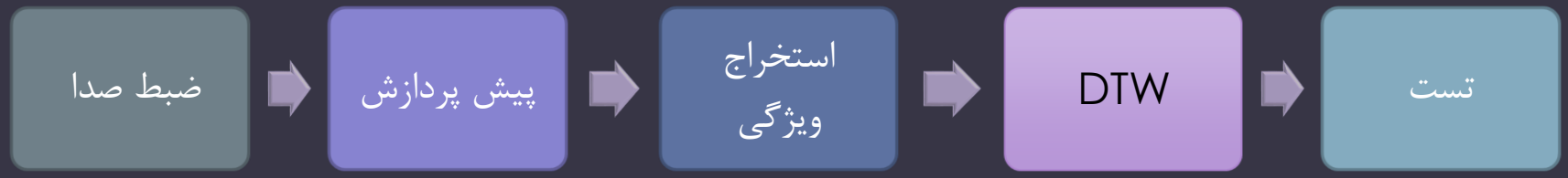
DTW

راضیه غفوری دشت بیاض-۹۸۴۴۰۱۲۹

بهار۱۴۰۲

مقدمه...

- در این پروژه قصد داریم فرایند تایید هویت در پردازش گفتار را اجرا کنیم به اینصورت که مدل آموزشی ما DTW است.
- برای این کار ابتدا باید روی صوت فرایند پیش پردازش را انجام داده و سپس مدل را آموزش دهیم.
- در نتیجه این پروژه به سه بخش کلی پیش پردازش، آموزش و تست تقسیم می شود.



ضبط صدا

- در این پروژه به کمک دو کتابخانه `sounddevice` و `scipy` در زبان پایتون ۱۰ صدایی آموزش از اسم "راضیه غفوری" و ۱۰ صدای متفرقه ضبط و با فرمت .wav ذخیره شد.

```python
import time
import sounddevice as SD
from scipy.io import wavfile

def rec_voice(who,where,name,duratio=3,freq=12000):
    print('Please speak now, I\'m listening... ')
    samples=SD.rec(frames=int(duratio*freq),samplerate=freq,channels=1)
    SD.wait()
    print('Your time is up... ')
    wavfile.write(filename=f'{where}/{who}{name}.wav',data=samples,rate=freq)
    time.sleep(3)


for i in range(10):
    rec_voice("my voice",'my',i+1)

for i in range(10):
   rec_voice("other voice",'other',i+1)
```

پیش پردازش

- در این بخش از فرایند ما سه مرحله به صورت زیر داریم که به ترتیب تمام صدا ها را از این فیلتر ها رد می کنیم تا فرایند پردازش دقیق تر و با نتایج بهتری نمایش داده شود.

پنجره سازی ← فریم بندی ← حذف سکوت

پنجره سازی | فریم بندی | حذف سکوت

- برای حذف سکوت باید ان داده هایی که بین بازه خیلی کمی از فرکانس هستند را پاک کرده و از یک مقدار به بالا در بانک داده قرار داشته باشد. سطح استانه اینکار ۰.۰۰۲ انتخاب شده است.

```
function [output] = remove_silence(input, threshold)
    f_input = abs(fft(input));
    rms_input = sqrt(mean(f_input.^2));
    low_freqs = find(rms_input < threshold);
    f_input(low_freqs) = 0;
    output = real(ifft(f_input));
end
```

پنجره سازی | فریم بندی | حذف سکوت

- در زبان متلب برای فریم بندی از دستور buffer استفاده میشود.
- ما به این تابع صدا و طول ۱۰۰ برای هر فریم و اورلپ ۲۰ را داده ایم و نتیجه تعدادی فریم با طول ۱۰۰ شده است.

```
%framing..............
frames=buffer(sound,100,20);

[m,n]=size(frames);
```

- در زبان متلب تابع hamming را داریم که همان نمودار زنگوله است.
- تابع hamming نمودار زنگوله را به سایز هر فریم ما میسازد
- مرحله بعد این نمودار را در فریم ها ضرب میکنیم پس به تعداد فریم ها، به این نمودار نیاز داریم پس به کمک تابع ریپیت از این نمودار ایجاد کرده و ضرب را انجام میدهیم .

```
%hamming.................
ham=hamming(m);
rep=repmat(ham,1,n);
frames_w=frames.*rep;
```

استخراج ویژگی

- مرحله بعدی استخراج ویژگی هاست که ما برای این پروژه سه ویژگی زیر را استخراج کرده ایم...
  - حوزه زمان -> انرژی – zerocrossing
  - حوزه فرکانس-> MFCC
- پس از استخراج این ویژگی ها min ,max, mean,var را بدست اورده ایم و از این ویژگی های جدید برای اموزش استفاده کردیم.

استخراج ویژگی (ادامه)

```
%creat feather energy , zerocrossing , MFCC.............
energy=ones(1,m);
zero=ones(1,m);

for j = 1:size(frames_w,2)
    energy(1,j)=sum(frames_w(:,j).^2);
end
for j=1 : size(frames_w,2)
    zero(1,j) = size(zerocrossrate(frames_w (:,j)),1);
end

MFCC =mfcc(sound,fs);

% Energy
EnergyMean = mean(energy);
EnergyVar = var(energy);
EnergyMAx = max(energy);
EnergyMin = min(energy);
% zerocrossing
zeroMean = mean(zero);
zeroVar = var(zero);
zeroMax = max(zero);
zeroMin = min(zero);
% MFCC
meanmmel = mean(MFCC);
variancemmel = var(MFCC);
maxmmel = max(MFCC);
minmmel = min(MFCC);
```

استخراج ویژگی (ادامه)

- پس از محاسبه ویژگی ها آنها را در ماتریس ویژگی اجماع میکنیم.
- در متلب از دستور cat برای این کار استفاده میکنیم.

```
%Construction of feature matrix.................
feathers=cat(2,EnergyMin,EnergyMAx);
feathers=cat(2,EnergyVar,feathers);
feathers=cat(2,EnergyMean,feathers);
feathers=cat(2,zeroVar,feathers);
feathers=cat(2,zeroMax,feathers);
feathers=cat(2,zeroMin,feathers);
feathers=cat(2,zeroMean,feathers);
feathers=cat(2,meanmmel,feathers);
feathers=cat(2,variancemmel,feathers);
feathers=cat(2,maxmmel,feathers);
feathers=cat(2,minmmel,feathers);
```

DTW

```
my_thershold1_2=dtw(feathers1,feathers2);
my_thershold1_3=dtw(feathers1,feathers3);
my_thershold1_4=dtw(feathers1,feathers4);
my_thershold1_5=dtw(feathers1,feathers5);
my_thershold1_6=dtw(feathers1,feathers6);
my_thershold1_7=dtw(feathers1,feathers7);
my_thershold1_8=dtw(feathers1,feathers8);
my_thershold1_9=dtw(feathers1,feathers9);
my_thershold1_10=dtw(feathers1,feathers10);


thersholds=cat(2,my_thershold1_10,my_thershold1_9);
thersholds=cat(2,my_thershold1_8,thersholds);
thersholds=cat(2,my_thershold1_7,thersholds);
thersholds=cat(2,my_thershold1_6,thersholds);
thersholds=cat(2,my_thershold1_5,thersholds);
thersholds=cat(2,my_thershold1_4,thersholds);
thersholds=cat(2,my_thershold1_3,thersholds);
thersholds=cat(2,my_thershold1_2,thersholds);


thershold_T=mean(thersholds);
```

- در زبان متلب یک تابع برای اینکار وجود دارد که با دادن ماتریس ویژگی به ان مقدار سطح استانه را به ما میدهد.
- ما با دادن دو ماتریس ویژگی یک مقدار استانه بدست می آوریم در نتیجه تمام ماتریس ها را باهم مقایسه میکنیم و با میانگین گیری به مقدار سطح استانه نهایی دست پیدا میکنیم.
- از این سطح استانه برای تست سایر ورودی هااستفاده میشود.

تست

```
new_my_voice=new_dtw('new/new voice.wav');
Pattern_voice=new_dtw('my/my voice8.wav');
thershold=dtw(Pattern_voice,new_my_voice);

mint=thershold_T-5;
maxt=thershold_T+5;

disp(thershold_T)
disp(thershold)

if thershold<maxt
    if thershold>mint
        disp("Raziye ghafoori")
    end
end
if thershold>maxt
    disp("I dont no who you are !!!")
end
```

- فرایند به این صورت است که با محاسبه مقدارdtw صدایی جدید و مقایسه این عدد با سطح استانه پی به شباهت انها میبریم.
- اگر این دو عدد به هم نزدیک باشد پس صدا متعلق به یک نفرست.

نتایج خروجی...

- ما ویس جدید را با این چهار ویس به رندوم مقایسه کردیم در این ویس راضیه غفوری اسم خود را گفته است که در مقایسه با صداهایی گذشته او، یکسان بودن شخص تایید میشود...

Name voice	How you are??
my voice8	Raziye ghafoori
my voice6	Raziye ghafoori
other voice7	I dont no who you are !!!
Other voice3	I dont no who you are !!!