I started first setting up the offline speaker diarization using pyannote-audio. Which I used as a ground truth to compare to the streaming setting. Since it is shown to perform better.

Post this I set up the streaming speaker diarization using diart [4] library directly. I used the default settings and did not play with different models and latency values to understand their effect and leave for future work.

What is speaker diarization? [1, 2, 3]

- In speech processing, diarization is a process of automatically partitioning an audio recording into segments that correspond to different speakers. This is done by using various techniques to distinguish and cluster segments of an audio signal according to the speaker's identity.
    - Diarization implies finding speaker boundaries and grouping segments that belong to the same speaker and determining the number of distinct speakers.

How to do it in an offline setting? [1]

1. Speech activity detection.
2. Segment the input speech.
    a. Local speaker information.
3. Extract the segment embeddings.
    a. Speaker rich embeddings.
4. Clustering.
    a. Global speaker information

How to do it in a streaming setting? [4]

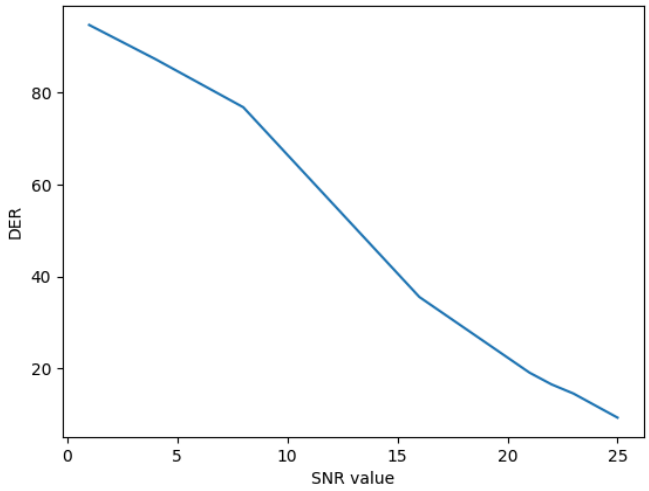
1. Incremental Speech activity detection.
2. Incremental Segment the input speech.
    a. Local speaker information.
3. Incremental Extract the segment embeddings.
    a. Speaker rich embeddings.
4. Incremental Clustering.
    a. Global speaker information

Internal testing

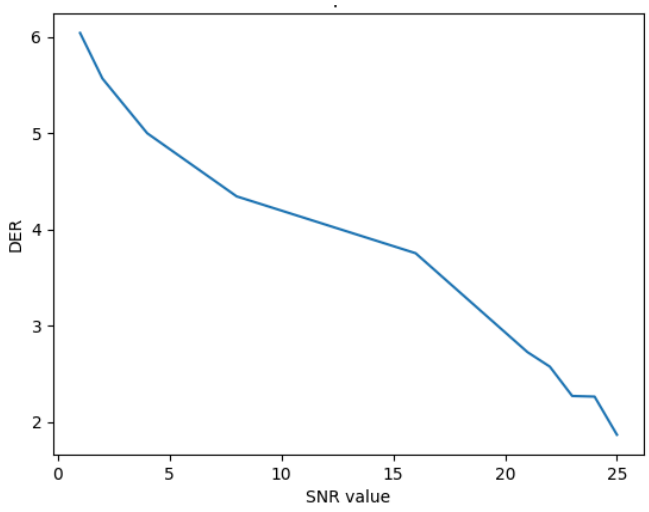
[Offline setting]

We test how robust the pyannote-audio pipeline is to the diarization error rate (DER) for two cases. Assuming the original speech sample as ground truth, we either add speech or noise with a SNR value and calculate the DER between these two. We find that it is **very-Very sensitive** to speech compared to noise.

1. Robustness to background speech.



2. Robustness to background noise.



[Streaming setting]

For segmentation. The stream is reformated to 5s duration in which a maximum 4 speakers can be detected. The buffer for incremental cluster is updated after a 500 ms shift. The lowest latency could be 500ms. or left pad the audio with 500ms of silence, if the latency is critical from the start. In the original paper, diart [4] is shown to be robust to low latencies. **The latency of the pipeline used in this work is 0.5s.**

1. Ground truth is the same as used in the offline setting.
2. Case1: Offline vs streaming setting. We achieve a DER of **12.5.** Showing a degradation.
3. Case 2: In the case of overlapped speech in a streaming setting. It was even less robust compared to offline settings.
4. Case 2: In the case of overlapped noise in a streaming setting. Similarly, it was less robust compared to offline settings.

Future work

1. Comparing different models in the diart library for different components.
2. Study the effect of latency.

References

1. https://www.gladia.io/blog/gladia-speech-to-text-api-speaker-diarization
2. https://umotion.univ-lemans.fr/video/9513-speech-segmentation-and-speaker-diarization/
3. https://www.assemblyai.com/blog/top-speaker-diarization-libraries-and-apis/
4. https://github.com/juanmc2005/diart/tree/main?tab=readme-ov-file