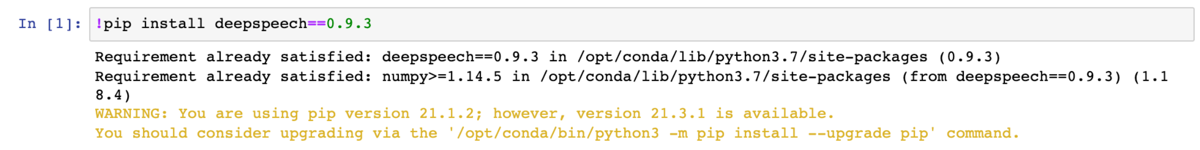
Exercise 4

Speech to Text basically takes an audio data and converts it into a machine readable form. So basically it takes an audio data as an input and gives out text data. deep speech is used to accomplish speech to text and it's an open source voice recognition or speech to text system that uses a neural network to convert speech spectrograms into text transcripts.

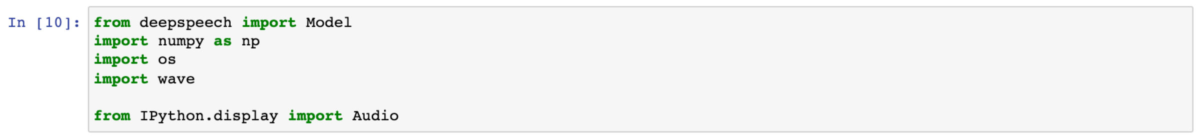
Deep speech is developed using an single end-to-end deep learning deep learning framework. There are two models typically in a speech to text system. There's an acoustic model and a language model. The acoustic model is an end-to-end deep learning system and there's a language model which is kind of used to increase the accuracy of the transcription output.

Firstly we need to install deepspeech version 0.9.3 (latest version). (Release DeepSpeech 0.9.3 · mozilla/DeepSpeech, 2020)



```
In [1]: !pip install deepspeech==0.9.3

Requirement already satisfied: deepspeech==0.9.3 in /opt/conda/lib/python3.7/site-packages (0.9.3)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/lib/python3.7/site-packages (from deepspeech==0.9.3) (1.1
8.4)
WARNING: You are using pip version 21.1.2; however, version 21.3.1 is available.
You should consider upgrading via the '/opt/conda/bin/python3 -m pip install --upgrade pip' command.
```

Import deepspeech model, import numpy to buffer the incoming audio file, import wave file to read wav audio format, IPython.display to play the audio in the jupyter notebook itself.

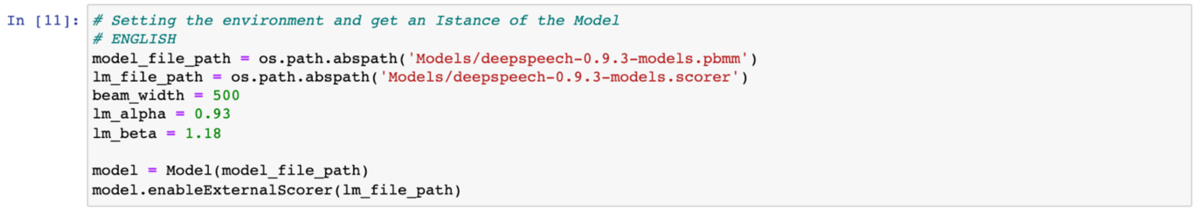


```
In [10]: from deepspeech import Model
         import numpy as np
         import os
         import wave

         from IPython.display import Audio
```

Now I am adding the language model file path (acoustic model) which is the .pbmm file and the language file path (language model) which is the .scorer file.

We set the optimal parameter for language model alpha and beta based on the github repository itself, that works best for transcription purposes. (Release DeepSpeech 0.9.3 · mozilla/DeepSpeech, 2020)

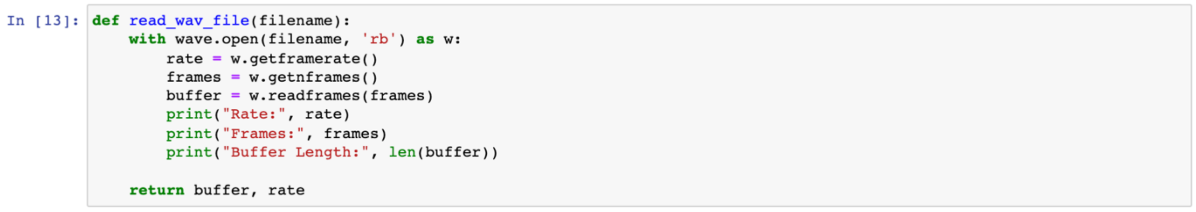
The other parameter is beam width. So beam width is basically the idea is in which how many different word sequences that needs to be evaluated by your acoustic model. The more you give beam width the better the acoustic model conversion will be, but it comes with the cost as well, so you need to kind of balance it and see what works best for your case. Afterwards, you take the model and load the model file path, which is the acoustic model, and then setting up the language model as well. So when an audio file comes in, it goes to acoustic model. The acoustic model converts the speech spectrogram into text transcript. However, the output may not be perfect. And therefore, the language model refines the output to get a better text prediction output.



```
In [11]: # Setting the environment and get an Istance of the Model
         # ENGLISH
         model_file_path = os.path.abspath('Models/deepspeech-0.9.3-models.pbmm')
         lm_file_path = os.path.abspath('Models/deepspeech-0.9.3-models.scorer')
         beam_width = 500
         lm_alpha = 0.93
         lm_beta = 1.18

         model = Model(model_file_path)
         model.enableExternalScorer(lm_file_path)
```

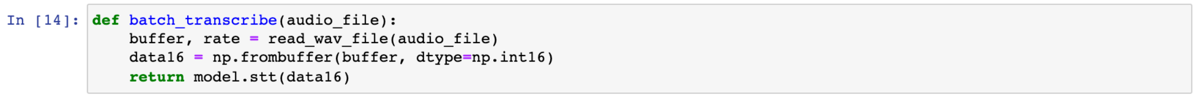
For English,
deepspeech-0.9.3-models.pbmm
deepspeech-0.9.3-models.scorer

I create a function, which basically takes in a .wav file name. Afterwards, I am opening the .wav file name. I will be getting the frame rate for this audio file, I will also be checking how many frames are there as well as the buffer length. Then I will return the buffer and rate.



```
In [13]: def read_wav_file(filename):
             with wave.open(filename, 'rb') as w:
                 rate = w.getframerate()
                 frames = w.getnframes()
                 buffer = w.readframes(frames)
                 print("Rate:", rate)
                 print("Frames:", frames)
                 print("Buffer Length:", len(buffer))

             return buffer, rate
```

I create another function, which is a transcription function. I will be getting the buffer and rate, and will be using the numpy from buffer to get the bytes data. Then I will be calling model.stt which is speech to text and will be getting the transcript output.



```
In [14]: def batch_transcribe(audio_file):
             buffer, rate = read_wav_file(audio_file)
             data16 = np.frombuffer(buffer, dtype=np.int16)
             return model.stt(data16)
```

This is to retrieve all the English audio files.

```
In [15]: #english
         checkinwav = os.path.abspath("Audio_Files/EN/checkin.wav")
         parentswav = os.path.abspath("Audio_Files/EN/parents.wav")
         suitcasewav = os.path.abspath("Audio_Files/EN/suitcase.wav")
         timewav = os.path.abspath("Audio_Files/EN/what_time.wav")
         wherewav = os.path.abspath("Audio_Files/EN/where.wav")
```

Afterwards, we are able to load the audio file and play it. The batch_transcribe function will help to transcribe the audio file as long as the rate, frame rate and the buffer length of that particular file. For example, in this case, the transcribed text of checkinwav is 'where is the checking desk" with a rate of 16000, frames of 44211 and buffer length of 88422.

```
In [16]: Audio(checkinwav)

Out[16]: 0:00 / 0:02

In [17]: batch_transcribe(checkinwav)

         Rate: 16000
         Frames: 44211
         Buffer Length: 88422

Out[17]: 'where is the checking desk'
```

Since the audio files were recorded in a noisy environment, I have implemented a noise reduction function. (GitHub - timsainb/noisereduce: Noise reduction in python using spectral gating (speech, bioacoustics, audio, time-domain signals), 2021)

I went to install 3 packages, noisereduce, spicy and wave.

```
In [2]: pip install noisereduce

        Requirement already satisfied: noisereduce in /opt/conda/lib/python3.7/site-packages (2.0.0)
        Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from noisereduce) (1.4.1)
        Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-packages (from noisereduce) (3.2.1)
        Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from noisereduce) (4.45.0)
        Requirement already satisfied: librosa in /opt/conda/lib/python3.7/site-packages (from noisereduce) (0.8.1)
        Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages (from noisereduce) (0.14.1)
        Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from noisereduce) (1.18.4)
        Requirement already satisfied: soundfile>=0.10.2 in /opt/conda/lib/python3.7/site-packages (from librosa->noisereduc
        e) (0.10.3.post1)

In [3]: pip install scipy

        Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (1.4.1)
        Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-packages (from scipy) (1.18.4)
        WARNING: You are using pip version 21.1.2; however, version 21.3.1 is available.
        You should consider upgrading via the '/opt/conda/bin/python -m pip install --upgrade pip' command.
        Note: you may need to restart the kernel to use updated packages.

In [4]: pip install wave

        Requirement already satisfied: wave in /opt/conda/lib/python3.7/site-packages (0.0.2)
        WARNING: You are using pip version 21.1.2; however, version 21.3.1 is available.
        You should consider upgrading via the '/opt/conda/bin/python -m pip install --upgrade pip' command.
        Note: you may need to restart the kernel to use updated packages.
```

Afterwards, I loaded the child wav files and then applied the noise reduction filter into it. This creates brand new audio files ending with _reduced_noise.wav, whereby this audio files are the ones which have undergone noise reduction.

```
In [5]: from scipy.io import wavfile
        import noisereduce as nr
        # to load data
        rate, data = wavfile.read("Audio_Files/EN/checkin_child.wav")
        # to perform noise reduction
        reduced_noise = nr.reduce_noise(y=data, sr=rate)
        wavfile.write("Audio_Files/EN/checkin_child_reduced_noise.wav", rate, reduced_noise)
```

```
In [6]: # to load data
        rate, data = wavfile.read("Audio_Files/EN/parents_child.wav")
        # to perform noise reduction
        reduced_noise = nr.reduce_noise(y=data, sr=rate)
        wavfile.write("Audio_Files/EN/parents_child_reduced_noise.wav", rate, reduced_noise)
```

```
In [7]: rate, data = wavfile.read("Audio_Files/EN/suitcase_child.wav")
        # to perform noise reduction
        reduced_noise = nr.reduce_noise(y=data, sr=rate)
        wavfile.write("Audio_Files/EN/suitcase_child_reduced_noise.wav", rate, reduced_noise)
```

```
In [8]: rate, data = wavfile.read("Audio_Files/EN/what_time_child.wav")
        # to perform noise reduction
        reduced_noise = nr.reduce_noise(y=data, sr=rate)
        wavfile.write("Audio_Files/EN/what_time_child_reduced_noise.wav", rate, reduced_noise)
```

```
In [9]: rate, data = wavfile.read("Audio_Files/EN/where_child.wav")
        # to perform noise reduction
        reduced_noise = nr.reduce_noise(y=data, sr=rate)
        wavfile.write("Audio_Files/EN/where_child_reduced_noise.wav", rate, reduced_noise)
```

I have implemented the noise reduction filter with hopes that the audio will be clearer and this will improve the transcribed output.

Now, we need to calculate the Word Error Rate (WER).

I will be using the jiwer package to aid my WER calculations.
(GitHub - jitsi/jiwer: Evaluate your speech-to-text system with similarity measures such as word error rate (WER), 2021)

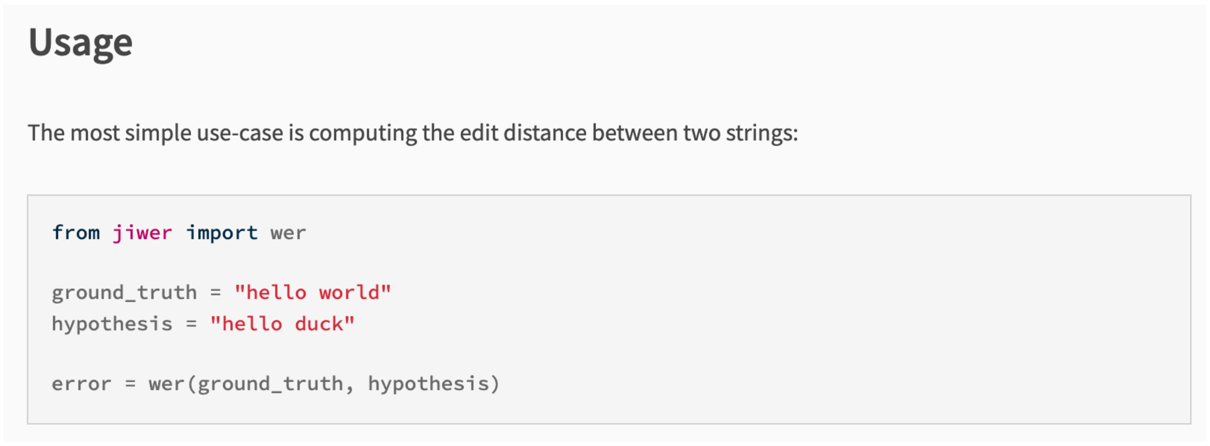
Firstly, what I need to do is to install jiwer.

```
In [27]: pip install jiwer

         Requirement already satisfied: jiwer in /opt/conda/lib/python3.7/site-packages (2.3.0)
         Requirement already satisfied: python-Levenshtein==0.12.2 in /opt/conda/lib/python3.7/site-packages (from jiwer) (0.1
         2.2)
         Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages (from python-Levenshtein==0.12.2-
         >jiwer) (46.1.3.post20200325)
         WARNING: You are using pip version 21.1.2; however, version 21.3.1 is available.
         You should consider upgrading via the '/opt/conda/bin/python -m pip install --upgrade pip' command.
         Note: you may need to restart the kernel to use updated packages.
```

Secondly, we need to load all the audio files into their respective variables.

```
In [26]: checkinwav = os.path.abspath("Audio_Files/EN/checkin.wav")
         checkinchildwav = os.path.abspath("Audio_Files/EN/checkin_child_reduced_noise.wav")

         parentswav = os.path.abspath("Audio_Files/EN/parents.wav")
         parentschildwav = os.path.abspath("Audio_Files/EN/parents_child_reduced_noise.wav")

         suitcasewav = os.path.abspath("Audio_Files/EN/suitcase.wav")
         suitcasechildwav = os.path.abspath("Audio_Files/EN/suitcase_child_reduced_noise.wav")

         timewav = os.path.abspath("Audio_Files/EN/what_time.wav")
         timechildwav = os.path.abspath("Audio_Files/EN/what_time_child_reduced_noise.wav")

         wherewav = os.path.abspath("Audio_Files/EN/where.wav")
         wherechildwav = os.path.abspath("Audio_Files/EN/where_child_reduced_noise.wav")
```

With reference to the image below, the ground_truth will be the actual 100% accurate text of the audio file. Meanwhile, hypothesis will be the transcribed audio file done by both the acoustic and language model.



Usage

The most simple use-case is computing the edit distance between two strings:

```python
from jiwer import wer

ground_truth = "hello world"
hypothesis = "hello duck"

error = wer(ground_truth, hypothesis)
```

So here we are transcribing both the original audio files and the child audio files.

```
In [28]: # transcribing text
         # checkin text
         checkintext = batch_transcribe(checkinwav)
         checkinchildtext = batch_transcribe(checkinchildwav)
         print("English checkin transcription:", checkintext)
         print("English checkin_child transcription:", checkinchildtext)

         # parents text
         parentstext = batch_transcribe(parentswav)
         parentschildtext = batch_transcribe(parentschildwav)
         print("English parents transcription:", parentstext)
         print("English parents_child transcription:", parentschildtext)

         # suitcase text
         suitcasetext = batch_transcribe(suitcasewav)
         suitcasechildtext = batch_transcribe(suitcasechildwav)
         print("English suit transcription:", suitcasetext)
         print("English suit_child transcription:", suitcasechildtext)

         # time text
         timetext = batch_transcribe(timewav)
         timechildtext = batch_transcribe(timechildwav)
         print("English time transcription:", timetext)
         print("English time_child transcription:", timechildtext)

         # where text
         wheretext = batch_transcribe(wherewav)
         wherechildtext = batch_transcribe(wherechildwav)
         print("English where transcription:", wheretext)
         print("English where_child transcription:", wherechildtext)
```

And here are the results:

```
Rate: 16000
Frames: 44211
Buffer Length: 88422
Rate: 16000
Frames: 40867
Buffer Length: 81734
English checkin transcription: where is the checking desk
English checkin_child transcription: where is the check in de
Rate: 16000
Frames: 40782
Buffer Length: 81564
Rate: 16000
Frames: 39381
Buffer Length: 78762
English parents transcription: i had lost my parents
English parents_child transcription: i have lost my parents
Rate: 16000
Frames: 45022
Buffer Length: 90044
Rate: 16000
Frames: 52756
Buffer Length: 105512
English suit transcription: please i have lost my suitcase
English suit_child transcription: yes it was my sin
Rate: 16000
Frames: 29536
Buffer Length: 59072
Rate: 16000
Frames: 40124
Buffer Length: 80248
English time transcription: what time is my plan
English time_child transcription: what time is my plan
Rate: 16000
Frames: 42353
Buffer Length: 84706
Rate: 16000
Frames: 46440
Buffer Length: 92880
English where transcription: where are the restaurants and shops
English where_child transcription: were you restrictions
```

Lastly, we calculate the Word Error Rate (WER).
I have performed pre-processing steps like converting it to lower case and removing punctuations.

In [29]:
```
import jiwer
from jiwer import wer

# checkin word error rate
# preprocessing
# lowercase
checkinGroundTruth = jiwer.ToLowerCase() ('Where is the check-in desk?')
# removing punctuation
checkinGroundTruth = jiwer.RemovePunctuation()(checkinGroundTruth)
print(checkinGroundTruth)

checkinwer = wer(checkinGroundTruth, checkintext) * 100
checkinchildwer = wer(checkinGroundTruth, checkinchildtext) * 100
print("English checkin WER:", checkinwer,"%")
print("English checkin_child WER:", checkinchildwer,"%")

# parents word error rate
# preprocessing
# lowercase
parentsGroundTruth = jiwer.ToLowerCase()('I have lost my parents.')
# removing punctuation
parentsGroundTruth = jiwer.RemovePunctuation()(parentsGroundTruth)
print (parentsGroundTruth)

parentswer = wer(parentsGroundTruth, parentstext) * 100
parentschildwer = wer(parentsGroundTruth, parentschildtext) * 100
print("English parents WER:", parentswer,"%")
print("English parents_child WER:", parentschildwer,"%")

# suitcase Word error rate
# preprocessing
# lowercase
suitcaseGroundTruth = jiwer.ToLowerCase()('Please, I have lost my suitcase.')
# removing punctuation
suitcaseGroundTruth = jiwer.RemovePunctuation()(suitcaseGroundTruth)
print(suitcaseGroundTruth)

suitcasewer = wer(suitcaseGroundTruth, suitcasetext) * 100
suitcasechildwer = wer(suitcaseGroundTruth, suitcasechildtext) * 100
print("English suitcase WER:", suitcasewer,"%")
print("English suitcase_child WER:", suitcasechildwer,"%")

# time word error rate
# preprocessing
# lowercase
timeGroundTruth = jiwer.ToLowerCase()('What time is my plane?')
# removing punctuation
timeGroundTruth = jiwer.RemovePunctuation()(timeGroundTruth)
print(timeGroundTruth)

timewer = wer(timeGroundTruth, timetext) * 100
timechildwer = wer(timeGroundTruth, timechildtext) * 100
print("English time WER:", timewer,"%")
print("English time_child WER:",timechildwer,"%")

# where word error rate
# preprocessing
# lowercase
whereGroundTruth = jiwer.ToLowerCase()('Where are the restaurants and shops?')
# removing punctuation
whereGroundTruth = jiwer.RemovePunctuation()(whereGroundTruth)
print(whereGroundTruth)

wherewer = wer(whereGroundTruth, wheretext) * 100
wherechildwer = wer(whereGroundTruth, wherechildtext) * 100
print("English where WER:", wherewer, "%")
print("English where_child WER:", wherechildwer,"%")
```

And below are the results:

```
where is the checkin desk
English checkin WER: 20.0 %
English checkin_child WER: 60.0 %
i have lost my parents
English parents WER: 20.0 %
English parents_child WER: 0.0 %
please i have lost my suitcase
English suitcase WER: 0.0 %
English suitcase_child WER: 83.33333333333334 %
what time is my plane
English time WER: 20.0 %
English time_child WER: 20.0 %
where are the restaurants and shops
English where WER: 0.0 %
English where_child WER: 100.0 %
```

From the results above, we can conclude that the audio files with noise reduction filter (_child) made it worse, as the error rates are higher compared to the original audio files with WER, which are all less than 20%.

Now, moving on to my own recordings. I have recorded two audio files using Audacity. These are the two sentences below which I have recorded.
my_sentence1.wav = ‘I am a boy.’
my_sentence2.wav = ‘Football is fun.’

Loading both my freshly recorded files and assigning them into respective variables.

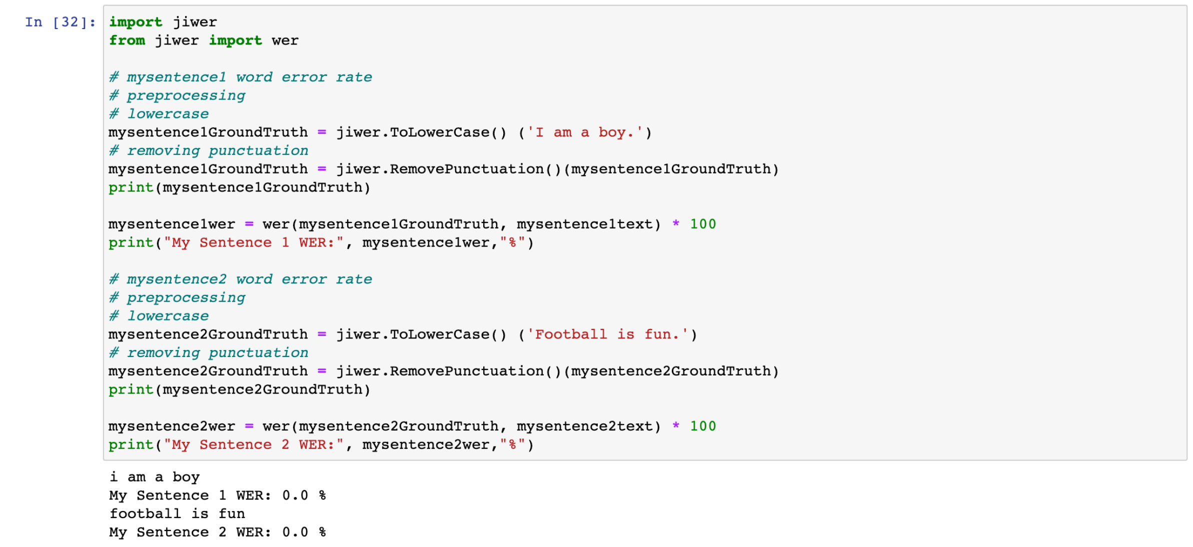
```
In [30]: # my two sentences
mysentence1wav = os.path.abspath("Audio_Files/EN/my_sentence1.wav")
mysentence2wav = os.path.abspath("Audio_Files/EN/my_sentence2.wav")
```

This is to display the rate, frame, buffer length and the transcription of both my recordings. We can see that the transcription output is extremely accurate, which is on point with the groundtruth.

```
In [31]: # transcribing text
# mysentence1 text
mysentence1text = batch_transcribe(mysentence1wav)
print("My Sentence 1 transcription:", mysentence1text)

# mysentence2 text
mysentence2text = batch_transcribe(mysentence2wav)
print("My Sentence 2 transcription:", mysentence2text)
```

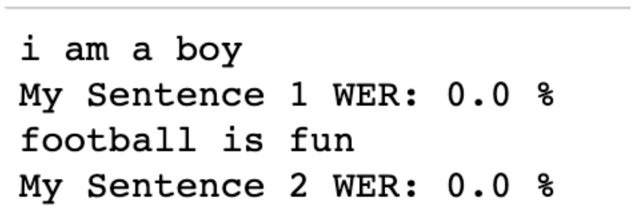
```
Rate: 16000
Frames: 31710
Buffer Length: 63420
My Sentence 1 transcription: i am a boy
Rate: 16000
Frames: 27696
Buffer Length: 55392
My Sentence 2 transcription: football is fun
```

Lastly, we calculate the WER.



```
In [32]: import jiwer
         from jiwer import wer

         # mysentence1 word error rate
         # preprocessing
         # lowercase
         mysentence1GroundTruth = jiwer.ToLowerCase() ('I am a boy.')
         # removing punctuation
         mysentence1GroundTruth = jiwer.RemovePunctuation()(mysentence1GroundTruth)
         print(mysentence1GroundTruth)

         mysentence1wer = wer(mysentence1GroundTruth, mysentence1text) * 100
         print("My Sentence 1 WER:", mysentence1wer,"%")

         # mysentence2 word error rate
         # preprocessing
         # lowercase
         mysentence2GroundTruth = jiwer.ToLowerCase() ('Football is fun.')
         # removing punctuation
         mysentence2GroundTruth = jiwer.RemovePunctuation()(mysentence2GroundTruth)
         print(mysentence2GroundTruth)

         mysentence2wer = wer(mysentence2GroundTruth, mysentence2text) * 100
         print("My Sentence 2 WER:", mysentence2wer,"%")

         i am a boy
         My Sentence 1 WER: 0.0 %
         football is fun
         My Sentence 2 WER: 0.0 %
```

And here are the results:



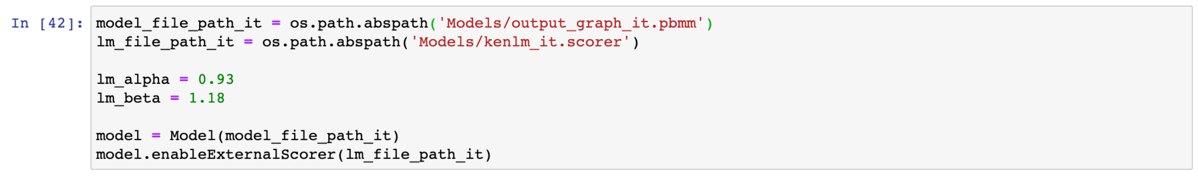
```
i am a boy
My Sentence 1 WER: 0.0 %
football is fun
My Sentence 2 WER: 0.0 %
```

From this results alone, we can conclude that the acoustic and language model have successfully managed to recognise my voice precisely. The WER stands at 0%, which means that it had succeeded to transcribe my recording altogether.

In short, based on the results above, I can safely say that the audio files I have recorded have a much lower WER compared to my client's recording.

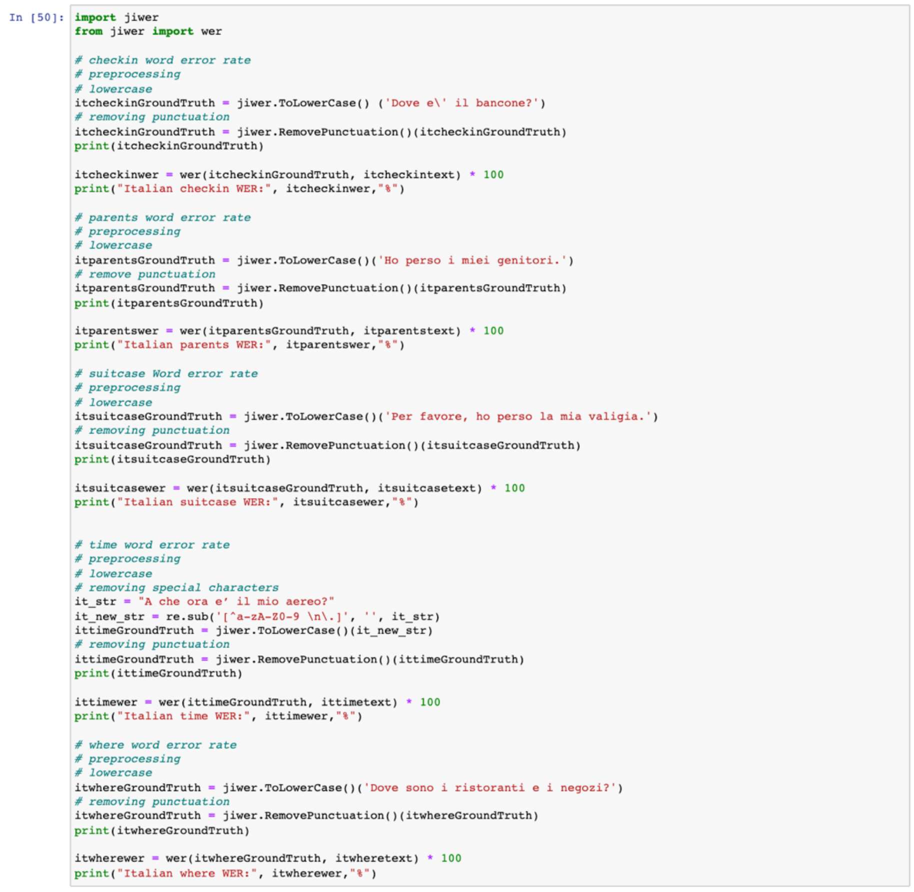
We can now repeat the steps above for the Italian audio files. However, we just need to load a different acoustic model and language model.

For Italian,
Models/output_graph_it.pbmm
Models/kenlm_it.scorer



```
In [42]: model_file_path_it = os.path.abspath('Models/output_graph_it.pbmm')
         lm_file_path_it = os.path.abspath('Models/kenlm_it.scorer')

         lm_alpha = 0.93
         lm_beta = 1.18

         model = Model(model_file_path_it)
         model.enableExternalScorer(lm_file_path_it)
```

This is to calculate the Italian WER

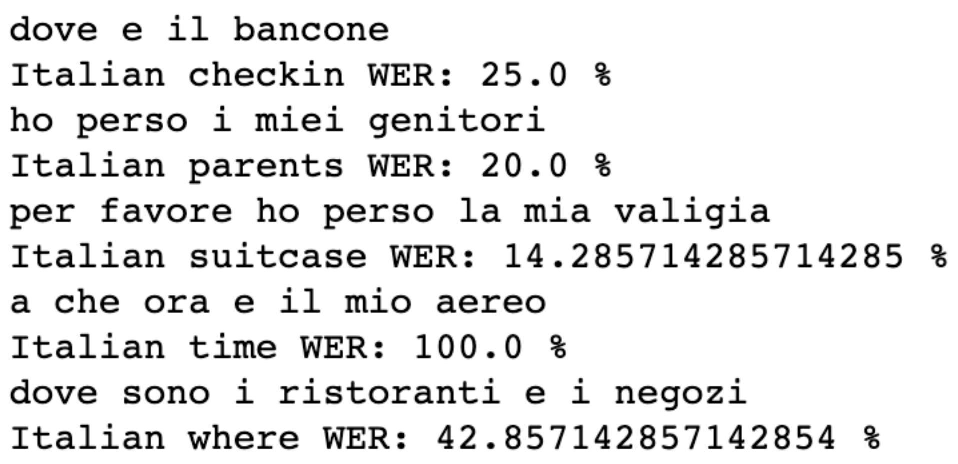


```
In [50]: import jiwer
         from jiwer import wer

         # checkin word error rate
         # preprocessing
         # lowercase
         itcheckinGroundTruth = jiwer.ToLowerCase() ('Dove e\' il bancone?')
         # removing punctuation
         itcheckinGroundTruth = jiwer.RemovePunctuation()(itcheckinGroundTruth)
         print(itcheckinGroundTruth)

         itcheckinwer = wer(itcheckinGroundTruth, itcheckintext) * 100
         print("Italian checkin WER:", itcheckinwer,"%")

         # parents word error rate
         # preprocessing
         # lowercase
         itparentsGroundTruth = jiwer.ToLowerCase()('Ho perso i miei genitori.')
         # remove punctuation
         itparentsGroundTruth = jiwer.RemovePunctuation()(itparentsGroundTruth)
         print(itparentsGroundTruth)

         itparentswer = wer(itparentsGroundTruth, itparentstext) * 100
         print("Italian parents WER:", itparentswer,"%")

         # suitcase Word error rate
         # preprocessing
         # lowercase
         itsuitcaseGroundTruth = jiwer.ToLowerCase()('Per favore, ho perso la mia valigia.')
         # removing punctuation
         itsuitcaseGroundTruth = jiwer.RemovePunctuation()(itsuitcaseGroundTruth)
         print(itsuitcaseGroundTruth)

         itsuitcasewer = wer(itsuitcaseGroundTruth, itsuitcasetext) * 100
         print("Italian suitcase WER:", itsuitcasewer,"%")


         # time word error rate
         # preprocessing
         # lowercase
         # removing special characters
         it_str = "A che ora e' il mio aereo?"
         it_new_str = re.sub('[^a-zA-Z0-9 \n\.]', '', it_str)
         ittimeGroundTruth = jiwer.ToLowerCase()(it_new_str)
         # removing punctuation
         ittimeGroundTruth = jiwer.RemovePunctuation()(ittimeGroundTruth)
         print(ittimeGroundTruth)

         ittimewer = wer(ittimeGroundTruth, ittimetext) * 100
         print("Italian time WER:", ittimewer,"%")

         # where word error rate
         # preprocessing
         # lowercase
         itwhereGroundTruth = jiwer.ToLowerCase()('Dove sono i ristoranti e i negozi?')
         # removing punctuation
         itwhereGroundTruth = jiwer.RemovePunctuation()(itwhereGroundTruth)
         print(itwhereGroundTruth)

         itwherewer = wer(itwhereGroundTruth, itwheretext) * 100
         print("Italian where WER:", itwherewer,"%")
```

I have used this below to remove special characters.

removing special characters
it_str = "A che ora e' il mio aereo?"
it_new_str = re.sub('[^a-zA-Z0-9 \n\.]', '', it_str)
ittimeGroundTruth = jiwer.ToLowerCase()(it_new_str)

The results are as seen here:



```
dove e il bancone
Italian checkin WER: 25.0 %
ho perso i miei genitori
Italian parents WER: 20.0 %
per favore ho perso la mia valigia
Italian suitcase WER: 14.285714285714285 %
a che ora e il mio aereo
Italian time WER: 100.0 %
dove sono i ristoranti e i negozi
Italian where WER: 42.857142857142854 %
```

We can see that 2 Italian audio files, the *time* audio file (100%) has a very high WER, followed by the *where* audio file (42.85%). The remaining files (*checkin, parents, suitcase)* has kept a rather low WER, whereby they are all less than 25%.

We can now repeat the steps above for the Spanish audio files. However, we just need to load a different acoustic model and language model.

For Spanish,
Models/output_graph_es.pbmm
Models/kenlm_es.scorer

```
In [33]: model_file_path_es = os.path.abspath('Models/output_graph_es.pbmm')
         lm_file_path_es = os.path.abspath('Models/kenlm_es.scorer')

         lm_alpha = 0.93
         lm_beta = 1.18

         model = Model(model_file_path_es)
         model.enableExternalScorer(lm_file_path_es)
```

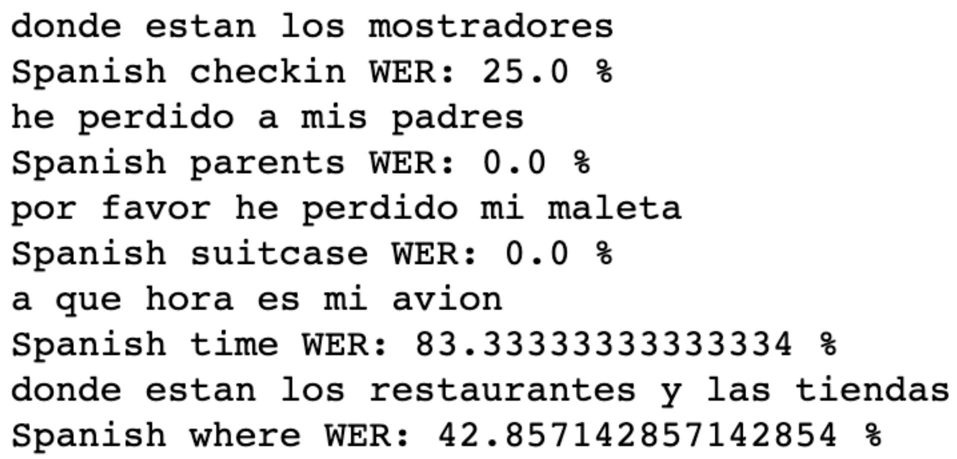
This is to calculate the Spanish WER

```
In [41]: import jiwer
         from jiwer import wer

         import re

         # checkin word error rate
         # preprocessing
         # lowercase
         # removing special characters
         es_str = "¿Dónde están los mostradores?"
         es_new_string = re.sub('[?|$|.|!|¿]', '', es_str)
         es_new_string = re.sub(u"[àáâãäå]", 'a', es_new_string)
         es_new_string = re.sub(u"[òóôõö]", 'o', es_new_string)
         escheckinGroundTruth = jiwer.ToLowerCase() (es_new_string)
         # removing punctuation
         escheckinGroundTruth = jiwer.RemovePunctuation()(escheckinGroundTruth)
         print(escheckinGroundTruth)

         escheckinwer = wer(escheckinGroundTruth, escheckintext) * 100
         print("Spanish checkin WER:", escheckinwer,"%")

         # parents word error rate
         # preprocessing
         # lowercase
         esparentsGroundTruth = jiwer.ToLowerCase()('He perdido a mis padres.')
         # removing punctuation
         esparentsGroundTruth = jiwer.RemovePunctuation()(esparentsGroundTruth)
         print(esparentsGroundTruth)

         esparentswer = wer(esparentsGroundTruth, esparentstext) * 100
         print("Spanish parents WER:", esparentswer,"%")

         # suitcase Word error rate
         # preprocessing
         # lowercase
         essuitcaseGroundTruth = jiwer.ToLowerCase()('Por favor, he perdido mi maleta.')
         # removing punctuation
         essuitcaseGroundTruth = jiwer.RemovePunctuation()(essuitcaseGroundTruth)
         print(essuitcaseGroundTruth)

         essuitcasewer = wer(essuitcaseGroundTruth, essuitcasetext) * 100
         print("Spanish suitcase WER:", essuitcasewer,"%")


         # time word error rate
         # preprocessing
         # lowercase
         # removing special characters
         es_str2 = "¿A qué hora es mi avión?"
         es_new_string2 = re.sub('[?|$|.|!|¿]', '', es_str2)
         es_new_string2 = re.sub(u"[èéêë]", 'e', es_new_string2)
         es_new_string2 = re.sub(u"[òóôõö]", 'o', es_new_string2)
         estimeGroundTruth = jiwer.ToLowerCase()(es_new_string2)
         # removing punctuation
         estimeGroundTruth = jiwer.RemovePunctuation()(estimeGroundTruth)
         print(estimeGroundTruth)

         estimewer = wer(estimeGroundTruth, estimetext) * 100
         print("Spanish time WER:", estimewer,"%")

         # where word error rate
         # preprocessing
         # lowercase
         # removing special characters
         es_str3 = "¿Dónde están los restaurantes y las tiendas?"
         es_new_string3 = re.sub('[?|$|.|!|¿]', '', es_str3)
         es_new_string3 = re.sub(u"[àáâãäå]", 'a', es_new_string3)
         es_new_string3 = re.sub(u"[òóôõö]", 'o', es_new_string3)
         eswhereGroundTruth = jiwer.ToLowerCase()(es_new_string3)
         # removing punctuation
         eswhereGroundTruth = jiwer.RemovePunctuation()(eswhereGroundTruth)
         print(eswhereGroundTruth)

         eswherewer = wer(eswhereGroundTruth, eswheretext) * 100
         print("Spanish where WER:", eswherewer,"%")
```

I have used this below to remove special characters. This is in hopes to increase the accuracy and bring down the WER.

```
# removing special characters
es_str = "¿Dónde están los mostradores?"
es_new_string = re.sub('[?|$|.|!|¿]', '', es_str)
es_new_string = re.sub(u"[àáâãäå]", 'a', es_new_string)
es_new_string = re.sub(u"[òóôõö]", 'o', es_new_string)
escheckinGroundTruth = jiwer.ToLowerCase() (es_new_string)

# removing special characters
es_str2 = "¿A qué hora es mi avión?"
es_new_string2 = re.sub('[?|$|.|!|¿]', '', es_str2)
es_new_string2 = re.sub(u"[èéêë]", 'e', es_new_string2)
es_new_string2 = re.sub(u"[òóôõö]", 'o', es_new_string2)
estimeGroundTruth = jiwer.ToLowerCase()(es_new_string2)

# removing special characters
es_str3 = "¿Dónde están los restaurantes y las tiendas?"
es_new_string3 = re.sub('[?|$|.|!|¿]', '', es_str3)
es_new_string3 = re.sub(u"[àáâãäå]", 'a', es_new_string3)
es_new_string3 = re.sub(u"[òóôõö]", 'o', es_new_string3)
eswhereGroundTruth = jiwer.ToLowerCase()(es_new_string3)
```

The results are as seen here:



```
donde estan los mostradores
Spanish checkin WER: 25.0 %
he perdido a mis padres
Spanish parents WER: 0.0 %
por favor he perdido mi maleta
Spanish suitcase WER: 0.0 %
a que hora es mi avion
Spanish time WER: 83.33333333333334 %
donde estan los restaurantes y las tiendas
Spanish where WER: 42.857142857142854 %
```

Similar to the Italian files, we can observe that 2 Spanish audio files, the *time* audio file (83.33%) has a very high WER, followed by the *where* audio file(42.86%). The remaining three files (*checkin, parents, suitcase)* has kept a relatively low WER, whereby they are all less than 25%.

Just to sum things up, I can come into a conclusion where the best results come from the English set of audio files. I guess this is due to the fact that the English acoustic and language model has been around for quite a long time, meaning it has more training time and data, compared to the Italian and Spanish models, and hence being able to transcribe the text more accurately.

I can also safely say that the provided clients' recordings have a relatively higher WER compared to my own audio recordings. Both my own recordings are at a very low WER of 0%.

I can also conclude that based on both of my own recordings, I have tried to record my voice as clear as possible, with minimal noise in the background, which led to the English language model being able to transcribe my speech to text in a very accurate manner. This means that having zero to little background noise plays a vital role in the accuracy of the transcribed text.

Therefore, I can suggest that transcribing an audio file in a busy crowded airport would be to provide these users an empty room, with soundproof walls so it can block out the sound, providing an environment which is as quiet as possible, and maybe also having an algorithm to break up the sentences into words slowly, and this will definitely improve the accuracy of the speech recognition system.

References

GitHub mozilla DeepSpeech 0.9.3. 2020. *Release DeepSpeech 0.9.3 · mozilla/DeepSpeech*. [online] Available at: <https://github.com/mozilla/DeepSpeech/releases/tag/v0.9.3> [Accessed 2 January 2022].

GitHub noisereduce. 2021. *GitHub - timsainb/noisereduce: Noise reduction in python using spectral gating (speech, bioacoustics, audio, time-domain signals)*. [online] Available at: <https://github.com/timsainb/noisereduce> [Accessed 3 January 2022].

GitHub Jiwer. 2021. *GitHub - jitsi/jiwer: Evaluate your speech-to-text system with similarity measures such as word error rate (WER)*. [online] Available at: <https://github.com/jitsi/jiwer> [Accessed 7 January 2022].