Who Spoke When?
State-of-the-Art in Speaker Diarization with Pyannote

Kevin KURTZ

December 16, 2024

Contents

1 Introduction

Speaker diarization, the process of identifying *'Who spoke when?'* in an audio recording, is a remarkable combination of innovative ideas and mathematical techniques, bringing together advancements in artificial intelligence, signal processing and speech recognition. By partitioning an audio recording into segments attributed to individual speakers, it plays a crucial role in both real-time and offline transcription, such as in emergency call processing, multi-speaker meeting analysis or even automatic subtitling.

The concept of speaker diarization appears to have first been mentioned in the late 1990s, primarily driven by the need to process and annotate broadcast news. Early applications focused on tasks like automatic transcription and metadata labeling, marking the first steps toward its broader adoption [1].

However, challenges persist. Overlapping speech is a major hurdle, especially when multiple speakers talk simultaneously. Noisy environments and real-time constraints complicate matters, as the lack of sufficient data limits the ability to perform effective clustering during real-time diarization [2].

Over the years, approaches to diarization have evolved from statistical methods (GMM, HMM) to deep learning-based methods. Techniques like x-vectors and ECAPA-TDNN have enhanced accuracy and End-to-End Neural Diarization (EEND) directly addresses overlapping speech and diarization errors [3, 4].

The process of speaker diarization can be decomposed into several key stages, as illustrated in Figure 1 below.

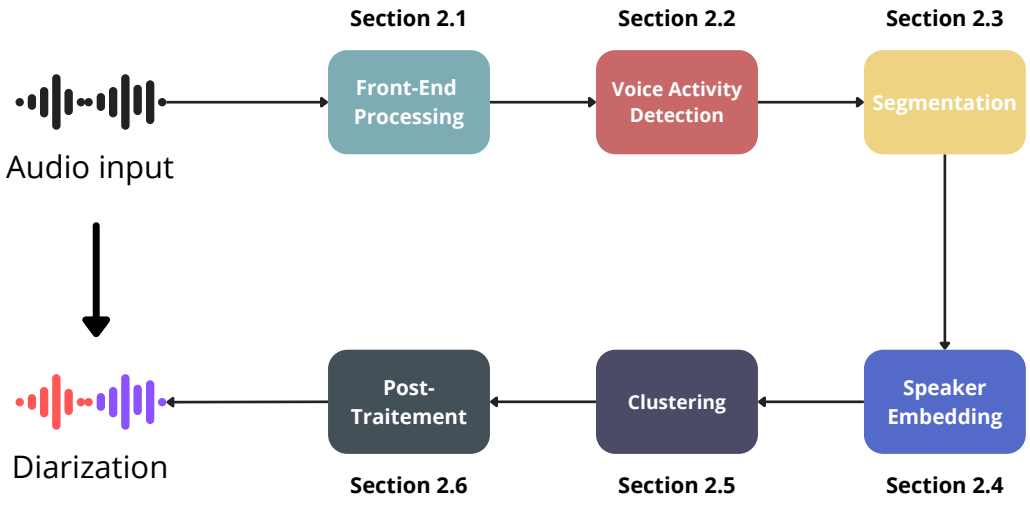

Figure 1: Speaker diarization pipeline: an overview of key stages.

Pyannote Audio exemplifies these advancements. Earlier versions, like 2022.07, demonstrated robust performance but faced challenges with overlapping speech. The VoxSRC2022 update, leveraging additional datasets and refined clustering techniques, achieved a diarization error rate (DER) of 5.6% on the VoxSRC2022 test set, marking a significant improvement [5]

Beyond just identifying segments, the evaluation of diarization quality is critical—much like judging a choir, where it's not just about knowing who is singing, but ensuring they hit the right notes in harmony. While metrics such as Word Error Rate (WER) or Jaccard Error Rate (JER) are useful in certain audio contexts, the Diarization Error Rate (DER) remains the most commonly used metric for evaluating speaker diarization systems. DER accounts for different types of errors, including False Alarms (FA), Missed Detections (MD) and Speaker Confusions (SC) [2].

The values for **FA**, **MD**, and **SC** are expressed in seconds, respecting the temporal nature of the task. The DER itself is a dimensionless value between 0 and 1, often expressed as a percentage.

DER Calculation

$$DER = \frac{FA + MD + SC}{\text{Total Duration}}$$

The lower the DER, the more accurate the speaker diarization. Understanding DER is thus essential for interpreting system performance.

2 Steps in Speaker Diarization

The diarization pipeline can be split into distinct stages, as shown in Figure 1, providing a clear way to handle the task. A more detailed breakdown is shown in Figure 2 below:

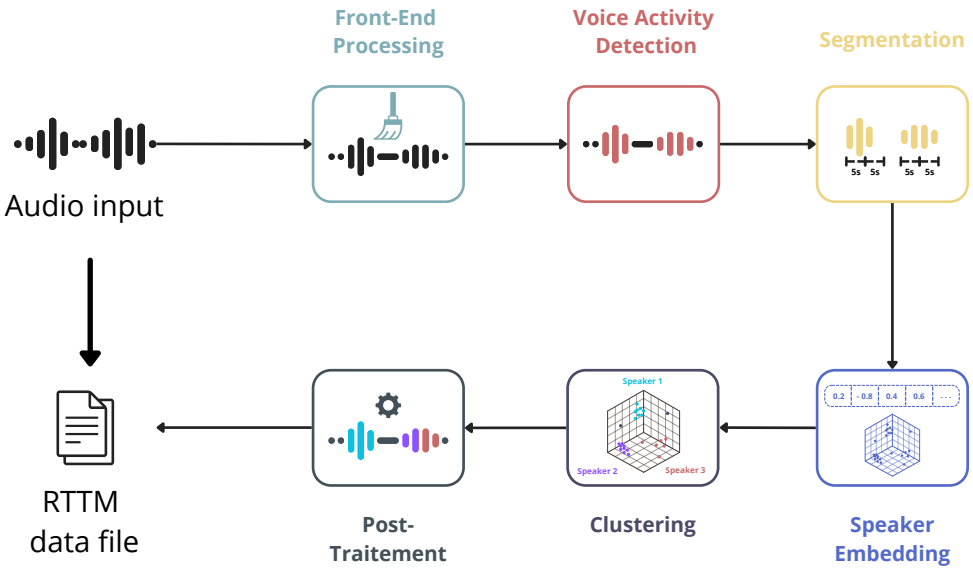

Figure 2: Detailed speaker diarization pipeline.

Each step in the pipeline brings its own challenges, from processing the audio files to producing the final output. Below, we explore each step and its role in ensuring accurate diarization.

2.1 Front-End Processing

The front-end processing stage is crucial in preparing the audio for analysis. It focuses on enhancing the quality of the input signal to ensure accurate diarization in later stages. This stage involves several techniques:

- **Beamforming:** Beamforming is used to enhance the audio by isolating the signal coming from a specific direction. This technique is particularly effective in environments with multiple noise sources, helping to focus on the speaker of interest.
- **Dereverberation:** Reverberation can distort the audio, especially in large or echo-prone environments. Dereverberation techniques work to reduce these echoes, providing a clearer signal that is easier to process in subsequent steps.
- **Speech Enhancement:** This process removes background noise and interference, ensuring the audio is clean and suitable for analysis. Techniques like noise suppression and signal filtering are often used here [2].

2.2 Voice Activity Detection (VAD)

Voice Activity Detection (VAD), also referred to as Speech Activity Detection (SAD), is a fundamental step in the speaker diarization pipeline. Its primary goal is to distinguish speech from non-speech events, such as silence or background noise, ensuring that only the important parts are sent to the next steps [2].

The VAD process has two main steps:

- **Detection of Speech Segments:** The audio signal is analyzed frame by frame to classify each segment as either speech or non-speech. Traditional methods relied on statistical models such as Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) to achieve this classification [6].
- **Application of Thresholds:** A predefined threshold is applied to the output probabilities to finalize the classification, filtering out background noise and silence. This ensures a clean input for the next steps in the diarization pipeline [2].

Accurate VAD is essential because errors at this stage can impact downstream tasks such as speaker segmentation and clustering. While traditional methods relied on statistical models, modern approaches leverage neural networks to achieve greater robustness in challenging conditions [2].

2.3 Segmentation

Segmentation divides the audio into smaller parts to simplify the diarization process. This step makes sure that the audio is analyzed in logical even portions, facilitating speaker identification and clustering. Three common approaches are:

- **Uniform Segmentation:** The audio is divided into fixed-length segments (usually 2 to 5 seconds) without considering when speakers change. This method is the easiest to implement but does not always match exactly the time where speakers change, leaving to potential inaccuracies in speaker attribution [2]. Nevertheless, uniform segmentation is often used in modern systems because it works well with methods like i-vectors or neural embeddings. Generally, shorter segments reduce the chance of mixing multiple speakers, while longer segments give more reliable and logistic speaker information, in order to create a better speaker embedding. Finding the right balance is key here.

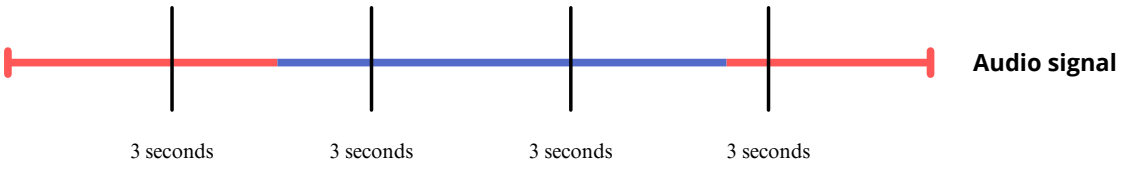

Figure 3: Uniform segmentation of audio into 3 second segments.

- **Speaker-Change Detection:** This approach identifies precisely the points in the audio where the speaker changes, creating segments that match more accurately to

the speaker's boundaries. Techniques such as Kullback-Leibler (KL) divergence, Generalized Likelihood Ratio (GLR) and Bayesian Information Criterion (BIC) are used to detect these changes based on statistical properties of the signal [6]. Speaker-change detection was *the gold standard* for earlier diarization systems, but in this way the segments it creates often have varying lengths, which can make speaker representation less consistent in modern systems. Even so, this approach is still useful for applications requiring precise segmentation.

- **End-to-End Segmentation:** End-to-end speaker segmentation represents a new and modern approach that combines multiple tasks into one single system, simplifying the process significantly. In this way, this method handles everything together using neural networks [7].

  These models divide the audio into overlapping chunks, typically 5 seconds long, and analyze them with fine-grained resolution, making predictions every 16 milliseconds. Each frame is analyzed not only to detect speech but also to identify speaker changes and overlapping speech. By treating the problem as a multi-label classification task, end-to-end models assign probabilities to each frame, showing which speakers are active at a given moment. Techniques like Permutation-Invariant Training (PIT) ensure accuracy, even in scenarios with overlapping speakers. This method outputs a continuous segmentation map that simplifies downstream processes like clustering.

2.4 Speaker Embeddings

Speaker embeddings are small sets of numbers that represent the unique sound of a person's voice. These include things like timbre, tone and speaking style, making it easier to tell speakers apart using their distinct vocal signatures. Speaker embeddings are a key part of modern diarization systems, linking older statistical methods to newer deep learning methods.

Role of Speaker Embeddings in Diarization

In speaker diarization, embeddings give a small but detailed representation of a speaker's voice. They make it easier to group parts of speech that come from the same person. By capturing the unique sound of each speaker, embeddings also help handle overlapping speech, which was a big problem for older methods.

Types of Speaker Embeddings

Different methods have been created to generate speaker embeddings, each with its own strengths. Here are some common types:

- **i-vectors (identity vectors):** i-vectors are statistical embeddings that turn speaker and session information into a small set of numbers. They were one of the first widely used methods and work well for short audio clips. However, i-vectors struggle in noisy environments or when speakers overlap, as they need clean audio.

- **d-vectors (deep vectors):** These are created using deep learning, such as neural networks trained to identify speakers. For example, a recurrent neural network (RNN) can process audio and produce a d-vector that represents the speaker. D-vectors are stronger than i-vectors but still have trouble with overlapping or very short speech.

- **x-vectors:** These improve on d-vectors by using Time Delay Neural Networks (TDNNs). X-vectors summarize information across time by capturing both short-term details, like quick changes in pitch or pronunciation, and long-term details, like the overall tone and speaking style of a person. For example, the TDNN looks at small parts of speech to understand quick changes but also connects them over time to see the bigger pattern in someone's voice. This balance makes x-vectors better for challenging situations, like noisy rooms or audio with many speakers.
- **ECAPA-TDNN:** The ECAPA-TDNN (Emphasized Channel Attention, Propagation and Aggregation in TDNN) builds on x-vectors with new features like attention, layer aggregation and residual connections. These upgrades make ECAPA-TDNN embeddings stronger in noisy conditions and with overlapping speech.

Extraction of Speaker Embeddings

Speaker embeddings are generated from the segments created during the segmentation step (as described in Section 2.3). Each segment is processed by a neural network, like a TDNN or ECAPA-TDNN, to create an embedding. This embedding captures the speaker's voice while ignoring noise and silence.

Speaker embeddings improve diarization by making clustering more accurate and handling overlapping speech better. For instance, ECAPA-TDNN embeddings have shown a significant reduction in Diarization Error Rate (DER) compared to older methods. These embeddings prepare the way for clustering, where similar representations are grouped to identify and separate speakers effectively.

2.5 Clustering

Clustering is a key step in speaker diarization that involves grouping similar speaker embeddings together to identify and separate speakers. This process is essential for assigning segments of speech to the correct speaker, especially when dealing with multiple speakers in the same audio.

Methods of Clustering

Over the years, various clustering methods have been used in speaker diarization systems, each with strengths and limitations:

- **Agglomerative Hierarchical Clustering (AHC):** This method starts with each segment as its own group. It merges the two most similar groups again and again until a set number of groups is reached. AHC often uses scores like cosine similarity or PLDA [2, 3].
- **Spectral Clustering:** Turns speaker similarity scores into a graph and splits it into groups by analyzing the graph's structure. This method works well with complex data, such as overlapping speech [8].
- **K-Means Clustering:** Groups embeddings into a set number of groups by finding the closest center for each embedding. The centers are updated repeatedly to reduce differences within each group. K-Means needs to know the number of speakers in advance [1].

- **Variational Bayesian (VB) Clustering:** Uses a probabilistic model to group embeddings and improves these groups step by step. It is strong in handling noise and changes in speaker data [6].
- **Deep Clustering:** Neural networks map speaker embeddings to a new space where it is easier to separate groups. Extra rules are added to improve accuracy [4].

Clustering in Pyannote

Pyannote makes use of advanced clustering techniques to enhance the accuracy of speaker diarization. Key features include:

- **Embedding Quality:** Pyannote relies on high-quality embeddings like ECAPA-TDNN to ensure that clustering is both precise and robust [3].
- **Dynamic Speaker Models:** Pyannote accommodates variations in the number of speakers dynamically, making it adaptable to diverse audio conditions [8].
- **Iterative Refinement:** Pyannote integrates resegmentation techniques, improving cluster boundaries and accuracy over time [6].

Challenges in Clustering

Despite advancements, clustering in diarization faces challenges:

- **Overlapping Speech:** Handling segments where multiple speakers talk simultaneously remains difficult for traditional clustering methods [2].
- **Unknown Speaker Counts:** Real-world audio often contains an unknown number of speakers, complicating clustering decisions [1].
- **Noise and Variability:** Background noise and speaker variability can degrade clustering accuracy [3].

Future Directions

While clustering remains essential for systems like Pyannote, future trends include:

- **Unsupervised Learning:** Enhancing clustering to work effectively without labeled data.
- **Hybrid Approaches:** Combining traditional clustering with neural models for improved robustness [7].

Clustering continues to be a critical step in speaker diarization, linking speaker embeddings to meaningful groupings that enable accurate speaker assignment and segmentation.

2.6 Post-Treatment

Refines and improves the diarization output:

- **Resegmentation:** Viterbi, VB-HMM
- **Fusion:** DOVER, DOVER-Lap
- **Outlier Management**

2.7 Output: RTTM File

Final results are stored in RTTM format with speaker labels and timestamps.

3 Methodology

3.1 Dataset, Tools and Implementation

We used:

- Audio File: `audio_DER.wav`
- Reference Labels: `ref_lab_audio_DER.lab`
- Hypothesis Labels: `hyp_lab_audio_DER.lab`
- Library: `pyannote.audio` (pre-trained `pyannote/speaker-diarization-3-1`)

Steps:

1. Apply the diarization pipeline.
2. Save results in RTTM or Lab format.
3. Compare reference and hypothesis to compute DER.

4 Results and Analysis

4.1 Diarization Results

The temporal segmentation of speakers is shown below (Speaker 00 in blue, Speaker 01 in red).

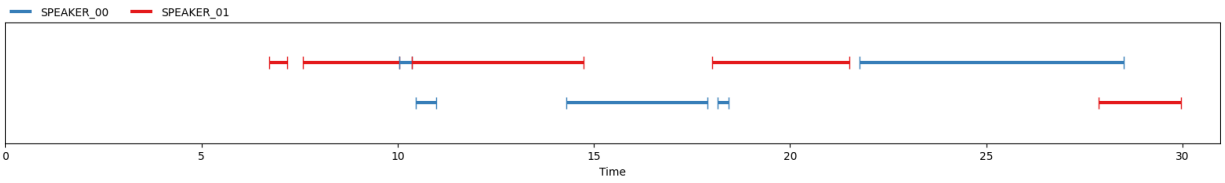

Figure 4: Diarization output showing speaker activity over time.

4.2 Hypothesis vs Reference

Comparing hypothesis labels against reference highlights differences in segments:

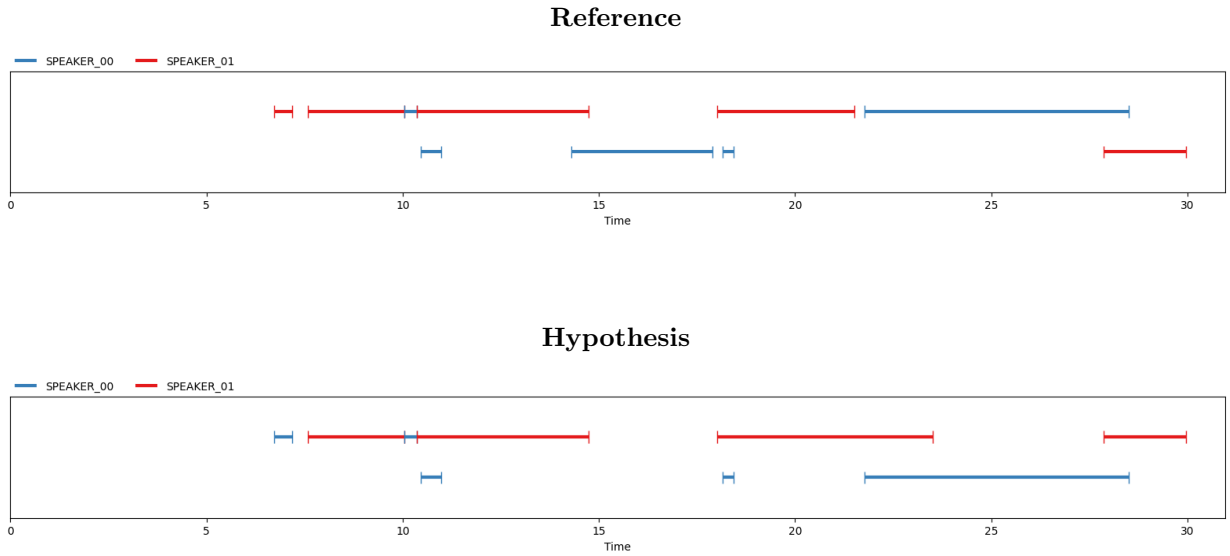

Figure 5: Reference (top) vs Hypothesis (bottom).

4.3 Error Types

Examining errors between reference and hypothesis:

- **Speaker Confusion (SC):** Segments where the speaker is misattributed in the hypothesis. For example:
  - From `6.730s` to `7.186s`, the hypothesis identifies `SPEAKER_00`, but the reference indicates `SPEAKER_01`.
- **Missed Detections (MD):** Segments present in the reference but missing in the hypothesis. For example:
  - From `14.307s` to `17.885s`, `SPEAKER_00` is present in the reference but is not detected in the hypothesis.
- **False Alarms (FA):** Segments present in the hypothesis but absent in the reference. For example:
  - From `18.020s` to `23.513s`, the hypothesis detects `SPEAKER_01`, but the reference indicates the segment ends earlier, at `21.513s`, so a mistake of `2.0s`.

Start (s)	End (s)	Reference Speaker	Hypothesis Speaker	Error Type	Gap (s)
6.730	7.186	SPEAKER_01	SPEAKER_00	**SC**	0.456
14.307	17.885	SPEAKER_00	-	**MD**	3.578
21.513	23.513	-	SPEAKER_01	**FA**	2.0

Table 1: Summary of identified errors.

The following errors were calculated by comparing the diarization results (used as reference labels) with the hypothesis labels:

- **False Alarms (FA):** 2.0 seconds.
- **Missed Detection (MD):** 3.578 seconds.
- **Speaker Confusion (SC):** 0.456 seconds.
- **Total Duration *(all the audio, even if silence)*:** 24.334 seconds.

Using these values, the Diarization Error Rate (DER) is calculated as follows:

1. Sum the errors:

$$\text{Total Errors} = 2.0 + 3.578 + 0.456 = 6.034\,\text{seconds}.$$

2. Divide by the total duration:

$$DER = \frac{6.034}{24.334} \approx 0.24797.$$

3. Convert to a percentage:

Diarization Error Rate

$$DER = 24.8\%.$$

5 Detailed Results

The detailed results are shown below:


```
{
  'confusion': 0.4559999999999995,
  'correct': 20.3,
  'diarization error rate': 0.24796580915591357,
  'false alarm': 2.0,
  'missed detection': 3.578000000000001,
  'total': 24.334
}
```

6 Conclusion

This work presented an overview of the speaker diarization pipeline and the application of the Diarization Error Rate (DER) to evaluate system performance. By analyzing errors such as FA, MD and SC, DER provides a comprehensive measure of quality. The example showed where improvements are needed, notably in handling overlaps and ensuring correct speaker attribution.

Future work could focus on alternative metrics (e.g. JER) and exploring advanced models to further minimize diarization errors, ultimately enhancing robustness and accuracy.

A Appendix: Reference and Hypothesis Labels

A.1 Reference Labels

```
6.730 7.186 SPEAKER_01
7.591 10.038 SPEAKER_01
10.038 10.358 SPEAKER_00
10.358 14.746 SPEAKER_01
10.460 10.983 SPEAKER_00
14.307 17.885 SPEAKER_00
18.020 21.513 SPEAKER_01
18.155 18.442 SPEAKER_00
21.766 28.499 SPEAKER_00
27.858 29.967 SPEAKER_01
```

A.2 Hypothesis Labels

```
6.730 7.186 SPEAKER_00
7.591 10.038 SPEAKER_01
10.038 10.358 SPEAKER_00
10.358 14.746 SPEAKER_01
10.460 10.983 SPEAKER_00
18.020 23.513 SPEAKER_01
18.155 18.442 SPEAKER_00
21.766 28.499 SPEAKER_00
27.858 29.967 SPEAKER_01
```

References

[1] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, page 1, 2010. ffhal-00733397f.

[2] 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, 2021.

[3] Nauman Dawalatabad, Mirco Ravanelli, François Grondin, Jenthe Thienpondt, Brecht Desplanques, and Hwidong Na. Ecapa-tdnn embeddings for speaker diarization. In *Interspeech 2021*, interspeech_2021. ISCA, August 2021.

[4] Yusuke Fujita, Shinji Watanabe, Shota Horiguchi, Yawen Xue, and Kenji Nagamatsu. End-to-end neural diarization: Reformulating speaker diarization as simple multi-label classification, 2020.

[5] Hervé Bredin. pyannote.audio speaker diarization pipeline at voxsrc 2022. 2022.

[6] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. pyannote.audio: neural building blocks for speaker diarization, 2019.

[7] Hervé Bredin and Antoine Laurent. End-to-end speaker segmentation for overlap-aware resegmentation, 2021.

[8] Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *24th INTERSPEECH Conference (INTERSPEECH 2023)*, pages 1983–1987, Dublin, Ireland, August 2023. ISCA.