BACHELOR OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

Inspiring Excellence

Early Threat Warning Via Speech and Emotion Recognition from Voice Calls

AUTHORS
Ifaz Ishtiak
Mohammad Mazedur Rahman
Md.Razaul Haque Usmani

SUPERVISOR
Hossain Arif
Assistant Professor
Department of CSE

A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
B.Sc. Engineering in CSE

Department of Computer Science and Engineering
BRAC University, Dhaka - 1212, Bangladesh

December 2018

Declaration

It is hereby declared that this thesis /project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

Authors:

Ifaz Ishtiak
Student ID: 15101118

Mohammad Mazedur Rahman
Student ID: 15101043

Md.Razaul Haque Usmani
Student ID: 14241005

Supervisor:

Hossain Arif
Assistant Professor, Department of Computer Science and Engineering
BRAC University

December 2018

The thesis titled Early Threat Warning Via Speech and Emotion Recognition from Voice Calls
Submitted by:
Ifaz Ishtiak Student ID: 15101118
Mohammad Mazedur Rahman Student ID: 15101043
Md.Razaul Haque Usmani Student ID: 14241005
of Academic Year 2018 has been found as satisfactory and accepted as partial fulfillment of the requirement for the Degree of Bachelors of Science in Computer Science and Engineering
(An example is shown. It must be replaced by the appropriate Board of Examiners)

1.	Name of Supervisor Designation Address	Chairman
2.	Name of Head of the Dept. Designation Address	Member
3.	Name of Internal Member Designation Address	Member
4.	Name of Internal Member Designation Address	Member
5.	Name of Internal Member Designation Address	Member
6.	Name of External Member Designation Address	Member

Abstract

The aim of this system is to identify potential cases of threats, and provide an early warning or alert to such cases. This will be based on voice such as voice chat over telecommunication networks or social media. The intended result will be achieved in three major steps. At first, the conversion of speech to text from both real time audio recordings and from accent groups will be applied using primarily IBM Watson's Speech to Text. This will then be used to identify possible trigger words or word patterns from a classified selection of threat-related and negative words. And finally, the same audio source will be utilized for detecting emotions from the frequency shifts through vocal feature extraction from audio input and processing it using multiple classifier algorithms such as Support Vector Machines (SVMs), Random Forests and Naïve Bayes. Libraries such as LibROSA will be applied to extract primary audio features such as Mel Frequency Cepstral Coefficients (MFCC) to generate accurate predictions. The system yields a result of approximately 84% using the SVM RBF (Radial Basis Function) kernel, which highlights the accuracy of emotion detected based on the speech.

Keywords— Emotion Recognition; Support Vector Machines; Speech to Text; Random Forest; Feature Extraction; MFCC

Table of contents

List of figures

List of tables

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Objective . . . 1
- 1.3 Motivation . . . 2
- 1.4 Orientation . . . 2

2 Literature Review **3**
- 2.1 Speech To Text . . . 3
- 2.2 Trigger Word Detection . . . 4
- 2.3 Emotion Recognition . . . 5
- 2.4 The Psychology of Threat and Speech Under Stress . . . 6

3 Algorithm, Database and Tools **9**
- 3.1 Classfiers and The Confusion Matrix . . . 9
- 3.2 Algorithms . . . 10
  - 3.2.1 Support Vector Machines . . . 10
  - 3.2.2 Random Forest Classification . . . 12
  - 3.2.3 Naive Bayes . . . 13
- 3.3 Databases . . . 15
- 3.4 Libraries . . . 17
  - 3.4.1 PyAudioAnalysis . . . 17
  - 3.4.2 LibROSA . . . 18
- 3.5 Software and Tools . . . 19
  - 3.5.1 Application Program Interfaces (APIs) . . . 19
  - 3.5.2 Program Dependencies and Tools . . . 21

4 System Design **23**

5 Implementation **27**
5.1 Speech to Text Conversion 27
5.1.1 Testing the IBM Watson STT Accuracy 27
5.2 Trigger Word Detection 28
5.2.1 Neutrino API Implementation 28
5.2.2 Customized Word Detection Algorithm 29
5.3 Emotion Recognition 33
5.3.1 Feature Extraction 33
5.3.2 Library Implementation 41
5.3.3 A Comparative Study 45

6 Experimental Results **49**

7 Conclusion and Future Work **51**
7.1 Conclusion 51
7.2 Future Work 52

References **53**

Appendix A **55**

Appendix B **61**

List of figures

3.1 Support Vector Machine and Hyperplane 11
3.2 Simplified Model of Random Forest Algorithm 12
3.3 Fundamental Principle of The Naïve Bayes Algorithm 14

4.1 Work flow of the Overall System 25
4.2 End User Application Workflow 26
4.3 Workings of the Trigger Word Detection Algorithm 26

5.1 Workings of the Trigger Word Detection Algorithm 30
5.2 Comprehensive Methodology of the Word Detection Algorithm 32
5.3 Simplified Description of the Zero Crossing Rate 34
5.4 Side-by-side Analysis of Amplitude and ZCR Frequencies Extracted from Different Audio. 35
5.5 Process of MFCC Feature Extraction 36
5.6 Mel Scale to Frequency Distribution Chart 37
5.7 Triangular Mel Filter Banks 38
5.8 Spectrograms generated depicting the emotions of (Left) Anger and (Right) Fearful 39
5.9 (Left) Represents the Centroid from the signal; (Right) Centroid from the signal over time 40
5.10 The Spectral Rolloff Point 41

6.1 Results Achieved from Trigger Word Detection 49

B.1 ZCR, Energy, MFCC and Spectrogram for CALM Emotion 61
B.2 ZCR, Energy, MFCC and Spectrogram for HAPPY Emotion 61
B.3 ZCR, Energy, MFCC and Spectrogram for SAD Emotion 62
B.4 ZCR, Energy, MFCC and Spectrogram for ANGRY Emotion 62
B.5 ZCR, Energy, MFCC and Spectrogram for FEARFUL Emotion 62

List of tables

5.1 Computed Results of Profane Word Detection from Neutrino Word Filter API 29
5.2 Confusion Matrix Generated from Training an SVM Classifier 43
5.3 Overall Accuracy, C parameter and F1 Score 43
5.4 Emotion Recognition Test Result from an Audio Clip 43
5.5 Test Results from LibROSA Feature Extraction 47
5.6 Test Scores Using SVM RBF Classifier on 5 Emotions 48

6.1 Highest Accuracy Obtained for the Emotion Recognition 50

A.1 78% Accuracy Using SVM RBF on 5 Emotions 55
A.2 69% Accuracy Using Random Forest on 5 Emotions 55
A.3 64% Accuracy Using SVM Poly Kernel on 5 Emotions 56
A.4 47% Accuracy Using SVM Linear Kernel on 5 Emotions 56
A.5 42% Accuracy Using Naive Bayes on 5 Emotions 56
A.6 71% Accuracy Using SVM RBF on 8 Emotions 57
A.7 71% Accuracy Using Random Forest on 8 Emotions 57
A.8 Confusion Matrix, Accuracy and F1 Score from PyAudioAnalysis on 5 Emotions 58
A.9 Confusion Matrix, Accuracy and F1 Score from SVM RBF on 8 Emotions . 58
A.10 Confusion Matrix, Accuracy and F1 Score from Random Forest on 8 Emotions 59

Chapter 1

Introduction

1.1 Introduction

In recent years, the advancements in networking and telecommunications has proved to be imperative for the global technological progress. Enhanced communication and instant updates in messaging, video-based communication, location tracking and even traffic conditions around the user are excellent examples of how much the world wide web has contributed to create a link around the globe. However, with its benefits, there must also be some drawbacks. Among the many are the crimes that are secretly taking place through the same instant messaging and voice calls. One article [1] illustrates this clearly; telling about a German male who ended up killing two women whom he claimed to have met in online chat rooms. This is just one of the minor cases in contrast to all the other indescribable events that have occurred to this day. This calls for taking certain measures and fast; hence the aid of machines and machine learning is required, where both voice and text can be effectively used to hint at potential danger and threat before any situation escalates.

1.2 Objective

The aim is to create a system that combines multiple disciplines such as speech to text, word recognition as well as emotion recognition that will aid in assisting in investigations, evaluating voice conversations and give predictions and estimations for potential threats. This will be done by both analyzing the words spoken during the call, as well as the predicted emotions using machine learning and the results will be compared to come to a conclusion

whether a threat exists or not, the topic of the conversation and the way the message is conveyed.

1.3 Motivation

The technology of speech recognition - also referred as STT (Speech to Text) or ASR (Automatic Speech Recognition) - has come a long way in terms of development and has opened up numerous potentials for use in practical everyday life, from creating language aid to medical prescription applications and now it is increasingly becoming famous in creating autonomous home appliances and wearable smart technology.

There has been great deal of advancement in technology that has made the work of humans a lot easier and reduced the need of large labor. But some functionalities are still being worked on for the betterment of mankind. The requirement of recognition of human emotion has been of great need as this can be a breakthrough in the identification of human psychology. Speaking can be of different tones which can identify how a person is meaning to say that particular speech. Thus, to identify the emotion or manner of saying behind the speech is important. This is where the need of machine learning comes into play that can identify the type of emotion being applied in a particular sentence by analyzing the vocal characteristics or in other words, audio features, of the particular audio sample. The features that are identified can be applied into the well-known classification algorithms in machine learning which then classifies the emotions according to the matching features.

1.4 Orientation

The paper is organized into the following chapters: Chapter 2 highlights some of the work related to the tasks relevant to our research (STT, word recognition as well as emotion classification). Chapter 3 gives a short introduction and working principle of the tools, libraries and algorithms used during the research. This is followed by Chapter 4, which describes the proposed design created for the system, its proposed interface and working process. Chapter 5 explains the implementation of the system - starting from the database selection, to the features extracted for the data using selected libraries. It also includes multiple findings in the duration of the research, an in-depth description on the modifications and comparisons needed to be made that lead to the final results. The results are discussed in Chapter 6, and the conclusions are drawn, along with potential future implementations in Chapter 7.

Chapter 2

Literature Review

2.1 Speech To Text

For the Speech to text conversion, multiple approaches are observed. A similar work [2] has been done where speech to text is performed for mobile devices to enable and enhance the voice recognition and command features in the newer smartphones. Through efficient algorithms and the addition of neural network, such as the General Regression Neural Network (GRNN), over 95 percent accuracy in speech recognition is achieved. A VAD or Voice Activity Detection algorithm pipeline is used which involves two algorithms performing distinct tasks. One to calculate the signal features directly from the audio energy and the other performs similarly, but is determined by the Zero Crossing Rate in the signal and Mel Frequency Cepstral Coefficient (MFCC) is used for the feature extraction. Based on these values, it provides an estimation (VAD decision variable) of speech recognition. Moreover, the introduction of Neural Networks and creation of distinct identifiers in the database which allows for particularly uttered syllable detection, boosts the overall performance of the system. The recognition results are effective and similar to our approach to convert text from speech.

Other relevant methods of speech to text conversion includes this article [3], where the STT uses MATLAB's Kalman filter, which provided excellent accuracy in noisy environments and over a certain Decibel threshold. Emphasizing on the reduction of background noise to imitate speech in practical life situations, a Bidirectional Kalman Filter is applied which the author claims to provide the best results, especially in noisy environments. Similarly, MFCC features are extracted along with Linear Predictive Coefficients (LPC) for increased accuracy. A comparative study has been made with another process where Hidden Markov Model or HMM is being used with a TIDIGIT database which proved to be better with an overall accuracy of 90 percent. Another approach [4] proposes the use of Microsoft's Speech

API (SAPI) to detect language such as distinct Bengali words through XML document feeds, which proved to be accurate 75 percent of the time.

Fortunately, in recent times, the principles of speech to text systems has been made easy and readily available in the form of applets and online services such as the IBM Watson's Speech to Text service [29]. This allows quick conversion of speech to text in real time, provided a volunteer speaker or even multiple formats of recorded audio files. Further in the discussion, many test cases are provided which highlights the accuracy, efficiency and the convenience of having a complex program readily available which is able to provide greater accuracy than the above mentioned.

2.2 Trigger Word Detection

For the detection of offensive or trigger words, [5] mentions a couple ways for recognizing profane or offensive words from a text, most of them sorts the text in some way and then compares it with one or more dictionaries, i.e. brute force, along with the incorporation of some other features. The most notable of them is the lexical syntactic feature which not only checks the offensiveness of the word but also checks the offensiveness in user level. It yields with the precision of 98.24% and recall of 94.34%. For the lexical feature extraction, 'Bag-of-Words' used to be very popular in the early research programs but using this approach yields low accuracy in subtle offensive language detection and gives high false positive rate during heated conversation. The N-gram approach, most notably the Bi-gram and Tri-gram, is a much safer and improved approach as it also includes information of the words nearby context. For the syntactic feature extraction, we introduce natural language parsers to parse sentences on grammatical structures to avoid unrelated word sets in the offensiveness detection of the user.

Also, after much research, we found multiple applets and online sources which takes in text as input and checks to see if the text is offensive or not, what are the offensive words within it, replace those words with symbols, etc. The one which seemed much reliable, fast and with a higher success rate was the Bad Word Filter API, developed by Neutrino API [30], which allowed us to quickly list out all the profane words from any given text.

However, due to certain restrictions, the optimal and desired results were not obtained. For instance, it lists a limited number of only profane words and slangs. Our system requires a lot more since only profane words are not enough to indicate a threat or a threatening situation during a conversation. There are numerous words which express negativity and may lead to a worse situation and these must be taken into account. Thus, we have devised a

small program that list most negative meaning words in the English language, as well as the profane words, to work in conjunction to identify them in the speech.

2.3 Emotion Recognition

For the emotion detection, there has definitely been intensive work and research on this field of how to recognize human emotion via speech or other physical traits. We have studied about such works and most of them deal with how to extract values of sound such as pitch, amplitude, frequency and time, to match the corresponding specific emotion. In one such work, the data has been used from the Berlin Database of Emotional Speech (EMO-DB) [6] to extract emotion from the German language and for American English the data was used from the Speech Under Simulated and Actual Stress (SUSAS). The software they used was the Waikato Environment for Knowledge Analysis (WEKA) [28] which is a JAVA based software and they have executed their emotion extraction using the most significant classification features of the emotional states. The algorithm used for this work was Sequential Minimal Optimization under the supervised learning model of SVM. The feature selection was done by the technique provided by WEKA was CFS Subset Eval. This algorithm uses a feature evaluator known as Correlation base Feature Selection (CFS). Thus, through this they have selected 10 different sets of parameters. The results that they have found out after performing several different algorithms on the features is that the Sequential Minimal Optimization (SMO) algorithm has given the best of the outputs on both for the German and American English languages.

Another work [7] has been carried out but here there has not been any use of algorithm rather the comparison of some acoustic parameters. The acoustic parameters used were mean overall fundamental frequency, overall mean energy, overall mean standard deviation of energy, mean overall jitter and mean overall shimmer. The data source for this research included a 27-year-old female and a 32-year-old male. Seven different emotions were considered for this study which were happiness, sadness, cold anger, hot anger, interest, elation and neutral. The number of sentences they have taken to compare the results of all the acoustic parameters of corresponding emotion were 70. Thus, the two subjects were exposed to neutral environment and recorded each of the emotional conditions for all the 70 sentences. The values recorded against each emotional state for each parameter was used to identify the differences in values of the parameters in each emotion.

Furthermore, a study [8] was conducted on the pitch difference of each form of speech and the best form of algorithm was found out executing the data that has been used to identify them. The source of the data used was from Pitch Tracking Database from Graz University

of Technology (PTDB-TUG). There source has provided 2342 recording of 10 males and 10 females. The algorithms used for speech identification using pitch where Entropic Signal Processing System (ESPS), Average Magnitude Difference Function (AMDF) and the Praat algorithm. The ESPS and AMDF algorithm was used in Wavesurfer and Praat algorithm was used in Praat Script. The final evaluation was done in Python. The end result of this evaluation was that the ESPS was the most accurate results compared to that of the other two algorithms.

A comparable study [9] of emotion recognition has also applied multiple machine learning algorithms, such as the SVM and Random forest, to compare between the efficiency and accuracy. The Random forest algorithm proved to be the most accurate, at 81.05%, followed by Gradient Boosting at 65.23% and finally the SVM algorithm, a comparatively poor performance score of 55.89%. Like most others, the training and testing of the model is done using the Berlin Database of Emotion Speech, on a total of 535 samples which includes 7 emotions in total: Neutral, Sad, Happy, Fear, Anger, Boredom and Disgust. The features selected are the MFCC and Energy, that are extracted and applied on the aforementioned classifiers.

This particular work [10] also relates to some level to what we are trying to achieve in our work. There is similarity in the use of the MFCC (Mel Frequency Cepstral Coefficients) features to detect any sort of noise that can remotely indicate possible threats. Such noise would include glass breaking, gun shots, explosions and other forms of threatening sounds. They have used both audio and visual effects of the possible threatening situation using network cameras and the samples of sound collected are then processed by main node, where a decision is made using supervised algorithms to identify and give the level of danger present in the monitored area. Simultaneously, a signal of warning is sent to a crisis management center. The results after application, has shown that there has been a 79 percent accuracy for threat identification. They have also tried to use the SVM along with PCA but that could not match the accuracy of the single frame SVM. The other algorithm used also did not give enough accuracy that is the Dynamic Time Wrapping (DTW).

2.4 The Psychology of Threat and Speech Under Stress

To justify the psychological aspect of potential threat from the emotion recognition using similar features such as energy, pitch and MFCC, etc. multiple studies have been done in the past. This is termed as stress or speech under stress. The method for identifying such a case is to compare the audio features with and without stress – whether or not the values have impacted greatly. This book [16] suggests that psychological threat revolves around the

relation between anger, fear and anxiety; and showcases a shift diagram to prove that during stress and arousal – the excitation level of a certain emotion – the results are different than otherwise. The frequency, pitch and the speaking rate changes when one is under stress and then eventually goes back to normal as stress decreases.

This book [17] also specifically defines psychological speech stress as "a response to a perceived threat" or any scenario involving an individual to experience anger, anxiety or fear. It mentions stress can be caused by a variety of reasons, from everyday workload to assuming chances of being inflicted with pain (under threat). However, both sources acknowledge and admit the fact that – as psychology goes – stress is different for each person and therefore is difficult to define it in a binary way. Each person handles, cope and react to stress differently and some through training, not react to stress at all. Lastly, more factors come into play with numerous accents, dialects, knowledge, frequency and mastery of the language itself, etc.

Using these as the underlying principle of speech under stress, more studies have been done where this psychology has been considered through the extraction of audio features in digital form and classified using suitable algorithms. This work [18] in particular, is performed on the SUSAS (Speech Under Simulated and Actual Stress) database, focusing only on the stress and its related emotions labeled Neutral, Angry, Lombard (an effect where sound quality is altered to adjust with a noisy environment) [19] and Loud. Multiple stress related features such as pitch, MFCC and other spectral features are used on a multiclass SVM to achieve around a 100 percent accuracy rate.

Chapter 3

Algorithm, Database and Tools

3.1 Classfiers and The Confusion Matrix

Below is described the supervised algorithms that we have used for the entire thesis work. The understanding of supervised algorithm is that the algorithm is being trained with data that consists of already known outcome to be able to predict the same outcomes using the previously learned training set of data. This work requires classification of data which has been achieved using classifiers under supervised algorithms. The confusion matrix each of the classifiers have given is used to identify the number of correct and incorrect prediction the classifying algorithm has been able to produce. The True Positives (TP) and True Negatives (TN) in a confusion matrix gives the number of correct classification and the False Positives (FP) and False Negatives (FN) gives the number of incorrect outcomes when we consider only two classes. In case of multiple classes, the matrix expands in dimensions accordingly. Of these positives and negatives, a few calculations are derived that define specific values. As part of the implementation of the pyAudioAnalysis library, these values are displayed to measure the accuracy of the classifiers:

• Precision – The number of correct predictions of a label out of all predictions made for the label/class. PRE = TP / (TP+FP)

• Recall – The number of correct predictions out of total predictions for all classes. REC = TP / (TP+FN)

• F1-Score – The weighted average of both precision and recall. F1 = (2*PRE*REC)/(REC+PRE)

3.2 Algorithms

3.2.1 Support Vector Machines

The supervised learning models which are associated with other learning algorithms in order to research on data for regression and classification analysis are called Support Vector Machines, or SVMs. It falls under the category of non-probabilistic binary linear classifiers. The advantage of using SVM is that it can deal with large set of feature dimensions and also work even when sample number is less than the feature number. The different kernels, provides diversity and versatility which can be used for specific decision functions. There are different SVM modules such as SVC and NuSVC which allows the libsvm library to use different kernels whereas the LinearSVC is based on liblinear library and only supports linear SVM kernel. The SVC used can have kernels such as linear, polynomial and RBF (Radial Basis Function)/Gaussian kernels. These are used in basis of the dataset that is at hand. If there is no basic prior knowledge of the type of dataset that is available, then it is recommended to try out the sample dataset in linear kernel to see if the features or dimensions are separable linearly. If there is a need of higher levels of dimension for the dataset then we use polynomial or RBF kernel.

To explain the basic concept of SVMs, let us refer to the figure below where the orange circles represent some data about Oranges and the red circles represent some data about Apples. If we were to separate the data of these two types of fruits, we could draw a lot of different lines on different angles in order to separate this two sets of data. But what SVM does is, it creates a line that best splits these two sets of data. This line is called the Hyperplane and it is said to be equally and as far away from the Support Vectors, which are the circles closest to the hyperplane, or another way of saying this is that, it maximizes the widest possible Margin between the support vectors. The support vectors are the decision functions. The support vectors are called so because they are the reason why this hyperplane exists. In fact, if there were no other points present in this figure except the support vectors, the hyperplane would have still looked the same. The hyperplane maximizes the margin between the support vectors in such a way that the support vectors are separate from each other with equal distance. So now if had some random set of data of some kind of fruit which is either an apple or an orange, we can pass this data through the SVM model and classify it, i.e. if it lies left of the hyperplane it is considered to be an orange or if it lies right of the hyperplane it is considered to be an apple.

So how does SVM maximize this margin between the hyperplanes? This is a form of constrained optimization problem because we are trying to maximize this margin which involves optimization and the circles are constrained because they cannot go within the margin

[26]. These types of constrained optimization problems can be solved by the implementation of the Lagrange Multipliers technique.

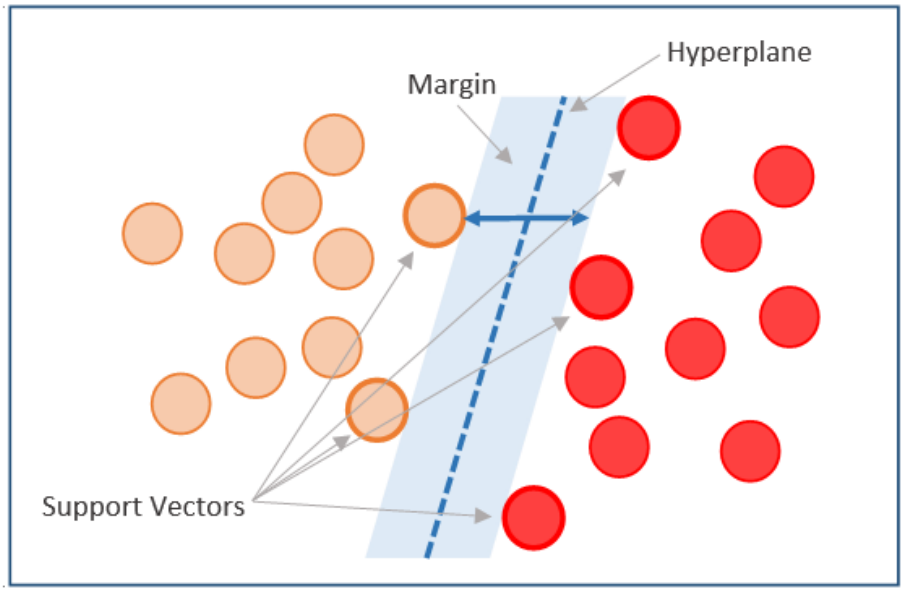

Fig. 3.1 Support Vector Machine and Hyperplane

The example that we have seen above involves only two features or dimensions. That is the reason why we can separate the data above linearly, i.e. with a line. But it is not always that we will have only two features for classification. If we had three features, we could separate the data with a three-dimensional plane, if we had four features or more, we would separate it with a hyperplane with multiple dimensions which would be hard to visualize but can still be implemented by SVM. Also, we are more likely to get data that are not perfectly separate from each other. We are more likely to run into one or more misclassified data which can disrupt our classification of the data. Fortunately, the SVM has a parameter called the C Parameter which allows us to set the amount of misclassified data that we want to penalize. Also, SVMs are not limited to just two sets of classes. We can have multiple number of classes and the SVM would still manage to classify each of them separately. Lastly, SVM is also popular for its Kernel Trick. For example, if we obtained a graph that is inseparable with a line, then what SVM does is, using the kernel trick it creates another dimension which would then help us to separate the data with a linear plane. The basics of how a Support Vector Machine is explained. There are multiple parameters taken by the classifier which include C parameter and Kernel.

The Kernel sets from SVM are used. On the other hand, there is also degree parameter which is only used in polynomial SVM and has a default value of 3. This basically gives

us the number of dimensions inside the polynomial function when the data is not linearly separable. As the number of features is high thus, we need greater dimensions and the degree specifies the number of dimensions required. The next parameter that is used is the gamma which is applicable in poly, RBF and sigmoid kernels. This is value is predefined to be 1/n, n is being the number of features. The max_int parameters take the maximum number of iterations and finally the random state parameter, which is used to randomize the provided dataset for the SVM available for all the types of kernels.

3.2.2 Random Forest Classification

The Random Forest, also known as Random Decision Forest, is a classification and regression technique that functions by constructing a series of multiple Decision Trees during the training phase of the algorithm, which is then used to decide the final decision, or classification, based on the maximum number of trees chosen by the random forest. This is a form of ensemble learning: meaning this algorithm takes another set of multiple machine learning algorithms and creates a bigger machine learning algorithm. In this case the use of Decision Trees to ultimately classify the dataset.

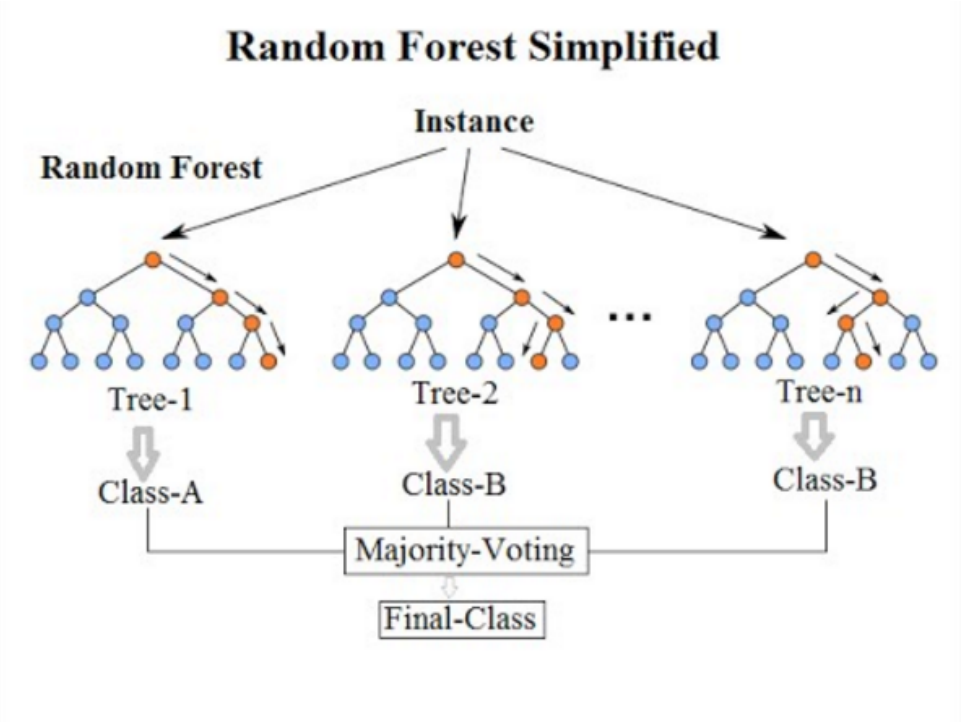

Fig. 3.2 Simplified Model of Random Forest Algorithm

To understand how random forest works, we first need to understand what is a Decision Tree. A Decision tree is a tree diagram, because of its tree-like structure, which is used to determine or classify some form of action. Each of the branches of the tree can represent an occurrence, reaction or some sort of decision made by that branch. A decision tree is based on some important terms –

• Entropy – It is the measure of the randomness of the given set of data. Everything on the decision tree and the decisions that it makes is based on entropy. The higher the entropy, the dataset becomes more unpredictable.

• Information Gain – It is the measure of the difference from going from higher entropy to lower entropy by splitting the dataset, i.e. Information Gain = High Entropy – Low Entropy.

• Leaf Node – It is used to carry the decision throughout the branches of the decision tree.

• Decision Node – It has two or more branches and is situated above the leaf node.

• Root Node – It is the decision node that is situated at the top of the decision tree.

So, what random forest does is, it creates multiple decision trees based on multiple trained datasets. Each decision trees do not contain all the data points of the entire set but instead, each decision tree have equal number of data points of different combinations from the list of data points. When we pass a test data through this decision trees, each of these trees have their own unique output. The train data is then assigned to the class which was common from the majority of the results given by the decision trees.

The parameters that are present in the Random Forest Algorithm includes n_estimators. This basically sets the number of trees in the forest. The criterion parameter can be of two values ether gini or entropy where the latter is used for information gain and the former is for Gini impurity. Other parameters include max_depth meaning the maximum number of nodes, sample_split is the minimum number of samples to make a leaf node, max_features are the number of features used for making best split and random_state which again like other algorithms, provides randomization of the samples. In the Random Forest algorithm, the split that is chosen is not the best split from all the features from a random subset of features. This increases the overall bias of the algorithm and due to the averaging, the variance also decreases to compensate for the higher bias resulting in a better model.

3.2.3 Naive Bayes

The Naïve Bayes algorithm is a machine learning algorithm that is being used based on the Bayes Theorem. So, to understand how this classifier works, it necessary to have some knowledge about the Bayes Theorem. The Naïve Bayes is a form of probabilistic algorithm

that uses the advantage of probability to classify considering the features present for the given dataset. The basics of the Bayes Theorem is that there is a certain set of possible outcomes of two different instances. Thus, the formula of the theorem as shown in the diagram is that we find the probability of something happening given that a certain condition is being maintained. Thus, to explain it in a particular scenario, we can consider an example. It can be considered that there are two set of machines that produces a certain number of wrenches. It is known that which machine produce which - meaning the wrenches are labelled by the machine producing it. Based on the certain number of productions by the two machines it can be found out the probability of a choosing a defect wrench among the collection of wrenches provided that it is from machine two. This can be done using the Bayes Theorem. The probability is found by finding out the probability of the number of wrenches that are from machine two given that it is defect and multiplying that with the probability of a chosen wrench being from machine two. The product of this calculation is then divided by the probability of choosing only wrenches that are defected. This in turn will give us the result of how to find the reverse condition probability that is choosing a wrench that is defective given that it is from machine two. This is the Bayes that is being used in the Naïve Bayes algorithm using the same concepts to make prediction and classification. The values that are being used have separate meaning. The finding that is on the left side of the equation is known as posterior probability. The values on the right side is likelihood multiplied with prior probability that is being divided with the marginal probability.

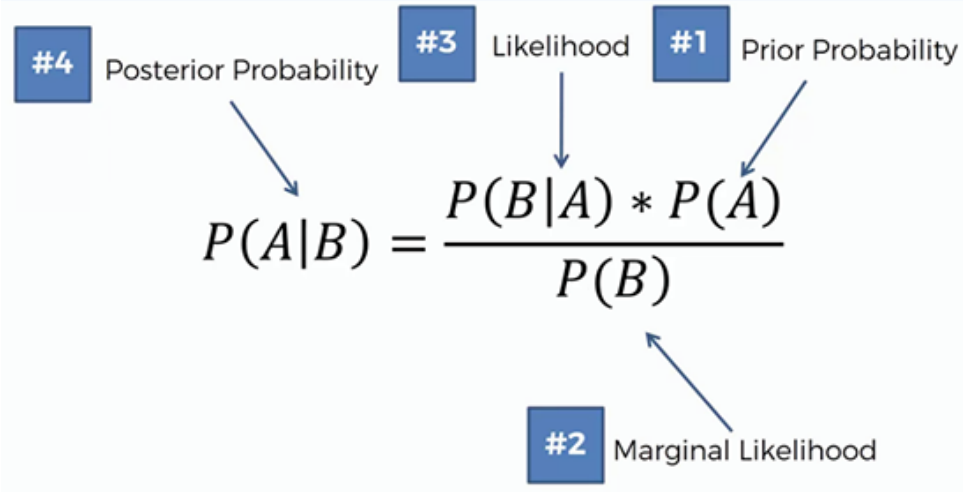

Fig. 3.3 Fundamental Principle of The Naïve Bayes Algorithm

The equation above describes how the prediction and classification is made. The values of B basically represent the features of the data set. The values of A are the specific classes. Thus, we are to find the posterior values and compare for each class. The prior probability of

the class is calculated meaning the probability of a certain point or item being in that class. The marginal likelihood is the calculation of the probability that a certain item would exhibit the features that are close to the points own feature of the different classes. This basically means we are to consider the number of occurrences of particular class that are closest to that of the added item and find the probability by dividing it with total of all the classes. The next value that we need to find out is the likelihood which is the probability that given it is of a certain class the probability that the added item will exhibit that class's features. Finally, using the found values we are able to find out the posterior probability. For each class this value is then found out and then compared to find out which is the most likely class does the added item belong to.

Naïve Bayes is a very common and highly recommended classifier Algorithm that is used in all dataset classification problems. But this algorithm executes based on a certain assumption that is the conditional feature independence assumption. This basically means that the features of the dataset are not interdependent on each other and all the feature are separate and independent not influencing any other feature present. This is not always the case as correlation between features is very much possible while classifying. Even though Naïve Bayes works under this assumption, it still performs remotely well as a classifying algorithm. This algorithm is very easy to implement and needs a small set of data for classification. In case of large sets of data there is the partial_fit method that can be used to incrementally use datasets so that it does not over fit the memory.

3.3 Databases

In our research period we have come across multiple sources of data availability. This gave us a large scope of choosing amongst those sources. In most cases of thesis work, gathering data is one of the most challenging parts. But in our line of thesis work we have not faced such difficulties. Nevertheless, the sources of our data were not all compatible to the requirements of our work. Thus, we needed to examine these sources separately to match the requirement we needed. Also, most researches [6, 9, 12] were conducted using a Berlin Database as primary database which contained German sentences and words and not executing their work on English database. There were use of Danish database [11] and this would significantly have different output in the type of database that is being used while carrying out the work. The form of data that we came across at first was the voice samples from [31] to convert the sound waves to digitally readable data. However, this dataset was only usable till the speech to text part of our program since it lacked the emotional expression when delivering

the speech i.e. the speech was dictated in a neutral tone rather than being read to express emotion.

An alternate was required to be found against the data we had seen in [31]. Thus, we ended up with The Toronto emotional speech set (TESS) [35]. This had the credibility of The University of Toronto as they would execute this kind of work under great surveillance and perfect form of discipline. The database here was created using two female actors each uttering a set of 200 words under seven general emotional states. They were tested to have thresholds within the normal range and had English as there first spoken language. The actors were aged 24 and 64 to also take into consideration about the variation of vocal characteristics due to age. A common speech phrase of "Say the word ..." was used to utter each of the 200 words that were used. It was still not compatible enough with the type of data set we required, our work needed complete sentences that would have these type of triggering words inside general sentences. This data base did not provide the structure of data we required and only the words were spoken in the seven emotions that have been mentioned. We required a general sentence to have a single emotion flow through it.

With the aforementioned considerations, a quantifiably greater and rigorously tested database in the common English language is selected for our research. This is the Ryerson Audio-Visual Database of Emotional Speech and Song or RAVDESS in short [13]. It is a 'multimodal' database consisting of speech with emotions and songs performed by 24 professional actors. Some fundamental features to make this database relevant and well developed consists of having a diverse cast – different race and genders – of professional volunteers who achieved a total of around 7400 distinct set of recordings. It is moreover performed in 3 possible expressions, namely normal, strong and in a neutral tone. The accuracy, throw, and natural emotion validity is also ensured by over 300 selected group of participants and researchers. It is further versatile by providing visual expressions with its own dataset alongside the audio. Relevant to our research for emotion recognition, the huge database includes 8 distinct emotions: happy, calm, angry, fearful, disgust, neutral, surprised and sad emotions and this particular database is based on a neutral North American accent which further classifies for the ease of data use. Access to a large database, which is so critically tested (by many) for accuracy and genuineness of expression is the primary reason for RAVDESS to be the operating database for the research. For this, the audio data will be fed to the program which will extract the vocal features and create models for the classification algorithms such as SVM to identify given emotions.

3.4 Libraries

3.4.1 PyAudioAnalysis

The pyAudioAnalysis [14] is an open source and versatile library which is able to easily handle audio related tasks. The aim of this library is to provide multiple sound and audio-based functions which are easy and ready to use. Command for these options can be either customized if there is a need for changes in parameters or can be easily called in a python supported command prompt window. The library also gives in depth and analytical contents such as the ability to produce spectrograms and chromagrams (for music related features), etc. This library has contributed to a large part of our study, primarily due to its multitude of available functionalities which proves to be user and beginner friendly in the line of audio processing and analysis. This is because it highlights and simplifies the important steps related to preparing the data, training the classifier model and testing the model directly on newer data sources. For our research, the pyAudioAnalysis library is used for feature selection from the RAVDESS recordings database. By default, it is capable of preparing a dataset consisting of 34 distinct features - directly from an audio file – and create a Comma Separated Values or CSV file, along with a numpy file for ease of data structure use in python. A total of four files are created, a csv and numpy file consisting of the short-term values of the 34 features, and another pair which contains the mid-term mean and standard deviation. Of the 34 features, 21 features have been used in our comparative study (Details in Section 5). Further convenience regarding feature extraction and visualization includes the ability to extract single or an entire folder of multiple audio files in the Wav (Waveform Audio) and the display of the spectrogram and chromagrams of the selected audio file (as shown in Fig. 5.8).

The library truly excels in providing detailed classification of multiple common classifier algorithms such as SVM, kNN (k Nearest Neighbours), Random Forest and Gradient Boosting. Hence, the data can be trained and classified with ease using this. The labeled data extracted from the previous steps can be used in parameters along with the desired algorithm, to train and generate a model file; classification of the test data will use this model file to predict the data as labeled. Similarly, single and multiple files and folders can be trained and classified. Traditionally, any new test data – before a classifier is to be applied – needs to follow the feature extraction, selection and scaling process. However, this can be easily done with the pyAudioAnalysis by directly setting the audio Wav file as the parameter during classification. The results displayed can be tuned as well; during classification, each file can be labeled individually as well as giving an overall result. In relation to our study, the labels are the five emotions (Calm, Happy, Sad, Angry and Fearful). After training a portion of the database, each test case can be classified separately (giving predictions for each emotion)

as well as give an overall prediction of the test set, based on the accuracy of the trained model. Consequently, it also showcases a confusion matrix to highlight the number of correct predictions made by the trained model.

Yet another useful tool in the library is the fixed segment classification which is also available for HMM based classification. The process allows for fixed sized segmentation of the audio file and performs the trained classifier model on each segment, labeling the estimates for each segment as well. More functionalities include the isolation and removal of silence (0 frequency) in the audio and the vice versa with energy/event detection found in the audio; the Speaker Diarization, where multiple speakers can be identified and their features and frequencies separated – which is done through feature extraction and then applying a K-means clustering and finally a smoothing method [15]. Other features available are audio thumbnailing for music representations, audio recording based on fixed sized segments, and audio conversions from Mp3 to Wav and Wav to CSV and vice versa.

3.4.2 LibROSA

For the purpose of feature extraction from the audio (or .wav) files in our dataset, one of the libraries that we came across was LibROSA [32]. It is an open source, popular python package for the purpose of analysis on signal processing, mainly audio and music. It holds its credibility under the license of ISC. Its documentation states that it provides the basic functions required to create audio information retrieval systems. Some of the basic functions included in LibROSA is described below:

• librosa.core – This is the default sub-module of LibROSA, i.e. can be accessed directly using top-level librosa.* namespace. It is used to load audio files from the directory, perform several spectrogram representations and a wide variety of other tools used in music analysis.

• librosa.decompose – This group of functions are used for harmonic-percussive source separation and generic spectrogram decomposition with the help of another library, called scikit-learn.

• librosa.display – This functions are used for the help of visualization with the help of matplotlib library.

• librosa.feature – This sub-module is used for the feature extraction, such as chroma-grams, mel spectrogram, etc., and manipulation, such as delta features, memory embedding, etc., purposes of audio data.

• librosa.onset – This functions are used for onset detection and strength computation.

For the purpose of our topic and after doing some research, we decided to use LibROSA

to compute MFCCs from each audio in our dataset. We are using the extraction function of mfccs, from the librosa.feature sub-module, i.e. librosa.feature.mfcc(y,sr), where the default parameters, y is the audio time series and sr is the sampling rate of y. LibROSA, by default, resamples audio signals to 22050 Hz or 22.05 kHz; and computes the first 20 MFCCs. The whole process would be described and discussed in the implementation part.

3.5 Software and Tools

3.5.1 Application Program Interfaces (APIs)

IBM Watson Speech to Text Service

For the STT system, the IBM Watsons Speech to Text API [29] and the voice samples from [31] to convert the sound waves to digitally readable data. Essentially, the voice data is sampled to smaller parts and passed on to a neural network which detects the next closest guess for the letter/word provided by the datasets. Moreover, a significant advantage of using the API is that it automatically detects multiple speakers in the speech (Speaker Diarization) as well as separate the strings based on the sentences in the converted text. This proves to be time-saving in terms of handling these tasks manually, or using other plugins or tools.

The functions of a common speech recognition are handled in the regular probabilistic approach where, from a given database, assuming a word is spoken, the language/vocal features will be extracted and multiple calculations will be compared for the highest probability score. Using this, the next letter and word will be recognized. This process of ASR or Automatic Speech Recognition is divided into following sub processes:

A. Speech Segmentation
B. Feature Extraction
C. Calculating language model and comparing with the database
D. Result processing

The types of data used in the workflow are voice samples of people from different nationalities, therefore conducting an accent group dependent recognition, as well as real time voice recording is conducted. Moreover, a dictionary database is employed which in addition includes syllables and pronunciation to segment and identify. The two main components to be extracted and segmented regarding speech are phonemes and prosodies.

Phonemes are the smallest unit of speech with a meaning or semantic. These can be extracted or distinguished from speech into smaller segments to aid in the vocabulary

database formation as well as recognizing the speech segments in terms of finding the next probability of letter or following word. Prosodies define the pitch, tone and stress that is put to enunciating a syllable, phenome, letter or a word in continuous speech. These are differentiated into linguistic, paralinguistic and non-linguistic parameters which can aid in the classification of speech as well as emotion analysis. Linguistic prosodies can be described in written context; whereas paralinguistic prosody refers to speaking styles, stress and accents. Finally, non-linguistic prosody deals with portraying emotions through physical and emotional factors.

A. Speech Segmentation: Based on energy of frequency, the phenomes can be segmented, then calculated for recognition probability. However, practically, in almost all use cases, the speaker will be in an environment with background noise which will impact the frequency, energy and the phenome probability being produced. There are a few ways in which the possibility of noise-based errors can be reduced.

B. Feature Extraction: With a set frequency and time, audio sample are extracted. The data is arranged in a multi-dimensional vector called a feature vector. These sets of concatenated vectors together determine the current formation of sound which helps to further identify the matching letter or word. Varying pass filters and frequencies can also help in optimizing the feature vector in order to achieve best results.

C. Calculating Language Model: In this stage, the feature vectors along with an 'acoustic model' and a language model to match with a dictionary library. A common acoustic model uses the Hidden Markov Model for each phenome or word. The Hidden Markov principle is a probabilistic function containing a Transitioning probability (A) and Output probability (B) based on A, therefore P (A, B). Each state A is checked over till B is obtained.

The language model is optional but can greatly increase accuracy and lower recognition and search times. Language Models are comprised of a set of grammatical rules which provides words and likelihood of the upcoming words. The probabilistic decoding to get the next most likely word is defined by the general equation of speech:

$$[\ w = \text{argmax}w(P(X|w).P(w))\ ]$$

Where: X is the number of observations;
w is the word from the dictionary
P(X|W) is the probability derived from the feature vectors in the Acoustic Model
P(W) is the word probability from the Language Model;

Neutrino Bad Word Filter API

As the name suggests, the Bad Word Filter developed by Neutrino [30], detects and censor negative words such as swears and profanities from a given text. In its current iteration, it can only work with the English language databases, but this simple program is highly efficient. It is comprised of four basic output parameters: providing a binary answer if the input text contains any bad words, the total number and listing of the words, and a filter to allow or block censored contents. The input text can be provided directly as a string or as a URL link (Uniform Resource Locator).

Unfortunately, we were not able to use the Neutrino Bad Word Filter API due to the fact that we had to use a word detection algorithm with a specific set of words (or word dictionary). So, using the concept from the Neutrino Bad Word Filter API, we created our own word detection algorithm using python with our custom dictionary. The dictionary is a list of words, around 5500 of them, all within the context of in general negative, threatening and profane. As a result, we are able to detect any set of words from a string within the dictionary and also classify them accordingly, i.e. a detected word is either assigned negative, threatening or profane based on classifiers assigned to each word.

3.5.2 Program Dependencies and Tools

All the libraries and tools used in the system is based on and compatible with Python and Python Standard Library. To effectively run each tool, there are some plugins or dependencies required which serves different use for data structure, data visualization, data preprocessing, feature extraction, etc.

NumPy – Python library for handling, scaling and indexing data structures such as matrices, vectors and n-dimensional arrays and computing complex mathematical solutions and supporting large variety of mathematical functions; from linear algebra, polynomials, and statistics to complex Discrete and Fast Fourier Transform. This base library has been utilized to create dataframes for our dataset and for indexing purposes.

Matplotlib –Matplotlib [34] is a Python library that supports a huge number of scientific and mathematical expressions and calculations, ability to plot in various map, charts, histograms, etc. and generate both raster (pixel) and vector as output. The display of each visualization can be fully customized, for instance the font size, colors, labels, etc. This is used in out study to generate frequency charts along with spectrograms for each feature.

Sklearn – Sklearn or scikit-learn [33] is an open source machine learning library in python which has numerous uses such as data collection and analysis, data preprocessing and clustering. It also supports regression and classification of a multitude of classifier algorithms

and hence is of significant importance in our study. Moreover, it is based on NumPy, SciPy and Matplotlib which allows for an array of mathematical, scientific and visualization tasks to be easily performed.

Hmmlearn – Sequentially, this open source library is built on the previous dependencies mentioned, and provides algorithms and models compatible for learning Hidden Markov Models or HMM in python.

PCA – PCA or Principal Component Analysis is a small program in python that serves a significant purpose: to reduce the number of dimensions and therefore the training complexity of the classifier, meanwhile giving the highest accuracy. Using linear transformation, it is able to preserve the calculations done in a vector containing the data and hence, lower the size to a newer set of Principle Components [20]. This is used in the research to test the accuracy of the classifier by lowering the number of features to enhance performance.

Grid Search– Grid Search allows multiple trained models of varying parameters to be evaluated and provides the best parameters, or model with the highest accuracy based on the data. Multiple classifiers are selected with their parameters (for instance using linear, RBF and polynomial SVM with different C parameters) and then trained on the data. Then these models will be evaluated using Grid Search which uses a cross validation method, to highlight which parameters and model performed the best.

K-Fold Cross Validation – In conjunction with the Grid Search (which requires a cross validation method), the K-Fold Cross Validation technique is applied. The algorithm is simple: the data is randomly split and selected in a fixed range in K splits/observations. For K splits, K-1 is used for training and the other is used as the test set. Depending on the true and average error rate, the accuracy of each parameter is determined, up to K folds. The best performing parameters are then showcased for use to get optimal accuracy. In our comparative study, the Grid Search and K-Fold Cross Validation (CV) is used in an attempt to identify such parameters for best results.

Chapter 4

System Design

As aforementioned, the aim of creating this system is to propose such a study through our prototype that is able to identify potential threats and provide alerts in likely cases. Fig. 4.1 provides a comprehensive look at the overall working process of the proposed system. Firstly, the voice call being held will be recorded in the telecommunication device and will be send over to a server as soon as an internet connection is made; optionally it can be extracted manually as well. The audio recording can now be simultaneously utilized in two ways: first, using the IBM Watson service, the text can be reliably converted from the speech, with its effective automatic speaker diarization and sentence separation. This can allow one to understand the context of the conversation better, since the speaker are separated. Next, this string is passed to the trigger word detection algorithm, which provides a few language classifications, based on the words spoken – whether they are flagged as a threat, profane or slang or just define as a negative sense of the given word (The workings, utility and results algorithm are explained in Fig. 4.3 and further in Section 5).

On the other hand, the same audio file is utilized in the LibROSA library for feature extraction and selection. For our highest achieved results, we have chosen to extract 20 MFCC features, which is normalized for optimal distribution of data and then scaled to ensure an even range and minimize any outliers – exceptionally large or small values in the data. Sequentially, the extracted features will be evaluated using a trained classifier algorithm (in this case the SVM-RBF) to identify the emotion of the speech. Finally, the classified text and the resulting emotion from the algorithm will allow one to evaluate whether there exists any potential threat in the conversation. Overall, this system is designed with the focus to aid in the investigations of such cases where there are suspicions of danger or threat. The underlying principle of the system is that to correctly understand the message, one must first accurately interpret what the other is specifically saying and how it is being conveyed.

Thus, using the speech to text, the language processing and word filtering, the theme of the conversation – particularly, the level of threat – can be understood, and the emotion recognition can help estimate how were the words spoken. And what is the overall mood of the conversation. Both the words and the mood can prove to be a better estimation tool instead of working distinctly with either.

The work flow for an application of the system is shown in Fig. 4.2. The process for the end-user is straight-forward: first, the application is downloaded and installed. Starting the application enables the auto call recorder service which runs in the background until the process is manually stopped or disabled. This also overrides the default voice recorder through app permissions and after a voice call is recorded, it is sent to the storage, and the processing server – as soon as an internet connection is made – for the voice input to be analyzed. As proposed, it then goes through the Speech to Text service and the word recognition algorithm, as well as the emotion recognition classification.

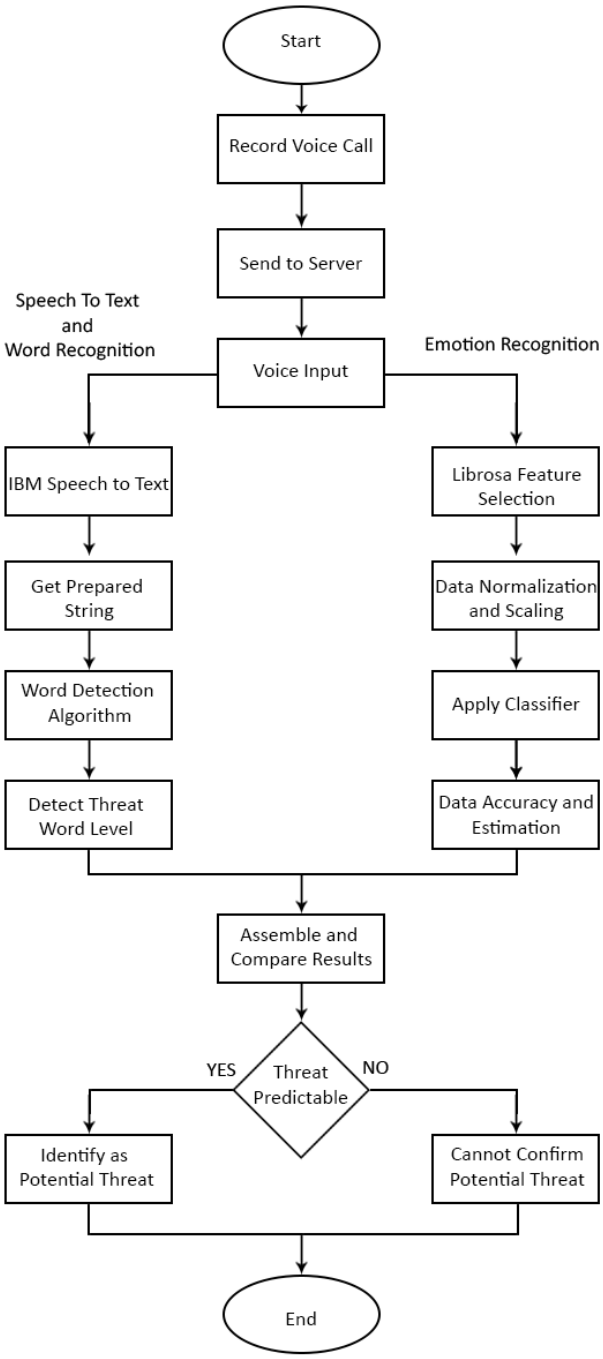

Fig. 4.1 Work flow of the Overall System

Fig. 4.2 End User Application Workflow

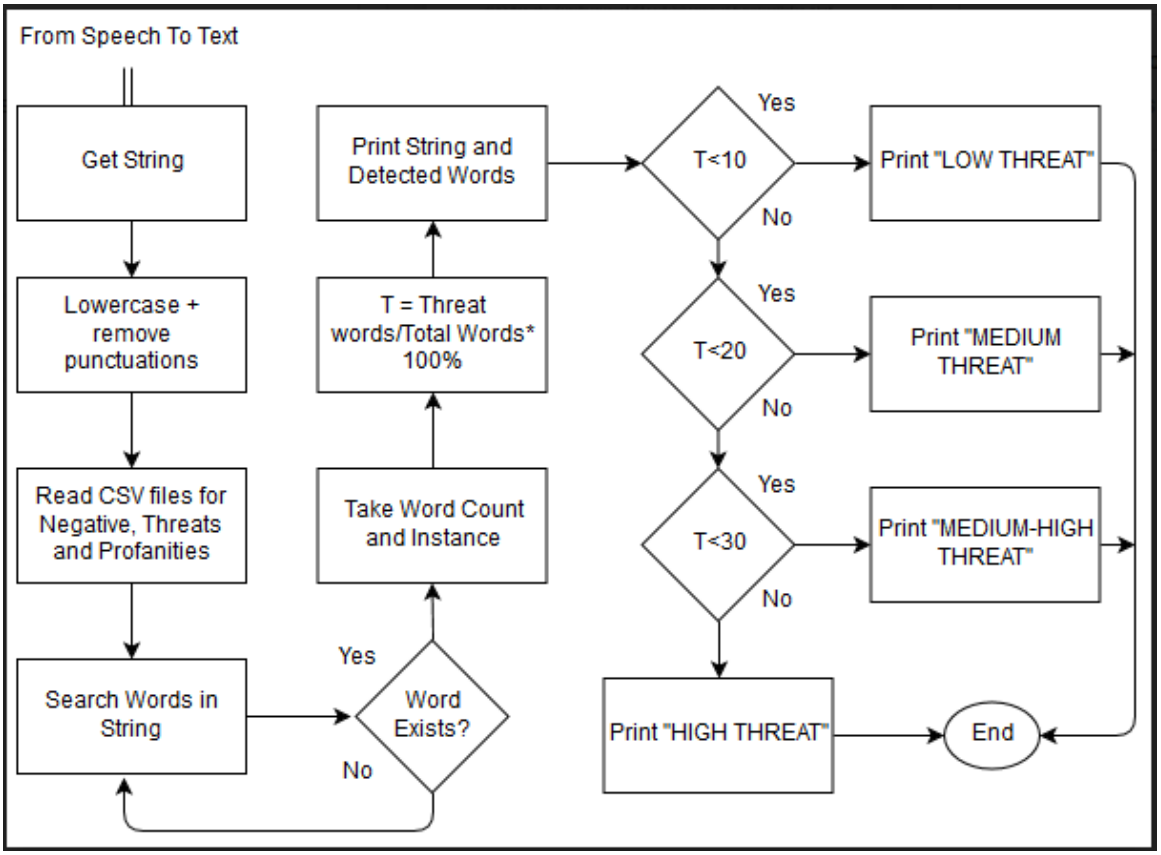

Fig. 4.3 Workings of the Trigger Word Detection Algorithm

Chapter 5

Implementation

5.1 Speech to Text Conversion

5.1.1 Testing the IBM Watson STT Accuracy

To make the program versatile, the data collected is accent group recognition oriented. Vocal data from a wide variety of nationalities and dialects is sampled - both pre-recorded and spoken – and implemented with the IBM speech service to convert the audio to text form. This has led to a variety of results in terms of accuracy. The recorded data [31] consists of over two thousand voice samples reciting a few sentences (67 words) in English:

"Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station."

The ones chosen for the current experiment are one from the British and the local Bengali dialects. However, it must be considered that there are numerous accent groups of the same nationality with different backgrounds and proficiencies at the language spoken. These are the results and their accuracy comparisons of the common accents of each nationality, gathered from the IBM Watson Speech to Text service:

British: *"Please call Stella ask you to bring these things with her from the store 6 spoons of fresh snow peas 5 thick slabs of blue cheese and maybe a snack for her brother Bob. We also need a small plastic snake in a big toy frog for the kids. She can scoop these things into*

3 red bags and we will go meet her Wednesday at the train station."

[65 out of approximately 67 words correct, an above 97% accuracy]

Bengali: *"Please call Stella ask her to bring these things with her from the store. 6 bowls of fresh snow peas 5 big slabs of blue cheese and maybe a snack bar her brother Bob we also need a small plastic snack and a big toy frog for the kids she can score these things into 3 red backs and we will go meet her Wednesday at the train station."*

[62 out of approximately 67 words correct, an above 92% accuracy]

With this small comparison, it is shown that the IBM's speech service alone enables users to gather satisfactory results when converting from speech to text. This data will then be extracted and implemented in an algorithm which will compare it with a set library of data to detect words and patterns in the speech which can help determine the situation of a voice conversation between people.

5.2 Trigger Word Detection

5.2.1 Neutrino API Implementation

During the implementation, one of the sample texts that obtained is being used from the speech to text part and four samples are made from it, each with different attributes, as shown in the table below. The samples derived from the speech to text conversion was implemented in varying ways with the Neutrino Word Filter API. The first sample was the plain text directly, without any modifications. As expected, this sample gave the least percentage error as it did not have any spelling mistakes or grammatical errors. Also, the next sample which is the spoken version of the plain text sample, almost gave the same results. But unfortunately, the desired results were not achieved when we used the sample which had incomplete or incorrect words, and the sample which had some of its characters replaced with some other symbol, as both of these two samples gave the highest percentage error. If we do want to incorporate this feature of detecting any sort of offensive word, it can be achieved quite easily by updating the dictionary and keeping multiple instances of the same words, i.e. incomplete, incorrect words or words with symbols replacing characters.

Table 5.1 Computed Results of Profane Word Detection from Neutrino Word Filter API

Input Type	Actual Result	Experiment Result	Percentage Error
Plain Text	11	9	18.181%
Spoken Text	11	8	27.272%
With Incomplete/ Incorrect Words	7	4	42.857%
With symbols (*,!,#,$,&)	3	1	66.667%

5.2.2 Customized Word Detection Algorithm

The aim of the Trigger Word Detection is to detect trigger words from a string or a list of strings. The strings are obtained from the speech to text part where each line spoken by each speaker is classified as a string. The detection part is done by a brute force comparison algorithm where each word in the string is compared with a list of words, also known as dictionary, and the words that match between these two lists are listed together. This list of words shows all the trigger words in the input string. Furthermore, with the help of specific classifiers associated with each word in the trigger word dictionary, we are able to classify each of the trigger words obtained from the trigger word detection algorithm accordingly.

For now, the trigger word list is based on three different types of words. Negative words are those which have an overall negative meaning or effect, threatening words are words usually used to cases of dealing with threats or verbally abusing someone or in regards to something, and profane words are list of words which fall under the boundaries of profanity. As it is a very basic program, so, we are using only three classes of words. But the program can be made much better by using more than three classes of words. So, in order to make the trigger word dictionary, we had to collect words based the three criteria mentioned above. The trigger word dictionary was created and updated from multiple sources but we had to clean it manually in order to remove redundant and misdirecting words. The original list of words that he had collected was around six thousand and two hundred, in total, which was then refined to around five thousand five hundred words. The program also displays the percentage of threat and profanity, and an alert message based on estimation obtained from

the percentage of threat, for example, ‘MEDIUM-HIGH THREAT’ for threats more than 20 percent.

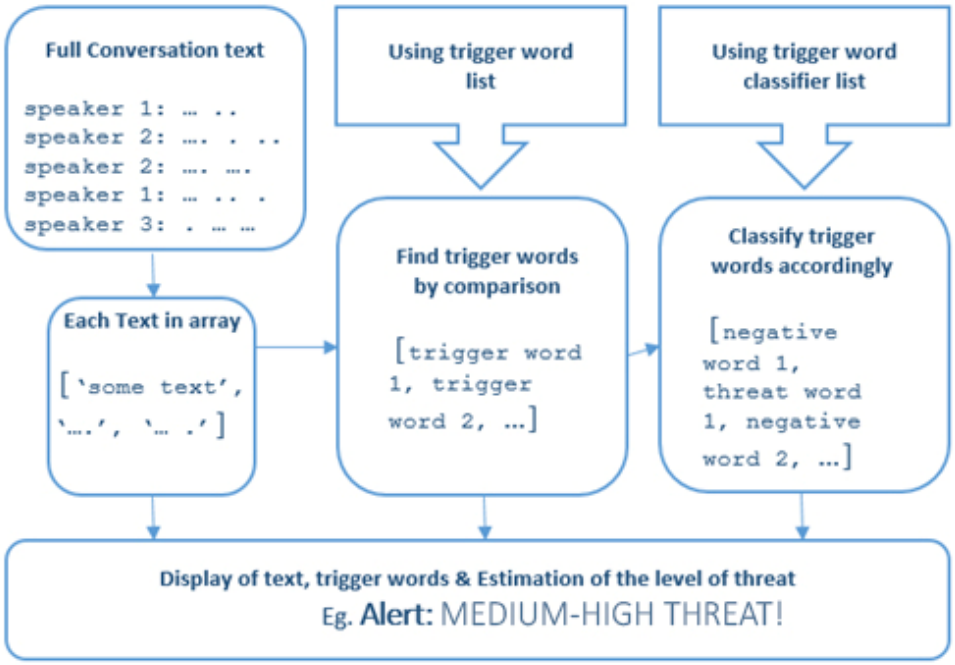

Fig. 5.1 Workings of the Trigger Word Detection Algorithm

Refer to the algorithm given in Fig. 5.2. For the trigger word detection of the string or list of strings collected from the Speech to Text part, we are following a simple brute force comparison algorithm with a predefined set of dictionaries. The comparison is done with the help of a default function in python, called intersection, which is a lambda expression. The dictionaries consist of a list of words collected from multiple sources throughout the internet based on the desired classes; for example, a list of negative words. After collecting the lists of words, we had to refine the list from irrelevant and misleading words manually. For now, we made three classifying lists of words or dictionaries, negative word list, to find in general negative expressions; threat word list, for words that dictate actions, phrases or nouns that – commonly or otherwise – indicate threat; and profane word list, to find swears, slang and profanities. These lists of words are converted to lists of arrays, separately, at the beginning of the algorithm. A single string is taken as input or from a list of strings and each word is separated and stored into an array. This list is, then, first compared with the list of negative words to find all the negative words within the string. This list of negative words is then displayed with a count of the total number of negative words detected. The same

process is done to the string word list with the threat word list and profane word list. The only difference is that, for the threat word list, after displaying the list of threat words detected, we calculate the percentage of threat words with respect to the total number of words in the string. This percentage is then displayed as the percentage of threat followed by an alert, which depends on the percentage of threat, where less than ten percent gives 'Low threat', less than twenty percent gives 'Medium threat', less than thirty percent gives 'Medium to High threat', and anything above that flags the result as 'High threat' alert message. The percentage of profanity is also displayed after the list of profane words detected within the word string list is displayed.

As mentioned before, the shift from a convenient and readily available program or API such as the Neutrino Bad Word Filter API, was due to the lack of access to language. The limitations to only profane words in the API led to the creation of a customized program suitable for the research. In a similar way, the dictionary that has been manually gathered, corrected for errors and organized, also falls short in regards to the ever-expanding database of the English language. During the implementation, we had to come to the consideration, that, despite collecting much of the commonly used words and words in phrases regarding the subject, more and more different words and especially expressions are being brought up every day as a form of communication, significantly so in the current generation of speakers. This is one factor take account as currently, all forms of phrases and emotions through words proves to be challenging to acquire more so than consider. Despite the difficulties, many common and relevant words have been recorded in our dictionary which compliments well with a simple, but effective way to filter out and give a statistical representation of the trigger word detection.

```
Step 1: Start

Step 2: Read input string

Step 3: Create list of words from input string

Step 4: Create list of negative words

Step 5: Compare list of negative words with list of
input string words

Step 6: Display number of negative words detected and
each negative word detected

Step 7: Create list of threat words

Step 8: Compare list of threat words with list of
input string words

Step 9: Display number of threat words detected and
each threat words detected

Step 10: Calculate percentage of threat

Step 11: Define alert message according to percentage
of threat

Step 11: Display percentage of threat and alert
message

Step 12: Create list of profane words

Step 13: Compare list of profane words with list of
input string words

Step 14: Display number of threat words detected and
each threat words detected

Step 15: Calculate percentage of profanity

Step 16: Display percentage of profanity
```

Fig. 5.2 Comprehensive Methodology of the Word Detection Algorithm

5.3 Emotion Recognition

5.3.1 Feature Extraction

The following list consists of the many features selected and extracted using multiple libraries, PyAudioAnalysis and LibROSA. In this case, majority of the feature's selection is done through the pyAudioAnalysis library. However, as described further below, the MFCC features from the LibROSA library proved to provide the most suitable dataset for the highest accuracy.

•Zero Crossing Rate
• Energy and Energy Entropy
• Mel Frequency Cepstral Coefficient
• Spectral Centroid and Spread
• Spectral Entropy
• Spectral Flux and Roll off

Zero Crossing Rate – In signal processing and speech recognition, ZCR or Zero Crossing Rate refers to the frequency at which the signal's sign switches, i.e. the wave crosses zero or the origin within a frame in the time domain. In the case where ZCR is high the wavelengths are shorter with a higher frequency, thus indicating that there is little or no sound being produced. On the other hand, a generally lower crossing rate produces longer waves and determines the generation of sound in the audio segment. This is an essential feature in audio processing as it serves the fundamental purpose of whether there is a sound being produced or silence.

However, for specific purposes such as segmenting one type or noise to another or especially in speech recognition, its capabilities are only limited to that. It can distinguish between noisy and silent environment but not whether the sound is a background or environment noise or from a human speech. Thus, it always relies on other features, especially the energy to obtain the end-point detection (to determine the end points of the unvoiced sound). The concept in principle works as follows:

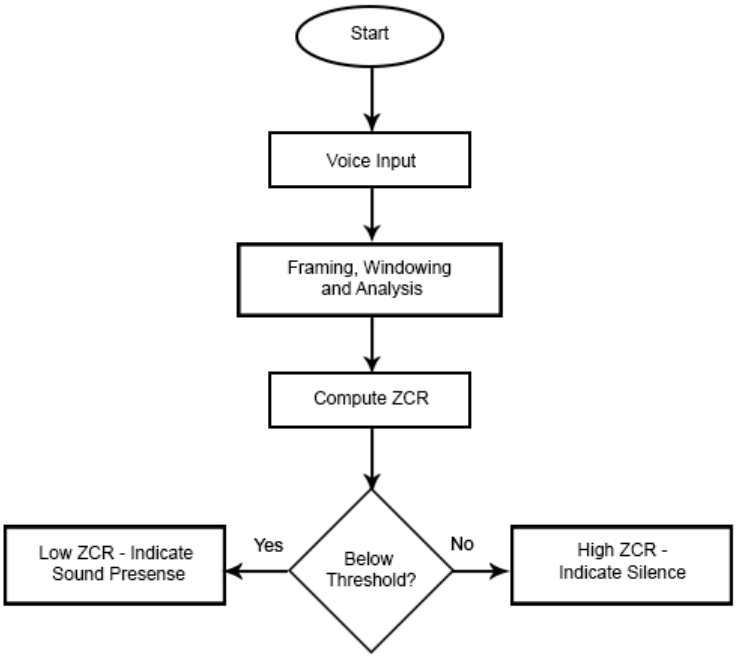

Fig. 5.3 Simplified Description of the Zero Crossing Rate

The function to define Zero Crossing Rate is:

$$ZCR = W(N\text{-}t).\ L\text{-}1Et\ |\ (St\ \text{-}St\text{-}1)\ |$$

Where W(N) is the windowing function of N samples as W = 1/2N for 0<=n<=N-1, L is the length of the signal, t is time in seconds and S(t) is the Signal at time t.

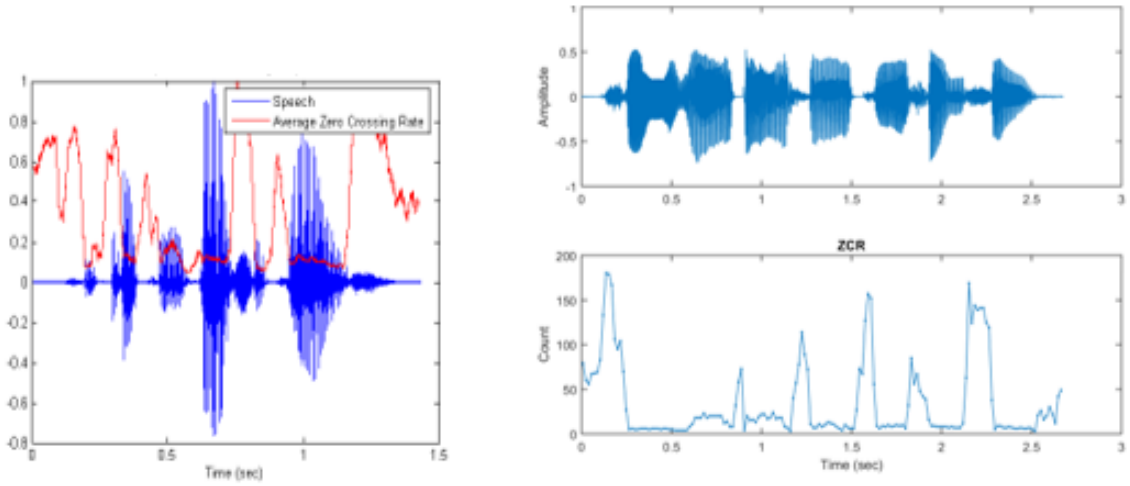

Fig. 5.4 Side-by-side Analysis of Amplitude and ZCR Frequencies Extracted from Different Audio.

Energy and Energy Entropy– Energy in signal processing simply refers to the displacement of sound with the force exerted – the strength of the signal. It generates normalized signal values and for a continuous signal, the energy can be defined as:

$$E(t) = \lim_{L\to\infty} \int_{-L}^{L} |x(t)|^2 dt = \int_{-\infty}^{\infty} |x(t)|^2 dt$$

Where L is the signal length, x(t) is the signal function over time and E is the energy.

This feature is vital as it indicates the changes in amplitude when a voiced or unvoiced segment is present during the audio processing. Therefore, as mentioned, it is used in conjunction with the zero-crossing rate in order to distinguish between voice and background or environment noise.

Energy Entropy can be defined as the quantitative dispersion of energy – from a signal in this case. A given source of sound converted to digital signals may face disorders or energy dispersal. This feature is imperative to segment and filter out noise as it can identify audio jitters and indicate loss of sound information from the signal.

MFCC – MFCC or Mel Frequency Cepstral Coefficients refers to the determining coefficients for the Mel Frequency Cepstrum. It has become one of the most widely and commonly used features in audio processing and speech recognition. This is due to its ability to identify and differentiate between logical and linguistic content between all the others influencing factors in a speech such as noise, the speaker's tone and emotions, background sound information, etc. Commonly, MFCC features are calculated in the following steps and as shown in Fig. 5.5:

1. Window or Segment the signal into smaller frames, ranging from 20-40ms per frame, usually 25ms.
2. Apply Fast Fourier Transformation on each frame and take the logarithmic values (the Quefrencies). This makes up the power spectrum needed for speech recognition
3. A process called 'Warping' is applied on the Mel frequency filter banks (Fig 5.7) which passes through 26 filters in general. This means that the frequencies are distributed linearly in the filter bank.
4. Inverse Discrete Cosine Transformation is Applied on each filter
5. Of these 26 filters, 13 MFCC values are hence produced.

Fig. 5.5 Process of MFCC Feature Extraction

Although 13 MFCC features are generally used and having extracted 13 features using PyAudioAnalysis and 20 features using the LibROSA library, we found better results using the latter.

Three fundamental concepts of this feature must be known to acknowledge the use and functionalities of MFCC. These are:

1. Mel Scale: Used to derive the Mel or the Melody frequency, this concept explains the perception of the pitch of the sound heard at distances equally apart from one another. This helps to identify the lengths of each pitch increment. The frequency can hence be converted to Mel by the following:

$$m = 2595\log10(1+f/700)$$

Where m is the mel and f the frequency. The diagram below shows the relation between changes in mel to the frequency. Mel scale is therefore needed in our research to understand the speaker's pitch, especially in cases when he/she will be stressed. Since the emotions such as anger, disgust, excitement or surprised, etc. portray their own patterns of pitch, this feature becomes ideal to extract.

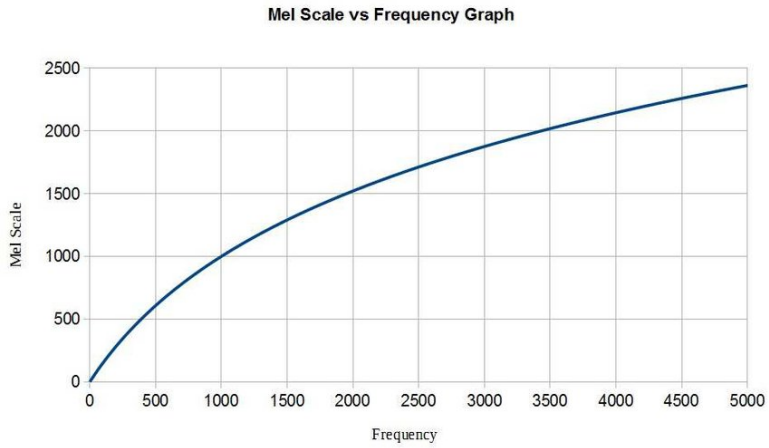

Fig. 5.6 Mel Scale to Frequency Distribution Chart

2. Cepstrum: Cepstrum is the product derived from the inverse Fourier transform of the logarithm of a spectrum. Spectrum in audio is the scale of frequency measured in Hertz (Hz) (and its discrete features extracted will be mentioned further). Cepstrum varies in multiple forms such as the real, the complex, the phase cepstrum and the power cepstrum, but for our research, the power cepstrum is to be considered since it deals with speech analysis and recognition. The concepts of cepstrum follows an anagram pattern against spectrum in terms of naming conventions – as the name itself suggests. Similarly, the measuring unit in cepstrum graphs is known as quefrency and the filtering process is called liftering. The cepstrum is then transformed via the Mel scale which results in the Mel Frequency Cepstral Coefficients which is extracted for the purpose of voice identification and pitch detection.

3. Filter Banks – As the name suggest, a filter bank contains a set of filters of various passes that helps to obtain multiple spectral features from signals through audio decomposition [27]. The operation of a bandpass filter is to filter the signal energy within a given acceptable frequency range – known as a band – and reject all others outside the range. The difference between the start and the end 'edges' of this frequency range is known as

bandwidth, and its center the center frequency. Thus, from a given signal, a filter bank allows for signals to be separated into frequency bands and each bandwidth are sequentially placed with connected edges. Filter banks are commonly used to generate audio equalizers where the low and high frequencies can be controlled.

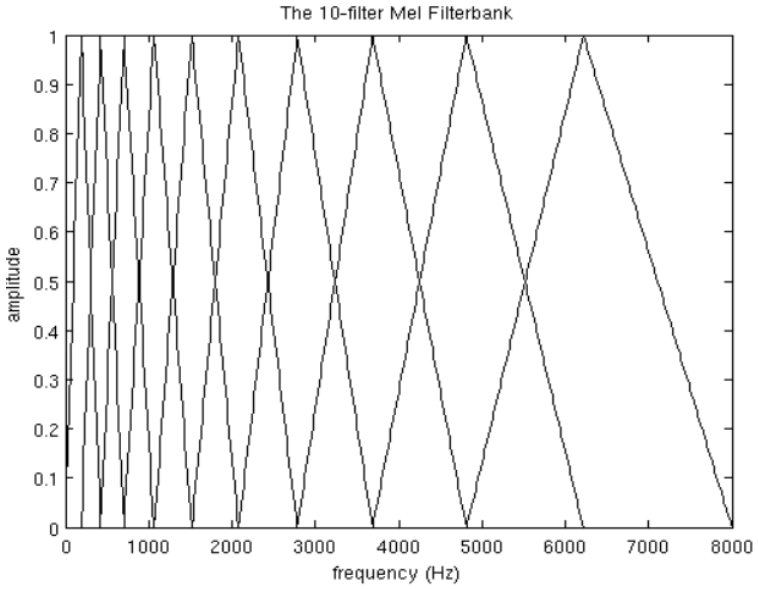

Fig. 5.7 Triangular Mel Filter Banks

There are similarities in function between the MFCC and its features to Filter Banks and the ways where extracting MFCC features could be complex and challenging and rather to consider filter banks. Primarily due to the additional steps required to calculate the Cepstral Coefficients using Discrete Cosine Transform (DCT) over the Fourier Transformation. Moreover, filter banks seem to have gathered data in the time domain (through speech signal and its perception) and proved to be more useful when correlated inputs were not a significant factor for the system to consider.

Spectrum – Spectrum in audio and signal processing refers to the scale of frequency and the amplitude. Amplitude is the measurement of changes in the magnitude of the curve in a given period of time. This is the most significant set of features needed to be extracted for any voice or audio related tasks since it deals directly with the digitally defining unit of sound (frequency in Hz). To analyze the frequencies, a spectrogram can be generated as such:

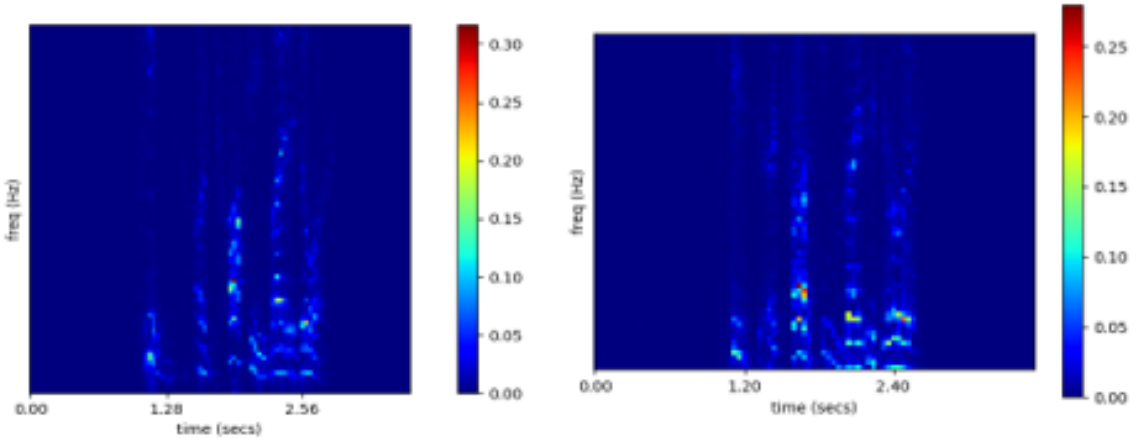

Fig. 5.8 Spectrograms generated depicting the emotions of (Left) Anger and (Right) Fearful

Spectral Centroid and Spread – Spectral Centroid is the measurement of the 'center of mass' of the spectrum and the brightness or harmonics of the sound which ranges from the upper mid to higher frequency values. This provides a weighted average and median of the amplitudes which can be derived by applying a Discrete Fourier Transform on the spectral values:

$$Spectral \quad Centroid \quad = \frac{\sum_{k=1}^{N} kF\,[k]}{\sum_{k=1}^{N} F\,[k]}$$

Where F[k] is the amplitude values and k represent bins/intervals. This study visually represents the spectral centroid and where it lies among the changing amplitude spectrum and over time:

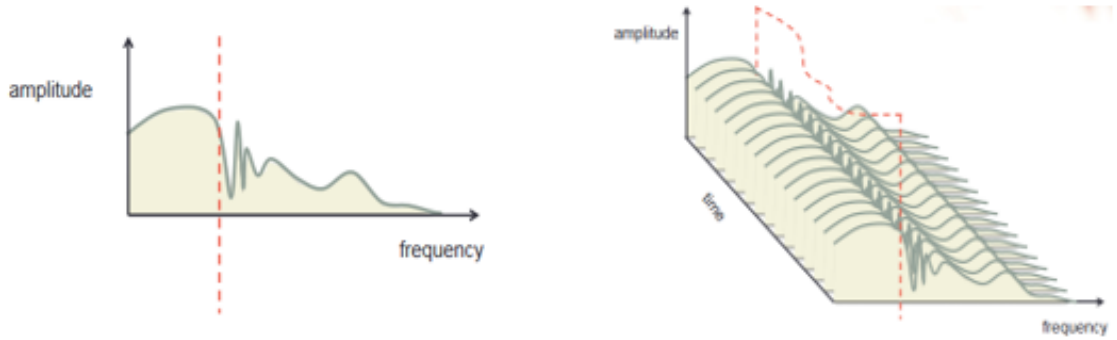

Fig. 5.9 (Left) Represents the Centroid from the signal; (Right) Centroid from the signal over time

Spectral Spread on the other hand provides the bandwidth of the signal via measuring the average peaks and changes in the Spectral Centroid. This is a common feature used in telecommunications since it creates a bandwidth greater than the frequency signal relaying the information and produces “noise-like” signals which makes it difficult to trace the original information and makes it resistant to outside noise and interference. Using the Centroid, the spectral spread can be calculated using the following:

$$Spectral\ Spread = \frac{\sum_{k=1}^{N}(k- C)^{2}F[k]}{\sum_{k=1}^{N}F[k]}$$

Where C is the Spectral Centroid, F[k] is the amplitude and k the intervals.

Spectral Entropy [23] – This refers to the peaks or the formants in the spectrum. Formants are the acoustic energy distributed in frequency intervals in the signal. There are different frequency levels of formants that determines different forms noises made. F1 formants (which is used in the creating of the classification model for our system) identifies the frequencies where vowels are uttered. Speech signals with narrow curves and higher peaks have lower entropy, while flatter ones have high entropy. This is used to identify the tonal values in the voiced portion of the signal and can be used to guess the tone of the speakers. The spectral entropy can be derived from the spectrum through the following steps:

1. Calculate spectrum S(ai) with ai amplitude

2. Calculate Power Spectrum by squaring the spectrum and averaging with number of intervals N

PS = 1/N. (S(ai))2

3. Normalize 2 to define it within a range of values (Probability Density Function)

Pi = P(ai) / i P(ai)

4. This can be used to calculate entropy based on the above: E = - ni=1 Pi ln Pi

Spectral Flux and Roll off [24]– Spectral Flux shows the variation in the power spectrum of a signal through comparison between each frame. The power spectrum is the concentration on energy in a signal in a given time frame. Spectral Roll off represents the range of frequencies where 85-95 percent of the magnitude of a signal is concentrated. According to the diagram the roll off point starts at the last few moments of the signal interval where the magnitude begins ¬dips down or "rolls off" to 0.

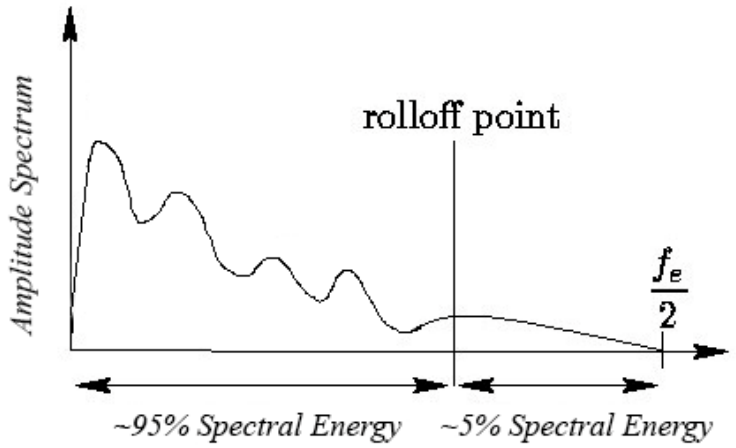

Fig. 5.10 The Spectral Rolloff Point

5.3.2 Library Implementation

PyAudioAnalysis

A majority of our study have been based around the pyAudioAnalysis library which provides more convenient solutions for audio analysis through various functionalities. Primarily, it is utilized for the feature extraction tool. It can directly take an audio file (or multiple) as input using the readAudioFile() method from the audioBasicIO.py, and then through the

stFeatureExtraction() method, it extracts the default 34 selected features from the audio and gives a CSV file the output which contains the feature vectors. The audio signal is framed using a default value of 50ms for frame or window size and 25ms for step size or a hop length, creating an overlap of 50%. Thus, for an exact 3 second audio clip, it extracts around 75 rows for each feature. This way, the entire 1440 clips from the RAVDESS database has been extracted. The output is 1440 files containing 34 short term features and another 1440 files consisting of 68 mid-term features - 34 mean and 34 standard deviation. Of the 34 features, the first 21 are selected for the implementation since features 22 to 33 are the Chroma vector, which represents pitch class of a particular genre of music and the 34th being its standard deviation (Chroma Deviation). Since these deal with music, they are not utilized.

The library also aids in easily training, testing and classifying the data by multiple classifier algorithms. For the study, the SVM and Random Forest has been chosen. After the feature selection and extraction is done, the dataset is labeled accordingly to their emotions and the featureAndTrain() method from the audioTrainTest package is used with the labeled folders of data, the window and step sizes, the required classifier and the model name as the parameters. This generates a model file in an Attribute-Relation File Format (ARFF) file. The svm rbf, svm (linear), and randomforest are used to classify the data. During the classification, the library automatically tests the data by tuning each classifier's tuning parameters and selects and highlights the best accuracy and parameters. For the SVM, the optimal parameter is the soft margin C parameter which indicates the level of restriction in the influence of the support vectors, and the number of trees N in the random forest algorithm. The results shown contains both the confusion matrix, as well as the calculation of the precision, recall and the f1 formant feature, from which an overall accuracy is drawn.

Table 5.2 showcases one of the resulting confusion matrices from training an SVM RBF classifier. This was done on the RAVDESS dataset with all the included 8 emotions, with parameters of a 50% overlap in window and step sizes. Calculating the truth values from the confusion matrix and from Table 5.3, show that the accuracy achieved is over 64% and the selected C parameter for the classifier is 20. The trained classifier can then be tested directly on any other audio clip and the model will label the clip accordingly and as close to the emotion as possible. In Table 5.4, a WAV audio file is tested on the trained model and the results show that the emotion detected was 94% calm. Finally, the library also helps to generate and visualize charts and spectrograms for each label and features; this is utilized as well.

Table 5.2 Confusion Matrix Generated from Training an SVM Classifier

	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	4	1	0	1	0	0	0	0
Calm	1	10	0	1	0	0	0	0
Happy	1	0	7	1	1	2	1	1
Sad	1	1	1	7	1	2	0	0
Angry	0	0	1	0	10	0	1	1
Fearful	0	0	1	1	0	9	1	0
Disgust	0	0	1	1	1	1	8	1
Surprised	0	0	2	0	1	0	1	9
Accuracy: 64%								

Table 5.3 Overall Accuracy, C parameter and F1 Score

Overall		
C	ACC	F1
0.001	40.4	35.6
0.01	40.8	35.9
0.5	52	48.5
1	27.2	55.5
5	63.9	63.4
10	63.3	62.8
20	*64.7*	*64.1*

Table 5.4 Emotion Recognition Test Result from an Audio Clip

Class	**Probability**
Calm	0.94
Happy	0.01
Sad	0.03
Angry	0.01
Fearful	0.01
Winner Class: CALM	

However, despite the ease and numerous audio analysis features provided in the pyAudioAnalysis Library, the main feature extraction, training and classification did not provide the best results for the study in comparison to the LibROSA library. After many changes in parameters and variations in the extracted dataset, it can be assumed that this is due to some outliers and deviations in the data values itself, as the pyAudioAnalysis handles feature extraction based on the time in milliseconds and on the other hand, LibROSA performs this using time in bits.

LibROSA

Initially we import all the important libraries needed to do the feature extraction process of each audio sample from the dataset, most notably librosa. The first part of the program finds the path to all the audio samples of a certain emotion from the dataset using the findfiles operation from the librosa.util package and stores it in a string variable. The second part of the program initializes the header for each column of each CSV files of the audio samples, in this case, from 'mfcc-1', 'mfcc-2', and so on, till 'mfcc-20'.

The third step uses the load operation from librosa.core package which loads the audio file and decodes it into a time series variable, in this case, y. It is stored in y as a one-dimensional NumPy floating list or array. The variable followed by y, in this case, sr, is used to store the sample rate of y, which is the number of samples per second of the audio being processed. As mentioned before, at load time, all the audio is resampled to 22050 Hz.

Using this variable y, in the fourth step, we extract the Mel-frequency cepstral coefficients, which is said to be the short term power spectrum of a signal which is derived from the linear cosine transformation of the log power spectrum on a non-linear Mel scale frequency, of each audio sample from the dataset by the help of the mfcc operation from the librosa.feature package. After running this operation, we are able to generate the mfcc of each audio sample in matrix form, which is a numpy.ndarray of size (nmfcc, T), where the variable T is the duration of each track in frames and the nmfcc is used to denote the number of mfccs to be generated, which is set to 20 by default. We can easily change this value by assigning a different number of not more than 40 to indicate the number of mfccs that we want. Two important parameters here are hop length, which denotes the number of samples between each frames of an audio sample, and the frame length or number of Fast Fourier transform, which is the number of frames in an analysis window or frame. By default, nfft is set to 2048 and the value of hoplength is 512 bits per sample.

In step five, we scale each vectors of the matrix array of each audio sample to unit form using the normalize operation, imported from the package of sklearn.processing, where we provide the matrix and the axis at which it would be normalized. The definition of

normalization states that it is the process of scaling each sample into unit form. It is a part of the preprocessing part of data. As stated by the definition, data normalization is important so that all data are in the same scale with each other.

Step six, is used to sort or arrange the matrix of each audio sample in such a way that the columns represents each of the features whereas the rows represents each of the audio samples of the entire set of a particular emotion in the dataset. We take the mean or average of all the segmented samples of each audio for each feature. Similar to the pyAudioAnalysis library, features such as the ZCR, spectral centroid and spectral bandwidth, LibROSA also allows these features to be selected and extracted.

5.3.3 A Comparative Study

Throughout the research, during the implementation of the tools, libraries and algorithms, many modifications have been made to each aspect of the study to achieve the optimum results. Thus, in the process, many techniques were tried, tested and altered. This section briefly enlists the adjustments and the gradual improvement made in various portions of the system, be it the dataset and features, the algorithm and parameters, or the libraries themselves.

Firstly, it is already mentioned that the study involved testing a couple of English language databases for the emotion recognition. The RAVDESS database is selected over TESS as it contains full sentences conveyed in one of eight different emotions, whereas the latter contains a specific word in said emotion. For the feature extraction and selection from the pyAudioAnalysis library, 34 short term features are generated in total for each audio clip. From this data set, a multitude of possible data subsets were trained and tested. These also include the utilization of manually normalizing and randomizing the data as well as the Principle Component Analysis (PCA) for feature reduction:

• 80 and up to 400 Added mid-term features i.e. Mean values of 34 selected features
• 80 and up to 400 Added Mean values of 21 selected features excluding Chroma Vector and Deviation
• 120 and up to 400 Added short-term features of the 21 selected features
• The entire database of 1440 audio files containing the short-term features using Linear SVM
• The entire database of 1440 audio files containing the short-term features using RBF SVM Kernel
• 1200 short term features using linear and RBF kernels and Random forest
• Normalized, randomized and added short-term 34 and 21 features

• PCA ran on 13 MFCC features for feature reduction to 4 features
• PCA ran on 21 short-term features for feature reduction to 8 features

Feature combinations used in various studies [21,22] are also considered and trained, which mostly revolves around the MFCC, ZCR and the Energy features. Unfortunately, all these combination of datasets for 8 emotions, created by pyAudioAnalysis did not provide the best results; a maximum of 64% could be achieved using the Radial Basis Kernel (19(i)). Few possibilities were observed and considered for obtaining such results: At first, the number of data used to train the classifier being too small, especially when the number of outcome or dimensions is as high as 8 (1 for each emotion). It was discovered during the implementation that higher train to test ratio gave slightly better results. However, the ratio became too high to deem the algorithm efficient (95% and above was being required to be set as the trained data). SVM provides the best results with a huge quantity of data. With high number of features and dimensions and a comparatively small amount of data to work with, the classifier is likely to predict incorrectly during the training. More so with the chances of having outliers in the data i.e. exceptional cases, where all features are uniform but the outcome is unexpected. In practical scenarios regarding emotion and moreover stress and threat recognition, this is highly likely. It has been mentioned that many things factor the behavior of a person during stress.

This called for a change in the dataset entirely. Thus, the LibROSA library is tested next that has parameters for hop lengths and window sizes in bits, which, according to assumption, is easily to compute rather than millisecond in time. Keeping complexity to data size ratio in mind, this time only the first 20 MFCC features are selected and extracted to be trained. The data is normalized and standardized as scale, and applied to the SVM, Random Forest and Naïve Bayes algorithms and the results produces are greatly improved (especially the SVM RBF kernel) and also in an acceptable 80%: 20% train to test ratio. In order to achieve an even higher score, the algorithm parameters were evaluated (aside from the base parameter of each algorithm). These additional parameters are listed as follows:

• Random State – Randomly selects and splits the train and test data sets. This ensures that no two runtime results will be the same as the rows of data selected are randomized. If the data is organized accordingly, the random state is set (as by default) to 0.
• Gamma – Gamma is an additional parameter that tunes the influence of each support vector and hence, the bias and variance in the prediction. Higher gamma would mean a greater bias to the selection of support vectors and so variance in data decreases. This is by default set to 1 divided by the number of features.

Table 5.5 Test Results from LibROSA Feature Extraction

	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	13	4	0	1	1	0	0	1
Calm	3	33	1	0	0	1	1	0
Happy	0	1	23	2	2	1	1	0
Sad	1	0	3	28	0	3	0	2
Angry	0	1	0	3	31	1	3	1
Fearful	0	0	4	9	0	31	0	1
Disgust	0	3	2	3	3	0	34	4
Surprised	1	0	3	1	1	3	0	19
Accuracy: 74%								

• Criterion – This is a decision tree and random forest specific additional parameter that controls the data split quality. It can be either Gini impurity or entropy information gain. Table 5.3 shows the resulting accuracy from the confusion matrix. The model used is the SVM RBF, with an 80:20 train test split, gamma set to 1 of 10 (0.1), C = 10 and random state =1.

Further consideration to lower the complexity of the algorithm and raise the score led to the reduction of the number of emotions to suit the needs of the research. To specifically identify potential threat cases, and according to studies regarding speech under psychological stress, the most influential emotions were chosen – Sadness or Anxiety, Anger and Fearful. Besides these, the Calm and neutral emotions have been generalized to one and the Happy emotion is also being considered. This is for the optimization of the overall program and to identify and distinguish between cases where the threatening words may be spoken, but the emotion of the conversation shows otherwise (words uttered in a mocking, joking or light-hearted way). It ensures that the result from either word or emotion detection does not lead to completely one-sided assumptions. Hence, the quantity of emotions is reduced from 8 to 5. This change led to a greater increase in accuracy scores as shown in Table 5.6. Again, the RBF kernel with C=10 is used with a random state of 0 and a gamma of 0.1 or 1 of 10. It gives an impressive 81% score.

[Figures and Tables in APPENDIX A and B, show the classifier test results on more combinations of algorithms and emotions.]

Table 5.6 Test Scores Using SVM RBF Classifier on 5 Emotions

	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	24	2	0	0	1
Happy	0	36	1	0	4
Sad	4	2	30	1	3
Angry	1	6	1	38	0
Fearful	3	2	4	2	27
Accuracy: 155/192 = 81%					

Chapter 6

Experimental Results

To reiterate, the study proposes a prototype of a system that is divided in three parts: speech to text, word recognition and emotion recognition. At the end of the research, from varieties of options available in each part, the ones with the most accuracy, convenience and efficiency has been used. For the speech to text, a readily available system such as the IBM Watson Speech to Text Service proves to be sufficient, with as high as 95% accurate predictions of words spoken. For the word recognition, a simple program is applied that uses the text from the previous part, highlights the different criteria of words listed to be as the trigger words, as well as provides a percentage of threatening or threat related words used in the conversation and flags the level of threat. Results shown in Fig 6.1.

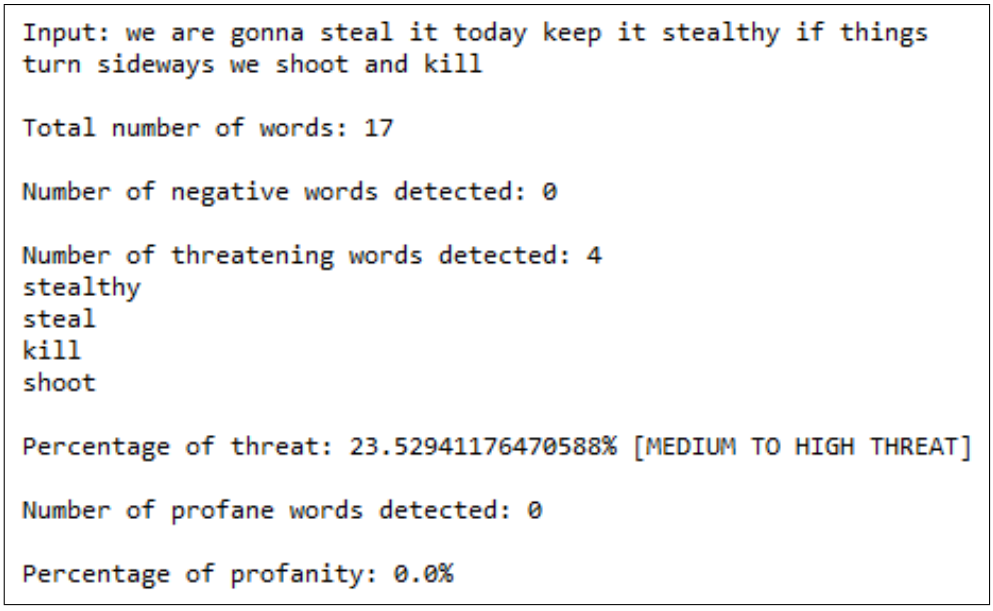

Fig. 6.1 Results Achieved from Trigger Word Detection

For the emotion recognition, after numerous changes and modifications being made and various tests of dataset combinations being conducted, we arrived at a high score of 84% accuracy in the emotion recognition. In comparison to the Random Forest, Naïve Bayes, Polynomial and Linear SVM classifiers, the Radial Basis Function (RBF) SVM performed the best. This trained model can then be used on other audio recordings directly through a library like pyAudioAnalysis to get the predicted emotion (Table 5.4), or broken down into features and processed manually through the LibROSA library. Despite the ease of use of the pyAudioAnalysis library, LibROSA provided the current high score from its feature extraction. Of a wide variety of features tested on, the 20 MFCC features extracted by LibROSA has given the best results. As such, the train-test split is a standard 80-20 and the parameters tuned for the classification are: Gamma set to 0.2 or 1 of 5 features, C =10, and random state set to 0. Therefore, of the 960 feature vectors from 5 emotions, 192 were tested and 161 were correctly predicted (Table 6.1).

Table 6.1 Highest Accuracy Obtained for the Emotion Recognition

	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	23	0	0	0	4
Happy	0	34	1	2	4
Sad	2	1	35	0	2
Angry	1	5	0	39	1
Fearful	2	1	4	1	30
Accuracy: 161/192 = 84%					

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The level of criminal offences greatly rising and risking the lives and wealth of individual nations as a whole could be cut down in great proportion if prior knowledge of such actions can be recognized in advance. The use of this system can be of assistance in concluding possible criminal activities in a shorter time period by recognizing threat possibilities. The drawback of breaching personal privacy thus cannot limit the access of this system only to the government or safety and defense sector where the risk of misuse of this information is limited to the maximum. This line of work has been deeply considered in forms of human emotions but has yet to be executed in identifying threat. The thesis here tries to find the possible threat detection and also levels of threat to give an extra edge in deciding whether the speaker has any motive in executing threatening activities.

Our work is a comparative study on multiple algorithms, from which the SVM Radial Basis Function (RBF) produces the highest accuracy of 84 percent. The database used is the RAVDESS (Ryerson Audio Vision Database for Emotion Speech and Song), and the feature selection and extraction are done using LibROSA library in python. After training and testing for the desired results, the model can then be utilized directly on audio files for emotion recognition using the PyAudioAnalysis library. These are the steps taken to propose such a system to analyze voice calls for potential threats. The studies mentioned above are just a few sources for our claim to identify stress in emotion for the emotion recognition, and along with the word recognition, classify the scenario as a potential threat.

7.2 Future Work

There has been a lot of effort put in this thesis and has resulted a great deal of study. As mentioned before the use of multiple libraries to get the features required and applying different supervised classifying algorithms. There is still a large scope of improvement in this work which includes increasing the level of accuracy using more features. The use of LPCC a common audio feature that has been used in similar form of work and also other features that has an effect in audio characteristics could be used to give better accuracy in our work which we can apply in the future, better functioning of the system.

For the time being we are only considering the use of natural languages in the detection of the emotions. There is a possibility of sarcasm in speech which is yet to consider. The use of code languages is also very much possible which can also be included in our work, that would be considered in the speech to text part where the code language can be translated into normal language. There is also the scope of being able to incorporate the use of other languages and deriving the emotions based on that. The thought of turning the work in multilingual consideration would definitely make the system useable internationally. There have been great limitations in dataset access that was able to be collected. The works that were remotely close to this thesis had mostly used the EMO-DB which consists of German speeches and in case of English speech SUSAS was used. These were beyond our access thus these audio datasets might also give us greater accuracy. The way we have detected trigger words was by brute force programming which required a dictionary. Thus, we require an updated dictionary or if there is a specific category of words, then a new form of dictionary would be needed to get matches for trigger words considered as threatening. When making the system multilingual there would also be a requirement of dictionaries of different languages. The tests were conducted based on separate sentences thus it would be necessary that speech of each speaker is being recorded and divided accordingly into sentences.

The whole program is being executed separately. Thus, it would be very much convenient if the entire program can be executed in real time with better accuracy. The recording in done using a recording app that has already been described, which can be improved in this way. There is a large scope of using more classifications algorithm which is yet to be tested on such as XGboost algorithm which is a more improved version of gradient boosting algorithm. It is also a form of ensemble learning and is well known for its performance and execution speed. The use of gradient boosting and also decision tree gives the algorithm of executing two very effective machine learning algorithm to get better results. There was also a need to reduce the number of emotions that were considered thus reduces the number of classes. Therefore, executing the same program with more emotions would give a larger scope of classifying and making it even more accurate in distinguishing threat possibilities.

References

[1] "Internet killer admits murdering women he met in online chat rooms," *The Telegraph*, 15-Jan-2009.[Online].Available:https://www.telegraph.co.uk/news/worldnews/europe/germany/4243030/Internet-killer-admits-murdering-women-he-met-in-online-chat-rooms.html. [Accessed: 13-Nov-2018].

[2] R.Sandanalakshmi, P.A. Viji, M.Kiruthiga, M.Manjari and A.Sharina, " Speaker Independent Continuous Speech to Text Converter for Mobile Application", arXiv:1307.5736 [cs], Jul. 2013.

[3] N.Sharman and S.Sardana, "A Real Time Speech to Text Convesion System Using Bidirectional Kalman Filter In MATLAB", In *the International Conference on Advances in Computng, Communications and Informatics (ICACCI),* Jaipur, pp.2353-2357, 2016.

[4] S.Sultana, M.A.H. Akhand, P.K.Das and M.M.H.Rahman, "Bangla Speech-to-Text Conversion Using SAPI", In *the International Conference on Computer and Communication Engineering (ICCCE),* Kuala Lumpur, pp.385-390, 2012.

[5] Y.Chen, Y.Zhou, S.Zhu and H.Xu, "Detecting Offensive Language in Social Media to Protect Adolescent Online Safety", In *the International Conference on Privacy, Security Risk and Trust* and *the International Conference on Social Computing,* Amsterdam and The Netherlands, pp.71-80, 2012.

[6] S.Casale, A.Russo, G.Scebba and S.Serrano, "Speech Emotion Classification Using Machine Learning Algorithms", In *the IEEE International Conference on Semantic Computing,* Santa Clara CA, pp.158-165, 2008.

[7] S.P.Whiteside, "Simulated Emotions: An Acoustic Study of Voice and Perturbation Measures", In *the International Conference on Speech and Language Processing (ICSLP),* Sydney, 1998.

[8] B.Reichardt, O.Julian, B.Zapata, K.Saurty and E.Fleißwasser,"Pitch Tracking – Comparison of Different Algorithms for Pitch Tracking".[Online]. Available: https://nats-www.informatik.uni-hamburg.de/pub/SLP16/WebHome/POSTER-pitch-tracking-poster.pdf. [Accessed: 13-Nov-2018].

[9] M.Ghai, S.Lal, S,Duggal and S.Mani, “Emotion Recognition on Speech Signals Using Machine Learning”, In *the International Conference on Big Data Analytics and Computational Intelligence (ICBDAC),* Chirala, pp.34-39, 2017.

[10] A.Glowacz and G.Altman, “Automatic Threat Classification Using Multiclass SVM from Audio Signals”, In *Proceedings of the IEEE International Conference on Emerging Technologies & Factory Automation,* Krakow, 2012.

[11] Y.Lin and G.Wei, “Speech Emotion Recognition Based on HMM and SVM”, In *the International Conference on Machine Learning and Cybernetics,* Guangzhou, pp.4898-4901, Aug. 2005.

[12] P.Shen, Z.Changjun and X.Chen, “Automatic Speech Emotion Recognition Using Support Vector Machine”, In *Proceedings of the International Conference on Electronic & Mechanical Engineering and Information Technology,* Harbin, pp. 621-625, Aug. 2011.

[13] S.R.Livingstone and F.A.Russo, ”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. e0196391. 10.1371/journal.pone.0196391, May 2018.

[14] T.Giannakopoulos, “pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis”, PLOS ONE. 10. e0144610. 10.1371/journal.pone.0144610, Dec. 2015.

[15] T.Giannakopoulos and S.Petridis, “Fisher Linear Semi-Discriminant Analysis for Speaker Diarization”, In *the IEEE Transactions on Audio, Speech, and Language Processing,* vol. 20, no.7, pp.1913-1922, Sept. 2012.

[16] H.Hollien, *Forensic Voice Identificaiton.* London: Academic Press, 2002, pp.51-57.

[17] J.H.L.Hansen and S.Patil, *Speech Under Stress: Analysis, Modeling and Recognition*, Chapter Speaker Classification I of the series Lecture Notes in Computer Science, vol. 4343, pp. 108-137, Jan. 2007.

[18] S.Besbes and Z.Lachiri, “Multi-class SVM for Stressed Speech Recognition”, In *the 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP),* Monastir, Mar. 2016.

[19] J.H.L.Hansen and S.E.Bou-Ghazale, “Getting Started with SUSAS: A Speech Under Simulated and Actual Stress Database”, In *the European Conference on Speech Communication and Technology (EUROSPEECH),* Rhodes, Sept. 1997.

[20] I.Jolliffe, “Principal Component Analysis”, *International Encyclopedia of Statistical Science*, M.Lorvic (eds), Berlin, p.22, 2011.

[21] M.S.Likitha, S.R.R.Gupta, K.Hasitha and A.U.Raju, “Speech Based Human Emotion Recognition Using MFCC”, In *the International Conference on Wireless Communication, Signal Processing and Networking (WiSPNET),* Chennai, pp.2257-2260, Mar. 2017.

[22] S.Basu, J.Chakraborty and M.Aftabuddin, “Emotion Recognition From Speech Using Convolutional Neural Network with Recurrent Neural Network Architecture”, In *the 2nd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, pp.333-336, Oct. 2017.

[23] H.Misra, S.Ikbal, H.Bourlard and H.Hermansky, “Spectral Entropy Based Feature for Robust ASR”, In *the IEEE International Conference on Acoustics, Speech and Signal Processing,* Montreal, Quebec, pp.193-196, May 2004.

[24] S.Lee, J.Kim and I.Lee, “Speech/Audio Signal Classification Using Spectral Flux Pattern Recognition”, In *the IEEE Workshop on Signal Processing Systems*, Quebec City, Quebec, pp.232-236, Oct. 2012.

[25] F.Rong, “Audio Classification Method Based on Machine Learning”, In *the International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS),* Changsha, pp.81-84, Dec. 2016.

[26] L.Grama, L.Tuns and C.Rusu, “On the Optimization of SVM Kernel Parameters for Improving Audio Classification Accuracy”, In *the 14th International Conference on Engineering of Modern Electric Systems (EMES),* Oradea, pp.224-227, Jun. 2017.

[27] A.Mertins, “Filter Banks”, In *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications,* New York: John Wiley & Sons, Inc., 1999, pp.143-195.

[28] E.Frank, M.A.Hall and I.H.Witten, “Data Mining: Practical Machine Learning Tools and Techniques (Online Appendix)”, *The WEKA Workbench,* 2016.

[29] IBM, *IBM Watson Speech to Text Service Demo,* 2015.

[30] Neutrino, *Neutrino Bad Word Filter API*, 2014.

[31] R.Tatman and S.Weinberger, *Speech Accent Archive,* 2013.

[32] B.McFee, C.Raffel, D.Liang, D.P.W.Ellis, M.McVicar, E.Battenberg and O.Nieto, “Librosa: Audio and Music Signal Analysis in Python”, In *Proceedings of the 14th Python in Science Conference,* Austin, Texas, pp.18-25, Jul. 2015.

[33] F.Pedregosa *et al.,* “Scikit-learn: Machine Learning in Python”, In *the Journal of Machine Learning Research*, vol.12, pp.2825-2830, 2011.

[34] J.D.Hunter, “Matplotlib: A 2D Graphics Environment”, In *Computing In Science & Engineering,* vol.9, no.3, pp.90-95, Jun. 2007.

[35] K.Dupuis and M.K. Pichora-Fuller, *Toronto Emotional Speech Set (TESS),* 2010.

Appendix A

Each Figure mentions the classifier and its tuned parameters for its given results. The data, as mentioned, has an 80-20 train-test split. Figures A.1 – A.5 shows the resulting accuracies in the confusion matrix, from the features extracted using the LibROSA library on 5 emotions.

Table A.1 78% Accuracy Using SVM RBF on 5 Emotions

Classifier: SVM RBF					
Params: Random State 0, Gamma 0.067, C = 10					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	24	2	0	0	1
Happy	0	35	1	0	5
Sad	7	4	26	1	2
Angry	1	6	0	39	0
Fearful	4	2	5	2	25
Accuracy:149/192 = 78%					

Table A.2 69% Accuracy Using Random Forest on 5 Emotions

Classifier: Random Forest					
Params: Random State 0, Criterion: Entropy,N = 100					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	21	1	3	0	2
Happy	1	27	3	2	8
Sad	6	4	24	1	5
Angry	1	4	2	38	1
Fearful	4	1	9	2	22
Accuracy:132/192 = 69%					

Table A.3 64% Accuracy Using SVM Poly Kernel on 5 Emotions

Classifier: SVM Poly					
Params: Random State 0, Gamma 0.2, Deg. = 5					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	24	0	1	2	0
Happy	0	28	6	5	2
Sad	9	5	19	3	4
Angry	1	9	1	35	0
Fearful	4	6	9	2	17
Accuracy:123/192 = 64%					

Table A.4 47% Accuracy Using SVM Linear Kernel on 5 Emotions

Classifier: SVM Linear					
Params: Random State 0, Gamma 0.2, C = 10					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	15	4	3	1	4
Happy	3	14	4	7	13
Sad	12	9	11	3	5
Angry	5	1	6	32	2
Fearful	7	9	3	0	19
Accuracy:91/192 = 47%					

Table A.5 42% Accuracy Using Naive Bayes on 5 Emotions

Classifier: Naïve Bayes					
Params: Random State 0					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	16	9	0	2	0
Happy	5	15	3	10	8
Sad	9	18	7	5	1
Angry	3	4	1	34	4
Fearful	9	11	8	1	9
Accuracy:81/192 = 42%					

Tables A.6 and A.7 shows the resulting accuracies in the confusion matrix, from the features extracted using the LibROSA library on 8 emotions.

Table A.6 71% Accuracy Using SVM RBF on 8 Emotions

Classifier: SVM RBF								
Params: Random State 1, Gamma Default, C = 10								
	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	11	3	1	1	1	1	0	2
Calm	4	30	1	1	0	1	1	1
Happy	1	2	21	1	3	1	1	0
Sad	2	1	4	25	1	2	0	2
Angry	0	1	0	3	31	1	2	2
Fearful	0	0	4	8	0	32	0	1
Disgust	1	0	1	2	7	0	34	4
Surprised	1	0	2	2	1	2	0	20
Accuracy: 71%								

Table A.7 71% Accuracy Using Random Forest on 8 Emotions

Classifier: SVM RBF								
Params: Random State 0, Gamma Default, C = 5								
	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	13	0	2	1	1	1	0	2
Calm	4	31	1	1	0	1	1	0
Happy	1	1	20	1	4	1	1	1
Sad	1	3	3	24	1	3	0	2
Angry	0	0	0	4	30	1	3	2
Fearful	0	1	3	9	1	30	1	0
Disgust	1	1	1	2	5	0	35	4
Surprised	0	1	2	2	1	2	0	20
Accuracy: 70%								

Tables A.8 to A.10 shows the resulting accuracies in the confusion matrix and an overall calculation of Accuracy and F1 Formants (respectively), from the features extracted using the PyAudioAnalysis library on both 8 and 5 emotions.

Table A.8 Confusion Matrix, Accuracy and F1 Score from PyAudioAnalysis on 5 Emotions

Classifier: SVM RBF					
Params: 50ms Frame, 25ms Step Size C = 5					
	Calm	**Happy**	**Sad**	**Angry**	**Fearful**
Calm	17	0	2	0	0
Happy	1	14	2	1	2
Sad	3	2	11	1	3
Angry	1	2	1	16	1
Fearful	0	2	2	1	15
Accuracy:73/100 = 73%					

Overall		
C	**ACC**	**F1**
0.001	51.3	49.8
0.01	51.5	49.8
0.5	61.5	60.7
1	66.4	66
5	*73.4*	*73.3*
10	72.9	72.8
20	73.2	73.1

Table A.9 Confusion Matrix, Accuracy and F1 Score from SVM RBF on 8 Emotions

Classifier: SVM RBF								
Params: 50ms Frame, 25ms Step Size C = 20								
	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	4	1	0	1	0	0	0	0
Calm	1	10	0	1	0	0	0	0
Happy	1	0	7	1	1	2	1	1
Sad	1	1	1	7	1	2	0	0
Angry	0	0	1	0	10	0	1	1
Fearful	0	0	1	1	0	9	1	0
Disgust	0	0	1	1	1	1	8	1
Surprised	0	0	2	0	1	0	1	9
Accuracy: 64%								

Overall		
C	**ACC**	**F1**
0.001	40.4	35.6
0.01	40.8	35.9
0.5	52	48.5
1	27.2	55.5
5	63.9	63.4
10	63.3	62.8
20	64.7	64.1

Table A.10 Confusion Matrix, Accuracy and F1 Score from Random Forest on 8 Emotions

Classifier: Random Forest								
Params: 50ms Frame, 25ms Step Size, N = 500								
	Neutral	**Calm**	**Happy**	**Sad**	**Angry**	**Fearful**	**Disgust**	**Surprised**
Neutral	2	3	0	0	0	0	1	1
Calm	0	11	0	1	0	0	1	0
Happy	0	1	5	1	1	2	1	2
Sad	0	3	1	5	0	2	1	1
Angry	0	0	1	0	9	1	2	1
Fearful	0	1	1	1	1	8	1	1
Disgust	0	1	1	1	2	1	8	1
Surprised	0	1	1	1	1	1	1	9
Accuracy = 57%								

Overall		
N	**ACC**	**F1**
10	38.6	37.6
25	46.9	45.6
50	51.2	49.6
100	54.2	52.4
200	54.5	52.6
500	57.4	55.4

Appendix B

Figures B.1 – B.5 shows (from left to right), the Zero Crossings Rate, Energy, MFCC and the Spectrograms generated for each of the 5 emotions (Calm, Happy, Sad, Angry and Fearful) respectively:

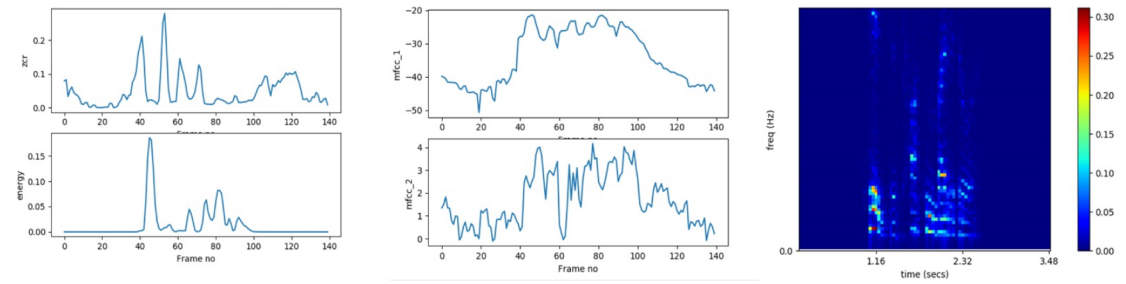

Fig. B.1 ZCR, Energy, MFCC and Spectrogram for CALM Emotion

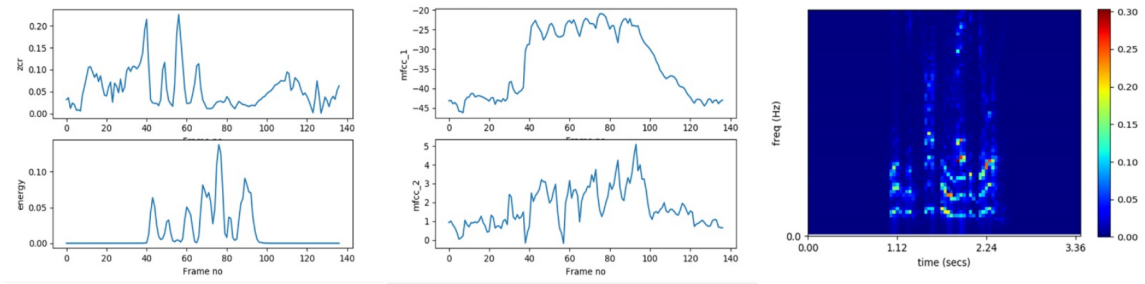

Fig. B.2 ZCR, Energy, MFCC and Spectrogram for HAPPY Emotion

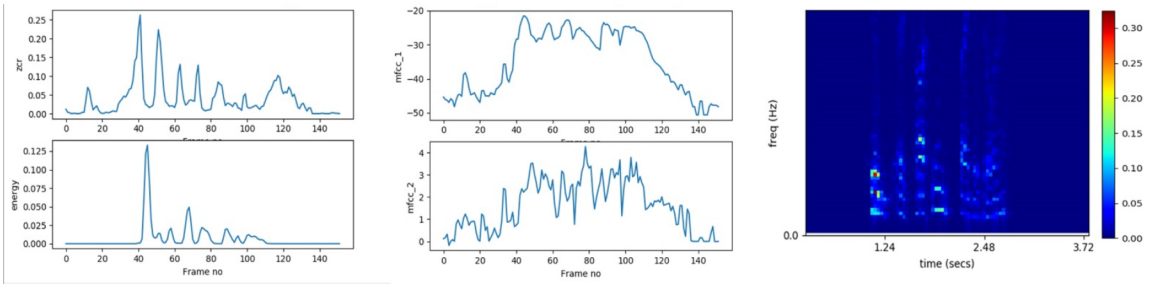

Fig. B.3 ZCR, Energy, MFCC and Spectrogram for SAD Emotion

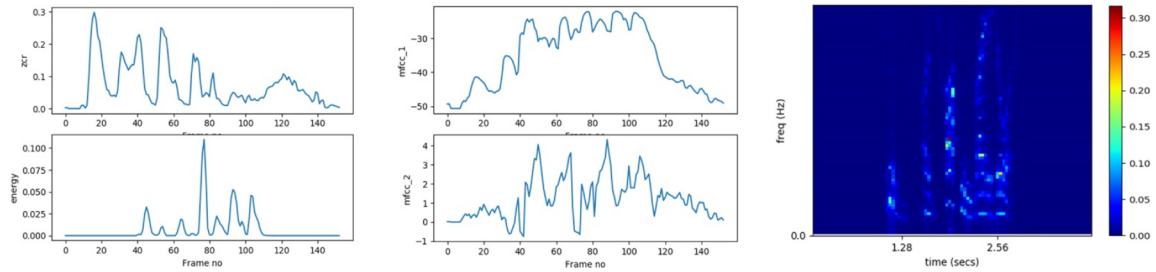

Fig. B.4 ZCR, Energy, MFCC and Spectrogram for ANGRY Emotion

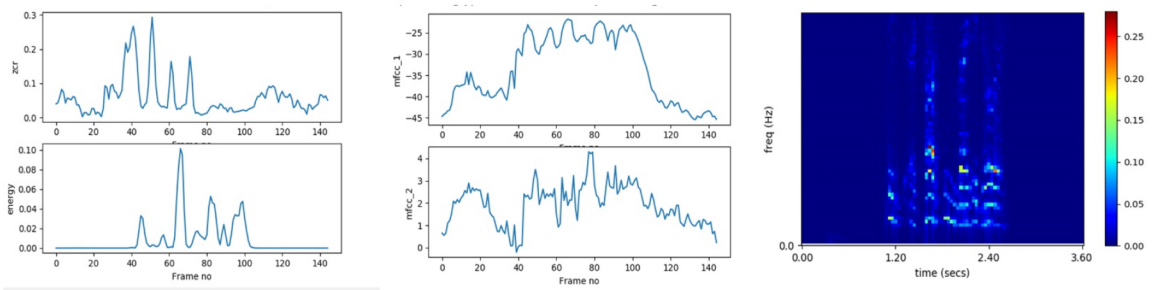

Fig. B.5 ZCR, Energy, MFCC and Spectrogram for FEARFUL Emotion