This involves passing the audio file through the Pyannote pipeline, which outputs the diarization results. The results are then saved in RTTM format.

Listing 5.2: Performing diarization with Pyannote.

```
def perform_diarization(self, audio_file_path):
    self.pipeline.to(torch.device('cuda')) # switch to GPU

    # Hardcoding the number of speakers
    diarization = self.pipeline(audio_file_path, num_speakers=2)

    with open("sample.rttm", "w") as rttm:
        diarization.write_rttm(rttm)
```

In the **perform_diarization** method, the **pipeline.to(torch.device('cuda'))** line ensures that the processing is done on the GPU, which significantly speeds up the computation. The **audio_file_path** is the path to the audio file to be diarized. The **num_speakers** parameter is hardcoded to 2 because the dataset consists only of phone conversations between two people. For future projects, it can be adjusted based on specific requirements.

Converting RTTM to DataFrame The diarization results are saved in RTTM format, which needs to be processed further. The RTTM file is read and converted into a DataFrame for easier manipulation and analysis.

Listing 5.3: Converting RTTM to DataFrame.

```python
def rttm_to_dataframe(self, rttm_file_path):
    columns = [
    "Type", "File ID",
    "Channel", "Start Time",
    "Duration", "Orthography",
    "Confidence", "Speaker",
    "x", "y",
    ]

    data = []
    with open(rttm_file_path, "r") as rttm_file:
        lines = rttm_file.readlines()
        data = [line.strip().split() for line in lines]

    df = pd.DataFrame(data, columns=columns)
    df = df.drop(["x", "y", "Orthography", "Confidence"], axis=1)
    return df
```

The **rttm_to_dataframe** method reads the RTTM file, processes each line to extract relevant information, and stores it in a Pandas DataFrame. Unnecessary columns are dropped to retain only essential information, such as Type, File ID, Channel, Start Time, Duration, and Speaker.

Complete Workflow Integration To integrate the Pyannote processor within a complete workflow, an **AudioProcessor** class is used. It manages interactions between Pyannote and other components like Whisper and the language model.

The **process_and_append_to_csv** method in the **AudioProcessor** class integrates Pyannote for diarization, normalizes audio data, transcribes using Whisper, and integrates the language model for diarization correction.

Segmentation Refinement To enhance data interpretability, consecutive segments attributed to the same speaker are merged into cohesive intervals. This step ensures that uninterrupted speech segments by the same speaker are treated as continuous units for analysis.

By following these steps, Pyannote is effectively implemented for Speaker Diarization, leveraging its capabilities for accurate and efficient speaker segmentation.

5.3 Implementation of Whisper

Whisper is used on chunks of audio that were previously saved after processing with Pyannote. Since we aimed to create a reliable Speaker Diarization model, we opted for the best model available in Whisper, which is the 1550M model. Given its size, it was essential to utilize GPU acceleration. For this purpose, we employed the NVIDIA GeForce RTX 4090. Another crucial setup involved setting the temperature to 0. This ensures that there are no hallucinations during the language model's processing of input data.

Segmenting the audio To process specific segments of audio, we implemented the `process_audio_segment` method. This method extracts a segment from the specified audio file (`audio_file`) based on the provided `start_time` and `end_time` parameters. It then transcribes the segment using `transcribe_audio_with_whisper`, collapses the resulting transcription segments using `collapse_segments`, and removes temporary audio files once processing is complete. Our goal was to optimize efficiency throughout this process.

Listing 5.4: Usage of pre-trained Whisper.

```
def process_audio_segment(
        self, audio_file, start_time, end_time, detected_language
    ):
        start_time = float(start_time * 1000)
        end_time = float(end_time * 1000)

        audio = AudioSegment.from_file(audio_file)
        audio_segment = audio[start_time:end_time]

        audio_segment_path = f"audio_segment_{start_time}.wav"
        audio_segment.export(audio_segment_path, format="wav")
        # Transcribe the audio segment
        transcription_result = self.transcribe_audio_with_whisper(
            audio_segment_path, detected_language
        )
        whisper_transcript = transcription_result["text"]

        # Split the transcript into segments
        segments = whisper_transcript.split("\n")

        # Collapse the segments
        collapsed_transcript = self.collapse_segments(segments)

        # Delete the temporary audio segment file
        os.remove(audio_segment_path)

        return collapsed_transcript
```

Combing the results The `collapse_segments` method processes individual transcript segments by combining words and spaces into a cohesive transcript format. This ensures the readability and coherence of the transcribed output from Whisper ASR.

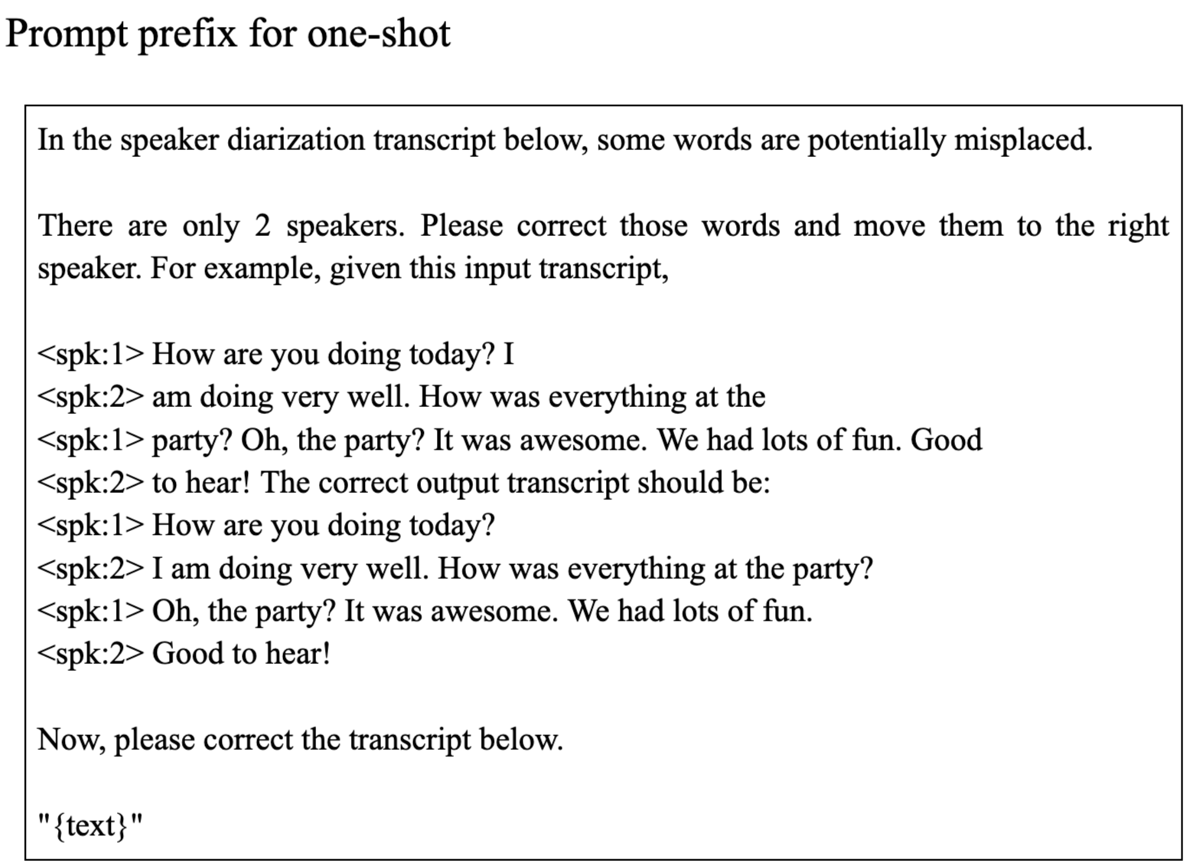
Listing 5.5: Processes an audio segment within a specified time range.

```
def collapse_segments(self, transcript_segments):
        segment_counter = 0
        collapsed_segments = []

        for segment in transcript_segments:
            if segment.startswith("Segment"):
                segment_counter += 1
                collapsed_segments.append(segment)
            else:
                words = segment.split()
                for word in words:
                    collapsed_segments.append(word)
                    collapsed_segments.append("␣")

        collapsed_transcript = "".join(collapsed_segments)
        return collapsed_transcript
```

In the `AudioProcessor` class, the `process_and_append_to_csv` method integrates the Whisper result as a final step of diarization. It handles the dataset, processes each audio file, and performs the necessary operations. Additionally, the language is hardcoded as English because the entire dataset is in English. However, leaving the language parameter blank or None is also possible since Whisper can identify the language automatically, even when different languages are spoken in the same conversation.

By following these steps, Whisper is effectively implemented for Speaker Diarization, leveraging its capabilities for accurate and efficient Speech-to-Text conversion.

5.4 Language Model Integration

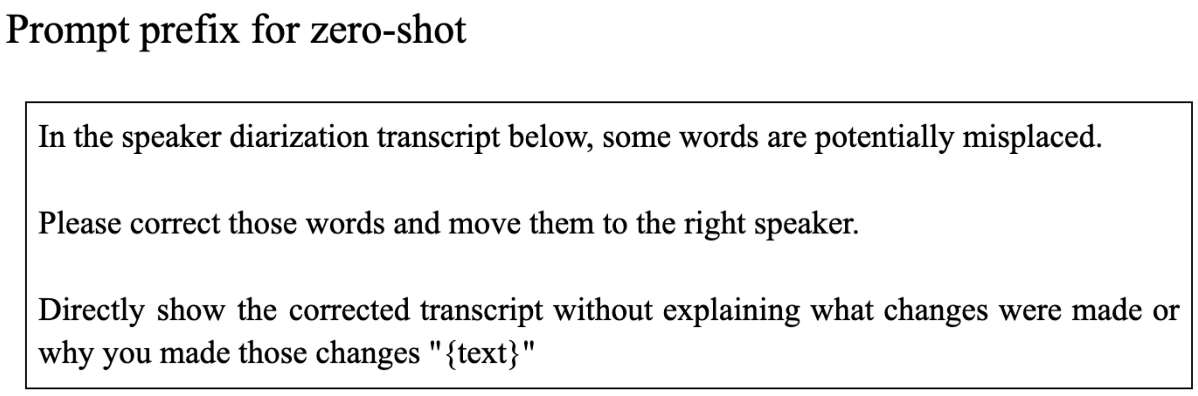


Prompt prefix for one-shot

```
In the speaker diarization transcript below, some words are potentially misplaced.

There are only 2 speakers. Please correct those words and move them to the right
speaker. For example, given this input transcript,

<spk:1> How are you doing today? I
<spk:2> am doing very well. How was everything at the
<spk:1> party? Oh, the party? It was awesome. We had lots of fun. Good
<spk:2> to hear! The correct output transcript should be:
<spk:1> How are you doing today?
<spk:2> I am doing very well. How was everything at the party?
<spk:1> Oh, the party? It was awesome. We had lots of fun.
<spk:2> Good to hear!

Now, please correct the transcript below.

"{text}"
```

Figure 7: One-shot prompt.

For the one-shot setup, which is in Figure 7, the prompt prefix includes both the task description and an example. But, in Figure 8, the zero-shot has only the prompt.


Prompt prefix for zero-shot

In the speaker diarization transcript below, some words are potentially misplaced.

Please correct those words and move them to the right speaker.

Directly show the corrected transcript without explaining what changes were made or why you made those changes "{text}"

Figure 8: Zero-shot prompt.

We experimented with two different language models, GPT-3.5 Turbo and GPT-4, without fine-tuning. For the one-shot setup, the prompt prefix includes both the task description and an example, which were provided to the language models.

5.5 Code Structure and Modules

Imports necessary libraries and modules such as `os, csv, subprocess, time`, various **AI/ML** libraries (`torch, transformers`), **audio processing** libraries (`soundfile, pydub`), and specific modules (`whisper, Pyannote, datasets`).

Classes and Functions

- PyannoteProcessor: Handles Speaker Diarization using the Pyannote library. Converts diarization results to a Pandas DataFrame.
- WhisperProcessor: Utilizes the Whisper model for ASR tasks. Functions include transcribing audio, detecting language, processing audio segments, and collapsing transcription segments.
- LLM: Implements methods to generate prompts and obtain completions from a language model. Uses OpenAI's GPT models (GPT-3.5 Turbo and GPT-4).

AudioProcessor Orchestrates the entire workflow, integrating PyannoteProcessor and WhisperProcessor. It handles dataset processing, performs Speaker Diarization, generates transcriptions, and uses language models to refine the transcriptions.

All sensitive data were handled with Hydra. To avoid exposing tokens, access credentials, or other sensitive information to external platforms like GPT or others, everything was kept local and managed within a `config.yaml` file.

Listing 5.6: Usage of Hydra.

```
def load_config():
    with open("./config/config.yaml", "r") as config_file:
        config = yaml.safe_load(config_file)
    return config

    GlobalHydra.instance().clear()
    # Load Hydra configuration
    initialize(config_path="../config")
    cfg = compose(config_name="prompts.yaml")
```

5.6 Flowcharts and Diagrams

The software architecture comprises an **AudioProcessor** that centrally coordinates the **PyannoteProcessor**, **WhisperProcessor**, and **LLM**. The **PyannoteProcessor** manages Speaker Diarization, leveraging the Pyannote library to convert results into structured DataFrames. Meanwhile, the **WhisperProcessor** utilizes the Whisper model for automatic speech recognition (ASR), handling language detection, audio segment processing, and transcript consolidation. The **LLM** serves as a language model interface, employing GPT models to generate prompts and refine transcriptions seamlessly. Diagram of the structure is in Figure 9.

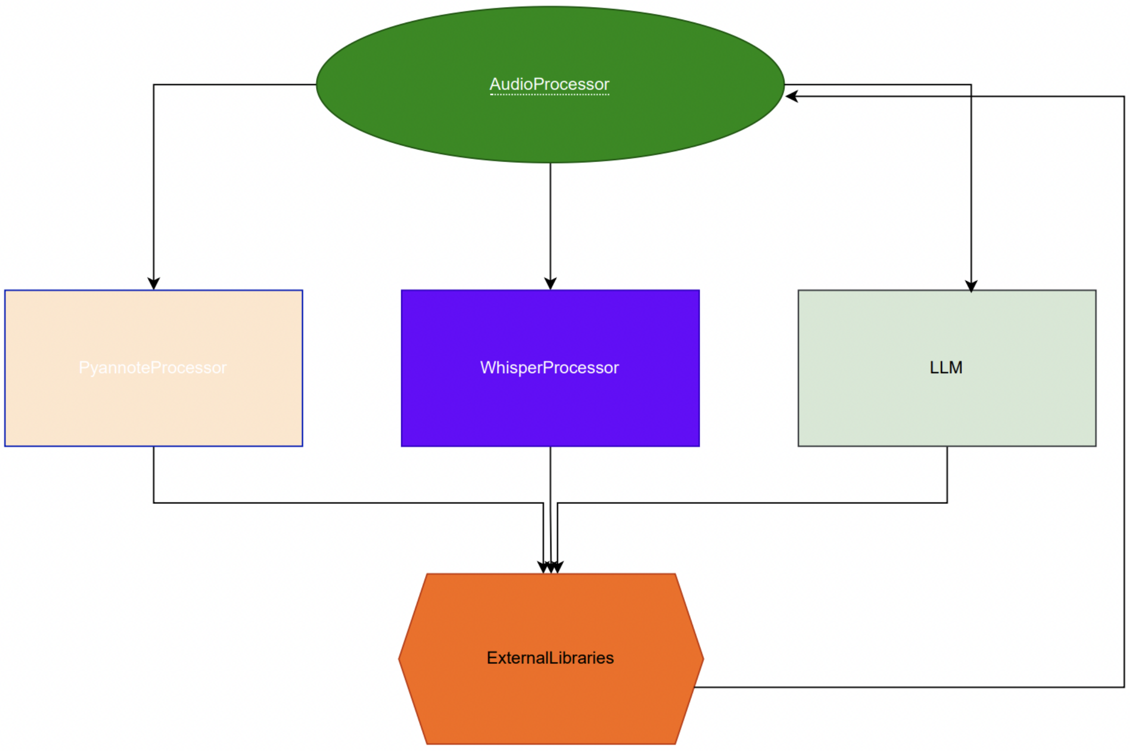


Figure 9: Diagram of the Code Structure.

- **AudioProcessor**: Central orchestrator, coordinates PyannoteProcessor, WhisperProcessor, and LLM.
- **PyannoteProcessor**: Handles Speaker Diarization using the Pyannote library, converts results to a DataFrame.
- **WhisperProcessor**: Utilizes the Whisper model for ASR tasks, detects language, processes audio segments, and collapses transcriptions.
- **LLM**: Language model interface, generates prompts, and refines transcriptions using GPT models.

Dependencies

- **Pyannote**: For Speaker Diarization.
- **Whisper**: For ASR tasks.
- **OpenAI GPT Models**: For language model-driven corrections.
- **Pandas**: For data manipulation.
- **NumPy, tqdm, etc.**: Supporting libraries for various operations.

Here are all used libraries together with a brief explanation of usage:

Listing 5.7: Libraries.

```python
# Whisper specific imports
import whisper
from whisper.audio import
    pad_or_trim, log_mel_spectrogram, N_FRAMES

# Pyannote specific imports
from Pyannote.audio import Pipeline
from Pyannote.core import Annotation, Segment
from Pyannote.metrics.diarization import
    DiarizationPurity, DiarizationCoverage

# OpenAI specific import
from openai import AzureOpenAI

# Configuration related imports
from config.config import load_config
import config.config as config

# Hydra related imports
from hydra import compose, initialize
from hydra.core.global_hydra import GlobalHydra
```

6 EXPERIMENTAL RESULTS

6.1 Diarization Scenarios

without_LM	**SPEAKER_01**: I said, well you know, that far away from home,
	SPEAKER_00:
	SPEAKER_01: Anyway, well, it was fun. The food was great, it was
	SPEAKER_00:
	SPEAKER_01: and get my pictures. Yeah. Oh, you see, they can't
	SPEAKER_00:
	SPEAKER_01: yup what i told the first time to get your room number.
35_turbo_zero_shot	**SPEAKER_01**: I said, well
	SPEAKER_00: And get my pictures.
	SPEAKER_01: Yeah. Oh, you see,
	SPEAKER_00: Yup, what I told the first
35_turbo_one_shot	**SPEAKER_01**: I said, well you
	SPEAKER_00: Ha ha ha!
	SPEAKER_01: Uh... advocate uncomplacent what's the
4_zero_shot	**SPEAKER_01**: I said, well you know, .
	SPEAKER_00: And get my pictures.
	SPEAKER_01: Yeah. Oh, you see, they
	SPEAKER_00:
	SPEAKER_01: yup what i told the first time to get your room number
4_one_shot	**SPEAKER_01**: I said, well
	SPEAKER_00:
	SPEAKER_01: and get my

Table 5: Outputs of the pipeline.

In Table 5 is an example, but in appendix B are presented more. What we wanted to discuss is about the improved accuracy for GPT-3.5 Turbo and 4. We can see that there are no empty speaker labels with the LLM, so it is a good improvement.

The Ground Truth is exactly like GPT-3.5 Turbo with a one-shot prompt. This could be the best example, but more will be discussed in the next chapters. The worst option is obviously the one without LM.

6.2 Performance Analysis and Metrics

The metrics that were evaluated were both word-level accuracy and speaker-level segmentation quality. There is a process for word-level evaluation where the speech Ground Truth (GT) obtained with Whisper is run over the initial audio. Every type of diarization was compared with this GT, and the two metrics for word-level, WDER and cpWER, were obtained. For the speaker-level segmentation, the speakers GT was obtained from the dataset. By extracting every sequence of non-consecutive repetition of speakers, which was also removed from GT, we obtained purity and coverage.

Every split is transformed into text with Whisper. This idea came from the fact that the interrogation of OpenAI's API takes too long for the entire audio. However, it was necessary to do this after the process of Speaker Diarization in order to evaluate the model.

The corpus did not have very long text in them; there is enough context for GPT-3.5 Turbo, 16K, and even for GPT-4, 8k, in order to pass with a prompt. The method is using two types of prompts: zero-shot and one-shot. Both of them solved the initial problem, one of them better than the other. The idea for two kinds of prompts came from the fact that the LLM can understand the task better and the hallucinations can tend to 0. The temperature was established earlier in the code with a value of 0. All of these examples can be easily reproduced with any kind of commercial LLM. The choice for these GPTs was made because they are widely used by humans.

In Figure 10, titled "Diarization without LM," the image illustrates instances where labels are empty, highlighting a challenge in the diarization process when not utilizing a Language Model (LM). This can lead to inaccuracies or missing speaker identifications, impacting overall segmentation quality.

Contrastingly, Figure 11 showcases an example result using GPT-3.5 Turbo with a zero-shot prompt. Here, the output demonstrates significant improvement: there are no misplaced words and no instances of empty speakers' dialogue. This success underscores the effectiveness of leveraging advanced language models like GPT-3.5 Turbo in enhancing diarization accuracy and ensuring coherent dialogue segmentation.

These figures provide visual insights into the impact of employing sophisticated AI models in speech processing tasks, emphasizing the importance of technological advancements in refining audio analysis and transcription capabilities.

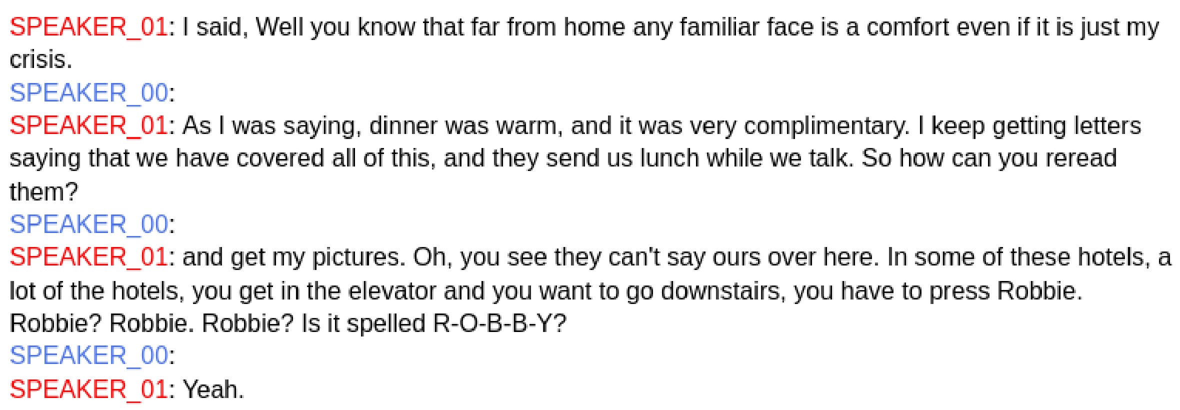

SPEAKER_01: I said, Well you know that far from home any familiar face is a comfort even if it is just my crisis.
SPEAKER_00:
SPEAKER_01: As I was saying, dinner was warm, and it was very complimentary. I keep getting letters saying that we have covered all of this, and they send us lunch while we talk. So how can you reread them?
SPEAKER_00:
SPEAKER_01: and get my pictures. Oh, you see they can't say ours over here. In some of these hotels, a lot of the hotels, you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? Is it spelled R-O-B-B-Y?
SPEAKER_00:
SPEAKER_01: Yeah.

Figure 10: Diarization without LM. There are labels that are empty.

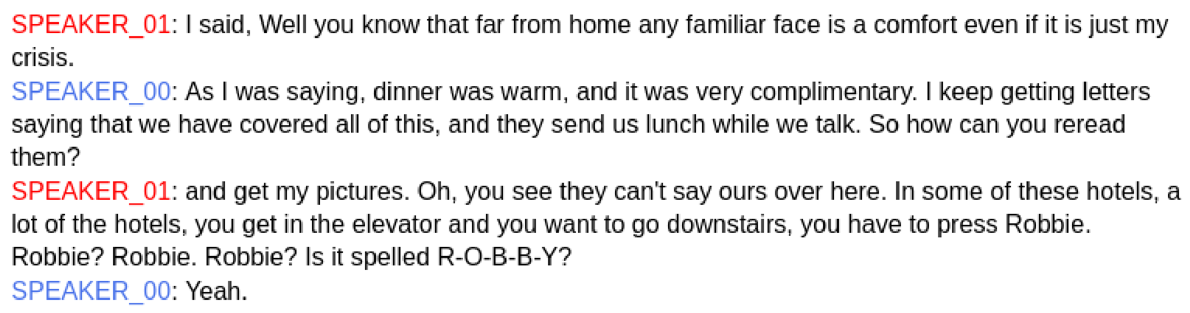

SPEAKER_01: I said, Well you know that far from home any familiar face is a comfort even if it is just my crisis.
SPEAKER_00: As I was saying, dinner was warm, and it was very complimentary. I keep getting letters saying that we have covered all of this, and they send us lunch while we talk. So how can you reread them?
SPEAKER_01: and get my pictures. Oh, you see they can't say ours over here. In some of these hotels, a lot of the hotels, you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? Is it spelled R-O-B-B-Y?
SPEAKER_00: Yeah.

Figure 11: Example result using GPT-3.5 Turbo with a zero-shot prompt. The problem was solved: there are no misplaced words and no empty speakers' dialogue.

The runtime for one sample was about 3.5 minutes, but the overall process took about 23 hours. This extended duration was due to numerous queries to the LLM, and Whisper takes longer with larger audio files. It is important to mention that not all datasets have the same length.

In Table 6 are the results and metrics.

	WDER	cpWER	Diarization Purity	Diarization Coverage
without_LM	0.0	0.730058	0.407282	0.340947
35_turbo_zero_shot	0.0	0.792664	0.600697	0.550390
35_turbo_one_shot	0.0	0.739704	0.469220	0.395633
4_zero_shot	0.0	0.721414	0.522158	0.467135
4_one_shot	0.0	0.721169	0.566625	0.508129

Table 6: Performance Metrics for Different Model Configurations.

From WDER, which is consistently 0, it indicates that all models perform equally well on this task. For further research, we can delve deeper into larger datasets and explore variations in prompts to observe any potential changes.

cpWER indicates how well the word recognition accuracy improves after diarization corrections: lower values indicate better model performance. The best cpWER was achieved by GPT-4, which performed almost equally well with both zero-shot and one-shot prompts. The worst performance was observed with GPT-3.5 Turbo using the zero-shot prompt. This metric should ideally be minimized for relevance.

Diarization Purity measures how accurately segments are assigned to the correct speakers, with higher values indicating better performance. GPT-3.5 Turbo with the zero-shot prompt achieved a value nearly 0.04

Diarization Coverage measures the proportion of speech correctly attributed to speaker segments, with higher values indicating better performance. Once again, GPT-3.5 Turbo with the zero-shot prompt outperformed GPT-4 by almost 0.04

Based on these results, the winner of this experiment is GPT-3.5 Turbo with the zero-shot prompt. Future experiments will focus on comparing GPT-4 with the one-shot prompt against GPT-3.5 Turbo with the zero-shot prompt.

6.3 Case Studies

In this section, we delve into some of the most compelling results from our Speaker Diarization experiments, focusing on both exemplary performances and areas where improvements are still needed. The cornerstone of our comparisons lies in evaluating diarization outcomes with and without the integration of a Language Model (LM).

Our methodology revolves around leveraging advanced tools such as Pyannote for initial speaker segmentation and Whisper for precise Speech-to-Text (STT) conversion. These processes are enhanced significantly through the incorporation of sophisticated Large Language Models (LLMs), specifically GPT-3.5 Turbo and GPT-4, known for their proficiency in natural language understanding and generation tasks.

The primary objective of this study is to demonstrate how integrating LLMs can refine the accuracy of speaker segmentation and transcription in various speech scenarios. By examining outcomes across different configurations—diarization without LM, GPT-3.5 Turbo with zero-shot and one-shot prompts, and GPT-4 with similar configurations—we aim to highlight the impact of these models on overall diarization quality.

Throughout our analysis, we identify key performance metrics such as Word Error Rate (WER), Corrected Word Error Rate (cpWER), Diarization Purity, and Diarization Coverage. These metrics serve as benchmarks to assess the efficacy of each configuration in achieving accurate, contextually relevant speaker segmentation and transcription. The findings not only underscore the capabilities of each model but also inform future strategies for optimizing real-time and offline diarization processes.

By presenting contrasting examples of both successful and challenging diarization instances, we illustrate the potential benefits and limitations of integrating advanced LLMs into speech processing pipelines. This exploration sets the stage for deeper insights into how these technologies can be harnessed to enhance efficiency and accuracy in diverse speech analytics applications.

6.3.1 cpWER Results

The analysis will start with cpWER, because the WDER was 0% overall. This example was the best with a cpWER value of 0.268917 which was achieved by GPT-4 with zero-shot on the 66th conversation and the worst value was 1.768859 for the 76th.

The table from Appendix C presents four examples of diarization outputs along with their corresponding cpWER (Corrected Word Error Rate) values, which measure the accuracy of word recognition after applying diarization corrections. Here's an analysis of the differences observed in these examples:

1. Best cpWER = 0.268917 (GPT-4 zero-shot):
   - Diarization Output: The dialogue is correctly segmented and attributed to the respective speakers without significant errors. The conversation flows logically and coherently.
   - Observation: This example demonstrates high accuracy in speaker segmentation and word recognition, indicated by the low cpWER. It shows that GPT-4 with a zero-shot prompt effectively manages speaker turns and maintains context well all the transcript.
2. Worst cpWER = 1.768859 (GPT-3.5 Turbo zero-shot):
   - Diarization Output: There are noticeable errors such as misplaced words and unclear speaker transitions. The conversation lacks coherence and may confuse the listener.
   - Observation: This higher cpWER indicates poorer performance in diarization accuracy. GPT-3.5 Turbo with a zero-shot prompt struggles more with segmenting speakers correctly and maintaining the flow of conversation compared to GPT-4.

Analysis of Differences

- Accuracy in Speaker Segmentation: Examples with lower cpWER values (like GPT-4 zero-shot) show better segmentation of dialogue into distinct speaker turns. This is crucial for maintaining context and clarity in transcriptions.
- Word Recognition: Lower cpWER values also reflect better word recognition accuracy after considering diarization corrections. This is essential for ensuring that transcribed text accurately reflects what was spoken.
- Impact on Overall Quality: Higher cpWER values (like GPT-3.5 Turbo zero-shot) indicate more errors in segmentation and transcription, potentially leading to misunderstandings or inaccuracies in the final output.

In conclusion, cpWER serves as a critical metric for evaluating the effectiveness of diarization systems in accurately transcribing multi-speaker conversations. Lower cpWER values signify better performance, ensuring that transcriptions are faithful to the spoken dialogue and enhancing the overall usability of automated speech processing systems.

6.3.2 DiarizationPurity Results

The best Diarization Purity was achieved by a lot of contestants, [9, 11, 13, 16, 20, 27, 28, 30, 32, 42, 43, 45, 50, 51, 52, 53, 57, 65, 69, 72, 74, 75, 76, 78, 82, 87, 90, 91, 92, 93, 97, 99, 100, 102, 110, 118, 120, 122, 127, 129], with the maximum value of 1.0. On the other hand, the worst value was 0.384804 achieved by 28th data. We will put as an example the first occurrence, the 9th. Both of the values were achieved by 3.5 Turbo zero-shot.

The table from Appendix D compares diarization outputs based on their Diarization Purity values, highlighting significant differences in accuracy and segmentation quality:

1. Best Diarization Purity = 1.0 (GPT-3.5 Turbo zero-shot):
    - Diarization Output: The segmentation accurately assigns each segment of speech to the correct speaker with perfect purity. The conversation flows smoothly and maintains clarity throughout.
    - Observation: This example demonstrates the highest level of accuracy in speaker segmentation, achieving perfect purity where every segment is correctly attributed without any errors. This level of precision is crucial for applications requiring exact speaker identification, such as in transcribing interviews or meetings.
2. Worst Diarization Purity = 0.384804 (GPT-3.5 Turbo zero-shot):
    - Diarization Output: There are noticeable issues in segmentation, with segments of speech incorrectly attributed or unclear transitions between speakers. This results in a lower purity score, indicating significant errors in speaker identification.

- Observation: The lower Diarization Purity score in this example suggests challenges in correctly identifying and segmenting speakers. Such inaccuracies can lead to confusion in transcriptions and affect the overall usability of automated speech processing systems.

Analysis of Results

- Impact of Diarization Purity: Higher purity values (like 1.0) reflect precise and accurate speaker segmentation, ensuring that each speaker's segments are correctly identified without overlap or errors. This enhances the reliability and usefulness of transcriptions. Challenges and Adjustments: Lower purity values (such as 0.384804) indicate areas where improvements are needed in diarization algorithms or prompts used with GPT-3.5 Turbo. Adjusting prompts tailored to specific conversation contexts could potentially improve segmentation accuracy and overall performance.

In conclusion, the Diarization Purity metric provides critical insights into the effectiveness of automated systems in correctly segmenting and attributing speech to speakers. Addressing these challenges through prompt adjustments and algorithm refinements can significantly enhance the reliability and usability of automated speech processing technologies.

6.3.3 Diarization Coverage

The best Diarization Coverage was achieved by several audios, [9, 11, 13, 20, 27, 28, 32, 42, 43, 50, 51, 52, 53, 57, 65, 69, 74, 75, 76, 78, 82, 87, 90, 91, 92, 97, 99, 100, 102, 110, 118, 120, 122, 127, 129], with perfect score of 1.0, The worst was achieved by [6, 40, 61, 77, 83, 132] with the value of 0.33333. Both of them by GPT-3.5 Turbo zero-shot. For the best, we will talk about 9 and the worst about 6. The tables for the best and the worst Diarization Coverage can be found at Appendix E.

1. Best Diarization Coverage = 1.0 (GPT-3.5 Turbo zero-shot):
   - Diarization Output: The segmentation accurately covers all speech segments, correctly attributing each segment to the respective speakers without omission. The dialogue is well-structured and coherent throughout.
   - Observation: This example achieves a perfect Diarization Coverage score, indicating that every portion of the speech is correctly assigned to a speaker segment. This high level of accuracy ensures comprehensive coverage of the conversation, which is crucial for generating accurate transcriptions and maintaining context.
2. Worst Diarization Coverage = 0.333333 (GPT-3.5 Turbo zero-shot):
   - Diarization Output: There are segments of speech that are not correctly attributed to speakers, leading to gaps and inaccuracies in segment coverage. The dialogue lacks clarity and may be difficult to follow.

- Observation: The lower Diarization Coverage score in this example suggests significant issues in correctly segmenting and attributing speech to speakers. This results in gaps where segments of conversation are not properly assigned, impacting the overall coherence and comprehensiveness of the transcript.

Analysis of Results

- Impact of Diarization Coverage: A higher Diarization Coverage score (like 1.0) indicates robustness in accurately segmenting and covering all speech segments within the conversation. This ensures that no parts of the dialogue are missed or incorrectly assigned, enhancing the reliability of automated transcription systems.
- Challenges and Adjustments: Lower Diarization Coverage scores (such as 0.333333) highlight areas where improvements are needed in diarization algorithms or the prompts used with GPT-3.5 Turbo. Adjusting prompts to better fit the conversation context and refining segmentation techniques could potentially improve coverage accuracy.

In conclusion, Diarization Coverage serves as a critical metric for evaluating the effectiveness of automated systems in segmenting and attributing speech segments to speakers. Addressing these challenges through prompt adjustments and algorithm refinements can significantly enhance the reliability and usability of automated speech processing technologies.

7 CONCLUSIONS

This paper presents a robust pipeline integrating Pyannote for initial speaker segmentation, Whisper for precise Speech-to-Text (STT) transcription, and advanced Large Language Models (LLMs) such as GPT-3.5 Turbo and GPT-4 to enhance Speaker Diarization accuracy. The experimental results demonstrate overall strong performance, with GPT-3.5 Turbo emerging as particularly effective across key metrics as Diarization Purity and Coverage. Notably, GPT-4 excels in reducing the concatenated minimum-permutation word error rate (cpWER), showcasing its proficiency in accurate transcription tasks. These outcomes validate the effectiveness of integrating state-of-the-art LLMs in enhancing both word-level accuracy and speaker-level segmentation in diarization scenarios.

Looking forward, optimizing the pipeline for offline processing involves initiatives like preloading audio files into memory for Pyannote, aimed at enhancing efficiency. While Automatic Speech Recognition (ASR) processes were deemed satisfactory in terms of processing time for audio chunks, online processing of LLM interrogation highlighted challenges. Future strategies may involve tailoring prompts for each transcript to mitigate irrelevant outputs from the LLM, as indicated in Section 6.3 of the study.

Moving ahead, the focus shifts towards advancing live diarization capabilities to surpass current offline methodologies. This strategic direction aims to leverage real-time insights for improved accuracy and responsiveness in capturing and segmenting speaker interactions. By addressing these areas for refinement and future research, this study contributes to the ongoing evolution of Speaker Diarization technologies, promising more effective tools for speech processing and analysis across diverse applications and environments.

BIBLIOGRAPHY

[1] Alexis Plaquet and Hervé Bredin. Powerset multi-class cross entropy loss for neural speaker diarization. In *Proc. INTERSPEECH 2023*, 2023.

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

[3] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. arXiv.org perpetual, non-exclusive license.

[4] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2023.

[5] Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Biplav Srivastava, Lior Horesh, Francesco Fabiano, and Andrea Loreggia. Understanding the capabilities of large language models for automated planning, 2023.

[6] A. Canavan, D. Graff, and G. Zipperlen. Callhome american english speech ldc97s42, 1997. LDC Catalog.

[7] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J Han, Shinji Watanabe, and Shrikanth Narayanan. A review of speaker diarization: Recent advances with deep learning. *Computer Speech & Language*, 72:101317, 2022.

[8] Chao Zhang and Quan Wang. Speaker diarization: A journey from unsupervised to supervised approaches. In *Odyssey: The Speaker and Language Recognition Workshop*, 2022. Tutorial session.

[9] Hernawan Sulistyanto. Speaker diarization: Its developments, applications, and challenges. https://core.ac.uk/download/pdf/11734645.pdf, year not specified. School of Computer Science and Software Engineering, University of Wollongong, NSW, Australia, and Informatics Engineering, Muhammadiyah University of Surakarta, Central Java, Indonesia.

[10] Jiaming Wang, Zhihao Du, and Shiliang Zhang. Told: A novel two-stage overlap-aware framework for speaker diarization, 2023.

[11] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Paola Garcia. Encoder-decoder based attractors for end-to-end neural diarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021. Accepted to IEEE/ACM TASLP. This article is based on our previous conference paper arXiv:2005.09921.

[12] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. `https://librosa.org/doc/latest/index.html`, latest version accessed 2024. Accessed: 2024-06-24.

[13] C.-A. Deonise, T.-M. Coconu, T. Rebedea, and F. Pop. Improved speech activity detection model using convolutional neural networks. In *2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–8, Craiova, Romania, 2023.

[14] S. Sharma, P. Rattan, and A. Sharma. Recent developments, challenges, and future scope of voice activity detection schemes—a review. In M.S. Kaiser, J. Xie, and V.S. Rathore, editors, *Information and Communication Technology for Competitive Strategies (ICTCS 2020)*, volume 190 of *Lecture Notes in Networks and Systems*. Springer, Singapore, 2021.

[15] S. W. Chin, K. P. Seng, L.-M. Ang, and K. H. Lim. Improved voice activity detection for speech recognition system. In *2010 International Computer Symposium (ICS2010)*, pages 518–523, Tainan, Taiwan, 2010.

[16] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, Patrice Castonguay, Mariya Popova, Jocelyn Huang, and Jonathan M. Cohen. Nemo: a toolkit for building ai applications using neural modules. https://arxiv.org/abs/1909.09577, 2019. Accessed: 2024-06-24.

[17] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. Whisperx: Time-accurate speech transcription of long-form audio. *INTERSPEECH 2023*, 2023.

[18] Fan Yu, Shiliang Zhang, Yihui Fu, Lei Xie, Siqi Zheng, Zhihao Du, Weilong Huang, Pengcheng Guo, Zhijie Yan, Bin Ma, Xin Xu, and Hui Bu. M2MeT: The icassp 2022 multi-channel multi-party meeting transcription challenge. https://arxiv.org/abs/2203.09877, 2022. Accessed: 2024-06-24.

[19] Quan Wang, Yiling Huang, Guanlong Zhao, Evan Clark, Wei Xia, and Hank Liao. Diarizationlm: Speaker diarization post-processing with large language models. arXiv:2401.03506, 2024.

[20] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Paola Garcia. Encoder-decoder based attractors for end-to-end neural diarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30(2325-9285):1152–1164, 2022. Submitted on 20 Jun 2021 (v1), last revised 28 Mar 2022 (v2).

[21] Hugging Face. pyannote/speaker-diarization-3.1: PyAnnote Speaker Diarization Toolkit v3.1. `https://huggingface.co/pyannote/speaker-diarization-3.1`, 2023. Accessed: 2024-06-24.

[22] OpenAI. OpenAI Whisper Large v3. `https://huggingface.co/openai/whisper-large-v3`, 2023. Accessed: 2024-06-24.

[23] OpenAI. OpenAI API Documentation. `https://platform.openai.com/docs/api-reference/chat/create`, 2024. Accessed: 2024-06-24.

[24] PyAnnote. PyAnnote Metrics Documentation. `https://pyannote.github.io/pyannote-metrics/reference.html#purity-and-coverage`, 2024. Accessed: 2024-06-24.

[25] Laurent El Shafey, Hagen Soltau, and Izhak Shafran. Joint speech recognition and speaker diarization via sequence transduction. In *Proc. Interspeech*, pages 396–400, 2019.

[26] Shinji Watanabe, Michael Mandel, Jon Barker, Emmanuel Vincent, Ashish Arora, Xuankai Chang, Sanjeev Khudanpur, Vimal Manohar, Daniel Povey, Desh Raj, and et al. Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. *arXiv preprint arXiv:2004.09249*, 2020.

A GITHUB CODE

https://github.com/Maxxtra/Offline-Diarization-with-LM.git

B DIARIZATION SCENARIOS

Without Diarization

diarization_without_LM	**SPEAKER_01**: I said, well you know, that far away from home, any familiar face is a comfort even if it is just Mike Price's. **SPEAKER_00**: SPEAKER_01: Anyway, well, it was fun. The food was great, it was catered, it was just wonderful. Wow! And I won't tell you anything, because it's all in your letter. Ok, yeah, I notice that every time I get a letter, I say, Well, we've covered all this and it's still good to get the letters. Yeah, you can re-read them. Re-read them, yeah, I may have. Re-read them, Rory. SPEAKER_00: SPEAKER_01: and get my pictures. Yeah. Oh, you see, they can't say ours over here. In some of these hotels, a lot of the hotels, when you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? It's spelled R-O-B-B-Y? SPEAKER_00: SPEAKER_01: yup what i told the first time to get your room number

Diarization 3.5 Turbo

35_turbo_zero_shot	**SPEAKER_01**: I said, well you know, that far away from home, any familiar face is a comfort even if it is just Mike Price's. Anyway, well, it was fun. The food was great, it was catered, it was just wonderful. Wow! And I won't tell you anything, because it's all in your letter. Ok, yeah, I notice that every time I get a letter, I say, Well, we've covered all this and it's still good to get the letters. Yeah, you can re-read them. Re-read them, yeah, I may have. Re-read them, Rory. **SPEAKER_00**: And get my pictures. **SPEAKER_01**: Yeah. Oh, you see, they can't say ours over here. In some of these hotels, a lot of the hotels, when you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? It's spelled R-O-B-B-Y? **SPEAKER_00**: Yup, what I told the first time to get your room number.

35_turbo_one_shot	**SPEAKER_01**: I said, well you know, that far away from home, any familiar face is a comfort even if it is just Mike Price's. Anyway, well, it was fun. The food was great, it was catered, it was just wonderful. Wow! And I won't tell you anything, because it's all in your letter. Ok, yeah, I notice that every time I get a letter, I say, Well, we've covered all this and it's still good to get the letters. Yeah, you can re-read them. Re-read them, yeah, I may have. Re-read them, Rory. **SPEAKER_00**: Ha ha ha! **SPEAKER_01**: Uh... advocate uncomplacent what's the name for the players and he said mister element Bedford while dowell they have a problem with that out um... head emphasize that you can do back into words that i am finally applications for jogging try to get them can have an official monday yeah where you have a viewpint of you about there Alexi, where are y'all meeting tonight? I'll tell ya we disguised at a party!

Diarization 4

4_zero_shot	**SPEAKER_01**: I said, well you know, that far away from home, any familiar face is a comfort even if it is just Mike Price's. Anyway, well, it was fun. The food was great, it was catered, it was just wonderful. Wow! And I won't tell you anything, because it's all in your letter. Ok, yeah, I notice that every time I get a letter, I say, Well, we've covered all this and it's still good to get the letters. Yeah, you can re-read them. Re-read them, yeah, I may have. Re-read them, Rory. **SPEAKER_00**: And get my pictures. **SPEAKER_01**: Yeah. Oh, you see, they can't say ours over here. In some of these hotels, a lot of the hotels, when you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? It's spelled R-O-B-B-Y? SPEAKER_00: SPEAKER_01: yup what i told the first time to get your room number. Ha ha ha!

4_one_shot	**SPEAKER_01**: I said, well you know, that far away from home, any familiar face is a comfort even if it is just Mike Price's. Anyway, well, it was fun. The food was great, it was catered, it was just wonderful. Wow! And I won't tell you anything, because it's all in your letter. Ok, yeah, I notice that every time I get a letter, I say, Well, we've covered all this and it's still good to get the letters. Yeah, you can re-read them. Re-read them, yeah, I may have. Re-read them, Rory. **SPEAKER_00**: **SPEAKER_01**: and get my pictures. Yeah. Oh, you see, they can't say ours over here. In some of these hotels, a lot of the hotels, when you get in the elevator and you want to go downstairs, you have to press Robbie. Robbie? Robbie. Robbie? It's spelled R-O-B-B-Y?

C CPWER RESULTS

cpWER Value	Diarization without LM	Diarization with GPT-4 zero-shot
0.268917	SPEAKER_01: Is the program geared primarily toward beginning level of Spanish speakers? No, we offer seven different levels ranging from true beginners, people who know absolutely nothing about Spanish, all the way up through advanced in literature levels. SPEAKER_00: and SPEAKER_01: So we have something for everybody. SPEAKER_00: SPEAKER_01: So we would actually do a two-part test. And so after you arrive, we would give you a placement test. It's a two-part test. The first part is written. And it would test you on present tense, direct and indirect	SPEAKER_01: Is the program geared primarily toward beginning level of Spanish speakers? SPEAKER_00: No, we offer seven different levels ranging from true beginners, people who know absolutely nothing about Spanish, all the way up through advanced in literature levels. So we have something for everybody. So we would actually do a two-part test. And so after you arrive, we would give you a placement test. It's a two-part test. The first part is written. And it would test you on present tense, direct and indirect objects.
1.768859	SPEAKER_01: Oh so you can get involved in that, yeah that's good for you sure. SPEAKER_00: So you can get involved in that. Yeah, that's good for you. SPEAKER_01: and they humdemb in forgotten would you like they've gaston man date abroad than another party-time SPEAKER_00: oh they're going to be in it SPEAKER_01: lillium agenda appellate say that they're coming into sunday uh... SPEAKER_00: rebellion and	SPEAKER_01: Oh so you can get involved in that, yeah that's good for you sure. SPEAKER_00: So you can get involved in that. Yeah, that's good for you. SPEAKER_01: and they humdemb in forgotten would you like they've gaston man date abroad than another party-time. SPEAKER_00: oh they're going to be in it. SPEAKER_01: lillium agenda appellate say that they're coming into sunday uh... SPEAKER_00: rebellion and.

Table 7: Comparison of Diarization outputs with cpWER values.

D DIARIZATION PURITY RESULTS

Diarization Purity Value	Diarization without LM	Diarization with GPT-3.5 Turbo zero-shot
1.0	SPEAKER_01: U seafood will load who appreciated the SPEAKER_00: SPEAKER_01: it would have a photo HereI have a picture with a player who had the reserve the other day rather than a month SPEAKER_00: I don't know. SPEAKER_01: If you take a picture with me You know I'll send it to you Thank you very much Then I'll take another picture SPEAKER_00: SPEAKER_01: Let's see. SPEAKER_00: SPEAKER_01: Well, I'm still working there and I'm on a pretty good project. SPEAKER_00:	SPEAKER_01: U seafood will load who appreciated the SPEAKER_01: it would have a photo Here I have a picture with a player who had the reserve the other day rather than a month SPEAKER_01: If you take a picture with me You know I'll send it to you Thank you very much Then I'll take another picture SPEAKER_01: Let's see. SPEAKER_01: Well, I'm still working there and I'm on a pretty good project.
0.384804	SPEAKER_01: Wow, very nice. It's really sweet. Fast Yeah, it is. SPEAKER_00: SPEAKER_01: Hm, wow It certainly is So, how's your family? Yeah everybody's cool man I went home And how's your dad? He's uh... SPEAKER_00: SPEAKER_01: It's not it's not that serious, uh-huh, but let's just say you won't ever be eating salt as much as you used to before SPEAKER_00: SPEAKER_01: Right, right. SPEAKER_00: You SPEAKER_01: And a thank god he's not one of those	SPEAKER_01: Wow, very nice. It's really sweet. Fast Yeah, it is. SPEAKER_00: SPEAKER_01: Hm, wow It certainly is So, how's your family? Yeah everybody's cool man I went home And how's your dad? He's uh... SPEAKER_00: SPEAKER_01: It's not it's not that serious, uh-huh, but let's just say you won't ever be eating salt as much as you used to before SPEAKER_00: SPEAKER_01: Right, right. SPEAKER_00: You SPEAKER_01: And a thank god he's not one of those

Table 8: Comparison of diarization outputs with Diarization Purity values.

E DIARIZATION COVERAGE RESULTS

Diarization Coverage Value	Diarization without LM	Diarization with GPT-3.5 Turbo zero-shot
1.0	SPEAKER_01: U seafood will load who appreciated the SPEAKER_00: SPEAKER_01: it would have a photo Herel have a picture with a player who had the reserve the other day rather than a month SPEAKER_00: I don't know. SPEAKER_01: If you take a picture with me You know I'll send it to you Thank you very much Then I'll take another picture SPEAKER_00: SPEAKER_01: Let's see. SPEAKER_00: SPEAKER_01: Well, I'm still working there and I'm on a pretty good project. SPEAKER_00: SPEAKER_01: along wrong term next generation uh... fighter plane so that uh... entitled experience again and uh... westinghouse supposedly is making a bid to buyterious cbs broadcasting network reporter marie SPEAKER_00: SPEAKER_01: Well, you know, it takes a week to extend the deal, so...	SPEAKER_01: U seafood will load who appreciated the SPEAKER_01: it would have a photo Here I have a picture with a player who had the reserve the other day rather than a month SPEAKER_01: If you take a picture with me You know I'll send it to you Thank you very much Then I'll take another picture SPEAKER_01: Let's see. SPEAKER_01: Well, I'm still working there and I'm on a pretty good project. SPEAKER_01: along wrong term next generation uh... fighter plane so that uh... entitled experience again and uh... westinghouse supposedly is making a bid to buy CBS broadcasting network reporter Marie SPEAKER_01: Well, you know, it takes a week to extend the deal, so...

Table 9: Comparison of diarization outputs with Diarization Coverage values.

Diarization Coverage Value	Diarization without LM	Diarization with GPT-3.5 Turbo zero-shot
0.333333	SPEAKER_01: send dining good not beautifully big thank they do that is not the kind of they can bring it into any way comp gap incurred junya reacting advance SPEAKER_00: Well. SPEAKER_01: Which I think is more Persian than the Mediterranean. I used that all the time still. It's kind of amusing. Things never change. SPEAKER_00: SPEAKER_01: yes I do. so what is happening with you and sharon? well we are here in Italy in Vela Verla which is this little basically nowhere town about half an hour from post and uh its gorgeous I look out the window I look at the Alps um everywhere you look its just like little red tiled roofs just like you see in the postcards in the picture books. It's awesome. It's amazing. So we live here in this third floor apartment, go into work everyday and then come back. On the weekends we try to get out and go places, see the country. Getting a little more difficult as of late, we're kind of busy over here, as I'm sure you saw on the news. The whole...anyway, we're busy. Let me put it that way.	SPEAKER_01: send dining good not beautifully big thank they do that is not the kind of they can bring it into any way comp gap incurred junya reacting advance SPEAKER_00: Well. SPEAKER_01: Which I think is more Persian than the Mediterranean. I used that all the time still. It's kind of amusing. Things never change. SPEAKER_00: SPEAKER_01: yes I do. so what is happening with you and sharon? well we are here in Italy in Vela Verla which is this little basically nowhere town about half an hour from post and uh its gorgeous I look out the window I look at the Alps um everywhere you look its just like little red tiled roofs just like you see in the postcards in the picture books. It's awesome. It's amazing. So we live here in this third floor apartment, go into work everyday and then come back. On the weekends we try to get out and go places, see the country. Getting a little more difficult as of late, we're kind of busy over here, as I'm sure you saw on the news. The whole...anyway, we're busy. Let me put it that way.

Table 10: Comparison of diarization outputs with Diarization Coverage values.