101804

Speech Processing and Recognition

Project Report

Speech emotion recognition

Author :
Arnaud Ruymaekers
S5298338

Course instructors:
Vito Paolo Pastore
Zied Mnasri
Stefano Rovetta

Contents

Chapter 1

Intro

For this end of course project, the goal was the create a system to automatically detect the emotion of a spoken audio file. The emotion was to be detected from the basis of the 6 basic emotions (anger, joy, fear, disgust, sadness and surprise) along with a neutral emotion. This was to be done using Low Level Descriptors (LLD) of the audio file. These LLD's could then be fed in a auto-encoder network to extract an embedding with could finally used to train a Deep Neural Network classifier.

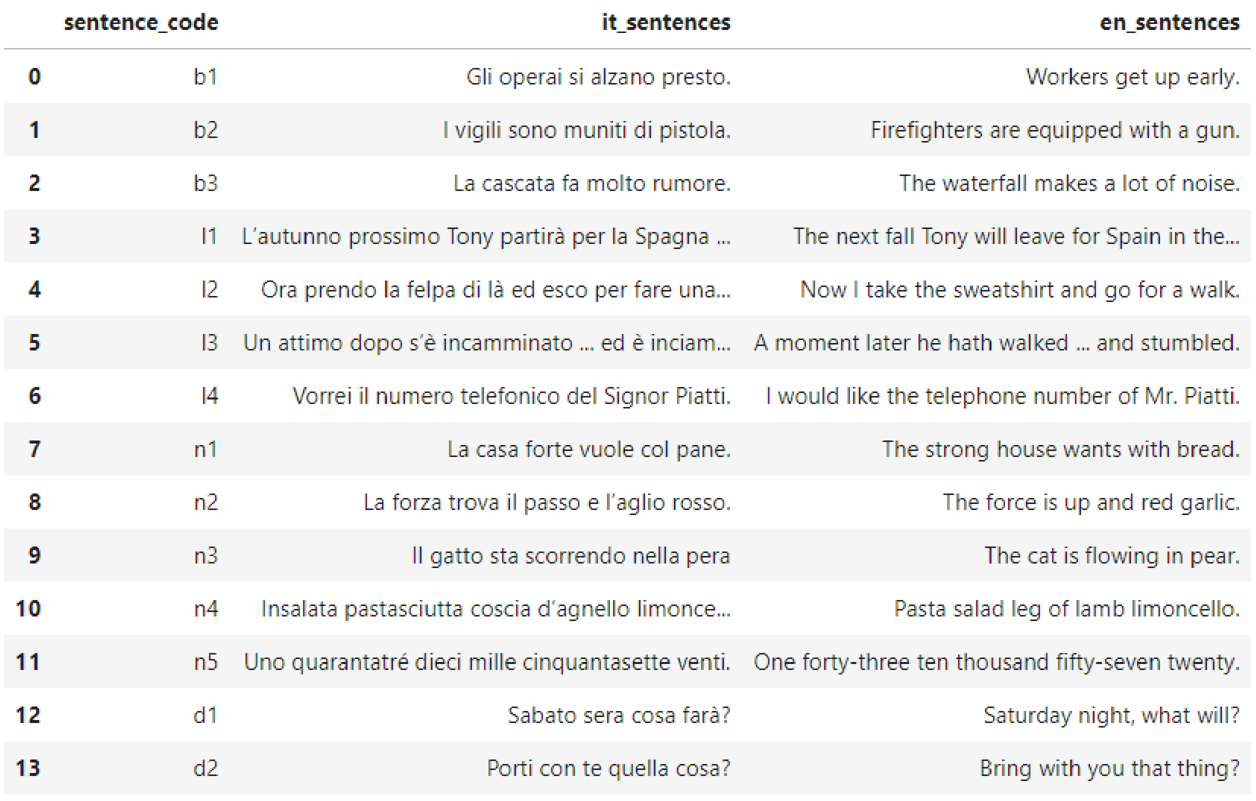
In the following pages, I will first describe the dataset used as source to the network. Following which I will describe the various ways I tried to train an auto-encoder in order to have a feature embedding. Then, I will explain the classifiers I tried and how they performed. And finally, I will reflect on the results of the various experimentation and describe the final pipeline to detect emotions from an audio file.

Chapter 2

Dataset

2.1 EMOVO

The dataset chosen for this project was the EMOVO dataset, which is made of 588 recordings. The recordings are made, in Italian, by 3 male and 3 female actors, each speaking 14 different short sentences, and in each of the emotions:

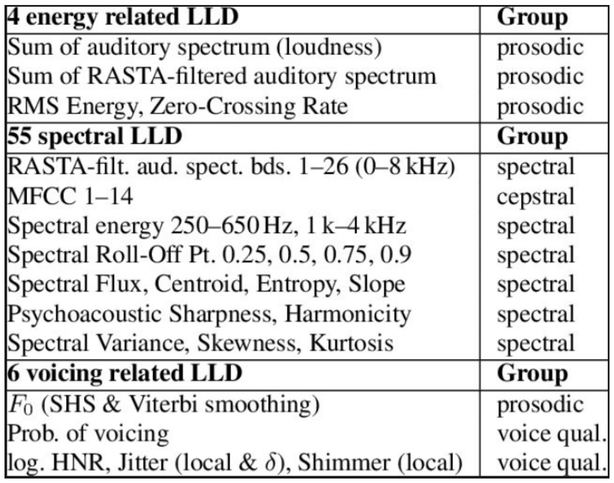


	sentence_code	it_sentences	en_sentences
0	b1	Gli operai si alzano presto.	Workers get up early.
1	b2	I vigili sono muniti di pistola.	Firefighters are equipped with a gun.
2	b3	La cascata fa molto rumore.	The waterfall makes a lot of noise.
3	l1	L'autunno prossimo Tony partirà per la Spagna ...	The next fall Tony will leave for Spain in the...
4	l2	Ora prendo la felpa di là ed esco per fare una...	Now I take the sweatshirt and go for a walk.
5	l3	Un attimo dopo s'è incamminato ... ed è inciam...	A moment later he hath walked ... and stumbled.
6	l4	Vorrei il numero telefonico del Signor Piatti.	I would like the telephone number of Mr. Piatti.
7	n1	La casa forte vuole col pane.	The strong house wants with bread.
8	n2	La forza trova il passo e l'aglio rosso.	The force is up and red garlic.
9	n3	Il gatto sta scorrendo nella pera	The cat is flowing in pear.
10	n4	Insalata pastasciutta coscia d'agnello limonce...	Pasta salad leg of lamb limoncello.
11	n5	Uno quarantatré dieci mille cinquantasette venti.	One forty-three ten thousand fifty-seven twenty.
12	d1	Sabato sera cosa farà?	Saturday night, what will?
13	d2	Porti con te quella cosa?	Bring with you that thing?

Figure 2.1: Table of sentences in Italian and English

2.2 LLD extraction

Low Level Descriptors are characteristics that can computed from an audio signal. In this projects, the Low Level Descriptors of the audio files were extracted using OpenSMILE with the feature set "ComParE" that is made of 65 LLD's:



4 energy related LLD	**Group**
Sum of auditory spectrum (loudness)	prosodic
Sum of RASTA-filtered auditory spectrum	prosodic
RMS Energy, Zero-Crossing Rate	prosodic
55 spectral LLD	**Group**
RASTA-filt. aud. spect. bds. 1–26 (0–8 kHz)	spectral
MFCC 1–14	cepstral
Spectral energy 250–650 Hz, 1 k–4 kHz	spectral
Spectral Roll-Off Pt. 0.25, 0.5, 0.75, 0.9	spectral
Spectral Flux, Centroid, Entropy, Slope	spectral
Psychoacoustic Sharpness, Harmonicity	spectral
Spectral Variance, Skewness, Kurtosis	spectral
6 voicing related LLD	**Group**
F_0 (SHS & Viterbi smoothing)	prosodic
Prob. of voicing	voice qual.
log. HNR, Jitter (local & δ), Shimmer (local)	voice qual.

Figure 2.2: LLDs in ComParE

The LLD's are extracted over very small range of the audio signal. This yields a sequence of length depending on the length of the audio segment and with 65 dimensions.

2.3 Data preparation

To be able to feed that to neural network, the various LLD sequences have to be standardized. For this I first limited the length to double the mean length, so that a minimal amount of sequence would need to be cropped but the overly long sequence are cropped down. Additionally the sequence are zero filled when under the limit set.

Then normalization is done, per LLD, we compute the mean and standard deviation to bring all the values around 0 with a standard deviation of 1.

Chapter 3

Feature embedding

To achieve feature embedding, I tried to train auto-encoder networks. These kind of network are build with two parts, an encoder and a decoder. The encoder compresses the data to a more compact form. The decoder part is there to make sure that from the compressed data, we can reconstruct the input.

In the case of this project, the goal would be for the sequences of LLD sequences to be compressed 1D vector.

3.1 LSTM auto-encoder

For the first auto-encoder I tried to build a LSTM network as it accounts for the temporal aspect of the sequences. I built the network to encode to a 400 node vector.

Unfortunately, this network did not perform well at encoding the sequence of LLD's. I believe this is due to fact that there is not a general way of compressing a sequence and being able to recreate it since all the samples are clearly different, there is no real temporal patterns to be learned.

3.2 Convolutional auto-encoder

Following this I tried to build a convolutional network , that would therefore not care about the temporality of sequence but that would still acknowledge that LLD next to one another temporally are related. This was done by having multiple convolutional layers with 1D kernel, in the direction of the time, so not mixing LLD's. And for the decoder we use transpose convolutional layersto re-increase back to the normal size.

But, once again, this did not yield satisfying results. Most likely due to the fact, the whole audio's LLD's can't be compressed as a single vector with no temporality.

3.3 Deep auto-encoder

As a final resort, I attempted to build a simple, flat, deep neural network where the 2D input is directly flattened and reshaped at the end. In the middle, we have fully connected layers decreasing in size down to a length of 500. Followed by decoder layers that increase the size back to size required for reshaping in the original form.

This also did not perform up to expectation, as the middle layer probably becomes too general to be able to reconstruct the inputs from the compressed version.

Chapter 4

Classifier

To classify the different audio sequences I attempted to build different model. All model have been based on the sequences of LLD's as making a feature embedding was unsuccessful.

4.1 Deep Neural Network Classifier

This network was built by first using multiple convolutional layers to compress the temporal aspect of the sequences to at the end flatten the rest and feed it in a fully connected 3 layer classifier.

With this approach, I managed to get around 60% accuracies.

4.2 Other Classifiers

Other classifiers I tried out are more classical ones such as Logistic regression (multi-class version) SGD or Random Tree. But they performed less good than the Neural Network described above.

Chapter 5

Functionals dataset

Another approach that was explored was to use an array of functionals based on the ComParE feature set to represent the audio files. This allowed to not have to deal with the temporal aspect and the array was representing the characteristics over the whole audio file.

Following an auto-encoder was trained to embed the feature in a lower dimension vector. The auto-encoder was a simple deep neural network with a bottleneck layer of 100 nodes.

The embedded data was then fed in another deep neural network classifier.

With this method I managed to get accuracies closer to 70%.

Chapter 6

Conclusion

6.1 Results

Overall the results were a bit underwhelming as I was expecting the accuracies going up to 80%. With a better auto encoder for the LLD's, I believe this would have been achievable. So in the end the performance of the classifier was not what was holding back.

6.2 Final product

The final product is a function that from the path of an audio file performs the following operations:

1. Extract the audio signal
2. Extract the LLDs sequence from the signal
3. Normalize the sequence data
4. Predict the emotion using a save model
5. Return the two most likely emotions