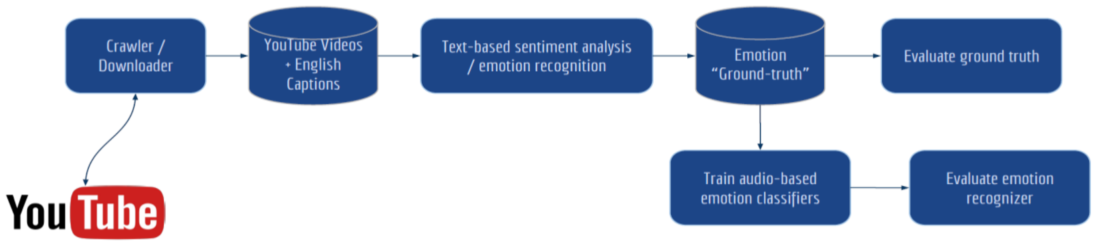
Multimodal Information Processing and Analysis

P3 Large – scale cross – modal speech emotion recognition



Crawler / Downloader
YouTube Videos + English Captions
Text-based sentiment analysis / emotion recognition
Emotion "Ground-truth"
Evaluate ground truth
Train audio-based emotion classifiers
Evaluate emotion recognizer
YouTube

Alevizopoulou Sofia 2022201704002

Avgeros Giannis 2022201704003

Tsiatsios George 2022201704024

Msc Data Science

Athens 2019

Contents

Table of figures

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 125 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 containing 10 random videos. The testing dataset has the same format as training dataset. Its name is "*dataset_test.csv*".

3. 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

4. Sentiment classification for negative, neutral and positive sentiments

4.1. 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

4.2. 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.

4.3. 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
- Sentiment.py→ Sentiment analysis with 3 different sentiment classifiers. Hold just the relative segments for further analysis.

4.3.1. Vader

VADER produces four sentiment metrics. [4] 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.

4.3.2. Text blob

The sentiment property of text blob returns a tuple of (polarity, subjectivity). [5] 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.

4.3.3. Pattern.en

The pattern.en module contains a fast part-of-speech tagger for English sentiment analysis. [6] Positive result means positive sentiment, negative means negative sentiment and zero means neutral sentiment.

4.3.4. 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.

5. Audio-based emotion recognizer

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

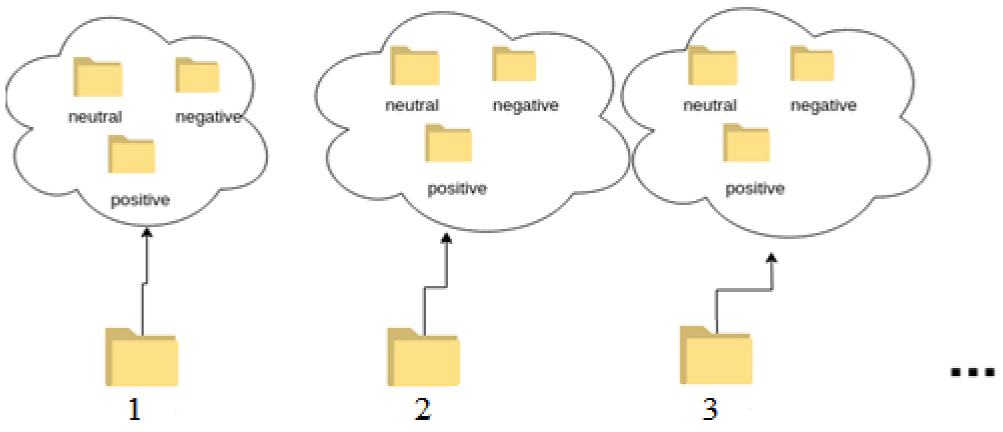
Parser_audio.py →Train an audio svm classifier with the outcomes of captions' sentiment analysis of training dataset. Split the audio in the related segments and create folders with positive, negative and neutral segments. At the path "*/Audio_Functions*" there are 2 python scripts that implement this 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.

There is also implemented 1 more python script: FtrainTest.py

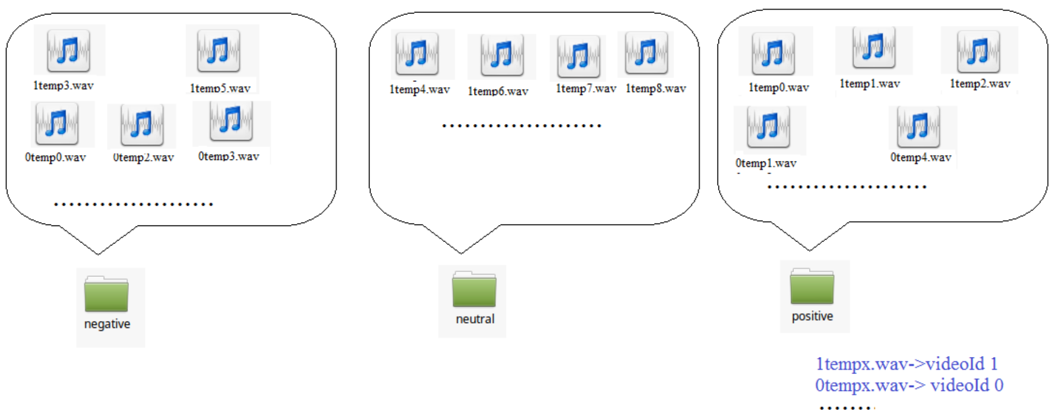
- FtrainTest.py→ use functions from from pyAudioAnalysis [1]. Using the Video IDs as explained above we select the videos for train and test. When the videos are selected we call dirsWavFeatureExtraction of pyaudio analysis to do audio sentiment classification.Features set are normalized to 0-mean and 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:



5.1. 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.

5.2. 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).

5.3. SVM classifier

On *FtrainTest.py* script the audio classifier is implemented. SVM (Support Vector Machine) classifier [7] used as audio classifier, another classifier can be used also. *svm_train_evaluate* function 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 metrics for each class(plotly and plotly_classification_results functions have been used). The model will be saved as pickle under name svm3classes file for future train or test purposes.

5.4. Model performance

5.4.1. F1 score

We decided to use F1 score as performance measure of the model. [8] 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.

5.5. Evaluate the ground-truth (sentiment analysis) of train dataset

To evaluate the ground truth of training data, we listened 100 segments from the train dataset, from all sentiments to confirm its polarity. With that way, we would like to know the performance of the sentiment classifiers we used. More specifically, we created a csv file at which we declared the polarity of segments based on our opinion. 3 different polarities declared and at the final state the majority sentiment to used as ground truth. This file is located at the path:

"ground_truth/ground_truth_sentimentAnalysis_traindataset.csv"

Based in these results we made the confusion matrix [9] for the predicted sentiments and we also calculated performance metrics.

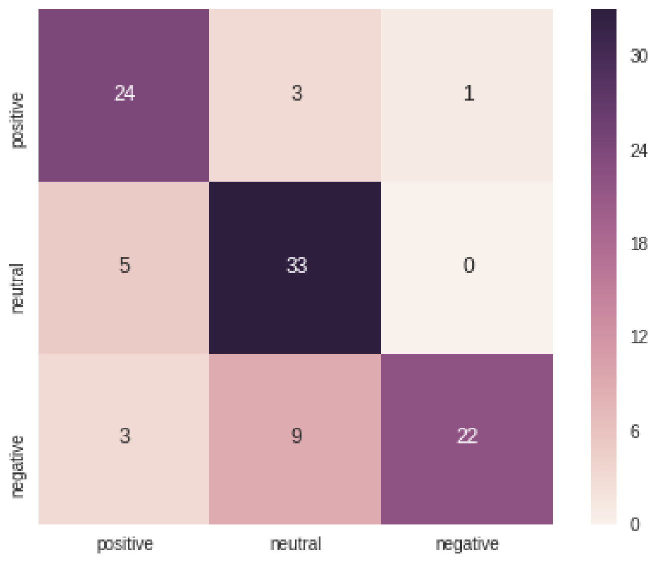


Figure 1: Confusion matrix for sentiment classifiers on train data

On x-axis are presented the sentiment analysis form the sentiment classifiers and on y-axis the results which are declared by us, listening the segments. Because we had 3 different annotation on each segment (3 different people listened them) we hold the majority sentiment. As it is shown from the heatmap, neutral sentiments are well predicted. Very few of them are false predicted as positive. The next well classified sentiment is the positive followed by negative sentiments. At both cases the misclassified segments have been classified on neutral class.

Recall calculates how many of the Actual Positives the model capture as Positive (True Positive). Recall shall be the model metric for someone to select the best model when there is a high cost associated with False Negative.

Precision is a good measure to determine how precise/accurate the model is out of those predicted positive, how many of them are actual positive. F1 Score is needed when you want to seek a balance between Precision and Recall.

Trained model has very high precision, close to 1 on negative sentiments, means that the segments which have been predicted as negative are indeed negative. F1 measure is identical for all sentiments close to 80%, meaning that the model is accurate about 80% on predicting sentiments from text. The average F1 score of the model is:

```
F1_score 0.7890368491509899
Precision: 0.8132850241545894
Recall: 0.7908742444346161
```

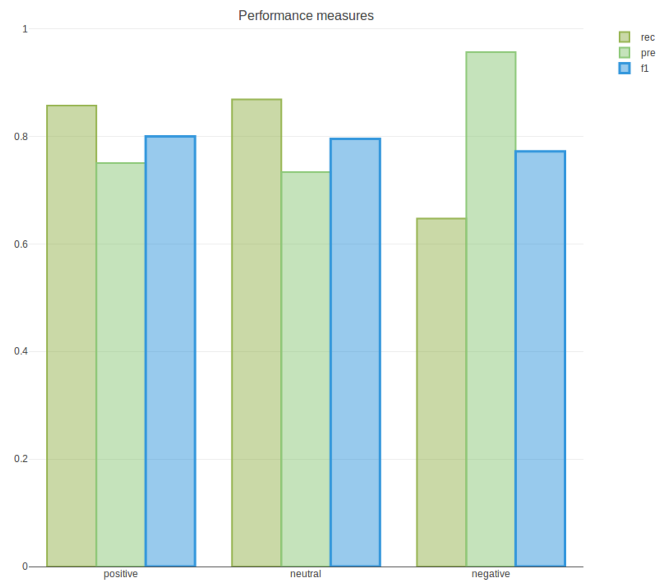


Figure 2: Performance measures or o sentiment classifiers on train data

5.6. Confusion matrix and F1 score for train dataset

The results from confusion matrix, F1 score, precision and recall for the train set are displayed above. As ground truth the results from sentiment classifiers have been used. Basically, we have used heatmap for to visualize confusion matrix results.

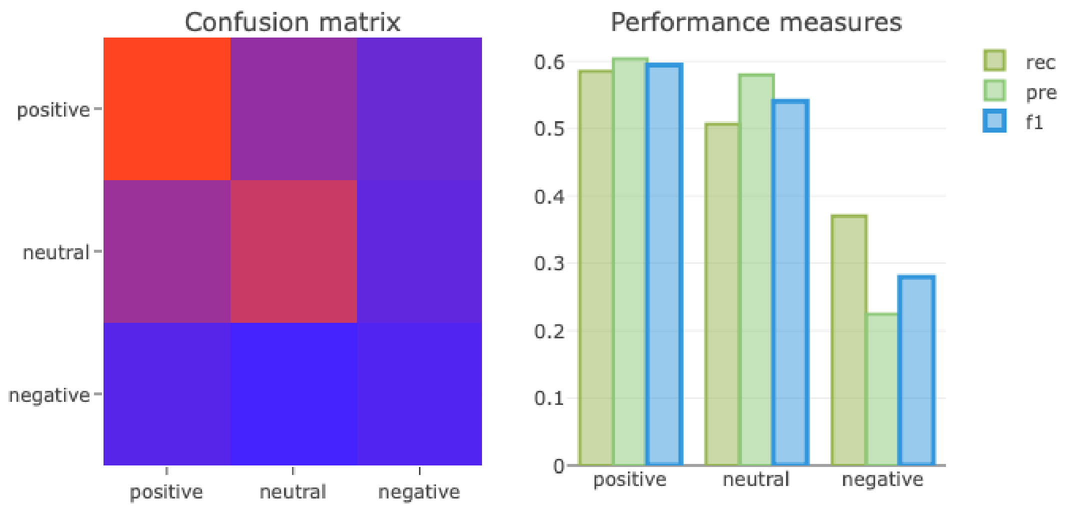


Figure 3: Confusion matrix and performance measures for training data

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

On the x-axis are presented the predictions and on y-axis the results from sentiment analysis. The values of the diagonal elements represent the degree of correctly predicted classes. The confusion is expressed by the false classified off diagonal elements, since they are mistakenly confused with another class. The classifier does good on positive results– red color-, although there are some positive that have been classified as neutral – close to red- . For results that should be class neutral, most of them are classified well -close to red- , but there are also some false predicted as positive. For the negative segments the classifier predicts bad results, the diagonal is blue (close to zero). It should be noticed that the classifier doesn't preform good results for negative sentiments segments.

Trained model has bigger precision, recall and F1 score for positive class, followed by neutral and negative. About 60% of the positive are predicted well, 50% of neutral and 30% of negative.

5.7. Ground-truth of test dataset

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 all segments from the test dataset, from all sentiments in order to confirm its polarity. More specifically, we created 3 folders: positive, negative and neutral and we put inside the several segments after listening them and evaluate their polarity. Finally, these folders used to calculate the performance of the model and the confusion matrix. These results are located on a dropbox folder. [3]

5.8. Evaluate the sentiment analysis on test dataset

To evaluate the ground truth of sentiment analysis on testing data, we listened 100 segments from the test dataset, the same procedure as described in Sec 5.5. The file with the results located at the path:

“ground_truth/ground_truth_sentimentAnalysis_testdataset.csv”

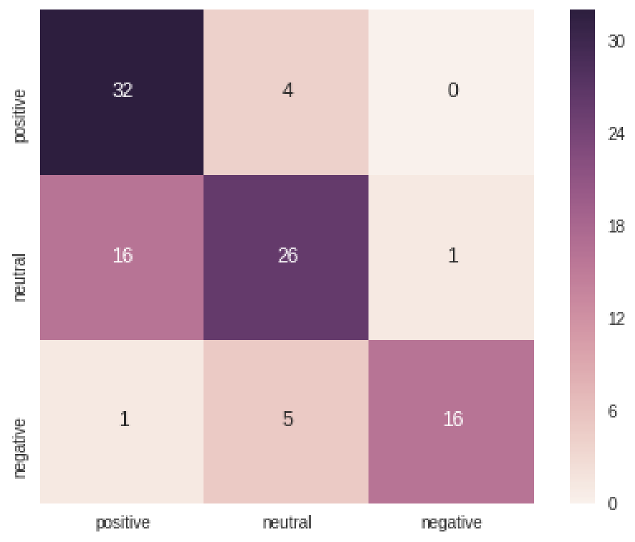


Figure 4: Confusion matrix for sentiment classifiers on test data

As it is shown from the heatmap, positive sentiments are well predicted. Very few of them are false predicted as neutral. The next well classified sentiment is the negative followed by neutral sentiments. At neutral case the misclassified segments have been classified on positive class, whereas the misclassified segments of negative have been classified as neutral.

More specifically, negative sentiments are predicted well 80%, positive 75% and neutral 65%. It is easily noticed that many segments which characterized form us as neutral sentiments, classifier classified them as positive. Precision of negative and positive segments is close to 1 means that the segments which have been classified as negative or positive indeed express this sentiment.

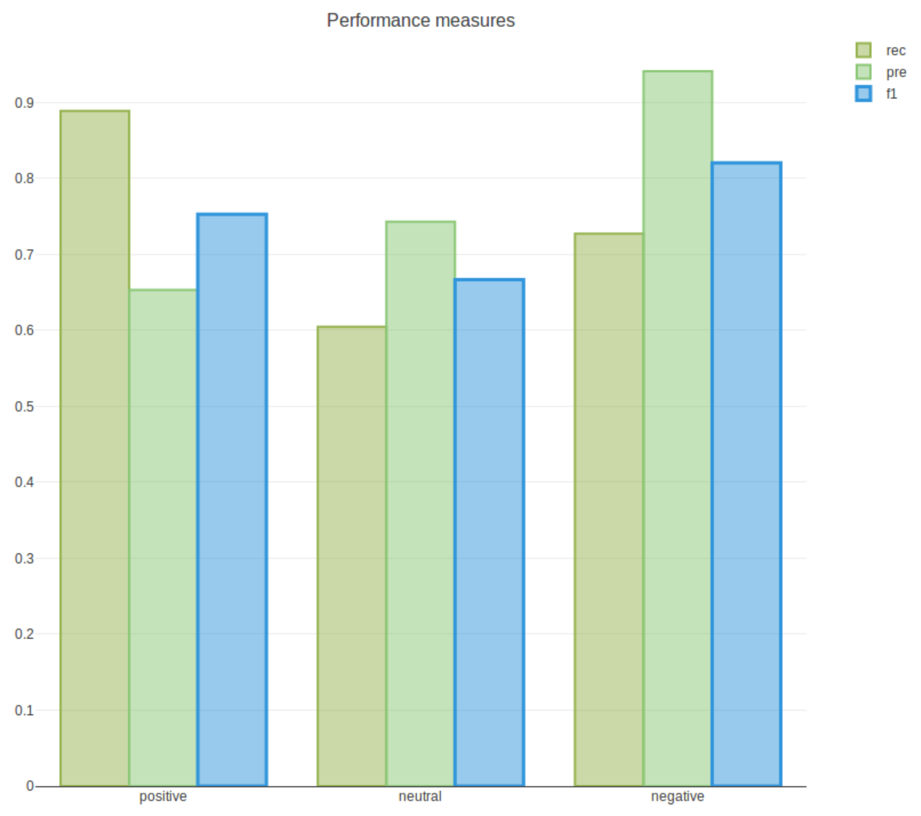


Figure 5:Performance measures or o sentiment classifiers on test data

The average F1 score of the model is:

```
F1_score 0.7467068878833585
Precision: 0.779031612645058
Recall: 0.740270926317438
```

Comparing with the performance metrics of train dataset, results from test dataset are identical as we expected.

Train: `F1_score 0.7890368491509899`
Test: `F1_score 0.7467068878833585`

5.9. Evaluate the performance of the audio-based emotion recognizer

Captions and audio files for test videos have been downloaded. The captions are being preprocessed using word_tokenize. We keep the segments with more than 4 words. This approach helped us in the selection of the sentiment segment. Videos of small duration or videos which containing music, laughing, clapping etc are not useful for our task and have been excluded. In order to test our trained audio classifier, we use *final* function from *FtrainTest.py*. Finally, we load the pretrained model svm3classes *load_model* function from *FtrainTest.py*. We normalize the features and split them in x_test, y_test and predict the values with the svm3classes classifier.

The performance of the model for the testing dataset is evaluated using the ground truth that we have already created, see Sec 5.7.

The results displayed below.

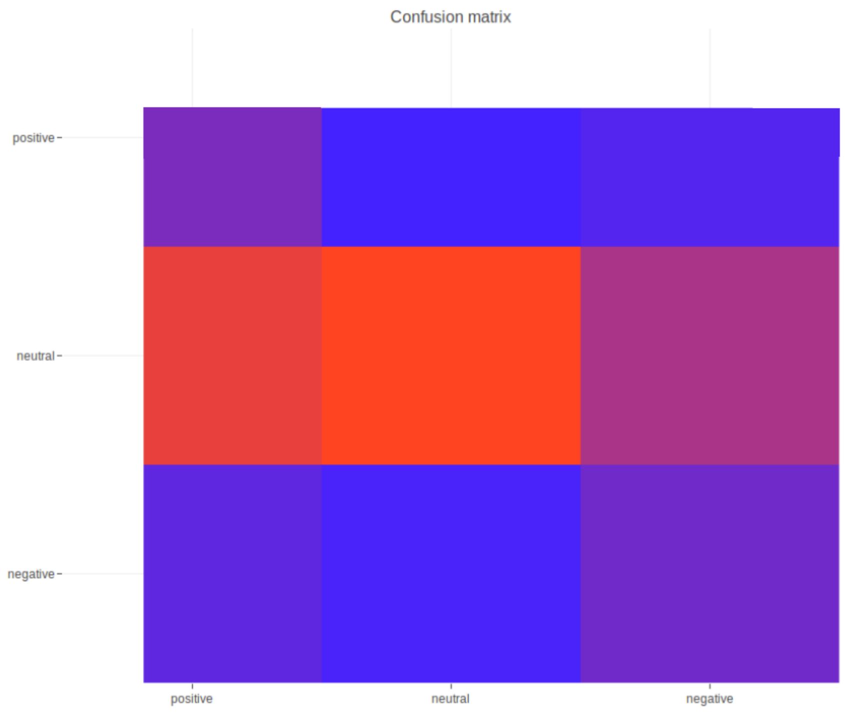


Figure 6: Confusion matrix for testing data

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

The classifier does good on neutral results– red color- . For results that should be class positive, most of them are classified well. For the negative segments the classifier predicts not so good results (purple color) as it classifies many of them as neutral. Generally, classifier doesn’t preform reliable results for negative segments, also on train dataset.

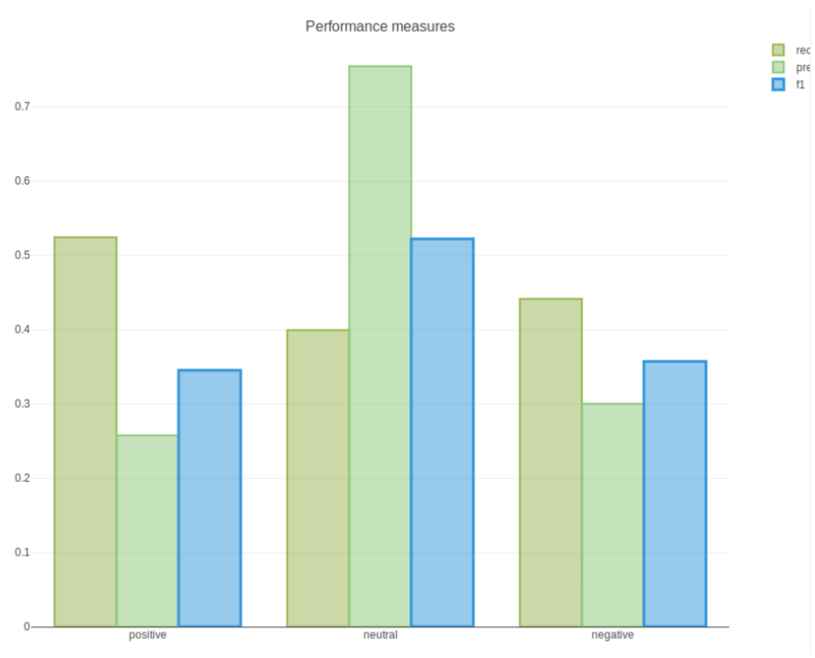
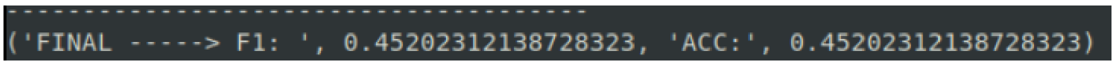


Figure 7: Performance measures for testing data

Trained model has 50% recall for positive segments whereas has low precision. That means that model predicts 50% of the actual positives whereas predicts many other positives which are not actual positive.

For the neutral prediction model has high precision. About 80% of the neutral predicted segments are indeed neutral. Negative has similar results for precision and recall metrics, about 35%- 40%. Negative results are similar with the training dataset. In general model doesn't perform good results for negative predictions, whereas seems to have similar results for neutral segments. For the positive segments, less positive segments are predicted on testing dataset in comparison with the training dataset. True positive segments are classifier with probability about 35%, neutral 50% and negative 35%. There is a small deviation in percentages.

The f1 score of the model is 45%.



```
---------------------------------------
('FINAL -----> F1: ', 0.45202312138728323, 'ACC:', 0.45202312138728323)
```

Not good but is was expected as we don't have a formal ground truth and we based only on our notion to evaluate the ground truth.

5.10. Final results

At point we can list the final results of Large – scale cross – modal speech emotion recognition model we described above.

	Train dataset	Test dataset
automatic text sentiment	About 80/% of segments are well classified with sentiment analysis	About 75% positive, 65% neutral and 80% of segments are well classified with sentiment analysis.
Audio-based classifier	About 60% of the positive, 50% of neutral and 30% of negative segments are predicted well.	About 35% of the positives, 50% of the neutral and 35% of negative segments are predicted well.

Note: There is a small deviation in percentages.

6. Source code

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

```
multimodal_audio
|-- Audio_Functions
|    |-- File_Functions.py
|    |-- Parse_Functions.py
|
|-- Caption_Functions
|    |-- File_Functions.py
|    |-- Parse_Functions.py
|    |-- Sentiment.py
|
|-- Download_Functions
|    |-- Download_Audio.py
|    |-- Download_Subtitles.py
|    |-- Files_And_Dirs.py
|
|-- test
|    |-- audio
|        |-- positive
|        |-- negative
|        |-- neutral
|        |--0.wav
|        |-- ...
|    |-- pickle_lists
|        |--polarity_0.p
|        |-- ...
|    |-- polarity_csv
|        |--polarity_0.csv
|        |-- ...
|    |-- subtitles
|        |--0.srt
|        |-- ...
|    |-- dataset_test.p
|
|-- train
|    |-- audio
|        |-- positive
|        |-- negative
|        |-- neutral
|        |--0.wav
|        |-- ...
|    |-- pickle_lists
|        |--polarity_0.p
|        |-- ...
|    |-- polarity_csv
|        |--polarity_0.csv
|        |-- ...
|    |-- subtitles
|        |--0.srt
|        |-- ...
|    |-- dataset_test.p
```

6.1. Running commands

```
Python Downloader.py
python Parser_caption.py
python Parser_audio.py <pyaudioanalysis_path>
```

7. Repository of the project and management tool

7.1. Repository

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

7.2. 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

[3] https://www.dropbox.com/sh/6zivh9r5rnwp1di/AABXw1_gRSTXRFBkpnFfxvzna?dl=0

[4] https://nlpforhackers.io/sentiment-analysis-intro/

[5] https://textblob.readthedocs.io/en/dev/

[6] https://www.clips.uantwerpen.be/pages/pattern-en

[7] https://scikit-learn.org/stable/modules/svm.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html