Automatic Speech Recognition

October 5, 2020

1 Automatic Speech Recognition

1.1 First step:

Extracting the Audio from video

```
In [1]: import moviepy.editor as mp
        import librosa as lr
        import matplotlib.pyplot as plt
        import numpy as np
        import librosa.display as display
        import scipy
        from IPython.display import Audio
        import pydub
        from pydub import AudioSegment
        from pydub.silence import split_on_silence
        import speech_recognition as sr
        import os
        import jiwer
        print("Done! necessary libraries are imported")
```

```
Done! necessary libraries are imported
```

1.1.1 Extracting the audio from video file

```
In [2]: clip = mp.VideoFileClip(r"iceland.mp4")
        clip.audio.write_audiofile(r"converted.wav")
        Audio("converted.wav")
```

```
chunk:   0%|          | 12/3176 [00:00<00:34, 92.80it/s, now=None]
```

```
MoviePy - Writing audio in converted.wav
```

```
MoviePy - Done.
```

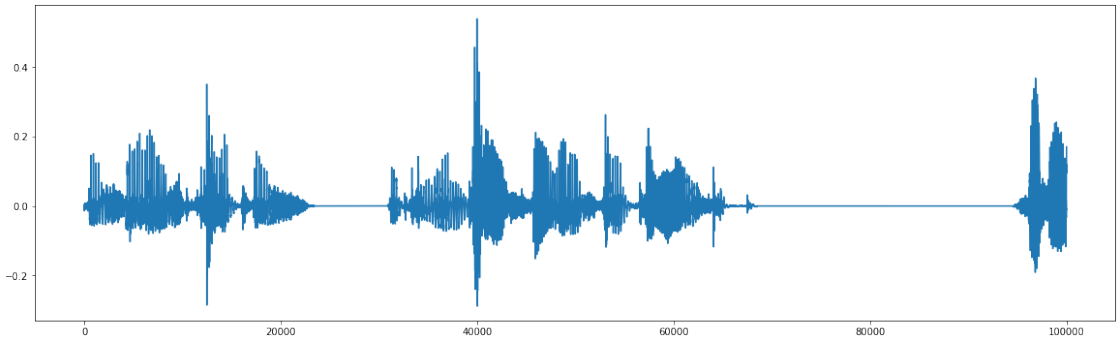
```
Out[2]: <IPython.lib.display.Audio object>
```

1.1.2 Signal visualization

```
In [3]: #read the audio
        samples, sample_rate = lr.load("converted.wav")

        plt.figure(figsize=(20, 6))
        plt.plot(samples[500000: 600000])
        n_samples = len(samples)
        print("number of samples is:",n_samples)
```

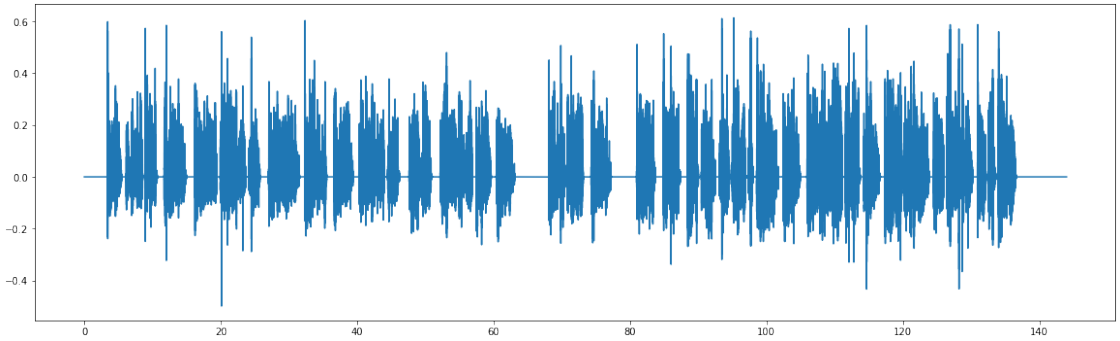
```
number of samples is: 3175200
```



1.1.3 plot the audio in time domain

```
In [4]: time = np.arange(0, len(samples))/sample_rate
        plt.figure(figsize=(20, 6))
        plt.plot(time, samples)
        print("n_time_points:",len(time))
```

```
n_time_points: 3175200
```

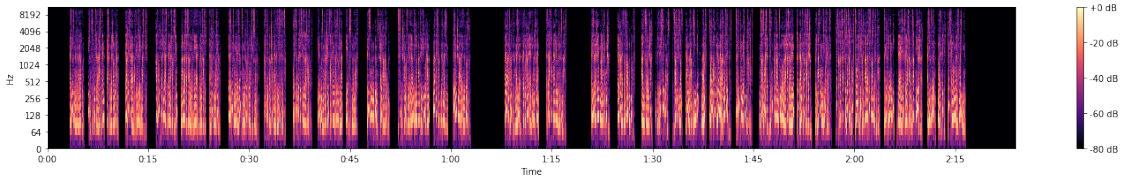


```
In [5]: total_duration = time.shape[0]/sample_rate
        print("total_duration in seconds :",total_duration)

total_duration in seconds : 144.0
```

1.1.4 Short time fourier transform

```
In [6]: y =samples
        stft_feature = lr.amplitude_to_db(np.abs(lr.stft(y,
                                                         n_fft=1024,
                                                         hop_length=512,
                                                         window=scipy.signal.hanning
                                                        )),
                                          ref=np.max
                                         )
        # Plot Spectrogram
        plt.figure(figsize=(20,3))
        display.specshow(stft_feature, y_axis='log', x_axis='time')
        plt.colorbar(format='%+2.0f dB')
        plt.tight_layout()
        plt.show()
```



1.1.5 split the audio into segments of 3-5 seconds

```
In [7]: sound = AudioSegment.from_wav(file="converted.wav")
        print("total duration:", sound.duration_seconds)

        # split on silence
        chunks = split_on_silence(sound, min_silence_len=200, silence_thresh=-34, keep_silence=T


        # remerge segments if duration less than 3 second
        durations_b_m = []
        for chunk in chunks:
            durations_b_m.append(chunk.duration_seconds)


        duration_minimun = 3
```

```
        chunks_merged = []

        currently_merging = False
        merging_chunk = None

        for j in range(0, len(chunks)):

            if not currently_merging:
                current_chunk = chunks[j]
            else:
                current_chunk = merging_chunk + chunks[j]

            if current_chunk.duration_seconds > duration_minimun:
                 chunks_merged.append(current_chunk)
                 merging_chunk = None
                 currently_merging = False
            else:
                currently_merging = True
                merging_chunk = current_chunk

        # saving the segments after merging as wav file
        durations = []
        for i, chunk in enumerate(chunks_merged):
            durations.append(chunk.duration_seconds)
            chunk.export("chunks/chunk{:02d}.wav".format(i), format="wav")


        print("total number of chunks before merging is:", len(chunks))
        print("total number of chunks after merging is:", len(chunks_merged))

        print("durations of segments before merging in seconds")
        for i, d in enumerate(durations_b_m[0:5]):
            print("segment %d: %.2f "%(i,d))

        print("duration of segments after merging in seconds")
        for i, d in enumerate(durations[0:10]):
            print("segment %d: %.2f "%(i,d))

        #print("minimum %.2f sec, maximum %.2f sec" %(min(durations), max(durations)))
```

```
total duration: 144.0
total number of chunks before merging is: 56
total number of chunks after merging is: 30
durations of segments before merging in seconds
segment 0: 5.71
segment 1: 0.95
segment 2: 0.62
segment 3: 1.43
```

```
segment 4: 2.36
duration of segments after merging in seconds
segment 0: 5.71
segment 1: 3.00
segment 2: 5.80
segment 3: 5.15
segment 4: 4.21
segment 5: 6.84
segment 6: 5.29
segment 7: 3.61
segment 8: 4.67
segment 9: 5.11
```

```
In [8]: Audio("chunks/chunk00.wav")

Out[8]: <IPython.lib.display.Audio object>

In [9]: Audio("chunks/chunk01.wav")

Out[9]: <IPython.lib.display.Audio object>

In [10]: Audio("chunks/chunk03.wav")

Out[10]: <IPython.lib.display.Audio object>
```

1.1.6 Transcribe the segments using SpeechRecognition python library

```
In [11]: files = os.listdir("chunks")
         sorted_files = sorted(files)
         sorted_files = sorted_files[1:]
         #print("files:", sorted_files)

         r = sr.Recognizer()
         text_segments = []
         for file in sorted_files:
             audio = sr.AudioFile("chunks"+"/"+file)
             with audio as source:
                 audio_file = r.record(source)
                 result = r.recognize_google(audio_file, language='de-DE')
                 text_segments.append(result)
             print(result)

         with open('recognized_segments.txt', mode='w') as file:
             for item in text_segments:
                 file.write("%s\n" % item)
```

```
Geysire sind ein Wunderwerk der Natur
Island ist berühmt für seine hei$$en Quellen
```

```
eiskalten Gletscher und Vulkane etwa 30 Feuerberge auf der Insel gelten als
tief aber warum Plätschern viele hei$$e Quellen einfach vor sich hin
und andere lassen das Wasser mehr oder weniger explosionsartig
dann der dampfenden fontainea für das seltene Phänomen Geysir müssen mehrere Faktoren perfekt
zusammenpacken ein unverzichtbarer Bestandteil eines Geysirs ist hei$$es Wasser
unter Island befindet sich ein sogenannter Hotspot
wie ein Bunsenbrenner schmilzt aufsteigendes hei$$es Material
die Erdkruste immer wieder auf durch diese sogenannten blooms
steigt Magma nach oben das flüssige Gestein aus dem Erdinneren ist die Grundlage für die
vielen Vulkane und das einzigartige thermal System
dass ich der Mensch überall auf der Insel zunutze macht
damit aus einer Thermalquelle ein Geysir wird
reicht vulkanische Energie aber nicht allein
es bedarf noch einer geologischen Besonderheit
Geysire and springen immer einen verengten Schacht
der Vulkan heizt das Wasser von unten auf
aber die Wassersäule im Schacht übt durch irgend
zu viel Druck aus dass das Wasser nicht wie sonst
bei 100 Grad zu 7 beginnt und verdampft
sondern auch noch bei höheren Temperaturen flüssig bleibt
das Prinzip kennen wir vom Dampfkochtopf
wenn das Wasser so hei$$ wird dass ich unten angefangen bis in den oberen Bereich Gasblasen bilde
drückt der Dampf etwas Wasser aus dem Schacht durch die Entlastung ändern sich die Druck
Verhältnisse jetzt kocht das Wasser explosionsartig auf
das Volumen vervielfacht sich dabei um das bis zu 1600
das erklärt auch die hohe Energie mit der der Geysir das Gemisch aus Dampf und Wasser
in die Höhe schleudert Geysire sind selten
und auch sehr empfindlich kleinste Veränderungen können Sie zum Versiegen bringen
```

1.1.7 Evaluation

```
In [12]: with open("transcribed.txt", "r") as test:
             refs = test.readlines()
         with open("recognized.txt", "r") as pred:
             preds = pred.readlines()

         reference = refs[0]
         predicted = preds[1]

         measures = jiwer.compute_measures(refs[0], preds[1])
         wer = measures['wer']
         mer = measures['mer']
         print("Evaluation")
         print("word error rate is: ", wer*100)
         print("match error rate is: ", mer*100)

Evaluation
```

```
word error rate is:  9.386281588447654
match error rate is:  9.31899641577061
```