|  |  |  |
| --- | --- | --- |
| 20EC3190 |  | BATCH ID -16 |

A Project Report

On

## “MULTI-SPEAKER AUDIO TRANSCRIPTION AND

## DIARIZATION SYSTEM”

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in

**Electronics and Communication Engineering** by

**D.Pavan Shankar -RO201086**

**A.Santhi -RO200106**

**A.YaswanthKumar -RO200520**

**B.Shalem Bakth Singh -RO200822**

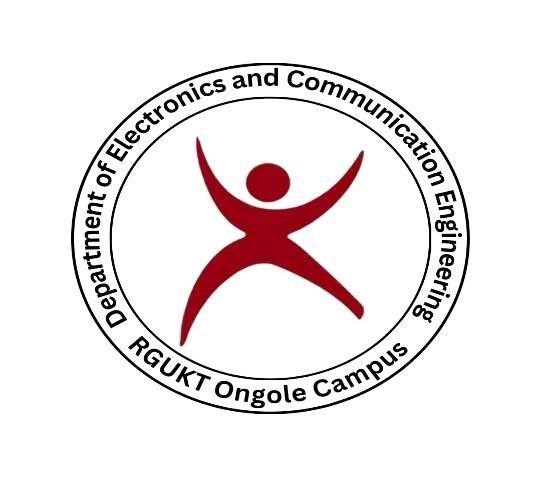
**M.Pydi Raju -RO200595**

**Under the Guidance of**

**Mrs. Nasreen**

Assistant Professor

Department of Electronics and Communication Engineering



**Department of Electronics and Communication Engineering**

### RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

**ANDHRA PRADESH, INDIA**

ONGOLE CAMPUS

Kurnool Road, Ongole, Prakasham District Andhra Pradesh - 523225

(AY 2025 – 2026)

**i**

**Approval Sheet**

This report entitled “MULTI-SPEAKER AUDIO TRANSCRIPTION AND DIARIZATION SYSTEM” by D.Pavan Shankar-(RO201086), A.Santhi - (RO200106), A.Yaswanth Kumar - (RO200520), B.Shalem Bakth Singh - (RO200822) and M.Pydi Raju - (RO200595) is approved for the degree of Bachelor of Technology Electronics and Communication Engineering

**Examiner(s):**

**Name of Examiner -1 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Designation of Examiner -1 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature of Examiner -1 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Name of Examiner -2 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Designation of Examiner -2 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature of Examiner -2 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Supervisor:**

**Name of Supervisor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Designation of Supervisor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature of Supervisor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date of Submission: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Place of Submission:\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mrs. Padmavathi**

Head of Department

Electronics and Communication Engineering

**Stamp of the Department**

**ii**

# Candidate’s Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission had not been taken when needed.

**Name of the Candidate with ID Number Signature of the Candidate**

**D.Pavan Shankar -(RO201086) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**A.Santhi - (RO200106) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**A.Yaswanth Kumar -(RO200520) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**B.Shalem Bakth Singh -(RO200822) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**M.Pydi Raju - (RO200595) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date***:*

**iii**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING ©

### RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

ONGOLE CAMPUS



# CERTIFICATE

This is to certify that the project entitled **“Multi-Speaker Audio Transcription and Diarization System”** being submitted **D.Pavan Shankar** bearing ID Number **O201086** and **A.Santhi** bearing ID Number **O200106** and **A.Yaswanth Kumar** bearing ID Number **O200520** and **B.Shalem Bakth Singh** bearing ID Number **O200822** and **M.Pydi Raju** bearing ID Number **O200595** in partial fulfillment of the requirements for the award of the Bachelor of Technology in Electronics and Communication

Engineering is a bonafide work carried by them under my supervision and guidance

**Project internal Guide Head of Department**

**Mrs.Nasreen Mrs. N.padmavathi**

Assistant Professor Assistant Professor

Department of ECE Department of ECE

**IV**

# ACKNOWLEDGEMENT

It is our privilege to express a profound sense of respect, gratitude and indebtedness to our guide **Mrs.Nasreen, Assistant Professor,** Department. of Electronics and Communication Engineering, Rajiv Gandhi University of Knowledge Technologies – Ongole Campus for her guidance, technical and moral support and for her efforts in successful completion of our project.

We express our sincere gratitude to **Mrs N. Padmavathi, Assistant Professor and Head of Department (i/c) Electronics and Communication Engineering,** Rajiv Gandhi University of Knowledge Technologies – Ongole Campus, for her suggestions and co-operation for the successful completion of the work.

We extend our sincere thanks to **Mr. M. Rupas Kumar, Dean Academics,** Research and Development, Rajiv Gandhi University of Knowledge Technologies – Ongole Campus, for his constant help and support in our academic activities.

We extend our sincere thanks **Prof. Dr. Bhaskar Patel , Director,** Rajiv Gandhi University of Knowledge Technologies – Ongole Campus for his overall vision and guidance.

**D.Pavan Shankar -RO201086**

**A.Santhi -RO200106**

**A.Yaswanth Kumar -RO200520**

**B.Shalem Bakth Singh -RO200822**

**M.Pydi Raju -RO200595**

**V**

# DECLARATION

We hereby declare that the project work entitled **“MULTI-SPEAKER AUDIO TRANSCRIPTION AND DIARIZATION SYSTEM"** submitted to the **Rajiv Gandhi University Of Knowledge Technologies Ongole** Campus in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech)** in Electronics and Communication Engineering is a record of an original work done by us under the guidance of **Nasreen, Assistant Professor, Department of Electronics and Communication Engineering** and this project work have not been submitted to any university for the award of any other degree or diploma. The copyright to this project belongs to us along with **Rajiv Gandhi University Of Knowledge Technologies Ongole Campus ©**

**D.Pavan Shankar -RO201086**

**A.Santhi -RO200106**

**A.Yaswanth Kumar -RO200520**

**B.Shalem Bakth Singh -RO200822**

**M.Pydi Raju -RO200595**

**VI**

# ABSTRACT

In recent years, the demand for accurate transcription and speaker diarization systems has grown significantly with the rise of virtual meetings, online interviews, and podcasts. A major challenge in such environments is to not only transcribe audio content but also identify and distinguish between multiple speakers. This research focuses on developing an efficient system that combines speech recognition with speaker identification using pre-trained deep learning models. The system leverages OpenAI's Whisper model for transcription and SpeechBrain's ECAPA-TDNN for speaker embedding. The proposed solution is designed to handle audio files with overlapping conversations and varying speaker tones. The goal is to enhance the accuracy of segment-wise transcription while tagging each segment with the correct speaker label.

The pipeline begins with audio preprocessing and segmentation into smaller uniform chunks for consistent feature extraction. Each segment is passed through a speaker embedding model to obtain high-dimensional vector representations of the speaker's voice characteristics. Reference audios from known speakers are used to generate embedding templates, which help in mapping the clustered embeddings to actual speaker identities. Clustering algorithms like Agglomerative Clustering are applied to group segments based on vocal similarity. The system then assigns names to these clusters by comparing them with reference speaker embeddings using cosine similarity. This allows for speaker diarization even in cases where speakers may have similar pitch or overlapping speech.

Finally, the transcribed output is organized in both tabular and graphical formats for easy interpretation and analysis. A speaker timeline visualization shows who spoke when, while a timestamped table provides a detailed breakdown of speaker turns. The system has been tested on audio files with up to four speakers, achieving promising results for known and unknown speaker scenarios. This solution can be beneficial in various real-world applications like courtroom transcriptions, academic discussions, and call center analytics. Future work may involve improving the system's performance in noisy environments and optimizing real-time diarization. The project demonstrates a practical integration of advanced speech processing tools to address multi-speaker audio transcription challenges.

**VII**

## TABLE OF CONTENTS

[Approval Sheet ii](#_Toc195024334)

[Candidate’s Declaration iii](#_Toc195024335)

[CERTIFICATE iv](#_Toc195024336)

[ACKNOWLEDGEMENT v](#_Toc195024337)

DECLARATION vi

[ABSTRACT vii](#_Toc195024339)

1. **Introduction............................................................................................................................. 1** 
   1. Project Overview…………………………….......................................................................................2
   2. Objective of the Code…………….................................................................................................2
   3. Tools & Libraries Used……………………………………..……………………………………………….…….……….….4
2. **Theoretical Concepts................................................................................................................. 7**
   1. Speech Signal Representation…………...................................................................................... 8
   2. Audio Transcription Using Whisper………................................................................................ 8
   3. Speaker Diarization………………….............................................................................................. 9
   4. Speaker Embedding and Voiceprint......................................................................................... 9
   5. Reference speaker Matching…………........................................................................................10
   6. Agglomerative Clustering…………..............................................................................................11
   7. Cosine Similarity in Speaker Identification…………....................................................................11
   8. Visualization Techniques…………...............................................................................................12

**3 . Code Implementation………………………………………………………..…………………………………………………13**

3.1. Audio upload and configuration………………….………………………………………………………………………14  
3.2. Audio Transcription using Whisper ………….…….……………………………………………………………………15  
3.3. Embedding Generation ………………………………….…………………………………………………………………..17

3.4 Speaker Reference Embeddings …………………….…………………………...…………..………..…………………19

3.5 Speaker Clustering and Identification …………………………………………………………………………………..20

* 1. Transcript Generation ……….………………………………………………..………….……………………………………23
  2. Visualization………………………………………………………………………………………………………………………..25

**4 Results And Analysis………………………………………………………………………………………..…………………..…29**

4.1 Input Mixed Audio File……….…………...…………..……………………………..….…………………..……………….30

4.2 Segments of Mixed Audio file With Timestamps……..……………………………..….…………………..…….30

4.3 Embedding For Segments…….…………...…………..……………………………..….…………………………..…….31

4.4 Uploading The Reference Files…………..……………………………..….…………………..…………………………31

4.5 Dialogue Breakdown By Speaker And Time.…………..……………………………..….…………………..…….32

* 1. Speaker Clustering And Visualization………...…………..……………………………..….…………………..…..33

4.7 Speaker Diarization Timeline………..……………………………..….…………………..…………………………….33

**5. Applications And Future Scope…………………….………………………………………..……………………..………..…34**

* 1. Real-World Applications……….…………...……………………………………..….…………………..……………….35

5.2 Research And Development Opportunities.…………...……………………………………..….………………..35

5.3 Limitations And Future Scope………...……………………………………..….…………………..……………………36

**6. References…………………….………………………………………..……………………..………..………………………………...37**

**CHAPTER 1**

## INTRODUCTION

**1.1 Overview**

**Introduction to the Problem**

Speech data often involves multiple speakers talking in a single audio stream, making it difficult to determine who said what and when. Traditional transcription tools may provide text but fail to distinguish between different speakers. This project addresses the challenge by combining **speech-to-text** and **speaker diarization**.

**Key Technologies Used**

* **Whisper** by OpenAI is used for accurate audio transcription.
* **SpeechBrain's ECAPA-TDNN model** extracts speaker-specific features from speech.
* **Agglomerative Clustering** classifies segments into speaker groups based on similarity.
* **Matplotlib** is used to create a visual timeline for better analysis of who spoke when.

**Scope of the Project**

This project focuses on audio with **Four speakers**, but the architecture is scalable to more. It supports English language transcription and is designed to run in environments like Google Colab, making it accessible and replicable.

**1.2 Objective of the Code**

**Primary Goal**

The primary objective of this project is to **analyze a mixed audio file** containing speech from multiple speakers and automatically perform two critical tasks:

1. **Transcribe** the spoken content into text.
2. **Identify and label** each speaker's speech segments correctly (speaker diarization).

This helps in creating readable, structured dialogues from raw audio recordings without manual intervention, saving time and improving efficiency in documentation and analysis.

**Dual Functionality Integration**

The project combines two advanced AI models:

* **OpenAI Whisper**: Performs automatic speech recognition (ASR) to convert audio into text.
* **SpeechBrain ECAPA-TDNN**: A pretrained speaker embedding model used to extract speaker-specific voice features, which help in distinguishing speakers.

These components are seamlessly integrated to ensure the speech is not only transcribed but also attributed to the correct speaker, enabling the creation of dialogue-style transcripts.

**Speaker Identity Mapping**

Using **cosine similarity** between speaker embeddings, each detected speaker segment is matched with the closest reference, ensuring meaningful speaker names (e.g., "Alice", "Bob") instead of generic labels.

This technique is especially useful in interviews, podcasts, or meeting recordings, where speaker identity adds crucial context to the conversation.

**Visual Timeline Generation**

A unique visual feature of this system is its ability to **generate a timeline plot**. This timeline represents each speaker's activity using color-coded horizontal bars over time, allowing users to:

* Visualize speaker shifts
* Identify overlaps or pauses in conversation
* Analyze speaking patterns

This adds clarity and insight to the transcript and is valuable for both technical and non-technical users.

The project described here exemplifies this evolution by utilizing efficient algorithms and accessible tools to achieve precise separation of audio sources. By focusing on practical implementations, it addresses key challenges in audio separation, such as handling noisy environments and overlapping frequencies. This historical progression highlights how

**Comprehensive Output**

The code outputs:

* A text file (transcript.txt) containing the entire conversation with timestamps and speaker names.
* A timeline plot that visually displays the distribution and switching of speakers over the duration of the audio.

These outputs are useful for **f**urther research, case studies, interviews, transcriptions, or accessibility improvements.

**Real-World Applicability**

This project has high relevance in various real-world applications such as:

* Legal proceedings and court recordings
* Journalistic interviews and podcast processing
* Call center analytics
* Academic research on conversations or debates

It significantly reduces manual effort, enhances productivity, and opens opportunities for data analysis and machine learning training datasets.

**1.3 Tools & Libraries Used**

This project integrates a combination of powerful Python libraries, AI models, and frameworks to achieve high-quality transcription and speaker diarization. Below are the key tools and libraries used in this system: like Demucs (Deep Extraction of Music Sources), the system aims to outperform traditional signal processing methods.

**Whisper by OpenAI**

* **Purpose**: Automatic Speech Recognition (ASR)
* **Description**: Whisper is a state-of-the-art model developed by OpenAI capable of transcribing speech in multiple languages with high accuracy.
* **Usage in Project**: Converts mixed speaker audio into text and segments it based on detected utterances.

**SpeechBrain**

* **Purpose**: Speaker Embedding and Recognition
* **Description**: SpeechBrain is an open-source toolkit for speech processing. It includes models like ECAPA-TDNN for extracting unique voice features (embeddings) from audio clips.
* **Usage in Project**: Generates speaker embeddings to identify and cluster speaker voices accurately.

**Scikit-learn**

* **Purpose**: Clustering and Classification
* **Description**: A robust machine learning library used here for applying **Agglomerative Clustering** on speaker embeddings to group speech segments by speaker identity.
* **Usage in Project**: Clusters all speech segments into groups based on voice similarity.

**NumPy**

* **Purpose**: Numerical Computation
* **Description**: Core library for handling arrays and numerical operations.
* **Usage in Project**: Manages the speaker embedding arrays and performs statistical operations.

**Matplotlib**

* **Purpose**: Data Visualization
* **Description**: A plotting library used to create static, interactive, and animated visualizations.
* **Usage in Project**: Plots the Speaker Diarization Timeline, showing when each speaker was active.

**Google Colab / Jupyter Notebook**

* **Purpose**: Interactive Development Environment
* **Description**: Cloud-based Python notebooks ideal for rapid prototyping, visualization, and GPU-based inference.
* **Usage in Project**: Hosts the code execution and allows audio file uploads for analysis.

**Other Supporting Libraries**

* os, subprocess, wave, contextlib: Handle file operations, audio conversions, and duration calculations.
* datetime: Used to format time in HH:MM:SS for better readability in transcripts.
* IPython.display: Helps in interacting with Google Colab interface (like file upload dialogs).

## Chapter-2

## THEROTICAL CONCEPTS

**2.1 Speech Signal Representation**

Speech signals are a form of acoustic energy generated by the human vocal system and captured through microphones or recording devices. These signals are inherently non-stationary, meaning their frequency and amplitude characteristics vary over time. To effectively analyze and process speech, it is essential to represent it in a digital format.

The digitization process involves sampling (capturing audio at discrete intervals) and quantization (assigning numerical values to signal amplitudes). The most commonly used sampling rate for speech processing is 16,000 Hz, which balances quality and computational efficiency.

Once digitized, the audio signal is typically divided into frames of 20–40 milliseconds. Each frame is considered approximately stationary and can be analyzed using methods like the Short-Time Fourier Transform (STFT) to produce a spectrogram, which shows how the signal's frequency content changes over time.

Additionally, the Mel-frequency cepstral coefficients (MFCCs) and log-Mel spectrograms are commonly extracted features that imitate human auditory perception. These are frequently used as input to machine learning models for tasks like speech recognition and speaker identification.

**2.2 Audio Transcription Using Whisper**

Whisper is a powerful open-source Automatic Speech Recognition (ASR) system developed by OpenAI. It is based on a **transformer architecture** and trained on a large corpus of multilingual and multitask supervised data. Whisper's strength lies in its ability to perform transcription on a wide range of accents, noisy recordings, and real-world audio.

The transcription process involves several stages:

* **Preprocessing**: Converts raw audio into a **log-Mel spectrogram**.
* **Segmentation**: Divides audio into chunks for manageable processing.
* **Language Identification**: Determines the spoken language in each segment.
* **Decoding**: Translates spectrogram features into readable text using a decoder model trained on audio-text pairs.

Whisper can perform **zero-shot transcription**, meaning it can transcribe languages or accents it has never explicitly seen during training. It supports **timestamped transcription**, which is beneficial for aligning text with audio for applications such as subtitling or diarization.

**2.3 Speaker Diarization**

Speaker diarization is often referred to as the process of answering the question, *“Who spoke when?”* It involves dividing an audio stream into segments based on speaker identity. Unlike speech recognition, which transcribes what was said, diarization focuses on labeling and separating the voices.

A typical diarization system involves the following steps:

1. **Voice Activity Detection (VAD):** Identifies portions of audio that contain speech.
2. **Segmentation: Splits** the audio into smaller chunks where a single speaker is active.
3. **Embedding Extraction:** Converts each segment into a fixed-dimensional vector (speaker embedding) that represents the speaker’s vocal characteristics.
4. **Clustering:** Groups embeddings that likely belong to the same speaker.

Diarization is useful in applications like meeting analytics, podcast indexing, call center monitoring, and legal transcription. It helps improve the readability and usefulness of transcripts by distinguishing between different speakers.

**2.4 Speaker Embedding and Voiceprint**

Speaker embeddings are numerical representations of a person’s voice. These fixed-length vectors capture unique vocal features such as pitch, tone, speaking rate, and accent. The embedding acts like a **voice fingerprint** or **voiceprint**, which can be compared across different speech segments.

One popular model for generating speaker embeddings is **ECAPA-TDNN (Emphasized Channel Attention, Propagation and Aggregation Time-Delay Neural Network)**. It enhances speaker recognition performance by integrating:

* + - **Squeeze-and-excitation blocks** for attention.
    - **Residual connections** for deeper modeling.
    - **Multi-layer feature aggregation** for robustness.

Speaker embeddings are commonly used in:

* + - **Speaker verification**: Matching a voice to a known identity.
    - **Speaker diarization**: Grouping segments from the same speaker.
    - **Speaker adaptation**: Personalizing models to specific users.

### ****2.5 Reference Speaker Matching****

In diarization tasks, labels like “Speaker 1” and “Speaker 2” are arbitrary and not tied to real identities. To resolve this, **reference speaker matching** maps these anonymous clusters to actual people using known audio samples.

The process involves:

* Uploading short audio clips from each known speaker.
* Extracting speaker embeddings from the references.
* Averaging embeddings from each diarized cluster.
* Comparing cluster averages to reference embeddings using **cosine similarity**.
* Assigning the most similar reference label to the cluster.

This step is essential for use cases where personalization or tracking speaker behavior across recordings is important—such as educational lectures, interview analysis, or customer support calls.

### ****2.6 Agglomerative Clustering****

Agglomerative Clustering is a hierarchical clustering algorithm used to group similar data points—here, speaker embeddings. It starts with each data point as its own cluster and merges the two closest clusters at each step until the desired number of clusters (speakers) is reached.

The clustering relies on a distance metric (usually **cosine** or **Euclidean**) and a **linkage criterion** (such as average linkage). The choice of parameters can significantly impact the performance of the clustering step.

In speaker diarization, agglomerative clustering is often used because:

* It does not require pre-training.
* It allows flexibility in determining the number of speakers.
* It’s robust for short recordings with fewer speakers.

.

**2.7 Cosine Similarity in Speaker Identification**

Cosine similarity is a measure of similarity between two non-zero vectors by calculating the cosine of the angle between them. It is particularly effective in high-dimensional spaces like speaker embeddings.

The formula is:

**Cosine Similarity(A, B) = (A · B) / (||A|| × ||B||)**

In this project, cosine similarity is used for:

* Comparing diarized embeddings to reference embeddings.
* Grouping similar speech segments.
* Evaluating how close two audio samples are in voice characteristics.

A cosine similarity score near **1.0** indicates that the voices are very similar, while a score closer to **0** means they are dissimilar. This metric plays a central role in both clustering and reference matching stages.

**2.8 Visualization Techniques**

Visualizing diarization results provides intuitive understanding and helps validate model output. The most common form of visualization is a **speaker timeline**, which displays:

* Time progression on the X-axis.
* Color-coded bars indicating segments spoken by different speakers.
* A legend mapping colors to speaker identities.

We use tools such as matplotlib or specialized packages like pyannote-audio to render these timelines. Other useful visual tools include:

* **Waveforms** overlaid with speaker labels.
* **Spectrograms** showing frequency content per speaker.
* **Embedding projections** (e.g., t-SNE or PCA) to show cluster separability.

These visualizations assist in:

* Diagnosing diarization errors.
* Understanding conversation dynamics.

## Chapter 3

**CODE IMPLEMENTATION**

**3.1 Audio upload and configuration**

This section sets up the environment required to perform multi-speaker audio transcription and diarization. It includes installing libraries, uploading the mixed audio, and configuring model parameters.

**Libraries Used :**

1. **openai-whisper**
   * An automatic speech recognition (ASR) system developed by OpenAI.
   * It supports multiple languages and model sizes (tiny, base, small, medium, large).
   * In this project, it is used to transcribe audio into text segments along with timestamps.
2. **pyannote.audio**
   * A deep learning toolkit for speaker diarization based on PyTorch.
   * It includes pre-trained pipelines for tasks like speaker embedding and segmentation.
   * In this project, it's used to extract speaker-specific embeddings from audio and compare voice similarities.

#### ****Uploading the Mixed Audio File :****

The user is prompted to upload the mixed audio file containing multiple speakers:

Code :

from google.colab import files

uploaded = files.upload()

path = next(iter(uploaded))

**Number of Speakers**

You can define how many distinct speakers are expected in the audio. This number is later used in clustering to group voice segments by speaker.

Code:

num\_speakers = 4 # This example assumes 4 speakers

**Whisper Model Configuration**

You set the language and model size (e.g., small, medium, large). Whisper has two modes:

* Multilingual
* English-only (optimized for English)

The model name is dynamically adjusted depending on the chosen configuration:

Code :

language = 'English'

model\_size = 'large'

model\_name = model\_size

if language == 'English' and model\_size != 'large':

model\_name += '.en'

This configuration step ensures all necessary tools and inputs are ready before moving to transcription and speaker analysis.Steps:

**3.2 Audio Transcription using Whisper**

This section focuses on converting spoken audio into written text using OpenAI's **Whisper** model.

**What is Whisper?**

Whisper is an automatic speech recognition (ASR) model developed by OpenAI. It is capable of transcribing speech in multiple languages and provides timestamps for each segment of speech. It also handles different accents, background noise, and even mixed conversations quite robustly.

Code :

import whisper

model = whisper.load\_model(model\_size)

result = model.transcribe(path)

segments = result["segments"]

**Explanation :**

**1. whisper.load\_model(model\_size)**

* Loads the specified Whisper model (e.g., large, base, small, etc.).
* Larger models are more accurate but require more computation.

**2. model.transcribe(path)**

* Transcribes the uploaded audio file into text.
* It returns:
  + A full transcript
  + A list of **segments**, where each contains:
    - start and end times
    - text spoken in that duration

**3.segments = result["segments"]**

* Extracts the list of speech segments with time-aligned transcripts.

### ****3.3 Embedding Generation****

Once we have the transcribed audio split into segments (via Whisper), the next step is to extract speaker embeddings. These embeddings help us to understand the "voice print" or audio characteristics of each speaker's voice, which is crucial for speaker diarization—identifying who spoke when.

**What is an Embedding?**

An embedding is a high-dimensional numerical representation of audio. In the case of speaker embeddings, it captures features unique to a speaker’s voice—like pitch, tone, speaking style, etc.

Think of it as a fingerprint of a voice segment**.**

**Code :**

from pyannote.audio import Audio

from pyannote.audio.pipelines.speaker\_verification

import PretrainedSpeakerEmbedding

from pyannote.core import Segment

import torch

import numpy as np

# Load audio interface

audio = Audio()

# Load speaker embedding model

embedding\_model = PretrainedSpeakerEmbedding(

"speechbrain/spkrec-ecapa-voxceleb",

device=torch.device("cuda" if torch.cuda.is\_available() else "cpu")

)

# Define function to get embedding for each segment

def segment\_embedding(segment):

start = segment["start"]

end = min(duration, segment["end"])

clip = Segment(start, end)

waveform, sample\_rate = audio.crop(path, clip)

waveform = waveform.mean(axis=0, keepdims=True) # Convert to mono

return embedding\_model(waveform[None])[0]

# Generate embeddings for each segment

embeddings = np.zeros((len(segments), 192)) # 192 is the embedding dimension

for i, segment in enumerate(segments):

embeddings[i] = segment\_embedding(segment)

embeddings = np.nan\_to\_num(embeddings)

**Explanation**

1. **Audio()**  
   Loads an audio interface to crop audio segments using start and end times.
2. **PretrainedSpeakerEmbedding()**  
   Loads a pre-trained model that generates a speaker's embedding using ECAPA-TDNN (a powerful model for speaker verification).
3. **segment\_embedding() function**
   * Takes a segment (from Whisper).
   * Extracts the audio for that time range.
   * Converts stereo audio to mono.
   * Passes it through the embedding model to get a 192-dimensional vector.
4. **embeddings**   
   Stores embeddings for each transcribed segment.
5. **np.nan\_to\_num()**  
   Removes NaN values that may arise due to silence or corrupt segments

**3.4 Speaker Reference Embedding**

Once we’ve generated embeddings for the audio segments, the next step is to generate reference embeddings for known speakers. These reference embeddings allow us to label which speaker is which in the diarization step — so instead of calling them *Speaker A*, *Speaker B*, etc., we can say *Alice*, *Bob*, etc.

**What is a Speaker Reference Embedding?**

A reference embedding is a precomputed speaker embedding obtained from a sample audio file of a known speaker. This embedding is used to compare against the segment embeddings (from the transcription) to identify the closest match.

**Code :**

embedding\_model = PretrainedSpeakerEmbedding(

"speechbrain/spkrec-ecapa-voxceleb",

device=torch.device("cuda" if torch.cuda.is\_available() else "cpu")

)

# Dictionary to store reference embeddings

SPEAKER\_REFERENCE\_EMBEDDINGS = {}

# Load reference audio for each speaker

for speaker\_name, reference\_file\_path in REFERENCE\_SPEAKERS.items():

waveform, sample\_rate = audio.crop(reference\_file\_path, Segment(0, 10)) # Use first 10 seconds

waveform = waveform.mean(axis=0, keepdims=True) # Convert to mono

embedding = embedding\_model(waveform[None])[0] # Generate embedding

SPEAKER\_REFERENCE\_EMBEDDINGS[speaker\_name] = embedding

**3.5 Speaker Clustering and Identification**

After generating embeddings for both audio segments and known speaker references, we now cluster the audio segments based on speaker similarity and assign names to each cluster using the reference embeddings.

* Group similar audio segments (spoken by the same person) using unsupervised clustering.
* Identify the speaker of each cluster by comparing its average embedding to the reference embeddings using cosine similarity.

### Techniques Used :

|  |  |
| --- | --- |
| **Technique** | **Purpose** |
| **Agglomerative Clustering** | Groups similar embeddings into clusters |
| **Cosine Similarity** | Matches cluster centroids to speakers |

**Required Libraries**

* sklearn.cluster.AgglomerativeClustering – For hierarchical clustering of embeddings.
* sklearn.metrics.pairwise.cosine\_similarity – To compare unknown embeddings with known speaker embeddings.

**Code :**

from sklearn.cluster import AgglomerativeClustering

from sklearn.metrics.pairwise import cosine\_similarity

# Perform clustering to group audio segments by speaker

clustering = AgglomerativeClustering(n\_clusters=num\_speakers)

labels = clustering.fit\_predict(embeddings)

# Create a mapping from cluster ID to reference speaker name

cluster\_to\_name = {}

for cluster\_id in range(num\_speakers):

# Get embeddings belonging to the current cluster

cluster\_embeddings = [embeddings[i] for i in range(len(segments)) if labels[i] == cluster\_id]

avg\_embedding = np.mean(cluster\_embeddings, axis=0)

# Compare with reference embeddings using cosine similarity

scores = {

name: cosine\_similarity([avg\_embedding], [ref\_emb])[0][0]

for name, ref\_emb in reference\_embeddings.items()

}

best\_match = max(scores, key=scores.get)

cluster\_to\_name[cluster\_id] = best\_match

# Assign speaker name to each segment

for i, segment in enumerate(segments):

segment["speaker"] = cluster\_to\_name[labels[i]]

**Explanation**

1. **Agglomerative Clustering**:
   * Groups audio segments into n clusters (equal to the number of speakers).
   * Uses Euclidean distance between embeddings by default.
2. **Average Cluster Embedding**:
   * Computes the **mean embedding** for each cluster.
3. **Cosine Similarity**:
   * Compares the average cluster embedding with each reference speaker embedding.
   * Speaker with highest similarity is assigned to that cluster.
4. **Assign Speaker Name**:
   * The final speaker name is injected into the segments[i]["speaker"] for each segment.

## 3.6 Transcript Generation

Once speaker identification and transcription are complete, we can **compile a structured transcript**. This step organizes all transcribed segments along with the **speaker names** and **timestamps**, making it easy to read or use in downstream applications.

**Objective**

To generate a clean and readable transcript that:

* Shows who said what and when.
* Can be saved as a text file, CSV, or rendered on a web interface.

**Required Libraries**

* pandas – For organizing transcript data into a table.
* datetime (optional) – To convert seconds into hh:mm:ss format.
* Built-in open() – To write transcript to a file.

Code :

import pandas as pd

from datetime import timedelta

def format\_timestamp(seconds):

return str(timedelta(seconds=int(seconds)))

# Build a transcript DataFrame

def generate\_transcript\_table(segments):

data = []

for seg in segments:

start\_time = format\_timestamp(seg["start"])

end\_time = format\_timestamp(seg["end"])

speaker = seg.get("speaker", "Unknown")

text = seg["text"]

data.append({

"Start": start\_time,

"End": end\_time,

"Speaker": speaker,

"Text": text

})

df = pd.DataFrame(data)

return df

# Generate the transcript table

transcript\_df = generate\_transcript\_table(segments)

# Display a sample

print(transcript\_df.head())

# Optionally save it

transcript\_df.to\_csv("final\_transcript.csv", index=False)

**Explanation :**

|  |  |
| --- | --- |
| **Step** | **Description** |
| format\_timestamp() | Converts raw seconds (e.g. 163.2) into readable format (00:02:43). |
| generate\_transcript\_table() | Builds a structured table with speaker, time range, and transcribed text. |
| to\_csv() | Saves the table for sharing or future analysis. |

**3.7 Visualization**

**Objective**

To visualize speaker diarization results in two intuitive ways:

1. **t-SNE Cluster Plot** – Shows how embeddings group by speaker.
2. **Speaker Timeline Plot** – Displays who spoke and when in a time-based visual format.

These visualizations help you understand:

* The distinctness of speakers in feature space.
* The temporal distribution of speaker activity.

### Libraries Used

|  |  |
| --- | --- |
| **Library** | **Purpose** |
| matplotlib | Plotting timeline and scatter plots. |
| seaborn | Styling the t-SNE cluster plot. |
| TSNE (from sklearn) | Reducing high-dimensional embeddings into 2D for visualization. |

**Code :**

**1. t-SNE Cluster Visualization**

from sklearn.manifold import TSNE

import matplotlib.pyplot as plt

import seaborn as sns

# Dimensionality reduction

tsne = TSNE(n\_components=2, perplexity=5, random\_state=42)

reduced\_embeddings = tsne.fit\_transform(embeddings)

# Prepare color mapping

unique\_speakers = list(set([seg['speaker'] for seg in segments]))

palette = sns.color\_palette("hsv", len(unique\_speakers))

color\_map = {speaker: palette[i] for i, speaker in enumerate(unique\_speakers)}

colors = [color\_map[seg['speaker']] for seg in segments]

# Plot

plt.figure(figsize=(10, 6))

for speaker in unique\_speakers:

idxs = [i for i, seg in enumerate(segments) if seg['speaker'] == speaker]

plt.scatter(reduced\_embeddings[idxs, 0], reduced\_embeddings[idxs, 1],

label=speaker, alpha=0.7, s=60)

plt.title("t-SNE Clustering of Speakers", fontsize=14)

plt.xlabel("t-SNE Dim 1")

plt.ylabel("t-SNE Dim 2")

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**2. Speaker Timeline Plot**

import matplotlib.patches as mpatches

# Assign colors

speaker\_color\_map = {name: plt.cm.get\_cmap('tab10')(i) for i, name in enumerate(unique\_speakers)}

# Plot each segment as a block

fig, ax = plt.subplots(figsize=(15, 3))

for segment in segments:

start, end = segment["start"], segment["end"]

speaker = segment["speaker"]

ax.plot([start, end], [1, 1], linewidth=8, color=speaker\_color\_map[speaker])

# Styling

ax.set\_yticks([])

ax.set\_xlim(0, duration)

ax.set\_xlabel("Time (seconds)")

ax.set\_title("Speaker Diarization Timeline")

# Add legend

legend\_patches = [mpatches.Patch(color=color, label=speaker)

for speaker, color in speaker\_color\_map.items()]

ax.legend(handles=legend\_patches, loc='upper right')

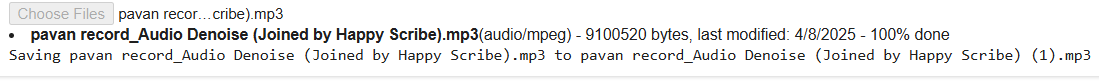
plt.tight\_layout()

plt.show()

**Chapter 4**

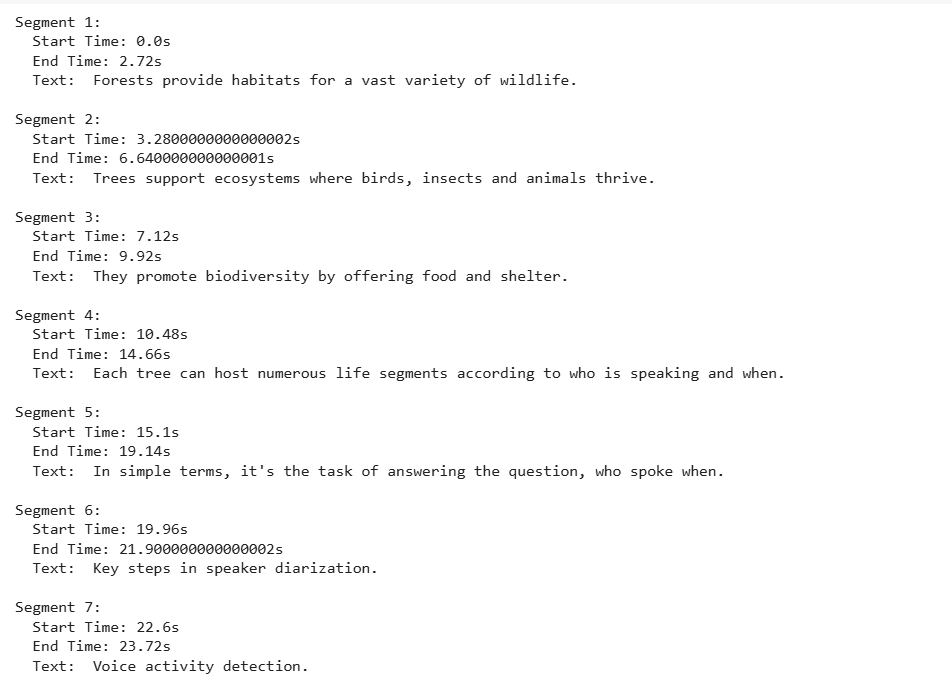
## RESULTS AND ANALYSIS

**4.1 Input Mixed Audio File**



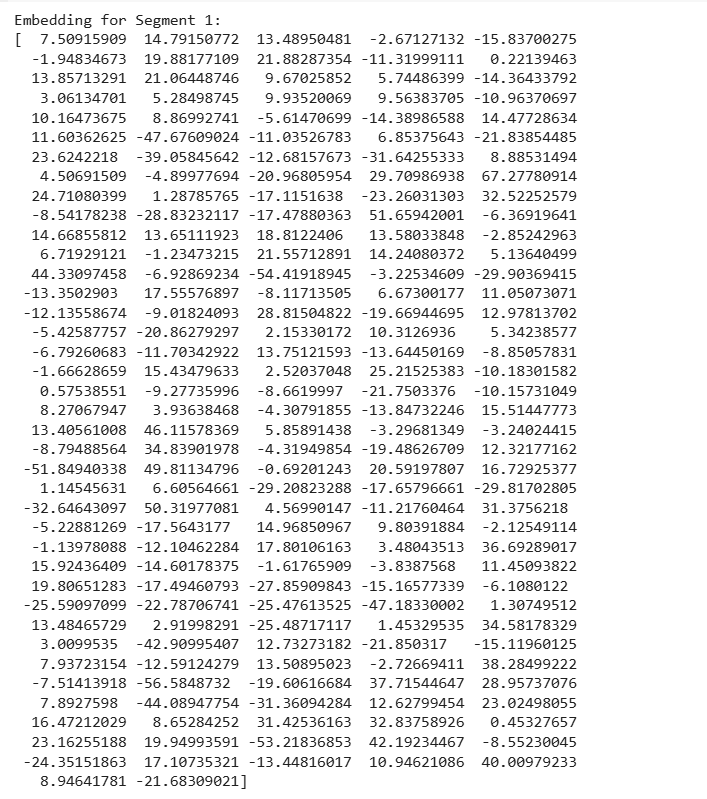
**Fig: To upload the mixed audio file**

**4.2 Segments of mixed audio file with time stamps**

****

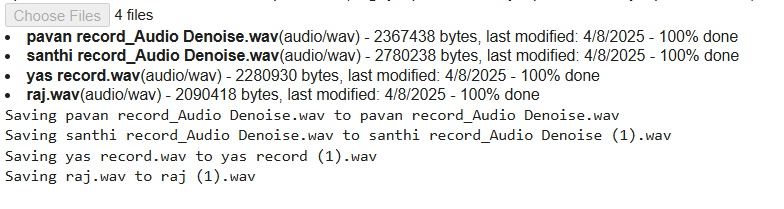
**Fig: segments of mixed audio file with time stamps**

**4.3 Embedding for Segments**



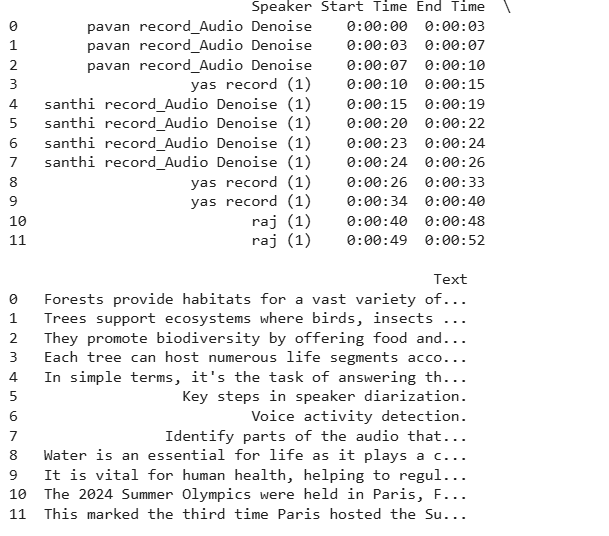
**Fig: Embedding for segments**

**4.4 Uploading the reference files**



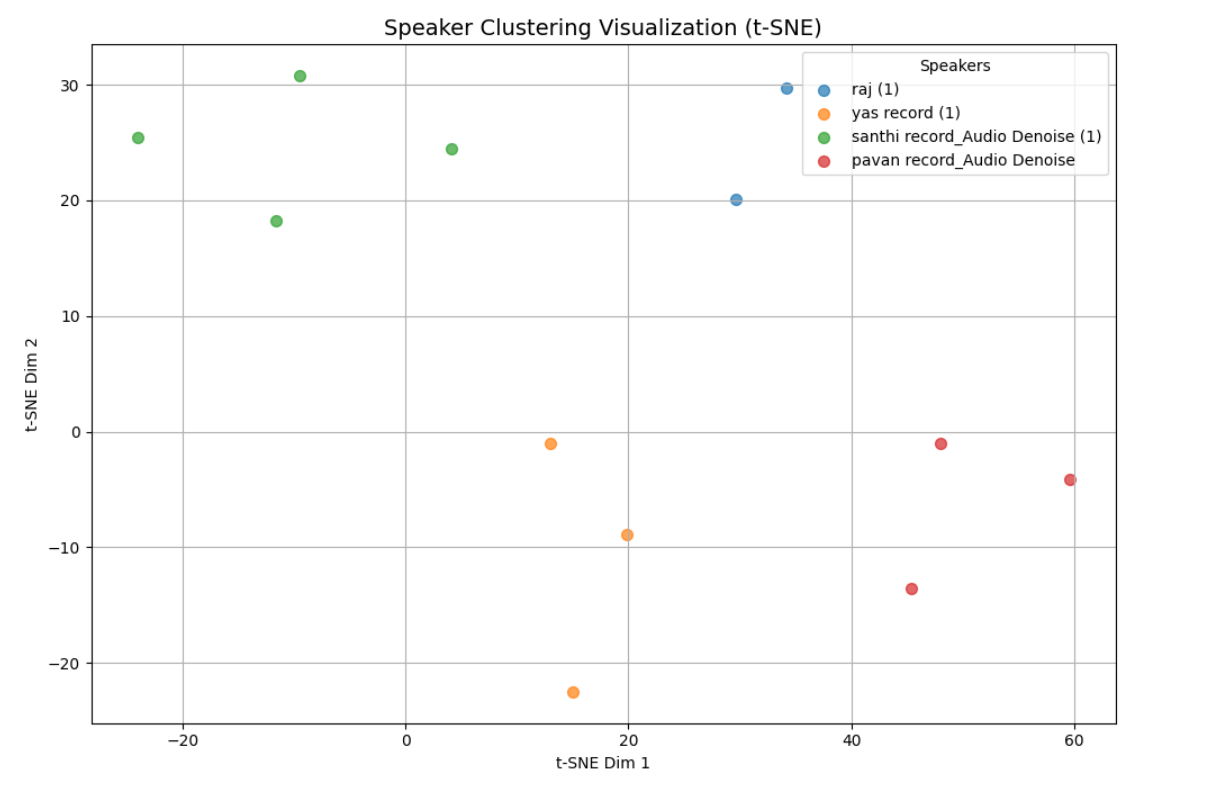
**Fig: uploading refence files**

**4.5 Dialogue Breakdown by Speaker and Time**



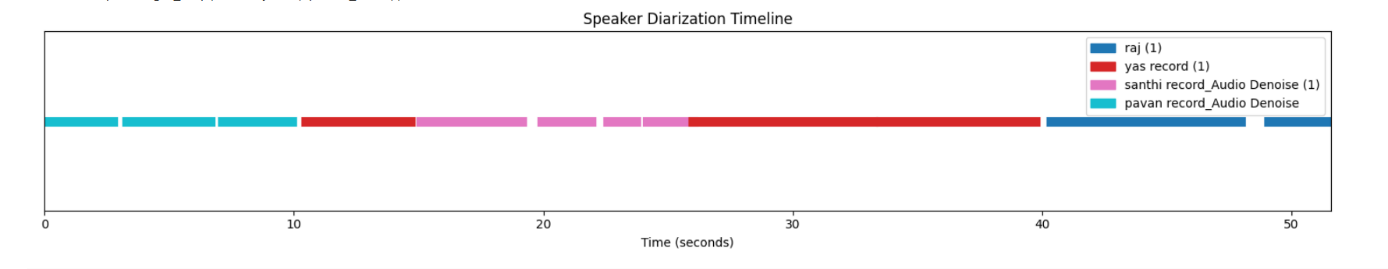
**Fig: Dialogue Breakdown by Speaker and Time**

**4.6 Speaker clustering visualization**

****

**Fig: Speaker clustering visualization t-SNE**

**4.7 Speaker diarization timeline**

****

**Fig: Speaker diarization timeline using matplot**

**Chapter 5**

## APPLICATIONS AND FUTURE SCOPE

**5.1 Real-World Applications**

List where your system can be effectively used. For example:

**a. Meeting Transcriptions**

* Automatically transcribe and identify speakers in corporate or academic meetings.
* Generates structured transcripts for records and follow-ups.

**b. Courtroom Proceedings**

* Helps in maintaining accurate and speaker-attributed transcripts during court trials.
* Reduces manual effort of note-taking.

**c. Customer Service & Call Centers**

* Diarizes customer calls to analyze who said what during a conversation.
* Useful for training, quality checks, and dispute resolution.

**d. Podcast & Interview Processing**

* Automatically breaks down long audio content into speaker-attributed dialogues.
* Improves accessibility via subtitles.

**e. Multilingual Conference Transcription**

* Transcribes and distinguishes speakers in global webinars or summits.
* Can aid in translation and localization efforts.

**5.2. Research and Development Opportunities**

Highlight what future improvements or research directions can be pursued.

**a. Real-Time Diarization**

* Enhancing the system to work in real-time streaming audio.
* Useful for live applications like broadcasting or live meetings.

**b. Multilingual and Code-Switching Support**

* Extend support for speakers who switch between languages mid-speech.

**c. Emotion and Sentiment Analysis**

* Integrating emotional detection with diarization to capture tone and mood of conversations.

**d. Speaker Adaptation**

* Creating adaptive profiles that improve recognition accuracy over time per individual speaker.

**e. Integration with Voice Assistants**

* Enabling voice assistants to handle multi-speaker environments more effectively.

**5.3 Limitations and Future Scope**

**Briefly mention limitations and how they can be addressed:**

* **Current Limitations:**
  + Accuracy drops with noisy backgrounds.
  + Similar-sounding voices might confuse the clustering algorithm.
* **Future Enhancements:**
  + Improving noise robustness.
  + Enhancing speaker clustering accuracy using deep learning techniques.
  + Incorporating visual cues from video for improved diarization (audio-visual diarization).

**Chapter 6**

## REFERENCES

1. **Radford, A., et al. (2022).** Whisper: Robust Speech Recognition via Large-Scale Weak Supervision.  
   *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP).*
2. **Zhang, A., Wang, Q., Zhu, Z., Paisley, J., & Wang, C. (2019).** Fully Supervised Speaker Diarization.  
   *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019*, pp. 6301–6305.  
   DOI: 10.1109/ICASSP.2019.8682586
3. **Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018).** X-Vectors: Robust DNN Embeddings for Speaker Recognition.  
   *ICASSP 2018*, pp. 5329–5333.  
   DOI: 10.1109/ICASSP.2018.8461375
4. **Reynolds, D. A. (2002).** An Overview of Automatic Speaker Recognition Technology. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2002*, Vol. 4, pp. 4072–4075.  
   DOI: 10.1109/ICASSP.2002.5744466
5. **Bredin, H. (2017).** TristouNet: Triplet Loss for Speaker Turn Embedding.  
   *ICASSP 2017 - IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5430–5434.  
   DOI: 10.1109/ICASSP.2017.7953120
6. **Vaswani, A., et al. (2017).** Attention is All You Need.  
   *Advances in Neural Information Processing Systems (NeurIPS), 2017*.  
   <https://arxiv.org/abs/1706.03762>
7. **McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015).** Librosa: Audio and Music Signal Analysis in Python.  
   *Proceedings of the 14th Python in Science Conference*, pp. 18–24.

**©All copyrights reserved to Department of Electronics and Communication Engineering,**

**Rajiv Gandhi University of Knowledge Technologies – Ongole Campus**

**----------------------------------------------xxxxxx--------------------------------------------------------**