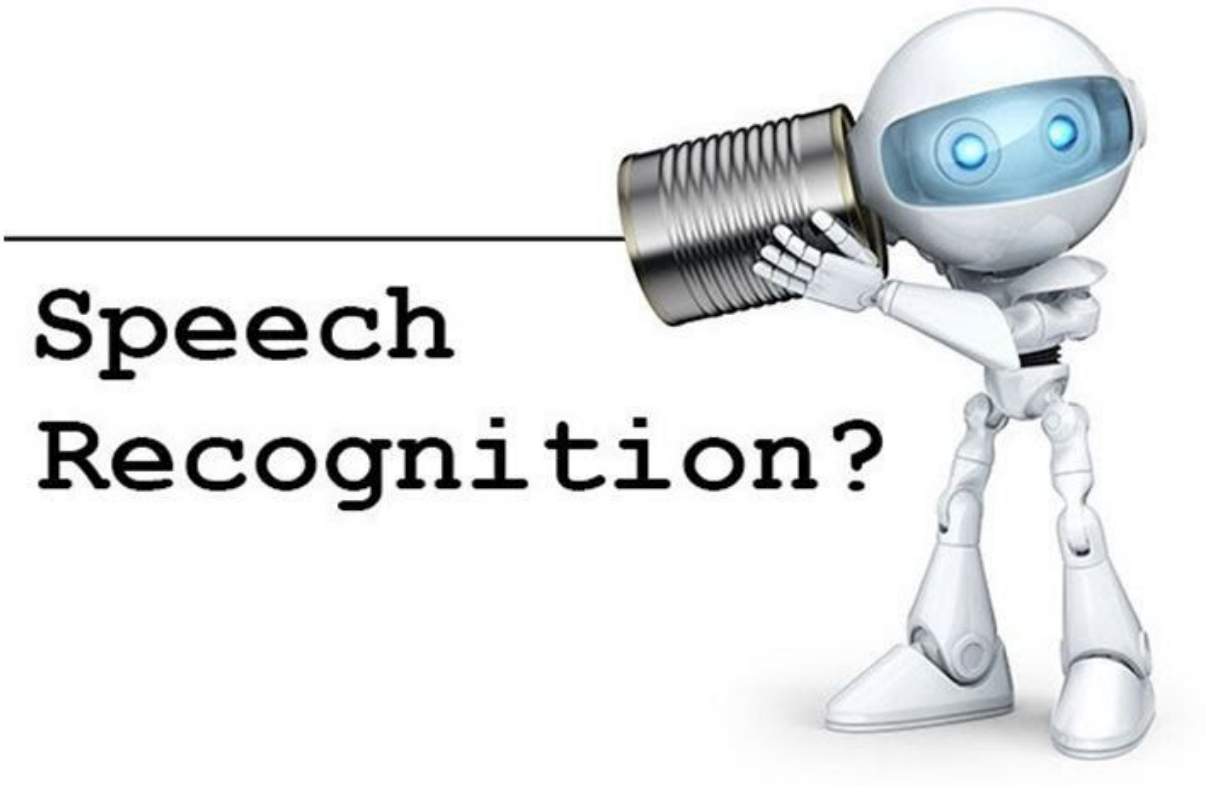
Speech Diarizer and Recognizer

7.Machine Learning Assignment



Jigar Joshi

24/04/2020

SY Computer Science Engineering Student,

KJ Somaiya College of Engineering, Mumbai.

Introduction:-

Problem Statement

Rohan is in a startup and has frequent discussions with his colleagues. He has a system that automatically transcribes the verbal discussions but he does not know what part was spoken by whom.

My task

Help Rohan develop a system that correctly detects each different speaker in an audio file. The system should give a text file containing the different speakers and the time during which they spoke. It should be able to recognize a minimum of 3 different speakers. Your submission should include:

1. A document describing your approach to the problem
2. Your code files
3. A sample audio file and a text file constructed by the system on this audio file

Approach

We first examine the problem that we are going to solve and realize the domain to which the problem belongs. Here this problem is based on speech diarization and speech recognition. Considering the time span of the deadline and my grasp over the topic and experience.I had no experience in audio processing so I came to the conclusion that training a complete machine learning model on my own would be a big task and it itself can take days or weeks to complete.

I went through technical papers mentioned in the bibliography to get the basic knowledge about sound processing and the libraries provided by python that can ease the task. After clearing the availability of the libraries, I started researching about the pre-trained models that can be directly used for the task of prediction without wasting time in creating a model, gathering data to train it, waiting until the model trains with no guarantee that it will be correct, resulting in wasting time.

After the research work is completed I was ready with the various pre-trained model. I then examine the input and output of these models. I went through the documentation of each of this model and implemented them. Various models found by me were either incorrectly predicting or were outdated and required a lot of dependencies error to be solved before they could function correctly. Finally, I found a model that was fitting the requirement to solve the problem statement.

I started pre-processing the data that was going to be the input of this model. After getting data ready for the input, I visualized the data so I can myself have an idea about how the data should behave. I pushed the data into the model and examined the output.

We required output in the text file so I used file reader and writer of Python to write every line of output in the text file. After this, I used a collab library to download this file output by me.

Github was used to maintain an opensource copy of this project and an online copy of code was maintained on google docs for my reference.

Code:

Libraries Used:-

- typing-extensions==3.7.4.2
- SpeechRecognition
- Pyannote-audio
- Pyannote-core
- Scipy
- wavio
- Ipython
- Google.colab
- Torch
- Numpy
- Matplotlib

Code and explanation:

```
pip install typing-extensions==3.7.4.2
```

We are using this command to download the required version of type extensions as the library pyannote supports only certain versions of typing-extensions.

```
!pip install -q https://github.com/pyannote/pyannote-audio/tarball/develop
```

We are using this command to download the required version of pyannote as if this is not meant then we might get an error about “Database_yml” not found which I encountered and took me hours to come to this solution.

```
!wget -q
http://groups.inf.ed.ac.uk/ami/AMICorpusMirror/amicorpus/ES2004a/audio/ES2004a.Mix-Headset.wav
```

We are using this command to download the sample audio file. Here it comes with .rttm file which has logs written in it.

```
DEMO_FILE = {'uri': 'ES2004a.Mix-Headset', 'audio':
'ES2004a.Mix-Headset.wav'}
```

We are using this command to create a DEMO_FILE varaible which will point to our sample audio file.

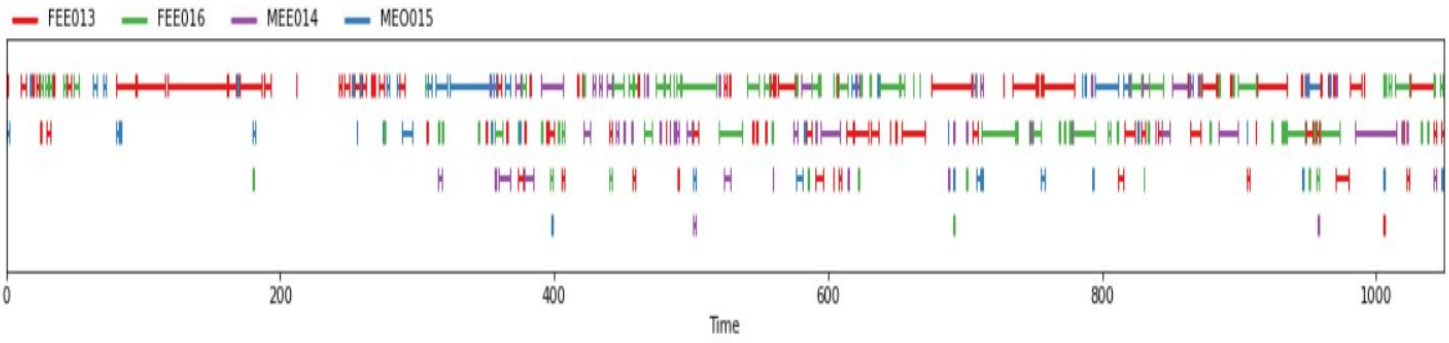
```
!wget -q
https://raw.githubusercontent.com/pyannote/pyannote-audio/develop/tutorials/data_preparation/AMI/MixHeadset.test.rttm
```

We are using this command to get a direct tutorial link to pre-process the data(Not required)

```
# load groundtruth

from pyannote.database.util import load_rttm

groundtruth = load_rttm('MixHeadset.test.rttm')[DEMO_FILE['uri']]
```

```
# visualize groundtruth

Groundtruth
```

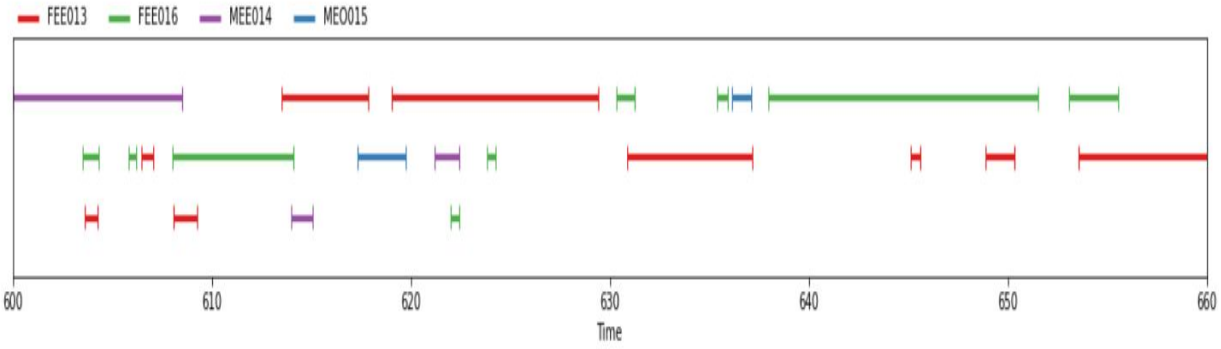
We use this command to visualize the data. Each colour represents a speaker. When a line starts that the speaker starts to speak, and he ends his speech when that colour line ends.



.

```
from pyannote.core import Segment, notebook

# make notebook visualization zoom on 600s < t < 660s time range

EXCERPT = Segment(600, 660)

notebook.crop = EXCERPT


# visualize excerpt groundtruth

Groundtruth
```

This command is used to get a detailed view of a small section of the audio file. In this case only 600-660 second gap.




```
from pyannote.audio.features import RawAudio

from IPython.display import Audio

from scipy.io import wavfile
```

```
# load audio waveform, crop excerpt, and play it

waveform = RawAudio(sample_rate=16000).crop(DEMO_FILE, EXCERPT)

Audio(data=waveform.squeeze(), rate=16000, autoplay=True)

demo_new = wavfile.write('result.wav',16000,waveform)

Audio(data=waveform.squeeze,filename
='result.wav',rate=16000,autoplay=True)

print(demo_new)
```

Processing a complete 10-minute file to see whether a particular section of code is behaving as required or not is a time-consuming task. So we crop this audio file into a minute-long audio file and save it in the directory in .wav format

```
import torch

scd = torch.hub.load('pyannote/pyannote-audio', 'scd_ami')

# obtain raw SCD scores (as `pyannote.core.SlidingWindowFeature` instance)

new_DEMO_FILE = {'uri': 'ES2004a.Mix-Headset', 'audio': 'result.wav'}

from pyannote.audio.features import Pretrained

# speech activity detection model trained on AMI training set
```

```
sad = torch.hub.load('pyannote/pyannote-audio', 'sad_ami')

# speaker change detection model trained on AMI training set

scd = torch.hub.load('pyannote/pyannote-audio', 'scd_ami')

# overlapped speech detection model trained on AMI training set

ovl = torch.hub.load('pyannote/pyannote-audio', 'ovl_ami')

# speaker embedding model trained on AMI training set

emb = torch.hub.load('pyannote/pyannote-audio', 'emb_ami')

# sad = Pretrained(validate_dir='/path/to/validation/directory')

scd_scores = scd(DEMO_FILE)

ovl_scores = ovl(DEMO_FILE)

sad_scores = sad(DEMO_FILE)

from pyannote.audio.utils.signal import Binarize

binarize = Binarize(offset=0.52, onset=0.52, log_scale=True,

                    min_duration_off=0.1, min_duration_on=0.1)
```

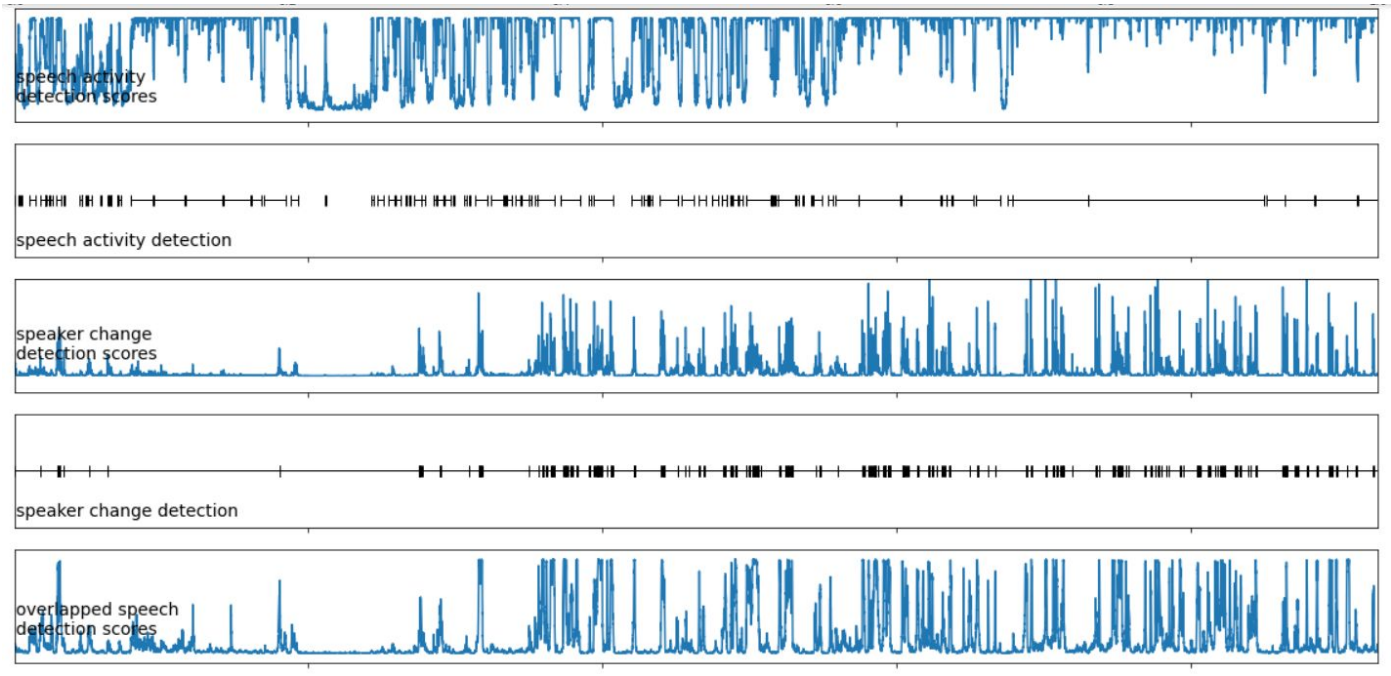
```
# speech regions (as `pyannote.core.Timeline` instance)

speech = binarize.apply(sad_scores, dimension=1)

# detect peaks and return speaker homogeneous segments

# NOTE: both alpha/min_duration values were tuned on AMI dataset.

# you might need to use different values for better results.

from pyannote.audio.utils.signal import Peak

peak = Peak(alpha=0.10, min_duration=0.10, log_scale=True)



# speaker change point (as `pyannote.core.Timeline` instance)

partition = peak.apply(scd_scores, dimension=1)
```

```
# let's visualize SAD, SCD and OVL results using pyannote.core
visualization API

import numpy as np

from matplotlib import pyplot as plt

from pyannote.core import Segment, notebook


# only plot one minute (between t=120s and t=180s)

notebook.crop = Segment(1,927)


# helper function to make visualization prettier

from pyannote.core import SlidingWindowFeature

plot_ready = lambda scores: SlidingWindowFeature(np.exp(scores.data[:,
1:]), scores.sliding_window)


# create a figure with 8 rows with matplotlib

nrows = 8

fig, ax = plt.subplots(nrows=nrows, ncols=1)

fig.set_figwidth(20)

fig.set_figheight(nrows * 2)

overlap = binarize.apply(ovl_scores, dimension=1)

# # 1st row: reference annotation

# notebook.plot_annotation(DEMO_FILE['uri'], ax=ax[0], time=False)

# ax[0].text(notebook.crop.start + 0.5, 0.1, 'reference', fontsize=14)
```

```
# 2nd row: SAD raw scores
notebook.plot_feature(plot_ready(sad_scores), ax=ax[1], time=False)
ax[1].text(notebook.crop.start + 0.5, 0.1, 'speech activity\ndetection
scores', fontsize=14)
ax[1].set_ylim(-0.1, 1.1)

# 3rd row: SAD result
notebook.plot_timeline(speech, ax=ax[2], time=False)
ax[2].text(notebook.crop.start + 0.5, 0.1, 'speech activity detection',
fontsize=14)

# 4th row: SCD raw scores
notebook.plot_feature(plot_ready(scd_scores), ax=ax[3], time=False)
ax[3].text(notebook.crop.start + 0.5, 0.1, 'speaker change\ndetection
scores', fontsize=14)
ax[3].set_ylim(-0.1, 0.6)

# 5th row: SCD result
notebook.plot_timeline(partition, ax=ax[4], time=False)
ax[4].text(notebook.crop.start + 0.5, 0.1, 'speaker change detection',
fontsize=14)

# 6th row: OVL raw scores
notebook.plot_feature(plot_ready(ovl_scores), ax=ax[5], time=False)
ax[5].text(notebook.crop.start + 0.5, 0.2, 'overlapped speech\ndetection
```

```
scores', fontsize=14)

ax[5].set_ylim(-0.1, 1.1)


# 7th row: OVL result

notebook.plot_timeline(overlap, ax=ax[6], time=False)

ax[6].text(notebook.crop.start + 0.5, 0.1, 'overlapped speech detection',
fontsize=14)


# 8th row: reference annotation

notebook.plot_annotation(test_file['annotation'], ax=ax[7], legend=False)

_ = ax[7].text(notebook.crop.start + 0.5, 0.1, 'reference', fontsize=14)
```

We are using the above commands to just visualize the data.



```
# load pipeline

import torch

#pre-trained model is loaded

pipeline = torch.hub.load('pyannote/pyannote-audio', 'dia')



# apply diarization pipeline on your audio file

# diarization = pipeline({'audio': '/content/ES2004a.Mix-Headset.wav.1'})

diarization = pipeline({'audio': '/content/ES2004a.Mix-Headset.wav'})



fe= open("out.txt","w")

fe.write("We start")

# dump result to disk using RTTM format

with open('result.wav', 'w') as f:

    diarization.write_rttm(f)
```

```
# iterate over speech turns

for turn, _, speaker in diarization.itertracks(yield_label=True):

    print(f'Speaker "{speaker}" speaks between t={turn.start:.1f}s and
t={turn.end:.1f}s.')

    fe.write(f'Speaker "{speaker}" speaks between t={turn.start:.1f}s and
t={turn.end:.1f}s.')

    fe.write(" \n ")

# Speaker "A" speaks between t=0.2s and t=1.4s.

# Speaker "B" speaks between t=2.3s and t=4.8s.
```

This is the most important part of code where we load the pre-trained model, pass the sample audio file as an input to the pipeline loaded by us. We write every output track into the console as well as in the file.

```
import speech_recognition as sr

print(sr.__version__)

r = sr.Recognizer()

inputted= sr.AudioFile('/content/orignal.wav')

print(type(inputted))

with inputted as source:

  audio = r.record(source)

print(type(audio))

# try:

#   print(r.recognize_google(audio))

try:

    text = r.recognize_google(audio, language = 'en-IN', show_all = True )

    print("I thinks you said '" + r.recognize_google(audio) + "'")


    f_text='https://www.google.co.in/search?q=' + text

    wb.get(chrome_path).open(f_text)

except Exception:

  pass

print("Hey")
```

We are using google speech recognition API to convert speech to text.

References and Resources:

1. https://stackoverflow.com/questions/52249985/python-speech-recognition-tool-does-not-recognize-wav-file
2. https://stackoverflow.com/questions/56359106/speech-recognition-unknownvalueerror
3. https://colab.research.google.com/drive/1wG8vCJdnZzxfACNfwXh6DwqPCKq9IsKe#scrollTo=Hgu9KQk_8DRv
4. https://github.com/pyannote/pyannote-audio/issues/257
5. https://stackoverflow.com/questions/51524964/google-colaboratory-shows-error-when-saving-a-file-to-github-could-not-fetch-b
6. https://github.com/pyannote/pyannote-audio/tree/develop/tutorials/pretrained/model
7. https://pypi.org/project/pyannote.audio/tutorials/pretrained/pipeline
8. Technical Paper: Facilitating the Manual Annotation of Sounds When Using Large Taxonomies
9. http://groups.inf.ed.ac.uk/ami/download/
10. https://docs.python.org/3/library/wave.html
11. https://github.com/nryant/dscore
12. https://www.tutorialspoint.com/read-and-write-wav-files-using-python-wave
13. https://colab.research.google.com/drive/1--xY78_ZTFwpI7F2ZfaeyFKiAOG2nkwd
14. https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.write.html
15. https://drive.google.com/drive/u/0/search?q=result.wav
16. https://github.com/googlecolab/colabtools/issues/34
17. https://stackoverflow.com/questions/18952716/valueerror-i-o-operation-on-closed-file
18. https://github.com/pyannote/pyannote-database#installation
19. https://towardsdatascience.com/how-to-use-google-speech-to-text-api-to-transcribe-long-audio-files-1c886f4eb3e9
20. https://blog.api.rakuten.net/top-10-best-speech-recognition-apis-google-speech-ibm-watson-speechapi-and-others/
21. Technical Paper: SIDEKIT for diarization(S4D)
22. https://pypi.org/project/s4d/

Links for Open Source Project:

Github: https://github.com/JigarJoshi04/Speech_Recognizer

Report:
https://docs.google.com/document/d/1Q-Z9oIWlxI0_6YUuNHRdscctyUm45N5mwBrcf_czLa0/edit#

Code:

https://docs.google.com/document/d/1wCfqvFpjA1gT8QuInl5F7gl25Gwp2T03lWpDB8Knw_w/edit