Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Mutian Li Date

Joint Text and Audio Multi-modal Speaker Diarization

By

Mutian Li
Master of Science

Computer Science and Informatics

Jinho D. Choi, Ph.D.
Advisor

Wei Jin, Ph.D.
Committee Member

Hyeokhyen Kwon, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Joint Text and Audio Multi-modal Speaker Diarization

By

Mutian Li
B.E., ShanghaiTech University, China, 2023

Advisor: Jinho D. Choi, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Computer Science and Informatics
2025

Abstract

Joint Text and Audio Multi-modal Speaker Diarization
By Mutian Li

Speaker diarization is a speech processing task that aims to determine the timeline and content of each speaker. Despite significant advances in deep learning, speaker diarization continues to be challenging, particularly in scenarios involving short utterances and noisy audio environments.

This thesis analyzes the frequent error types of audio-based speaker diarization models and proposes a novel approach to addressing these errors using GPT-4o. Then, we examine whether speech segment clustering is a feasible choice. Finally, we introduce a novel approach of fine-tuning a speech recognition model, Whisper[18], for the speaker diarization task.

Our research begins with the Trauma Interview dataset, collected by Emory Health, utilizing annotations generated by the Azure tool. The speaker label error rate of the Azure annotations is estimated to be 7.35%. Open-source models perform even worse; for instance, the speaker label error rate of Pyannote Powerset[17] on the Trauma Interview dataset is 9.36%. By leveraging GPT-4o for error correction, the final speaker label error rate is reduced to an estimated 4.26%. Although the proposed fine-tuned Whisper model does not outperform the baseline, it shows potential for improvement with further refinement.

Experiments are conducted not only on Trauma Interview data from our laboratory but also on publicly available datasets, including AMI and DailyTalk.

The findings of this research highlight the limitations of current speaker diarization research, and provide a direction for future research.

Joint Text and Audio Multi-modal Speaker Diarization

By

Mutian Li
B.E., ShanghaiTech University, China, 2023

Advisor: Jinho D. Choi, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Computer Science and Informatics
2025

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Jinho Choi, for supporting and guiding my research in Emory NLP. He answers my questions and clarifies my confusions, and helps me form my own research ideas. He has taught me essential research skills, such as conducting a thorough literature survey and performing error analysis, which will be helpful in my future work. He also encourages and motivates me to explore deeper and wider in speech processing areas.

I also want to thank my thesis committee members, Wei Jin and Hyeokhyen Kwon, for their valuable time and effort. Their questions have guided me towards a deeper understanding and better explanation, and their suggestions have helped a lot in refining and improving my work.

I would like to acknowledge my parents, Guoqing Li and Lili Mu, who have provided financial and mental support, allowing me to study at Emory University without additional worries. They have encouraged me to explore my own academic journey.

Finally, I would like to thank my colleagues for their support. Their discussions and motivations have helped me endure the hardest and most challenging times during my graduate studies. Off-campus activities with them, such as traveling and singing, have added joy and enrichment to my graduate life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Speaker Diarization (SD) addresses "who speaks when" question during speech processing. For text-based Speaker Diarization, we also need to determine "who speaks what". Downstream applications, such as dialogue analysis, often prioritize the identification of "who speaks what" over "who speaks when." In this thesis, we conduct a series of experiments to identify the optimal system for answering the "who speaks what" question.

1.1 Motivation

The motivation for researching this topic is to process data intended for training a trauma chatbot (referred to as the Trauma Interview dataset in following chapters). Our raw data consists of audio recordings of interviews between psychologists and trauma patients. The required data format is transcript with speaker labels, which necessitates the use of both Automatic Speech Recognition (ASR) and Speaker Diarization. Although ASR models, bolstered by advancements in Large Language Models (LLMs), can generate high-quality transcripts, Speaker Diarization task remains challenging.

Initially, we use the paid Azure tool to generate transcripts with speaker labels. For the ASR task, Whisper[18], an open-source model developed by OpenAI, outperforms Azure. For the Speaker Diarization task, Azure outperforms current open-source models. However,

Azure is expensive and still has fatal errors, especially in case of short utterances. Below is an example from the Azure annotation of Trauma Interview dataset:

```
Speaker1:  Have you ever had a period of time in your life when you
felt like super hyper high, full of energy?
Speaker1:  No.  No.  OK. Have there ever been times in your life when
you've felt like that persistent irritability over several days?  No.
```

In the conversation above, apparently, "no" should be labeled as another speaker, like "Speaker2". However, in Azure's annotation, all utterances are incorrectly labeled as "Speaker1." This is because audio-based Speaker Diarization models rely on speaker embeddings. However, speaker embeddings are only precise on utterances longer than 3 seconds, while a short sentence like 'no' typically takes about 0.5 second. Another cause of this error is the misalignment between Speaker Diarization and ASR result.

Current LLMs, such as ChatGPT, can partially detect and fix this kind of error. Consequently, in the following chapters, we will investigate whether language models can enhance the accuracy of speaker diarization results.

1.2 Overview

In this thesis, we explore multi-modal speaker diarizaion models which leverage both text and audio to improve speaker diarization performance. We hypothesize that the introduction of semantic information can help improve the accuracy of speaker diarization (SD), as diarization errors such as 1.1 are semanticly apparent. We first apply several state-of-the-art models and perform error analysis. Then, we try improving diarization performance with several methods, such as prompting ChatGPT, and clustering based on ASR segments. Finally, we modify an ASR model to incorporate SD results, enabling it to generate transcripts with speaker labels.

Our main contribution: 1. While most systems align speaker diarization results with transcripts by word-level timeline, we demonstrate that aligning with sentence-level timeline achieves a better performance. 2. We analyze and summarize common mistakes of SOTA models, which facilitate targeted training in future researches. 3. We propose a novel approach to improve speaker diarization result by combining diarization result of 2 models, with relatively less ChatGPT api call. 4. We propose a novel idea of adapting pretrained speech recognition model to fix speaker diarization errors and generate accurate transcript.

1.3 Research Questions

In this thesis, we address the following three research questions:

1. Why are current speaker diarization models insufficient for the Trauma Interview dataset?

This question examines the limitations of Azure and other open-sourced speaker diarization models. In Chapter 3, we provide an evaluation of these models on our dataset, along with a detailed error analysis.

2. Can Azure annotation on the Trauma Interview dataset be improved using audio and text models?

To address this question, we first explore how GPT can help improve the accuracy of speaker labels in Azure annotation. Then, we explore whether multi-modal methods achieve better performance on this dataset. Details see Chapter 4.

3. Can fine-tuning an ASR model improve speaker diarization performance across different datasets?

The method proposed in Chapter 4 does not perform well on public datasets, which pushes us to explore more generalizable approaches. Specifically, we investigate whether the Whisper [18] model can be fine-tuned for speaker diarization. The system design details and evaluation are illustrated in Chapter 5.

1.4 Thesis Statement

By leveraging large language models, we can correct a portion of speaker labels and enhance speaker diarization results. Additionally, developing a specialized joint ASR and SD model which takes in multi-modal input has the potential to improve the quality of transcripts with speaker labels.

Chapter 2

Background

2.1 Related Works

2.1.1 Audio-based Speaker Diarization Models

Audio-based speaker diarization models decide 'who speaks when' via voice features. They generally fall into two types: Clustering-based and End-to-End Neural Diarization (EEND).

Clustering-based methods usually consist 3 steps: 1. Using a voice activity detection (VAD) model to detect homogeneous speech segments; 2. Using a speaker embedding model to obtain embeddings of speech segments; 3. Applying clustering to speaker embeddings. One advantage of Clustering-based methods is that the system is cheap and easy to implement, as both VAD and speaker embedding steps can utilize pretrained models. Besides, each step can be optimized independently to achieve a better performance.

However, naive clustering-based methods struggle in handling overlapping speech. Therefore, recent clustering-based methods, such as [9][2][17], use EEND models in the first step to mediate this problem. For example, at step 1, [2] applies an EEND model (proposed in [30]) for local speaker diarization on sliding windows (see figure2.1). The EEND contains BiLSTM layers followed by three binary classification heads, each corresponding to one speaker (assuming the number of speakers within a sliding window is at most three).

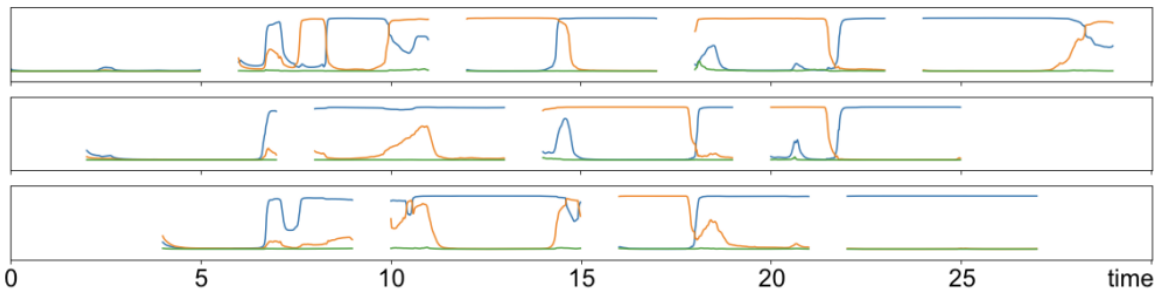

Figure 2.1: Local neural speaker segmentation in pyannote[2]. Different colors corresponds to different local speakers.

Then, generate embeddings on segments that contain the same predicted speaker label for each sliding window, cluster those embeddings, and vote to aggregate the final results. [17] follows the same structure as [2], except that the classification heads are replaced with a powerset classification head (referred to as Pyannote Powerset in the following chapters).

EEND models were first introduced in [30]. Unlike clustering-based methods, EEND models output speaker diarization results directly from a neural network. These models have separate classification heads for different speakers, and utilize permutation-invariant training for speaker labels. Each classification head performs binary classification to determine whether a speaker is active at a given time. Therefore, EEND models can predict overlapping speakers.

Recently, to handle a flexible number of speakers, EEND models ([10][22][14]) also predict an inner speaker embedding called an attractor. Attractors are predicted by a sequential decoder from audio encoding; each attractor corresponds to one speaker. Then, similarities are calculated between the attractors and audio frame encoding by cross-production, or any other suitable method, to help decide whether the corresponding speakers are active on the frame.

Different EEND models usually have their unique structures, which makes it difficult to use pre-trained models. Therefore, compared to clustering-based methods, training an EEND model is extremely data-hungry. Due to the lack of real-world speaker diarization

data, training an EEND model requires a huge amount of synthetic data. Simulating real-world dialogues has been a longstanding research topic([13]); however, a distribution gap between synthetic and real-world data persists, leading to degraded performance in EEND models.

2.1.2 Automatic Speech Recognition Models

In our experiments, all the transcripts are generated by Whisper[18] model. Whisper consists of an audio encoder and a text decoder. In the audio encoder, the audio log-mel spectrogram is passed to two 1-D convolution layers followed by a transformer encoder with sinusoidal positional encoding. The audio embedding is then passed to a text decoder, which is a transformer decoder with learnable positional encodings. The overall architecture is illustrated in Figure2.2.

2.1.3 Joint ASR and SD Models

Compared to training each task separately, multi-task training offers several advantages. First, it enables the model to capture more general features, reducing overfitting to any single dataset. Second, when one task lacks sufficient labeled data, multi-task training allows a low-resource task to benefit from another high-resource task. Moreover, running multiple tasks with one model costs less than running several models. As ASR and SD share common "subtasks", such as speech detection and "noise removal", and are often used together in speech processing, training ASR and SD together is likely to be beneficial.

For example, [31] applies joint-training of ASR and speaker classification tasks to fine-tune a wav2vector model. Furthermore, some researches([20][27]) achieve joint training of ASR and speaker change detection by simply adding speaker change tokens to the reference transcripts while training ASR. However, speaker change detection can hardly be post-processed into speaker diarization labels without extra models, even in a 2-speaker conversation scenario, because of the accumulative error of long sequences. Recent works have

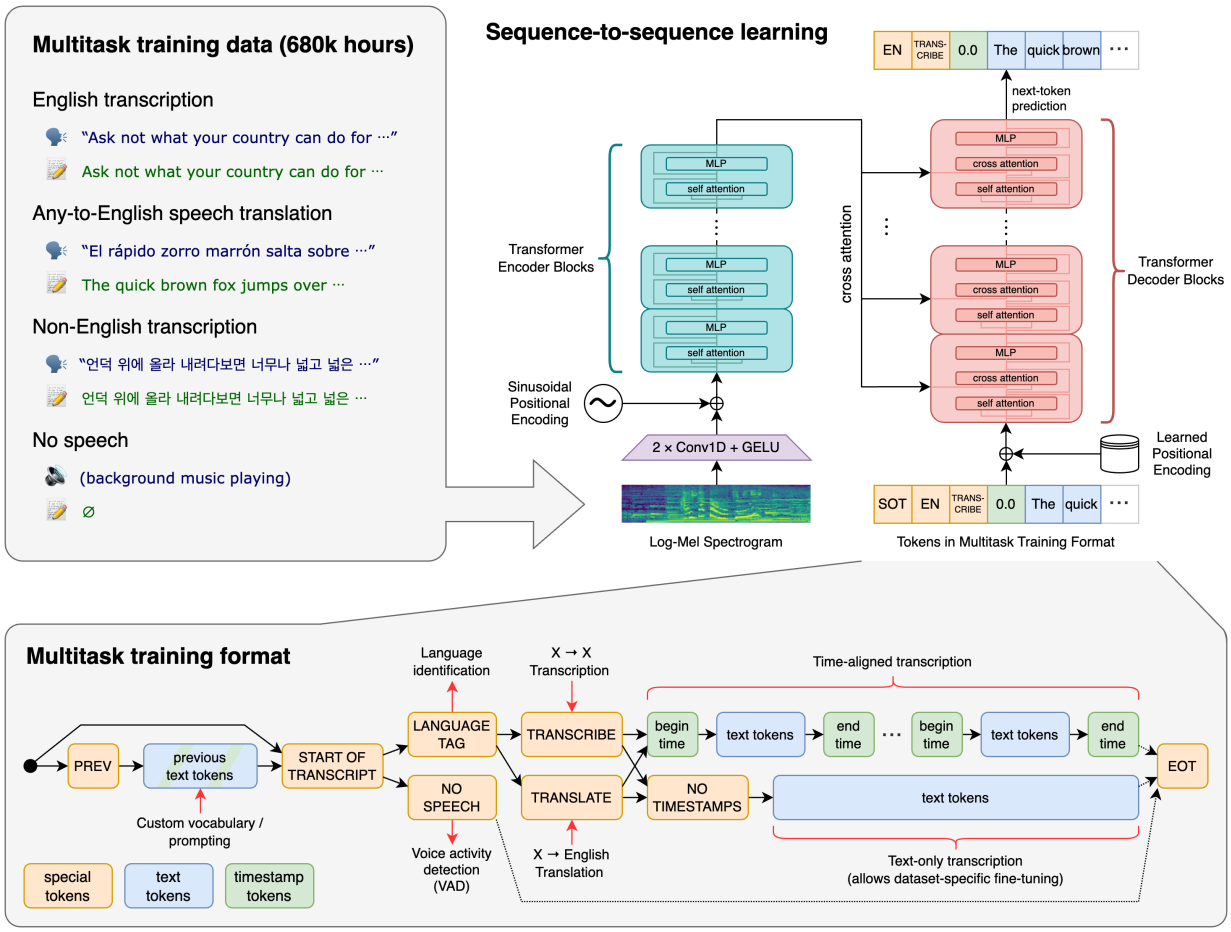

Figure 2.2: Whisper Model Pipeline[18]

realized both ASR and SD tasks in one model by adding extra modules. [11] combines an ASR model with an EEND by adding an auxiliary network, while [5] performs both ASR and SD on local sliding windows, and the results are aggregated to produce the final output.

2.1.4 Speaker Diarization Correction Models

Some SD errors, like the example illustrated in 1.1, are semantically apparent. Therefore, recent researches have also explored the ability of the language model to correct SD errors. DiarizationLM[24] and [7] prompt and fine-tune LLMs for SD error correction. However, those two models only handle word alignment errors, like the example in Figure 2.3. They fail to handle the case in 1.1, where a whole sentence is assigned to a wrong speaker. AG-

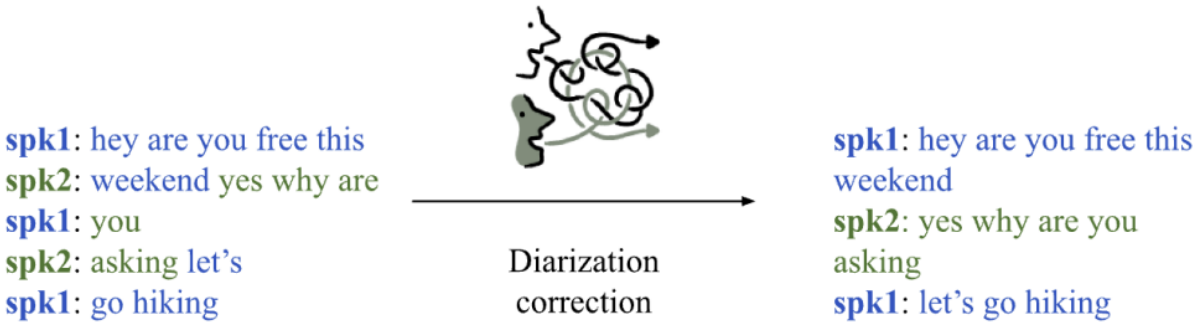

Figure 2.3: Speaker Diarization Error Correction[7]

LSEC[16] corrects SD errors with a multi-modality fusion of semantic information extracted by a pre-trained language model and speaker probability from an EEND model. This system is able to handle the error in 1.1; however, its ability is limited to 2-speaker conversations.

2.1.5 Multi-modal Speaker Diarization Models

Audio-based SD models only catch speaker features, lacking the ability to incorporate semantic information. Therefore, recent researches have also explored whether semantic information can help improve SD performance. MMSCD[25] concatenates audio and text embeddings together, then passes them to an encoder-decoder model to perform word-level speaker change detection. [4] utilizes speaker change detection and dialogue detection to manipulate the clustering process. Multi-modal speaker diarization using joint text and audio information remains an underexplored area, and this thesis aims to further investigate its potential.

2.2 Evaluation Metrics

2.2.1 Diarization Error Rate

Diarization Error Rate is often used for evaluate SD performance. This measurement is time based:

$$DER = \frac{\sum_{s \in S}(dur(s) \cdot (max(N_r(s), N_h(s)) - N_c(s)))}{\sum_{s \in S} dur(s) \cdot N_r(s)}$$

Where S is a set of timeline segments in the audio; each segment s contains homogeneous speaker labels in both reference (ground truth) and hypothesis annotation. $N_r(s)$ and $N_h(s)$ are number of active speakers within s in reference and hypothesis annotation respectively. $N_c(s)$ is the number of correctly identified speakers in s. This metric contains three types of errors: speaker confusion, where a segment is attributed to wrong speakers; false alarm, where the system predicts more speakers than ground truth; missed speech, where the system omits speakers within a segment.

2.2.2 Word-level Diarization Error Rate

For downstream applications, the focus is often on the speech content of each speaker rather than the timeline. Consequently, the Word-level Diarization Error Rate (WDER) has been introduced to measure errors in the lexical aspect. To calculate WDER, the system must generate transcripts with speaker labels (the output of an audio-based SD model must be aligned with an ASR transcript). The speaker labels from the hypothesis transcripts are then mapped to the ground truth transcripts with word level. WDER represents the proportion of words with incorrectly assigned speaker labels:

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

Where S is the number of ASR substitutions, C is the number of correctly transcribed words. S_{IS} and C_{IS} are the subsets of S, C with incorrectly assigned speaker labels.

2.2.3 Text-based Diarization Error Rate

Despite WDER's ability to evaluate SD performance at the text level, it fails to consider the deleted (missed speech) and inserted (false alarm) words in the hypothesis transcripts. Text-based Diarization Error Rate (TDER)[8] is introduced to provide a more comprehensive measurement of SD at the text level.

$$TDER = \frac{\sum_{u \in U} len(u) \cdot (max(N_r(u), N_h(u)) - N_c(u)))}{\sum_{u \in U} len(u) \cdot N_r(u)}$$

Where U is the set of utterances in the reference transcripts. Similarly to DER, $N_r(u)$ and $N_h(u)$ are the numbers of active speakers within u in reference and hypothesis annotations, respectively, and $N_c(u)$ is the number of correctly identified speakers in u. Note that transcripts do not support overlapping speakers, so $N_r(u)$ is always 1. Therefore, the formula can be written as:

$$TDER = \frac{\sum_{u \in U} len(u) \cdot (max(1, N_h(u)) - N_c(u)))}{\sum_{u \in U} len(u) \cdot N_r(u)}$$

However, the definition of 'utterances' requires further deliberation. In the official implementation of TDER, 'utterances' are defined by grouping words with the same speaker label in the reference transcript together. However, in the case where an utterance is extremely long, meaning one speaker has a long speech without interruption, and the hypothesis transcript has only one word mislabeled within the utterance, then all words within the utterance would be counted as wrong, which significantly improves TDER. Usually, this is not desirable, and this also causes the fact that while evaluating the same SD system output, the value of TDER is often much higher than DER. Referencing the segments in DER, TDER should also consider reference transcripts while generating utterances.

2.2.4 Diarization F1

Proposed by [8], Diarization F1 is another metric used to evaluate speaker diarization at the text level. Like the standard F1 formula, this metric requires two terms: precision and recall.

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Here, precision denotes the proportion of words with correct speaker labels in the hypothesis transcript, and recall denotes the proportion of words with correct speaker labels in the reference transcript.

$$precision = \frac{|speaker\_match(T_h, Tr)|}{|T_h|}$$
$$recall = \frac{|speaker\_match(T_h, Tr)|}{|T_r|}$$

Where T_h denotes all words in the hypothesis transcript, T_r denotes all words in the reference transcript, and $speaker\_match(T_h, T_r)$ denotes words that have the same speaker label in both the hypothesis and reference transcript.

2.3 Dataset

2.3.1 Trauma Interview

The Trauma Interview dataset comprises medical data from Emory's mental health department. The dataset includes audio recordings of interviews between psychologists and trauma patients. Annotations of the dataset are generated by Azure, with text transcripts, speaker labels, and segment timelines. However, although Azure outperforms open-sourced models, its annotations still fail to meet our expectations. In the following chapters, we will discuss how to improve the annotations.

In this dataset, the length of each audio file ranges from 20 minutes to 3 hours. Although

there are many more trauma interview recordings available, we conduct our experiments on only 117 audio recordings, with a total length of 138 hours. Hopefully, the remaining audio recordings can be processed using our model, instead of Azure.

This dataset consists of 411 interviews conducted with 336 participants, including longitudinal follow-ups (>1 month interval) for a subset. Demographic characteristics reveal: 93.4% female representation, 79.5% Black or African American (mean age = 31.4 years), with 38.7% having a high school education or less and 57.9% reporting monthly household income below $1,000.

2.3.2 Other dataset

Since the Trauma Interview dataset does not have golden-standard annotations, public datasets with expert annotation can better help us develop and evaluate models. We use the following datasets in our research:

AMI[1]: In this study, we utilize the Headset Mix version of the AMI dataset. The AMI dataset consists of 100 hours of meeting recordings, with each meeting containing 3-5 speakers. The AMI dataset has transcripts with a word timeline for each speaker, making it a feasible choice for both speaker diarization and speech recognition research. Since most recent works report evaluation performance on the AMI dataset, this dataset also allows us to compare our work with state-of-the-art models.

DailyTalk[15]: The DailyTalk dataset is proposed for conversational Text-to-Speech development, which contains 2,541 daily dialogues with a total duration of 22 hours. This is an easy dataset for the speaker diarization task, as it contains only two-speaker conversations, and the audio recording is "clean" – without noises, echoes, or overlapping speeches. Therefore, this dataset helps us determine whether a speaker diarization model handles basic cases well. Moreover, as most speaker diarization models do not include this dataset in their training set, this dataset can also evaluate the generalizability of models.

[1]https://groups.inf.ed.ac.uk/ami/corpus/

Chapter 3

Speaker Diarization Models and Error Analysis

Recent works on speaker diarization that achieve state-of-the-art (SOTA) performance are rarely open-sourced. To begin our research, we first try several open-sourced models, and identify the best one to start with. Then, we will compare its performance with Azure, and analyze the common mistakes of both Azure and the suboptimal open-sourced model.

3.1 Text-only Speaker Diarization Models

Intuitively, speaker diarization relies more on speaker voice features rather than speech content. Therefore, little research has focused on text-only speaker diarization, except [26]. In this work, the speaker diarization task is formulated as text generation, utilizing T5[19] model. The model inputs are transcripts with a special change token between two adjacent sentences, and the outputs are lists of labels marking whether a speaker change occurs in the corresponding change token position. This method uses the sliding window technique, and voting to aggregate the final speaker change detection (SCD) labels. Then, the SCD labels are transformed into a two-speaker diarization output, where each speech segment is explicitly assigned to one of the two speakers. However, this method is limited to handling 2-

speaker conversations, and the results suffer from accumulative errors on long audios. In my own experiment, when there are over 30 sentences within a transcript, the word diarization error rate (WDER) is close to 0.5, which is similar to random guessing.

3.2 Audio-based Speaker Diarization Models

3.2.1 DiaPer

DiaPer[14] is an EEND model; its structure is illustrated in Figure 3.1. For the attractor-decoder backbone, DiaPer uses a perceiver[12] instead of a transformer. Unlike transformers, where the query, key, and values are all from the input sequence while calculating self-attention, perceiver uses a trainable latent sequence to calculate query, and calculating cross-attention with key and value from the input sequence. The latent sequence is much smaller than the original input sequence, therefore, compared to a transformer, a perceiver is much more time and space efficient, with a space complexity reduced from $O(n^2)$ to $O(n)$.

The DiaPer model exchanges frame encodings and attractor information progressively on a layer-by-layer basis. By training on simulated 2-speaker conversation data, DiaPer achieves a DER close to SOTA models on the CallHome Dataset, even without fine-tuning. However, this model does not perform well on most datasets, especially those with multiple speakers, even with fine-tuning. We evaluate this model on the DailyTalk dataset, and obtain a DER of 0.4433, which is not desirable. Therefore, we proceed with an alternative model for our research.

3.2.2 Pyannote Powerset

Pyannote is an open-sourced speech processing python library that employs clustering-based method for speaker diarization. Pyannote Powerset is a model built on the pyannote pipeline. We start our research based on Pyannote Powerset for its outstanding performance: it outperforms the standard Pyannote 2.1 model on most datasets. We also evaluate this

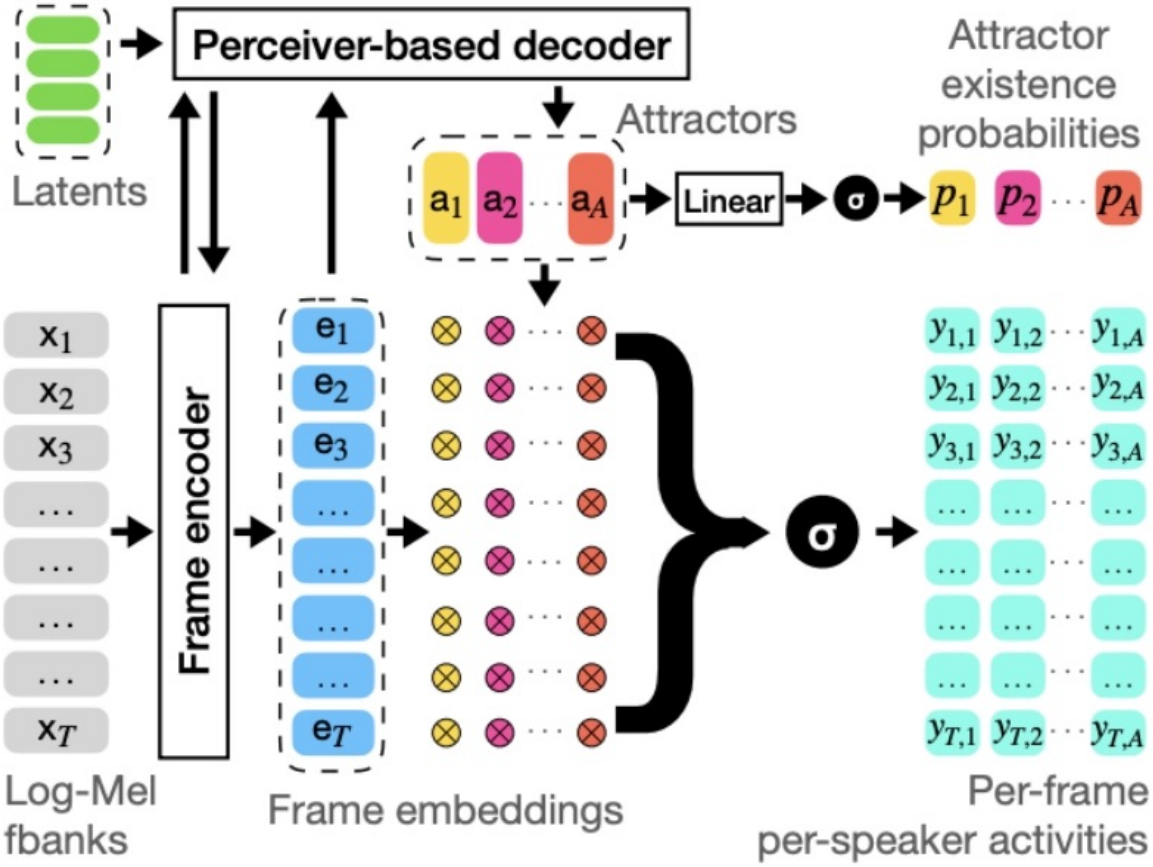

Figure 3.1: DiaPer diagram[14]

model on the DailyTalk dataset without fine-tuning, and achieve a DER of 0.157, which is acceptable. The whole pipeline contains the following four steps.

First, segmentation. In this step, the entire audio is converted into a sliding window sequence, and local speaker diarization is applied to each window using a simple EEND model. Visualization of the segmentation step's output is illustrated in Figure 3.2. The output is an $N \times f \times s$ boolean vector, where N is the total number of sliding windows (depending on the audio length), f is the number of frames per sliding window, and s is the maximum number of speakers within a sliding window. Each entry of the vector marks whether a speaker is active on the corresponding frame.

Second, embedding. For each sliding window, the corresponding speech segment is ex-

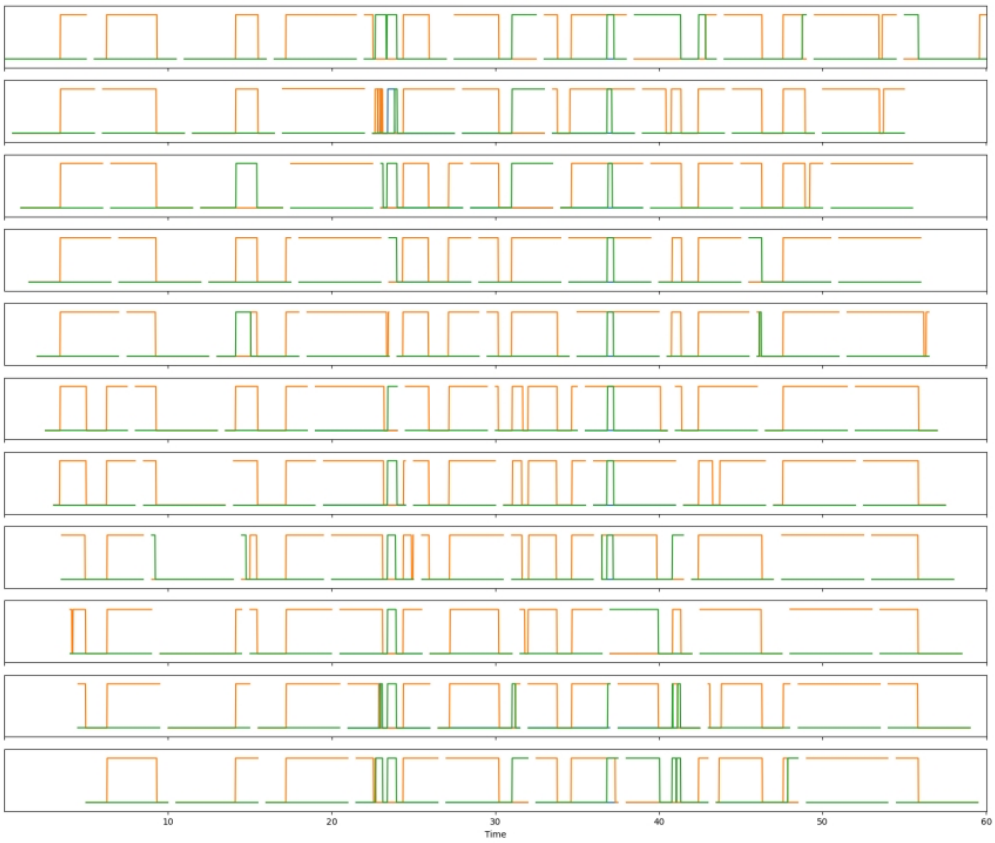

Figure 3.2: Output of pyannote segmentation. For each iteration of the sliding window and each local speaker label, the segmentation process determines the active status of each speaker on a frame-by-frame basis. In the illustration, distinct colors represent different local speakers.

tracted from the audio, given the start and end time. For each speaker within the window, frames where the speaker is inactive are masked. Finally, the speech segment is passed to a speaker embedding model. The output of this step is an $N \times s \times d$ vector, where d denotes the embedding dimension. The Pyannote Powerset utilizes the ECAPA-TDNN[6] model for speaker embedding.

Third, clustering. The Pyannote Powerset system uses agglomerative clustering to handle a flexible number of speakers. The clustering output is an $N \times s$ integer vector, which represents a mapping from local speaker labels to global speaker labels.

Finally, reconstruction. For overlapping parts of sliding windows, this step applies a voting strategy to aggregate the final result.

3.3 Post-processing

Audio-base speaker diarization result requires post-processing to obtain transcripts with speaker labels. In this research, we use WhisperX[1] to generate transcripts with a word-level timeline. WhisperX applies Whisper[18] for speech recognition, and employs forced alignment to synchronize transcript words with the audio.

We first try aligning the speaker diarization result and transcript by word-level timeline. However, this would cause word-level offsets due to the inherent imprecision in both the speaker diarization results and the transcript timelines. In the example below, the underlined words highlight instances of word-level alignment errors.

```
Speaker1:  Yeah.  What's a typical daylight for you?  Early
Speaker2:  riser before the sun shines, because I work at this job from
to It's a hour day.  I haven't been doing, well, I've been doing this
work a long time.  A typical day for me is work and home.  That's
Speaker1:  it. Gotcha.  What do you like to do in the evenings or on
the weekends?  Sit in front of the TV and binge watch.
```

Therefore, under the assumption that each sentence always contains a single speaker, we align the speaker diarization result with the transcript by sentence-level timelines. We first group the words in the transcript into sentences and determine the corresponding start and end times. Next, we assign a speaker label by identifying the speaker with the longest overlapping active speech duration within the sentence's time span. The result of this alignment is illustrated in the same example below.

```
Speaker1:  Yeah.  What's a typical daylight for you?
Speaker2:  Early riser before the sun shines, because I work at this
job from to It's a hour day.  I haven't been doing, well, I've been
doing this work a long time.  A typical day for me is work and home.
That's it.
Speaker1:  Gotcha.  What do you like to do in the evenings or on the
weekends?  Sit in front of the TV and binge watch.
```

Given the results of two types of alignment applied to a specific example, we assume that sentence-level alignment outperforms word-level alignment. To verify our assumption, we test two alignment methods on the AMI dataset, using Pyannote Powerset speaker diarization result and whisperX transcript. The results, as seen in Table 3.1, show that sentence-level alignment achieves a better performance than word-level alignment on all metrics listed below.

	WDER	TDER	Precision	Recall	F1
word-level	12.07	82.55	68.05	69.56	68.79
sentence-level	10.65	68.53	69.25	70.7	69.97

Table 3.1: Evaluation of two alignment methods on AMI dataset

3.4 Error Analysis

There are several types of errors in the speaker-labeled transcripts:

1. Short utterance: this happens when the whole sentence (utterance) less than 1 second is assigned to a wrong speaker label, such as in 1.1. This can result from alignment errors or speaker diarization errors themselves. Since speaker embeddings achieve optimal performance on speech segments of 3 to 5 seconds, shorter utterances can degrade embedding quality, ultimately affecting the clustering results.

2. Diarization Error: this occurs when a sentence is incorrectly attributed to a speaker. Such errors typically stem from clustering inaccuracies or local speaker diarization (SD) errors (the first step of pyannote pipeline). Note that we exclude short utterances from diarization error analysis because (1) errors occur more frequently on short utterances, and (2) they typically stem from different causal factors.

3. Segment Error: this refers to cases where a sentence in the transcript contains speech from more than two speakers.

4. Missed Speaker: this happens when no speaker is marked as active in the diarization output for a given utterance in the transcript.

5. Overlapping Speech: this happens when two or more speakers are active simultaneously, and the corresponding sentence incorrectly assigns a speaker during the alignment process.

To evaluate the overall performance and error distribution, we conducted a manual analysis of speaker-labeled transcripts from three audio files in the Trauma Interview dataset, each approximately 20 minutes in length. We counted the speaker label errors for each utterance. The total error rate was 9.36%, and the proportion of each error type is illustrated in Figure 3.3. As shown, short utterances account for the majority of errors.

We also analyze the errors of Azure transcript using the same method. The total error rate is 7.35%, and error distribution see figure3.4. Compared to Pyannote Powerset, Azure has a better overall performance, especially when handling short utterances. However, short utterances still make up most of the portion among all types of errors. Besides, Azure is more sensitive to overlapping speech than Pyannote Powerset.

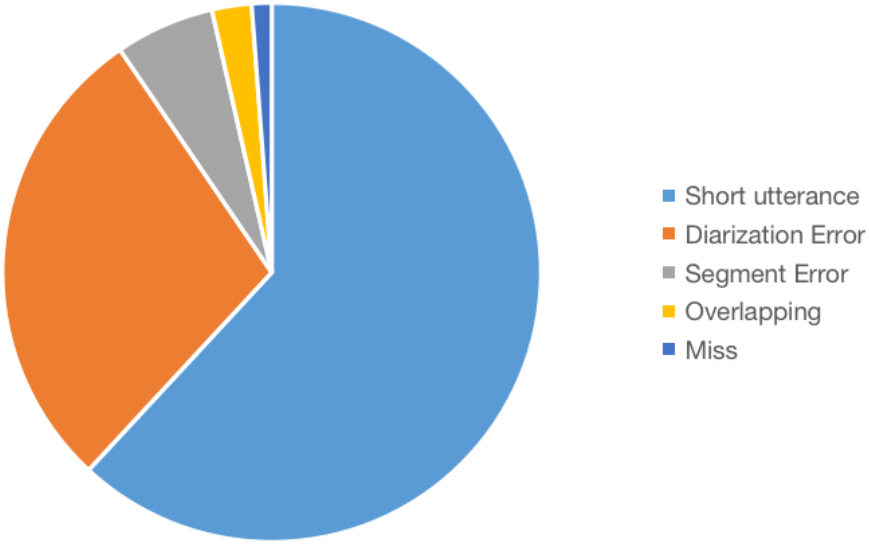

Figure 3.3: Pyannote Powerset Diarization Error Distribution

3.5 Discussion

Besides the models discussed in this chapter, other open-source models, such as [22] and [29], can also be explored. If these models achieve comparable performance, we may consider majority voting to improve the final speaker diarization performance.

Although sentence-level alignment outperforms word-level alignment in our experiment, Section 2.1.4 introduces several models to help correct alignment errors. We plan to apply these models to transcripts generated through word-level alignment, and assess whether they can achieve better performance than sentence-level alignment.

In Setcion 3.4, we conclude that incorrect speaker labels on short utterances are a common error among different speaker diarization models. However, in our literature survey, no methods or datasets focusing on this kind of error exists. Since most public datasets contain only a small proportion of short utterances, errors in this case have a minimal impact on overall evaluation results. To bridge this gap, new datasets with a significantly higher proportion of short utterances need to be developed. This would open a new research direction for future speaker diarization studies.

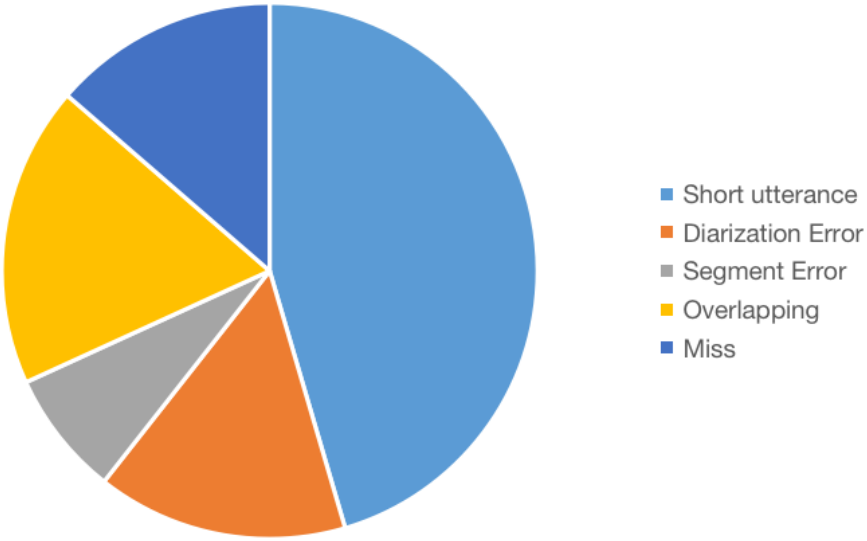

Figure 3.4: Azure Diarization Error Distribution

Finally, we address Research Question 1 with two key findings: 1. Current open-source speaker diarization models generally perform worse than Azure; and 2. Among all open-source models evaluated, Pyannote Powerset demonstrates the best performance. However, both Azure and Pyannote Powerset fails to achieves a high accuracy in processing short utterances.

Chapter 4

Speaker Diarization Error Corretcion and Multi-modal Model Development

4.1 Error Correction with ChatGPT

At this stage, we have two speaker-labeled transcripts from both Azure and Pyannote Power-set outputs of the same audio file. To improve the diarization result, we reserve the utterances that both models agree on (which are more likely to be correct), and consult ChatGPT for the utterances on which the two models disagree. In this experiment, we use GPT-4o for this purpose.

4.1.1 Motivation

Why not using existing speaker diarization error correction models introduced in Section 2.1? Most SD error correction models are focused on alignment error, which we already fixed by sentence timeline alignment. AG-LSEC[16] can correct speaker label error of a whole sentence, but it is not open-sourced. Therefore, we explore our own speaker diarization error correction method.

Why not generating speaker-labeled transcripts using more than three models and ap-

plying a voting strategy to aggregate the final result? Such an ensemble strategy is only effective when all models exhibit a similar performance. However, pyannote powerset is the best open-sourced model we have found so far, but still underperforms Azure; other models perform even worse. In this case, applying ensemble learning may even downgrade the performance.

Why not invoking ChatGPT for every utterance? ChatGPT makes mistakes as well; without speaker voice features, ChatGPT performs even worse than audio-based speaker diarization models. Additionally, calling ChatGPT on all utterances is expensive. Therefore, we apply ChatGPT only on utterances which two models disagree, as the speaker labels on these utterances are more likely to be wrong in Azure transcripts.

Why not using another audio-based model for "consulting," given that GPT also does not perform well on SD tasks? As explained in Section 3.4, mislabeling short utterances is a common issue among audio-based SD models. However, errors like 1.1 are easy to detect and fix semantically given the context window. Therefore, language models like ChatGPT have an advantage over audio-based models in this case.

4.1.2 Prompt

We fix speaker diarization errors by calling the ChatGPT API. In a chat completion, we provide the user input as a context window of approximately six sentences. The context window contains the utterance whose speaker label need to be examined and fixed, along with 2-3 sentences preceding and succeeding the utterance. We also provide additional user and assistant messages to guide ChatGPT for few-shot learning. We have tried several prompt strategies:

1. Selection from two predictions. Unfortunately, its performance is not desirable, especially when Pyannote Powerset's performance is much worse than Azure's.

2. Masked Speaker Label. We mask the disputed speaker label and ask GPT to fill in the correct label. This method also performs poorly, often producing results close to random

guessing.

3. Context Window with Azure Predictions. We provide a context window with all speaker labels from Azure's prediction, and instruct ChatGPT to fix when necessary. Its performance is the best among all prompt strategies we have tried, so we finally choose this method for error fixing of the Trauma Interview dataset. The final chat completion is provided below.

```
<System>: The following is the speaker-diarization result of a conversation
between two speakers. There may be some utterances assigned to a wrong
speaker. Correct the error if necessary. Make sure that the conversation
makes sense and is coherent after correction. Please use speaker labels
of format 'Speaker1', 'Speaker2'. Please format the output as a list
of speaker labels.
<User>: Speaker1: Have there been times when you felt especially irritable
or angry and it showed in your behavior? Speaker1: No. Speaker1:
OK. Speaker1: Or in the past month, have there been times when you
were taking more risks or doing things that might have caused you harm?
<Assistant>: Speaker1, Speaker2, Speaker1, Speaker1
<User>: [input context window]
```

In the chat completion framework, the initial 'User' message and subsequent 'Assistant' message form an input-output demonstration pair. This structure guides ChatGPT to generate responses following the specified pattern. The 'input context window' refers to the contextual segment containing the target speaker label for correction, formatted identically to the first 'User' message example.

4.1.3 Post-processing

Given the modified label of a whole context window, we only choose the label which needs to be fixed. However, the output speaker labels of ChatGPT are not always correspondent with the original input transcript. For example, if the input is the following context window:

```
Speaker2:  Have you ever had a period of time in your life when you
felt like super hyper high, full of energy?
Speaker2:  No.  No.
Speaker2:  OK.
Speaker2:  Have there ever been times in your life when you've felt
like that persistent irritability over several days?
Speaker1:  No.
```

The output might be this:

```
Speaker1, Speaker2, Speaker1, Speaker1, Speaker2
```

We can easily tell that in this example, ChatGPT relabels the entire conversation window, and swaps Speaker1 and Speaker2. To handle this case, we require a speaker mapping from chat completion output to original input. We manage this by simply counting the number of speaker labels within the window which the chat completion output is different from original transcript. If more than half of the labels within the window differ, we choose a speaker label different from the modified speaker label within the context window.

4.1.4 Evaluation Result

We evaluate the performance of correction on three complete audio files from the interview dataset. These transcripts contain a total of 3,992 utterances, with 499 of them having different speaker labels from Pyannote Powerset and Azure. We analyze these 499 utterances

by counting the number of speaker label errors, from the results of Pyannote Powerset and Azure, and after fixing with ChatGPT. The results are presented in Table 4.1. From the results, we can conclude that ChatGPT can fix over half of the errors in the disagreement utterances.

Pyannote Powerset	Azure	GPT
349	213	90

Table 4.1: Speaker label error count of Pyannote Powerset, Azure output and after fixing by GPT, out of 3,992 utterances.

Finally, we run error fixing on the entire Trauma Interview dataset. In the following experiments, we view the labels after fixing as ground truth labels of the Trauma Interview dataset. The error rate of the corrected labels can be estimated as $(3992 \times 7.35\% - (213 - 90))/3992 = 4.26\%$.

4.2 Multi-modal Model Development

Having processed the Trauma Interview dataset, we now explore models that may achieve better performance on this dataset.

4.2.1 Multi-modal Segmentation

Pyannote Powerset performs local speaker diarization on sliding windows, based on audio waves. We first explore whether we can replace the audio-only model with a multi-modal model for local speaker diarization.

The most straightforward approach is to concatenate audio and text tokens. To enable this, we first need to synchronize audio with text tokens. Here, we use the same alignment method as [25]: aligning text tokens with the nearest audio chunks. Audio chunks are converted to speaker embeddings using Redimnet[28]. Text embeddings are generated using BGE[3]. The audio and text embeddings are then concatenated and passed to a transformer

encoder followed by a classification head to predict speaker labels. However, when we train the model, the loss does not decrease. Therefore, we proceed with other methods.

4.2.2 Speech Segment Clustering

Although multi-modal local speaker diarization is ineffective, multi-modal speaker change detection (SCD) remains a feasible approach. Inspired by [4], we plan to utilize speaker change detection to improve speaker diarization results by manipulating the clustering procedure. For example, refining the affinity matrix in spectral clustering.

Before exploring SCD's ability to improve clustering results, we first explore whether clustering on speech segments, which are audio segments cut by the start and end times of sentences, has a reasonable performance. We first try this strategy on the Trauma Interview dataset. Here, we use the spectral clustering method introduced in [23]. The predicted speaker label accuracy is 90.05%, which is acceptable for future exploration.

Then, we explore whether speech segmentation improves speaker diarization results, compared to audio segmentation. Specifically, we use the same embedding and clustering pipeline as Pyannote Powerset, and compare the performance with Pyannote Powerset on the AMI dataset. The evaluation results are shown in Table 4.2. It turns out that speech segment clustering has no advantage, even when evaluated with WDER. Since offsets exist in the transcript sentence timeline, the DER gap is even larger. Since speech segment clustering does not generalize well on datasets with a flexible number of speakers, we will explore more generalizable models in the next chapter.

	DER	WDER
Pyannote Powerset Baseline	20.1	10.52
Speech Segment Clustering	35.4	12.05

Table 4.2: Evaluation result of speech segment clustering method, and comparison with Pyannote Powerset baseline. When calculating WDER, baseline model uses sentence-level speech-transcript alignment.

4.3 Discussion

4.3.1 Future Exploration of LLM Speaker Label Fixing

Providing more examples in chat completion may help solve the label mismatch problem in 4.1.3. Besides few-shot learning of ChatGPT, we may even try fine-tuning.

Since our experiments involve using LLM speaker label fixing in a specific scenario: an interview between a psychologist and a trauma patient, we can provide more information in the system message. We may first try to predict the identity (psychologist or patient) of a speaker label with ChatGPT, or other language models, by providing a few conversation windows within a transcript. Then, we provide the identity information of speaker labels in each API call. This may help ChatGPT decide the speaker of each sentence more accurately.

Moreover, in 3.4, we conclude that both Pyannote Powerset and Azure fail to handle short utterances well. However, we only pass the segments where the two models disagree to ChatGPT. This method fails to fix many short utterances with incorrect speaker labels. Otherwise, we may try passing the disagreement segments as well as short utterances to chat completion, and manually evaluate the performance of ChatGPT to see if this can improve overall speaker label accuracy.

Recently, more LLMs are emerging. We have tried speaker error fixing with both Claude-3.5 and Deepseek-v3 on the user interface. Although Claude-3.5 fails in this task, Deepseek-v3 can correctly fix speaker labels in several examples. Since Deepseek-v3 is much cheaper, and may perform as well as, or even better than GPT-4o, it is worth trying with prompts and API calls.

4.3.2 Speaker Change Detection to Improve Speech Segment Clustering

By predicting whether a speaker change exists between two sentences (speech segments), speaker change detection (SCD) models can potentially help improve speech segment clus-

tering. For example, in spectral clustering with an affinity matrix of the distance between segment embeddings, if SCD indicates that a speaker change exists between two adjacent segments, we can increase the corresponding entries in the affinity matrix; if the detection model indicates that no speaker change exists, we can decrease the corresponding entries. Although we do not currently have a suitable SCD system, this method is worth exploring.

4.3.3 Conclusion

Therefore, we can answer research question 2: with the annotation of another audio-based method, we can identify the speaker labels in the Azure annotation that are more likely to be incorrect. Then, by calling ChatGPT APIs, we can correct over half of these labels, which can significantly improve the overall transcript quality. However, we still struggle to find multi-modal models that perform better in both the Trauma Interview dataset and other publicly available datasets, and speech segment clustering with SCD refinement can potentially be effective.

Chapter 5

Fine-tuning Automatic Speech Recognition Model for Speaker Diarization

Automatic Speech Recognition (ASR) transcription timeline contains errors, which can lead to text diarization errors when aligning outputs of an audio-based speaker diarization model with ASR transcripts. Besides, since ASR models take in audio wave and generate text transcripts, they are capable of handling both audio features and context information. Therefore, we explore whether ASR models can catch temporal information of Speaker Diarization (SD) timeline, utilize context information to fix Speaker Diarization errors, and finally enhance text diarization result. In our system, a modified ASR model takes both audio features and speaker diarization timeline as the input, and generates transcriptions with speaker labels. Here, we use Whisper[18] as the backbone ASR model, and the output of the Pyannote Powerset[17] model as input SD timeline.

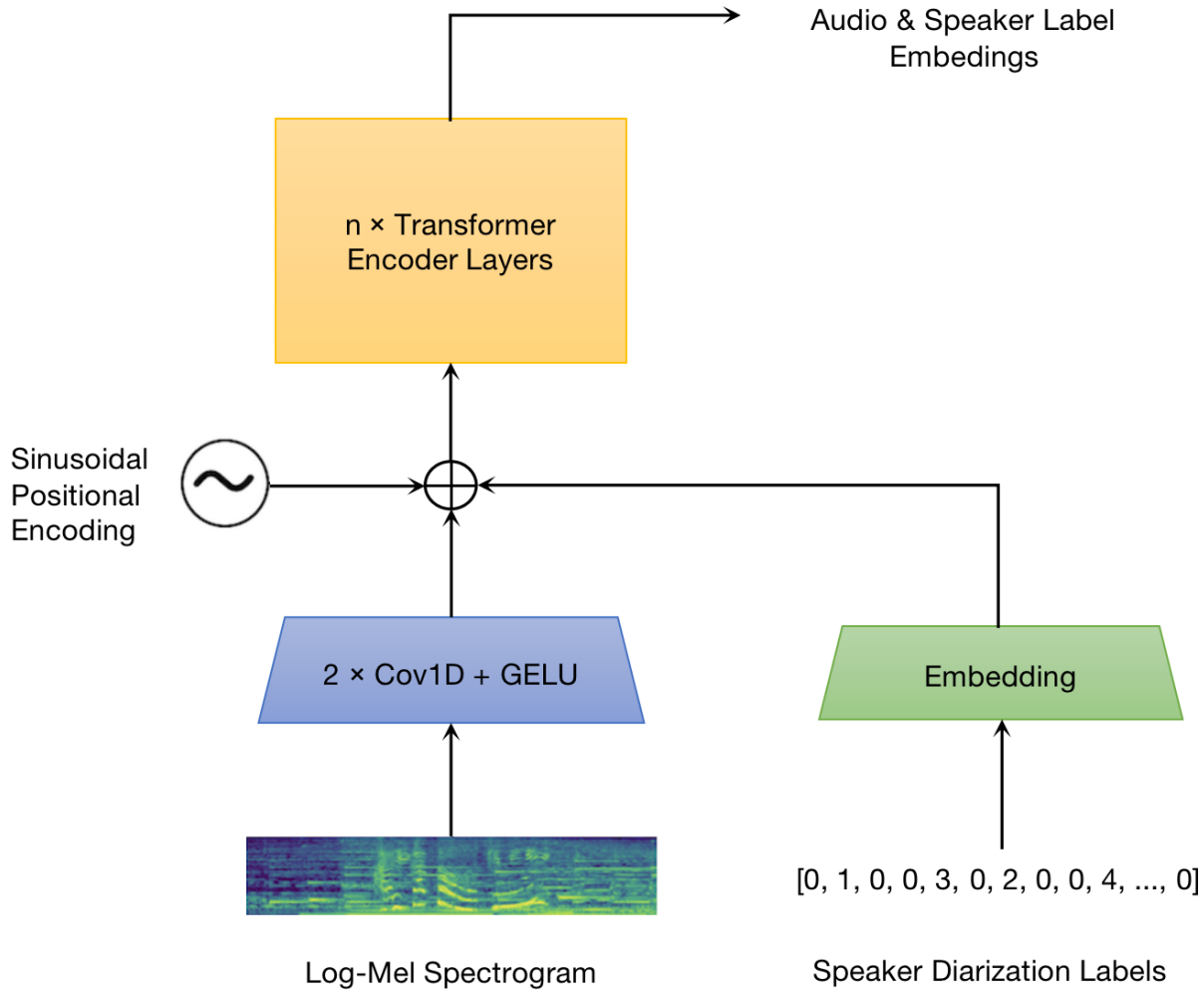

Figure 5.1: Proposed Audio & Speaker Diarization Labels Encoder

5.1 Model

Whisper[18] model's structure has been introduced in Section 2.1. Whisper is an encoder-decoder model, while the encoder extracts audio features, and the decoder generates transcripts. The audio encoder takes an audio log-mel spectrogram, which is passed to two 1-D convolution layers, followed by a transformer encoder. The modified encoder layer adds speaker diarization embeddings between the last convolution layer and the transformer encoder. The structure of the modified encoder is illustrated in Figure 5.1.

5.1.1 Speaker Diarization Embedding

The speaker diarization annnotations consist of lists of segments, each containing start time, end time and active speaker. Note that time overlaps may exist between two different segments.

To adapt speaker diarization annnotations for a neural network, we represent them as vector embeddings. Before that, we need to convert annotations into labels (integer vectors). The labels are created with a resolution of 0.02 second, to match audio encoding dimension after convolutions. On the labels, we mark the start and end times of each segment. Specifically, if speaker i is active from t_1 second to t_2 second, then $label[t_1 \times 50] = 2i + 1$, and $label[t_2 \times 50] = 2i + 2$. All other positions of the labels are filled with 0. In the encoder model, we add a learnable embedding layer to generate label embeddings.

5.1.2 Output Format

Whisper model's original output transcripts contain only speech contents. Our modified output adds speaker labels <spk$i$ > preceding each sentence, and a newline sign '\n' at the end of each sentence. Below is an example:

```
<spk0>:  Hi, guys.
<spk0>:  Good morning, everybody here.
<spk0>:  And I want to introduce myself.
```

5.1.3 Sliding Window

To further improve our model's performance, we also apply the sliding window aggregation technique for speaker labels during the inference stage. The input audio wave of our model contains several adjacent speech segments. In our default method, the input audio waves contain no overlapping segments. However, in the sliding window method, the next window

simply moves one segment forward, resulting in overlapping parts between adjacent windows. Ideally, except for the speaker label, the model generates the same text for the same overlapping segments, regardless of their position within the window. Then, we can align different windows by finding the identical text segments, and vote to decide the speaker labels. However, slight differences often exist among different windows, so we require an alignment technique. We use the word-level Needleman–Wunsch algorithm to align adjacent windows, with the Levenshtein distance for scoring. Details can be seen in Algorithm 1.

In the algorithm, seq_1 represents all words from the transcript in the front sliding window, and seq_2 corresponds to those in the back sliding window. Unlike the first row, which is initialized with increasing values, the first column of the dynamic programming (DP) matrix remains zero. This ensures that a gap at the beginning of seq_1 and seq_2 does not incur any alignment cost, as such a gap is naturally expected between two adjacent sliding windows. After aligning the sliding windows, for each sentence and corresponding speaker labels, we use the voting strategy to aggregate the final predicted speaker label.

5.2 Results and Discussion

5.2.1 Dataset

We use the AMI Headset Mix dataset for training. The dataset is split into training, validation, and testing sets using an 8:1:1 ratio.

5.2.2 Baseline Model

Our baseline model utilizes the diarization results from the Pyannote Powerset model, and aligns them with the WhisperX transcript using sentence-level timelines.

5.2.3 Experiment Setup

In our experiments, we utilize the speaker diarization annotations generated by Pyannote Powerset as the speaker diarization labels. During the training process, each input window contains several adjacent sentences from the ground truth transcripts, as well as the corresponding audio chunks and SD model predictions extracted with the start and end times of the window. For inference, we utilize Silero[21], a voice activity detection (VAD) model to obtain speech segments that are short enough for our model, while avoiding cutting within a sentence. The model was fine-tuned using initialized weights from the pre-trained Whisper base model.

5.2.4 Evaluation Results

We use WDER and TDER metrics to evaluate the text-level diarization performance of our model. We also evaluate our model on the Trauma Interview dataset without further adaptation. The evaluation results are presented in Table 5.1.

5.2.5 Discussion

The results are unsatisfactory, even worse than the baseline model, and the sliding window aggregation technique provides little improve. We attribute this result to the fact that the model fails to capture the time alignments between speaker labels and audio, or overfits on the error correction of the speaker labels, as the provided speaker labels are not the actual ground truth. To address this, we plan to provide the actual ground truth speaker labels during the training stage, and see if this can help bridge the gap.

There are also problems with the output format. The decoder model only has information about the previous tokens while generating the next token. However, in our current output format, the speaker labels are placed before the speech content. Therefore, the model lacks text information of the current utterance while predicting the speaker label. Intuitively, when

deciding the speaker of a sentence, the content of the sentence matters more than the previous context. This means that in our model, text information helps little to improve speaker diarization results, or even disturbs the results. Therefore, we plan to try placing speaker labels behind the speech content for each sentence. Besides, we may also try adding extra transformer layers or bidirectional LSTM to correct speaker labels, using context information from both the front and the back.

Regarding Research Question 3, our experiments fail to enhance speaker diarization performance by fine-tuning the Whisper model. However, with further modifications, this approach could potentially be effective.

Algorithm 1: Two sequences alignment algorithm

Input: Two sequences $seq1$ and $seq2$
Output: Indices of aligned sequence pairs

$m \leftarrow \text{length}(seq1)$;
$n \leftarrow \text{length}(seq2)$;
Initialize dp as a $(m+1) \times (n+1)$ matrix with zeros;
for $j \leftarrow 1$ **to** n **do**
    $dp[0][j] \leftarrow j$;
end
for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
        $cost \leftarrow 1 - \text{S}(seq1[i-1], seq2[j-1])$;
        $dp[i][j] \leftarrow \min(dp[i-1][j]+1, dp[i][j-1]+1, dp[i-1][j-1]+2 \times cost)$;
    end
end
Initialize $aligned$ as an empty list;
$i \leftarrow m$, $j \leftarrow n$;
while $i > 0$ **or** $j > 0$ **do**
    if $j == 0$ **or** $dp[i-1][j] < \min(dp[i][j-1], dp[i-1][j-1])$ **then**
        Append $(i-1, \text{None})$ to $aligned$;
        $i \leftarrow i-1$;
    end
    else if $i == 0$ **or** $dp[i][j-1] < \min(dp[i-1][j], dp[i-1][j-1])$ **then**
        Append $(\text{None}, j-1)$ to $aligned$;
        $j \leftarrow j-1$;
    end
    else
        Append $(i-1, j-1)$ to $aligned$;
        $i \leftarrow i-1$, $j \leftarrow j-1$;
    end
end
Reverse $aligned$;
return $aligned$;

	AMI WDER/TDER	**Trauma Interview** WDER/TDER
baseline	10.52/65.81	3.04/25.78
finetuned whisper		
w/o sliding window	18.02/74.94	11.01/61.52
with sliding window	14.4/89.68	9.37/77.25

Table 5.1: Evaluation Results of Modified Whisper Model on AMI dataset.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we have analyzed different types of errors produced by audio-based speaker diarization models, and explored how language models can help improve speaker diarization results. By leveraging LLMs like GPT-4o, we demonstrated their capability of fixing nearly half of speaker label errors. Besides, we also propose two novel multi-modal SD models that utilize both text and audio: speech segment clustering and ASR model fine-tuning, which have the potential to outperform current state-of-the-art models.

6.2 Future Directions

6.2.1 Speaker Diarization on Short Utterances

We conclude in Section 3.4 that incorrect speaker labels on short utterances are a common issue across various speaker diarization models, while this problem has received relatively little attention in prior research. We plan to focus our future research on addressing this specific challenge.

First, we need to create a dataset with a significant portion of short utterances. We will

select audio chunks along with their corresponding speaker diarization annotations and transcripts from publicly available datasets. Unlike the training process for EEND models, we will not generate data by merging different speech segments, as such synthetic conversations lack semantic meaning and are unsuitable for research involving language models.

Next, we will evaluate various open-source state-of-the-art models on our dataset, including both SD-only models and joint ASR and SD models. For audio-based SD models, we will post-process the results into transcripts with speaker labels using sentence-level alignment, as described in Chapter 3. We will evaluate these transcripts with text-based speaker diarization metrics such as WDER, TDER, and Diarization F1 score. We will also examine if the performance of these models can be improved by fine-tuning. These experiments aim to provide an overview of different models' ability to handle this specific case.

Finally, we will develop our new model on this dataset. The model can be a speech segment clustering method, as described in Chapter 4, or a fine-tuned Whisper model, as described in Chapter 5, or other methods. We expect our model to outperform existing open-source models.

6.2.2 Whisper Finetuning for General Speaker Diarization Error Correction

The Whisper fine-tuning method has not achieved a desirable performance so far. Despite the refinement methods proposed in Chapther 5, another significant issue remains unresolved. Since the model is trained to generate transcripts with speaker labels derived from Pyannote Powerset speaker diarization results, there is no guarantee that it will function effectively with alternative speaker diarization models.

To handle this limitation, it may be necessary to fine-tune the model on speaker diarization results from different models. Alternatively, we could generate hypothetical speaker diarization data by adding noise to ground truth speaker diarization annotations, which would then serve as input to the modified Whisper model. The noise contains: temporal jit-

tering, which means adding random offsets to utterance start and end; deletions or insertions at the utterance level; and speaker label substitutions.

The evaluation experiments for this modified Whisper model will be conducted using speaker diarization results from various SD models. For each speaker diarization model, the baseline approach consists of mapping to Whisper transcripts with sentence timelines. We will compare the output of the modified Whisper model with the corresponding baseline for each model. The desirable outcome would be that the modified Whisper model outperforms the baseline across most, if not all, of the tested SD models.

6.2.3 Experiment of Whisper Finetuning on 2-Speaker Conversation

When a human expert annotates speaker labels of a transcript without audio, usually a 2-speaker conversation is relatively manageable, while a multi-speaker meeting is more confusing. We assume that language models face similar challenges. Therefore, we plan to fine-tune and evaluate the Whisper model on two-speaker conversation datasets, such as DailyTalk and Callhome[1], to determine whether this approach can achieve superior performance in this more constrained scenario.

[1]https://ca.talkbank.org/access/CallHome/eng.html

Bibliography

[1] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. Whisperx: Time-accurate speech transcription of long-form audio. In *Interspeech 2023*, pages 4489–4493, 2023. doi: 10.21437/Interspeech.2023-78.

[2] Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *Interspeech 2023*, pages 1983–1987, 2023. doi: 10.21437/Interspeech.2023-105.

[3] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL `https://aclanthology.org/2024.findings-acl.137/`.

[4] Luyao Cheng, Siqi Zheng, Zhang Qinglin, Hui Wang, Yafeng Chen, and Qian Chen. Exploring speaker-related information in spoken language understanding for better speaker diarization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14068–14077, Toronto, Canada, July 2023. Association for Computational Linguis-

tics. doi: 10.18653/v1/2023.findings-acl.884. URL https://aclanthology.org/2023.findings-acl.884/.

[5] Samuele Cornell, Jee-Weon Jung, Shinji Watanabe, and Stefano Squartini. One model to rule them all ? towards end-to-end joint speaker diarization and speech recognition. pages 11856–11860, 04 2024. doi: 10.1109/ICASSP48485.2024.10447957.

[6] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. In *Interspeech 2020*, pages 3830–3834, 2020. doi: 10.21437/Interspeech.2020-2650.

[7] Georgios Efstathiadis, Vijay Yadav, and Anzar Abbas. Llm-based speaker diarization correction: A generalizable approach, 2024. URL https://arxiv.org/abs/2406.04927.

[8] Chen Gong, Peilin Wu, and Jinho D. Choi. Aligning Speakers: Evaluating and Visualizing Text-based Speaker Diarization Using Efficient Multiple Sequence Alignment . In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 778–783, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi: 10.1109/ICTAI59109.2023.00119. URL https://doi.ieeecomputersociety.org/10.1109/ICTAI59109.2023.00119.

[9] Antoine Laurent Hervé Bredin. End-to-end speaker segmentation for overlap-aware resegmentation. In *Interspeech*, 2021.

[10] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Paola García. Encoder-decoder based attractors for end-to-end neural diarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1493–1507, 2022. doi: 10.1109/TASLP.2022.3162080.

[11] Yiling Huang, Weiran Wang, Guanlong Zhao, Hank Liao, Wei Xia, and Quan Wang. On the success and limitations of auxiliary network based word-level end-to-end neural

speaker diarization. In *Interspeech 2024*, pages 32–36, 2024. doi: 10.21437/Interspeech.2024-561.

[12] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. *ArXiv*, abs/2103.03206, 2021. URL `https://api.semanticscholar.org/CorpusID:232110866`.

[13] Federico Landini, Mireia Diez, Alicia Lozano-Diez, and Lukáš Burget. Multi-speaker and wide-band simulated conversations as training data for end-to-end neural diarization. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10097049.

[14] Federico Landini, Mireia Diez, Themos Stafylakis, and Lukáš Burget. Diaper: End-to-end neural diarization with perceiver-based attractors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3450–3465, 2024. doi: 10.1109/TASLP.2024.3422818.

[15] Keon Lee, Kyumin Park, and Daeyoung Kim. Dailytalk: Spoken dialogue dataset for conversational text-to-speech. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095751.

[16] Rohit Paturi, Xiang Li, and Sundararajan Srinivasan. Ag-lsec: Audio grounded lexical speaker error correction. In *Interspeech 2024*, pages 1650–1654, 2024. doi: 10.21437/Interspeech.2024-845.

[17] Alexis Plaquet and Hervé Bredin. Powerset multi-class cross entropy loss for neural speaker diarization. In *Interspeech 2023*, pages 3222–3226, 2023. doi: 10.21437/Interspeech.2023-205.

[18] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[19] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019. URL `https://api.semanticscholar.org/CorpusID:204838007`.

[20] Laurent El Shafey, Hagen Soltau, and Izhak Shafran. Joint speech recognition and speaker diarization via sequence transduction. In *Interspeech 2019*, pages 396–400, 2019. doi: 10.21437/Interspeech.2019-1943.

[21] Silero Team. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier. `https://github.com/snakers4/silero-vad`, 2024.

[22] Jiaming Wang, Zhihao Du, and Shiliang Zhang. Told: a novel two-stage overlap-aware framework for speaker diarization. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096436.

[23] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno. Speaker diarization with lstm. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5239–5243, 2018. doi: 10.1109/ICASSP.2018.8462628.

[24] Quan Wang, Yiling Huang, Guanlong Zhao, Evan Clark, Wei Xia, and Hank Liao. Diarizationlm: Speaker diarization post-processing with large language models. In *Interspeech 2024*, pages 3754–3758, 2024. doi: 10.21437/Interspeech.2024-209.

[25] Jee weon Jung, Soonshin Seo, Hee-Soo Heo, Geonmin Kim, You Jin Kim, Young ki Kwon, Minjae Lee, and Bong-Jin Lee. Encoder-decoder multimodal speaker change detection. In *Interspeech 2023*, pages 5311–5315, 2023. doi: 10.21437/Interspeech.2023-2289.

[26] Peilin Wu. Beyond audio: Advancing speaker diarization with text-based methodologies and comprehensive evaluation, 2024.

[27] Wei Xia, Han Lu, Quan Wang, Anshuman Tripathi, Yiling Huang, Ignacio Lopez Moreno, and Hasim Sak. Turn-to-diarize: Online speaker diarization constrained by transformer transducer speaker turn detection. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8077–8081, 2022. doi: 10.1109/ICASSP43922.2022.9746531.

[28] Ivan Yakovlev, Rostislav Makarov, Andrei Balykin, Pavel Malov, Anton Okhotnikov, and Nikita Torgashov. Reshape dimensions network for speaker recognition. In *Interspeech 2024*, pages 3235–3239, 2024. doi: 10.21437/Interspeech.2024-2116.

[29] Gaobin Yang, Maokui He, Shutong Niu, Ruoyu Wang, Yanyan Yue, Shuangqing Qian, Shilong Wu, Jun Du, and Chin-Hui Lee. Neural speaker diarization using memory-aware multi-speaker embedding with sequence-to-sequence architecture. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11626–11630, 2024. doi: 10.1109/ICASSP48485.2024.10446661.

[30] Shota Horiguchi Kenji Nagamatsu Shinji Watanabe Yusuke Fujita, Naoyuki Kanda. End-to-end neural speaker diarization with permutation-free objectives. In *Interspeech*, 2019.

[31] Xianrui Zheng, Chao Zhang, and Phil Woodland. Tandem multitask training of speaker diarisation and speech recognition for meeting transcription. In *Interspeech 2022*, pages 3844–3848, 2022. doi: 10.21437/Interspeech.2022-11368.