SPEECH EMOTION RECOGNITION

A project report submitted in partial fullfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Kamarthapu Nikhil Kumar	(16071A0585)
M.A Soheb Moin	(16071A0596)
Pillodi Mahitha	(16071A05A7)
Pogaku Pranav Kumar	(16071A05A8)

Under the guidance of

Mrs. Y.Bhanusree

(Assistant Professor, VNR VJIET)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institute, NAAC Accredited With 'A++' Grade, NBA

Accredited, Approved by AICTE, New Delhi, Affiliated to JNTUH)

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute)
Hyderabad – 500090

CERTIFICATE

This is to certify that **Kamarthapu Nikhil (16071 A0585), M.A Soheb Moin (16071A0596), Pillodi Mahitha (16071 A05A7), Pogaku Pranav Kumar (16071A05A8)** have successfully completed their project work at Department of CSE, VNR VJIET, Hyderabad entitled **"SPEECH EMOTION RECOGNITION"** in partial fulfilment of the requirements for the award of B.Tech degree during the academic year 2019-20

Mrs. Y.Bhanusree
Assistant Professor &Internal Guide
Department of Computer Science

VNR VJIET

Mrs. B.V. Kiranmayee
Professor & HOD
Department of Computer Science
VNR VJIET

DECLARATION

We hereby declare that the project entitled **"SPEECH EMOTION RECOGNITION"** submitted in partial fulfilment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering at **VNR Vignana Jyothi Institute of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University, Hyderabad, is a bonafide report of the work carried out by us under the guidance and supervision of Mrs. Y.Bhanusree (Assistant Professor), Department of CSE, VNRVJIET. To the best of our knowledge, this report has not been submitted in any form to any University/Institute for award of any degree or diploma.

Kamarthapu Nikhil Kumar	**M.A Soheb Moin**	**Pillodi Mahitha**	**Pogaku Pranav Kumar**
(16071A0585)	**(**16071A0596)	(16071A05A7)	(16071A05A8)
IV B. Tech CSE	IV B. Tech CSE	IV B. Tech CSE	IV B. Tech CSE
VNRVJIET	VNRVJIET	VNRVJIET	VNRVJIET

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it,without it would ever never have come into existence. To them we lay the words of gratitude imprinting within us.

We are indebted to our venerable principal **Dr. C. D. Naidu** for this unflinching devotion, which lead us to complete this project. The support, encouragement given by him and his motivation lead us to complete this project. We are very much thankful to our H.O.D., **Mrs. B.V. Kiranmayee** madam for extending her cooperation in doing this project.

We extend our heartfelt thanks to our guide, **Mrs. Y.Bhanusree** madam for her enthusiastic guidance throughout the course of our project. We also express our sincere thanks to our Mini project co-ordinators **Mrs. P. Radhika and Mrs. L. Jai Vinita** who extended their valuable support in helping us complete the project in a correct way.

Last but not the least, our appreciable obligation also goes to all staff members of Computer Science & Engineering Department and to our fellow classmates who directly or indirectly helped us.

Kamarthapu Nikhil Kumar (16071A0585)
M.A Soheb Moin (16071A0596)
Pillodi Mahitha (16071A05A7)
Pogaku Pranav Kumar (16071A05A8)

ABSTRACT

The interaction between human beings and computers would be more natural if computers are able to perceive and respond to human non-verbal communication such as emotions.

Although several approaches have been proposed to recognize human emotions based on facial expressions or speech, relatively limited work has been done to fuse these two, and other, modalities to improve the accuracy and robustness of the emotion recognition system.

In this project we are developing an emotion recognition model using Deep Networks. Speech being a sequential data model, Recurrent Neural Network can be used to process emotion of a human being. There are a variety of temporal and spectral features that can be extracted from human speech. The features are chosen to represent intended information. We have used the most used features that are available in librosa library including MFCC, Chromagram, MEL Spectrogram Frequency (mel).

Our Application recognizes three emotions like sad, neutral and happy. Speech emotion data bases like Berlin database of Emotional speech, Ravdess Emotional speech database can be used for training and testing the data.Speech Emotion Recognition (SER) using Machine learning has its own limitations like, Language, accent dependent. Deep networks may overcome this limitation.

INDEX

1.INTRODUCTION

1.1 Universal Emotions

Human beings are eminently emotional, as their social interaction is based on the ability to communicate their emotion and perceive the emotional status of others. The main objective of emotion recognition from speech is to cap and process information with the aim of enhancing the communication between them and computer. As we know speech is a complex signal containing information about message, speaker, language, emotion and so on. Most of the existing speak systems such as speech recognition, speaker recognition and speech synthesis process studio recorded read speech effectively; however, their performance is poor in the case of natural speech because of difficulty in modelling and characterizing emotions present in natural speech Emotions makes the speech more realistic and natural. Humans extensively use emotions to express their intentions through speech.

Humans understand other persons intention by pursuing emotions embedded in the speech. Therefore there is a need to develop system that can analyze emotions along with the message. The major goals of emotional speech processing are recognizing the emotions in given speech and evaluating the desired emotions in speech according to the intended message. Synthesis of emotions can be viewed as incorporating the emotion-specific knowledge during speech synthesis. Emotion specific knowledge is required from the emotion models, designed for capturing the emotion-specific characteristics.

Emotion: Distribution of emotions on a 2-D emotional plane

Emotions are analyzed at various levels like psychological physiological and speech perspectives. Psychology deals with consciousness of a person physiology deals with bodily sensations of a person and speech perspectives deals with behaviour of a person. There are more than 300 crisply identified emotions by researchers. Although all of these emotions are not experienced in daily life, most researchers agree on the principles of Palette theory that quotes, any emotion is the composition of primary emotions as any

color is the combination of primary colors. The basic emotions are anger, disgust, fear, happiness, sadness and surprise. These are also referred as archetypal emotions. In psychology expression of emotions is considered as response to stimuli that involves characteristic physiological changes. The properties of emotions can be illustrated in a three-dimensional space. The dimensions of emotional state are the features of activation(arousal measured as an intensity), affect(valence or pleasure measured as positive or negative feeling after emotion perception) and power (control measured as dominance or submissiveness in emotion expression).

As per physiology,nervous system is stimulated by the expression of high arousal emotions like fear anger and happiness. This phenomenon causes an increased heart beat rate,higher blood pressure, changes in respiration pattern, greater subglottal air pressure in the lungs and dryness of the mouth. The resulting speech is correspondingly louder, faster and characterized with strong high-frequency energy.higher average pitch, and wider pitch range.

On the other hand, for the low arousal emotions like sadness, the nervous system is stimulated causing the decrease in heart beat rate,blood pressure, leading to increased salivation, slow and low-pitched speech with little high-frequency energy. Thus acoustic features such as pitch, energy, timing, voice quality and articulation of the speech signal highly correlate with the underlying emotions. Distinguishing emotions without any ambiguity, using any one of the above mentioned dimensions is a difficult task. For example both anger and happiness have high activation but the convey different effect(valence or pleasure information). This difference is characterized by both activation dimensions. Therefore emotion can be defined as the complex psycho-physiological experience of an individual's state of mind as interacting with biochemical (internal) and environmental (external) influences.

In humans, emotion fundamentally involves physiological arousal, expressive behaviors, and conscious experience. Therefore the three aspects of emotions are

psychological(what one is thinking), physiological(what one's body is doing), and expressive (how one reacts) in nature.

EMOTION: Speech Signal

Emotion state of human beings is recognized with one of the feature called emotion. A speech signal is produced from the contribution of the vocal tract system excited by excitation source signal. Hence.speech specific information may be extracted from vocal tract system and excitation source characteristics. The emotion-specific characteristics of the speech can be attributed to

- Characteristics of the excitation source.
- The sequence of the shapes of the vocal tract system while producing different emotions.
- Supra-segmental characteristics(duration pitch and energy) and
- Linguistic information.

During speech production, a vocal tract as a resonates and emphasizes certain frequency components depending on the shape of the oral cavity. Formants are the resonances of the vocal tract system at a given instance of time. These formants are represented by bandwidth and amplitude, and these parameters are unique to each of the sound units. These features are clearly observed in the frequency domain. For example,speech signal is segmented into frames of size 20-30 ms with a frame shift of 10ms.

1.2 Applications:

In human-computer interaction emotion recognition in human speech is very important as it is primary communication tool of humans.

- **Monitoring the driver of a vehicle**

SER can be used to detect the emotion of a driver of a vehicle and he could be cautioned to drive safely or at least certain action can be performed to make his emotion state better so that accidents can be avoided.

- **In Medical field**

SER is used to identify the clients emotional state and can be used by the doctor to guide the patient especially during counselling.

- **E-Learning**

During online tutoring, based on the responses of the students, SER can help us identify the emotions of the students so that the learning can be made more interesting for student and easier for the tutor.

- **Law**

SER can help Lawyers to validate if their clients are speaking truth and so can work accordingly.

- **Call Centre**

In Call centre conversation SER may be used to analyze behavioral study of call attendants with the customers which helps to improve quality of service of a call attendant

1.3 Benefits:

Currently a lot of research is going on in human-computer interaction using voice. Many applications are developed based on this. When SER is added to the existing features then the probability of machine detecting the correct word is very high. Since, in languages we use words with sound similar but have a different meaning and these can be identified based on the context of the emotion.

There are many fields in which the original emotion of the speaker needs to be identified in order to make the correct decision. Generally, humans tend to show a different emotion than what they really feel. But based on some parameters of speech we can identify the emotion correctly. For example, lawyers, counsellors, doctors..etc.

1.4 Features:

Mel Frequency Cepstral Coefficients (MFCC)

The first step in an automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguist content and discussing all the other stuff which carries Information like background noise, emotion etc.

The main point to understand about speech is that the sounds generated by a woman are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out If we can determine the shape accurately, this should give is an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum and the job of MFCC is to accurately represent this envelope. This page will provide a short tutorial on MFCC.

Mel Frequency Cepstral Coefficients (MFCC) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Prior to the introduction of MFCCS Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCC) and were the main feature type for automatic speech recognition (ASR), especially with HMM classifier.This page will go over the main aspects of MFCCs, why they make good feature for ASR, and how to implement them.

Steps

- Frame the signal into short frames.
- For each frame calculate the periodogram estimate of the power spectrum.
- Apply the Mel filterbank to the power spectra, sum the energy in each filter.
- Take the logarithm of all filterbank energies.
- Take the DCT of the log filterbank energies.
- Keep DCT coefficients 2-13, discard the rest.

There are a few more things commonly done, sometimes the frame energy is appended to each feature vector. Delta and Delta-Delta features are usually also appended. Liftering is also commonly applied to the final features.

An audio signal is constantly changing, so to simplify things we assume that a short time scales the audio signal doesn't change much (when we say it doesn't change, we mean statistically Le statistically stationary, obviously the samples are constantly changing on even short time scales). This is why we frame the signal into 20-40ms frames. If the frame is much shorter we don’t have enough samples to pet a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

The periodogram spectral estimate still contains a lot of information not required for Automatic Speech Recognition (ASR). In particular the cochlea cannot discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filter bank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations We are only interested in roughly how much energy occurs at each spot. The Mel scale tells us exactly how to space your filter banks and how wide to make them.

Once we have the filterbank energies we take the logarithm of them. This is also motivated by human hearing: we don't hear loudness of a linear scale. Generally to double the perceived volume of sound we need to put 8 times in mach energy into it, This means that large variations in energy may it sound all that different if the sound is loud to begin with. This compression operation makes our features match more closely what humans actually hear. Why the logarithm and not a cube root? The logarithm allows us to use cepstral mean subtraction, which is a channel normalisation technique.

The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in HMM classifier. But notice that only 12 of the 26 DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade ASR performance, so we get a small improvement by dropping them.

Mel scale

The Mel scale relates perceived frequency, or pitch of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely with humans hear.

The formula for converting from frequency to Mel scale is:

$$M(f) = 1125 \ln(1 + f/700) \qquad (1)$$

To go from Mels back to frequency:

$$M^{-1}(m) = 700(\exp(m/1126) - 1) \qquad (2)$$

1.5. Emotional Speech Corpora:

Either for Synthesis or recognition of emotions, we need emotional speech database. A single speaker is not sufficient for recognizing emotions, we need multiple speakers to identity emotions and various styles of expressing the emotions. There are three different methods to collect emotional speech databases. They are Simulated databases, Elicited databases, Naturalistic Databases.

Simulated databases are one of major resource of database where actors are asked to portray the given emotion. Elicited database are recorded by involving people in particular situation for extracting particular emotion. Naturalistic databases are recorded from our daily life activities.

In Simulated databases, artists are given with recording sessions at different timings for variations in expressiveness of emotions in the speech and physical speech production mechanism of human beings. As 60% of the databases collected, it is one of the easier and reliable methods of collecting expressive speech databases containing wide range of emotions. These are typically intense, as they are enacted. Generally, it is found that acted/simulated emotions tend to be more expressive than real ones, these are also defined as full blown emotions.

In Elicited speech, all collected by involving the speaker in emotional conversation with other person, creating an artificial emotional situations and recording without knowledge of the speaker. Here, different contextual situations are created by person through the conversation to elicit different emotions from the subject, without his/her knowledge, These databases may be more natural compared to their simulated databases, but subjects may not be properly expressive, if they know that they are being recorded. Sometimes these databases are recorded by asking the subjects to involve in verbal interaction with computer whose speech responses are in turn controlled by the human being without the knowledge of the subjects.

In Natural databases,emotions are mildly expressed. Mostly, we take call center conversations, cockpit recordings during abnormal conditions, a dialog between patient and a doctor, emotional conversations in public places and so on. As they are not identified properly. these are termed as underlying emotions. Here, emotion recognition is a difficult task as they concentrate on subject. We find only specific set of emotions as variations of emotions cannot be collected in our daily life activities. We are having privacy issues and copyright issues to take data.

While considering database, there are also other factors to be considered. They are:

- size of database
- speaker
- gender
- Language, all these play major role in scalability, reliability and generalizability of speech systems.

2.EXISTING AND PROPOSED SYSTEM

2.1 Existing System:

Some facial recognition algorithms identify faces by extracting landmarks or features from an image. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face detection. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

2.1.1. Facial Recognition methods

Recognition algorithms can be divided into two main approaches: i) geometric which looks at distinguishing features (feature based) and ii) photometric which is a statistical approach that distill an image into values and compares the values with templates to eliminate variances (viewbased). Popular recognition algorithms Principal Component Analysis and Linear Discriminate Analysis are based on geometric approach. Elastic Bunch Graph Matching and the Hidden Markov model are based on statistical approach.

Principal Component Analysis (PCA)

PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible and each succeeding component accounts for as much of the remaining variability as possible.

Linear Discriminate Analysis (LDA).

LDA is a method to find a linear combination of features which characterize or separate two or more classes of objects or events. The resulting combination may be used as a linear classifier. In computerized face recognition, each face is represented by a large

number of pixel values. Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values which form a template.

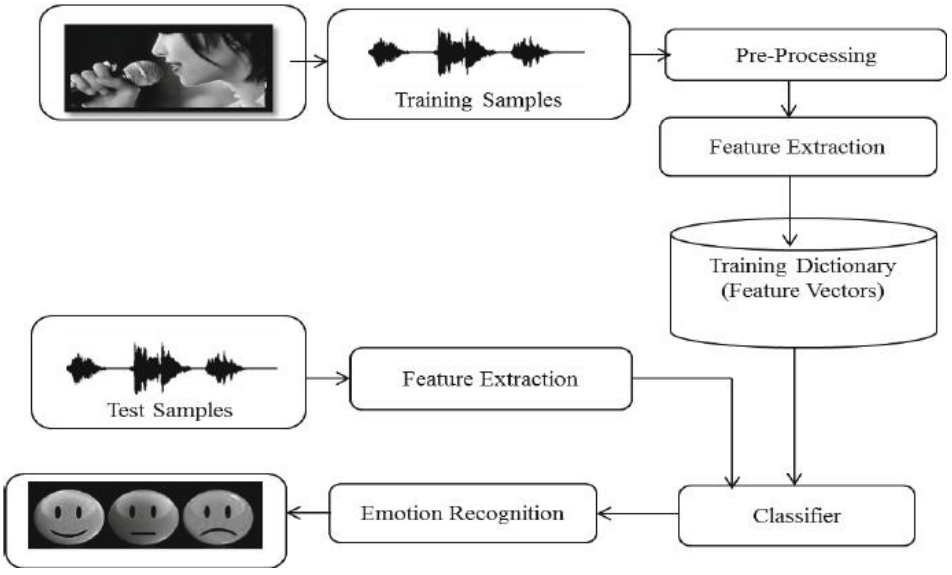
Elastic bunch graph mapping

Elastic Matching (EM) is one of the pattern recognition techniques in computer science. It is also known as deformable template, flexible matching, or nonlinear template matching

2.2 Proposed System:

The objective of the proposed system is to detect the emotion of a person by analysing various features of his speech. Once the features are detected then it will determine the emotion of the person from the pool of trained emotion set. The emotion of the person will be measured in terms of frequency and pitch of the voice and also focusing on the text spoken The two prominent measures of detecting the emotion are frequency and pitch.

The first criterion is to create a .mov file from the recording voice and feed it to the RNN network .All the outputs from these the network will be computed and compared to accurately judge the emotion of the person. The following shows the proposed system architecture and its various stages.



Speech recognition databases like berlin for emotion, speech and ravdees database have been used for training and testing.

We used a three layered RNN model to perform emotion recognition through speech by recording and feeding the .mov files to the model.

Proposed system has been trained on sequence of accoustic features calculated over small speech intervals

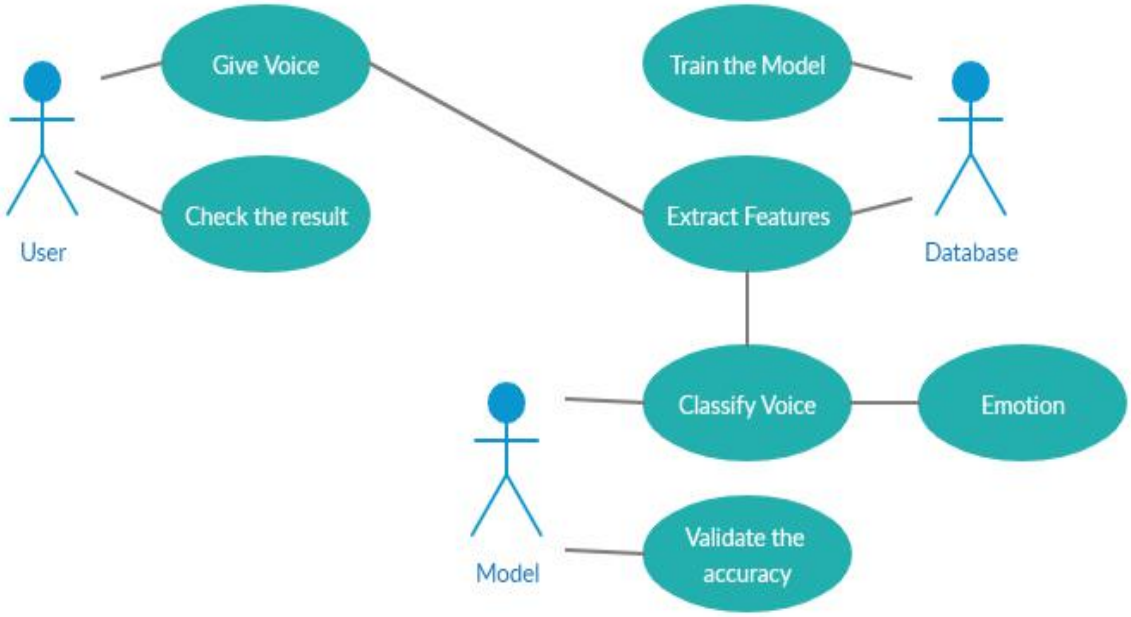
3. SOFTWARE DESIGN

3.1 Introduction

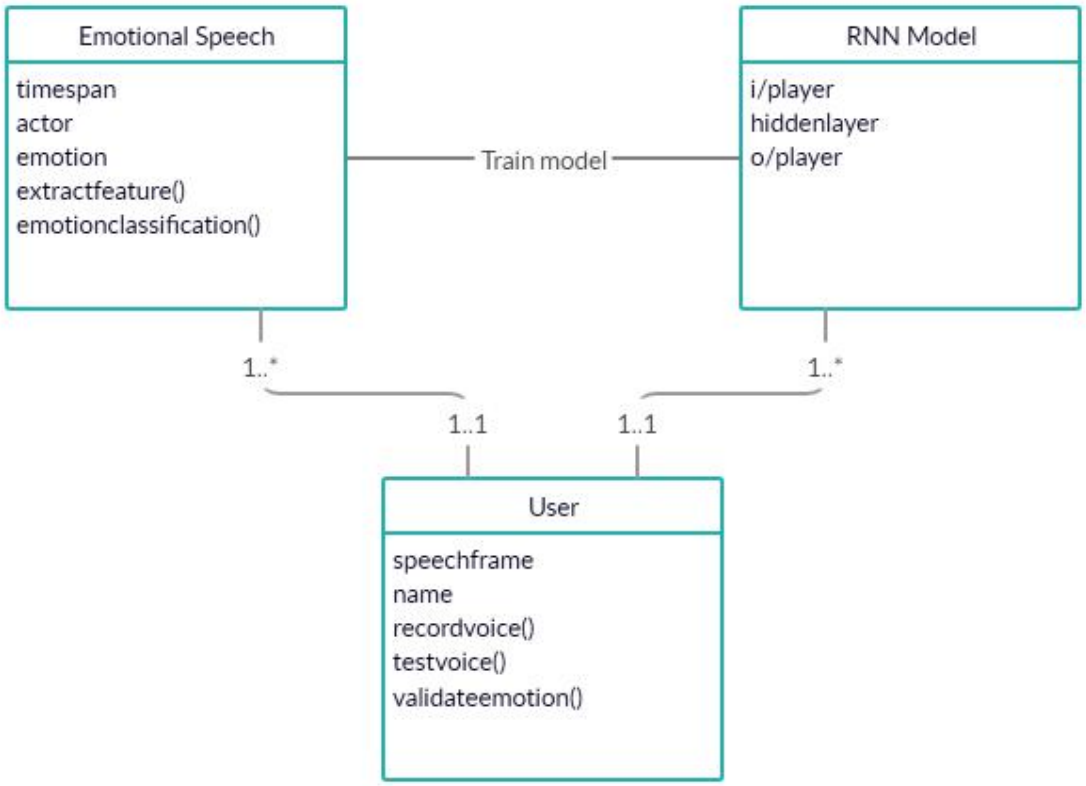
System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development logical design reviews the present physical system prepare input and output specification details of implementation plan and prepare a logical design walk through. The database tables are designed by analyzing functions involved in the system format of the fields is also designed. The fields in the Database tables should define their role in the system .The unnecessary fields should be avoided because I affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

3.2 UML Diagrams:

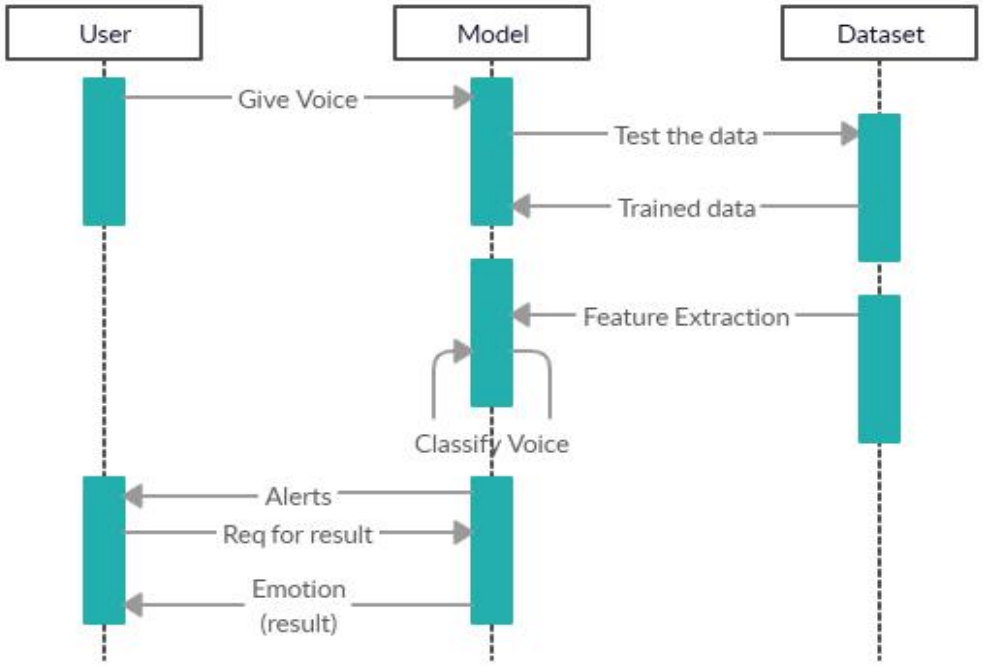
Use Case Diagram



Class Diagram

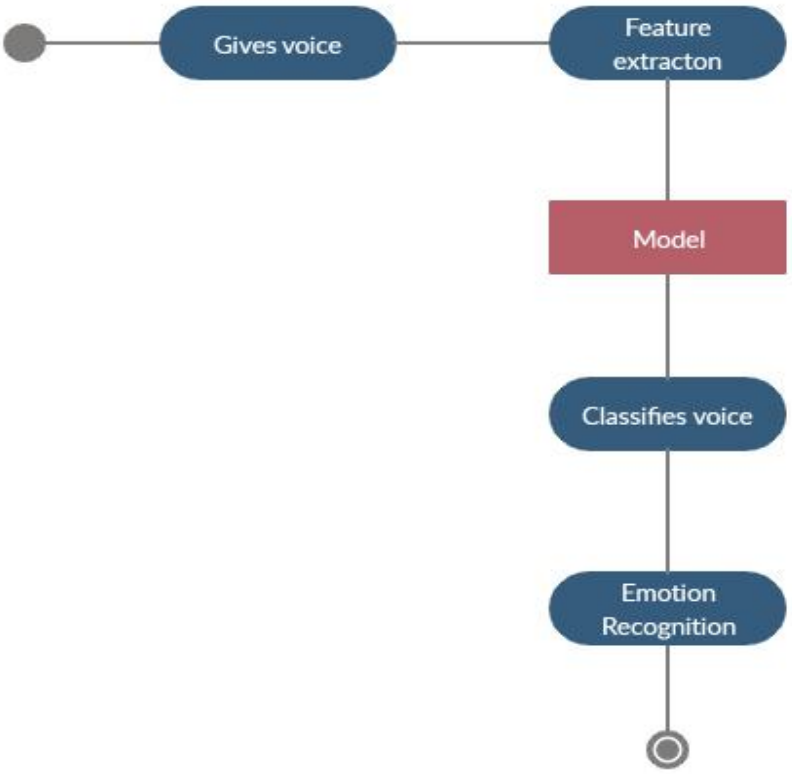


Sequence Diagram



X X X

Activity Diagram



Chapter 4: IMPLEMENTATION

4.1 Source codes and outputs

- **deep_emotion_recognition.py**

```
import os
# disable keras loggings
import sys

stderr = sys.stderr
sys.stderr = open(os.devnull, 'w')
import keras

sys.stderr = stderr
# to use CPU uncomment below code
os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"  # see issue #152
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
# disable tensorflow logs
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf

config = tf.ConfigProto(intra_op_parallelism_threads=5,
                        inter_op_parallelism_threads=5,
                        allow_soft_placement=True,
                        device_count={'CPU': 1,
                                      'GPU': 0}
                        )
from keras.layers import LSTM, GRU, Dense, Activation, LeakyReLU, Dropout
from keras.layers import Conv1D, MaxPool1D, GlobalAveragePooling1D
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, TensorBoard
```

```
from keras.utils import to_categorical

from sklearn.metrics import accuracy_score, mean_absolute_error, confusion_matrix

from data_extractor import load_data
from create_csv import write_emodb_csv, write_tess_ravdess_csv
from emotion_recognition import EmotionRecognizer
from utils import get_first_letters, AVAILABLE_EMOTIONS, extract_feature,
get_dropout_str

import numpy as np
import pandas as pd
import random


class DeepEmotionRecognizer(EmotionRecognizer):
    """
    The Deep Learning version of the Emotion Recognizer.
    This class uses RNN (LSTM, GRU, etc.) and Dense layers.
    #TODO add CNNs
    """

    def __init__(self, **kwargs):
        """
        params:
            emotions (list): list of emotions to be used. Note that these emotions must be
available in
                RAVDESS_TESS & EMODB Datasets, available nine emotions are the
following:
                    'neutral', 'calm', 'happy', 'sad', 'angry', 'fear', 'disgust', 'ps' ( pleasant
surprised ), 'boredom'.
```

Default is ["sad", "neutral", "happy"].

tess_ravdess (bool): whether to use TESS & RAVDESS Speech datasets, default is True.

emodb (bool): whether to use EMO-DB Speech dataset, default is True.

custom_db (bool): whether to use custom Speech dataset that is located in `data/train-custom`

and `data/test-custom`, default is True.

tess_ravdess_name (str): the name of the output CSV file for TESS&RAVDESS dataset, default is "tess_ravdess.csv".

emodb_name (str): the name of the output CSV file for EMO-DB dataset, default is "emodb.csv".

custom_db_name (str): the name of the output CSV file for the custom dataset, default is "custom.csv".

features (list): list of speech features to use, default is ["mfcc", "chroma", "mel"]

(i.e MFCC, Chroma and MEL spectrogram).

classification (bool): whether to use classification or regression, default is True.

balance (bool): whether to balance the dataset (both training and testing), default is True.

verbose (bool/int): whether to print messages on certain tasks.

==========

Model params

n_rnn_layers (int): number of RNN layers, default is 2.

cell (keras.layers.RNN instance): RNN cell used to train the model, default is LSTM.

rnn_units (int): number of units of `cell`, default is 128.

n_dense_layers (int): number of Dense layers, default is 2.

dense_units (int): number of units of the Dense layers, default is 128.

dropout (list/float): dropout rate,

- if list, it indicates the dropout rate of each layer.

```
        - if float, it indicates the dropout rate for all layers.
        Default is 0.3.

================================================================
=====
        Training params
        batch_size (int): number of samples per gradient update, default is 64.
        epochs (int): number of epochs, default is 1000.
        optimizer (str/keras.optimizers.Optimizer instance): optimizer used to train,
default is "adam".
        loss (str/callback from keras.losses): loss function that is used to minimize during
training,
            default is "categorical_crossentropy" for classification and
"mean_squared_error" for
            regression.
        """
        # init EmotionRecognizer
        super().__init__(None, **kwargs)

        self.n_rnn_layers = kwargs.get("n_rnn_layers", 2)
        self.n_dense_layers = kwargs.get("n_dense_layers", 2)
        self.rnn_units = kwargs.get("rnn_units", 128)
        self.dense_units = kwargs.get("dense_units", 128)
        self.cell = kwargs.get("cell", LSTM)

        # list of dropouts of each layer
        # must be len(dropouts) = n_rnn_layers + n_dense_layers
        self.dropout = kwargs.get("dropout", 0.3)
        self.dropout = self.dropout if isinstance(self.dropout, list) else [self.dropout] * (
                self.n_rnn_layers + self.n_dense_layers)
        # number of classes ( emotions )
```

```
        self.output_dim = len(self.emotions)

        # optimization attributes
        self.optimizer = kwargs.get("optimizer", "adam")
        self.loss = kwargs.get("loss", "categorical_crossentropy")

        # training attributes
        self.batch_size = kwargs.get("batch_size", 64)
        self.epochs = kwargs.get("epochs", 1000)

        # the name of the model
        self.model_name = ""
        self._update_model_name()

        # init the model
        self.model = None

        # compute the input length
        self._compute_input_length()

        # boolean attributes
        self.model_created = False

    def _update_model_name(self):
        """
        Generates a unique model name based on parameters passed and put it on
`self.model_name`.
        This is used when saving the model.
        """
        # get first letters of emotions, for instance:
        # ["sad", "neutral", "happy"] => 'HNS' (sorted alphabetically)
```

```
        emotions_str = get_first_letters(self.emotions)
        # 'c' for classification & 'r' for regression
        problem_type = 'c' if self.classification else 'r'
        dropout_str = get_dropout_str(self.dropout, n_layers=self.n_dense_layers +
self.n_rnn_layers)
        self.model_name = f"{emotions_str}-{problem_type}-{self.cell.__name__}-layers-
{self.n_rnn_layers}-{self.n_dense_layers}-units-{self.rnn_units}-{self.dense_units}-
dropout-{dropout_str}.h5"

    def _get_model_filename(self):
        """Returns the relative path of this model name"""
        return f"results/{self.model_name}"

    def _model_exists(self):
        """
        Checks if model already exists in disk, returns the filename,
        and returns `None` otherwise.
        """
        filename = self._get_model_filename()
        return filename if os.path.isfile(filename) else None

    def _compute_input_length(self):
        """
        Calculates the input shape to be able to construct the model.
        """
        if not self.data_loaded:
            self.load_data()
        self.input_length = self.X_train[0].shape[1]

    def _verify_emotions(self):
        super()._verify_emotions()
```

```
        self.int2emotions = {i: e for i, e in enumerate(self.emotions)}
        self.emotions2int = {v: k for k, v in self.int2emotions.items()}

    def create_model(self):
        """
        Constructs the neural network based on parameters passed.
        """
        if self.model_created:
            # model already created, why call twice
            return

        if not self.data_loaded:
            # if data isn't loaded yet, load it
            self.load_data()

        model = Sequential()

        # rnn layers
        for i in range(self.n_rnn_layers):
            if i == 0:
                # first layer
                model.add(self.cell(self.rnn_units, return_sequences=True,
input_shape=(None, self.input_length)))
                model.add(Dropout(self.dropout[i]))
            else:
                # middle layers
                model.add(self.cell(self.rnn_units, return_sequences=True))
                model.add(Dropout(self.dropout[i]))

        if self.n_rnn_layers == 0:
            i = 0
```

```
        # dense layers
        for j in range(self.n_dense_layers):
            # if n_rnn_layers = 0, only dense
            if self.n_rnn_layers == 0 and j == 0:
                model.add(Dense(self.dense_units, input_shape=(None, self.input_length)))
                model.add(Dropout(self.dropout[i + j]))
            else:
                model.add(Dense(self.dense_units))
                model.add(Dropout(self.dropout[i + j]))

        if self.classification:
            model.add(Dense(self.output_dim, activation="softmax"))
            model.compile(loss=self.loss, metrics=["accuracy"], optimizer=self.optimizer)
        else:
            model.add(Dense(1, activation="linear"))
            model.compile(loss="mean_squared_error",
metrics=["mean_absolute_error"], optimizer=self.optimizer)

        self.model = model
        self.model_created = True
        if self.verbose > 0:
            print("[+] Model created")

    def load_data(self):
        """
        Loads and extracts features from the audio files for the db's specified.
        And then reshapes the data.
        """
        super().load_data()
        # reshape X's to 3 dims
```

```
        X_train_shape = self.X_train.shape
        X_test_shape = self.X_test.shape
        self.X_train = self.X_train.reshape((1, X_train_shape[0], X_train_shape[1]))
        self.X_test = self.X_test.reshape((1, X_test_shape[0], X_test_shape[1]))

        if self.classification:
            # one-hot encode when its classification
            self.y_train = to_categorical([self.emotions2int[str(e)] for e in self.y_train])
            self.y_test = to_categorical([self.emotions2int[str(e)] for e in self.y_test])

        # reshape labels
        y_train_shape = self.y_train.shape
        y_test_shape = self.y_test.shape
        if self.classification:
            self.y_train = self.y_train.reshape((1, y_train_shape[0], y_train_shape[1]))
            self.y_test = self.y_test.reshape((1, y_test_shape[0], y_test_shape[1]))
        else:
            self.y_train = self.y_train.reshape((1, y_train_shape[0], 1))
            self.y_test = self.y_test.reshape((1, y_test_shape[0], 1))

    def train(self, override=False):
        """
        Trains the neural network.
        Params:
            override (bool): whether to override the previous identical model, can be used
                when you changed the dataset, default is False
        """
        # if model isn't created yet, create it
        if not self.model_created:
            self.create_model()
```

```
        # if the model already exists and trained, just load the weights and return
        # but if override is True, then just skip loading weights
        if not override:
            model_name = self._model_exists()
            if model_name:
                self.model.load_weights(model_name)
                self.model_trained = True
                if self.verbose > 0:
                    print("[*] Model weights loaded")
                return

        if not os.path.isdir("results"):
            os.mkdir("results")

        if not os.path.isdir("logs"):
            os.mkdir("logs")

        model_filename = self._get_model_filename()

        self.checkpointer = ModelCheckpoint(model_filename, save_best_only=True,
verbose=1)
        self.tensorboard = TensorBoard(log_dir=f"logs/{self.model_name}")

        self.history = self.model.fit(self.X_train, self.y_train,
                                      batch_size=self.batch_size,
                                      epochs=self.epochs,
                                      validation_data=(self.X_test, self.y_test),
                                      callbacks=[self.checkpointer, self.tensorboard],
                                      verbose=self.verbose)

        self.model_trained = True
```

```
        if self.verbose > 0:
            print("[+] Model trained")

    def predict(self, audio_path):
        feature = extract_feature(audio_path, **self.audio_config).reshape((1, 1,
self.input_length))
        if self.classification:
            return self.int2emotions[self.model.predict_classes(feature)[0][0]]
        else:
            return self.model.predict(feature)[0][0][0]

    def predict_proba(self, audio_path):
        if self.classification:
            feature = extract_feature(audio_path, **self.audio_config).reshape((1, 1,
self.input_length))
            proba = self.model.predict(feature)[0][0]
            result = {}
            for prob, emotion in zip(proba, self.emotions):
                result[emotion] = prob
            return result
        else:
            raise NotImplementedError("Probability prediction doesn't make sense for
regression")

    def test_score(self):
        y_test = self.y_test[0]
        if self.classification:
            y_pred = self.model.predict_classes(self.X_test)[0]
            y_test = [np.argmax(y, out=None, axis=None) for y in y_test]
            return accuracy_score(y_true=y_test, y_pred=y_pred)
        else:
```

```
            y_pred = self.model.predict(self.X_test)[0]
            return mean_absolute_error(y_true=y_test, y_pred=y_pred)

    def train_score(self):
        y_train = self.y_train[0]
        if self.classification:
            y_pred = self.model.predict_classes(self.X_train)[0]
            y_train = [np.argmax(y, out=None, axis=None) for y in y_train]
            return accuracy_score(y_true=y_train, y_pred=y_pred)
        else:
            y_pred = self.model.predict(self.X_train)[0]
            return mean_absolute_error(y_true=y_train, y_pred=y_pred)

    def confusion_matrix(self, percentage=True, labeled=True):
        """Compute confusion matrix to evaluate the test accuracy of the classification"""
        if not self.classification:
            raise NotImplementedError("Confusion matrix works only when it is a
classification problem")
        y_pred = self.model.predict_classes(self.X_test)[0]
        # invert from keras.utils.to_categorical
        y_test = np.array([np.argmax(y, axis=None, out=None) for y in self.y_test[0]])
        matrix = confusion_matrix(y_test, y_pred, labels=[self.emotions2int[e] for e in
self.emotions]).astype(
            np.float32)
        if percentage:
            for i in range(len(matrix)):
                matrix[i] = matrix[i] / np.sum(matrix[i])
            # make it percentage
            matrix *= 100
        if labeled:
            matrix = pd.DataFrame(matrix, index=[f"true_{e}" for e in self.emotions],
```

```
                                columns=[f"predicted_{e}" for e in self.emotions])
        return matrix


    def n_emotions(self, emotion, partition):
        """Returns number of `emotion` data samples in a particular `partition`
        ('test' or 'train')
        """
        if partition == "test":
            if self.classification:
                y_test = np.array([np.argmax(y, axis=None, out=None) + 1 for y in
np.squeeze(self.y_test)])
            else:
                y_test = np.squeeze(self.y_test)
            return len([y for y in y_test if y == emotion])
        elif partition == "train":
            if self.classification:
                y_train = np.array([np.argmax(y, axis=None, out=None) + 1 for y in
np.squeeze(self.y_train)])
            else:
                y_train = np.squeeze(self.y_train)
            return len([y for y in y_train if y == emotion])


    def get_samples_by_class(self):
        """
        Returns a dataframe that contains the number of training
        and testing samples for all emotions
        """
        train_samples = []
        test_samples = []
        total = []
        for emotion in self.emotions:
```

```
            n_train = self.n_emotions(self.emotions2int[emotion] + 1, "train")
            n_test = self.n_emotions(self.emotions2int[emotion] + 1, "test")
            train_samples.append(n_train)
            test_samples.append(n_test)
            total.append(n_train + n_test)

        # get total
        total.append(sum(train_samples) + sum(test_samples))
        train_samples.append(sum(train_samples))
        test_samples.append(sum(test_samples))
        return pd.DataFrame(data={"train": train_samples, "test": test_samples, "total":
total},
                            index=self.emotions + ["total"])

    def get_random_emotion(self, emotion, partition="train"):
        """
        Returns random `emotion` data sample index on `partition`
        """
        if partition == "train":
            y_train = self.y_train[0]
            index = random.choice(list(range(len(y_train))))
            element = self.int2emotions[np.argmax(y_train[index])]
            while element != emotion:
                index = random.choice(list(range(len(y_train))))
                element = self.int2emotions[np.argmax(y_train[index])]
        elif partition == "test":
            y_test = self.y_test[0]
            index = random.choice(list(range(len(y_test))))
            element = self.int2emotions[np.argmax(y_test[index])]
            while element != emotion:
                index = random.choice(list(range(len(y_test))))
```

```
            element = self.int2emotions[np.argmax(y_test[index])]
        else:
            raise TypeError("Unknown partition, only 'train' or 'test' is accepted")

        return index

    def determine_best_model(self, train=True):
        # TODO
        raise TypeError("This method isn't supported yet for deep nn")


if __name__ == "__main__":
    rec = DeepEmotionRecognizer(emotions=['angry', 'sad', 'neutral', 'fear', 'happy'],
                                epochs=300, verbose=0) #mahitha
    rec.train(override=False)
    print("Test accuracy score:", rec.test_score() * 100, "%")
```

- **test.py**

```
from emotion_recognition import EmotionRecognizer

import pyaudio
import os
import wave
from sys import byteorder
from array import array
from struct import pack
from sklearn.ensemble import GradientBoostingClassifier, BaggingClassifier

from utils import get_best_estimators

THRESHOLD = 500
```

```
CHUNK_SIZE = 1024
FORMAT = pyaudio.paInt16
RATE = 16000

SILENCE = 30


def is_silent(snd_data):
    "Returns 'True' if below the 'silent' threshold"
    return max(snd_data) < THRESHOLD


def normalize(snd_data):
    "Average the volume out"
    MAXIMUM = 16384
    times = float(MAXIMUM) / max(abs(i) for i in snd_data)

    r = array('h')
    for i in snd_data:
        r.append(int(i * times))
    return r


def trim(snd_data):
    "Trim the blank spots at the start and end"

    def _trim(snd_data):
        snd_started = False
        r = array('h')

        for i in snd_data:
```

```
            if not snd_started and abs(i) > THRESHOLD:
                snd_started = True
                r.append(i)

            elif snd_started:
                r.append(i)
        return r

    # Trim to the left
    snd_data = _trim(snd_data)

    # Trim to the right
    snd_data.reverse()
    snd_data = _trim(snd_data)
    snd_data.reverse()
    return snd_data


def add_silence(snd_data, seconds):
    "Add silence to the start and end of 'snd_data' of length 'seconds' (float)"
    r = array('h', [0 for i in range(int(seconds * RATE))])
    r.extend(snd_data)
    r.extend([0 for i in range(int(seconds * RATE))])
    return r


def record():
    """
    Record a word or words from the microphone and
    return the data as an array of signed shorts.
```

```
Normalizes the audio, trims silence from the
start and end, and pads with 0.5 seconds of
blank sound to make sure VLC et al can play
it without getting chopped off.
"""
p = pyaudio.PyAudio()
stream = p.open(format=FORMAT, channels=1, rate=RATE,
                input=True, output=True,
                frames_per_buffer=CHUNK_SIZE)

num_silent = 0
snd_started = False

r = array('h')

while 1:
    # little endian, signed short
    snd_data = array('h', stream.read(CHUNK_SIZE))
    if byteorder == 'big':
        snd_data.byteswap()
    r.extend(snd_data)

    silent = is_silent(snd_data)

    if silent and snd_started:
        num_silent += 1
    elif not silent and not snd_started:
        snd_started = True

    if snd_started and num_silent > SILENCE:
        break
```

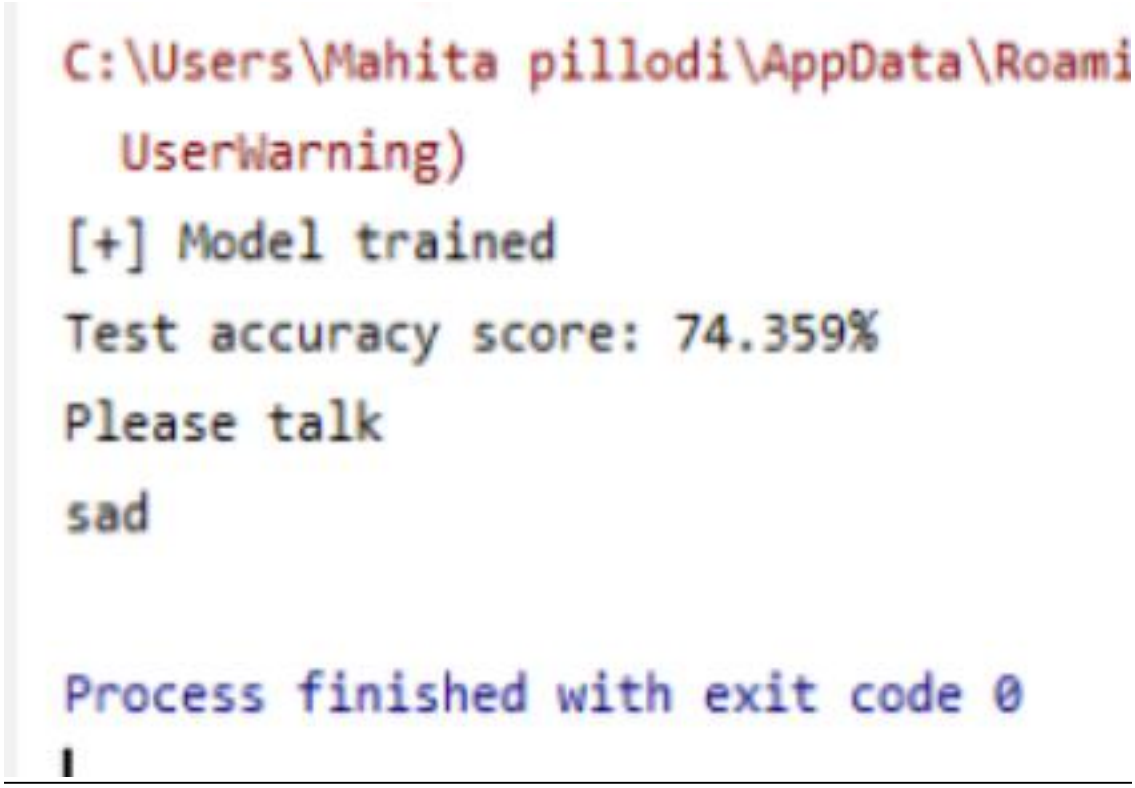
```
    sample_width = p.get_sample_size(FORMAT)
    stream.stop_stream()
    stream.close()
    p.terminate()

    r = normalize(r)
    r = trim(r)
    r = add_silence(r, 0.5)
    return sample_width, r


def record_to_file(path):
    "Records from the microphone and outputs the resulting data to 'path'"
    sample_width, data = record()
    data = pack('<' + ('h' * len(data)), *data)

    wf = wave.open(path, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(sample_width)
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()


def get_estimators_name(estimators):
    result = ['"{}"'.format(estimator.__class__.__name__) for estimator, _, _ in
estimators]
    return ','.join(result), {estimator_name.strip('"'): estimator for
estimator_name, (estimator, _, _) in
                              zip(result, estimators)}
```

```
if __name__ == "__main__":
    estimators = get_best_estimators(True)
    estimators_str, estimator_dict = get_estimators_name(estimators)
    import argparse

    parser = argparse.ArgumentParser(description="""
                            Testing emotion recognition system using your voice,
                            please consider changing the model and/or parameters
as you wish.
                            """)
    parser.add_argument("-e", "--emotions", help=
    """Emotions to recognize separated by a comma ',', available emotions
are
    "neutral", "calm", "happy" "sad", "angry", "fear", "disgust", "ps"
(pleasant surprise)
    and "boredom", default is "sad,neutral,happy"
    """, default="sad,neutral,happy")
    parser.add_argument("-m", "--model", help=
    """
    The model to use, 8 models available are: {},
    default is "BaggingClassifier"
    """.format(estimators_str), default="BaggingClassifier")

    # Parse the arguments passed
    args = parser.parse_args()

    features = ["mfcc", "chroma", "mel"]
    detector = EmotionRecognizer(estimator_dict[args.model],
emotions=args.emotions.split(","), features=features,
```

```
                    verbose=0)
detector.train()
print("Test accuracy score: {:.3f}%".format(detector.test_score() * 100))
print("Please talk")

filename = "test.wav"
record_to_file(filename)
result = detector.predict(filename)
print(result)
```

Output (Screenshots)

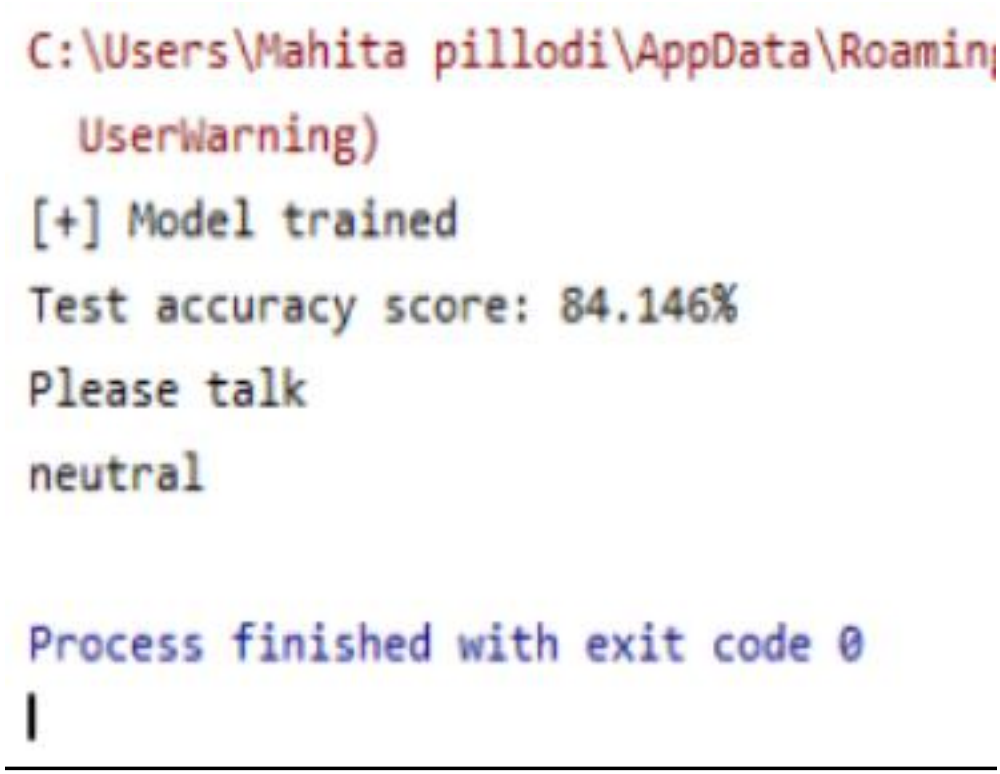
- **Sad**



- **Happy**

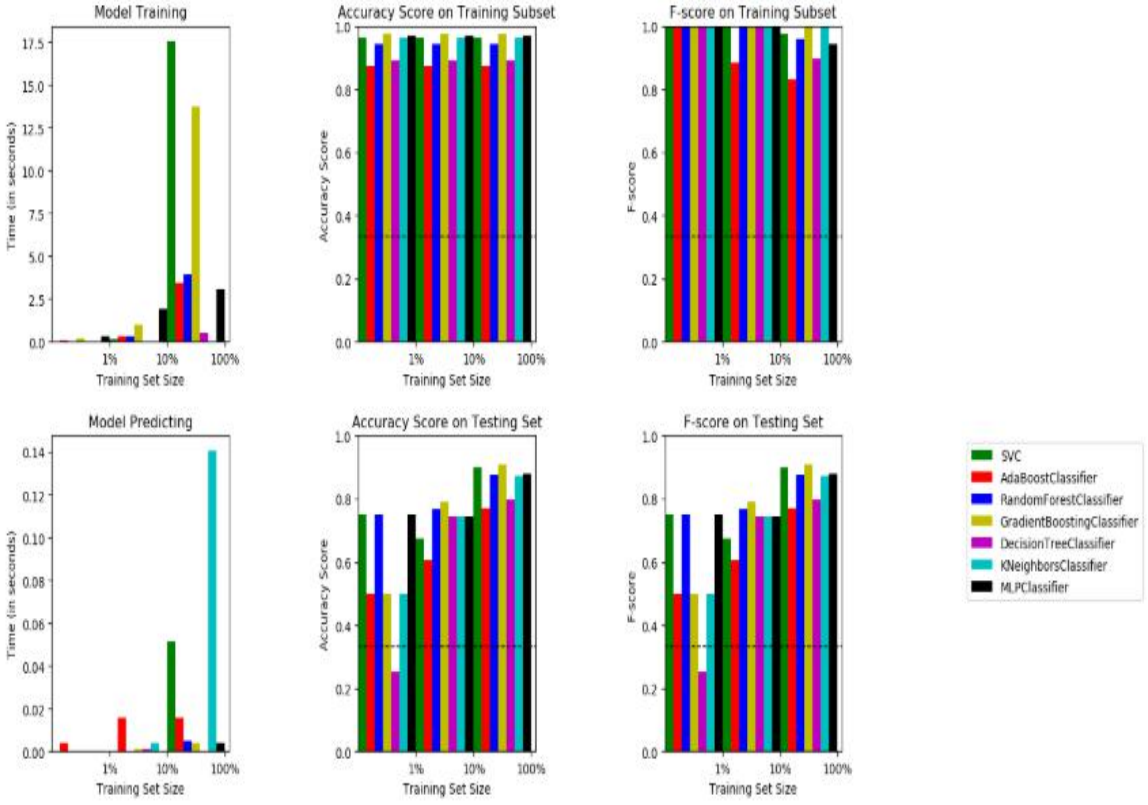
```
C:\Users\Mahita pillodi\AppData\Roam
  UserWarning)
[+] Model trained
Test accuracy score: 83.333%
Please talk
happy

Process finished with exit code 0
```

- **Neutral**


C:\Users\Mahita pillodi\AppData\Roamin
UserWarning)
[+] Model trained
Test accuracy score: 84.146%
Please talk
neutral
Process finished with exit code 0

Bar Graph:

Output:



5. Conclusion

We reviewed and discussed various speeches emotional recognition systems based approaches. We also compare its performance in terms of classifier, features, recognition rate, and datasets. Well-design classifiers have obtain high classification accuracies between different types of emotions. Most of the current research concentrate on investigating different features and their correlation with emotional state in spoken speech. In this fact some researcher develop their own feature like MLS to achieve high performance in recognition rate. The majority of the current datasets are not capable for evaluation of speech emotion recognition. In most of them, it is hard even for human to specify different emotion of certain collected utterances; e.g. the human recognition accuracy was 80% for Berlin . In conclusion, there are only limited studies that considered applying multiple classifier to speech emotion recognition . We consider multiple classifier methods (MCM) as a further research direction, which has to be, explore in future.

In this current study, we presented an automatic speech emotion recognition (SER) system using RNN as the machine learning algorithmsto classify six emotions. Thus, two types of features (MFCC and MS) were extracted from two different acted databases (Berlin and Spanish databases), and a combination of these features was presented. In fact, we study how classifiers and features impact recognition accuracy of emotions in speech. A subset of highly discriminant features is selected. Feature selection techniques show that more information is not always good in machine learning applications. The machine learning models were trained and evaluated to recognize emotional states from these features. SER reported the best recognition rate of 94% on the Spanish database using RNN classifier without speaker normalization (SN) and with feature selection (FS). For Berlin database, all of the classifiers achieve an accuracy of 83% when a speaker normalization (SN) and a feature selection (FS) are applied to the features. From this result, we can see that RNN often perform better with more data and it suffers from the problem of very long training times.

6.Future Work

The general experimental evaluation of the RNN model guarantees better voice recognition rates. Having examined techniques to cope with voice variation, in future it may be investigated in more depth about the voice recognition problem and optimal fusion of pitch and context information. Further study can be laid down in the direction of using sequencial context analysis to generate better results for same phrases. The RNN model for voice recognition can be studied to suit the requirement of different security models such as criminal detection, governmental confidential security breaches. This model can be used to find out the emotions of the persons suffering from various problems to understand their emotions by adding more detailed emotions and more features to determine the emotion.

Moreover, emotion recognition can also be used devising marketing strategies by understanding the customers behaviours and emotions, Being able to adjust one’s marketing dynamically, based on the real-time reactions of your audience, will empower marketers to provide the right message at the right time to the right person.In the future, it's likely that you'll be able to calibrate your marketing mid-stream with just about any digital experience -- no two prospects may experience a brand’s marketing in the same exact way.

Going beyond advertising, you can already see how Facebook is on a path to incorporating emotional reaction into a user’s News Feed by introducing "Reactions" this past February. Rather than simply "Liking" a post, users can now designate their reaction across six emotions: Like, Love, Haha, Wow, Sad, and Angry.

It's safe to say that we should expect a great deal of change in the world of marketing moving forward. Gabi Zijderveld, CMO at Affectiva, provides a peek into the future by explaining the evolution of emotion recognition technology:

7.References

- Yoon, Seunghyun, Seokhyun Byun, and Kyomin Jung. "Multimodal speech emotion recognition using audio and text." 2018 IEEE Spoken Language Technology Workshop. (SLT). IEEE, 2018

- Tripathi, Samarth, and Homayoon Beigi. "Multi-modal emotion recognition on iemocap dataset using deep learning." arXiv preprint arXiv:1804.05788 (2018).

- Sahu, Gaurav. "Multimodal Speech Emotion Recognition and Ambiguity Resolution." arXiv preprint arXiv:1904.06022 (2019).

- Lian, Zheng, et al. "Investigation of Multimodal Features, Classifiers and Fusion Methods for Emotion Recognition." arXiv preprint arXiv:1809.06225 (2018).

Presented the project in "SHOW AND TELL"