BTP - 1

Speech Transcription Verification

Srujana Vanka - 2020102005
Shreeya Singh - 2020102011

Objective

To develop a system to verify speech transcriptions by splitting a large audio file into smaller chunks, transcribing each chunk, and presenting them to the user for verification and correction.
Enabling them to either "save" or "discard" the transcriptions.

Understanding of our project

- Our project aims to develop a system for **verifying speech transcriptions** by dividing a large audio file into smaller segments, converting each segment into text using the AP provided, and presenting it to the user for verification and correction.
- Develop an interface that **displays the transcript text** and allows the user to **play the corresponding audio chunk.**
- The user will have the option to edit the text if it is incorrect and save/discard the transcription.
- The system will ensure that each audio chunk is reviewed only once and will move on to the next chunk once the user has verified the transcription i.e should not appear again once the application is restarted.
- Overall, our project aims to verify the efficiency of speech transcriptions and make the process more manageable for transcribers through an easy to use interface.

- **There are 3 main components:**

1. Audio Splitting
2. Speech-to-text conversion
3. Transcription verification

Audio splitting

Why do we split the audio file into chunks?

Easier processing:

- By splitting into smaller chunks, it becomes easier to process each chunk separately and in parallel, which can speed up the overall processing time.

Improved accuracy:

- It improves the accuracy of speech recognition and transcription.
- Becomes easier to isolate and analyze each segment separately, which can lead to more accurate transcriptions.

User interface:

- Makes it more manageable for the user to review and verify the transcriptions.

Overall, splitting a large audio file into smaller chunks is done to make processing more efficient, improve accuracy, and make the transcription review process more manageable for users.

How is the audio being split?

- Audio splitting is done **based on silence.**
- Silence is used as a criterion for splitting audio because it is often a **natural separator** between speech or other sound events.
- Non-speech sections of audio are filtered out to improve the accuracy of the silence detection.
- The `frame_generator()` function generates frames of audio of a fixed duration (30 ms) from the input audio.
- These frames are then passed to the `vad_collector()` function which processes the frames in a sliding window technique to determine if segment is voiced or unvoiced.

Modules used

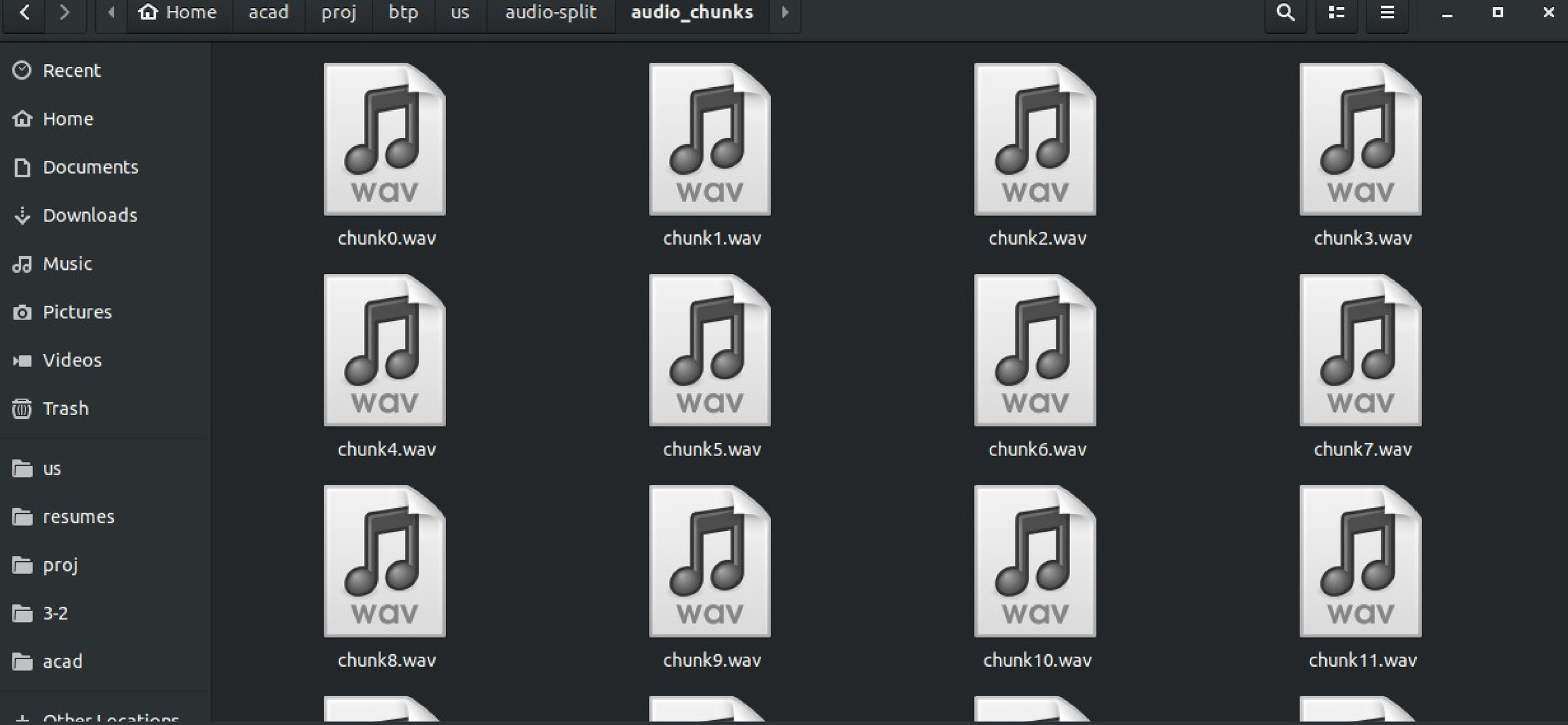
The code is written in Python and is used to split an audio file into smaller segments based on silence using the Pydub and WebRTC Voice Activity Detection (VAD) libraries.

PyDub library

- The library is used for reading and writing audio files in various formats.
- Splits the audio based on silence.

WebRTC Voice Activity Detector (VAD)

- Filters out non-speech sections of audio to improve the accuracy of the silence detection
- The `vad_collector` function processes the frames(30ms) to detect voiced and unvoiced segments, using a sliding window of size to determine if a segment is voiced or not.

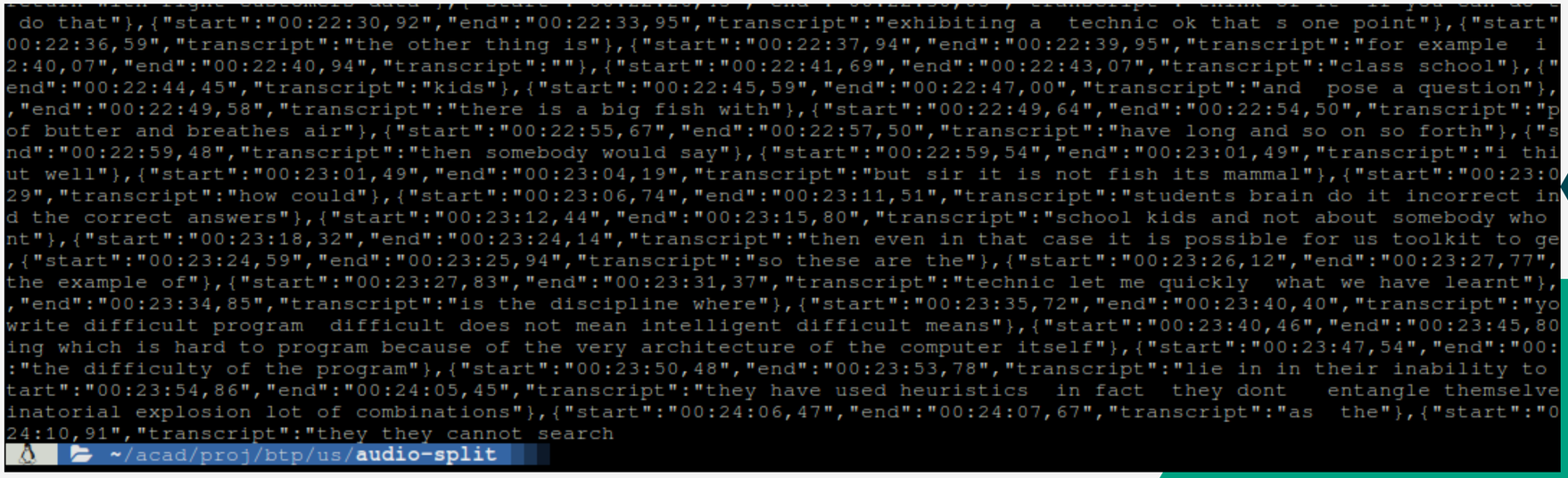


Splittling the audio file into chunks based on silence

Speech-to-Text Conversion

Speech to text conversion

- Each audio chunk is converted into text transcripts using the **Speech-to-Text API** provided.
- The API is hosted at: https://asr.iiit.ac.in/ssmtapi/
- A POST request is made to the API with the audio file as input.
- The response from the API is in JSON format



Speech to text conversion

- so the code converts this JSON data into a Python dictionary to extract the desired response data
- The code then **extracts the text from the 'transcript' key** of each dictionary and concatenates them to form the complete transcription of the audio chunk. This complete transcription is saved to a text file.

Transcription Verification

What is transcription verification?

- Transcription verification is the process of **reviewing and checking the accuracy of a transcription.**
- It involves comparing the transcribed text with the original audio recording to ensure that the words and phrases have been transcribed correctly.
- Any errors or discrepancies are identified, and the transcription is corrected as necessary.

Why is it done?

- The purpose of transcription verification is to ensure the accuracy and completeness of the transcription.
- It is done to reduce the risk of errors or misunderstandings.
- It is an important step in many applications of speech recognition and transcription, such as captioning, subtitling, and data analysis.

Interface

- Audio player to play the audio chunk
- Textbox displaying the transcription
- Transcription can be edited in the textbox
- Save the final transcription
- Discard the transcription
- Move to the next chunk

File Edit View History Bookmarks Tools Help

Speech Transcription Ver

localhost:3000/#

Speech Transcription Verification App Home Enable DarkMode

Verify the transcription of the below audio chunk

0:00 / 0:02

Transcript 24

you still are able to recognise him

Save Discard Next

Components of the interface

Create-react-app

We made use of this command line tool to build the application installs a project template, required dependencies, development server
It is hosted on `localhost:3000` using npm start

Backend server

running on `localhost:5000`
The server is responsible for running two Python scripts that do audio chunking and transcript generation using the `child_process` module in Node.js.
Also used for saving and discarding through the backend

Displaying Transcripts and Audio Chunks

The React interface maintains an index number called the transcript number to display the respective transcript and audio chunk.
Local storage is also used to ensure that the transcript doesn't start again from 1 when the application is restarted.

Next

Uses the Axios library to fetch the transcript and audio files based on the transcript number. When the user clicks the "next" button, the transcript number is incremented, and the corresponding audio chunk and transcript are displayed on the interface.

Save/Discard

The client-side JavaScript application is making an **HTTP POST request to the backend server to save the text entered by the user.**

The axios library is used to make the HTTP POST request, and the destination URL for the request is `http://localhost:5000/save-text`, which is where the backend server is listening for POST requests to the `/save-text` endpoint.

The `text` and `transcriptNumber` parameters are passed in the request body.

The backend server then handles the POST request, reads the `text` and `transcriptNumber` parameters from the request body, and saves the `text` to a file named `transcript<transcriptNumber>.txt`.

Scope of extension

- Incorporating a previous button for navigating backward in the transcripts.
- Adding a reset button to go back to transcript 1 or adding an option to set the transcript number to any value and navigate to it.
- Accommodating multiple audio files for the app.
- Adding support for different languages.
- Adding a feature for downloading the transcripts in different formats.
- Adding support for editing the transcripts.