Midterm Exercise 4 Report-Joseph Shen Fan

Implementation

The application is based on the Python Jupyter Framework by importing Mozilla DeepSpeech library to create an offline automatic speech recognition (ASR) system. Importing DeepSpeech allows the use of speech models to train and recognize speech patterns across various languages. JiWER is another python library that calculates the word error rate(WER) of the application, it runs a mathematical formula in the background to find the error rate of the application when compared to the transcript provided.

```
########## RUN THIS TO IMPORT libraries ###############
#!pip install deepspeech // If doing in lab environment
#!pip install librosa // If doing in lab environment
# jiwer is a package to approximate the Word Error Rate
# !pip install jiwer
# !pip install noisereduce
# !pip install pydub
####################################################
#Import DeepSpeech Libraries
from deepspeech import Model, version
from jiwer import wer
```

Language Selection

The application is capable of switching across multiple languages via a dropdown box. The function contains 3 language states plus one empty state. This helps to reduce clutter and modularize the code into separate functions each in their own environment.

[Language Selector] - Choose from Dropdown box below

```
# Language Selector
Lang_dropDown = ipywidgets.Dropdown(options=[('None', 0), ('English', 1), ('Spanish', 2), ('Italian', 3)],
                                value=0,
                                description='Language Select:',
                                disabled=False)
def SelectLanguage(state):
    if state == 1:
        print("English Language Selected. Please wait for loading...")
        English()
        display(df)
    if state == 2:
        print("Spanish Language Selected. Please wait for loading...")
        Spanish()
        display(df)
    if state == 3:
        print("Italian Language Selected. Please wait for loading...")
        Italian()
        display(df)
ipywidgets.interact(SelectLanguage, state=Lang_dropDown );
```

Language ... None

None
English
Spanish
Italian

English Langu

The dropdown box will call the selected language function, these functions are only compatible with their own individual deepspeech language models.

```
def English():
    scorer = "Models/deepspeech-0.9.3-models.scorer"
    model = "Models/deepspeech-0.9.3-models.pbmm"
    result = []
    WER = []
    Total = []

    ds = Model(model)
    ds.enableExternalScorer(scorer)
    desired_sample_rate = ds.sampleRate()
```

Def English() function

```
def Spanish():
    scorer = "Models/kenlm_es.scorer"
    model = "Models/output_graph_es.pbmm"
    result = []
    WER = []
    Total = []

    ds = Model(model)
    ds.enableExternalScorer(scorer)
    desired_sample_rate = ds.sampleRate()
```

Def Spanish() function

```
def Italian():
    scorer = "Models/kenlm_it.scorer"
    model = "Models/output_graph_it.pbmm"
    result = []
    WER = []
    Total = []

    ds = Model(model)
    ds.enableExternalScorer(scorer)
    desired_sample_rate = ds.sampleRate()
```

Def Italian() function

English Language (WER < 25%)

The function iterates all audio files within the **"Ex4_audio_files\EN"** folder into the deepspeech model and append the text result into a list array called 'results'.

```python
# For Loop to iterate every wav file
for i in range(0,len(audio_fileEN)):
    #Get Filename
    filepath = audio_fileEN[i]
    filename = os.path.basename(filepath)
    print(filename)

    audio = lr.load(audio_fileEN[i], sr=desired_sample_rate)[0]
    audio = (audio * 32767).astype(np.int16)  # scale from -1 tc
    res = ds.stt(audio)
    result.append(res)
```

Another for loop will be used to compare the results list to the transcript and get the average WER value for the English language.

```python
#Loop to compare each element in result with transcription
for i in range(0, len(result)):
    #Use wer() to find error rate
    error = wer(result[i],transcription[i])
    #Convert into percentage
    percentage = str(round(error*100)) + '%'
    avg = round(error*100)
    #Print percentage for debugging
    #print(percentage)
    WER.append(percentage)
```

The English speech model seems to pick up speech patterns very effectively with an **8.57%** error rate and therefore does not require any adjustments.

```
The Average WER is: 8.57%
```

	File	Transcription	Result	WER
0	checkin.wav	where is the checkin desk	where is the checking desk	20%
1	parents.wav	i have lost my parents	i had lost my parents	20%
2	suitcase.wav	please i have lost my suitcase	please i have lost my suitcase	0%
3	what_time.wav	what time is my plane	what time is my plan	20%
4	where.wav	where are the restaurants and shops	where are the restaurants and shops	0%
5	your_sentence1.wav	this is my first sentence	this is my first sentence	0%
6	your_sentence2.wav	this is my second sentence	this is my second sentence	0%

your_sentence1.wav and your_sentence2.wav

The two audio files are two short sentences that I personally recorded and save at **"Ex4_audio_files\EN"** to evaluate the effectiveness of the application. It had passed with 0% WER flawlessly as shown below.

5	your_sentence1.wav	this is my first sentence	this is my first sentence	0%
6	your_sentence2.wav	this is my second sentence	this is my second sentence	0%

Spanish Language (WER < 35%)

For the Spanish language when I tried to run the application as it is, it returned a high average WER as shown in the picture below. It is likely due to the noisy background that hinders the applications ability to recognize Spanish as too many voices are stacked on top of each other.

Before Adjustments:

```
The Average WER is: 47.0%
```

	Files	Transcription	Result	WER
0	checkin_es.wav	donde estan los mostradores	adande estan los mostradores	25%
1	parents_es.wav	he perdido a mis padres	he perdido a mis padres	0%
2	suitcase_es.wav	por favor he perdido mi maleta	por favor he perdido mi maleta	0%
3	what_time_es.wav	a que hora es mi avion	ahora es miedo	167%
4	where_es.wav	donde estan los restaurantes y las tiendas	adande estan los restaurantes en las tierras	43%

Noisy Environment removal

First, import noisereduce library into jupyter, the function removes any background noise/white noise from the audio file and creates a new audio file that filters out unwanted noise from background.

```
# Start noise reduction
rate, data = wavfile.read(audio_fileES[i])
# perform noise reduction
reduced_noise = nr.reduce_noise(y=data, sr=rate)
wavfile.write(NewPath, rate, reduced_noise)
```

Second, apply low-pass filter to remove high frequency sound from the new audio clip. Doing so isolates the audio file to only clean audible voices and removes any noise for the model to compile.

```
spectrum2.low_pass(4000)
spectrum.low_pass(4000)
filtered_wave = spectrum.
filtered_wave2 = spectrum
filtered_wave.unbias()
filtered_wave.normalize()
```

The picture below is the final result of all the process and the application has successfully reduced the **WER to 32.8% average.**

After Adjustments:

```
The Average WER is: 32.8%
```

	Files	Transcription	Result	WER
0	checkin_es.wav	donde estan los mostradores	adande estan los mostradores	25%
1	parents_es.wav	he perdido a mis padres	he perdido a mis padres	0%
2	suitcase_es.wav	por favor he perdido mi maleta	por favor he perdido mi maleta	0%
3	what_time_es.wav	a que hora es mi avion	la cara es miedo	125%
4	where_es.wav	donde estan los restaurantes y las tiendas	adande estan los restaurantes y las tiendas	14%

Italian Language (WER < 35%)

The Italian language has the highest average WER with a staggering 167.8%, what_time_it.wav file is even getting a 700% error rate.

Before Adjustments:

```
The Average WER is: 167.8%
```

	Files	Transcription	Result	WER
0	checkin_it.wav	dove e il bancone	dove e il pancone	25%
1	parents_it.wav	ho perso i miei genitori	perso i miei genitori	25%
2	suitcase_it.wav	per favore ho perso la mia valigia	per fare ho perso la mia valigia	14%
3	what_time_it.wav	a che ora e il mio aereo	nero	700%
4	where_it.wav	dove sono i ristoranti e i negozi	dove sono ristoranti negozi	75%

The previous noise-reduce method is not very effective in this language as after applying the noise-reduction the WER does not seem to have any improvements. Therefore, the best approach is to add silence to the audio files, this gives the language model more time to analyze speech patterns and improves recognition algorithm.

Adding 1 second padding to audio

The first for loop iterate the "**Ex4_audio_files\IT**" and 1 second segment padding to all the audio files in the folder.

```
###### ADDING 1second silence in front #######
#Load every IT audio and assign variable
loadAudio = audio_fileIT[i]
#read wav file to an audio segment
SegmentLoad = AudioSegment.from_wav(loadAudio)
#Add above two audio segments
audio_Out= one_sec_segment + SegmentLoad
```

The new padded audio will be saved into a new folder under a new name "SilenceAdded_*.wav". The second for loop iterates the new folder and uses the audiofiles as input for the deepspeech model to compile. This method significantly reduced the error rate for what_time_it.wav file from a 700% to 33% and successfully improves the WER to an acceptable **average of 26.6%.**

After Adjustments:

```
The Average WER is: 26.6%
```

	Files	Transcription	Result	WER
0	checkin_it.wav	dove e il bancone	dove e il pancone	25%
1	parents_it.wav	ho perso i miei genitori	ho perso i miei genitori	0%
2	suitcase_it.wav	per favore ho perso la mia valigia	per favore ho perso la mia valigia	0%
3	what_time_it.wav	a che ora e il mio aereo	a che ora il mio ero	33%
4	where_it.wav	dove sono i ristoranti e i negozi	dove sono ristoranti negozi	75%