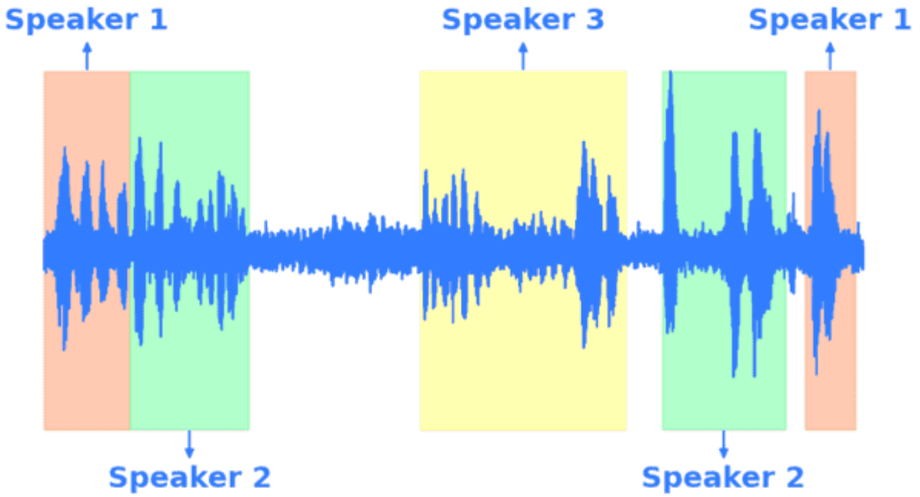
EE698r Project

Speaker Diarization



Made by:-
Krishiv Geriani(220545)
Vishap Raj(221208)

PROBLEM STATEMENT

The goal of this diarization project is to develop a system that can accurately segment and label the speakers in an audio recording, identifying "who spoke when and what" This is commonly referred to as speaker diarization. The system should be able to distinguish multiple speakers in a conversation, track each speaker's activity over time, and output the timing and identities of each speaker.

PROJECT OBJECTIVE

Our Objective was to perform Speaker Dlarization on both **real-time recorded** and **pre-recorded audio.**

Accurately determine:

- Number of speakers:- No. of speakers involved in a particular audio
- Speaker segments:-Time frame at which a particular speaker spoke.
- Confidence and uncertainty levels:- Ouputs Confidence and aleotoric uncertainty.
- Detected language

TOOLS AND LIBRARIES USED

- AssemblyAI: Transcription and utterance alignment
- PyAnnote.audio: Speaker embedding extraction
- KMeans Clustering: Speaker classification
- Softmax & Entropy: Confidence and uncertainty estimation
- Langdetect: Language detection
- Pyaudio: Real-time audio recording
- Others: Scikit-learn, Torchaudio, Torch

PRE-TRAINED MODEL

Model Name: pyannote/embedding
Publisher: pyannote-audio
Hosted on: Hugging Face Model Hub
Function: Extracts fixed-length speaker embeddings from audio segments
Use case: These embeddings can be aligned with the speaker utterances obtained using AssemblyAI, The aligned embeddings can then be clustered (using KMeans & Silhouette score) to assign speaker labels to different parts of an audio file.
Characteristics:-

- Trained on **VoxCeleb1/2 datasets** (large speaker identification datasets).
- Embedding size: 512-dimensional vectors (suitable for clustering and similarity comparisons).
- Used widely in speaker diarization, verification, and recognition tasks.

Model Name: AssemblyAI

Function:

Used to transcribe recorded audio into text and extract utterances with precise timestamps.

Use Case:

Helped convert audio conversations into structured utterance segments. However, due to its limited speaker recognition (rarely more than 2), we relied only on the utterances and implemented our own logic for speaker identification.

Characteristics:

- State-of-the-art speech-to-text accuracy
- Provides utterance-level segmentation with timestamps
- Built-in speaker diarization is limited
- Easy API integration and scalable

APPROACH OVERVIEW

Step 1: Record or Select Audio

- Two Modes:
  - Live Recording: Capture microphone input in real time using pyaudio.
  - Pre-recorded Audio: Load existing .mp3 or .wav files.
- Purpose: Prepares audio input for transcription and speaker analysis.

Step 2: Transcribe using AssemblyAI

- Use AssemblyAI's Speech-to-Text API with speaker_labels=False.
- Returns:
  - Transcribed text with time-aligned utterances.
  - Initial speaker segmentation (used for extracting audio segments).
- Provides accurate and language-agnostic transcription.

Step 3: Extract Speaker Embeddings using PyAnnote

- Load pre-trained pyannote/embedding model.
- For each utterance:
  - Extract corresponding audio segment from the waveform.
  - Pass it through the model to get a 512-dimensional speaker embedding.
  - Embeddings represent who is speaking, not what is said.

Step 4: Cluster using KMeans (Auto-select Optimal K)

- Stack all embeddings into a matrix.
- Use Silhouette Score to determine the optimal number of speakers (K).
- Perform KMeans clustering to group similar-sounding speakers.
- Labels (0, 1, 2...) are assigned to each speaker.

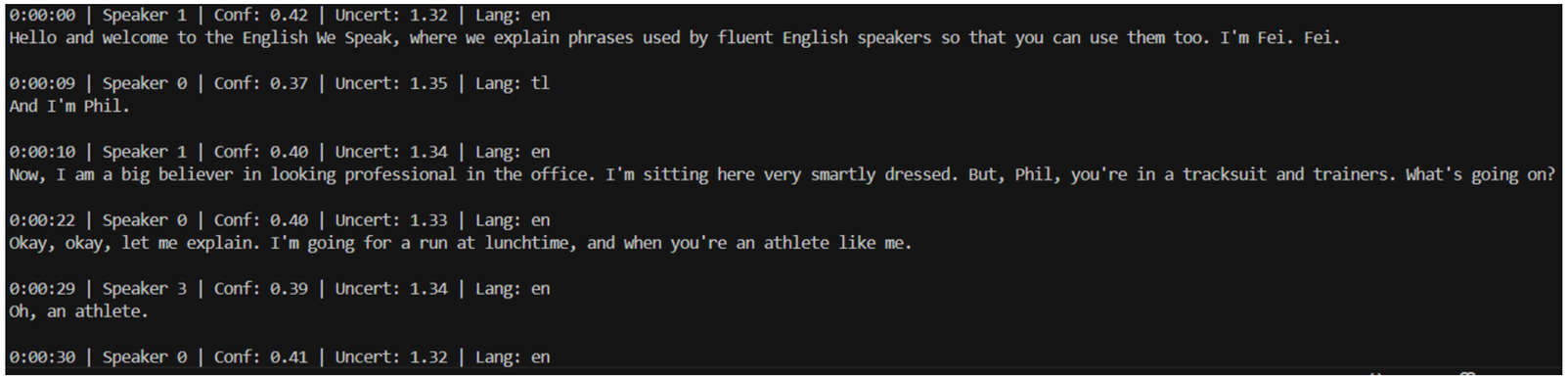
Step 5: Compute Speaker Probabilities, Confidence, Language

- For each utterance:
  - Compute cosine similarity with cluster centers.
  - Use softmax to convert to probabilities.
  - Measure:
    - Confidence: Highest probability.
    - Uncertainty: Entropy of probability distribution.

Step 6: Display Diarization Results

- Show a human-readable format:
  - Start time
  - Speaker label (e.g., Speaker 0)
  - Transcribed text
  - Confidence and uncertainty
  - Detected language

FINAL RESULTS (EXAMPLE OUTPUT)



INSIGHTS & LEARNINGS

Insight 1: Embedding quality directly affects clustering accuracy.
Insight 2: Silhouette score effectively helps in selecting k.
Insight 3: Language detection on short utterances can be noisy.
New Additions: Real-time mode, language detection, uncertainty score.

CHALLENGES FACED

- Short utterances causing unreliable embeddings.
- Memory bottlenecks on long audio using GPU.
- Real-time recording interruptions.

THANK YOU