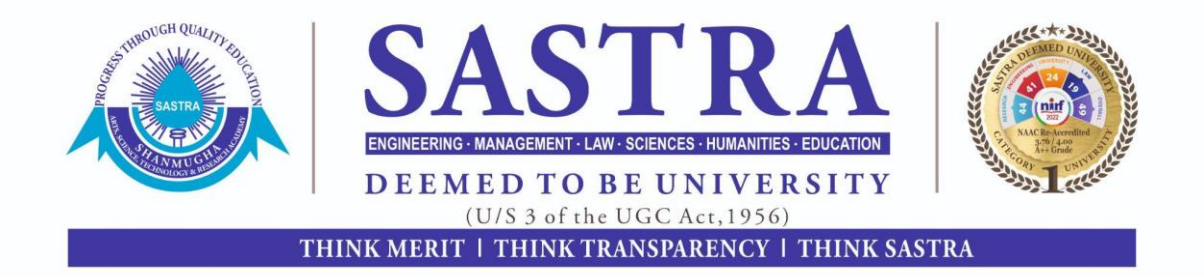
Speaker Change Detection and Diarization

Submitted by
JAYAPRADEEPKUMAR P (Reg No. 125015056, B. Tech I.T)
RAMANATHAN K (Reg No. 125003249, B. Tech Computer Science and Engineering)

CSE300 – Mini Project

Report submitted to SASTRA Deemed to be University

As per the requirement for the course



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "Project Title (Title Case – First Letter Caps)" submitted as a requirement for the course, CSE300 / INT300 / ICT300: MINI PROJECT for B.Tech. is a bonafide record of the work done by Mr. Ramanathan K(Reg. No.125003249, B.tech C.S.E) and Mr. Jayapradeepkumar P(Reg. no. 125015056.B.tech I.T) during the academic year 2023-24, in the School of Computing, under my supervision.

Signature of Project Supervisor :

Name with Affiliation :

Date :

Mini Project *Viva voice* held on _____

Examiner 1 Examiner 2

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend my heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K. Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Algeswaran**, Associate Dean, Students Welfare.

My guide **Dr. V. S. Shankar Sriram**, Dean, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped me in making progress throughout our project work. I also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. I thank you all for providing me/us an opportunity to showcase my skills through the project.

List of Figures

Abbreviations

Abbreviations	Meaning
SCD	Speaker Change Detection
SD	Speaker Diarization
LSTM	Long Short Term Memory
MFCC	Mel Frequency Cepstral Coefficient
VAD	Voice Activity Detection
BERT	Encoder Representations from Transformers
CLV	Cross lingual Vector
GE2E	Generalized End to End
DER	Diarization error rate

Notations

Notation	Meaning
e = (eij, …, ek)	Embedding vector
ck	centroid
N	Speakers
M	Utterances
$\boldsymbol{\phi(x,y)}$	$cosine(x,y)$
N	Size of segment

Abstract

This project addresses limitations in existing methods for multilingual speaker change detection and diarization, such as Long Short-Term Memory (LSTM) and Mel Frequency Cepstral Coefficients (MFCC) , with a focus on achieving accurate speaker identification in voice-based applications across multiple languages. The proposed framework incorporates a language-agnostic version of BERT, known for its contextual understanding in text, enhancing its versatility to seamlessly handle multiple languages. Departing from conventional techniques, the framework adopts a text-independent speech-embedding approach, transforming speech signals into numerical representations that capture distinct vocal characteristics independent of speech content. The innovation extends to the inclusion of Cross Lingual Vectors (CLV) equipped with a vectorial space, facilitating training by projecting examples from one language into this space. This approach empowers the model to make predictions across various languages, enhancing its adaptability to nuanced linguistic intricacies. The experimental evaluation utilizes the ICSI Meeting Corpus, a dataset comprising approximately 70 hours of meeting recordings. The results demonstrate the framework's superior performance compared to conventional methods like Based Speaker Diarization (BIC), with a substantial increase in performance in terms of Speaker change detection precision and Diarization error rate. Moreover, the proposed system is acclaimed for its flexibility and scalability, positioning it as a valuable tool for applications demanding precise speaker identification in multilingual environments. The framework not only enhances accuracy but also demonstrates adaptability and efficiency in handling diverse linguistic scenarios, establishing it as a powerful solution for real-world multilingual applications.

Keywords : Speaker change detection, Speaker diarization, Multilingual Application

Table of Contents

Chapter 1 Summary of Base Paper

Title	:	**Speech and multilingual natural language framework for speaker change detection and diarization**
Publisher	:	**Or Haim Anidjar , Ariel University , Israel**
Year	:	**2023**
Indexed in	:	**Science Direct**
Base paper URL	:	**Speech and multilingual natural language framework for speaker change detection and diarization - ScienceDirect**

Primary contributions

The Research Paper presents a novel approach to solving the speaker detection problem .The voice and textual feature extraction suggested by the paper is a major innovation of this project. The Language agnostic BERT model referred through the paper projects contextual similarities in speech to diarize the speaker change moments. The GE2E model for voice embedding analyses overall utterance in gaps and gives relative similarity between speakers.

1.1 Introduction

Speaker Change Detection (SCD) plays a pivotal role in numerous applications, it is used tasks such as automatic speech transcription and audio indexing. In the context of speech recognition, an SCD system aims to find out "who is speaking and when" within an audio recording. This involves identifying the intervals of speech utterances and associating them with specific speaker identities. Consequently, the accuracy of the SCD process is important. the effectiveness of these systems hinges on the quality of speaker change detection.

Traditional speaker change detection models struggle with maintaining accuracy across languages due to linguistic variations. existing methods for multilingual speaker change detection and diarization, such as Long Short-Term Memory (LSTM) and Mel Frequency Cepstral Coefficients(MFCC).
Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) architecture designed to model sequential data with long-range dependencies. Long short term Memory (LSTM) still struggle with modelling relationships that span very long sequences. LSTM also is computationally expensive, time consuming and overfitting for the problem. In speaker change detection, these MFCC-based feature vectors are often analysed over time to detect changes in speakers. For example, changes in the distribution or similarity of MFCC vectors between consecutive frames may indicate a transition between speakers. MFCC may

not capture long term dependencies as they compute vectors in short spans, it is also highly sensitive to noise and has limited adaptability.

1.2 System Description

Basically, an SD system is composed of the following components:

(i) a Voice Activity Detection (VAD) engine, which locates and omits non-speech segments, and next split the audio-recording into speech utterances that are supposed to be spoken by one speaker each

(ii) a module of embedding extraction, on which speaker-discriminative embeddings such as the MFCC

(iii) a component that estimates the number of speakers in the audio recording, and a clustering module that clusters speaker identities to the speech utterances

Previous works have already shown that auxiliary textual information can be of great use for detection of speaker-turns and the diarization systems' performance. In this project, we suggest a framework for speaker-turn estimation, as well as the determination of clustered speaker identities to the SD system.

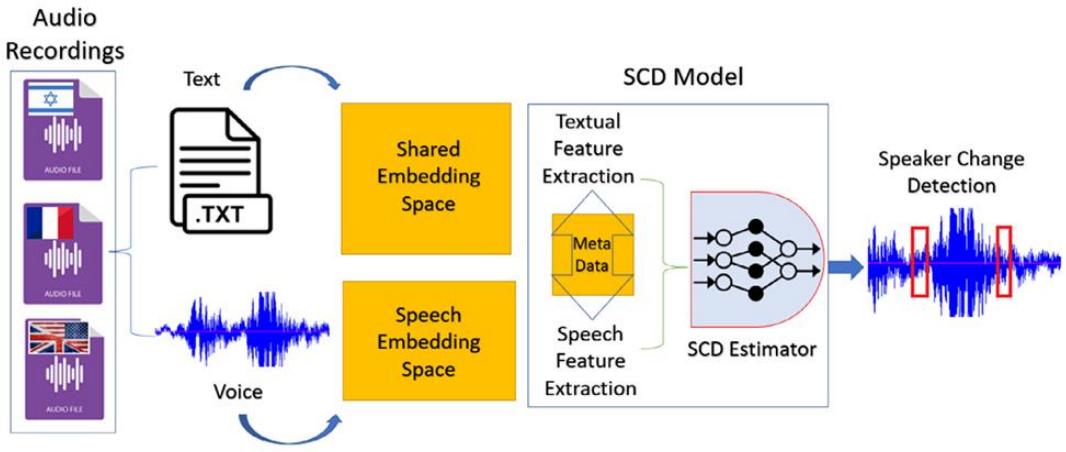


Fig.1 illustration of the suggested voice and textual data framework. the automatic transcription as well as the spectrogram are extracted. Then, the textual information and the speech signal are extracted into features

1.3 Voice Embedding

we use a resemblyzer python package for analyse the audio alone without any textual interpretation initially. The model uses input utterances in order to learn speaker-discriminative embeddings. For this purpose, proposed the Generalized End-to-End (GE2E) loss function, whose training process is done by a parallel process of a respectable amount of speech segments at once, where each such batch contains a mixture of N different speakers, for which M speech segments of each speaker. The embedding vector (d-vector) is defined ,where eji represents the embedding vector of the jth speaker's ith utterance.

Formula:-

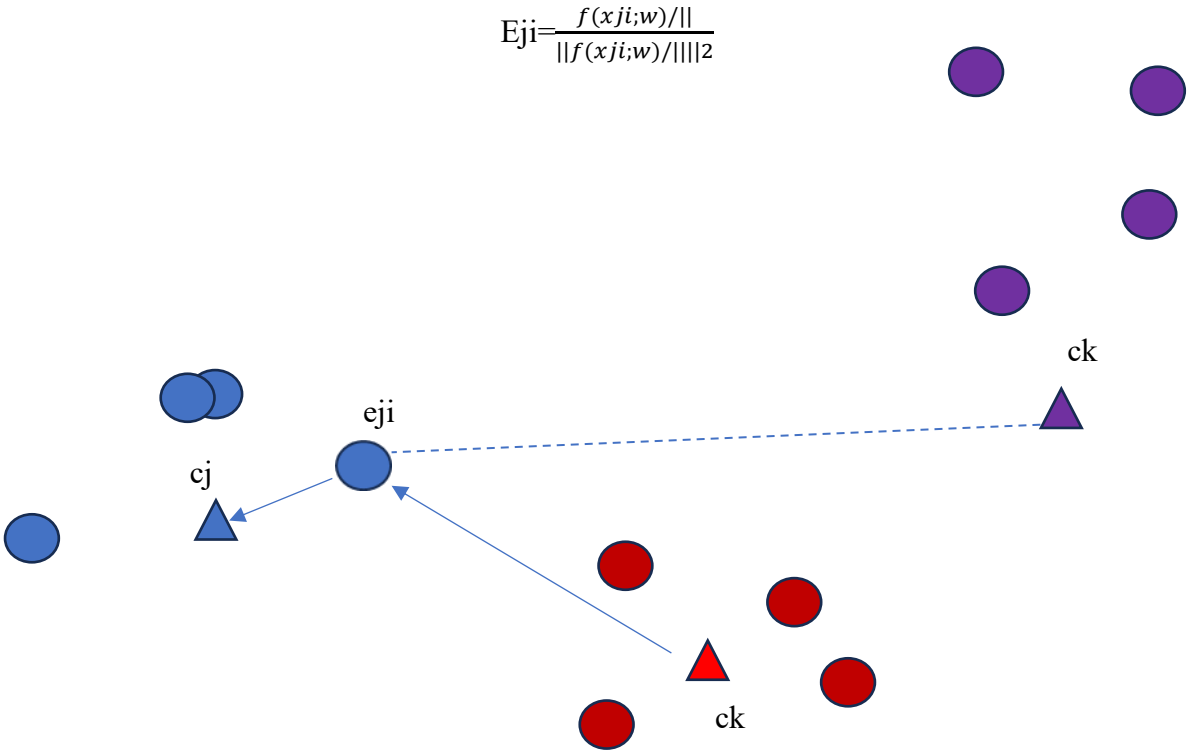


Fig. 2. GE2E loss pushes the embedding towards the centroid of the true speaker, and away from the centroid of the most similar different speaker.

The similarity matrix Sj,i,k is defined as the scaled cosine similarities between each embedding vector ej,i to all centroids ck

Formula:-

$$s = w\,cos(eij, ck) + b$$

1.4 Textual embedding

we chose Language Agnostic-BERT as the text-encoding method employed, since the Language Agnostic BERT supports multiple-languages word and sentence embedding due to the utilization of transformers and the attention mechanism. Bert distinguishes similarities based on cosine similarity on the given formula where N is size of batch. BERT is trained by google over a large amount if data, we used the BERT package for textual feature selection

$$L = -\frac{1}{N}\sum_{i=1}^{N} \quad \frac{e^{e\phi(xi,yi)-m}}{e^{e\phi(xi,yi)-m} + \sum_{n=1}^{N} e^{\phi(xi,yn)}}$$

$\phi(x,y)$ is the embedding space similarity of the sentence x and the sentence y, and defined as $\phi(x,y) = cosine(x,y)$

1.5 Aassesment and Accuracy

For Each Sliding Window (SW) Two distinct measures have been used by the authors to assess their model: F1score, DER(Diarization Error Rate). The outcomes demonstrate that the model

generates More Confident Speaker recognition than by previous models. Even in presence of external disturbances and unclear low volume sounds at audio.

Chapter 2 Merits and Demerits of Base Paper

2.1 Why Speaker detection

The problem taken into account in the paper is a new issue faced in the world of artificial intelligence. State Nowadays, SCD components are used for state-of-the-art SD systems, mainly those that exploit textual information. Some of the major applications of SCD and SD are

2.1.1 Automatic Speech Recognition (ASR) – SCD is used to segment audo reccordings into distinct speaker turns, facilitating accurate transcription of each speaker's speech.Diarization helps in adapting ASR models to individual speakers by providing labeled speaker-specific training data.

2.1.1 Speaker Recognition and Verification - Diarization provides labeled segments of speech for training speaker recognition models, enabling systems to identify speakers based on their unique characteristics. SCD helps in determining speaker boundaries for verifying speaker identities in authentication systems.

2.1.2 Meeting Transcription and Summarization - SCD and SD are essential for transcribing meetings with multiple speakers, ensuring accurate attribution of speech to each participant.By segmenting and identifying speakers, diarization facilitates the summarization of meeting discussions, enabling the extraction of key points and action items.

2.1.3 Voice User Interfaces (VUIs) - Diarization assists VUIs in providing personalized responses by recognizing individual speakers and tailoring interactions accordingly.SCD helps in managing speaker turns in conversational VUIs, ensuring smooth and coherent interactions between users and the system.

2.2 Merits

This study showcases the effectiveness of integrating textual information and speech analysis to create a hybrid model for Speaker Change Detection (SCD) within a Speaker Diarization (SD) system.

The paper indulges various studies from different fields of study and research to solve this problem.

Dataset from different languages can be applied in the model and valid results are outputted.

2.3 Demerits

The paper doesn’t solve the problem of implementation of a new language , so training the system to other languages is required before we can effectively diarize the multilingual audio in textual context.

Due to limitation of technology we have reviewed the work of the paper . training hours of data is not possible for Personalised computer execution. Hence we have set a limit of 3 min audio.

Chapter 3 Code

3.1 audio

```
from resemblyzer import preprocess_wav, VoiceEncoder
from demo_utils import *
from pathlib import Path
from pydub import AudioSegment
import csv


# DEMO 02: we'll show how this similarity measure can be used to perform speaker diarization
## Get reference audios
# Load the interview audio from disk
def cut_audio(input_file, output_file, start_time, end_time):
    """
    Cut an audio file to the desired time range.
    Args:
        input_file (str): Path to the input audio file.
        output_file (str): Path to the output audio file.
        start_time (int): Start time in milliseconds.
        end_time (int): End time in milliseconds.
    """
    # Load the audio file
    audio = AudioSegment.from_mp3(input_file)
    # Cut the audio to the desired time range
    audio_cut = audio[start_time:end_time]
    # Export the cut audio to a new file
    audio_cut.export(output_file, format="mp3")
```

```
input_file = "input_audio.mp3"
output_file = "output_cut.mp3"
start_time = 0# Start time in milliseconds (e.g., 5 seconds)
end_time = 180000   # End time in milliseconds (e.g., 15 seconds)
cut_audio(input_file, output_file, start_time, end_time)
wav_fpath = "output_cut.mp3"
wav = preprocess_wav(wav_fpath)

# Cut some segments from single speakers as reference audio
segments = [[0,10],[39,44],[59,104]]
speaker_names = ["reporter-male","interviewer-female","topG-male"]
speaker_wavs = [wav[int(s[0] * sampling_rate):int(s[1] * sampling_rate)] for s in segments]

## Compare speaker embeds to the continuous embedding of the interview
# Derive a continuous embedding of the interview. We put a rate of 16, meaning that an
# embedding is generated every 0.0625 seconds. It is good to have a higher rate for speaker
# diarization, but it is not so useful for when you only need a summary embedding of the
# entire utterance. A rate of 2 would have been enough, but 16 is nice for the sake of the
# demonstration.
# We'll exceptionally force to run this on CPU, because it uses a lot of RAM and most GPUs
# won't have enough. There's a speed drawback, but it remains reasonable.
encoder = VoiceEncoder("cuda")
print("Running the continuous embedding on cpu, this might take a while...")
_, cont_embeds, wav_splits = encoder.embed_utterance(wav, return_partials=True, rate=16)

# Get the continuous similarity for every speaker. It amounts to a dot product between the
# embedding of the speaker and the continuous embedding of the interview
speaker_embeds = [encoder.embed_utterance(speaker_wav) for speaker_wav in
speaker_wavs]
similarity_dict = {name: cont_embeds @ speaker_embed for name, speaker_embed in
                   zip(speaker_names, speaker_embeds)}
```

```
## Run the interactive demo
interactive_diarization(similarity_dict, wav, wav_splits)

# Create a list to store segment names and their similarity scores
segment_similarity_data = []

# Iterate through similarity_dict to gather segment names and similarity scores
for segment_name, similarity_scores in similarity_dict.items():
    # Create a dictionary to store segment name and similarity score
    segment_data = {'Segment': segment_name}

    # Add similarity scores to the dictionary
    for i, score in enumerate(similarity_scores):
        segment_data[f'Similarity_{i+1}'] = score.item()

    # Append the dictionary to the list
    segment_similarity_data.append(segment_data)

# Define the output CSV file path
output_csv_file = 'segment_similarity_scores.csv'
# Write the data to a CSV file
with open(output_csv_file, 'w', newline='') as csvfile:
    fieldnames = ['Segment'] + [f'Similarity_{i+1}' for i in range(len(similarity_scores))]
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    # Write the header
    writer.writeheader()
    # Write the segment data
    for segment_data in segment_similarity_data:
        writer.writerow(segment_data)
```

```
print(f"Segment similarity scores saved to {output_csv_file}")
```

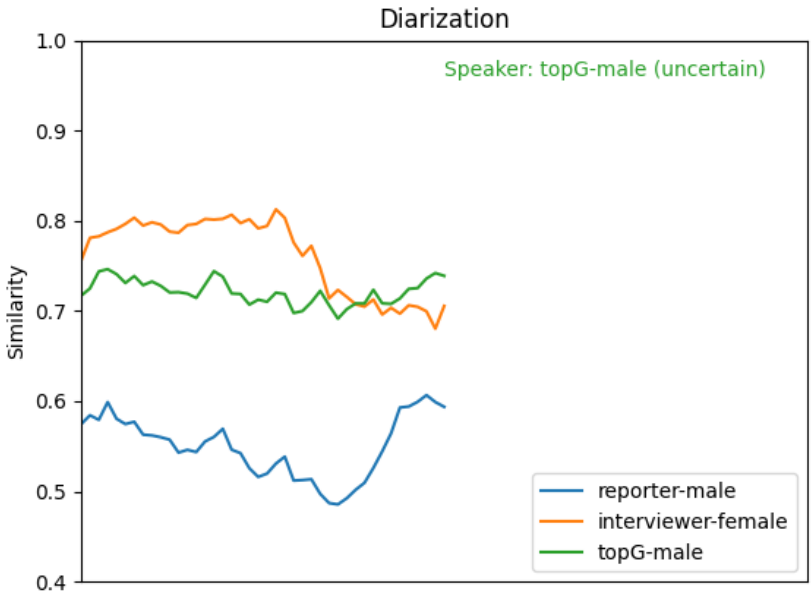
3.2 Text

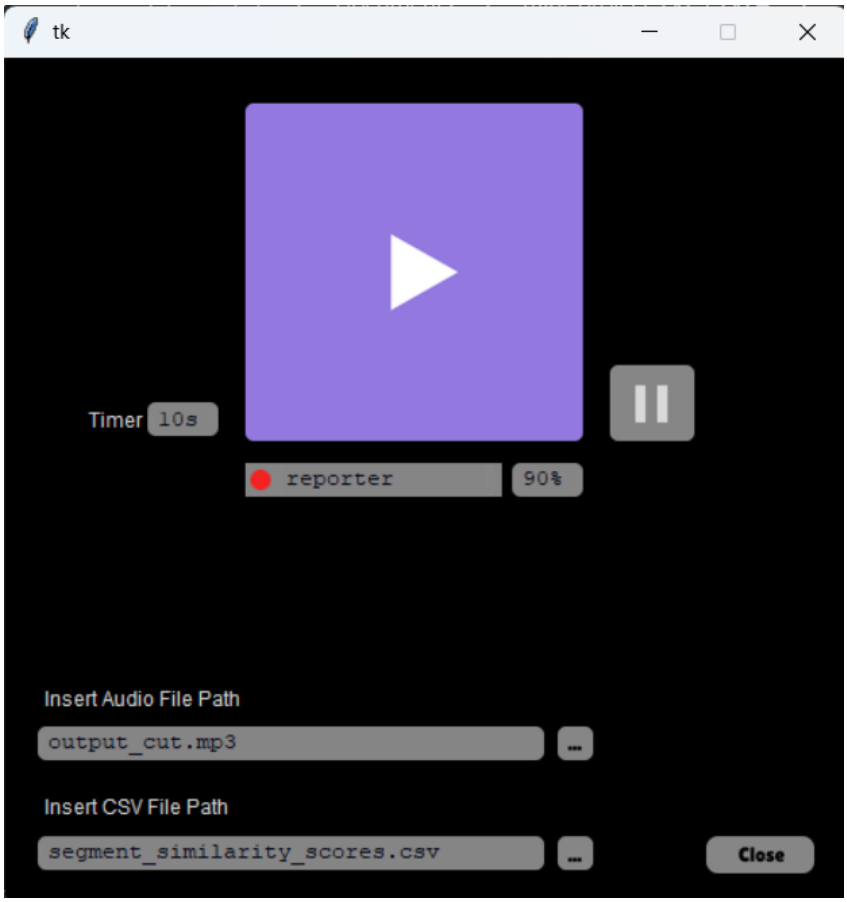
```
from transformers import BertTokenizer, BertModel
import torch
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Function to encode text using BERT and obtain embeddings
def encode_text(text):
    # Tokenize input text
    inputs = tokenizer(text, return_tensors='pt', padding=True, truncation=True)
    # Get BERT embeddings
    with torch.no_grad():
        outputs = model(**inputs)
        embeddings = outputs.last_hidden_state[:, 0, :]  # Extract the [CLS] token embedding
    return embeddings.numpy()

# Function to detect speaker changes based on cosine similarity of BERT embeddings
def detect_speaker_changes(transcript):
    # Encode each text segment in the transcript using BERT
    embeddings = [encode_text(segment['text']) for segment in transcript]

    # Calculate cosine similarity between consecutive segments
    speaker_changes = []
```

```
    for i in range(1, len(embeddings)):
        # Compute cosine similarity between embeddings of consecutive segments
        similarity = cosine_similarity(embeddings[i-1], embeddings[i])[0][0]

        # Consider contextual information from previous and next segments
        if i > 1:
            prev_similarity = cosine_similarity(embeddings[i-2], embeddings[i-1])[0][0]
        else:
            prev_similarity = 0.0  # No previous segment
        if i < len(embeddings) - 1:
            next_similarity = cosine_similarity(embeddings[i], embeddings[i+1])[0][0]
        else:
            next_similarity = 0.0  # No next segment

        # If cosine similarity is below a threshold and is lower than both previous and next similarities,
        # consider it a speaker change
        if similarity < 0.9 and similarity < prev_similarity and similarity < next_similarity:
            speaker_changes.append(transcript[i]['timestamp'])
    return speaker_changes

# Sample transcript data
transcript = [ {"timestamp": "0:00", "text": "now the PPC has questioned the controversial social media influencer Andrew Tate at his home in the Romanian capital Bucharest"},
    {"timestamp": "0:04", "text": "Tate is under house arrest and being investigated by Romanian prosecutors for accusations including rape, human trafficking, and exploiting women which he denies"},
    {"timestamp": "0:08", "text": "the BBC challenged him on whether his views about women broadcast to his millions of online followers harmed young people as many teachers and police officers claim"}
]
# Detect speaker changes using text-based BERT embeddings with contextual analysis
```

```
speaker_changes = detect_speaker_changes(transcript)
print("Speaker changes detected at timestamps:", speaker_changes)
```

Chapter 4 SNAPSHOTS





Chapter 5 Conclusion and Future Plans

In conclusion, the speaker change detection and diarization project has demonstrated accurate segmentation and identification of speakers in audio data processing. Through the implementation of robust algorithms and techniques, the challenge of detecting speaker changes and clustering speech segments to attribute them to specific speakers effectively has been addressed. the benefits of exploiting textual information and speech analysis in order to construct a multilingual hybrid model for the SCD problem, and how to apply it over an SD system has been highlighted With datasets in French, and English.

With availability of resources we can train a Unsupervised clustering model to interpret text and voice data and classify them wherever the results come out best. With use of server we can feed hours of text audio to calculate multi lingual similarities and train a model such that different speakers audio will be identified in intervals along with the language in wich he or she may have spoken. Additional future work might be the development of an end-to-end system that detects the spoken language in an audio-recording, apart from detecting who spoke and when.

Chapter 6 References

- Park, T. J., & Georgiou, P. G. (2018). Multimodal speaker segmentation and diarizationusing lexical and acoustic cues via sequence to sequence neural networks. InINTERSPEECH
- Wan, L., Wang, Q., Papir, A., & Moreno, I. L. (2018). Generalized end-to-end loss for speaker verification. In ICASSP (pp. 4879–4883).
- Anidjar, O. H., Lapidot, I., Hajaj, C., Dvir, A., & Gilad, I. (2021). Hybrid speech and text analysis methods for speaker change detection. IEEE/ACM Transactions on Audio,Speech, and Language Processing, 29, 2324–2338.
- Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2020). Language-agnostic Bert sentence embedding

Chapter 7 Appendix

Anidjar OH, Estève Y, Hajaj C, Dvir A, Lapidot I. Speech and multilingual natural language framework for speaker change detection and diarization. Expert Systems with Applications. 2023 Mar 1;213:119238. [Link]