

Contents

[Contents 1](#_30j0zll)

[1. Introduction 2](#_1fob9te)

[2. Create dataset 2](#_3znysh7)

[2.1. Train dataset 2](#_2et92p0)

[2.2. Test dataset 3](#_tyjcwt)

[3. Folder structure 3](#_3dy6vkm)

[4. Download captions and audio files 3](#_1t3h5sf)

[5. Sentiment classification for negative, neutral and positive sentiments 4](#_4d34og8)

[5.1. Srt caption file 4](#_2s8eyo1)

[5.2. Captions data Preprocessing 4](#_17dp8vu)

[5.3. Sentiment classifier 4](#_3rdcrjn)

[5.3.1. Vader 5](#_26in1rg)

[5.3.2. Text blob 5](#_lnxbz9)

[5.3.3. Pattern.en 5](#_35nkun2)

[5.3.4. Final sentiment decision 5](#_1ksv4uv)

[6. Audio-based emotion recognizer 6](#_44sinio)

[6.1. Audio data augmentation 8](#_z337ya)

[6.2. Audio data preprocessing 8](#_3j2qqm3)

[6.3.](#_1y810tw) SVM classifier 8

[6.4. Model performance 8](#_4i7ojhp)

[6.5. Evaluate the ground-truth 9](#_2xcytpi)

[6.6.](#_1ci93xb) F1 score 9

[6.7.](#_3whwml4) Confusion matrix 9

[7.](#_2bn6wsx) Evaluate the performance of the audio-based emotion recognizer 9

[8. Repository of the project and management tool 9](#_qsh70q)

[8.1. Repository 9](#_3as4poj)

[8.2. Project management tool 9](#_1pxezwc)

[References 10](#_49x2ik5)

# 1. Introduction

At this assignment we will create a model that used for modal speech emotion recognition on YouTube interviews. As first step we crawled videos that contain English dialogs from YouTube. After that, 3 sentiments classifiers used to recognize negative, neutral and positive sentiments. The most confident of these decisions (negative, neutral and positive sentiments) used to train an audio-based emotion recognizer. As next step, this trained audio model will be used to classify other testing videos. At the end we are going to evaluate the model by defining the ground truth. A subset of the audio classifier outcomes, specifically 100 samples per each class negative, neutral and positive were evaluated by listening them and put a sentiment tag. After these 2 different tags are compared.

# 2. Create dataset

## 2.1. Train dataset

The dataset created by us, choosing 35 random interviews from YouTube. A csv file “dataset.csv” has been created and contains information about these videos.

The format of csv file is:

*Id,URL,Title,Audio,Captions,Polarity,Pickle*

For example:

1,https://www.youtube.com/xxx,myvideo,audio/1.wav,subtitles/1.srt,polarity\_csv/polarity\_1.csv,pickle\_lists/polarity\_1.p

|  |  |
| --- | --- |
| Id | 1 |
| *URL* | https://www.youtube.com/xxx |
| *Title* | myvideo |
| *Audio* | *audio/1.wav* |
| *Captions* | *subtitles/1.srt* |
| *Polarity* | *polarity\_csv/polarity\_1.csv* |
| *Pickle* | *pickle\_lists/polarity\_1.p* |

More specifically, each video has a unique id. All the related files with that video will be saved with the name of videoId in the specific subfolders. At each entry of csv file is also defined the path of all related files for this video: path where audio file located, path for the subtitles, path for the results of captions’ sentiment analysis.

## 2.2. Test dataset

Following the same procedure, we have also created a dataset for testing proposes. The testing dataset has the same format as training dataset and contains 11 videos. Its name is “*dataset\_test.csv*”

# 3. Folder structure

Running the scripts several folders will be created. Specifically, the folder structure of the project is described here.

There will be two subfolders: “train” and “test”. Each of them will have a subfolder “audio” that will contain all the mp3/wav files which downloaded from *Downloader.py*. Inside audio folder will also be created 3 subfolders, negative, positive, neutral that will contain audio segments for each sentiment sentiments. Also, there will be a folder “subtitles” at which will be contained all srt (files with captions from the videos).

Inside the train and test subfolder, as well there will be also a folder “polarity\_csv” will be created that will contain a csv file for each video. Each csv file will have as name the videoId and will contain the segments intervals and the result of the speech emotional sentiment classifier. This info will be also saved as pickle file at the subfolder “pickle\_lists”. These files used form the audio classifier.

# 4. Download captions and audio files

The downloading functionality is implemented at the script downloader.py. For crawling YouTube videos, youtube-dl has been used.

*Downloader.py* → Download the captions and audio file for the videos contained at the training and testing dataset. A video can’t be at the same time on training and testing dataset. At the path “*/Download\_Functions*” there are 3 python scripts that implement this functionality. Text and audio files of training and testing dataset are stored in different location - *train* subfolder and *test* subfolder-.

* *Download\_Audio.py* → download mp3 file for each video and convert them to wav files
* *Download\_Subtitles.py* → download subtitles for each video and convert .vtt format of the downloaded file to .srt format
* *Files\_And\_Dirs.py* → utility functions: create folder, change path, read txt file, read csv file, etc

# 5. Sentiment classification for negative, neutral and positive sentiments

## 5.1. Srt caption file

The next step is to classify the captions of each video as negative, positive or neutral based on the text. The .srt file of each video has the below format. Firstly, is defined the segment’s duration and after the text.

*00:00:05,900 --> 00:00:07,999*

*This is a subtitle for the course multimodal Analysis*

*00:00:10,000 --> 00:00:14,000*

*This is another subtitle for the course multimodal Analysis*

*00:00:15,000 --> 00:00:15,100*

*[Laughing]*

*00:00:16,000 --> 00:00:17,000*

*[Laughing] And guess, this is another subtitle*

## 5.2. Captions data Preprocessing

Captions usually contains tags like: cheering, laughing etc. We have excluded the segments that has no actual text and contain only tags. Tags that are included in sentences are not excluded but considered on the sentiment analysis. Also, we have excluded segments whose duration is less than 2 secs. As our task it to recognize the sentiment of an audio, very small segments have no valuable information.

## 5.3. Sentiment classifier

Since we have cleaned the data, the next step is to use some sentiment classifiers for classifying the captions into 3 sentiments (positive, negative, neutral). Vader lexicon, text blob and pattern.en module have been used for that. This functionality is on script Parser\_caption.py.

*Parser\_caption.py* →sentiment analysis on segments based on captions context. Hold just the relative segments. At the path “*/Caption\_Functions*” there are 3 python scripts that implement this functionality and offer some other utility functions.

* File\_Functions.py→ utility functions: create folder, create csv, create pickle file, etc
* Parse\_Functions.py→ empty ????
* Sentiment.py→ Sentiment analysis with 3 different sentiment classifiers. Hold just the relative segments for further analysis.

### 5.3.1. Vader

VADER produces four sentiment metrics. The first three, positive, neutral and negative, represent the proportion of the text that falls into those categories. The final metric, the compound score, is the sum of all of the lexicon ratings which have been standardised to range between -1 and 1 and shows how strong is the recognized sentiment. Positive score means positive sentiment, negative means negative sentiment and zero means neutral sentiment.

### 5.3.2. Text blob

The [sentiment](https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.TextBlob.sentiment) property of text blob returns a tuple of (polarity, subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. Positive polarity means positive sentiment, negative means negative sentiment and zero means neutral sentiment.

### 5.3.3. Pattern.en

The pattern.en module contains a fast part-of-speech tagger for English sentiment analysis. Positive result means positive sentiment, negative means negative sentiment and zero means neutral sentiment.

### 5.3.4. Final sentiment decision

We choose to create a model with high precision, for that reason we took in mind only the segments that we were somehow “sure” about their sentiment (positive, negative, neutral). We have excluded segments at which the above sentiment classifiers had different results. After, as final score of sentiment we decided to use the average outcome of the 3 classifiers for each segment. Positives and negatives segments whose polarity is less than 0.25 (absolute value), will be rejected. Only these segments will be used at the next step of audio analysis, segment with different polarities based on 3 sentiment classifiers excluded.

We have also excluded segments whose duration is less than 2 seconds as there were many segments with very few duration and no valuable information. Most of them were noise.

The outcome of each video will be saved at a csv file located at */polarity\_csv*/ with the name polarity\_videoId.csv. At this file will be written the several segments duration [strat- path */pickle\_lists/* with the name *polarity\_videoId.p.* These files will be used from the audio classifier at the next step. The subfolders of /polarity\_csv and */pickle\_lists/* are inside tarin and test folders.

For example, the context of the file polarity\_1.csv :

*"00:00:51,319",0.10695*

*"00:00:58,930",0.67985*

*"00:01:04,640",0.43745*

*"00:01:17,030",-0.1839838383838384*

*"00:01:20,690",0.0*

*"00:01:28,700",-0.446825*

*"00:01:32,240",0.0*

*"00:01:36,040",0.0*

*"00:01:46,700",0.0*

So, for example from the video 1.srt that initially has 100 segments we hold just the 9 segments that are written above. The other 81 segments either their duration was less than 2 seconds, or the 3 sentiment classifiers didn’t have the same outcome. For these 9 segments, 3 are positive, 2 negative and 4 neutral.

# 6. Audio-based emotion recognizer

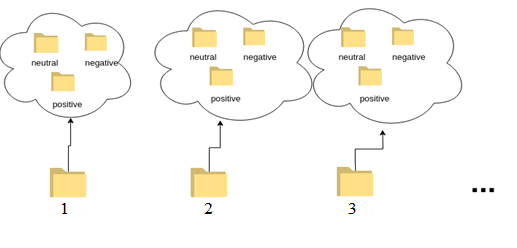
The next step is to train an audio-based emotion recognizer. The audio classifier is on script Parser\_audio.py

*Parser\_audio.py* →Train an audio svm classifier with the outcomes of captions’ sentiment analysis of training dataset. Split the audio in the related segments and create folders with positive, negative and neutral segments. At the path “*/Audio\_Functions*” there are 3 python scripts that implement this functionality.

* File\_Functions.py → utility functions: create folder, remove folder, read pickle file, etc
* Parse\_Functions.py→ Make audio segmentation based on the extracted segments s form sentiment analysis. Wav segmentation is done with: ffmpeg.

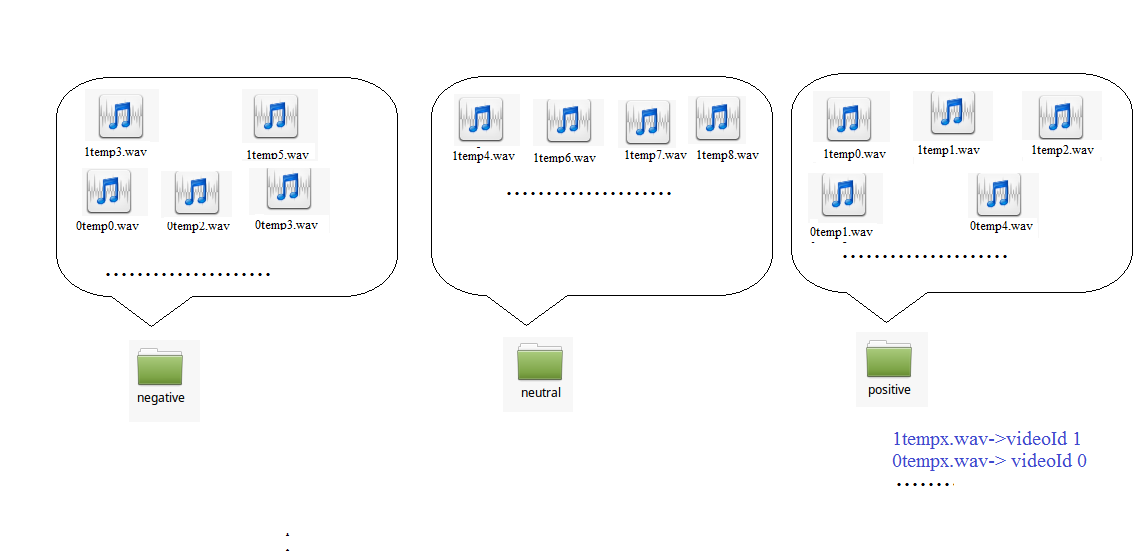
* FtrainTest.py→ use functions from from pyAudioAnalysis [1]. Using the Video IDs as explained above we select the videos for train and test. When the videos are selected we call dirsWavFeatureExtraction of pyaudio analysis to do audio sentiment classification.Features set are normalized to 0-mean and 1-std, to avoid any outliers (too big or too small values). Performing Cross Validation we select the best C parameter for the SVM classifier based on the best fold. With the best C we fit the whole test set with the SVM classifier and save the trained model at a pickle file in order to use it on test data. Negative, Positive and Neutral classes become balanced using SMOTE from *imblearn* library[2].

More specifically, each wav from the train dataset is splitted in the segments that are referred at the pickle file of this video. Only the segments that are referred at the pickle file will be used. The name of the segment will be a substring of videoID and an increased integer eg: 0temp1.wav. After, a folder per VideoID will be created. Each VideoID folder will contain the corresponding wavs per sentiment separated in folders (Positive, Neutral, Negative). This approach will help us during the *K-Fold validation* and *Leave one out* approach.



For each selected video ID, the segments based on the label are copied in the corresponding folder in the Train or Test folder as below. We would like to split the dataset based on videoId and not randomly on segments. This approach will lead to the generalization of the classifier. We used the name of each wav file to do it, as all the segments that come from a specific video its name is a substring of the video Id. Generally, All the segments from one video should be totally on the train or on the test dataset.

The subfolder of positive, sensitive, neutral look like the below graph:



## 6.1. Audio data augmentation

To deal with unbalanced data that occur due to the nature of the task, SMOTE[2] from *imblearn* library is used. SMOTE will balance the dataset for training and generate equal distribution of all classes. We would like balanced data to avoid the overfitting or avoid any dominance of the class with the majority of the samples. As we have observed from the collected data, the most frequent sentiment is the neutral, so neutral class will have more segments than the other 2 classes and as a result the classifiers would be more biased to neutral sentiments. Data augmentation process is followed on training data.

## 6.2. Audio data preprocessing

Before the training of the model, training dataset will be also normalized to avoid any outliers (too big or too small values).

## 6.3. SVM classifier

SVM (Support Vector Machine) classifier used as audio classifier, other classifier can be used also. svm\_train\_evaluate contains the functionality of the classifier. We seek the best hyper parameter for this based on the best F1 score and we fitted on it the whole train dataset. For the classifier with the best results a confusion is generated and a plot with F1,recall and accuracy for each class using plotly and plotly\_classification\_results function. The model will be saved as pickle under name svm3classes file for future train or test purposes.

## 6.4. Model performance

We decided to use F1 score as performance measure of the model. F1 is a function of Precision and Recall and used when we would like to have balance between Precision and Recall. F1 score is more relative as accuracy can be largely contributed by a large number of True Negatives whereas F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

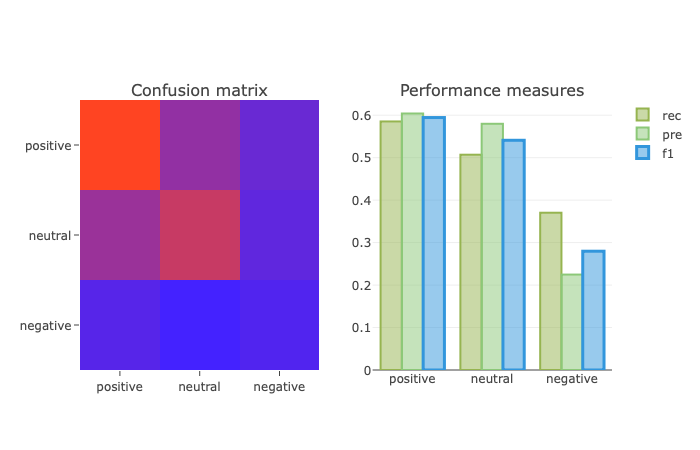
In our model we decided to create a model with high precision and low recall, means that for the train data we have used we are somehow sure about each polarity. At the phase of sentiment analysis, the selection of the segments that will be used for training was very strict. We didn’t include segments with different polarities from the different sentiment classifiers or segments with very small duration for example.

## 6.5. Evaluate the ground-truth

Training dataset as well test datasets are created by us, crawling random videos from YouTube using youtube dl and as a result there is no ground truth about the sentiments expressed at each video. To create a ground truth and evaluate the performance of the final model we listened 100 random segments from each sentiment in order to confirm its polarity. Finally, we used these results to calculate some performance metrics

## 

## 6.6. Confusion matrix and F1 score

For the Train set the results are above:

The color scale from red #4422ff (0) to blue (1) #ff4422 .

For the Test dataset are:

# 7. Evaluate the performance of the audio-based emotion recognizer

To evaluate the performance of the classifier we crawled exactly as in train set videos from Youtube. The videos are being pre proceed using word\_tokenize. We keep the segments with more than 4 words. This approach helped us in the selection of the sentiment segment. Videos without small duration or containing music, laughing are not useful for our task. After listening all segments we split all segments we separate the segments in folders positive/neutral/negative. In order to test our classifier we use final function from FtrainTest.py. Finally we load the model svm3classes from the train phase with load\_model function from FtrainTest.py. As in the train phase the features are being extracted from the Test segments. We normalize the features and split them in x\_test, y\_test and predict the values with the svm3classes classifier.Using Sklearn metrics for F1 , accuracy, confusion matrix and plotly final results are shown in the output.

# 8. Repository of the project and management tool

## 8.1. Repository

The source code of the project is here: <https://github.com/salevizo/multimodal_audio.git>

## 8.2. Project management tool

As project management tool we used Trello. The url for our board is:

https://trello.com/b/dTThH7Ep/project3

# References

[1] <https://github.com/tyiannak/pyAudioAnalysis>

[2]<https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html>