Early Spark classroom Participation v2

Contributor: Hassan Khawaja
Date Started: 11/18/2020
Last Edit: 12/07/2020
Intended Use Case: The intended use case of this algorithm is for the specified objective alone and may or may not accurately represent other applications if used in a different context.

Goal: `Measure class participation through student vs. teacher speaking`

Details:

- Research speaker voice identification APIs (such as AWS Transcribe). If noise reduction is not done implicitly, please complete the noise reduction step (below) before proceeding
- Distinguish between adult and child voices
- Calculate % time of adult voices vs. individual student voices vs. all-class responses
- Determine Class Participation Score #1
- Evaluate performance of computed participation score against participate score assigned through video observation

Tutorial: https://medium.com/saarthi-ai/who-spoke-when-build-your-own-speaker-diarization-module-from-scratch-e7d725ee279

In [1]:
```
# Set Up

# !pip install pydub
# !pip3 install spectralcluster
# !pip install moviepy
```

In [2]:
```
# imports
import pandas as pd
import numpy as np
from pydub import AudioSegment
from resemblyzer import preprocess_wav, VoiceEncoder
from pathlib import Path
import IPython.display as ipd # https://musicinformationretrieval.com/ipython_au
from spectralcluster import SpectralClusterer
import os
from moviepy.editor import *

# visualization imports
import plotly
import plotly.express as px
plotly.offline.init_notebook_mode(connected=True)

# hide warnings
```

```
import warnings
warnings.filterwarnings('ignore')
```

Running all of the code w/ Graphical Outputs

In [3]:
```
audio_file_paths = ['teach-videos/Example4-720.wav', 'teach-videos/Example 6.wav
                    'teach-videos/TEACH English Teaching 3.wav','teach-videos/T
                    'teach-videos/TEACH English Teaching 1.wav','teach-videos/E
```

In [20]:
```
# audio 0
# t0_df = audio_file_run(audio_file_paths[0])
```

In [21]:
```
# audio 1
# t1_df = audio_file_run(audio_file_paths[1])
```

In [22]:
```
# audio 2
# t2_df = audio_file_run(audio_file_paths[2])
```

In [23]:
```
# audio 3
# t3_df = audio_file_run(audio_file_paths[3])
```

In [24]:
```
# audio 4
# t4_df = audio_file_run(audio_file_paths[4])
```

In [30]:
```
# audio 5
# t5_df = audio_file_run(audio_file_paths[5])
```

In [31]:
```
# audio 6
# t6_df = audio_file_run(audio_file_paths[6])
```

In [ ]:
```

```

Visualizations

In [4]:
```
audio_file_paths
```

Out[4]:
```
['teach-videos/Example4-720.wav',
 'teach-videos/Example 6.wav',
 'teach-videos/TEACH English Teaching 3.wav',
 'teach-videos/TEACH English Teaching 2.wav',
 'teach-videos/TEACH English Teaching 1.wav',
 'teach-videos/Example5.wav']
```

In [5]:
```
#key 0:teacher 1:students
```

In [6]:
```
# making df
# data is received from
#https://docs.google.com/document/d/1yTKkxJYrSyxqrXX0WoNeMdQM6ki2A2EVPRZlIDb8LYU

cols = ['speaker', 'tot_time_spoken', 'audio_file']
```

```
# example 4
e4 = [[0,  .439, 'teach-videos/Example4-720.wav'],
      [1,  .561, 'teach-videos/Example4-720.wav']]
df_e4 = pd.DataFrame(e4, columns=cols)
gen_df = pd.DataFrame(e4, columns=cols) # this is the df we append all stuff too

#example 5
e5 = [[0,  .674, 'teach-videos/Example5.wav'],
      [1,  .326, 'teach-videos/Example5.wav']]
df_e5 = pd.DataFrame(e5, columns=cols)

#example 6
e6 = [[0,  .684, 'teach-videos/Example 6.wav'],
      [1,  .316, 'teach-videos/Example 6.wav']]
df_e6 = pd.DataFrame(e6, columns=cols)

#English Teaching 1
et1 = [[0,  .674, 'teach-videos/TEACH English Teaching 1.wav'],
       [1,  .326, 'teach-videos/TEACH English Teaching 1.wav']]
df_et1 = pd.DataFrame(et1, columns=cols)

#English Teaching 2
et2 = [[0,  .58, 'teach-videos/TEACH English Teaching 2.wav'],
       [1,  .42, 'teach-videos/TEACH English Teaching 2.wav']]
df_et2 = pd.DataFrame(et2, columns=cols)

#English Teaching 3
et3 = [[0,  .907, 'teach-videos/TEACH English Teaching 3.wav'],
       [1,  .0925, 'teach-videos/TEACH English Teaching 3.wav']]
df_et3 = pd.DataFrame(et3, columns=cols)
```

In [12]:
```
# run this to get the dataframe
df_list_no_df4 = [df_e5 ,df_e6 ,df_et1 ,df_et2, df_et3] # left out e4
df_list = [df_e4, df_e5 ,df_e6 ,df_et1 ,df_et2, df_et3]
for df in df_list_no_df4:
    gen_df = gen_df.append(df)
```

In [13]:
```
gen_df.head()
```

Out[13]:

	speaker	tot_time_spoken	audio_file	new_audio_file
0	0	0.439	teach-videos/Example4-720.wav	Example4-720.wav
1	1	0.561	teach-videos/Example4-720.wav	Example4-720.wav
0	0	0.674	teach-videos/Example5.wav	NaN
1	1	0.326	teach-videos/Example5.wav	NaN
0	0	0.684	teach-videos/Example 6.wav	NaN

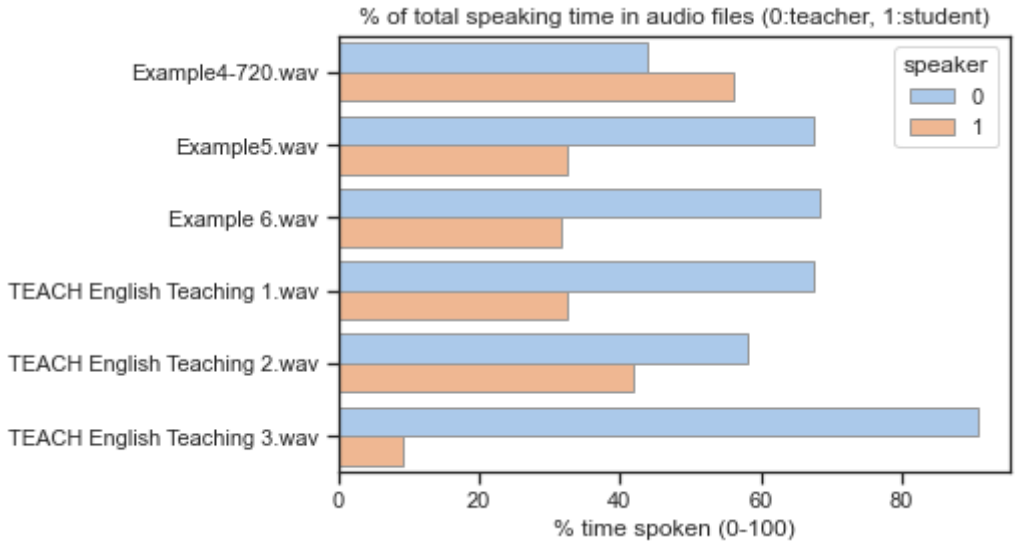
In [14]:
```
# removing the initial path to only show the file name
gen_df['new_audio_file'] = [file_path.split('/')[1] for file_path in list(gen_df
```

In [23]:
```
# added the percentage spoken rather than decimal
gen_df['perc_tot_time_spoken'] = [val*100 for val in list(gen_df['tot_time_spoke
```

In [24]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks", color_codes=True)
```

In [25]:
```
ax = sns.barplot(x="perc_tot_time_spoken", y="new_audio_file" , hue="speaker",
            palette="pastel", edgecolor=".6", data=gen_df,
            orient = 'h')
ax.set_title('% of total speaking time in audio files (0:teacher, 1:student)')
ax.set_xlabel('% time spoken (0-100)')
ax.set_ylabel('')
```

Out[25]:
```
Text(0, 0.5, '')
```



Functions

In [9]:
```
def pretty_print(path_name):
    print('-'*117)
    print('**'*18 + '   ' + path_name+ '   ' + '**'*18)
    print('-'*117)
    return None
```

In [10]:
```
def mp4_to_wav(vid_file_path):
    '''This function converts mp4 -> mp3 -> wav files'''
    # convert vid -> mp3
    video = VideoFileClip(os.path.join(vid_file_path))
    mp3_file_path = vid_file_path.split('.')[0] + '.mp3'
    video.audio.write_audiofile(os.path.join(mp3_file_path))

    # convert mp3 -> wav
    sound = AudioSegment.from_mp3(mp3_file_path)
    audio_file_path = mp3_file_path.split('.')[0] + '.wav'
    sound.export(audio_file_path, format="wav")
    return audio_file_path
```

In [11]:
```
def mp4_path_gen():
    '''This generated all of the paths for the mp4 video files'''
    for root, dirs, files in os.walk('./teach-videos/'):
        for filename in files:
            if os.path.splitext(filename)[1] == ".mp4":
                yield os.path.join(root, filename)
```

In [12]:
```
def create_labelling(labels,wav_splits):
```

```
    '''This function outputs labelling which has diarized speakers'''
    from resemblyzer import sampling_rate
    times = [((s.start + s.stop) / 2) / sampling_rate for s in wav_splits]
    labelling = []
    start_time = 0

    for i,time in enumerate(times):
        if i>0 and labels[i]!=labels[i-1]:
            temp = [str(labels[i-1]),start_time,time]
            labelling.append(tuple(temp))
            start_time = time
        if i==len(times)-1:
            temp = [str(labels[i]),start_time,time]
            labelling.append(tuple(temp))

    return labelling
```

In [13]:

```
# the main function that runs all other functions
def wav_to_labelling(audio_file_path):
    '''This function abstracts a lot of the general'''
    wav_fpath = Path(audio_file_path)

    wav = preprocess_wav(wav_fpath)

    encoder = VoiceEncoder("cpu")
    _, cont_embeds, wav_splits = encoder.embed_utterance(wav, return_partials=Tr
    print(cont_embeds.shape)

    clusterer = SpectralClusterer(min_clusters=2, max_clusters=100,
                                  p_percentile=0.90, gaussian_blur_sigma=1)

    labels = clusterer.predict(cont_embeds)

    labelling = create_labelling(labels,wav_splits)

    change_in_audio_len(labelling, audio_file_path)
    return labelling
```

In [14]:

```
def change_in_audio_len(labelling, audio_file_path):
    '''Displays a string showing the change in audio length after removing silen
    # full audio len
    audio = AudioSegment.from_file(audio_file_path)
    full_min_time = audio.duration_seconds/60

    # no silence audio len -- diarized
    sec_time = labelling[len(labelling)-1][2]
    min_time = sec_time/60
    print("**"*4 +
          'Audio went from '+
          str(full_min_time) +
          ' to ' +
          str(min_time) +
          ' minutes after removing silences'+
          "**"*4)
    return None
```

In [15]:

```
def labels_to_df(label_list):
    '''
    This function turns the variable labelling into a df
    labelling: label_list
```

```
    '''
    cols = ['speaker', 'start_time', 'end_time', 'time_speaking']
    df_data_lst = []
    for l in label_list:
        speaker, start_time, end_time = l[0],l[1],l[2]
        speaking_time = end_time-start_time
        df_data_lst.append([l[0], l[1], l[2], speaking_time])
    df = pd.DataFrame(df_data_lst, columns=cols)
    return df
```

In [16]:
```
def to_percentage_df(df):
    '''
    This function outputs the % of speak time per speaker in a df
    '''
    cols = ['speaker', 'tot_time_spoken']
    num_speakers = len(set(df['speaker'])) # getting unique num speakers
    df_data_lst = []

    for i in range(num_speakers):
        curr_speaker = i
        tot_speak = sum(df[df['speaker']==str(i)]['time_speaking'])
        df_data_lst.append([curr_speaker, tot_speak])
    return pd.DataFrame(df_data_lst, columns=cols)
```

In [17]:
```
def ouput_df_func(labelling):
    '''outputs 2 tables:
    1. speaker_df - contains every label in df form
    2. tots_df - aggregated w/ tot time and %
    '''
    speaker_df = labels_to_df(labelling) # this function turns labels to a df

    # next thing is to get a % for student and teacher talking time

    tots_df = to_percentage_df(speaker_df) # This function gets # of speakers an
    # adding a % column using the totals
    total_speech = sum(tots_df['tot_time_spoken'])
    tots_df['speech % (0-100)'] = [sp/total_speech*100 for sp in tots_df['tot_ti
    return speaker_df, tots_df
```

In [18]:
```
# running the whole files
def audio_file_run(audio_file_path):
    pretty_print(audio_file_path) # print out the path nicely
    speaker_df, tots_df = ouput_df_func(wav_to_labelling(audio_file_path)) # tot
    tots_df.head()
    # bar graph
    # https://plotly.com/python/bar-charts/
    fig = px.bar(tots_df, x="speaker", y="speech % (0-100)", color="speaker", op
    fig.show()
    # pi chart
    # https://plotly.com/python/pie-charts/
    fig = px.pie(tots_df, values='speech % (0-100)', names='speaker', title='Spe
    fig.show()
    return tots_df
```

In [ ]:
```

```

Changing the mp4 files to .wav

In [14]:
```
# teach_vid_paths = [mp4file[2:] for mp4file in mp4_path_gen()] # gets all of th
```

In [42]:
```
# running for multiple mp4 -> wav file
'''This code converts all of the videos in teach-video into .wav it only needs t
why I manually set the variable "audio_file_names_list" below so that we don't n
# audio_file_paths = []
# for path in teach_vid_paths:
#     audio_file_name = mp4_to_wav(path)
#     audio_file_paths.append(audio_file_name) # append .mp3 names
#     print('*'*10 + audio_file_name + '*'*10)
```

Out[42]:
```
'This code converts all of the videos in teach-video into .wav \nit only needs t
o be run once and that is why I manually set the variable "audio_file_names_lis
t"\nbelow so that we don\'t need to rerun everytime\n'
```

In [44]:
```
# running for a single mp4 -> wav file
# vid_file_path = "es_video.mp4"
# audio_file_path = mp4_to_wav(vid_file_path)
# print(audio_file_path)
```

Using Resemblyzer for Speaker Diarization

In [15]:
```
audio_file_paths[0]
```

Out[15]:
```
'teach-videos/Example4-720.wav'
```

In [7]:
```
ipd.Audio(audio_file_paths[0]) # load the WAV file
```

Out[7]:

0:00 / 15:11

In [37]:
```
# running the whole files
labelling = wav_to_labelling(audio_file_path)
```

```
Loaded the voice encoder model on cpu in 0.05 seconds.
(7447, 256)
Audio went from 10.457166288737717 to 7.459333333333333 minutes after removing s
ilences
```

Calculating the % of participation

In [49]:
```
speaker_df, tots_df = ouput_df_func(labelling) # tots is aggregated speaker_df
```

In [67]:
```
tots_df.head()
```

Out[67]:

	speaker	tot_time_spoken	speech % (0-100)
0	0	406.16	90.749844
1	1	41.40	9.250156

Visuals

Visualizing Audio

- https://musicinformationretrieval.com/ipython_audio.html

In [8]:
```
# bar graph
# https://plotly.com/python/bar-charts/

# fig = px.bar(tots_df, x="speaker", y="speech % (0-100)", color="speaker", opac
# fig.show()
```

In [9]:
```
# pi chart
# https://plotly.com/python/pie-charts/

# fig = px.pie(tots_df, values='speech % (0-100)', names='speaker', title='Speak
# fig.show()
```

In [75]:
```

```

In [ ]:
```

```