MID-TERM ASSIGNMENT REPORT

INTELLIGENT SIGNAL PROCESSING COURSEWORK 1 EXERCISE 4

Contents

Explanation of Application

Introduction and Menu

This python application will be a console command-based application.

The menu of this application simply asks the user in the 3 available languages to select the language which they can select by entering valid inputs for the respective languages.

For English language, any input of any case that results in “english” or “en” after using the str.lower() function will be accepted.

For Spanish language, any input of any case that results in “spanish” or “es” after using the str.lower() function will be accepted.

For Italian language, any input of any case that results in “italian” or “it” after using the str.lower() function will be accepted.

Once the Mozilla Deepspeech model and scorer for the respective languages are loaded, we will set valid to True and exit the while loop to proceed with the application. If not, a message notifying the users to enter a valid language will be presented in 3 different languages.

While setting up the Mozilla Deepspeech model, if we run into an error, the error will be caught and the message “Error reading model file." will be printed on the console.

```
print("Select your language------------Seleccione su idioma------------Seleziona la tua lingua")
valid = False
language = ""
while(not(valid)):
    username = input("English(EN), Espanol(ES), Italiana(IT): ")
    try:
        # change user input to lower case for easier comparison and check
        username.lower()
        # accepts english and en in any case for english language
        if username == "english" or username == "en":
            # set up language model and score
            language = "en"
            model_file_path = 'models/EN/deepspeech-0.9.3-models.pbmm'
            model = deepspeech.Model(model_file_path)
            scorer_file_path = 'models/EN/deepspeech-0.9.3-models.scorer'
            valid = True
        if username == "espanol" or username == "es":
            language = "es"
            model_file_path = 'models/ES/output_graph_es.pbmm'
            model = deepspeech.Model(model_file_path)
            scorer_file_path = 'models/ES/kenlm_es.scorer'
            valid = True
        if username == "italiana" or username == "it":
            language = "it"
            model_file_path = 'models/IT/output_graph_it.pbmm'
            model = deepspeech.Model(model_file_path)
            scorer_file_path = 'models/IT/kenlm_it.scorer'
            valid = True

        if valid == False:
            print("Please enter a valid language------------Introduzca un idioma válido----------
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

    username = input("English(EN), Espanol(ES), Italiana(IT): ")
KeyboardInterrupt
PS C:\Users\Kian Hui\Documents\GitHub\Intelligient-Signal-Processing\Ex4_files> python 'ISP Midterm-Assignment E
Select your language------------Seleccione su idioma------------Seleziona la tua lingua
English(EN), Espanol(ES), Italiana(IT):
```

After Language Selection

Once we have exited the while loop, we will complete the DeepSpeech model setup by enabling the scorer and set the alpha, beta and beam width of the model. We will next retrieve the sample rate of the model for later use.

Next, we will define the audio files that will be used to calculated the WER of the model based on the language selected.

```
model.enableExternalScorer(scorer_file_path)
lm_alpha = 0.75
lm_beta = 1.85
model.setScorerAlphaBeta(lm_alpha, lm_beta)
beam_width = 500
model.setBeamWidth(beam_width)

desired_sample_rate = model.sampleRate()

if language == "en":
    filenames = ['Ex4_audio_files/EN/checkin.wav','Ex4_audio_files/EN/checkin_child.wav',
                 'Ex4_audio_files/EN/parents.wav','Ex4_audio_files/EN/parents_child.wav',
                 'Ex4_audio_files/EN/suitcase.wav','Ex4_audio_files/EN/suitcase_child.wav',
                 'Ex4_audio_files/EN/what_time.wav','Ex4_audio_files/EN/what_time_child.wav',
                 'Ex4_audio_files/EN/where.wav','Ex4_audio_files/EN/where_child.wav',
                 'Ex4_audio_files/EN/your_sentence1.wav', 'Ex4_audio_files/EN/your_sentence2.wav']

if language == "es":
    filenames = ['Ex4_audio_files/ES/checkin_es.wav','Ex4_audio_files/ES/parents_es.wav',
                 'Ex4_audio_files/ES/suitcase_es.wav','Ex4_audio_files/ES/what_time_es.wav',
                 'Ex4_audio_files/ES/where_es.wav']

if language == "it":
    filenames = ['Ex4_audio_files/IT/checkin_it.wav','Ex4_audio_files/IT/parents_it.wav',
                 'Ex4_audio_files/IT/suitcase_it.wav','Ex4_audio_files/IT/what_time_it.wav',
                 'Ex4_audio_files/IT/where_it.wav']
converted_texts = []
```

Spectral Subtraction and Low Pass Filter

Next, we will now begin the process to modify the audio files to remove noise from the files.

We will first define an index m to 1 which will be used to modify the name of the filtered files that we will be writing.

Next, we will define the path to a crowd noise file that I have downloaded from the following source:

Source: https://mixkit.co/free-sound-effects/crowd/

This crowd noise will be used to subtract the crowd noise on the audio files provided. This crowd noise has been processed to reduce its amplitude from its original source.

Next, we will load the noise file with the previously set desired sample rate using the librosa library. We will then perform a short time fourier transform on the audio time series of the loaded noise file and modify it further.

```
# spectral subtraction and low pass filter
m = 1
noise_file = 'Ex4_audio_files/crowd_noise_2.wav'
ats, y_samping_rate = librosa.load(noise_file, sr=desired_sample_rate)
transformed_ats = librosa.stft(ats)
nss = np.abs(transformed_ats)
mns = np.mean(nss, axis=1)
```

Perform Spectral Subtraction

Next, we will iterate through each of the audio files and subtract them using the reshape function and write a new wavfile using the m index to differentiate the file names.

By using a crowd noise file and subtracting it from the actual audio signal, we could remove some of the signals that are similar which can ultimately lead us to an audio signal with a reduced or removed crowd noise.

```
# go through every audio file for the respective language
for filename in filenames:
    file_ats, y_sr = librosa.load(filename, sr=None, mono=True)
    transformed_ats = librosa.stft(file_ats)
    ss = np.abs(transformed_ats)
    angle = np.angle(transformed_ats)
    b = np.exp(1.0j * angle)

    # subtract crowd noise
    subtracted_audio = ss - mns.reshape((mns.shape[0], 1))
    subtracted_audio = subtracted_audio * b
    y = librosa.istft(subtracted_audio)

    # write a new file with the reduced crowd noise
    scipy.io.wavfile.write("mywav_reduced_noise" + str(m) + ".wav", y_sr, (y * 32768).astype(np.int16))
```

Perform Low Pass Filter

We will read just written, reduced noise file to perform the low pass filter on.

First, we will set the cut off frequency. The frequency range of an adult ranges from 85-255Hz while a child ranges from 250-400Hz. We could set a frequency slightly above these ranges to significantly reduce noises. However, as I tried different frequencies, it is found that the cut off frequency of 2000Hz works best for English and Italian language while 3000Hz works best for the Spanish language.

We will use the signal.lfilter() of scipy library to perform the low pass filter on the audio file signal.

```
    # perform low pass filter on reduced crowd noise file
    freq_sampling_rate, data = wavfile.read("mywav_reduced_noise" + str(m) + ".wav")

    # different cutoff frequency for different languages, explanation in report
    if language == "en" or language == "it":
        cut_off_frequency = 2000.0
    if language == "es":
        cut_off_frequency = 3000.0
    freq_ratio = cut_off_frequency / freq_sampling_rate

    N = int(math.sqrt(0.196201 + freq_ratio**2) / freq_ratio)

    win = np.ones(N)
    win *= 1.0/N
    # low pass filter
    filtered = scipy.signal.lfilter(win, [1], data).astype(np.int16)
```

Before we set the filtered signal data to a new file, we will first open the reduced noise wav file to get the amplitude width, number of channels and frames.

Once we have retrieved that information, we will write a new wav file with a similar name to the reduced noise file and include “filtered” at the end. We will then set the parameters with their respective data that we retrieved. We will then write the filtered frames into the file.

```
    # retrieve amp width, channel and frame count from reduced noise file
    w = wave.open("mywav_reduced_noise" + str(m) + ".wav", 'r')
    amp_width = w.getsampwidth()
    n_channels = w.getnchannels()
    n_frames = w.getnframes()
    w.close()

    # write new filtered file
    wav_file = wave.open("mywav_reduced_noise" + str(m) + "filtered.wav", "w")
    wav_file.setnchannels(1)
    wav_file.setsampwidth(amp_width)
    wav_file.setframerate(desired_sample_rate)
    wav_file.setnframes(n_frames)
    wav_file.writeframes(filtered.tobytes('C'))
    wav_file.close()
```

Once the file is written we will reopen it in read mode to retrieve the number of frames and use that to retrieve the frames of the filtered file. These frames will then be converted to np.int16 data type which can be used to run the speech to text model. Once the text is converted, we will store it in the converted_texts lists which we will calculate the WER later. We will increment the m index and repeat this cycle until noise reduction and filter is performed on all files.

```
    # open newly written file
    w = wave.open("mywav_reduced_noise" + str(m) + "filtered.wav", 'r')
    number_of_frames = w.getnframes()
    frames = w.readframes(number_of_frames)

    # Mozilla DeepSpeech model perform speech to text
    data16 = np.frombuffer(frames, dtype=np.int16)
    text = model.stt(data16)
    converted_texts.append(text)
    w.close()
    m += 1
```

Processing Transcript Texts

Before comparing the recognised texts of the DeepSpeech model, we will process the transcripts by removing all punctuations as the DeepSpeech is unable to predict that. All punctuations is removed except “-“ where we will replace it with a space as the “-“ is usually used to connect separate words. We will then convert all texts to lower case. Once the modifications are complete, we will append them to the transcript text list.

```
if language == "en":
    transcript_texts = ["Where is the check-in desk?", "Where is the check-in desk?",
                        "I have lost my parents.", "I have lost my parents.",
                        "Please, I have lost my suitcase.", "Please, I have lost my suitcase.",
                        "What time is my plane?", "What time is my plane?",
                        "Where are the restaurants and shops?","Where are the restaurants and shops?",
                        "Where is the departure hall?", "How do I get to the arrival hall?"]
if language == "es":
    transcript_texts = ["¿Dónde están los mostradores?", "He perdido a mis padres.",
                        "Por favor, he perdido mi maleta.", "¿A qué hora es mi avión?",
                        "¿Dónde están los restaurantes y las tiendas?"]
if language == "it":
    transcript_texts = ["Dove e' il bancone?", "Ho perso i miei genitori.",
                        "Per favore, ho perso la mia valigia.", "A che ora e' il mio aereo?",
                        "Dove sono i ristoranti e i negozi?"]


for i, text in enumerate(transcript_texts):
    # remove punctuations and convert everything to lower case
    final_text = "".join(character for character in text if character not in ("?", ".", ";", ":", "!", ",", "¿", "'"))
    final_text = final_text.replace("-", " ")
    final_text = final_text.lower()
    transcript_texts[i] = final_text
```

Evaluating Performance

During the evaluation, we will iterate through the converted texts, compare them to the transcripts, display the individual WER and add them to the total scores to produce the overall evaluation score.

First, we will split the text and transcripts up into a list which contains all the words in the text separated by spaces.

We will then calculate the substitution and deletion errors by subtracting the length of transcript by the length of intersection between the text_list and transcript_text_list. The intersection() function will find all texts in a set that matches the other in another set. If there are any missing texts or incorrect texts, they will not be matched and will not be considered in the intersected set, resulting in a lower length than the transcript list.

```
166  sum_of_SDI = 0
167  N = 0
168  total_WER = 0
169  total_SDI = 0
170  total_N = 0
171  for i, text in enumerate(converted_texts):
172      print("Recognised text: " + text)
173      print("Transcript text: " + transcript_texts[i])
174      text_list = text.split(" ")
175      transcript_text_list = transcript_texts[i].split(" ")
176
177      # any substitution and deletion errors will be detected as all those scenarios will result in unmatched texts
178      # correct number of texts in transcript text minus the total number of matches gives us the number of errors found
179      substitution_and_deletion_errors = len(transcript_text_list) - len(set(text_list).intersection(transcript_text_list))
```

The insertion error can be calculated by subtracting the length of transcript list by the text list. If there are more words in the text than the transcript, these words will be reflected in the length difference. However, since there is also a possibility that the length of text would be lower than the length of transcripts (deletion error) which have already been accounted, we will check if the length difference is negative which represents a larger text list length. If the condition is true, we will perform an absolute function on the length difference and add it to the insertion errors.

Once all errors are accounted for, we will sum them up and assign it to the sum_of_SDI. N will be the correct total number of words. The current WER is then computed using the provided formula and displayed before all we add the current sum_of_SDI and N value to the total. At the end of evaluating every sentence, we will calculate the total WER and print it in the console.

```
181      # insertion errors can be calculated by subtracting length of text by length of transcript text
182      # abs() is applied as deletion error occurs when length of text_list is lower than transcript while the opposite
183      # represents insertion error
184      length_diff = len(transcript_text_list) - len(text_list)
185      if length_diff < 0:
186          insertion_errors = abs(length_diff)
187      else:
188          insertion_errors = 0
189      sum_of_SDI = substitution_and_deletion_errors + insertion_errors
190      N = len(transcript_text_list)
191      print("Number of errors: " + str(sum_of_SDI))
192      print("Total number of words: " + str(N))
193      current_WER = sum_of_SDI/N * 100
194      print("WER for current transcript: " + str(current_WER) + "%")
195      total_SDI += sum_of_SDI
196      total_N += N
197
198  total_WER = total_SDI/total_N * 100
199  print("Overall " + language + " WER: " + str(total_WER) + "%")
```

WER Before and After

WER before noise reduction and filter:

Language	File	WER
English	checkin.wav	33.33%
English	parents.wav	20%
English	suitcase.wav	0%
English	what_time.wav	20%
English	where.wav	0%
English	your_sentence1.wav	20%
English	your_sentence2.wav	12.5%
English	checkin_child.wav	100%
English	parents_child.wav	20%
English	suitcase_child.wav	50%
English	what_time_child.wav	20%
English	where_child.wav	0%
Overall English WER		24.64%
Spanish	checkin.wav	50%
Spanish	parents.wav	0%
Spanish	suitcase.wav	0%
Spanish	what_time.wav	83.33%
Spanish	where.wav	57.14%
Overall Spanish WER		39.29%
Italian	checkin.wav	25%
Italian	parents.wav	20%
Italian	suitcase.wav	14.29%
Italian	what_time.wav	85.71%
Italian	where.wav	42.86%
Overall Italian WER		40%

WER after noise reduction:

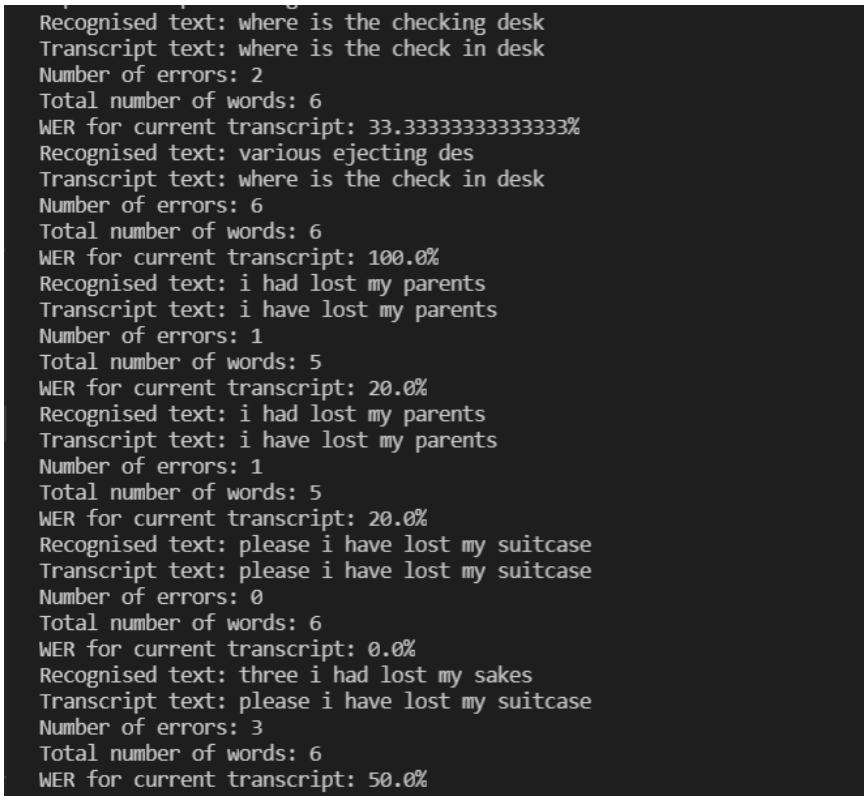
Language	**File**	**WER**
English	checkin.wav	33.33%
English	parents.wav	20%
English	suitcase.wav	33.33%
English	what_time.wav	20%
English	where.wav	0%
English	your_sentence1.wav	20%
English	your_sentence2.wav	0%
English	checkin_child.wav	50%
English	parents_child.wav	0%
English	suitcase_child.wav	50%
English	what_time_child.wav	20%
English	where_child.wav	0%
Overall English WER		20.29%
Spanish	checkin.wav	50%
Spanish	parents.wav	0%
Spanish	suitcase.wav	33.33%
Spanish	what_time.wav	50%
Spanish	where.wav	42.86%
Overall Spanish WER		35.71%
Italian	checkin.wav	25%
Italian	parents.wav	0%
Italian	suitcase.wav	14.29%
Italian	what_time.wav	85.71%
Italian	where.wav	57.14%
Overall Italian WER		40%

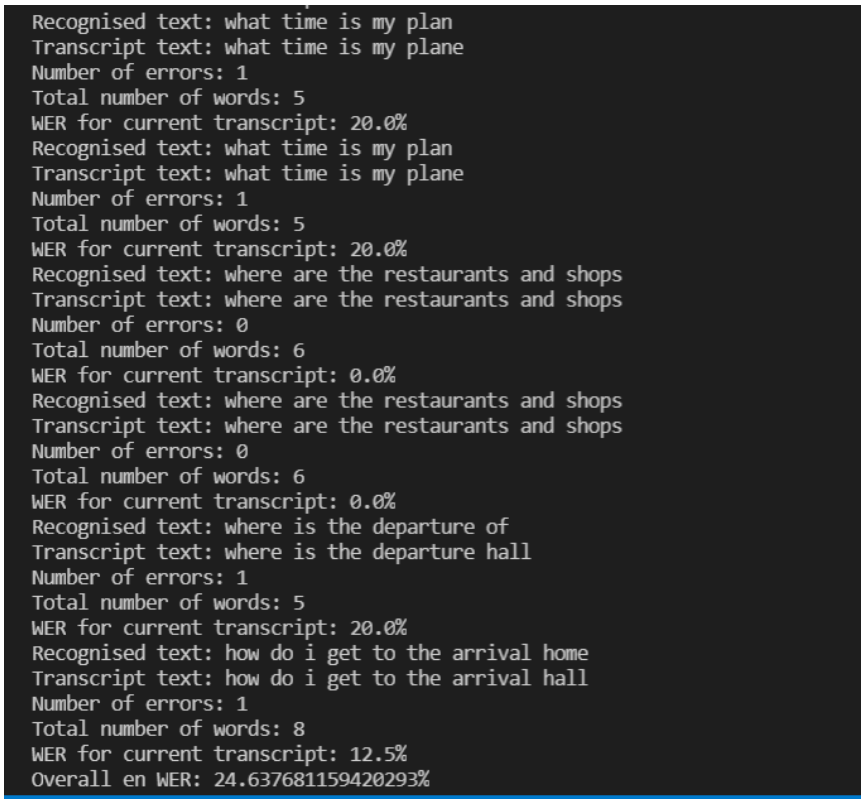
WER after noise reduction and low pass filter:

Language	**File**	**WER**
English	checkin.wav	33.33%
English	parents.wav	0%
English	suitcase.wav	33.33%
English	what_time.wav	20%
English	where.wav	0%
English	your_sentence1.wav	20%
English	your_sentence2.wav	0%
English	checkin_child.wav	66.66%
English	parents_child.wav	0%
English	suitcase_child.wav	33.3%
English	what_time_child.wav	20%
English	where_child.wav	0%
Overall English WER		18.84%
Spanish	checkin.wav	50%
Spanish	parents.wav	0%
Spanish	suitcase.wav	33.33%
Spanish	what_time.wav	50%
Spanish	where.wav	42.86%
Overall Spanish WER		35.71%
Italian	checkin.wav	25%
Italian	parents.wav	0%
Italian	suitcase.wav	14.29%
Italian	what_time.wav	85.71%
Italian	where.wav	42.86%
Overall Spanish WER		36.67%

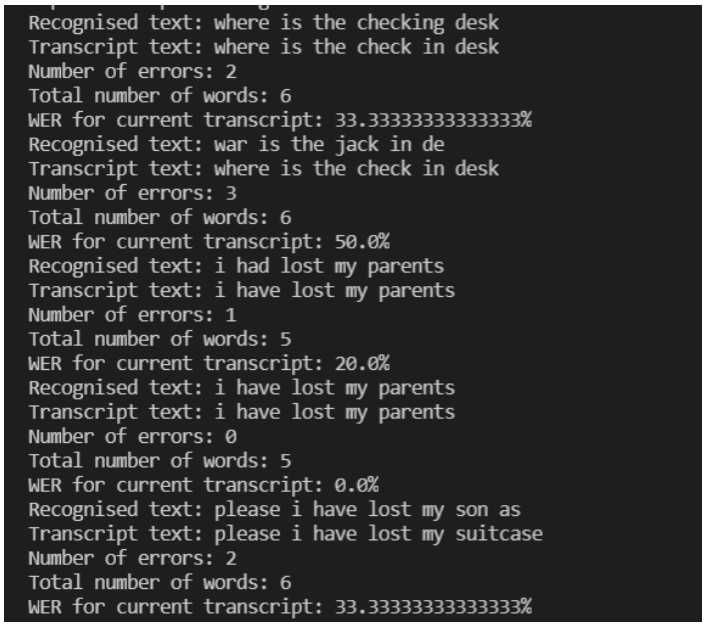
Screenshot of WER before noise reduction and filter (English):

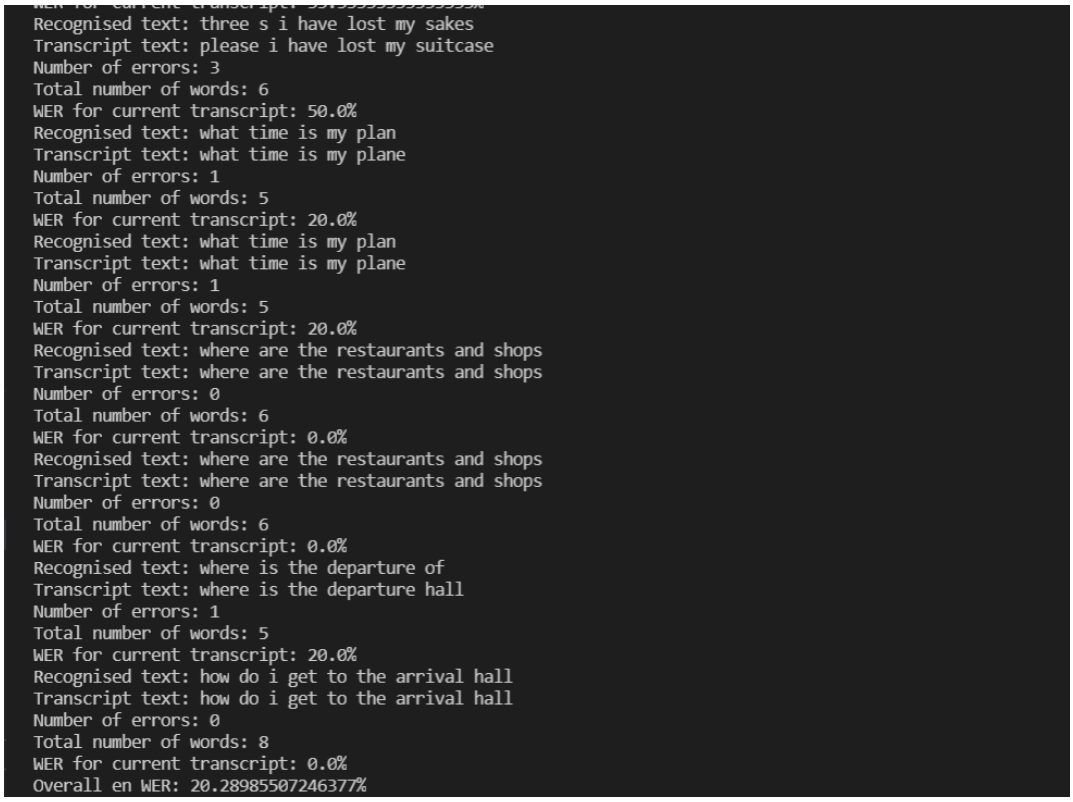
*note: Child version of audio comes after the adult version.



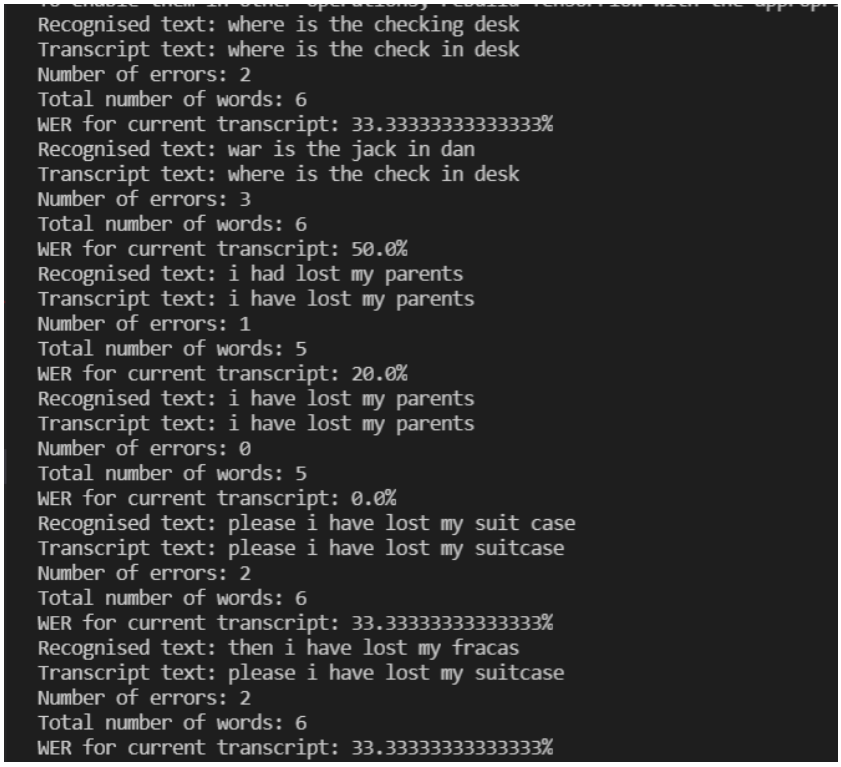


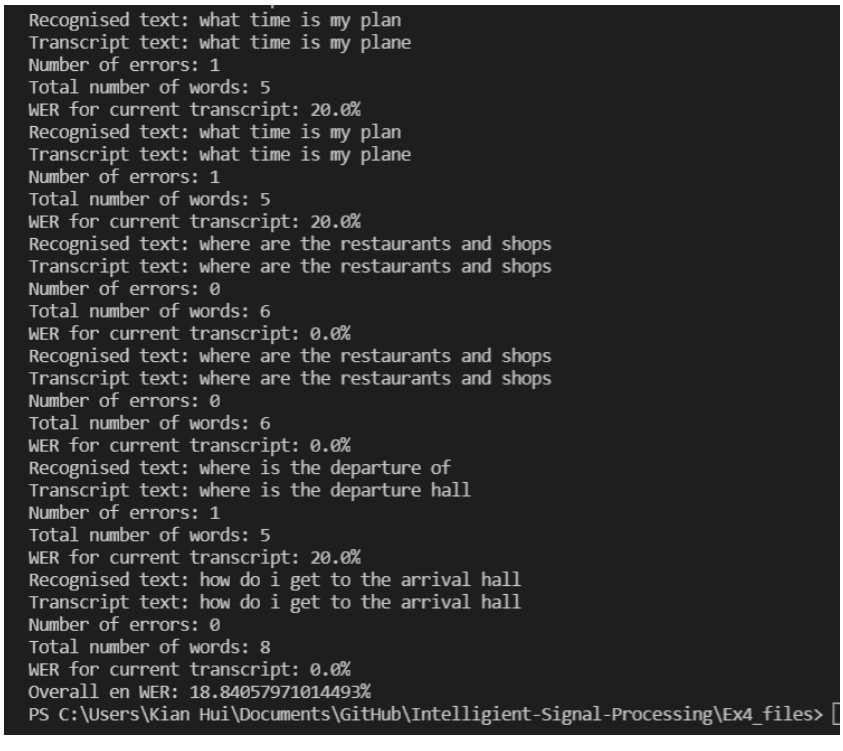
Screenshot of WER after noise reduction but before filter (English):





Screenshot of WER after noise reduction and filter (English):

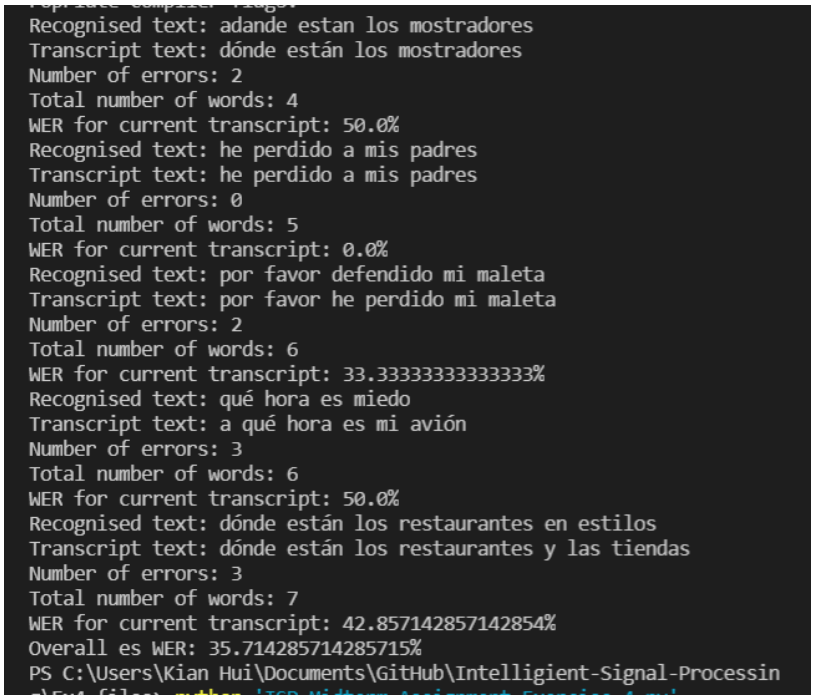




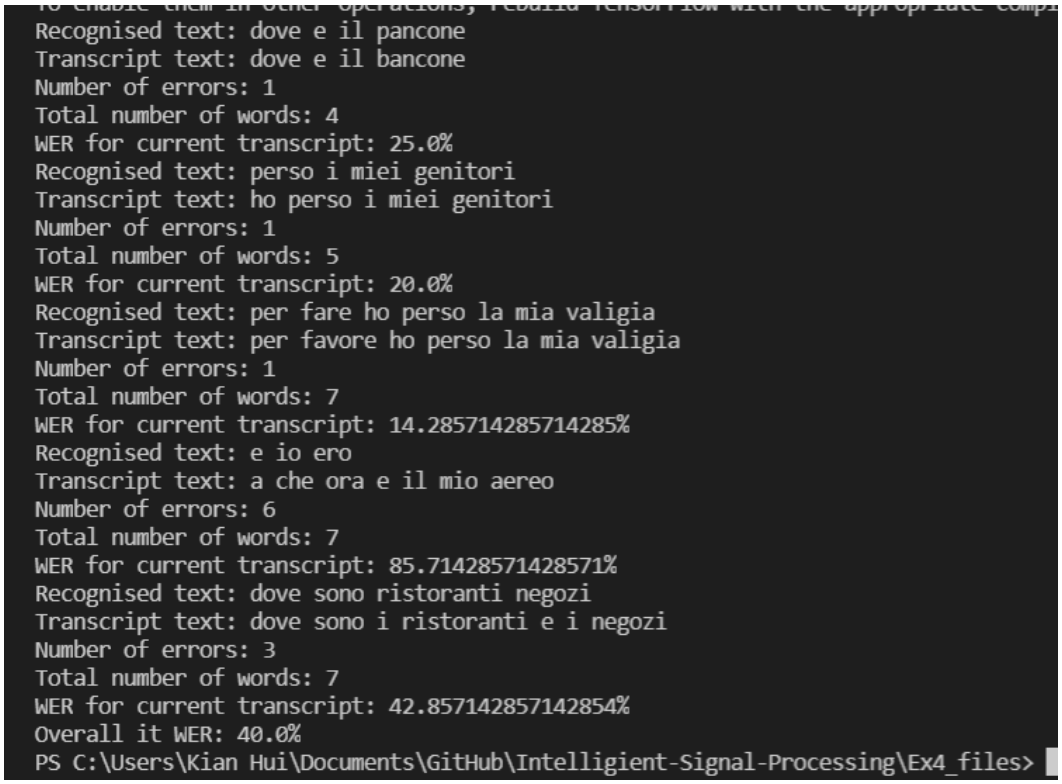
Screenshot of WER before noise reduction and filter (Spanish):

Screenshot of WER after noise reduction but before filter (Spanish):

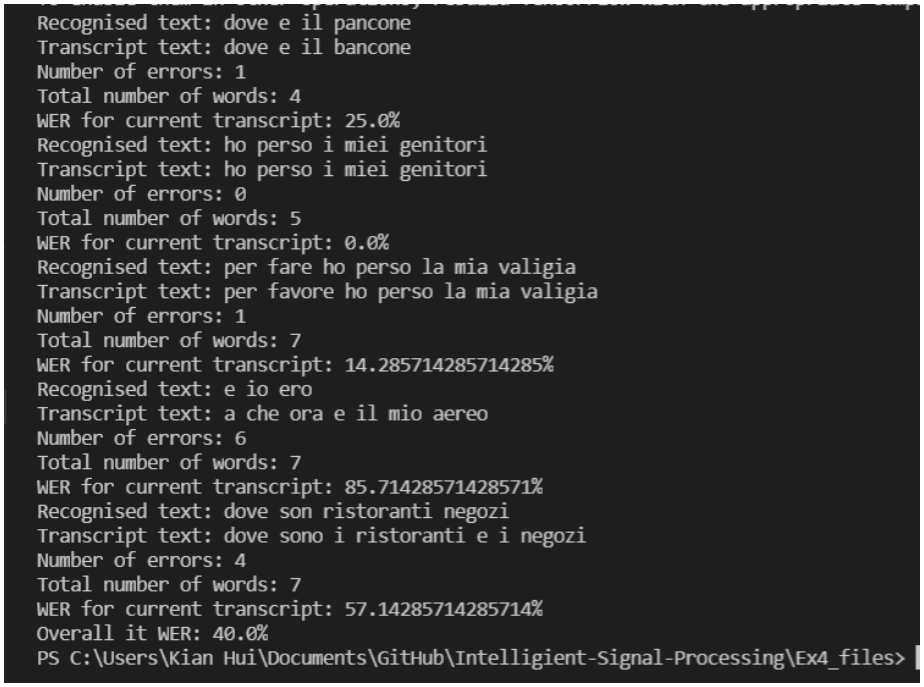
Screenshot of WER after noise reduction and filter (Spanish):



Screenshot of WER before noise reduction and filter (Italian):



Screenshot of WER after noise reduction but before filter (Italian):



Screenshot of WER after noise reduction and filter (Italian):