1/3/2023

Speech recognition

Fateme Emadi
97440152

libraries

این پروژه با زبان پایتون در محیط نوت بوک برنامه نویسی شده است و از کتابخانه هایی نظیر

1. *Librosa*
2. *Dtw*
3. *Matplotlib*
4. *Numpy*

استفاده شده است که برای خواندن فایل صوتی رسم نمودار و محاسبه فاصله بین سیگنال های صدا مورد استفاده قرار می گیرد.

ابتدا ده تا فایل صوتی (سلام) با صدای خودم با فرمت wav ضبط کردم و سپس یک فایل صوتی هم با صدای یک نفر دیگر .

برای خواندن فایل های صوتی

```
my_audio = []
audio = []
for i in range(10):
    path = editFiles[i]

my_audio.append(sr.AudioFile(path))
    with my_audio[i] as source:

audio.append(r.record(source))
```

سپس با استفاده از یکی از کتابخانه های مترجم گوگل متن هر فایل صوتی چک شده است که یک وقت اشتباه نباشند.
خروجی "سلام" است که همان محتوای فایل های صوتی است.

```
for i in range(10):
    r.recognize_google(audio[i],
language="fa_IR")
r.recognize_google(audio[0],
language="fa_IR")
```

Out[5]:

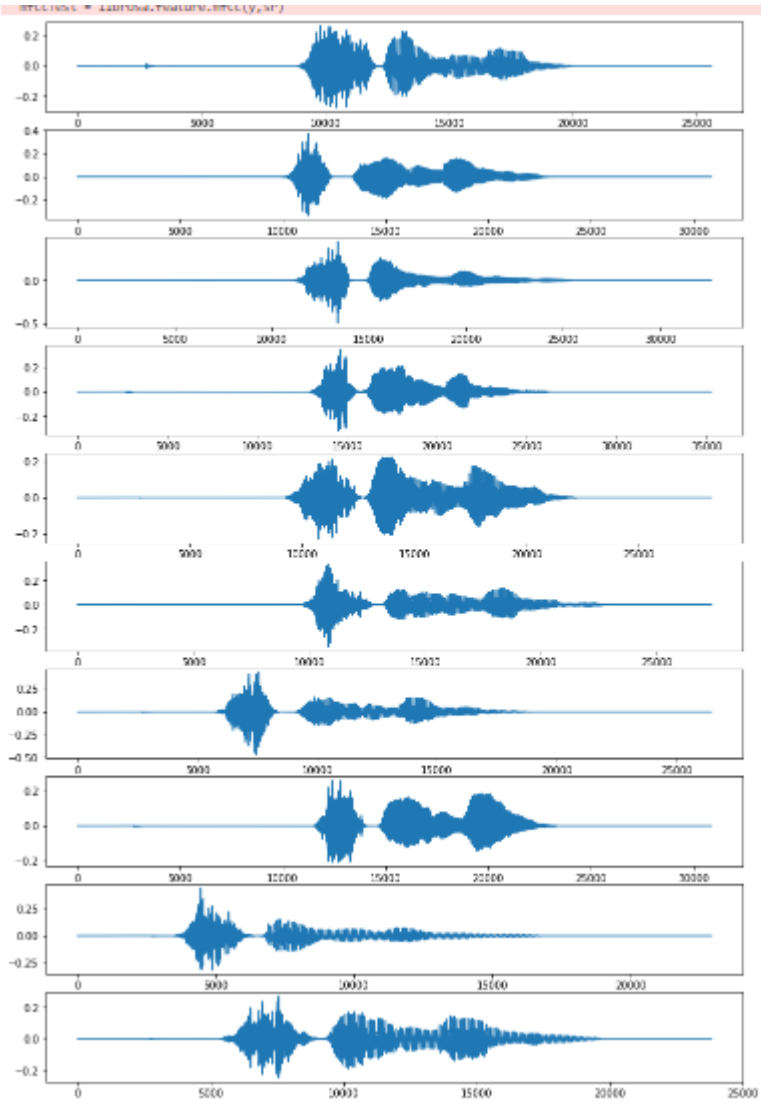
```
'سلام'
```

ویژگی های مورد استفاده:

1. *MFCC*
2. *Mean*
3. *Windowing*
4. *Energy*

```
train_audio = []
sr_arr = []
mfcc_train = []
for i in range(10):

    #define variable to save data and
rate audio
    path = editFiles[i]
    y, sr = librosa.load(path)

    #resize each figure to show
better
    fig = plt.gcf()
    fig.set_size_inches(12.5, 19.5)

    #show matplotlib speek
    plt.subplot(10, 1, i+1)
    plt.plot(y)
    IPython.display.Audio(data=y,
rate=sr)

    #Add data to train list:
    train_audio.append(y)
    sr_arr.append(sr)

    #Convert the data to mfcc:
    mfccTest =
librosa.feature.mfcc(y,sr)
    mfcc_train.append(mfccTest)
```

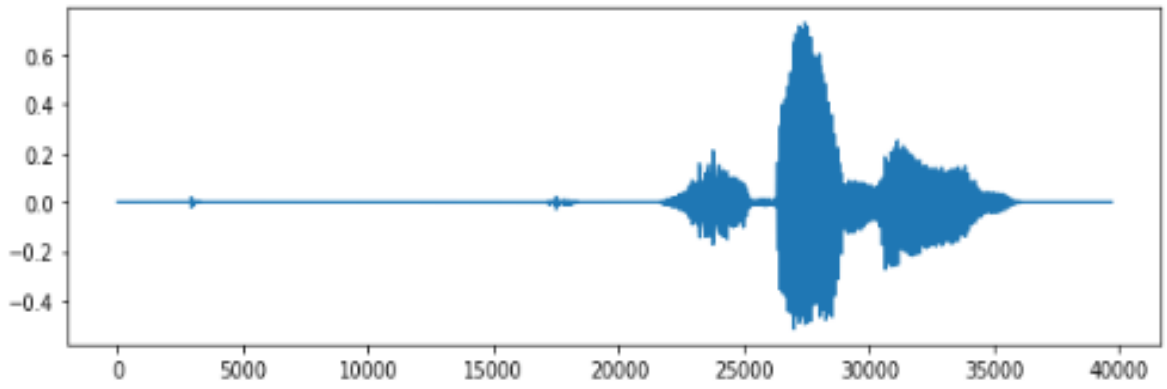
بعد از استفاده از ویژگی ضرایب کپسترال مل و ذخیره آن ها در متغیری از جنس numpy می خواهیم برای درک بهتر از سیگنال های صدا آن هارا نمایش بدهیم.



تمام این اعمال و پردازش ها را روی صدای فرد مقابل هم انجام می دهیم.

```
# add test audio
fig = plt.gcf()
fig.set_size_inches(10.25, 2.95)
yTest, srTest = librosa.load('E:\\speech_recognition\\salam2.wav')
plt.plot(yTest)

#Convert the data to mfcc:
mfccTest = librosa.feature.mfcc(yTest,srTest)
mfccTest = preprocess_mfcc(mfccTest)
```

سیگنال صدای فرد مقابل:

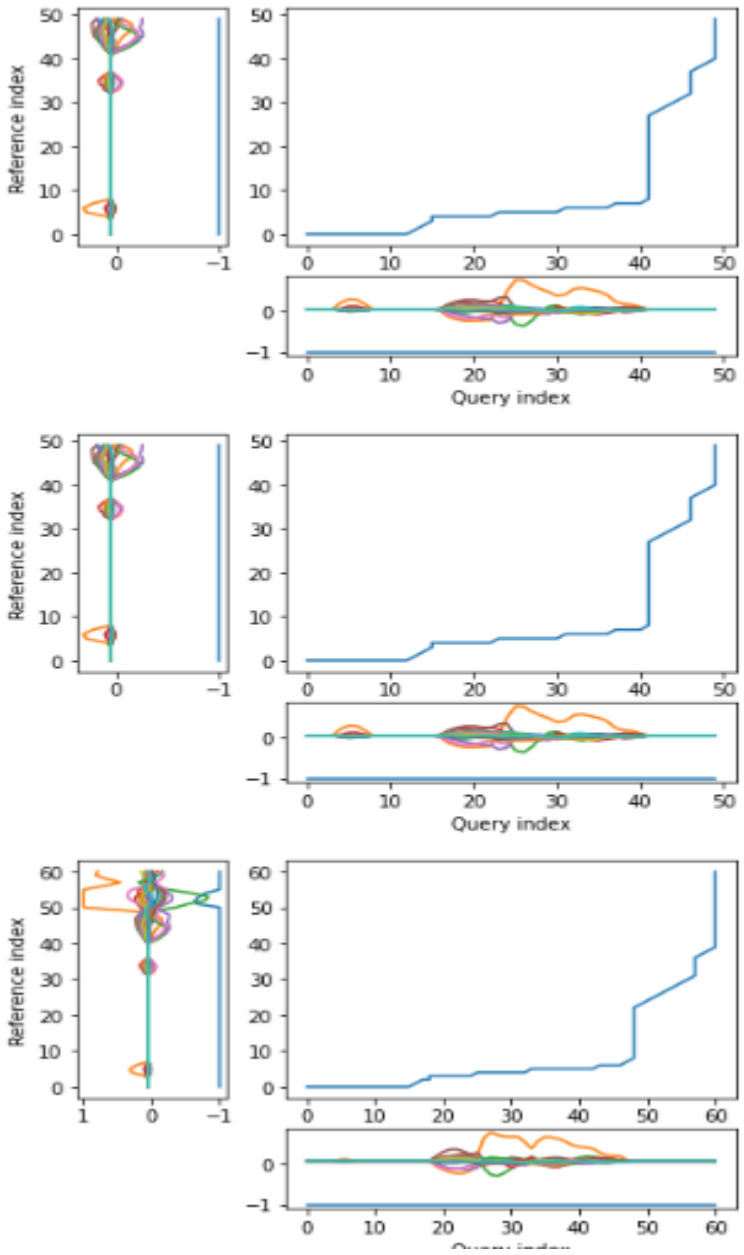


با استفاده از پنجره ای برابر با طول اولین مثال آموزشی و همچنین آزمایشی پنجره می گیریم. برای هر پنجره یک فاصله dtw تا داده های آموزشی را محاسبه می کنیم و پنجره ای با کمترین فاصله تا داده های آموزش انتخاب می کنیم.

ما انتخاب می کنیم که از وزن نمایی به عنوان معیار تشابه در dtw استفاده کنیم.

```
from dtw import *

window_size = []
dists = []
dists_secondary = []
for i in range(len(mfcc_final)):
    window_elemnts = len(mfcc_final[i][-1])
    dist = np.zeros(mfccTest.shape[1] - window_elemnts)
    window_size.append(window_elemnts)
    dists.append(dist)

    for j in range(len(dists)):
        mfcci = mfccTest[:,j:j+window_size[j]]
        dists_temp = dtw(mfcc_final[j].T, mfcci.T,keep_internals=True)
        dists_secondary.append(dists_temp)


    dists_secondary[j].plot(type="threeway")

#plt.plot(dists_temp)

print(dists_secondary[1])
```

با استفاده از همین پنجره گذاری و رسم نمودار مقایسه بین صدا های من و فرد مقابل با استفاده از یکی از متد های dtw.plot می توان مشاهده کرد و نتیجه ای حاصل شود که نشان دهنده شباهت فایل های صدای من با هم دیگه اند و تفاوت کاملا آشکار با صدای فرد مقابل وجود دارد.



برای محاسبه فاصله سیگنال ها با روش dtw از تابع آماده dtw.distance استفاده می کنیم به این شکل که فقط کافی است دو ماتریس مورد نظر را به عنوان پارامتر به آن بدهیم : در ابتدا فاصله سیگنال های صدای خودم را با هم مقایسه کردم:

```
from dtaidistance import dtw
from dtaidistance import dtw_visualisation as dtwvis
import numpy as np
temp = []
for i in range(len(mfcc_final)):
    for j in range(len(mfcc_final[i])):
        temp.append(dtw.distance(mfcc_final[i][i], mfcc_final[i][j]))
    path_between_my_voice = np.array(temp)


path_between_my_voice = np.reshape(path_between_my_voice,(10,20))


print(path_between_my_voice)
```

سپس مقایسه صداهای خودم و فرد مقابل با همین روش:

```
temp = []
for i in range(len(mfcc_final)):
    for j in range(len(mfcc_final[i])):
        temp.append(dtw.distance(mfcc_final[i][i], mfccTest[j]))
    path_between_me_aother_voice = np.array(temp)

path_between_me_aother_voice =
np.reshape(path_between_me_aother_voice,(10,20))


print(path_between_me_aother_voice)
```

Dtw بین 10 صدای خودم با هم:

	1	2	3	4	5	6	7	8	9	10
1	0	8.538	7.276	7.612	7.117	7.772	7.210	7.514	7.252	7.531
2	9.419	0	1.715	1.618	2.106	1.195	2.116	1.791	2.137	1.877
3	8.335	1.691	0	0.922	0.669	1.050	0.735	0.925	0.871	0.961
4	8.800	1.576	0.476	0	0.368	0.297	0.358	0.218	0.363	0.342
5	7.584	2.068	0.450	0.471	0	0.730	0.221	0.397	0.236	0.378
6	8.053	1.195	0.660	0.290	0.706	0	0.639	0.232	0.579	0.489
7	7.397	2.071	0.412	0.410	0.141	0.619	0	0.363	0.232	0.270
8	8.224	1.721	0.381	0.263	0.374	0.255	0.350	0	0.279	0.150
9	6.88	1.858	0.519	0.469	0.357	0.609	0.228	0.304	0	0.224
10	7.299	1.919	0.642	0.374	0.407	0.366	0.390	0.193	0.220	0

Dtw بین 10 صدای خودم و قرد مقابل:

صداهای ضبط شده	1	2	3	4	5	6	7	8	9	10
صدای فرد دیگر	0.617	9.520	8.181	9.099	8.344	9.333	8.509	9.039	8.727	9.095