Second part of report on the final project of the Digital Speech Processing

Hossein Simchi, Ali Yazdani, Dr. Yasser Shekofteh
Computer Science and Engineering , Shahid Beheshti University of Tehran , Iran
[h.simchi@mail.sbu.ac.ir]

February 2, 2021

Abstract

In the second part, we try to give a complete description of the project in various stages. Also in each stage we try to express the explanations related to that stage in very short sentences

1 Automatic SER using recurrent neural networks with local attention

To implement the algorithm described in the article mentioned, we must perform the steps stated at the time of presentation, ie extracting the feature and then categorizing them correctly.

1.1 Steps and How the code works

1) Specify the name of the dataset, dataset type = 'shEMO'
2) **Call dataset class** : The class of the data is called, and we give it the name and path of the data.
3) **dataset calss** : When we call the dataset class, we also need to get a label for our data. The fourth letter of our data name also specifies their label. In this project we have used the Librosa library to read audio files.
4) Then, using the **cPickle command**, we put each data along with its label inside **'shEMO db.p'**
5) **Feature Extraction Function** : Using the cPickle command, we read the data set stored in the previous step and pass it to the feature extraction function to extract the feature.
We will discuss feature extraction functions in detail in a separate section, and here we will discuss the general process of written code.
6) **feature extract function** : We do this for every data in the database
- The class of the stFeatureExtraction is called.

- **Harmonic ratio and pitch** : Using stFeatureSpeed Function and then Then we add its output to the feature vector obtained from the previous step.
- **Z-normalized** : We normalize the feature vector
- Add the obtained feature vector to the total feature vector.

In fact, in this step we try to get its features for each data and add it to the total Feature vector.
Finally, we put the total feature vector inside **'shEMO features.p'** using the **cPickle command** to use it later. Our final vector contains a number of feature vectors, which is equal to the number of data in the dataset.

1.2 stFeatureExtraction

This function implements the shor-term windowing process. For each short-term window a set of features is extracted.This results to a sequence of feature vectors, stored in a numpy matrix.
ARGUMENTS :
- signal: the input signal samples
- Fs: the sampling freq (in Hz)
- Win: the short-term window size (in samples)
- Step: the short-term window step (in samples)

RETURNS :
- stFeatures: a numpy array (numOfFeatures x numOfShortTermWindows)

1.2.1 stFeatureExtraction Steps

1) Signal normalization
2) Compute total number of samples
3) Compute the triangular filter banks used in the **mfcc** calculation

We must note that the final vector of features is equal to the sum of TimeSpectralFeatures and ChromaFeatures. For each window we have to do the following calculations :
- Get current window : x = signal[curPos:curPos+Win]
- Update window position
- get Fast Fourier Transform (FFT) magnitude : X = abs(fft(x))
- Normalize FFT
- To extract the Spectral Flux feature, we need to save the Fast Fourier Transform (FFT) magnitude of the previous frame as well.
- We need to define a list named **'curFV'** with the same number of rows as the number of attributes and one column.

Now, we extract the following features and enter the features in the list of Hussein from zero to the end, respectively
1) Zero crossing rate : stZCR Function
2) Short-term energy : stEnergy Function
3) Short-term entropy of energy : stEnergyEntropy Function
4) Spectral centroid and spread : stSpectralCentroidAndSpread Function
5) Spectral entropy : stSpectralEntropy Function

6) Spectral Flux : stSpectralFlux Function
7) spectral Rolloff : stSpectralRollOff Function
8) MFCC : stMFCC Function
9) Chroma : stChromaFeatures
As a result, we have extracted nine features for each window.

1.2.2 Feature Extraction Functions

Zero crossing rate (stZCR Function)
- applying the sign function to each window.
- calculating the difference in value at each point from the previous point (Numpy.diff())
- calculating the absolute value of the values obtained
- Return the sum of all the resulting values as the final value
Zero crossing rates are also used for Voice activity detection (VAD), which determines whether human speech is present in an audio segment or not.
Short-term energy (stEnergy Function)
Sum values of all frequencies in each window to the power of two and then divide by the number of frequencies in each window.Thus short term energy can be used for voiced, unvoiced and silence classification of speech
Short-term entropy of energy (stEnergyEntropy Function)
To compute entropy of energy. Spectral entropy features, used for voice activity detection and the entropy is a measure of disorganization and it can be used to measure the peakiness of a distribution.
Spectral centroid and spread (stSpectralCentroidAndSpread Function)
To compute spectral centroid and spread of frame (given abs(FFT))
The spectral centroid is a measure used to characterise a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound. Noise-like signals have usually a large spectral spread, while individual tonal sounds with isolated peaks will result in a low spectral spread
Spectral entropy (stSpectralEntropy Function)
To compute the spectral entropy.
Spectral Flux (stSpectralFlux Function)
To Compute the spectral flux feature of the current frame.
ARGUMENTS:
- X: the abs(fft) of the current frame
- Xpre: the abs(fft) of the previous frame
spectral Rolloff (stSpectralRollOff Function)
To computee spectral roll-off
MFCC (stMFCC Function)
To compute the MFCCs of a frame, given the fft magnitude.
ARGUMENTS:
- X: fft magnitude abs(FFT)
- fbank: filter bank (mfccInitFilterBanks Function)

RETURN:
- ceps: MFCCs (13 element vector)
Note that to calculate the MFCCs Features, we must obtain the MFCC filter bank and DCT matrix.
Chroma (stChromaFeatures)
To Compute the Chroma feature of the current frame.

1.3 Create Model (Bidirectional LSTM)

After extracting the desired features and then saving the features in different files (according to what was stated in the previous section), in this section, the model is designed to be trained on the data collected from audio files.

1.4 Prediction on Model

After training the model, in this step we need to give data to the trained model in the previous section, to receive the desired label.