DeepSpech speech-to-text model

PYTHON VERSION IMPORTANT INFORMATION:

Please ensure you are using a Python environment between versions 3.5 and 3.9, as the deepspeech library is not compatible with Python 3.10 or later versions.

Introduction

This report delves into a Python script specifically designed to process audio files and convert them into textual format using DeepSpeech, a cutting-edge speech-to-text model.

Code explanation

This Python script is designed for advanced audio processing and speech recognition, tailored to work with multiple languages. It starts by importing necessary libraries such as `scipy`, `wave`, `math`, `librosa`, `numpy`, and `deepspeech`, each playing a specific role in handling audio data and implementing the DeepSpeech model for speech-to-text conversion.

```python
import scipy
import wave
import math
import librosa
import numpy as np
from scipy.io import wavfile
import scipy.signal
import deepspeech
```

Initially, the script displays a multilingual menu, prompting users to select their preferred language (Spanish, English, or Italian). It accepts various input forms for each language and uses a validation loop to ensure the user selects a valid language. Once a language is selected, the script sets up the DeepSpeech model accordingly, loading the appropriate language model files (.pbmm and .scorer) and configuring it with specific parameters like the scorer's alpha and beta values and the beam width. These configurations are vital for optimizing the model's speech recognition accuracy.

```python
# Loop to ensure valid language input
while not language_valid:
    user_language = input("Español(ES), English(EN), Italiana(it) : ")
    try:
        # Normalize user input to lower case for easier comparison
        user_language = user_language.lower()

        # Check for different ways to write the language and set up the corresponding language model
        if user_language in ["en", "eng", "english", "ingles", "inglese"]:
            language = "ENG"
            path_of_the_model = 'language_models/EN/deepspeech-0.9.3-models.pbmm'
            language_models = deepspeech.Model(path_of_the_model)
            path_scorer = 'language_models/EN/deepspeech-0.9.3-models.scorer'
            language_valid = True

        elif user_language in ["es", "esp", "español", "spanish", "espanol", "spagnola"]:
            language = "ESP"
            path_of_the_model = 'language_models/ES/output_graph_es.pbmm'
            language_models = deepspeech.Model(path_of_the_model)
            path_scorer = 'language_models/ES/kenlm_es.scorer'
            language_valid = True

        elif user_language in ["it", "ita", "italiana", "italiano", "italian"]:
            language = "ITA"
            path_of_the_model = 'language_models/IT/output_graph_it.pbmm'
            language_models = deepspeech.Model(path_of_the_model)
            path_scorer = 'language_models/IT/kenlm_it.scorer'
            language_valid = True

        # Prompt for correct input if invalid language is entered
        if not language_valid:
            print("Please enter a valid language ********** Por Favor Introduzca un idioma válido
```

The script then defines paths to the audio files based on the selected language, preparing to process language-specific audio data. It includes a function `bandpass_filter`, which applies a bandpass filter to the audio data, a crucial step in enhancing audio quality by allowing only a specific range of frequencies to pass through.

```
# Function to apply a bandpass filter to audio data
def bandpass_filter(data, lowcut, highcut, fs, order=5):
    nyquist = 0.5 * fs  # Nyquist frequency is half the sampling rate
    low = lowcut / nyquist  # Normalize low cutoff frequency
    high = highcut / nyquist  # Normalize high cutoff frequency
    b, a = scipy.signal.butter(order, [low, high], btype='band')  # Create bandpass filter
    y = scipy.signal.lfilter(b, a, data)  # Apply filter to data
    return y
```

The solution of the noise environment:

The core functionality of the script revolves around processing each audio file for the chosen language. It involves loading a crowd ambience audio file for noise reduction. The script performs the Short-Time Fourier Transform (STFT) on this ambience audio to create a noise profile. This profile is used to reduce background noise in the actual audio files. Each audio file undergoes noise reduction, filtering, and conversion back into an audio time series. The processed audio is then saved as a new file.

```
g = 1
crowd_ambience_audio = 'Ex4_audio_files/EN/crowd_ambience.wav'
ats, y_samping_rate = librosa.load(crowd_ambience_audio, sr=ideal_sample_rate)
# Perform Short-Time Fourier Transform (STFT) on the ambience audio
STFT_audio = librosa.stft(ats)
magnitud_spec = np.abs(STFT_audio)
avg_mag_spec_over_time = np.mean(magnitud_spec, axis=1)
```

Subsequently, the script applies a low-pass filter to the noise-reduced audio. The parameters of this filter vary depending on the selected language. The filtered audio is prepared for the DeepSpeech model, which performs speech-to-text conversion. The recognized text is stored in a list for each processed audio file, the Spanish model get a better results with 5000 hz.

```
# Write the processed audio to a new file
scipy.io.wavfile.write("Ex4_audio_files/mywav_reduced_noise" + str(g) + ".wav",

# perform low pass filter on reduced crowd noise file
freq_sampling_rate, data = wavfile.read("Ex4_audio_files/mywav_reduced_noise" +

# different cutoff frequency for different languages, explanation in report
if  language == "ENG"  or language == "ITA":
    cut_off_frequency = 3000.0
if language == "ESP"  :
    cut_off_frequency = 5000.0
```

```python
if language == "ENG":
    # English transcriptions
    transcription_of_text = [
                            "I have lost my parents.", #1
                            "Please, I have lost my suitcase.", #2
                            "Where is the check-in desk?", #3
                            "Where are the restaurants and shops?", #4
                            "What time is my plane?", #4
                            "Are you running late", #5
                            ]
if language == "ESP":
    # Spanish transcriptions
    transcription_of_text = [
                            "He perdido a mis padres.", #1dd
                            "Por favor, he perdido mi maleta.", #2dd
                            "¿Dónde están los mostradores?", #3 ddd
                            "¿Dónde están los restaurantes y las tiendas?", #4
                            "¿A qué hora es mi avión?" #5

                            ]
if language == "ITA":
    # Italian transcriptions
    transcription_of_text = [
                            "Ho perso i miei genitori.", #1
                            "Per favore, ho perso la mia valigia.", #2
                            "Dove e' il bancone?", #3
                            "Dove sono i ristoranti e i negozi?", #4
                            "A che ora e' il mio aereo?"#5
                            ]
```

The script also includes predefined transcriptions for each audio file in the respective languages. It normalizes these transcriptions by removing punctuation, standardizing text format, and converting to lowercase, preparing them for comparison with the recognized text.

Finally, the script evaluates the accuracy of speech recognition. It compares the recognized text with the corresponding transcription, calculating errors (substitutions, deletions, and insertions) and the Word Error Rate (WER) for each transcript. The overall WER across all transcripts is then calculated and displayed, providing a quantitative measure of the speech recognition system's performance.

```python
for i, text in enumerate(converted_texts):
    print("Recognised text: " + text)
    print("Transcript text: " + transcription_of_text[i])

    # Split the recognized and transcribed texts into words
    list_of_text = text.split(" ")
    list_of_transcript_text = transcription_of_text[i].split(" ")

    # Calculate substitution and deletion errors
    # Substitution errors occur when words in the transcription don't match the recognized text
    # Deletion errors occur when words in the transcription are missing in the recognized text
    substitution_and_deletion_errors = len(list_of_transcript_text) - len(set(list_of_text).intersection(list_of_transcript_text))

    # Calculate insertion errors
    # Insertion errors occur when the recognized text contains extra words not in the transcription
    difference_length = len(list_of_transcript_text) - len(list_of_text)
    insertion_errors = abs(difference_length) if difference_length < 0 else 0

    # Update the total errors and word count
    sum_of_sub_and_del = substitution_and_deletion_errors + insertion_errors
    N = len(list_of_transcript_text)
    print("Number of errors: " + str(sum_of_sub_and_del))
    print("Total number of words: " + str(N))

    # Calculate the WER for the current transcript and update totals
    actual_WER = sum_of_sub_and_del / N * 100
    print("WER for current transcript: " + str(actual_WER) + "%")
    total_SDI += sum_of_sub_and_del
    total_N += N

# Calculate the overall WER across all transcripts
total_WER = total_SDI / total_N * 100
print("Overall " + language + " WER: " + str(total_WER) + "%")
```

For more details about the code, please check the comments inside the code.

Asserting the results of the different languages

English Results

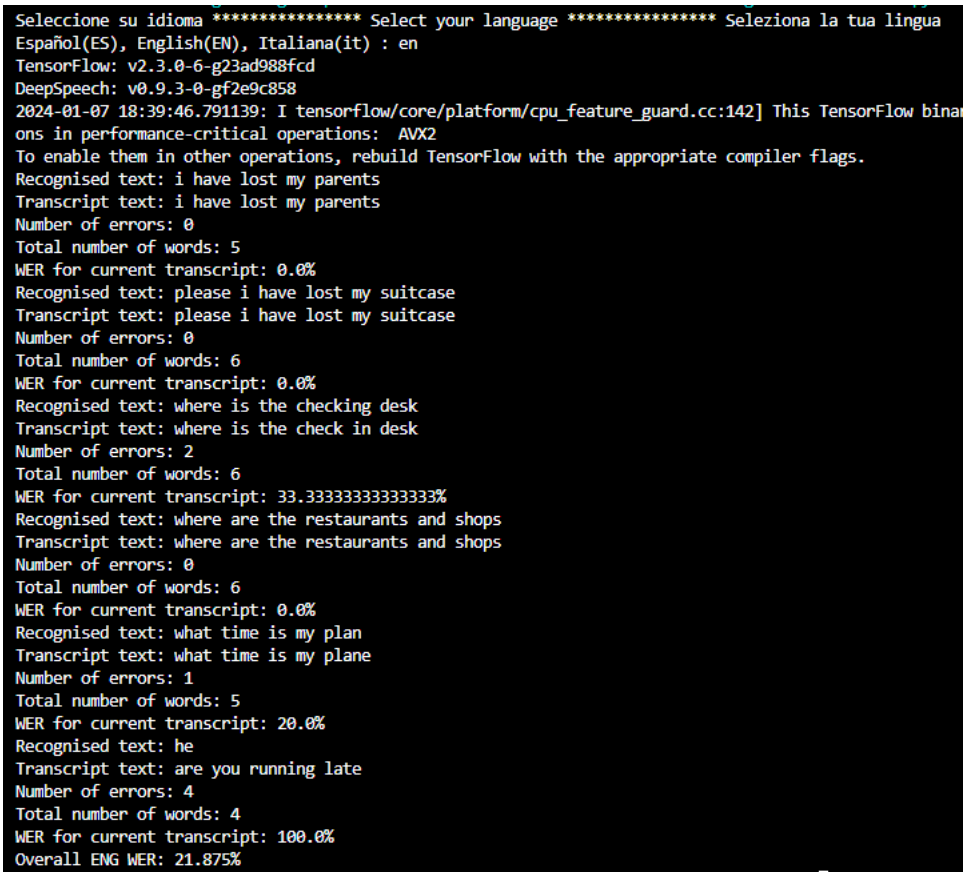
Here is a table with the results in English:

Language	File	Recognised Text	Transcript Text	Number of Errors	WER
English	parents_es.wav	i have lost my parents	i have lost my parents	0	0
English	suitcase_es.wav	please i have lost my suitcase	please i have lost my suitcase	0	0
English	checkin_es.wav	where is the checking desk	where is the check in desk	2	33.33
English	where_es.wav	where are the restaurants and shops	where are the restaurants and shops	0	0
English	what_time_es.wav	what time is my plan	what time is my plane	1	20
English	late.wav	he	are you running late	4	100
				overall WER	21.88%

The performance of the English ASR model reflects a fairly good understanding of the language, with an overall Word Error Rate (WER) of 21.875%. The model demonstrates perfect recognition in scenarios with clear and well-articulated speech, as shown by the 0% WER in two of the transcripts. However, the accuracy decreases notably with the sample I recorded, maybe that's an accent problem or a problem with the" noise" sound I add to the file. This problem should be resolve since in an airport there is a lot of different accents

Overrall the source for the English model is different and probably better. That is why it gets better results

English screenshot of the console

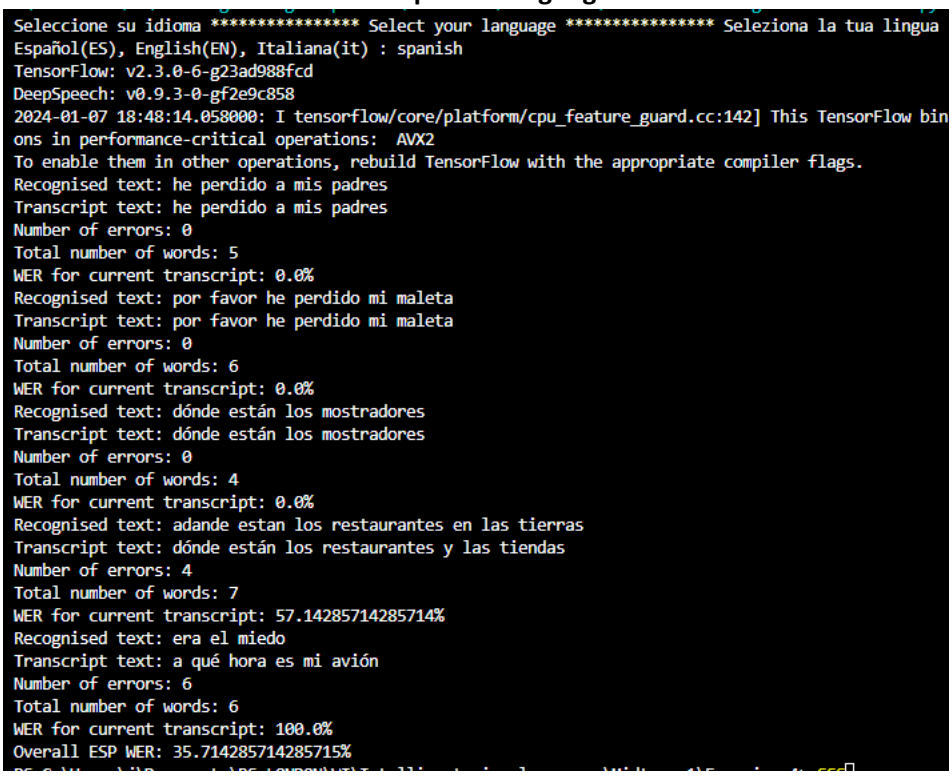


Spanish Results

Language	File	Recognised Text	Transcript Text	Number of Errors	WER
Spanish	parents_es.wav	he perdido a mis padres	he perdido a mis padres	0	0
Spanish	suitcase_es.wav	por favor he perdido mi maleta	por favor he perdido mi maleta	0	0
Spanish	checkin_es.wav	dónde están los mostradores	dónde están los mostradores	0	0
Spanish	where_es.wav	adande estan los restaurantes en las tierras	dónde están los restaurantes y las tiendas	4	57.14
Spanish	what_time_es.wav	era el miedo	a qué hora es mi avión	6	100
				overall WER	35.71%

The Spanish ASR model exhibits a strong ability to recognize and transcribe clearly spoken Spanish, as evidenced by the 0% WER in several transcripts. However, the overall WER of 35.71% suggests there are significant challenges when the audio includes more complex phrases or when the speech is not articulated clearly. Particularly, the model struggles with sentences that deviate from standard pronunciation or have background noise, leading to substantial errors in transcription.

Screenshot of the console in the Spanish language



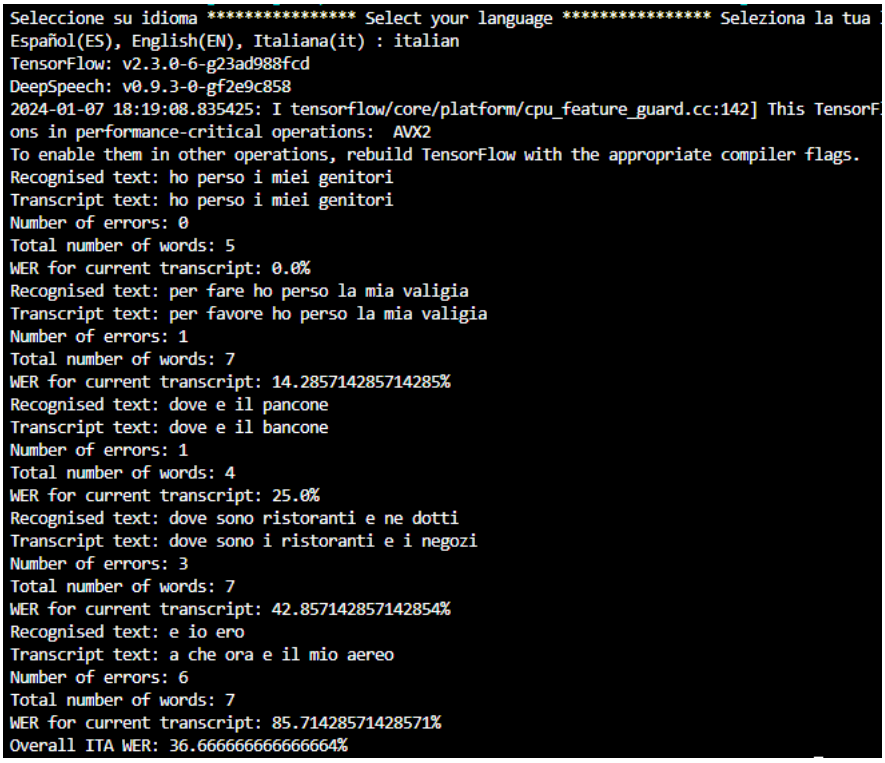
Italian Results

Here is a table with the results in Italian:

Language	File	Recognised Text	Transcript Text	Number of Errors	WER
Italian	parents_es.wav	ho perso i miei genitori	ho perso i miei genitori	0	0
Italian	suitcase_es.wav	per fare ho perso la mia valigia	per favore ho perso la mia valigia	1	14.29
Italian	checkin_es.wav	dove e il pancone	dove e il bancone	1	25
Italian	where_es.wav	dove sono ristoranti e ne dotti	dove sono i ristoranti e i negozi	3	42.86
Italian	what_time_es.wav	e io ero	a che ora e il mio aereo	6	85.71
				overall WER	36.67%

The evaluation of the Italian ASR model yields mixed results. While the model demonstrates the ability to accurately transcribe audio with no background noise (0% WER for "ho perso i miei genitori"), there is a notable increase in the Word Error Rate (WER) for more complex sentences. The overall Word Error Rate stands at 36.67%, suggesting that while the model can handle clear audio efficiently, its performance degrades with more complex phrases or possibly with background noise.

Screenshot of the console:



Conclusion of the analysis of other ASR systems

After evaluating ASR systems like DeepSpeech, Google Cloud Speech-to-Text, and IBM Watson, we find that Google stands out for its accuracy and ease of use, while DeepSpeech offers customization at the cost of greater resource needs. For environments like airports with varied accents, Google's ASR system is recommended for its comprehensive performance and ability to handle diverse linguistic challenges effectively.

Conclusion

The Italian and Spanish ASR models show promise with their ability to accurately transcribe clear speech, despite higher WERs in complex scenarios. To enhance their performance, expanding the training datasets to include a variety of speech patterns and noisy backgrounds will be key. Overall, the models are performing well, and with focused improvements, they can become robust tools for effective speech recognition in real-world applications.