Speech-to-Text Transcription System.

Dan Otieno

EE 384 – Spring 2024.

Due Date: 05/03/2024.

Introduction / Background

The development of speech-to-text (STT) systems has always been important, especially with the advancement of human-computer interaction technologies. 90% of the world's data is unstructured; unsearchable and unorganized, most of which is voice or video data that needs to be changed into machine-readable data to be used for decision-making. Speech-to-text is technology that facilitates the conversion of human speech into written or typed text.

This project utilizes both Python and MATLAB to implement a speech-to-text transcription method capable of processing audio files and converting spoken language into written text. The system functions through the integration of speech recognition libraries and custom scripts that accept an audio input and produce a transcription output.

Theory

The theoretical foundation of this system lies in digital signal and natural language processing. The program functions by initially processing the input audio file to normalize and prepare the data for transcription. This involves reading the waveform of the audio file, which is then passed through a series of signal processing algorithms to enhance the audio quality and isolate the spoken words.

Speech recognition utilizes complex models trained on large datasets to accurately recognize and transcribe spoken language. These deep-learning models analyze the spectral features of the audio signal and compare them to known linguistic patterns to generate the corresponding text.

Algorithms Used:

SHORT-TIME FOURIER TRANSFORM (STFT):

The STFT algorithm is used in the MATLAB "spectrogram" function to perform calculations for the signal spectrogram. STFT divides the audio signal into short overlapping segments and computes the Fourier Transform for each segment to analyze its frequency content over time. Due to the time-frequency representation of the signal, we can visualize the frequency components evolving over time.

SPEECH RECOGNITION ALGORITHM:

Sourced from GitHub, the speech recognition algorithm used in the Python script utilizes the Recognizer class from the speech_recognition library. The algorithm typically involves processing the audio signal to extract relevant features such as mel-frequency cepstral coefficients (MFCCs), fundamental frequency (pitch), and formants. Those extracted features are used to train or feed into a machine learning or statistical model, which maps them to textual representations using techniques like Hidden Markov Models (HMMs), deep neural networks (DNNs), or recurrent neural networks (RNNs).

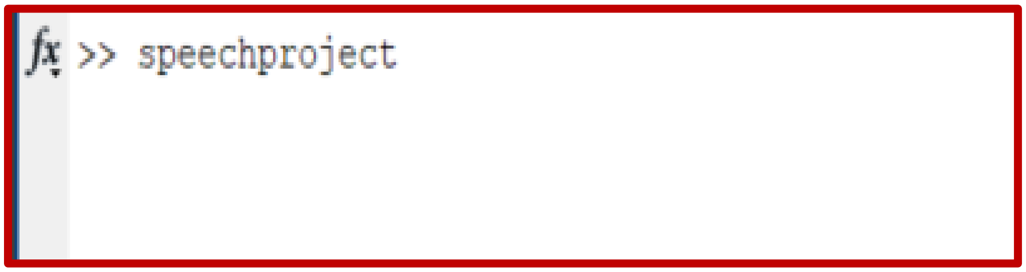
GOOGLE SPEECH RECOGNITION API:

Also utilized by the Python script utilizes to perform the actual transcription of the audio signal. This API combines advanced algorithms developed by Google to accurately transcribe speech into text, using deep learning techniques and large training datasets to achieve high accuracy in recognizing various spoken languages and dialects.

Simulation

INPUT SELECTION:

On running MATLAB script, user is first prompted for a python file by browsing through file explorer, then to browse and select audio file (this is in wav format).



```
>> speechproject
```

Figure 1: Running the MATLAB script to start program.

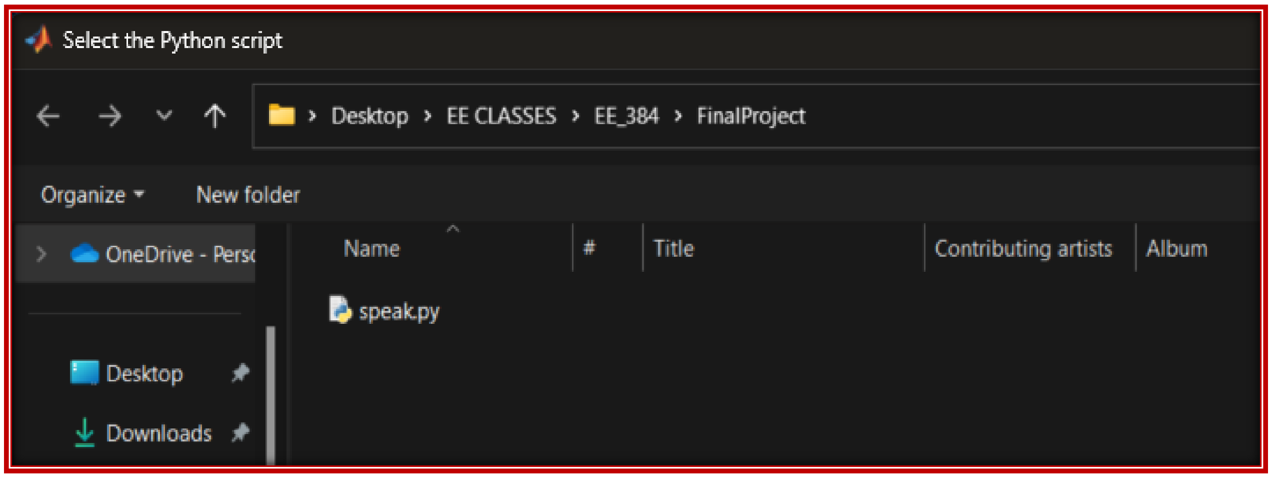


Figure 2: Selecting Python transcription file.

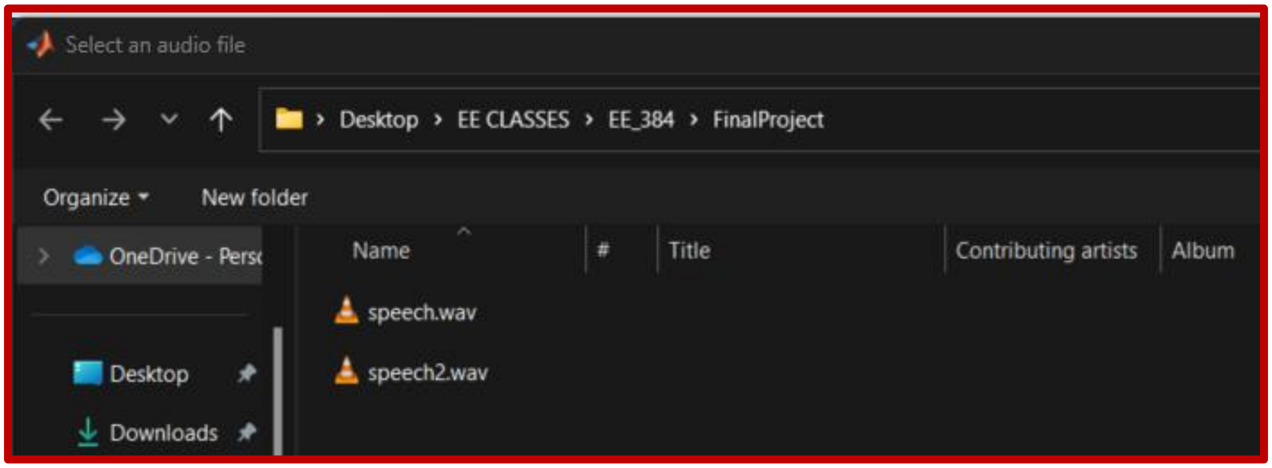


Figure 3: Selection of wav audio file.

SIGNAL PREPROCESSING:

MATLAB's "audioread" function loads the wav file and extracts the audio signal and sampling frequency (y and Fs respectively). The extracted signal is preprocessed to address issues, with methods such as normalization or noise reduction.

SIGNAL VISUALIZATION:

Using the "plot" and "spectrogram" MATLAB tools, signal waveform and spectrogram are generated. The waveform plot provides insights into the amplitude variations of the audio signal over time, while the spectrogram plot reveals its frequency content. Visualization of the signal helps us understand how different parts of the audio are processed and transcribed.

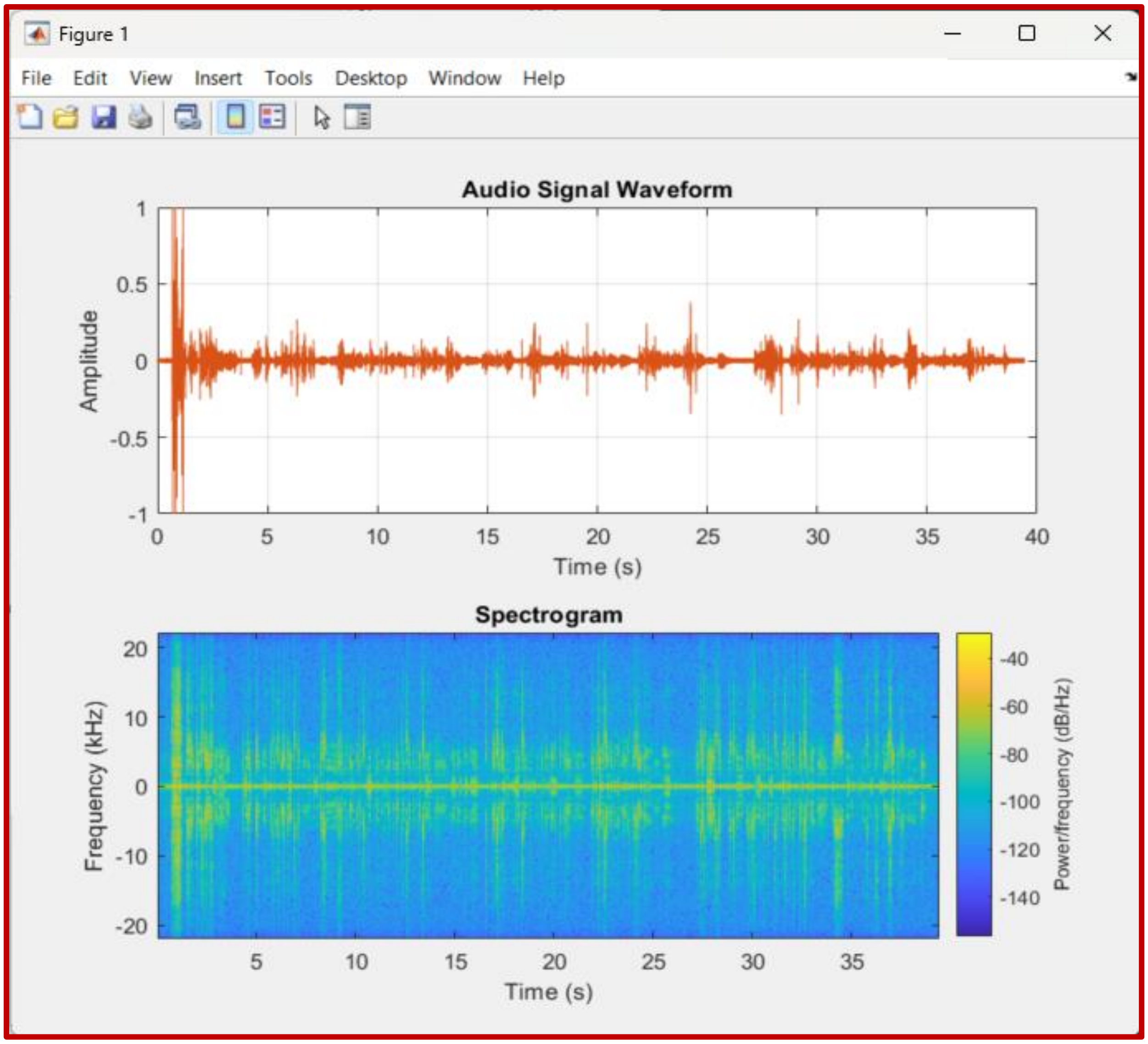


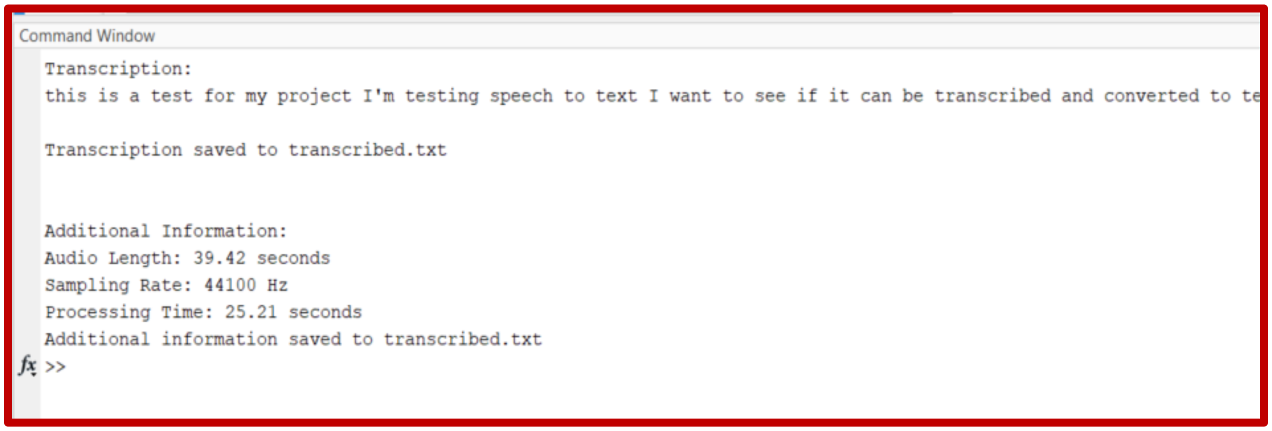
Figure 4: Waveform and Spectrogram output.

SPEECH-TO-TEXT CONVERSION:

MATLAB calls the Python script using the system function and passes the audio file path as an argument. Python utilizes the "speech_recognition" library and the Google Speech Recognition API to transcribe the audio signal into text and the result is returned to MATLAB for further processing.

OUTPUT HANDLING:

MATLAB displays the transcription result and additional information, such as audio length and processing time, in the console window; that output is also saved to a text file. Any errors encountered during the speech-to-text conversion process (if any) are also displayed in the console window.



```
Transcription:
this is a test for my project I'm testing speech to text I want to see if it can be transcribed and converted to te

Transcription saved to transcribed.txt


Additional Information:
Audio Length: 39.42 seconds
Sampling Rate: 44100 Hz
Processing Time: 25.21 seconds
Additional information saved to transcribed.txt
>>
```

Figure 5: Transcribed output displayed on console.

Result / Conclusion

This implementation transcribes spoken words into text with significant accuracy, demonstrating the capability of integrating Python and MATLAB for speech recognition tasks. The transcription accuracy and processing time are influenced by the quality of the audio input and the efficiency of the underlying speech recognition model. The system also provides a detailed output that includes the transcription and additional audio file information, such as the total length and processing time, which are crucial for evaluating system performance.

Works Cited

- https://deepgram.com/learn/best-python-audio-libraries-for-speech-recognition-in-2023
- Zhang, A. (2017). Speech Recognition (Version 3.8) [Software]. Available from https://github.com/Uberi/speech_recognition#readme.
- Mohan, V. (2013). Analysis and Synthesis of Speech using MATLAB‖. International Journal of advancements in Research and Technology, 2, 373-382.
- Amos, D. (2016). The ultimate guide to speech recognition with python. Real Python.
- Anggraeni, D., Sanjaya, W. S. M., Nurasyidiek, M. Y. S., & Munawwaroh, M. (2018). The implementation of speech recognition using mel-frequency cepstrum coefficients (MFCC) and support vector machine (SVM) method based on python to control robot arm. In IOP Conference Series: Materials Science and Engineering (Vol. 288, No. 1, p. 012042). IOP Publishing.

APPENDIX

MATLAB SCRIPT.

```
clear all; clc;
%-------------------------------------------------------------------------------------------
% Browse for Python file (.py extension specified).
[pythonScript, pythonScriptPath] = uigetfile('*.py', 'Select the Python script');
if pythonScript == 0
    disp('No Python script selected. Exiting program now...');
    return;
end
%-------------------------------------------------------------------------------------------
% Browse for audio file (using wav file for this implementation).
[audioFile, audioPath] = uigetfile('*.wav', 'Select an audio file');
if audioFile == 0
    disp('No audio file selected. Exiting...');
    return;
end
%-------------------------------------------------------------------------------------------
% Get audio file information.
audioFullPath = fullfile(audioPath, audioFile);
audioInfo = audioinfo(audioFullPath);
audioLengthSeconds = audioInfo.TotalSamples / audioInfo.SampleRate;
%-------------------------------------------------------------------------------------------
% Read the audio file and plot the signal waveform.
[y, Fs] = audioread(audioFullPath);

figure;
subplot(2,1,1);
plot((0:length(y)-1)/Fs, y);
xlabel('Time (s)');ylabel('Amplitude');grid on;
title('Audio Signal Waveform');
%-------------------------------------------------------------------------------------------
% Call Python script to transcribe audio and measure processing time.
tic;
[status, result] = system(['python "', fullfile(pythonScriptPath, pythonScript), '" "',
audioFullPath, '"']);
processingTime = toc;
%-------------------------------------------------------------------------------------------
% Display transcription information.
if status == 0
    disp('Transcription:');
    disp(result);

    % Plot the spectrogram.
    subplot(2,1,2);
    spectrogram(y(:,1), hamming(256), 128, 256, Fs, 'yaxis', 'centered');
    title('Spectrogram');
%-------------------------------------------------------------------------------------------
    % Save transcription output to a text file.
    outputFile = 'transcribed.txt';
    fid = fopen(outputFile, 'w');
    fprintf(fid, '%s', result);
    fclose(fid);
    disp(['Transcription saved to ' outputFile]);
%-------------------------------------------------------------------------------------------
    % Display additional information on console and save to text file.
    additionalInfo = sprintf('\n\nAdditional Information:\nAudio Length: %.2f seconds\nSampling
Rate: %d Hz\nProcessing Time: %.2f seconds', audioLengthSeconds, audioInfo.SampleRate,
processingTime);
    disp(additionalInfo);
    fid = fopen(outputFile, 'a');
    fprintf(fid, '%s', additionalInfo);
    fclose(fid);
    disp(['Additional information saved to ' outputFile]);
else
    disp('Error occurred during speech-to-text conversion:');
    disp(result);
end
%-------------------------------------------------------------------------------------------
```

PYTHON SCRIPT.

```
import sys
import speech_recognition as sr

def transcribe_audio(audio_file):
  recognizer = sr.Recognizer()
  with sr.AudioFile(audio_file) as source:
    audio_data = recognizer.record(source)
  try:
    transcript = recognizer.recognize_google(audio_data)
    return transcript
  except sr.UnknownValueError:
    return "Speech recognition could not understand audio"
  except sr.RequestError as e:
    return f"Could not request results from Google Speech Recognition service; {e}"

# Example usage
if __name__ == "__main__":
  if len(sys.argv) < 2:
    print("Usage: python speech_recognition.py <audio_file>")
    sys.exit(1)
  audio_file = sys.argv[1]
  transcript = transcribe_audio(audio_file)
  print(transcript)
```