A Comparative Analysis of Speaker Diarization Models: Creating a Dataset for German Dialectal Speech

Lea Fischbach
Research Center Deutscher Sprachatlas
Marburg, Germany
Lea.Fischbach@uni-marburg.de

Abstract

Speaker diarization is a critical task in the field of computer science, aiming to assign timestamps and speaker labels to audio segments. The aim of these tests in this Publication is to find a pretrained speaker diarization pipeline capable of distinguishing dialectal speakers from each other and an explorer. To achieve this, three pipelines, namely Pyannote, CLEAVER and NeMo, are tested and compared, across various segmentation and parameterization strategies. The study considers multiple scenarios, such as the impact of threshold values for speaker recognition and overlap handling on classification accuracy. Additionally, this study aims to create a dataset for German dialect identification (DID) based on the findings from this research.

1 Introduction

Speaker diarization (SD) models are essential in various applications, including speech-to-text systems. Since existing SD systems may not always meet specific requirements, comparing multiple models is necessary to identify the most suitable one, aiding both our research and guiding other researchers.

The annual DIHARD Challenge, focusing on SD for challenging audio files, has been held since 2018 (Ryant et al., 2020). However, as demonstrated in this contribution to the Challenge (Horiguchi et al., 2021) and in an paper, which provides an overview of SD systems (Tranter and Reynolds, 2006), the commonly used evaluation metric is the one called diarization error rate (DER), based on False Alarm, Missed Detection, and Speaker Confusion. In our case this metric is not applicable due to the nature of our ground truth data. The recordings used are from the REDE corpus (Schmidt et al., 2020ff.), featuring 1-2 elderly male speakers translating sentences into their local dialect and an explorer providing this sentences beforehand in Standard German. Our manually segmented ground truth data only contains these translated sentences from the dialectal speaker, however the original recordings include additional utterances from the dialectal speaker, between these sentences, that we aim to retain. Using DER could distort results because the SD model might correctly identify an dialectal speaker during these intervals not captured in the ground truth data. Therefore, we bypass DER and use a dialectal classification model, comparing its accuracy on our manually segmented ground truth data (the resulting performance is our baseline) with the accuracy from the model on audio files created by the different SD pipelines. With clearly separated speakers, the recordings used for testing the model contain only the desired dialectal speaker and thus the model should perform better, because the explorer does not speak any dialect and thus would interfere with classification.

We utilize SD models including Pyannote (Bredin et al., 2020; Bredin and Laurent, 2021) (v2.1), CLEAVER[1], and NVIDIA NeMo (Harper et al.), highlighting their strengths, weaknesses, and key features. This comparative analysis provides valuable insights, saving researchers time and effort in selecting the most suitable model. The goal is to establish a dataset for German dialect identification (DID), advancing research in this field.

2 Overview

Speaker diarization (SD) is the task of assigning timestamps and corresponding speaker labels to an audio track. In general, pipelines designed to accomplish this task consist of four sub-tasks. Depending on the categorization and assignment, there can be even more sub-tasks, as seen in works such as (Tranter and Reynolds, 2006), where all

[1] https://www.oxfordwaveresearch.com/products/cleaver/

Proceedings of the 3rd Workshop on NLP Applications to Field Linguistics (Field Matters 2024), pages 43–51

possible sub-tasks are analyzed, or in (Park et al., 2022) where pre- and post-processing also constitute sub-tasks.

The first of the four sub-tasks is Voice Activity Detection (VAD), which identifies when speech is present and removes non-speech sections from the audio. This is followed by segmentation, also known as Speaker Change Detection (SCD), which recognizes speaker transitions and divides the audio into individual speaker segments. Next, local speaker embeddings (SE) are extracted using a pretrained model. These embeddings are then utilized in a clustering algorithm, where speakers with similar embeddings are grouped together to label the speakers globally.

2.1 Pyannote

Pyannote is an open-source library built on PyTorch. There is a key difference between their model and many others: they use short audio chunks but with a higher temporal resolution of 16ms (Bredin and Laurent, 2021). This means that every 16ms, the model calculates the probability of each possible speaker being active. The use of shorter audio chunks plays a crucial role because they typically involve fewer speakers and exhibit less speaker variability, simplifying the task.

Another distinctive feature of Pyannote is its consideration of concurrent speakers. To account for this, a SE for an individual speaker is constructed only from the (concatenated) segments in which that speaker exclusively speaks. They refer to this approach as "overlap-aware" (Bredin, 2023). However, the accuracy of these segments depends on the primary segmentation task.

2.2 CLEAVER

Oxford Wave Research's CLEAVER (Cluster Estimation And Versatile Extraction of Regions) differs in that it utilizes phonetic features. For the SCD, it relies on pitch, which is extracted using Praat (Boersma and Weenink, 2023; Alexander and Forth, 2012). Whenever this pitch significantly deviates, either in time or frequency, a speaker change is detected, resulting in individual segments. Subsequently, using a statistical model, the most distinct segments are identified. These segments then undergo clustering, where all other segments are assigned to one of them. Following this, another clustering step takes place, where segments previous assigned to their respective speakers form the new start SE. This process continues until the clusters no longer change.

2.3 NeMo

NeMo (Neural Modules) is an open-source library developed by NVIDIA, built on PyTorch. This framework includes various tools in the field of Natural Language Processing. Its processes are optimized to work with a CUDA-compatible GPU[2]. Although NeMo is designed to be framework-agnostic, it currently supports only PyTorch as a backend.

A unique aspect of NeMo's SD pipeline is the inclusion of a "neural diarizer" after the clustering step[3]. This diarizer is applied to the speaker profiles obtained from clustering and is a trainable neural model. It assigns speaker labels even to overlapping speakers, which cannot be achieved with clustering alone. The process involves using a clustering diarizer to estimate the speakers profiles and the number of speakers by employing a pairwise (two-speaker) unit model for both training and inference.

Another advantage of NeMo is the concept of Multiscale Segmentation[3]. Normally, a speaker embedding (SE) is generated for each speaker segment. If long segments (over 3 seconds) are used, the speaker profile is reliable, but temporal information is lost since speaker changes can only be detected every 3 seconds. When short segments (0.5~3.0 seconds) are used, the speaker profile depends on a brief utterance by the speaker, making the SE unreliable. To address this issue, Multiscale Segmentation is employed. Segmentations of different lengths, which overlap, are utilized. For example, the audio is divided into segments of 0.5 seconds, 0.75 seconds, etc. Information from each segmentation is then combined and used for global speaker labels. Additionally, the smallest segmentation level is used as the temporal resolution, allowing the model to more accurately capture rapid changes in speaker activity.

3 Experimental Setup

In this section, we outline our experimental setup and the exploration of various parameters for our study. We analyze a total of 20 different Ger-

[2] https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/speaker_diarization/intro.html

[3] https://github.com/NVIDIA/NeMo/blob/main/tutorials/speaker_tasks/Speaker_Diarization_Inference.ipynb

man dialects, classified according to Wiesinger (Wiesinger, 1983), amounting to 60.5 hours of audio data. Our focus narrows down to two specific dialects, contributing a combined 11.75 hours of audio, which are already annotated and clearly segmented, allowing them to serve as ground truth. We establish a baseline by assessing the model's accuracy on these manually segmented dialects.

The entire pipeline[4], from the normalization of the audios to obtaining the classification accuracy, is depicted in Figure 1. As a first step, all audio files undergo preprocessing to standardize their format. For this we chose a sample rate of of 16kHz driven by the requirements of Googles TRILLsson models (Shor and Venugopalan, 2022), where the largest model is employed for extracting feature embeddings. To extract feature embeddings, the audio segments resulting from Speaker diarization (SD) are concatenated per speaker and then divided into segments of 3 seconds each. These segments are then processed through a small convolutional neural network (CNN). The CNN model architecture comprises three dense layers with LeakyReLU activations and dropout layers to mitigate overfitting. For validating and testing the dialect classification model, we randomly selected two speakers from each dialect. Since the results vary depending on the chosen speakers, the steps of dividing the data into training, validation, and test sets and running the model are repeated 250 times, selecting new random speakers for each run. We then compute the mean accuracy out of these 250 runs. This number of runs has proven sufficient in previous tests to detect significant differences between experiments. For testing if the differences of the experiments are significant we use the Mann-Whitney U test (Mann and Whitney, 1947). We examine whether there is a significant difference between the baseline and the models accuracy using the resulting audios from various SD models with their default settings. Additionally, we evaluate whether there is a significant difference between runs using the resulting audios with the standard settings of each SD pipeline (called standard pipeline) and runs using the resulting audios with parameter adjustments for the SD pipelines or different segment extraction methods. This is indicated by the p-value in the tables, which always refers to the top row of each table. If the distribution of accuracy for the respective experiment is significantly better than that of the top row, the p-value is bolded.

Parameter adjustments are explained in the upcoming subsections for each SD pipeline, while extraction methods are tested in the same manner for each SD pipeline. In this context, an extraction method means specifying a threshold in seconds, where only the resulting segments of the SD pipeline longer than this threshold are retained. This threshold helps remove non-contiguous utterances, such as clearing one's throat, if they haven't already been eliminated by the SD pipeline. We then incrementally increase the threshold to assess whether the results improve or worsen.

3.1 Pyannote

To test Pyannote with different parameters and segment extraction methods, we only specify the number of speakers between 1 and 4 in the standard settings. We then compare these standard settings with various segment extraction methods, which consider only segments longer than a set threshold and remove overlapping segments where multiple speakers talk simultaneously. We also evaluate the model's performance when we specify the exact number of speakers, increase the speaker recognition threshold (SR-TH) to make the model more confident in classifying speakers, and set the `min_duration_off` parameter to 0, meaning no intra-speaker gaps are bridged.

3.2 CLEAVER

Since we used only the demo version of CLEAVER, different parameters cannot be tested. In this demo version, an audio file is uploaded to the server via an API, and a segment is selected for each occurring speaker in which only that speaker is active. The results are then presented visually and can be downloaded as a CSV file.

3.3 NeMo

NVIDIA NeMo provides three different configuration YAML files (a human-readable data format used for configuration), each created during model training with different recordings. Detailed information about the used parameters in the YAML files is available on their website[5]. The general YAML file is optimized for balanced performance across various domains. The meeting YAML file

[4] https://github.com/WoLFi22/DialectClassificationPipeline

[5] https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/asr/speaker_diarization/configs.html

is designed for meetings with 3-5 speakers, and the telephonic YAML file is suited for telephone recordings involving 2-8 speakers, as stated in the comments in the corresponding YAML files. Parameters modified in the YAML files include `ignore_overlap` (whether overlapping speech is ignored), `oracle_num_speakers` (whether to use the exact number of speakers from the manifest file), and `min_duration_off` (the threshold for filling speech gaps within a speaker). This last parameter is akin to Pyannotes `min_duration_off` parameter. Other adjusted parameters include `onset` and `offset`. The onset parameter determines the threshold for identifying the start and end of speech segments, while the offset parameter determines the threshold for identifying the end of speech. Finally, `pad_onset` specifies the duration added before each speech segment.

4 Results

Table 1 presents the results of the standard pipelines (SP). The column *Avg. #Segments* refers to the average count of segments, assigned to the dialectal speaker after speaker diarization (SD), per original audio file. Similarly, *Avg. Sec.* represents the average duration of these segments. The column *Mean Accuracy* represents the average accuracy across 250 runs using different train/validation/test data splits.The SP of Pyannote, CLEAVER, and NeMo with the telephonic YAML file perform similarly well. CLEAVER generally extracts more segments than the other two models, but these segments are shorter on average. This is shown in Figure 2 (a), which displays a portion of a file with ground truth labels at the top and the labels of the respective SPs below, with overlapping segments shaded in gray. This figure also highlights that Pyannote is the only model by default that detects overlaps, though it struggles with identifying segments without speech.

NeMo with the telephonic YAML file initially yields the best results. Figure 2 (a) also shows that the segments from NeMo telephonic closely align with the ground truth segments. With the general YAML file, segments are often too long, as reflected in the higher average seconds shown in Table 1 and also visible in Figure 2 (a). The meeting YAML file improves this but still does not match the segmentation quality of the telephonic YAML file. This is likely because the used recordings resemble a telephone conversation, typically involving two speakers who occasionally overlap and speak in succession.

4.1 Pyannote

Specifying the exact number of speakers for segmentation with Pyannote makes little difference, as shown by the nearly identical values in the first and second rows of Table 2. Figure 2 (b) also shows that the segments of the three speakers are almost identical. However, a speaker was occasionally misclassified as another when we provided the exact number of speakers. Thus, providing the exact number of speakers seemed to confuse the model, and during clustering, more distant embeddings were assigned to the same speaker because of the predefined number of clusters.

Removing overlapping segments results in a slight, but not significant, improvement in accuracy. Without bridging intra-speaker gaps results in further subdividing previously connected segments. This leads to more segments of shorter duration, as indicated in Table 2 and shown in Figure 2 (b), but it does not significantly affect the classification models accuracy. When the speaker recognition threshold (SR-TH) is set to 0.8, a significant improvement in classification is observed. With this setting, embeddings are assigned to a speaker only when the model is more confident, resulting in better recognition of larger speaker gaps, as shown in Figure 2 (b).

4.2 CLEAVER

For CLEAVER we can only modify the segment extraction method. However, there is no significant difference between the results with different segment extraction thresholds, as shown in Table 3. The only observed difference is that segments become longer as the threshold increases, while the number of segments decreases accordingly.

4.3 NeMo

Specifying the exact number of speakers for the audio files makes little difference with NeMo (telephonic). This is evident in Figure 2 (c), where the segments from NeMo closely match those with the exact number of speakers, indicating that the speakers were already well recognized. When adding overlap, overlapping speakers are still not detected and the resulting segments are the same as before. The reason for this is unclear and cannot be determined at this time. When intra-speaker gaps are not filled, previously connected segments are further

subdivided, resulting in more segments on average with a shorter duration. However, this does not impact the accuracy of the classification model since the parameter is set to 0.2 by default for the telephonic YAML file, meaning only speaker gaps of 200ms are bridged. Reducing the parameter to 0.0 makes virtually no difference, as all segments of a speaker are concatenated for classification. The same applies to increasing the parameter to 0.5. When increasing the onset and corresponding offset thresholds for recognizing the start and end of speech segments, on average more but shorter segments are generated. As shown in Figure 2 (c), individual long segments are divided into multiple shorter segments as the threshold increases, aligning more closely with the ground truth data. Consequently, the mean accuracy significantly improves starting from a threshold of 0.5 for both parameters. Without padding on the onset, segments simply begin later, as clearly visible in Figure 2 (c). As a result, segments are shorter on average, and more segments are created since some segments are without padding no longer connected. However, this does not affect performance.

Regarding the segment extraction method, removing segments shorter than 0.5 to 1.0 seconds proves significantly better than the standard method, where all segments are retained. This improvement may be attributed to NeMo recognizing and labeling short speech segments, such as coughing or unclear brief expressions, which are filtered out with the extraction method.

5 Conclusion

This study investigated the performance of Speaker diarization (SD) models, Pyannote, CLEAVER, and NeMo, using various parameters and segmentation strategies. Our findings highlight the significant impact of model choice, segmentation method, and parameter settings on the accuracy and effectiveness of SD systems.

For our audio data and classification task, NeMo telephonic, using a higher threshold value of 0.5 for the onset and offset parameter and employing the extraction method that ignores segments shorter than 1.0 second, achieves the highest accuracy at 90.6%. The baseline, composed of manually segmented recordings, achieves a slightly higher accuracy of 91.4%. Achieving baseline accuracy through automatic segmentation based solely on SD poses challenges in our case, because manually segmented recordings contain only relevant dialectal speech, while automatically generated ones also include free, sometimes Standard German, speech by dialectal speakers.

Although NeMo performs slightly better than Pyannote, where segments shorter than 0.5 seconds were ignored or the speaker recognition threshold (SRTH) was increased, the difference is not substantial. Generally, however, it can be said that thanks to the concept of multiscale segmentation, NeMo also identifies shorter segments that are no longer recognized by Pyannote. Removing shorter segments resulting from speaker diarization, typically less than one second in duration, consistently improved accuracy. These segments are likely too short to contain coherent utterances from one speaker and instead often include background noise or filler words.

Since CLEAVER performs similarly well to Pyannote and NeMo without further adjustments, CLEAVER is a good alternative for those who prefer a visual representation.

It is also important to consider that perfect segmentation is not always necessary for practical purposes. Higher accuracy is of course better, but even slightly less accurate segmentation can still save time compared to manual segmentation. When adjusting the speaker recognition thresholds and the thresholds for identifying the start and end of speech segments, a balance must be struck between capturing every part of the audio where the desired speaker speaks (accepting more noise and larger speaker pauses or occasional misidentified speakers) and achieving higher precision (potentially missing some parts of the speakers speech and failing to assign some segments to the correct speaker).

With these insights, recordings from the REDE corpus can now be processed to create a new dataset for German dialect classification.

Acknowledgements

This research is supported by the Academy of Science and Literature Mainz (grant REDE 0404), the Federal Ministry of Education and Research of Germany (BMBF) (grant AnDy), and the Research Center Deutscher Sprachatlas in Marburg. Furthermore, thanks are due to Oxford Wave Research Ltd for granting access to the demo version of CLEAVER.

References

Anil Alexander and Oscar Forth. 2012. Blind speaker clustering using phonetic and spectral features in simulated and realistic police interviews.

Paul Boersma and David Weenink. 2023. Praat: doing phonetics by computer [Computer program]. http://www.praat.org/.

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

Hervé Bredin and Antoine Laurent. 2021. End-to-end speaker segmentation for overlap-aware resegmentation. pages 3111–3115.

Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. Pyannote.audio: Neural building blocks for speaker diarization. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7124–7128.

Eric Harper, Somshubra Majumdar, Oleksii Kuchaiev, Li Jason, Yang Zhang, Evelina Bakhturina, Vahid Noroozi, Sandeep Subramanian, Koluguri Nithin, Huang Jocelyn, Fei Jia, Jagadeesh Balam, Xuesong Yang, Micha Livne, Yi Dong, Sean Naren, and Boris Ginsburg. NeMo: a toolkit for Conversational AI and Large Language Models.

Shota Horiguchi, Nelson Yalta, Paola Garcia, Yuki Takashima, Yawen Xue, Desh Raj, Zili Huang, Yusuke Fujita, Shinji Watanabe, and Sanjeev Khudanpur. 2021. The hitachi-jhu dihard iii system: Competitive end-to-end neural diarization and x-vector clustering systems combined by dover-lap. *Preprint*, arXiv:2102.01363.

H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60.

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

Neville Ryant, Kenneth Church, Christopher Cieri, Jun Du, Sriram Ganapathy, and Mark Liberman. 2020. Third dihard challenge evaluation plan. *arXiv preprint arXiv:2006.05815*.

Jürgen Erich Schmidt, Joachim Herrgen, Roland Kehrein, and Alfred Lameli. 2020ff. Regionalsprache.de (REDE III). Forschungsplattform zu den modernen Regionalsprachen des Deutschen. Marburg: Forschungszentrum Deutscher Sprachatlas.

Joel Shor and Subhashini Venugopalan. 2022. TRILLsson: Distilled Universal Paralinguistic Speech Representations. In *Proc. Interspeech 2022*, pages 356–360.

S.E. Tranter and D.A. Reynolds. 2006. An overview of automatic speaker diarization systems. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1557–1565.

Peter Wiesinger. 1983. Die einteilung der deutschen dialekte. In Werner Besch, editor, *Dialektologie: Ein Handbuch zur deutschen und allgemeinen Dialektforschung*, volume 1.2 of *Handbücher zur Sprach- und Kommunikationswissenschaft*, pages 807–900. Berlin/New York: de Gruyter, Berlin, New York.

A Appendix

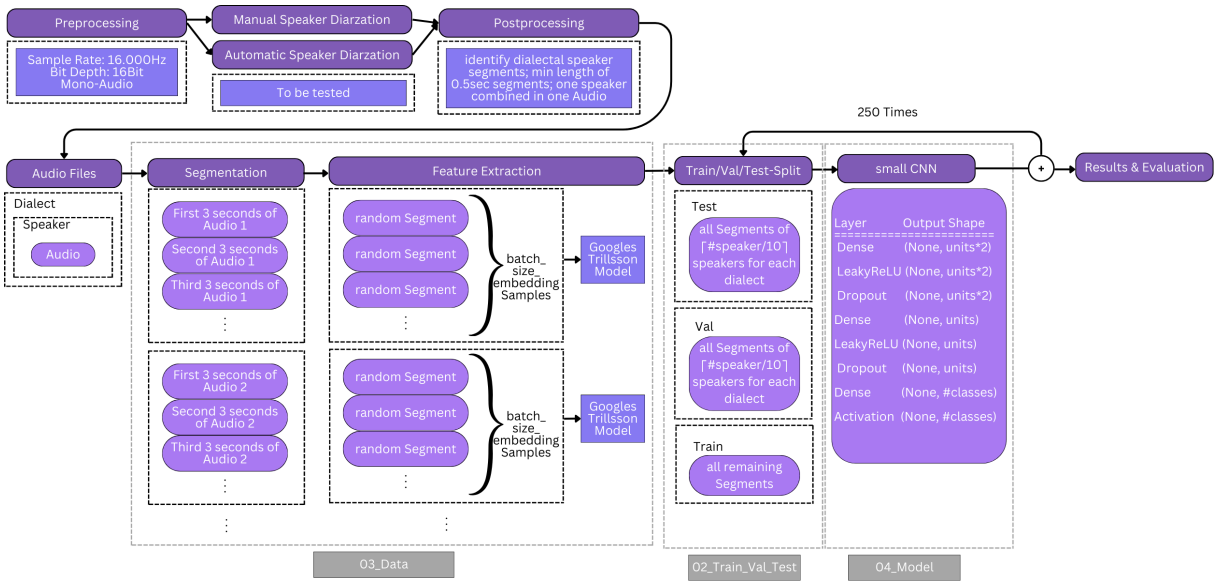
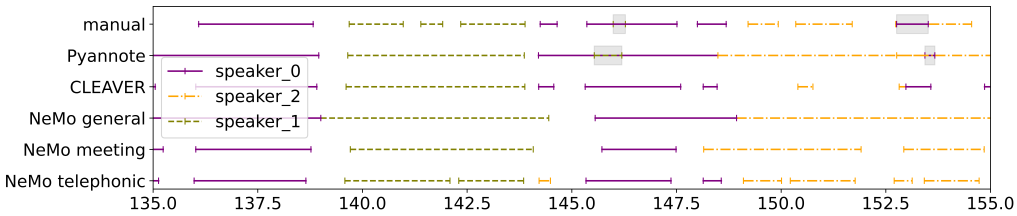


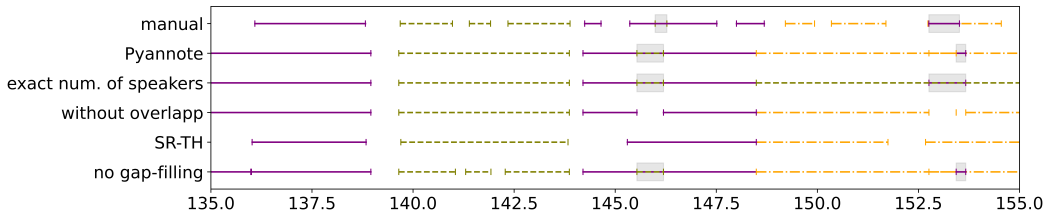
Figure 1: Visualization of used Pipeline

Model	Avg. #Segments	Avg. Time	Mean Accuracy
Baseline	85.257	1.974s	0.914
Pyannote	132.923	2.907s	0.878
CLEAVER	178.553	1.497s	0.877
NeMo general	139.205	3.251s	0.862
NeMo meeting	138.231	2.873s	0.867
NeMo telephonic	206.077	1.688s	0.880

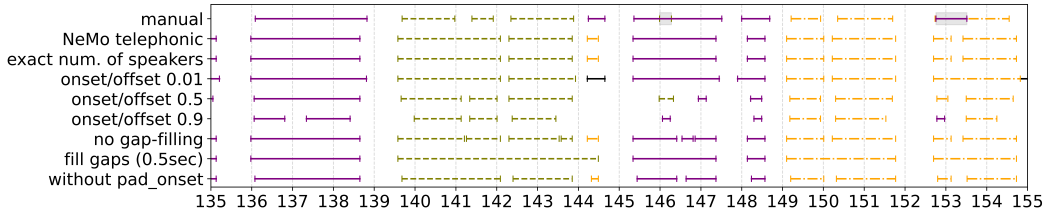
Table 1: Results of standard pipelines



(a) standard pipelines



(b) Pyannote



(c) NeMo

Figure 2: Part of one Audio visualized after speaker diarization for the different models and parameters.

Params	**Extract. Method**	**Avg. #Segments**	**Avg. Time**	**Mean Acc.**	**p-value**
SP	-	132.923	2.907s	0.878	-
exact num. of speakers	-	132.87	2.85s	0.882	0.28
-	0.2sec.	130.10	2.97s	0.888	0.11
-	0.5sec.	123.33	3.12s	**0.889**	**0.05**
-	1.0sec.	111.33	3.35s	0.884	0.20
-	without overlap	134.31	2.64s	0.877	0.74
no gap-filling	-	167.49	2.30s	0.874	0.61
SR-TH 0.8	-	140.31	2.39s	**0.890**	**0.01**
SR-TH 0.8	0.5sec	126.28	2.60s	**0.890**	**0.02**

Table 2: Results from Pyannote

Params	**Extract. Method**	**Avg. #Segments**	**Avg. Time**	**Mean Acc.**	**p-value**
SP	-	178.553	1.497s	0.877	-
-	0.2sec.	162.97	1.62s	0.885	0.74
-	0.5sec.	134.13	1.88s	0.880	0.79
-	1.0sec.	101.74	2.21s	0.881	0.25

Table 3: Results from CLEAVER

Params	Extract. Method	Avg. #Segments	Avg. Time	Mean Acc.	p-value
SP (telephonic)	-	206.077	1.688s	0.880	-
exact num. of speakers	-	207.39	1.70s	0.879	0.44
-	0.2sec.	204.41	1.70s	0.882	0.24
-	0.5sec.	184.13	1.83s	**0.893**	**0.04**
-	1.0sec.	133.62	2.18s	**0.894**	**0.01**
-	1.5sec.	95.03	2.52s	0.883	0.19
with overlap	-	206.08	1.69s	0.880	0.38
onset/offset 0.01	-	185.59	1.99s	0.882	0.31
onset/offset 0.5	-	239.67	1.24s	**0.902**	**0.00**
onset/offset 0.9	-	300.82	0.76s	**0.891**	**0.00**
no gap-filling	-	257.13	1.35s	0.884	0.13
fill gaps (0.5sec)	-	163.21	2.26s	0.873	0.88
without pad_onset	-	230.31	1.41s	0.889	0.05
onset/offset 0.5	1.0sec.	112.28	1.87s	**0.906**	**0.00**

Table 4: Results from NeMo (telephonic)