Speaker Diarization

Link to notebook: SpeakerDiarization-Vallabh
Link to EDA: EDA-SpeakerDiar

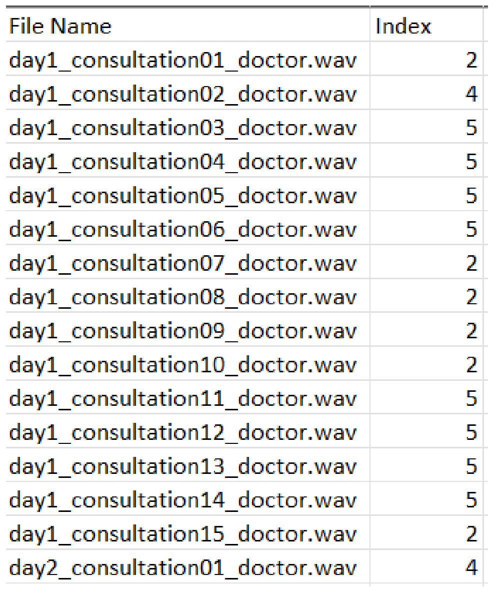
Dataset:

https://github.com/babylonhealth/primock57 The dataset consists of 57 mock medical primary care consultations held over 5 days by 7 Babylon clinicians and 57 Babylon employees acting as patients, using case cards with presenting complaints, symptoms, medical & general history etc.

Note: User need to split Train/Test/Dev data

Dataset study:

There are a total of 57 conversations between doctors and patients suggesting that each patient appeared only once . I have performed clustering using TSNE,using mfcc features, taking 7 clusters for doctor audio files and 57 clusters for patient audio files. So I was able to map in which audio file, which doctor/patient spoke. I cross verified with transcripts also as in some conversations, doctor mentioned his name, checking another file of same doctor, same name was told implying that clustering was correct.



File Name	Index
day1_consultation01_doctor.wav	2
day1_consultation02_doctor.wav	4
day1_consultation03_doctor.wav	5
day1_consultation04_doctor.wav	5
day1_consultation05_doctor.wav	5
day1_consultation06_doctor.wav	5
day1_consultation07_doctor.wav	2
day1_consultation08_doctor.wav	2
day1_consultation09_doctor.wav	2
day1_consultation10_doctor.wav	2
day1_consultation11_doctor.wav	5
day1_consultation12_doctor.wav	5
day1_consultation13_doctor.wav	5
day1_consultation14_doctor.wav	5
day1_consultation15_doctor.wav	2
day2_consultation01_doctor.wav	4

Snippet showing allocation of doctor indexing to files, obtained by tsne clustering.
Similarly done for patients, and verified that the results are correct by hearing and also through names in transcript.

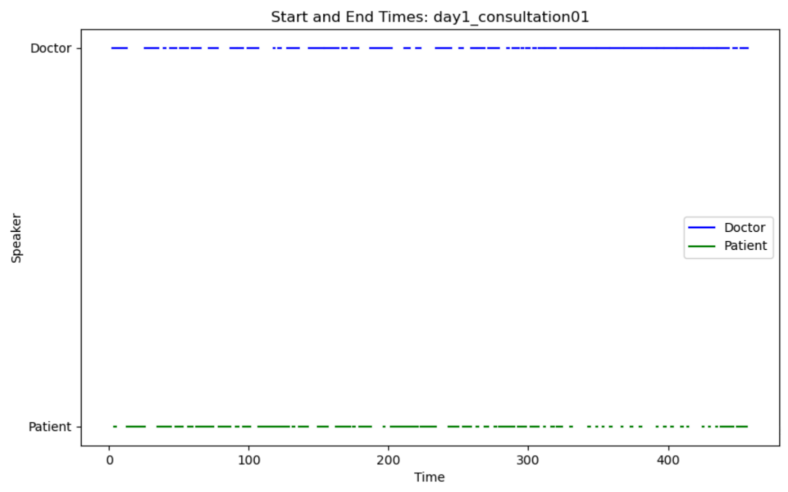
EDA:

Duration Study:

We observed the following statistics on duration of mixed audio:
Mean Duration: 9.09 minutes
Minimum Duration: 3.81 minutes
Maximum Duration: 14.30 minutes
Standard Deviation: 2.05 minutes
Median Duration: 9.32 minutes

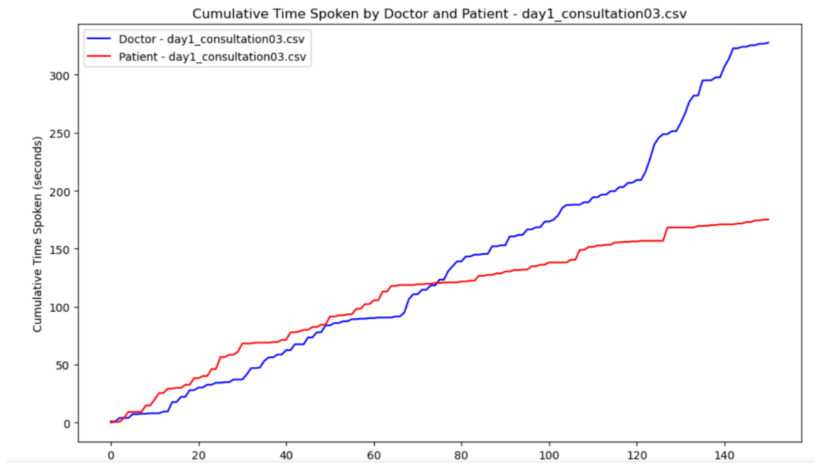
Overlap of doctor and patient study in the transcript start-end data:

```
Maximum overlap duration for day1_consultation01: 4.95 seconds
Maximum overlap duration for day1_consultation02: 1.76 seconds
Maximum overlap duration for day1_consultation03: 2.49 seconds
Maximum overlap duration for day1_consultation04: 2.94 seconds
Maximum overlap duration for day1_consultation05: 1.94 seconds
Maximum overlap duration for day1_consultation06: 2.26 seconds
Maximum overlap duration for day1_consultation07: 5.51 seconds
Maximum overlap duration for day1_consultation08: 3.28 seconds
Maximum overlap duration for day1_consultation09: 3.01 seconds
```



We observed overlaps in speech of doctor and patient, this might cause more der of pretrained model as it segments more robustly, compared to manual transcription. So if der is high,we can split the transcripts to minimize error caused by overlap.

Cummulative speech length study:



General trend: Patient speaks more in the beginning, doctor speaks more later which correlates correctly as patient tells his problems first, and later doctor gives consultation.

Preprocessing:

Audio files are in such a way that doctor and patient speech are separated in separate files, filled with silence. I combined both to get mixed audio using sox. Also I got transcripts for each conversation by combining independent transcripts. I made csvs for each conversation with Speaker, Start and End columns.

Accuracy Evaluation:

Checking accuracy:

Used DER metric

The Diarization Error Rate (DER) is calculated as the sum of three types of errors:

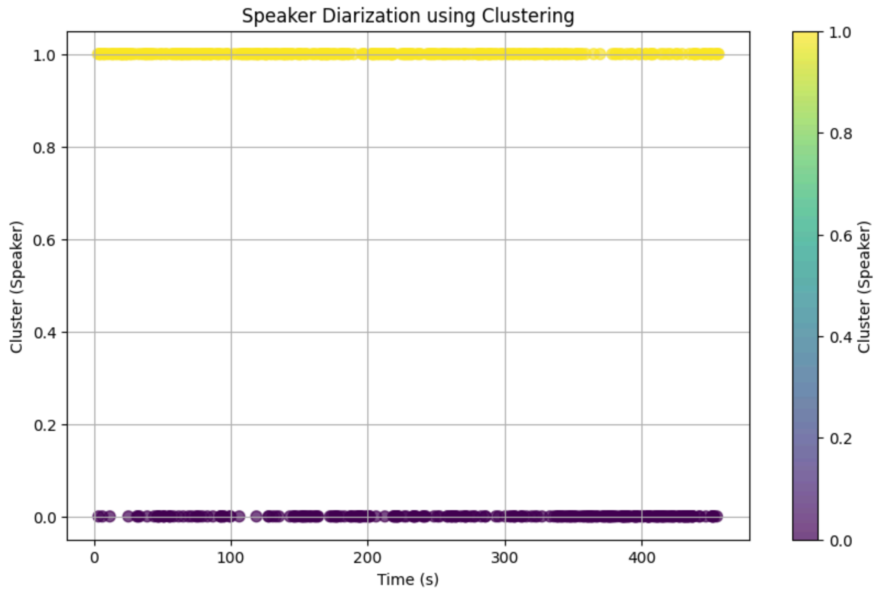
1. **Misses (M)**: When a speaker's speech is not detected (i.e., the system misses a segment where the speaker was talking).
2. **False Alarms (FA)**: When the system incorrectly assigns speech to a non-existent speaker or splits a segment incorrectly.
3. **Confusions (C)**: When the system mislabels a speaker (i.e., assigning the wrong speaker label to a segment).

The formula for DER is:

$$\mathrm{DER} = \frac{M + FA + C}{T}$$

Zero-shot pre-trained model evaluation

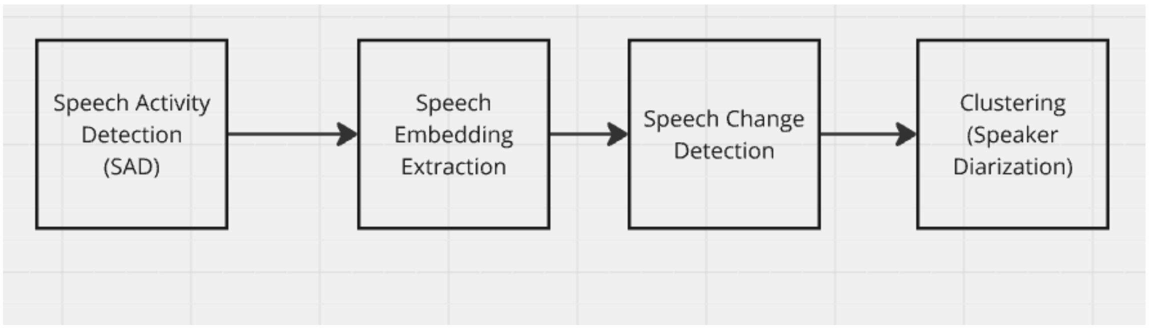
Using audio segmentation and clustering:
Steps:
Step 1: Perform segmentation (split on silence)
Step 2: Extract features (MFCCs) for each segment
Step 3: Apply clustering (KMeans) to the MFCC features
Step 4: Classify clusters to speakers and evaluate DER

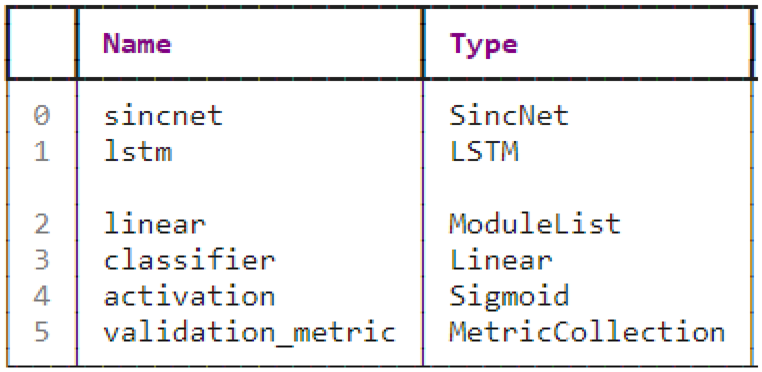


Diarization Error Rate (DER): 0.53 on day1_consultation01

This is not that much satisfactory compared to pyannote model

Using pretrained pyannote library:
Pyannote has capabilities for speaker diarization. I used the pretrained model on each file to obtain transcripts of speaker-start-end data. Again I performed fragmentation to remove overlaps.
Pyannote architecture:


Speech Activity Detection (SAD)
Speech Embedding Extraction
Speech Change Detection
Clustering (Speaker Diarization)



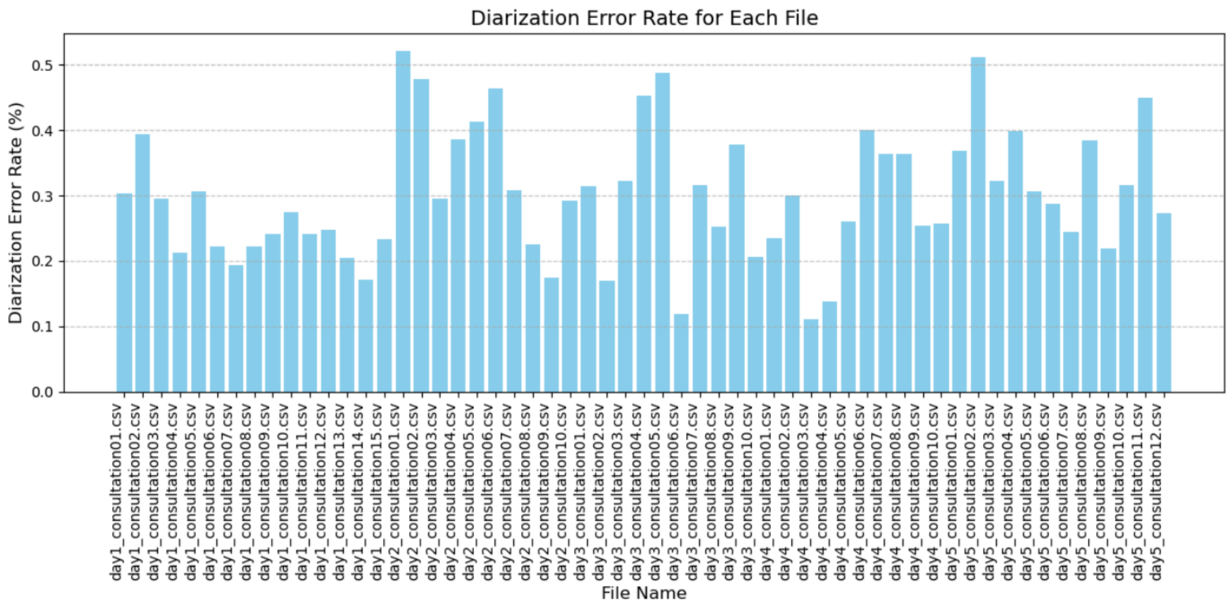
	Name	Type
0	sincnet	SincNet
1	lstm	LSTM
2	linear	ModuleList
3	classifier	Linear
4	activation	Sigmoid
5	validation_metric	MetricCollection

SAD: SincNet (feature extraction) and LSTM (temporal modeling) for detecting speech frames.
Speaker Embedding Extraction: LSTM (temporal modeling) and Linear layers (for embedding creation).
SCD: Classifier (speaker change detection) and Sigmoid (activation for classification).
Clustering: Linear layer (to refine embeddings or assist with clustering).

Summary of Layer Functions:

1. **SAD**: SincNet (feature extraction) and LSTM (temporal modeling) for detecting speech frames.
2. **Speaker Embedding Extraction**: LSTM (temporal modeling) and Linear layers (for embedding creation).
3. **SCD**: Classifier (speaker change detection) and Sigmoid (activation for classification).
4. **Clustering**: Linear layer (to refine embeddings or assist with clustering).

Results:
On complete dataset:

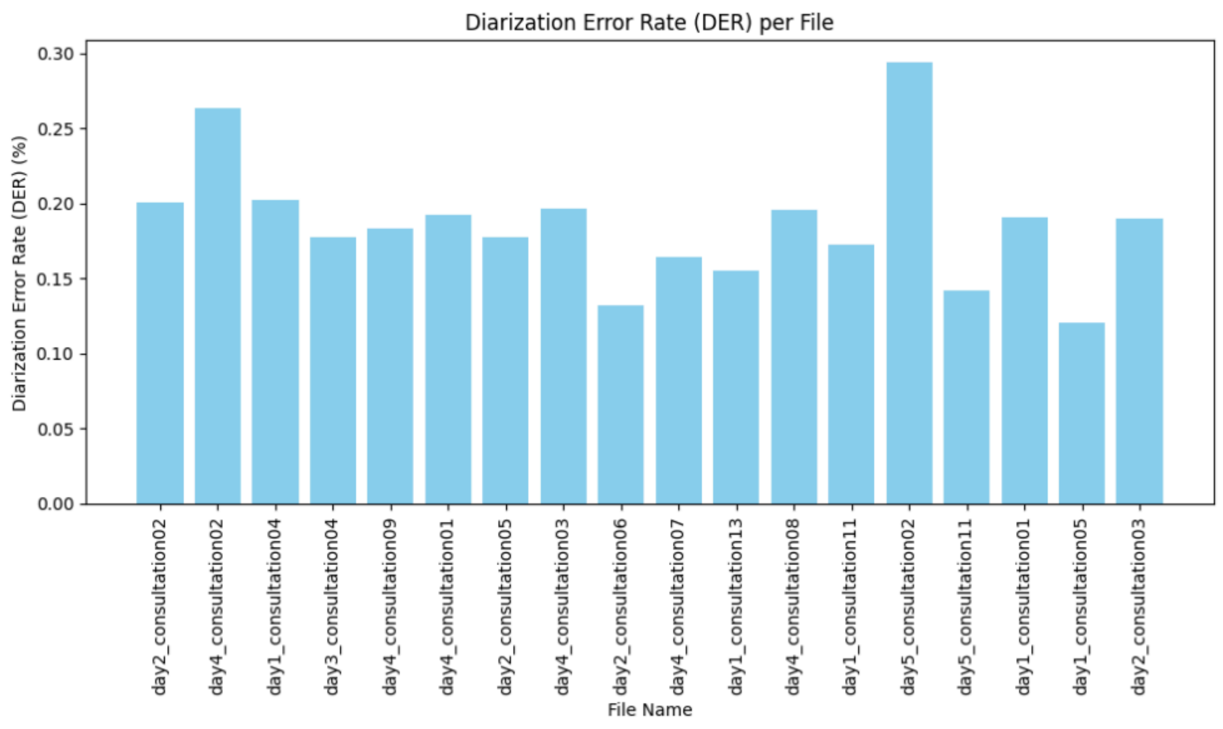


Finetuning pyannote on primock57 dataset:

```
When a sufficiently large training set of labeled conversations is
available, fine-tuning the internal speaker segmentation model may lead to
significant performance boost.
```

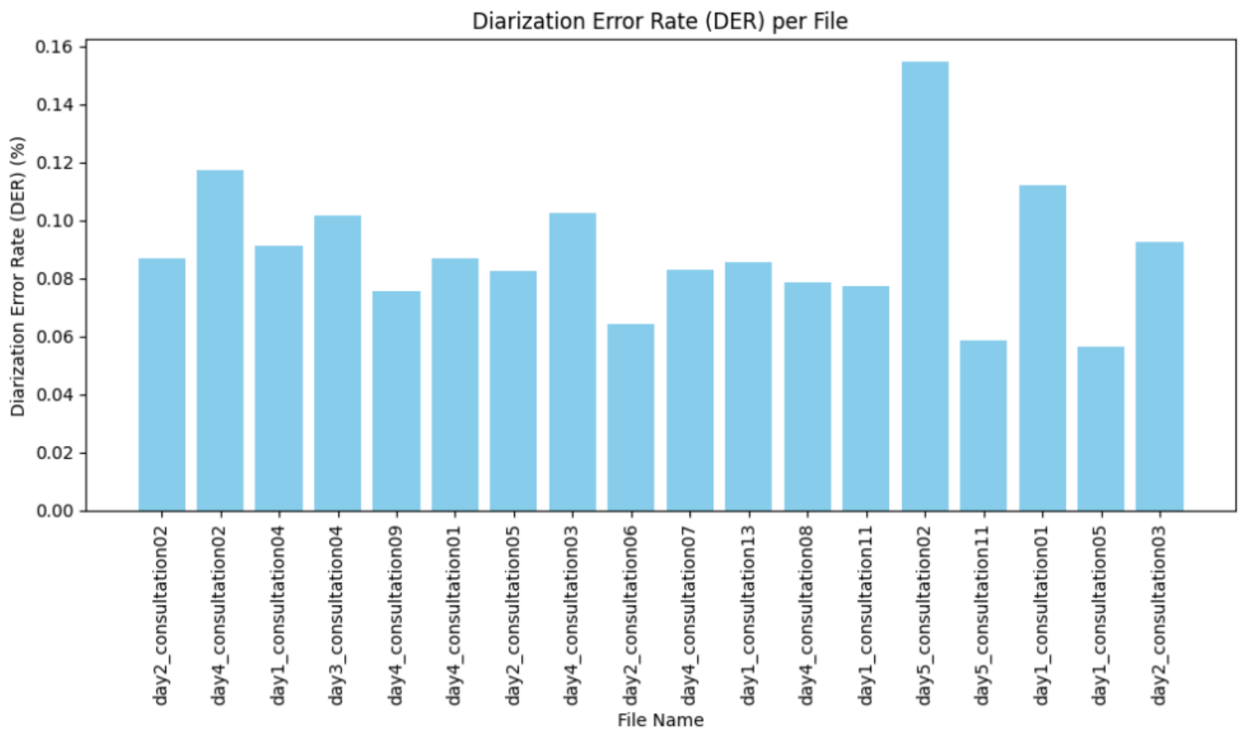
Training: 60% development: 10% testing: 30%

Result of pretrained model before finetuning on testing set:



Minimum DER: 12.1%
Maximum DER: 29.4%
Mean DER: 18.2%

Result after finetuning on primock57:



Maximum DER: 15.5%
Minimum DER: 5.6%
Mean DER: 8.6%

Comparison of different architectures used:
Segmentation+clustering is a pretty straight forward approach without any scope for finetuning, the results are also mid. But pyannote pretrained model gave much better results, even though training and testing times are large, it gave very good results compared to the naive method, even though the architecture is pretty complex.

Conclusion:
DER improved from 18.2% to 8.2% on testing set after finetuning on primock57 training data for pyannote model, which is far better than segmentation+clustering which gave 53%