COMPARATIVE CASE STUDY BETWEEN AUTOMATIC SPEECH RECOGNITION VERSUS MULTIMODAL AUTOMATIC EMOTION RECOGNITION SYSTEM USING VOICE ANALYTICS, FACIAL PATTERN RECOGNITION, NLP, PYTHON, AND DEEP LEARNING.

By

VENMATHI SHANMUGAM

A Capstone Project Paper Submitted in Partial Fulfillment of the

Requirement for the Degree of

Master of Science

In

Data Science

University of Wisconsin – Eau Claire

Eau Claire, Wisconsin

December 2018

ABSTRACT

ASR VS MULTIMODAL SPEECH EMOTION RECOGNITION SYSTEM

VENMATHI SHANMUGAM

Humans have always been fascinated by how computers can be trained to understand them. We are in an AI-driven world now where products like Amazon Echo, Alexa, Siri etc., can listen to us speak, and can answer our questions quickly and knowledgably. But how accurate could those answers be without recognizing the emotional state of the speaker?

This paper dives into a comparative case study between such Automatic speech recognition (ASR) algorithms utilizing just spoken text as input, versus a complete Multimodal Automatic Emotion detection system including multiple inputs like audio and video.

Using a dataset with audio and video clippings of 24 different actors, the strengths and limitations of using non-verbal cues in emotion detection have also been illustrated. Emotions were group into 7 major categories- neutral, happy, sad, surprise, fear, angry, disgust.

Speech text interpretation and sentiment analysis was performed using NLP and VADER. Audio emotion detection accuracy is determined using 4 different classifiers- Decision Tree, LSTM, MLP and CNN. CNN shows the best accuracy rate of 71.28%. Facial emotion recognition for the same dataset is demonstrated 2 ways-using Fischer face classifiers and Linear SVC classifiers using facial landmarks. Linear SVC gave a remarkable accuracy of 87.88%. A Beam Search fusion methodology then applied to the best models, gave 60.34% accurate predictions with the multimodal inputs of emotions derived from speech text, audio and facial features.

The paper ends with an interesting live demo in conclusion that proves the case in topic and discusses additional enhancements that would make the current system more robust.

Keywords : Multimodal Emotion recognition system, Deep learning, CNN, LSTM, MLP, SVM, OpenCV, landmark facial detection, voice analytics, Tensorflow, Keras, NLP.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Project Background

Automatic Speech Recognition (ASR) and Automatic Emotion classification system from human speech have been both a relatively new field. Most research and work done in the AI field focusses on the former algorithm, extracting meaning from "what" was said by the speaker- the semantic analysis which involves voice-to-speech and speech-to-text conversion followed by process of utilizing NLP on text to predict emotions/sentiments of the speech content.

Though there are many products out in the market trying to solve this challenge, they do not seem very effective/ meet fully the growing awareness and need in the booming voice analytic industry today. This brings in the necessity for a comprehensive system which can incorporate ASR and include emotion recognition factor into it to make the system more robust.

One potential cause for ASR fail could be that, with speech analysis systems, primary issue is that, though they can be quick to produce a report, human analysis would still be needed to analyze the results and create meaningful insights from the report. This is because human speech not only contains the linguistic content, but also the emotion of the speaker, in that the same word/phrase can change in meaning completely depending on the speaker's behavior and emotional state.

In this paper, I have taken to using this downfall of speech text emotion detection algorithm to prove how accurate emotion detection from non-verbal cues could be when the speaker(s) say the same text with different emotions. The non-verbal cues used here are – the audio signal

features and the facial features of 24 different actors uttering the same text with different emotions. Their emotions are categorized under 7 buckets- neutral, sad, happy, angry, fear, surprise and disgust.

Despite many recent researches coming out with approaches to recognize facial expressions or audio signals or speech, there has been relatively very few proposals illustrating the advantages of combining other modalities to improve accuracy and robustness of the current emotion detection system.

In order to make the human and digital interaction more natural and to allow the computer to recognize the emotional states the same way as humans do, this new path of research could be employed in the future in addition to the current speech text recognition system.

In other words, what we aim here, in this paper, is to not just concentrate on "what was said" by the person, but also on "how" it was said, thereby more accurately detecting the person's emotional state via a fusion of audio and video signals. We have also proven in this paper that this Bimodal emotion recognition system adds measurable improvement to the current system.

Objectives

The primary objective of this project is to extract meaningful features from audio and video clips of the same person uttering the same sentence- for deeper sentiment analytics and to get true emotional insights about "how" the content of the speech was voiced.

The other main objective is to build and analyze the efficiency of classifier models that have been historically researched about in the field of emotion recognition through spoken audio signals and facial features, to choose the best possible model with the greatest accuracy and fuse them based on their classification accuracy. This would prove that it is actually plausible to detect emotions from audio/video files and address the issues from speech-emotion-detection, with a more than acceptable performance rate.

The objectives/ goals of this project, in short, could be summarized as finding answers to the below listed 5 questions, with a demonstrated case study.

1. What are the primary shortcomings of the traditional ASR algorithm?
2. How can an Emotion recognition system help overcome those shortcomings?
3. How can emotions be identified from non-verbal cues like voice audio features and facial patterns/ features?
4. Which would be the best algorithm with the best possible accuracy that can classify emotions from voice and emotions from face?
5. How would a multimodal fusion method make the current ASR algorithm more robust?

Rationale for undertaking this project:

Emotion recognition from speech text has been a relatively recent topic of advancement in the Human Computer Interaction space. Despite many researches working in this area, there still seems to be a lack of a complete system which can recognize the ‘real’ emotion from speech texts because of various reasons/ limitations of the current system including-

1. Inability to differentiate between the tone in which the speech is delivered.

2. Different accents used in dialect.

2. Lack of ability to read sarcasm.

3. Same text said in different tones could take a totally different meaning.

This project when implemented, adds in a fusion of non-verbal cues and improves accuracy rate of the current speech detection system measurably. In addition to addressing the aforesaid limitations, it also provides added value in the following fields-

1. It can help detect customer's emotion and can be of great potential benefit in customer service centers to decrease customer churn.
2. Can also be of value in other fields like medical (to detect pain levels from patient voice), entertainment, crime detection, robotics (to develop companion robots), personalized AI products, law enforcement services, politics and marketing research.

Project Inspiration:

I have worked on sentiment analysis from real-time tweets, network analysis, text mining techniques and built emotion classification systems using Natural Language Processing earlier, at the end of my previous Data Mining and Visualization and Network analysis classes. But I always had to explain the numerous limitations. Personally, I felt I had to expand my horizon further to find a solution to this persistent issue. Coincidentally, I stumbled upon one too many data science jobs that exclusively asked for experience in voice analytics (companies like DELL, Tether etc) or facial pattern recognition methodologies (numerous start-ups).

As a machine learning fanatic, I got deeply intrigued on this topic and with the hopes of enhancing knowledge gained through my Masters program, started reading and researching more on the topic and its potential real-world purpose.

I also happen to personally know the CTO of SmartBeings Inc.,- A Silicon valley AI driven IoT start-up, which specializes in delivering exceptional personalized AI products. This upcoming successful smart-home and assisted living product venture, has a unique approach to solve real-world problems in a very interesting way utilizing AI, deep learning, voice-based technologies (including NLP & acoustics), facial recognition, security and data encryption and cloud connectivity.

Coincidentally, the CTO shared that one of the major issues pertaining to semantic analysis that his team of data engineers have been working to resolve in the recent days, was the one that would address the accuracy issue in emotion/mood prediction from human speech, caused due to different accents used in dialect.

This got me to start this project on trying to use a multimodal approach to emotion recognition. But by not just knowing "what did the person say" (words in the speech), knowing "how did the person say it" (with the help of voice features, facial landmarks other extracted audio/video features), we have an added advantage of predicting the emotion of the end user more accurately and taking our current speech recognition system to the next level.

This idea has been the inspiration for my project.

CHAPTER 2

LITERATURE REVIEW

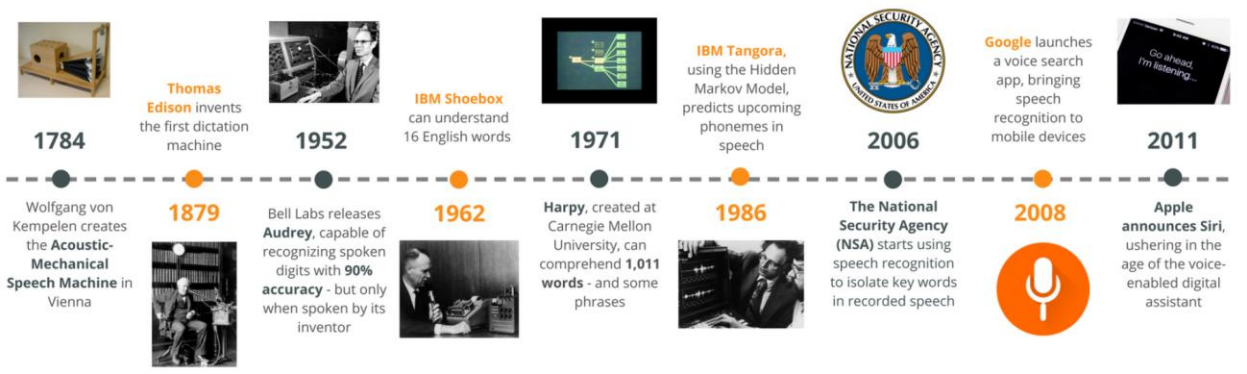
History:

Boyd,C. (2018) [13] states that –

"The history of the technology reveals that speech recognition is far from a new preoccupation, even if the pace of development has not always matched the level of interest in the topic."

The below figure dates back in history to illustrate the development we had over time in Speech Recognition system.

Fig 1: History of Speech Recognition system



This goes to show that speech recognition and emotion sensing has always been an important part of research for scientists who were looking for a machine's intelligence to match data-driven rationality.

Previous Work:

I have scoured through a lot of past work and selected very few of many related literatures that were interesting and inspirational. Below is the list with brief descriptions of work done-

- Cao et al. [8] proposed a ranking SVM method for synthesizing information about emotion recognition to solve the problem of binary classification. This ranking method, instructs SVM algorithms for particular emotions, treating data from every utterer as a distinct query then mixed all predictions from rankers to apply multi-class prediction.
- Chen et al. [7] aimed to improve speech emotion recognition in speaker-independent with a three level speech emotion recognition method. Four comparative experiments conducted include Fisher + SVM, PCA + SVM, Fisher + ANN and PCA + ANN.
- Nwe et al. [9] proposed a new system for emotion classification of utterance signals. The system employed a short time log frequency power coefficients (LFPC) and discrete HMM to characterize the speech signals and classifier respectively.
- Wu et al. [10] proposed a new modulation spectral features (MSFs) human speech emotion recognition. Appropriate feature extracted from an auditory-inspired long-term spectro-temporal by utilizing a modulation filterbank and an auditory filterbank for speech decomposition. This method obtained acoustic frequency and temporal modulation frequency components for convey important data which is missing from traditional short-term spectral features.

- Grimm et al. [11] proposed a multi-dimensional model by utilizing emotion primitives for speech emotion recognition. Three dimensions were made by composing of three different value of emotion primitives, which is called valence, activation, and dominance. The value of these factors assumed to be in the range of [-1, +1]. A text-free, image-based method was introduced to assess the emotion primitives and achieves best inter-evaluator agreement. For extracting acoustic feature such as energy, pith and spectral specifications, both fuzzy logic and rule-based estimator are employed. The approached are validate by testing two EMA and VAM datasets, which are acted emotion and spontaneous speech emotion.

A few other interesting internet resources:

- Behind the Mic: The Science of Talking with Computers. A short film about speech processing by Google. (youtube, 2014)
- A Historical Perspective of Speech Recognition by Huang, Baker and Reddy. Communications of the ACM (2014) [12]. This article provides an in-depth and scholarly look at the evolution of speech recognition technology.

Some good books about speech recognition:

- The Voice in the Machine: Building Computers That Understand Speech, Pieraccini, MIT Press (2012). An accessible general-audience book covering the history of, as well as modern advances in, speech processing.

- Fundamentals of Speech Recognition, Rabiner and Juang, Prentice Hall (1993). Rabiner, a researcher at Bell Labs, was instrumental in designing some of the first commercially viable speech recognizers. This book is now over 20 years old, but a lot of the fundamentals remain the same.
- Automatic Speech Recognition: A Deep Learning Approach, Yu and Deng, Springer (2014). Yu and Deng are researchers at Microsoft and both very active in the field of speech processing. This book covers a lot of modern approaches and cutting-edge research but is not for the mathematically faint-of-heart.

CHAPTER 3

PROJECT OVERVIEW

General Approach/Layout:

The basic steps undertaken to approach this project could be summarized as follows-

1. Define the problem.
2. Select a Data Source that would be appropriate to demonstrate our comparative case study between Automatic Speech Recognition and Emotion recognition.
3. Study the data source in depth.
4. Reverse engineer Automatic Speech Recognition algorithm and apply to dataset, thereby proving the shortcomings.
5. Illustrate emotion recognition accuracy using audio features extracted from .wav files extracted from .mp4 files of selected database.
6. Train multiple classifier models with extracted audio features and find the one that predicts the emotions in the database with the greatest accuracy.
7. Illustrate emotion recognition accuracy using facial patterns and features extracted from video mp4 files of selected database.
8. Train multiple classifier models with extracted facial patterns/ features and find the one that predicts the emotions in the database with the greatest accuracy.
9. Apply a fusion methodology and determine accuracy.

10. Discuss how Multimodal Emotion Recognition could be more robust than traditional Speech Recognition Algorithm.
11. Conclude with some proposed enhancements.

We will look into this approach in detail with process methodologies involved in each step in the forthcoming chapters.

CHAPTER 4

PRELIMINARY STEPS IN CASE STUDY

This comparative case study could be broken down based on the project approach steps briefed in the previous chapter. The end-to-end workflow steps carried out and thought process involved is explained in detail in the following subsections and chapters.

Problem Definition

For any given project, it is very important to define the problem statement clearly before we proceed to start the actual work. In our case, the problem statement, as discussed extensively prior, is to identify and demonstrate, with data, the shortcomings of the current Automatic Speech Recognition system and illustrate, with data again, how this could be overcome with building a comprehensive Multimodal Emotion Recognition system.

As a subset of this comparative case study illustration, we also include a comparative analysis on Emotion Prediction accuracies between various classifier models, both when trained on audio features as well as facial features from videos.

Data Source Selection:

Three categories of datasets are usually used in analyzing emotions: acted emotions, natural spontaneous emotions, and elicited emotions [14]. Although different actors may understand and

interpret instructions differently and may experience the emotions to different degrees, data obtained from acted emotions are less ambiguous because actors express the exact emotions they were instructed to act. In contrast, spontaneous speech and emotions can, for example, be collected from call center data [15] or through human-computer interaction [16]. These emotions are more diversified and are often difficult to classify given that the data must be mapped onto a limited number of classes. Even if it is evident that emotion research should ideally target natural databases, acted databases are more systematically controlled and useful, especially neural activity will be measured.

Furthermore, there is a direct correspondence between the collected data and their labels, generally resulting in higher accuracy in emotion recognition [17,18].

We therefore decided to use acted emotions for data collection in this work.

The other necessity for this particular case study was that we should be able to demonstrate the shortcomings of Automatic Speech Recognition(ASR) using the data to start with. As previously discussed, the major shortcoming of ASR algorithm is its inability to read into deeper emotional content into the speech, especially sarcasm. This is because human speech not only contains the linguistic content, but also the emotion of the speaker, in that the same word/phrase can change in meaning completely depending on the speaker's behavior and emotional state.

So, the dataset selected needed to have multiple actors speaking the same sentence over and over, each time with a different emotion.

The emotions had to be labelled and we needed to have enough data to train and test all our models and complete the entire case study process using the same dataset.

Based on all these criteria, the RAVDESS dataset was selected for our case study.

CHAPTER 5

SOURCE DATABASE EXPLAINED

Data Source Validation:

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) is an open-source dynamic, multimodal set of facial and vocal expressions in North American English [19]. The RAVDESS contains 7356 files. Each file was rated 10 times on emotional validity, intensity, and genuineness. Ratings were provided by 247 individuals who were characteristic of untrained adult research participants from North America. A further set of 72 participants provided test-retest data. High levels of emotional validity, interrater reliability, and test-retest intrarater reliability were reported.

Data Source Description:

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains 7356 files (total size: 24.8 GB). The database contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression. All conditions are available in three modality formats: Audio-only (16bit, 48kHz .wav), Audio-Video (720p H.264, AAC 48kHz, .mp4), and Video-only (no sound).

File naming convention:

Each of the 7356 RAVDESS files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 02-01-06-01-02-01-12.mp4). These identifiers define the stimulus characteristics:

Filename identifiers:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Filename example: 02-01-06-01-02-01-12.mp4

1. Video-only (02)
2. Speech (01)
3. Fearful (06)
4. Normal intensity (01)

5. Statement "dogs" (02)
6. 1st Repetition (01)
7. 12th Actor (12)
8. Female, as the actor ID number is even.

CHAPTER 6

METHODOLOGY AND OUTCOMES

The methodology comprises of 3 broad steps-

- Applying Emotion Classification algorithms on spoken text data extracted from the data source. This simulates ASR methodology and helps demonstrate the shortcomings discussed earlier.
- Building multiple Classifier models to predict emotions from audio and video features extracted from the same database and determining the best model with the greatest accuracy from each methodology- audio and video- individually.
- Applying Beam Search fusion on a data subset based on the best of classification accuracies obtained and get accurate predictions with the multimodal approach, involving the fusion of text, audio and video features.

Now, let us discuss each of these steps in depth to understand and compare outcomes.

Emotion Classification from Verbal Speech Text:

Based on our data source understanding, we already know that the 24 actors have been speaking only 2 sentences in 8 different emotions- happy, sad, neutral, anger, fearful, calm, disgust, surprised. The text from 2 sentences have also been given in the data source explanation, which was very convenient for a quick run of the simulated speech emotion recognition algorithm. Per the data source [19], the 2 sentences were-

“Kids are talking by the door” and “Dogs are sitting by the door”.

Here we adopt 3 different powerful speech emotion extraction techniques to compare and confirm our assumptions about the verbal text.

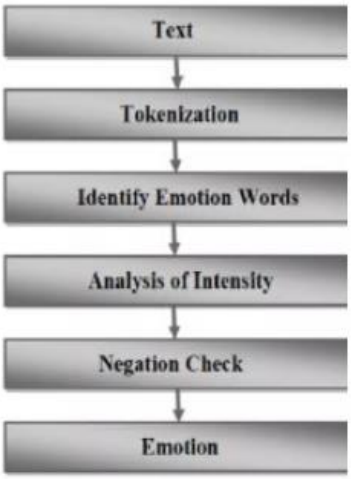
Using NLTK - VADER:

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.(Wikipedia).

VADER or Valence Aware Dictionary and Sentiment Reasoner is a powerful but easy to use python package used commonly for sentiment analysis. VADER belongs to a type of sentiment analysis that is based on lexicons of sentiment-related words. In this approach, each of the words in the lexicon is rated as to whether it is positive or negative, and in many cases, how positive or negative.[21]

In this approach, we are classifying the input text into different emotions by finding the emotional content from the given English text. The emotional contents are verbs, adverbs, adjectives, phrases or combination of these keywords. For example, “We are going on a vacation. I’m very excited”. The keyword “excited” represents “happiness” or “joy”, using such keywords emotions can be classified.[20].

Fig 2: Key word Spotting Technique

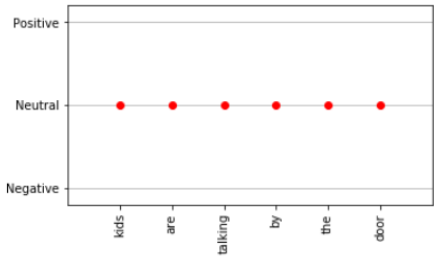


The quick and dirty code output looked like this-

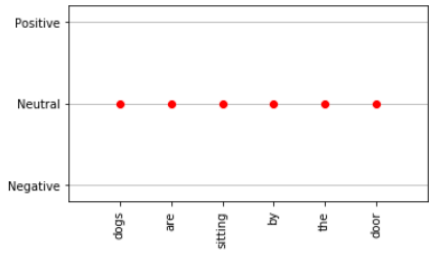
In [7]:
```
text11='Kids are talking by the door'
text21='Dogs are sitting by the door'

nltk.sentiment.util.demo_vader_instance(text11)
nltk.sentiment.util.demo_liu_hu_lexicon(text11, plot=True)
```

```
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Neutral
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently reuses the earlier instance.  In a future version, a new instance will always be created and returned.  Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



```
In [6]: nltk.sentiment.util.demo_vader_instance(text21)
        nltk.sentiment.util.demo_liu_hu_lexicon(text21, plot=True)

        {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
        Neutral

        C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107: MatplotlibDeprecationWarning: Adding an axes
        using the same arguments as a previous axes currently reuses the earlier instance.  In a future version, a new instance will
        always be created and returned.  Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a un
        ique label to each axes instance.
          warnings.warn(message, mplDeprecation, stacklevel=1)
```

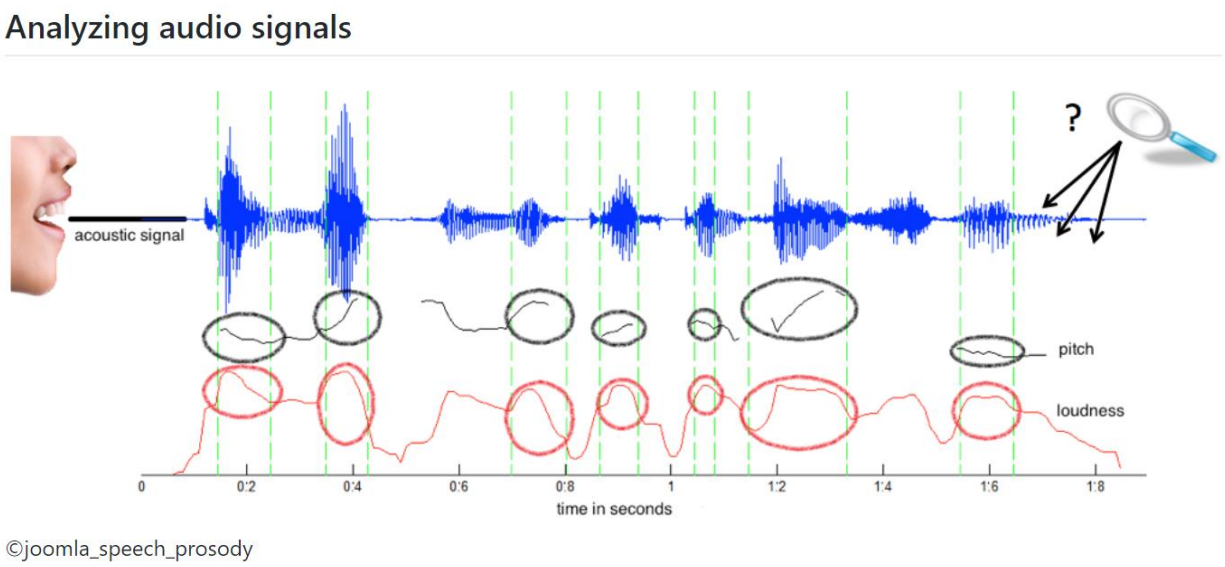


- The Positive, Negative and Neutral scores represent the proportion of text that falls in these categories. These should add up to 1. The above output means that our sentence was rated as 100% Neutral.

- The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive).

Since we know that all the video files in the database have only one of the 2 sentences, we can safely assume that when we apply Automatic Speech Recognition algorithm to this dataset, we are going to get “neutral” as our only emotion.

Emotion Classification from Speech signals (Audio):



The methodology adapted here could be broken down into the following steps-

Audio Pre-Processing:

Typically, the first step here would have been audio file extraction from the .mp4 files in the data source. But conveniently enough, I did find that the same database source that had the video files of the 24 actors also had .wav files of the corresponding speech audio files. I went with the Audio only zip file because which consisted of around 1500 audio files which were in wav format. I extracted these files from the source and started with the pre-processing step. The step as such was accomplished with a bunch of code establishing pipelines, installing packages and dependencies (codes attached in Appendix).

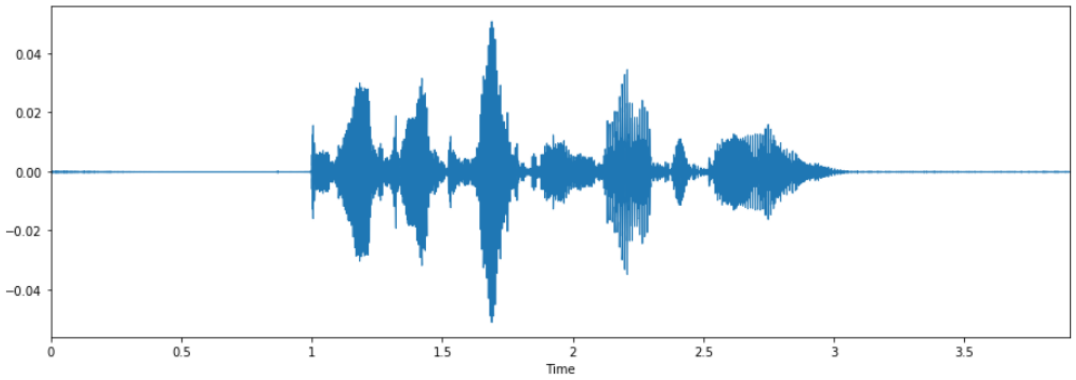
Data visualization:

I tested out the audio files by plotting out the waveform and a spectrogram to see the sample audio files.

An audio waveform is a time domain display, a display of amplitude vs time.

```
plt.figure(figsize=(15, 5))
librosa.display.waveplot(data, sr=sampling_rate)
```

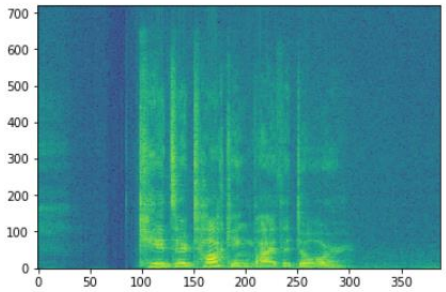
```
Out[7]: <matplotlib.collections.PolyCollection at 0x2dce9a19710>
```



An audio spectrum is a frequency domain display, a display of amplitude vs frequency. This means the audio signal consists of the following frequencies with corresponding magnitudes.

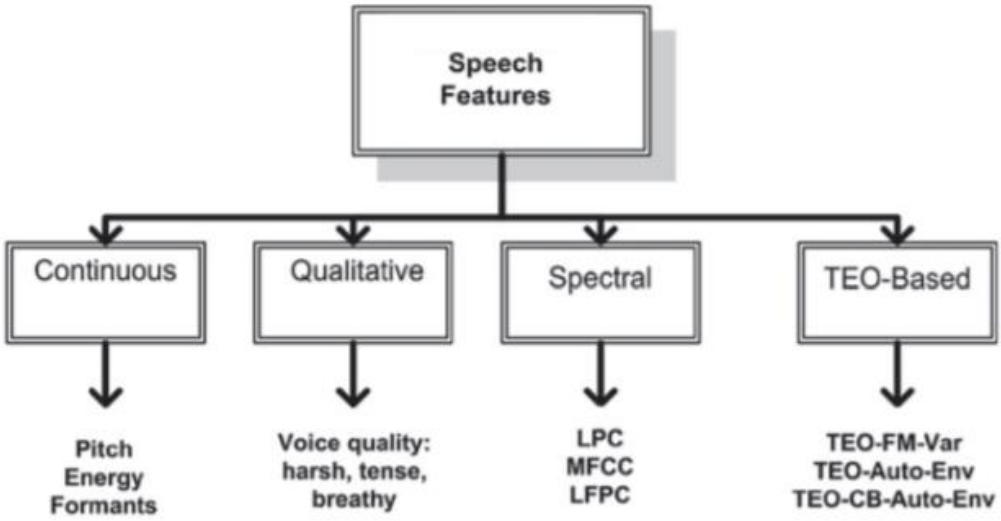
```
plt.imshow(X.T, interpolation='nearest',
    origin='lower',
    aspect='auto')

plt.show()
```

```
C:\Users\venil\AppData\Roaming\Python\Python36\site-packages\scipy\io\wavfile.py:273: WavFileWarning: Chunk (non-data) not u
nderstood, skipping it.
  WavFileWarning)
```



Feature Extraction:

Now the next step is to extract features from this audio representations, so that our algorithm can work on these features and perform the task it is designed for. Here's a visual representation of the categories of audio features that can be extracted.



I took MFCC feature for building my Decision Tree model.

Mel-Frequency Cepstrum Coefficients (MFCCs) are coefficients result from a transformation to a cepstrum space, in order to capture information of the time-varying spectral envelope. The cepstrum space is yet another spacial transformation that results from taking the inverse spectral transform of the logarithm of the spectrum. Its basic unit is the quefrency, and its main advantages are to be relatively robust space against noise that also empathizes changes or

periodicity in the spectrum. Specific features proven to be useful under practically any speech processing task can be obtained under this space: the so called MFCCs. [23].

For the other 3 deep learning models though, there are two stages in the audio feature extraction methodology:

- Short-term feature extraction: this is implemented in function stFeatureExtraction() of the audioFeatureExtraction.py file. It splits the input signal into short-term widnows (frames) and computes a number of features for each frame. This process leads to a sequence of short-term feature vectors for the whole signal.
- Mid-term feature extraction: In many cases, the signal is represented by statistics on the extracted short-term feature sequences described above. Towards this end, function mtFeatureExtraction() from the audioFeatureExtraction.py file extracts a number of statistcs (e.g. mean and standard deviation) over each short-term feature sequence.

The total number of short-term features implemented in pyAudioAnalysis is 34. In the following table the complete list of the implemented features is presented:

Index	Name	Description
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9–21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
22–33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

Complete list of implemented audio features. Each short-term window is represented by a feature vector of 34 features listed in the Table.

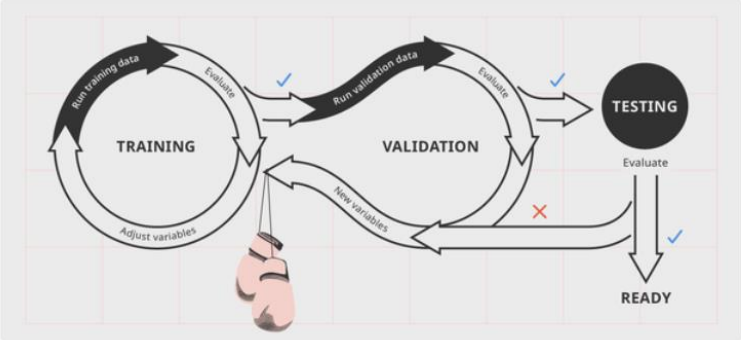
Data Splitting:

The most common problem when developing any machine learning model is overfitting. Splitting data helps overcome this issue. In our case, we split the data into training and test sets on a 80-20 criteria after a random shuffling is applied.

The training dataset and test dataset are used for different purposes.

The training dataset (also called training set, learning set, or AI training data) is the initial dataset used to train an algorithm to understand how to apply technologies such as neural networks, to learn and produce complex results. It includes both input data and the corresponding expected output. The purpose of the training dataset is to provide your algorithm with "ground truth" data.

The test dataset, however, is used to assess how well your algorithm was trained with the training dataset. You can't simply reuse the training dataset in the testing stage because the algorithm will already "know" the expected output, which defeats the purpose of testing the algorithm.



Decision tree Classifier:

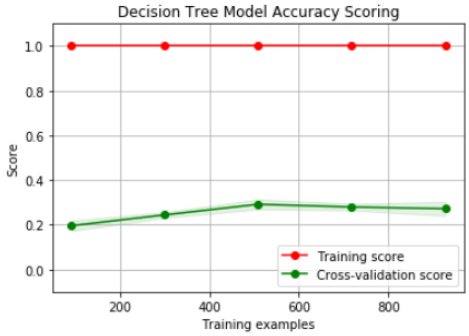
Decision trees are extremely intuitive ways to classify or label objects: you simply ask a series of questions designed to zero-in on the classification. A simple decision tree built on this data will iteratively split the data along one or the other axis according to some quantitative criterion, and at each level assign the label of the new region according to a majority vote of points within it. This process of fitting a decision tree to our data can be done in Scikit-Learn with the DecisionTreeClass estimator.

For our data, I was able to plot the Model accuracy scoring on our training and test data-

```
line_up = plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
line_down = plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")
plt.ylim(-.1, 1.1)
plt.legend(loc="lower right")
plt.show()
```

```
[learning_curve] Training set sizes: [ 92 301 509 718 927]
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    4.5s finished
No handles with labels found to put in legend.
```



Once the model was trained, I applied it on the test data to predict emotions and got a 33.45% accuracy for our Decision Tree model.

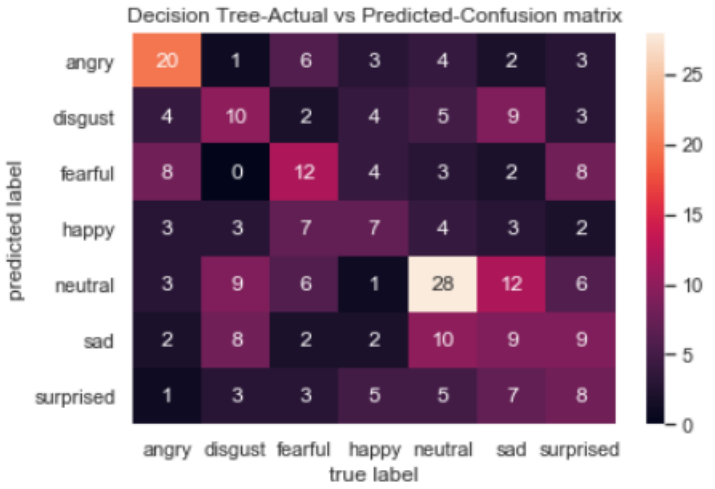
```
# True labels
test_true = ts_labels

# List to store prediction
test_predicted = []

# Iterate over model prediction and store it into list
for i, val in enumerate(prediction):
    test_predicted.append(val)


# Accuracy score of model
print('Accuracy Score of Decision Tree model is:', accuracy_score(test_true, test_predicted))

# Number of corrected prediction
print('Number of correct prediction by using Decision Tree is:', accuracy_score(test_true, test_predicted, normalize=False),
```

```
Accuracy Score of Decision Tree model is: 0.33451957295373663
Number of correct prediction by using Decision Tree is: 94 out of 281
```

Accuracy Score of Decision Tree model is: 33.45%

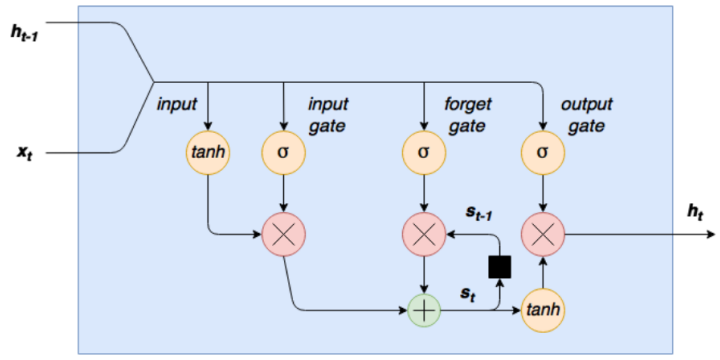
The confusion matrix between actual and predicted values looked like this-

```
df = pd.DataFrame(matrix, columns=classes, index=classes)
plt.figure()
#plt.title('Decision Tree-Actual vs Predicted-Confusion matrix')
sn.heatmap(df, annot=True)

#plt.show()

#sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.title('Decision Tree-Actual vs Predicted-Confusion matrix')
plt.show()
```



LSTM Model:

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. An LSTM network has LSTM cell blocks in place of our standard neural network layers. These cells have various components called the input gate, the forget gate and the output gate. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. [24].



LSTM cell diagram

With our data though, the LSTM model had the lowest training accuracy of 30.39% with 5 layers, tan h activation function, batch size of 32 and 50 epochs.

```
In [36]: modellstm.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 259)         310800
_________________________________________________________________
lstm_3 (LSTM)                (None, None, 400)         1056000
_________________________________________________________________
dense_3 (Dense)              (None, None, 256)         102656
_________________________________________________________________
lstm_4 (LSTM)                (None, 128)               197120
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 903
=================================================================
Total params: 1,667,479
Trainable params: 1,667,479
Non-trainable params: 0
_________________________________________________________________
```

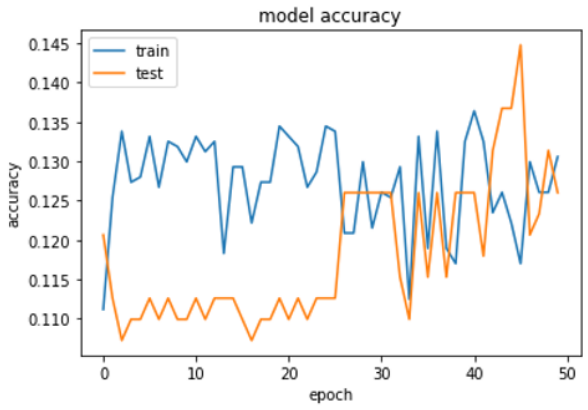
In [1]:
```
####Saving the Model
model_name = 'Model_LSTM.h5'
save_dir = os.path.join(os.getcwd(), 'saved_models')
## Save model and weights
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print("Loaded model from disk")
```

```
Loaded model from disk
```

In [2]:
```
# evaluate loaded model on test data
loaded_model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
score = loaded_model.evaluate(x_testlstm, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
acc: 30.39%
```

In [91]:
```
plt.plot(lstmhistory.history['acc'])
plt.plot(lstmhistory.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



MLP Model:

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.[25] Its multiple

layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.[26]

Multilayer perceptron is sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

In our data, MLP model had a validation accuracy of around 47.57% with 7 layers, softmax function at the output, batch size of 32 and 550 epochs.

```
In [21]: clf = MLPClassifier(activation='tanh', alpha=0.03, batch_size='auto', beta_1=0.9,
                beta_2=0.999, early_stopping=False, epsilon=1e-08,
                hidden_layer_sizes=(64, 64, 64), learning_rate='constant',
                learning_rate_init=0.001, max_iter=100000, momentum=0.9,
                nesterovs_momentum=True, power_t=0.5, random_state=48, shuffle=True,
                solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
                warm_start=False)


         clf.fit(X_train, y_train)

Out[21]: MLPClassifier(activation='tanh', alpha=0.03, batch_size='auto', beta_1=0.9,
                beta_2=0.999, early_stopping=False, epsilon=1e-08,
                hidden_layer_sizes=(64, 64, 64), learning_rate='constant',
                learning_rate_init=0.001, max_iter=100000, momentum=0.9,
                n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                random_state=48, shuffle=True, solver='lbfgs', tol=0.0001,
                validation_fraction=0.1, verbose=False, warm_start=False)

In [22]: y_pred = clf.predict(X_test)
         # print clf.best_score_
         # print clf.best_estimator_
         prediction1=clf.predict(X_test)
         print('Accuracy', metrics.accuracy_score(prediction1,y_test))

         Accuracy 0.4756944444444444
```

Accuracy of MLP model = 47.57%

CNN Model:

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristic. [28].

CNNs use relatively little pre-processing compared to other classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

CNN model was the best for our classification problem. After training numerous models we got the best validation accuracy of 71.28% with 18 layers, softmax activation function, rmsprop activation function, batch size of 32 and 700 epochs.

```
In [201]: #Removed the whole training part for avoiding unnecessary long epochs list
          cnnhistory=model.fit(x_traincnn, y_train, batch_size=16, epochs=700, validation_data=(x_testcnn, y_test))
```

```
67
Epoch 695/700
1151/1151 [==============================] - 12s 10ms/step - loss: 0.1991 - acc: 0.9679 - val_loss: 2.0164 - val_acc: 0.4
429
Epoch 696/700
1151/1151 [==============================] - 12s 11ms/step - loss: 0.2000 - acc: 0.9600 - val_loss: 2.0256 - val_acc: 0.4
602
Epoch 697/700
1151/1151 [==============================] - 12s 10ms/step - loss: 0.1963 - acc: 0.9618 - val_loss: 2.0371 - val_acc: 0.4
325
Epoch 698/700
1151/1151 [==============================] - 13s 12ms/step - loss: 0.1982 - acc: 0.9618 - val_loss: 2.0142 - val_acc: 0.4
498
Epoch 699/700
1151/1151 [==============================] - 12s 10ms/step - loss: 0.1940 - acc: 0.9687 - val_loss: 2.0136 - val_acc: 0.4
325
Epoch 700/700
1151/1151 [==============================] - 12s 11ms/step - loss: 0.1976 - acc: 0.9626 - val_loss: 2.0697 - val_acc: 0.4
464
```

```
In [214]: #Load the model
          # loading json and creating model
          from keras.models import model_from_json
          json_file = open('model.json', 'r')
          loaded_model_json = json_file.read()
          json_file.close()
          loaded_model = model_from_json(loaded_model_json)
          # load weights into new model
          loaded_model.load_weights("Model_CNN.h5")
          print("Loaded model from disk")
```

```
Loaded model from disk
```

```
In [21]: # evaluate loaded model on test data
         loaded_model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
         score = loaded_model.evaluate(x_testcnn, y_test, verbose=0)
         print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
acc: 71.28%
```

Emotion Classification from facial patterns (Video):

We perform facial pattern recognition 2 ways- using openCV and building a Fisherface classifier and using facial landmark recognitions to predict 7 different emotions from our RAVDESS dataset using a linear Support Vector classifier.

What I found surprising was that, both methods yielded much better precision and accuracy results, than the best model accuracy of the Audio emotion detection system. Let us dive into details of the steps involved. Here we will discuss both methods simultaneously as the process steps involved are pretty much the same with some extra steps for landmark recognition.

Pre-Processing dataset:

The data pre-process step was rather a long one in video emotion detection. To extract a face from a video and predict the emotion on it, we first need the face extracted from one or multiple images from the video. These frames are captured and saved in a separate folder for analysis. For my project scope, I took just a random middle frame from each video. In my working directory, I created a separate folder for extracting the images to and one another folder for sorting the images. We group the emotions into 7 buckets- considering ‘calm’ and ‘neutral’ as ‘neutral’ since this tuning helped improve accuracy of the final model. Under each emotion, we now would have about 350 or more frames to work with.

Extracting faces:

The classifier will work best if the training and classification images are all the same size and have (almost) only a face on them (no clutter). We need to find the face on each image, convert to grayscale, crop it and save the image to the dataset. We can use a HAAR filter from OpenCV to automate face finding.

OpenCV provides 4 pre-trained classifiers. They are-

- Haarcascade_frontalface_default
- Haarcascade_frontalface_alt2
- Haarcascade_frontalface_alt
- Haarcascade_frontalface_alt_tree

To detect as many faces as possible, we use all of them in sequence and abort the face search once we have found one. I got them from the OpenCV directory and extracted them to my working directory. Next would be the step to detect, crop and save faces as such.

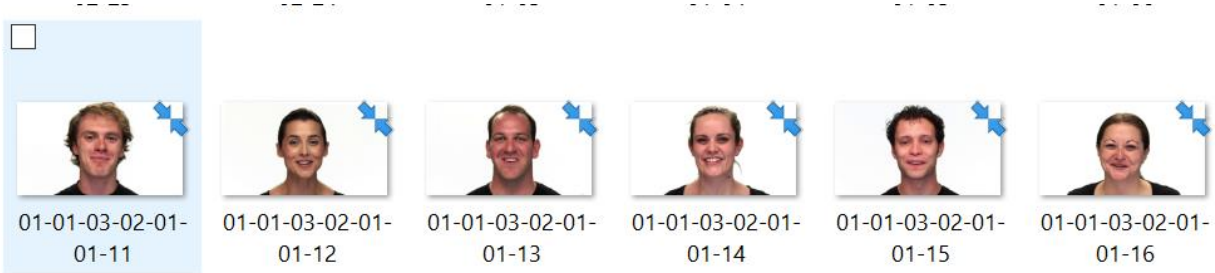
The images are processed by getting converted to greyscale, optimizing the contrast with an adaptive histogram equalization and displaying it in the new folder.

The output is rather long and looks like this-

```
        else:
            facefeatures = ""
        #Cut and save face
        for (x, y, w, h) in facefeatures: #get coordinates and size of rectangle containing face
            print ("face found in file: %s" %f)
            gray = gray[y:y+h, x:x+w] #Cut the frame to size
            try:
                out = cv2.resize(gray, (350, 350)) #Resize face so all images have same size
                cv2.imwrite("Output_dataset\\%s\\%s.jpg" %(emotion, filenumber), out) #Write image
            except:
               pass #If error, pass file
        filenumber += 1 #Increment image number
for emotion in emotions:
    detect_faces(emotion) #Call functiona
```

```
face found in file: Image_classdata\surprised\02-01-08-02-02-02-03.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-04.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-05.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-09.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-10.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-11.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-12.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-13.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-14.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-15.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-16.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-17.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-18.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-19.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-20.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-21.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-22.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-23.jpg
face found in file: Image_classdata\surprised\02-01-08-02-02-02-24.jpg
```

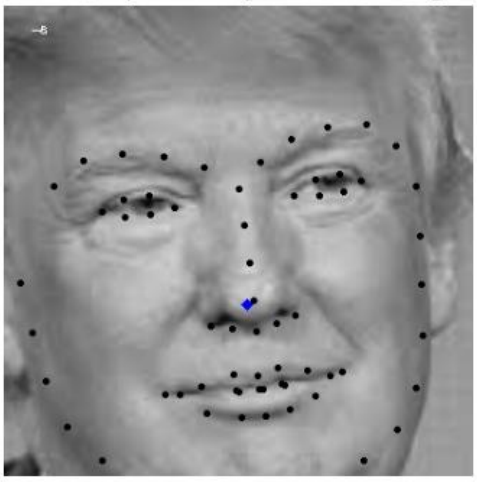
And a sample output image extraction and greyscale conversion happens like demonstrated below. These are a set of 6 different actors in RAVDESS dataset expressing “happy” emotion.

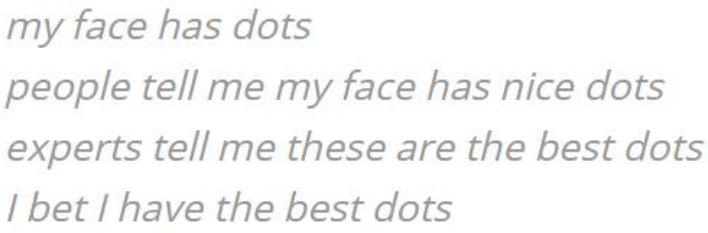


And with the magic of OpenCV, we identify, crop the faces in the above images, convert them to greyscale and save them under the ‘happy’ folder. Below are the same set of actors with faces identified-

Extracting features from faces:

This is a step we use for landmark facial recognition, which we don't use in Fischerface classifier technique. For this methodology, we use Dlib library, which is a C++ library in python, used for extracting facial landmarks. OpenCV cropping, conversion to greyscale process would be the same here with the additional code (attached in appendix) resulting in each face with a lot of dots outlining the shape and all the "movable parts". The latter is of course important because it is what makes emotional expressions possible.





The next step is to find ways to transform these nice dots overlaid on the image faces into *features* to feed the classifier.

Features are little bits of information that describe the object or object state that we are trying to divide into categories. Each bit of information that is passed to the classifier can be considered a *feature*, and based the same feature set for each image face, we would be pretty accurate if we chose the features well.

How we extract features from our source data is actually where a lot of research is, it's not just about creating better classifying algorithms but also about finding better ways to collect and *describe* data. The same classifying algorithm might function tremendously well or not at all depending on how well the information we feed it is able to discriminate between different objects or object states. If, for example, we would extract eye color and number of freckles on each face, feed it to the classifier, and then expect it to be able to predict what emotion is expressed, we would not get far. However, the facial landmarks from the same image material describe the position of all the "moving parts" of the depicted face, the things you use to express an emotion.

Next step, we extract the coordinates of all face landmarks. These coordinates are the first collection of features. look at the coordinates. They may change as the face in our image moves to different parts of the frame. The face could be expressing the same emotion in the top left of an image as in the bottom right of another image, but the resulting coordinate matrix would express different numerical ranges. However, the relationships between the coordinates will be similar in both matrices so some information is present in a location invariant form, meaning it is

the same no matter where in the picture the face is.

The most straightforward way to remove numerical differences originating from faces in different places of the image would be normalizing the coordinates between 0 and 1. However, there is a problem with this approach because it fits the entire face in a square with both axes ranging from 0 to 1.

A less destructive way could be to calculate the position of all points relative to each other. To do this we calculate the mean of both axes, which results in the point coordinates of the sort-of "centre of gravity" of all face landmarks. We can then get the position of all points relative to this central point. The "centre point method" also introduces extra variance; the centre of gravity shifts when the head turns away from the camera, but this is less than when using the nose-tip method because most faces more or less face the camera in our sets.

Also, faces may be tilted, which might confuse the classifier. We can correct for this rotation by assuming that the bridge of the nose in most people is more or less straight and offset all calculated angles by the angle of the nose bridge. This rotates the entire vector array so that tilted faces become like non-tilted faces with the same expression.

Data Splitting:

The dataset has been organized and is ready to be recognized, but first we need to actually teach the classifier what certain emotions look like. The usual approach is to split the complete dataset into a training set and a classification set. We use the training set to teach the classifier to

recognize the to-be-predicted labels, and use the classification set to estimate the classifier performance. The reason for splitting the dataset, as discussed previously, is because, estimating the classifier performance on the same set as it has been trained is unfair, because we are not interested in how well the classifier *memorizes* the training set. Rather, we are interested in how well the classifier *generalizes* its recognition capability to never-seen-before data.

In any classification problem, the sizes of both sets depend on what we are trying to classify, the size of the total dataset, the number of features, the number of classification targets (categories).

For now, we randomly sample and train on 80% of the data and classify the remaining 20%, and repeat the process 10 times.

Fisherface Classifiers:

In this step, we initiate the fisher face classifier using the Fisherfacerecognizer_create() call and then proceed with the usual steps of defining a function to get the file list, randomly shuffle it, split it into 80-20, append data to training and prediction list, generate labels 0-7, convert image to greyscale, append image array to training data set and repeat the whole process for the prediction set. We get the following output-

In [46]:
```
#Now run it
metascore = []
for i in range(0,10):
    correct = run_recognizer()
    print ("got", correct, "percent correct!")
    metascore.append(correct)
print ("\n\nend score:", np.mean(metascore), "percent correct!")
```

```
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 93.96887159533074 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 93.96887159533074 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 69.45525291828794 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 88.52140077821012 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 93.19066147859922 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 67.70428015564202 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 91.05058365758755 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 75.09727626459144 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 69.06614785992218 percent correct!
training fisher face classifier
size of training set is: 2061 images
predicting classification set
got 66.92607003891051 percent correct!


end score: 80.89494163424125 percent correct!
```

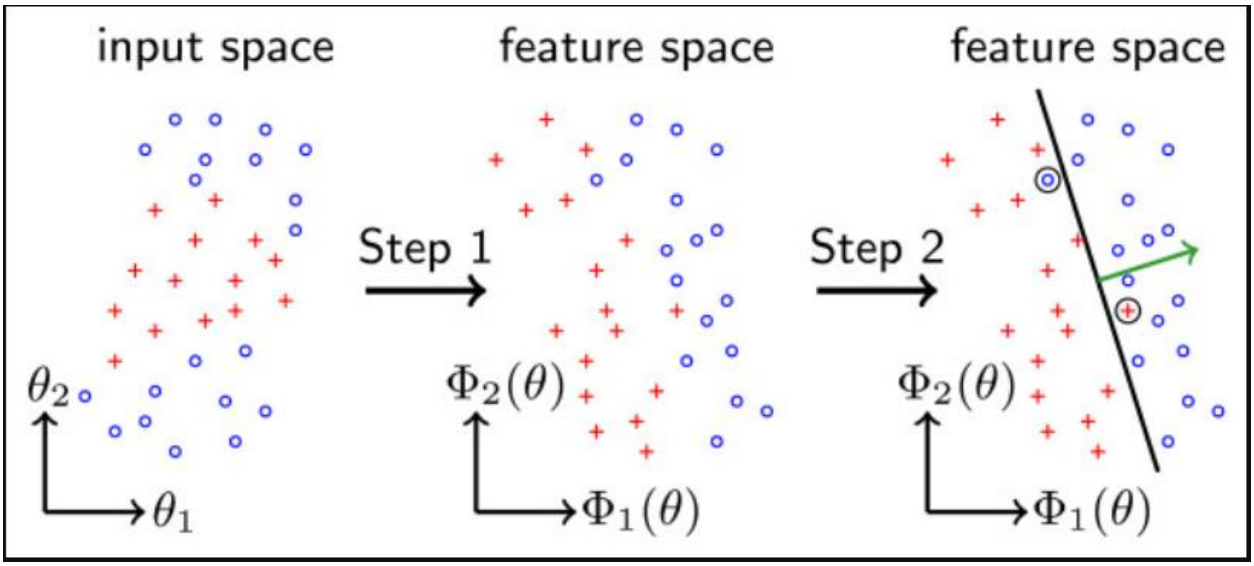
Fisher face classifier seems 80.89% accurate in predicting the emotions in the test data set!

Linear SVM Classifier:

Linear SVCs are classes capable of performing multi-class classification on a dataset. They are similar to SVC with parameter set as kernel = ‘linear’, the difference being that they are

implemented in terms of liblinear rather than libsvm, so Linear SVC has more flexibility in the choice of penalties and loss functions and should scale better to large number of samples.

Here is a 2-D visualization of what SVC tries to achieve-



With our dataset, we feed the features that we extracted earlier into the Linear SVC classifier. The amount of feature combinations fed as inputs should not affect the accuracy level of the Linear SVC.

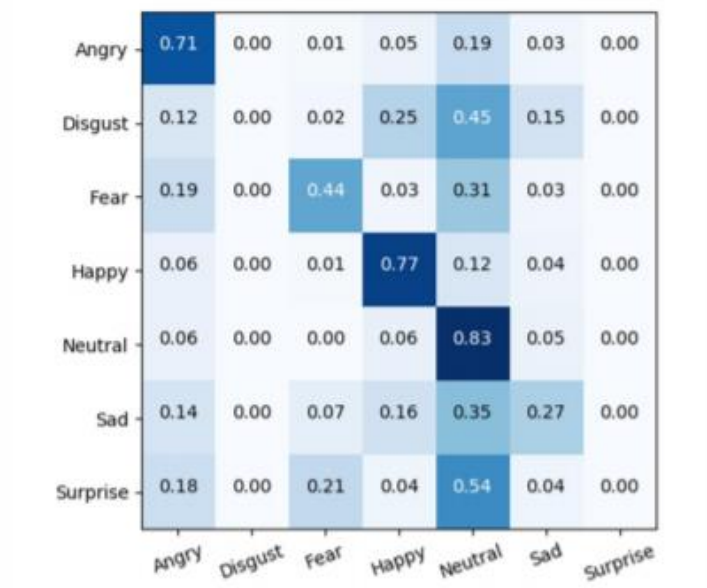
Linear SVC classifier showed 87.88% accuracy in predicting the emotions in the test data set!

Multimodal fusion accuracy:

For the fusion method, I used the BS-Fusion technique[30], which learns from the bream search method. As there is a combinatorial explosion in the number of feasible subset (2N subsets for N models), we employ a sampling procedure with the goal of filtering out subsets that are less likely to yield good results. We use a beam search of the size K and select top K subsets in each turn. The selection approach is based on the classification accuracy of the subset.

Through the BS-Fusion, a subset of emotion possibilities is selected according to the classification performance on the validation dataset. In the testing dataset, we achieve 60.34% accuracy.

The resulting confusion matrix is projected below-

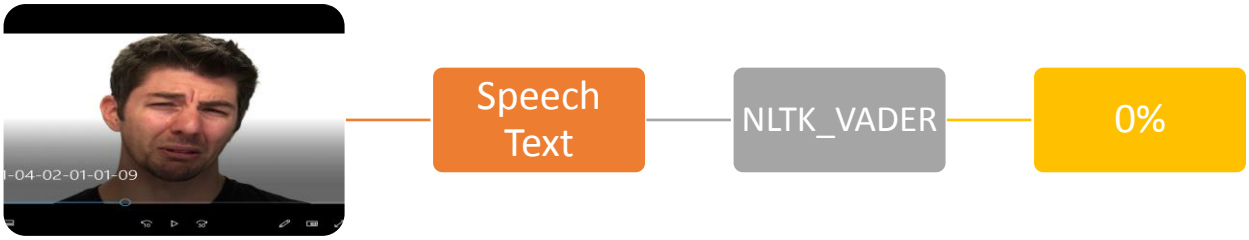


CHAPTER 7

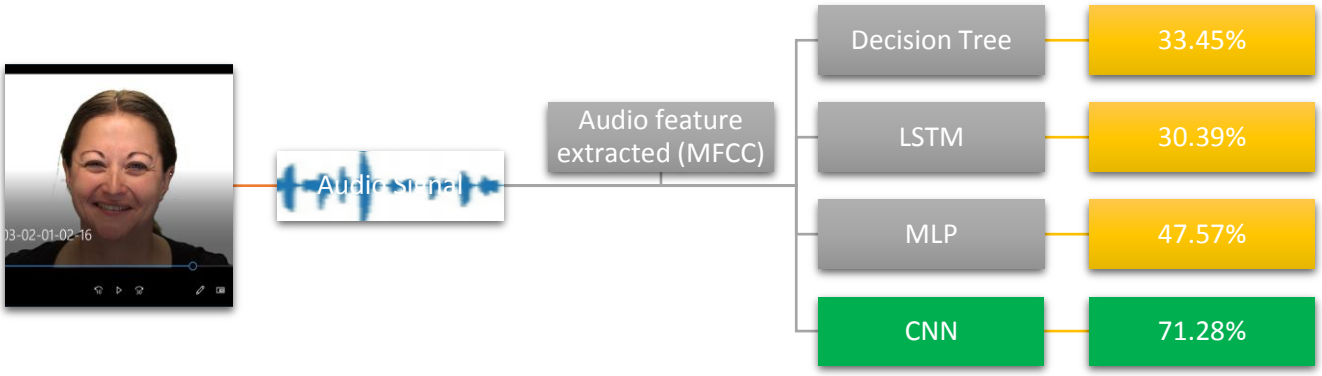
RESULTS COMPARISON

Project Outcome Summary:

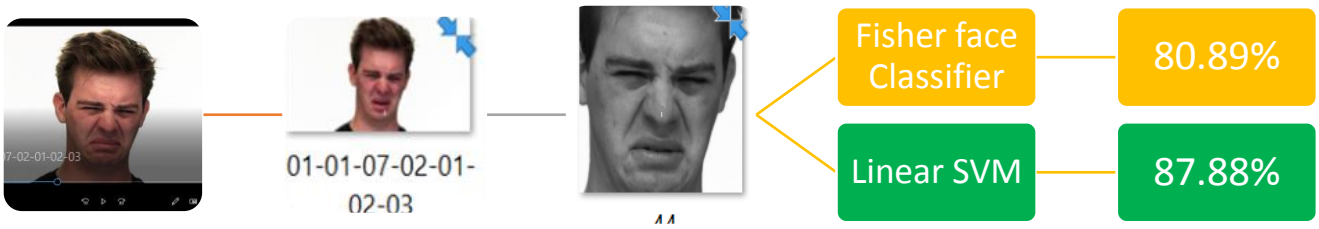
Emotion detection via speech text gave 0% accuracy.



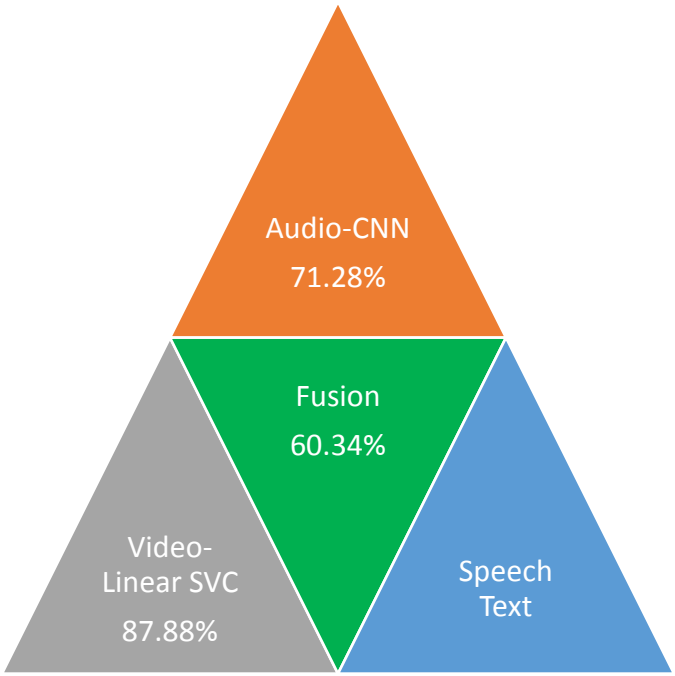
Emotion detection from audio features, at best, gave a prediction accuracy of 71.28%.



Emotion detection from facial features, at best, gave a prediction accuracy of 87.88%.



Emotion detection from a fusion methodology gave a prediction accuracy of 60.34%.



CHAPTER 8

CONCLUSION

An interesting demo to rest our case:

Now that we have summarized the results of the case study comparisons, I would like to conclude with style!

Let us look at the practical application and potential benefits of this case study outcome. Here, I will demonstrate with a rather interesting experiment I did with Google Siri, which is supposed to be our "voice assistant" with its Automatic Speech Recognition capability taken to the next level with voice-activated responses.

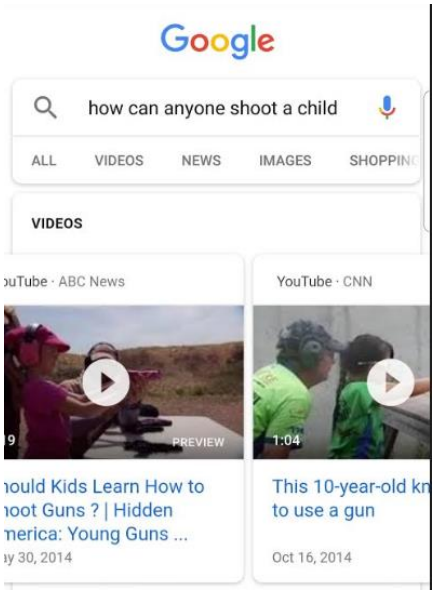
My conversation with Siri goes like this-

Me (frustrated tone) : How can anyone shoot a child?

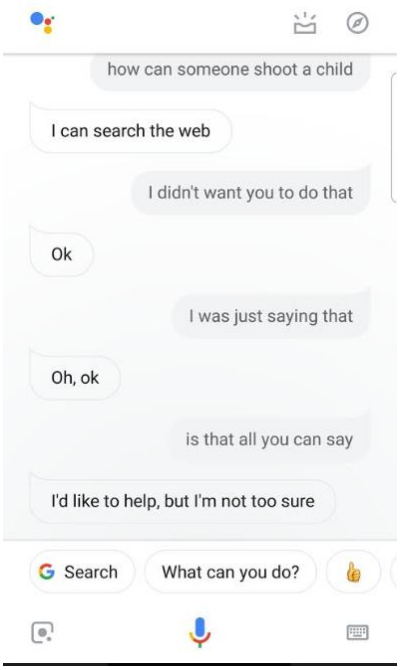
Siri: Let me search on that.

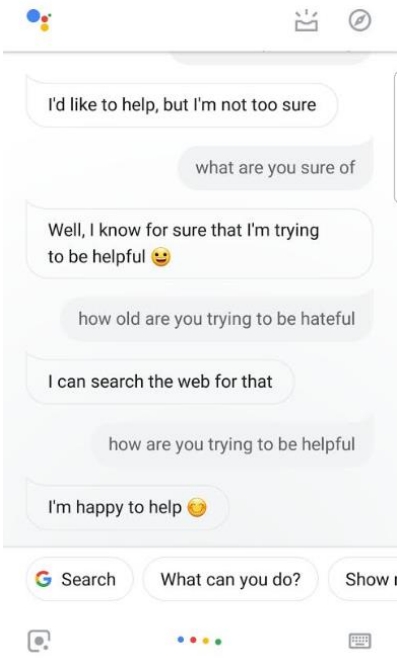
Me (sarcastic tone): yeah right you will!

Siri: (Gives me the below search result- videos about how kids learn to shoot!)



Me: (trying one more time from the start)





The last bit of that conversation was messed up because of my accent, which Siri was not able to follow. I had to spell it out again word by word, slowly- "how-are-you-trying-to-be-helpful" for which it goes- "I'm happy to help" with a smiling emoji!

That was one annoying conversation!

The purpose of this demonstration is that though Siri and other voice-activated products out in the market right now have been trained on algorithms that can help answer our questions about weather/ travel/ news etc., they still cannot be called a 'real' assistant, or neither can they have

‘real’ conversations with us, if they cannot understand the emotional tone of the speaker that is not expressed explicitly in words.

Yet another rather shorter conversation this time, let us call this **Scenario-1**:

Me (tired from work): I had a long day today Siri!

Siri: My apologies… I don’t understand

Me (a bit annoyed): Well, I was just saying you that I had a very long day today!

Siri: Take a look at these results. The first is from Quora. (proceeds to show me a screenshot of the question in Quora- “What does the phrase ‘I had a long day’ mean?)

Now, imagine **Scenario-2**:

Me (tired from work): I had a long day today Siri!

Siri: Oh, I am sorry to hear your stressed tone. Let me switch on the AC, dim the room lights using the smart-home system and play some soothing music for you to relax and take a nap. May I suggest that a warm bubble bath would be a great de-stresser to you right now?

Me: (feeling relaxed already) That sounds like a wonderful idea! You are the best!

Siri: I see the frown on your face has disappeared already! I am glad I was of help!

Me: You were! Thank you Siri!

Now, that is what we could call a smooth and real interactive ‘assistant’!

Hence, we prove with this case study and demo that Automatic Speech Recognition has the potential to be made not just more robust and useful but also extremely powerful when combined

with non-verbal cues like audio and video, to make it a complete and comprehensive Multimodal Automatic Speech Emotion Recognition system.

Future work and Enhancements Proposed:

Emotion detection, together with Affective Computing, is a thriving research field. Few years ago, this discipline did not even exist, and now there are hundreds of companies working exclusively on it, and researchers investing time and resources on building affective applications. However, emotion detection has still many aspects to improve in the future years. I have tried to list a few here-

- The fusion model that I have attempted to build in this case study could definitely be improved upon to a large extent with fine tuning of model parameters.
- It would have been very interesting to be able to extend our results and experimental processes to databases recorded in other languages. Comparing performance results between different language databases would have given us key information about the degree of universality of our acoustic features.
- Applications which extracts information from the voice needs to be able to work in noisy environments, to detect subtle changes, maybe even to recognize words and more complex aspects of the human speech, like sarcasm. RAVDESS audio files were relatively clean, so we did not get to apply 'silence removal' or 'noise removal' steps.

- Most people use glasses nowadays, and thus, I am curious to see how this could affect the emotion detection from facial features. Again, RAVDESS was a relatively clean dataset with not even one of the 24 actors wearing glasses.
- Just like facial feature recognition, body gesture movements, behavior, heart-rate etc., could be read too, which would be a great addition to the fusion model. As far as I researched, there is no known database right now available to train models on with respect to these inputs.

Humans can understand a fellow human, in most cases, within milliseconds of seeing the person. In order to match this accuracy rate, we need to have more than one input to train our algorithm on. Multimodal Automatic Emotion Recognition would be pave the right path for this in the future.

REFRENCES

(1) Giannakopoulos T (2015) pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis. PLoS ONE 10(12): e0144610. https://doi.org/10.1371/journal.pone.0144610

(2) Amos D (2018) The Ultimate guide to speech recognition with Python. Retrieved from: https://realpython.com/python-speech-recognition/#working-with-audio-files

(3) Valentin Vielzeuf, Stephane Pateux & Frederic Jurie (2017). Computer Vision and Pattern Recognition: Temporal Multimodal Fusion for Video Emotion Classification in the Wild.

(4) Ljubljana (2012) Exchange Project Work: Emotion Recognition from speech signals.

(5) Garcia-Garcia.J.M.,Lozano.M.D & Penichet V.M (2017) Emotion Detection-a technology review. DOI: 10.1145/3123818.3123852.

(6) Zheng Lian, Ya Li, Jianhua Tao & Jian Huang (2018). Computer Vision and Pattern Recognition : Investigation of Multimodal Features, Classifiers and Fusion Methods for Emotion Recognition. *arXiv:1809.06225v1 [cs.CV]*

(7) L. Chen, X. Mao, Y. Xue, and L. L. Cheng, "Speech emotion recognition: Features and classification models," Digit. Signal Process., vol. 22, no. 6, pp. 1154–1160, Dec. 2012.

(8) H. Cao, R. Verma, and A. Nenkova, "Speaker-sensitive emotion recognition via ranking: Studies on acted and spontaneous speech," Comput. Speech Lang., vol. 28, no. 1, pp. 186–202, Jan. 2015.

(9) T. L. Nwe, S. W. Foo, and L. C. De Silva, "Speech emotion recognition using hidden Markov models," Speech Commun., vol. 41, no. 4, pp. 603–623, Nov. 2003.

(10) S. Wu, T. H. Falk, and W.-Y. Chan, "Automatic speech emotion recognition using modulation spectral features," Speech Commun., vol. 53, no. 5, pp. 768–785, May 2011.

(11) M. Grimm, K. Kroschel, E. Mower, and S. Narayanan, "Primitives-based evaluation and estimation of emotions in speech," Speech Commun., vol. 49, no. 10–11, pp. 787–800, Oct. 2007.

(12) Huang, Baker and Reddy. Communications of the ACM (2014). "A Historical Perspective of Speech Recognition" Vol. 57 No. 1, Pages 94-103. Jan 2014.

(13) Boyd C (2018) "The Past, Present, and Future of Speech Recognition Technology" Retrieved from: https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf.

(14) L. Kessous, G. Castellano and G. Caridakis. (2010) Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis. Journal on Multimodal User Interfaces, 3(1), pp.33-48.

(15) P. Gupta and N. Rajput.(2007) Two-stream emotion recognition for call center monitoring. In 8th Annual Conference of the International Speech Communication Association.

(16) N. Fragopanagos and J. G. Taylor. (2005) Emotion recognition in human–computer interaction. Neural Networks, 18(4), pp.389-405.

(17) T. Vogt and E. André. (2005) Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition. In Multimedia and Expo, (pp. 474-477).

(18) F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier and B. Weiss.(2005) A database of german emotional speech. In Interspeech (Vol. 5, pp. 1517-1520).

(19) Livingstone, S.R. & Russo F.A.(2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. Retrieved from: https://doi.org/10.1371/journal.pone.0196391.

(20) Hardik S, Gohil H & Gosai D (2018) A Review On Emotion Detection And Recognization From Text Using Natural Language Processing.

(21) Python programming tips (2017) Retrieved from: http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

(22) Pandey P (2018). Simplifying Sentiment Analysis using VADER in Python (on Social Media Text). Retrieved from: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f

(23) Ljubljana, (2012). Emotion Recognition From Speech Signals Erasmus Exchange Project Work.

(24) Colah (2015). Understanding LSTM Networks. Retrieved from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

(25) Rosenblatt, Frank. x. (1961) Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC.

(26) Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams.(1986) "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press.

(27) Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome.(2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York,NY.

(28) LeCun, Yann.(2013). "LeNet-5, convolutional neural networks".

(29) Lian Z, Le Y, Tao J & Huang J (2018) Investigation of Multimodal Features, Classifiers and Methods for Emotion Recognition.

(30) Vielzuf V, Pateux S & Jurie F (2017). Temporal Multimodal Fusion for Video Emotion Classification in the Wild. **arXiv:1709.07200v1 [cs.CV]**

APPENDIX A

Description of Python code files

Code file	*Description*
AudioEmo_FeatureExt_DecTree.py	Audio feature extraction; Decision tree training; Scoring, fit and prediction accuracy of Decision tree on test data.
AudioEmo_LSTM.py	LSTM model building; Model summary; LSTM prediction accuracy on test data.
AudioEmo_MLP.py	Train MLP model, test and record prediction accuracy.
AudioEmoDetect_CNN.py	Train, model summary, load model, record prediction accuracy and predict emotion into csv.
CNN_Predictions.csv	The actual vs predictions table in a .csv file.
SpeechTextEmo.py	Finds the sentiment of extracted text using NLTK_Vader.
VideoImageEmo_Fisherface.py	Extract image frames from video .mp4 files; use opencv to extract face from image, crop and grey scale; train model, test and record prediction accuracy.
FusionEmo.py	Predicts accuracy rate for fusion methodology.