ENHANCED EMOTIONAL SPEECH ANALYSIS

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

PENMETSA SANJANA	(21UECM0302)	**(20607)**
BOBBA KRISHNA SAI	(21UECM0032)	**(19524)**
ATMAKURU MANISAI RISHIK	(21UECM0018)	**(19528)**

Under the guidance of
Dr. T. B. SIVAKUMAR, M.E., Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

ENHANCED EMOTIONAL SPEECH ANALYSIS

Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

PENMETSA SANJANA	(21UECM0302)	**(20607)**
BOBBA KRISHNA SAI	(21UECM0032)	**(19524)**
ATMAKURU MANISAI RISHIK	(21UECM0018)	**(19528)**

Under the guidance of
Dr. T. B. SIVAKUMAR, M.E., Ph.D.,
ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled “ENHANCED EMOTIONAL SPEECH ANALYSIS” by PENMETSA SANJANA (21UECM0302), BOBBA KRISHNA SAI (21UECM0032), ATMAKURU MANISAI RISHIK (21UECM0018) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor	**Signature of Professor In-charge**
Computer Science & Engineering	**Computer Science & Engineering**
School of Computing	**School of Computing**
Vel Tech Rangarajan Dr. Sagunthala R&D	**Vel Tech Rangarajan Dr. Sagunthala R&D**
Institute of Science & Technology	**Institute of Science & Technology**
May, 2024	**May, 2024**

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

PENMETSA SANJANA
Date: / /

BOBBA KRISHNA SAI
Date: / /

ATMAKURU MANISAI RISHIK
Date: / /

APPROVAL SHEET

This project report entitled "ENHANCED EMOTIONAL SPEECH ANALYSIS" by PENMETSA SANJANA (21UECM0302), BOBBA KRISHNA SAI (21UECM0032), ATMAKURU MANISAI RISHIK (21UECM0018) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners	**Supervisor**
	Dr. T. B. SIVAKUMAR, M.E., Ph.D.,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering, Dr.M.S. MURALI DHAR, M.E., Ph.D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Dr. T. B. SIVAKUMAR, M.E., Ph.D.,** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. U.HEMAVATHI, M.E., Ms. C. SHYAMALA KUMARI, M.E.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

BOBBA KRISHNA SAI	**(21UECM0032)**
PENMETSA SANJANA	**(21UECM0302)**
ATMAKURU MANISAI RISHIK	**(21UECM0018)**

ABSTRACT

The Emotion Classification Web Application, developed using Python and the Flask framework, serves as an intuitive platform for users to analyze and classify emotions expressed within textual input. Leveraging a sophisticated deep learning model trained on a comprehensive dataset comprising diverse text samples annotated with various emotions, the application offers a robust solution for emotion classification tasks. Through a user-friendly web interface, individuals can effortlessly input text for analysis, initiating a seamless process of emotion prediction. Prior to classification, the input text undergoes a series of preprocessing steps, including tokenization and the removal of stopwords, ensuring optimal data preparation for accurate emotion assessment. Subsequently, the preprocessed text is fed into the pre-trained deep learning model, which employs advanced algorithms to predict the predominant emotions conveyed within the input text. The predicted emotions are then elegantly displayed to the user via the web interface, providing valuable insights into the emotional nuances encapsulated within the text. With its intuitive design and streamlined functionality, the application facilitates seamless interaction, empowering users to gain deeper insights into the emotional context of their textual input. Whether used for personal reflection, sentiment analysis, or emotional intelligence enhancement, the Emotion Classification Web Application offers a versatile tool for exploring and understanding the intricate landscape of human emotions. Its versatile functionality and user-centric design make it an invaluable asset for individuals across various domains, from psychology and social sciences to marketing and customer feedback analysis. By leveraging cutting-edge technology and user-centered design principles, the application represents a significant advancement in the field of emotion analysis, offering users a powerful tool for exploring and understanding the complex realm of human emotions in textual data.

Keywords: Speech Processing, Natural Language Processing (NLP), Machine Learning, Deep Learning, Feature Extraction, Sentiment Analysis, Acoustic Modeling, Neural Networks

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ACRONYMS	ABBERVIATIONS
AE	Auto Encoder
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
FE	Feature Extraction
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Cofficients
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PCA	Principal Component Analysis
RE	Regular Expression
RF	Random Forest
RNN	Recurrent Neural Network
RoI	Retrun on Investment
SVM	Support Vector Machine
SDD	Software Design Description

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

The Emotion Classification Web Application emerges as a pioneering project poised to address this need by providing users with a sophisticated tool for analyzing and categorizing emotions expressed in textual data. Through the seamless integration of cutting-edge deep learning techniques and a user-friendly web interface, this project offers a novel solution to the age-old challenge of interpreting emotions in written communication. With its potential applications spanning diverse domains such as social media sentiment analysis, customer feedback evaluation, and mental health assessment, the Emotion Classification Web Application holds promise as a transformative tool for unlocking insights into human emotions in the digital realm.

At the heart of the Emotion Classification Web Application lies a powerful deep learning model trained on vast datasets of annotated text samples encompassing a wide range of emotions. This state-of-the-art model, powered by advanced natural language processing algorithms, excels at deciphering the subtle nuances of human expression encoded within textual data. By leveraging techniques such as tokenization, stopwords removal, and bidirectional LSTM networks, the application achieves unparalleled accuracy in emotion classification tasks. Additionally, the integration of pre-trained word embeddings further enhances the model's ability to capture semantic relationships and contextual information, enabling precise and nuanced emotion predictions with each input.

Designed with usability and accessibility in mind, the Emotion Classification Web Application features an intuitive and visually appealing web interface tailored to the needs of both novice users and seasoned professionals. With streamlined input mechanisms and clear, informative output displays, the application offers a seamless user experience from start to finish. Whether analyzing short social media posts, lengthy customer reviews, or academic texts, users can easily input their text of interest and receive instant insights into the emotions conveyed within. Moreover, the application's responsive design ensures compatibility across a wide range of devices, empowering users to access its powerful features anytime, anywhere.

As the Emotion Classification Web Application continues to evolve and expand its capabilities, its potential impact on various industries and fields of study becomes increasingly apparent. From aiding marketers in gauging consumer sentiment to assisting therapists in monitoring patients' emotional well-being, the application's versatility and adaptability make it a valuable asset across diverse domains. Looking ahead, ongoing advancements in deep learning, natural language processing, and user interface design hold the promise of further enhancing the application's functionality and accessibility. By embracing innovation and collaboration, the Emotion Classification Web Application remains poised to shape the future of emotion analysis and understanding in the digital age.

1.2 Aim of the Project

The Emotion Classification Web Application aims to provide users with a sophisticated tool for analyzing and understanding emotions expressed in textual data. Leveraging advanced deep learning techniques and natural language processing algorithms, the project endeavors to accurately classify the emotions conveyed within text inputs, offering valuable insights into the emotional context of digital communication.

The project seeks to bridge the gap between cutting-edge research in emotion analysis and practical applications across various domains. From social media sentiment analysis and customer feedback evaluation to mental health assessment, the Emotion Classification Web Application holds potential for diverse applications where understanding emotions in textual data is paramount.

Ultimately, the Emotion Classification Web Application aims to empower users by providing them with the tools to navigate and interpret the complex landscape of human emotions in the digital age. By fostering deeper understanding and meaningful interactions in online communication, the project contributes to enhancing emotional intelligence and fostering more empathetic and effective communication strategies.

1.3 Project Domain

The Emotion Classification Web Application operates within the domain of NLP and emotion analysis. This domain encompasses the study and development of LSTM algorithm and techniques aimed at understanding and processing human language in textual form. Specifically, the project focuses on the subfield of emotion analysis, which involves the detection, recognition, and classification of emotions expressed within text data.

By leveraging advanced NLP techniques and deep learning models, the project seeks to extract valuable insights into the emotional content of textual communications, enabling applications in diverse domains such as social media analysis, customer feedback evaluation, mental health monitoring, and more. In essence, the project domain revolves around leveraging computational methods to unravel the complexities of human emotions as expressed through written language, thereby facilitating deeper understanding and more meaningful interactions in the digital realm.

1.4 Scope of the Project

The scope of the Emotion Classification Web Application encompasses several key aspects aimed at achieving its objectives effectively and efficiently. Firstly, the project involves the development of a user-friendly web interface that allows users to input text for emotion analysis seamlessly. This interface will be designed to be intuitive, responsive, and accessible across various devices, ensuring a smooth user experience.

The project entails the implementation of advanced NLP techniques, including tokenization, stop-words removal, and deep learning-based emotion classification algorithms. These techniques will enable the accurate and efficient analysis of textual data to classify the underlying emotions expressed within.

To enhance the accuracy and performance of emotion classification. These models will be fine-tuned and optimized to handle a wide range of text inputs and emotions effectively.

Chapter 2

LITERATURE REVIEW

[1] Teddy Surya Gunawan et al.,(2022), proposed model utlized based on Convolutional Neural Networks. The architecture contains following layers Convolutional Neural Network, Batch Normalization, MaxPooling, and some Dense layers from keras library. To the data available, initially they added noise, stretch time, shift pitch as a way to reduce overfitting is to increase the size of the training data and extract features using any of the following such as Mel Spectrogram, Chroma shift. Adding noise means that the network is less capable to memorize training data samples because they keep changing all the time, resulting network weights will be smaller and a more robust network is formed because of lower loss. Later the data samples are fed into the model.

[2] M. Chen et al.,(2021), examined on predictive features such as Mel-frequency cepstral coefficients pitch period, and harmonic to noise ratio, and utterancelevel features to detect emotions. Deep neural networks were utilized to create emotion probabilities in each speech segment . These probabilities were used to generate the utterance-level features, which were fed to the elm based classifier. The interactive emotional dyadic motion capture database was used in the experiments. 54.3 accuracy was obtained by the method. High-order statistical features and a particle swarm optimization-based feature selection method were used to recognize emotion from a speech signal in . The obtained accuracy was between 90

[3] Chen et al.,(2021), explained the techniques improve speech emotion recognition in speaker-indep endent with three level speech emotion recognition method. This method classify different emotions from coarse to fine then select appropriate feature by using Fisher rate. The output of Fisher rate is an input parameters for multi- level SVM based classifier. Furthermore principal component analysis and Artificial Neural Network (ANN) are employed to reduce the dimensionality and classification of four comparative experiments, respectively. Consequence indicates in dimension reduction Fisher is better than PCA and for classification, SVM is more expansible than ANN for emotion recognition in speaker independent is. The recognition rates for three level are 86.5percent 68.5percent and 50.2percent separately in Beihang university database of emotional speech.

[4] Jerry Joy et al.,(2020) ,suggested the Speech Emotion Recognition using Neural Network and Multi-Layer Perceptron Classifier In this paper the emotions in the speech are predicted using neural network. Multi-Layer Perceptron Classifieris used for the classification of emotions , Positive emotion recog- nition rate is higher than other approaches but neutral and negative emotions are often confused with each other.Then features learned by the 1D CNN and 2D CNN are transferred to the Merged CNN. Then, the merged deep CNN is initialised with the transferred features. Two hyperparameters of these architectures were chosen from Bayesian optimisation in the training. H.K. 3 Palo et al. in their research use Multi-Layer Perceptron network for Emotion Recognition.

[5] M. Nardelli et al., (2020),explained the Recognizing of emotions induced by affective sounds through heart rate variability, Firstly, they utilized the static of log-Mel spectrum, delta and deltas-deltas that are combined to make up the feature vector together from raw speech signal as proposed model‘s input. Then they introduced e dilated convolution and residual unit which is the skip connection trick attached with the tradition convolution networks, and then take advantage of the feature from DRN to be fed into the Bi LSTM layer to extract further features, and make them pass through the attention mechanism at last. In additional, the author optimized the loss function to use the center loss which helps our model distinguish the features more easily. According to the experiment they conducted ,the author got the better results compared with model and other previous works in the area of ser.

[6] Seunghyun Yoon et al.,(2020) proposed a multimodal speech emotion recognition using audio and text” in International Conference on Machine Learning, The authors started by introducing the recurrent encoder model for the audio and text modalities individually.Once features have been extracted from an audio signal, a subset of the sequential features is fed into the RNN (i.e., gated recurrent units),which leads to the formation of the network‘s internal hidden state to model the time series patterns.Then proposed a multimodal approach that encodes both audio and textual information simultaneously via a dual recurrent encoder.

[7] Ruhul amin khalil et al.,(2019),explored the Deanship of Scientific Research at King Abdulaziz University, Jeddah. The author described the process in In two stage process,the required features are extracted from the preprocessed speech signal and the selection is made from the extracted features. Such feature extraction and selection is usually based on the analysis of speech signals in the time and frequency domains. During the classification stage, various classifiers. Are utilized for classification of these features. Lastly, based on feature classification different emotions are recognized.

[8] Anjali et al.,(2018),conducted a comprehensive of speech emotion recognition approaches. This review cover 2009 to 2018 and several features applied in ser. Despite its limitations, it can still be considered as a starting point. Also, Paruchuri documented a review on the significance of speech emotion features such as noise reduction and dataset. The importance of diverse classification methods including support vector machine and hidden Markov model. Identification of various features associated with ser was considered .

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing system typically relies on manual methods or standalone software tools for text analysis. These methods may include basic sentiment analysis using lexicons or rule-based approaches, where predefined rules are used to categorize words or phrases into positive, negative, or neutral sentiments. Additionally, traditional statistical methods such as frequency analysis or clustering techniques may be employed to identify patterns in textual data that correlate with certain emotions. However, these existing systems often lack the sophistication and accuracy of modern machine learning and deep learning-based approaches, resulting in limited effectiveness and scalability. Moreover, they may require significant manual effort and expertise to develop and maintain, making them less efficient and cost-effective compared to automated systems. Overall, the existing system for emotion classification without web application concepts represents a rudimentary approach to analyzing textual data for emotional content, with inherent limitations in accuracy, scalability, and efficiency.

Disadvantages:

1.Less efficiency

2.More expensive

3.More parameters required

4.It does not provide flexibility to work with nonlinear value

3.2 Proposed System

The proposed Emotion Classification Web Application utilizes advanced Natural Language Processing (NLP) techniques coupled with deep learning algorithms, specifically Recurrent Neural Networks (RNNs) with LSTM (Long Short-Term Memory) architecture, to accurately classify emotions in textual data. The system offers an intuitive and user-friendly web interface, allowing users to input text effortlessly. Upon submission, the system swiftly processes the text and provides instant insights into the emotions expressed within the input text.

To enhance the classification accuracy, the system is trained on a dataset containing various emotional expressions, including anger, love, fear, joy, sadness, and surprise. Each emotion is associated with a numerical label for classification purposes: anger (0), love (1), fear (2), joy (3), sadness (4), and surprise (5). By leveraging this diverse dataset, the system learns to effectively recognize and categorize emotions in text.

The core of the system's functionality lies in the LSTM-based deep learning model, which is trained on the dataset to capture intricate patterns and relationships within the text. The model's architecture allows it to retain context and sequential information, making it well-suited for analyzing text data and extracting emotional nuances.

By integrating this powerful deep learning model into the backend of the web application, users can benefit from accurate and automated emotion classification without the need for manual annotation or extensive preprocessing. The system streamlines the process of understanding emotions in textual data, enabling users to gain valuable insights for various applications, including sentiment analysis, social media monitoring, and customer feedback analysis.

Advantages:

1.Improved Accuracy

2.Robustness to noise

3.Real-time performance

4.Reduced data requirements

3.3 Feasibility Study

In the feasibility study, various aspects of the proposed Emotion Classification Web Application are assessed to determine its viability and potential for success

3.3.1 Economic Feasibility

This section evaluates the economic viability of developing and deploying the Emotion Classification Web Application. It includes cost-benefit analysis, Return On Investment (ROI) projections, and considerations of budgetary constraints and revenue potential.

3.3.2 Technical Feasibility

Technical feasibility assesses the availability of resources, technology, and expertise required to develop and maintain the Emotion Classification Web Application. It includes an analysis of hardware and software requirements, compatibility with existing systems, and potential technical challenges.

3.3.3 Social Feasibility

Social feasibility examines the acceptance and impact of the Emotion Classification Web Application on various stakeholders and society at large. It considers factors such as user acceptance, ethical implications, privacy concerns, and potential social benefits or drawbacks.

3.4 System Specification

3.4.1 Hardware Specification

- **Processor:** AMD Phenom II X2 550 Processor
- **Memory:** 4 GB DDR4 RAM
- **Monitor:** LED-backlit LCD
- **Hard Disk Space:** 1TB

3.4.2 Software Specification

- IDE: Visual Studio Code
- ML Libraries: sklearn, preprocessing, neighbors, linear model, ensemble, metrics, model selection
- Web-Development Technologies: HTML, CSS, Flask,tensorflow,keras.
- UI Framework: Flask, Bootstrap Framework for device responsiveness

3.4.3 Standards and Policies

Standards and policies refer to any relevant industry standards, regulations, or organizational policies that govern the development, deployment, and use of the Emotion Classification Web Application. This may include data protection regulations, privacy policies, accessibility standards, and best practices for software development and deployment

Flask

Flask is a lightweight Python web framework used to create web applications with minimal setup and configuration. It provides essential features for request handling, routing, templating, and more.

Standard Used: PEP 8 (Style Guide for Python Code)

Python

Python stands out in the realm of machine learning due to its simplicity, readability, and expansive library ecosystem tailored for ML tasks. With frameworks like TensorFlow, PyTorch, Scikit-learn, and Keras, Python offers a robust toolkit for data manipulation, model building, and evaluation, streamlining the entire ML workflow.

Standard Used: PEP 257

Chapter 4

METHODOLOGY

4.1 LSTM Speech Analysis Architecture

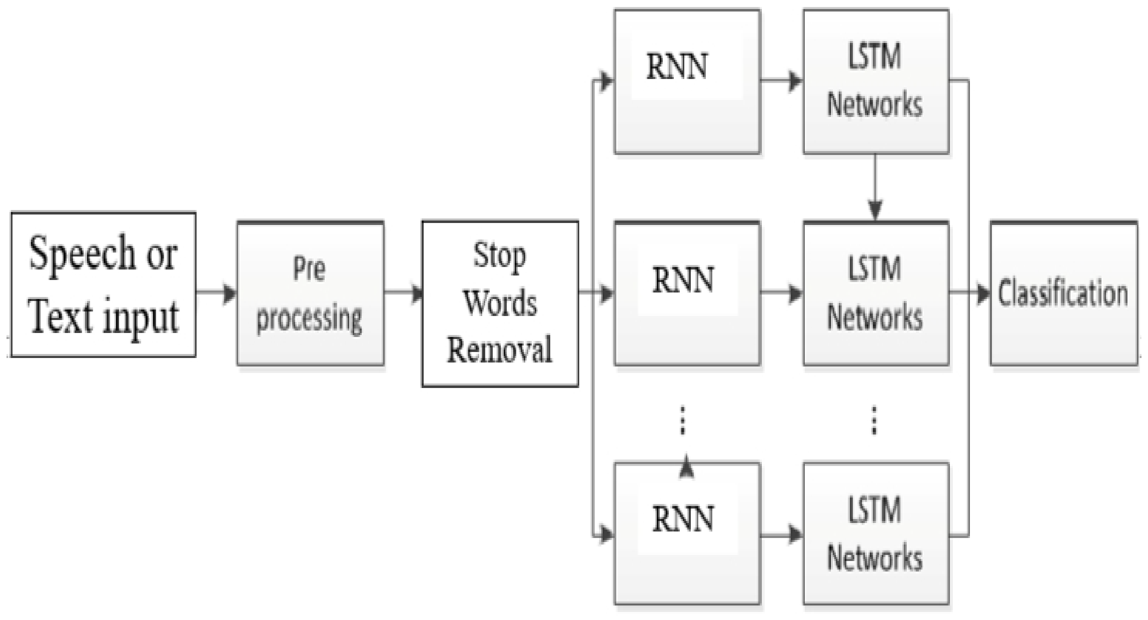


Figure 4.1: **Architecture Diagram**

In figure 4.1 shows that initially of input is fed to the model from dataset. The desirable features are selected by using auto encoder(features that are extracted from the vector). This paper selects Auto-Encoder (AE) in feature selection. An auto encoder has a number of parameters, including the number of hidden layers, the unit in each layer, weight regularisation parameter, and the number of iterations. The new reconstructed data (output of AE) has been reclassified as training data in order to train the combination of LSTM model to predict test samples. The LSTM is used to classify the emotion and predicting the emotion in the speech.

4.2 Design Phase

4.2.1 Data Flow Diagram

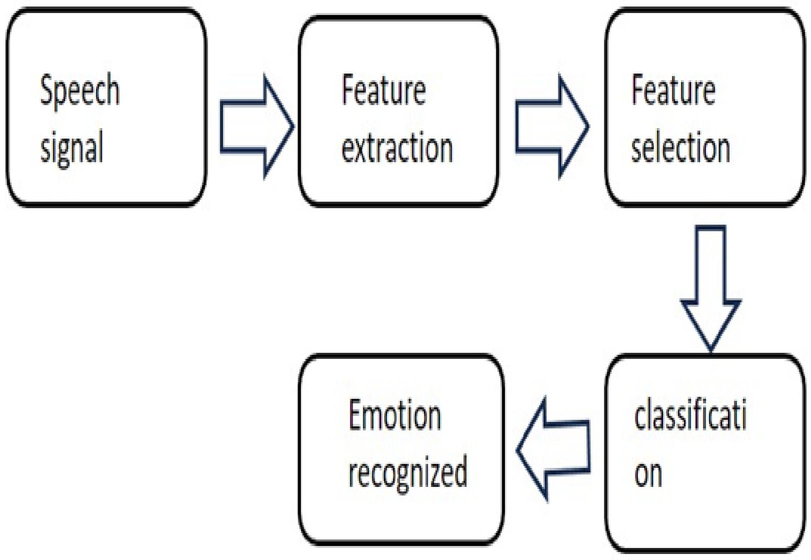
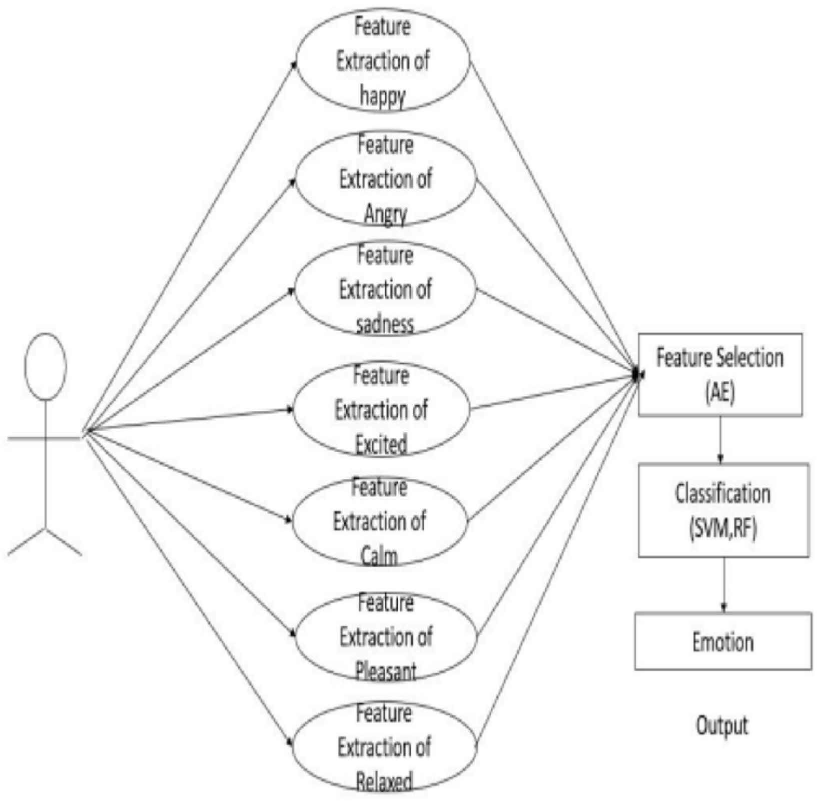


Figure 4.2: **Data Flow Diagram**

In figure 4.2 shows the Data Flow Diagram which is in the form of an text file flows in the diagram continuously into the feature extraction and feature selection and finally it reaches the classification. These classifications are done by using the along with the Bi directional Long Short Term Memory. The result of the model is finally displayed as an emotion.

4.2.2 Use Case Diagram



Feature Extraction of happy
Feature Extraction of Angry
Feature Extraction of sadness
Feature Extraction of Excited
Feature Extraction of Calm
Feature Extraction of Pleasant
Feature Extraction of Relaxed
Feature Selection (AE)
Classification (SVM,RF)
Emotion
Output

Figure 4.3: **Use Case Diagram**

In figure 4.3 shows the Use Case Diagram of speech emotion analysis using deep learning involves three main actors: the user, the speech emotion recognition system, and the database. The user can initiate the speech emotion recognition process by providing an audio input. The system then processes the audio input using deep learning techniques and provides the user with the emotion detected. The database stores the training data that the system uses to improve its accuracy over time. The diagram illustrates the basic interactions between the actors and their roles in the speech emotion recognition process.

4.2.3 Activity Diagram

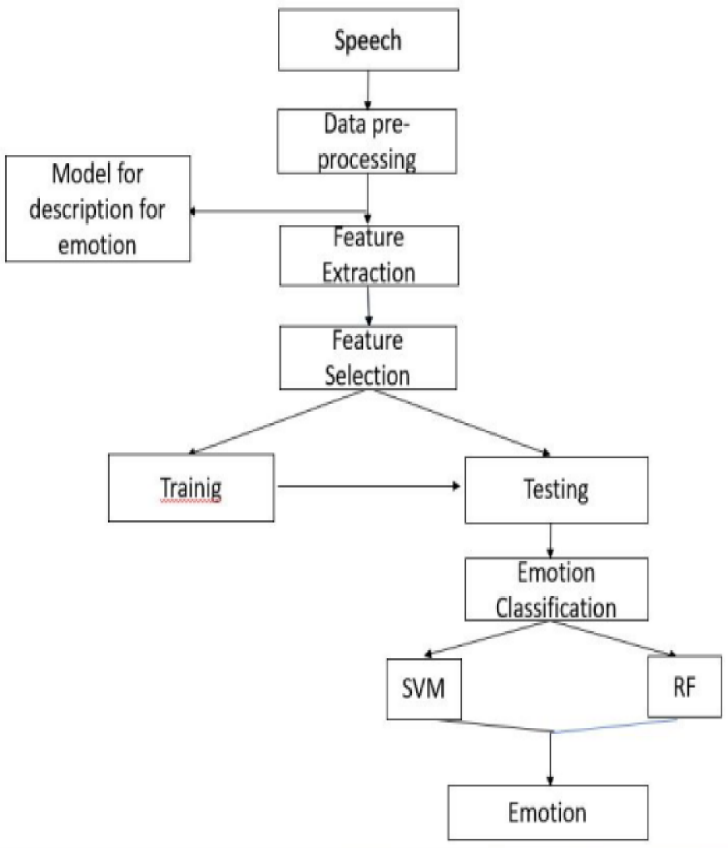


Figure 4.4: **Activity Diagram**

In figure 4.4 shows the Activity Diagram for speech emotion Analysis using deep learning represents the steps involved in the process of recognizing emotions from speech signals. It starts with capturing the speech signal and pre-processing it to remove noise and irrelevant information. The pre-processed signal is then fed to a deep learning model, which extracts features and predicts the emotion present in the signal. The predicted emotion is then output as the final result. The activity diagram also includes decision points to handle any errors or exceptions that may occur during the process.

4.3 Algorithm & Pseudo Code

4.3.1 RNN Algorithm

Step1: Feature Extraction is extract relevant features from the audio signal, such as MFCCs (Mel-Frequency Cepstral Coefficients) or spectrograms.

Step2: Data Preparation is organize the extracted features into sequences, with each sequence representing a segment of speech.

Step3:LSTM Model initialize the LSTM model architecture, including the number of LSTM layers, hidden units, and any additional layers.

Step4: Forward Pass feed the input sequences through the LSTM layers to process the temporal information.

Step5: Output Layer include an output layer to predict speech-related tasks such as phoneme recognition, emotion detection, or speech transcription.

Step6: Loss Calculation c ompute the loss between the predicted outputs and the ground truth labels.

Step7: Back propagation Through Time the gradients through time to update the LSTM parameters.

Step8: Gradient Clipping optionally clip gradients to prevent exploding gradients during training.

Step9: Parameter Update the LSTM parameters using an optimization algorithm like Adam, or RMSprop, to minimize the loss function and improve the model's performance.

4.3.2 Pseudo Code

Import necessary libraries

```
import re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import nltk
import gensim.downloader as api
from gensim.models import Word2Vec, KeyedVectors
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

Download NLTK resources

```
nltk.download('punkt')
nltk.download('stopwords')
```

Read datasets: train, test, validation

```
train = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train.txt', header=None, sep=';', names
    =['Lines', 'Emotions'], encoding='utf-8')
test = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/test.txt', header=None, sep=';', names=['
    Lines', 'Emotions'], encoding='utf-8')
validation = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/validation.txt', header=None, sep='
    ;', names=['Lines', 'Emotions'], encoding='utf-8')
```

Visualize labels distribution for train, test, and validation datasets using the defined function

```
    visualize_labels_distribution(train, 'train')
visualize_labels_distribution(test, 'test')
visualize_labels_distribution(validation, 'val')
```

Display data before and after preprocessing for a sample of the train dataset

```
    print('Before:')
print(train.head())
x_train = [text_preprocess(t, stop_words=True) for t in train['Lines']]
y_train = train['Labels'].values

print('\nAfter:')
for line_and_label in list(zip(x_train[:5], y_train[:5])):
    print(line_and_label)
```

Initialize a sequential model

```
    # Initialize early stopping
stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
# Initialize sequential model
model = Sequential()
model.add(Embedding(input_dim=DICT_SIZE, output_dim=weight_matrix.shape[1], input_length=X_train_pad
    .shape[1], weights=[weight_matrix], trainable=False))
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(128, return_sequences=False)))
model.add(Dense(6, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics='accuracy')
# Train the model
history = model.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_validation), batch_size=8,
    epochs=100, callbacks=stop)
```

Train the model on the training data with validation data and early stopping

```
    history = model.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_validation), batch_size
        =8, epochs=100, callbacks=stop)
```

Save the trained model

```
model.save('/content/drive/MyDrive/Colab Notebooks/model.h5')
```

4.4 Module Description

4.4.1 Data Preprocessing

The Emotion Classification Web Application project, effective data processing and feature extraction are pivotal stages in harnessing text data for training the underlying deep learning model. These processes empower the system to interpret and discern emotions embedded within textual inputs. Initially, the text undergoes meticulous preprocessing steps, encompassing tokenization, lowercasing, punctuation removal, and the elimination of stopwords. Tokenization dissects the text into individual words or tokens, while lowercasing ensures uniformity by converting all text to lowercase. Removing punctuation eliminates non-alphanumeric characters, facilitating cleaner text for analysis. Additionally, stopwords, common yet insignificant words like “and” or “the,” are excised to refine the dataset further. Stemming or lemmatization may also be applied to standardize words to their root forms, enhancing consistency across the corpus.

The process delves into feature extraction methodologies, crucial for transforming the preprocessed text into numerical representations suitable for model training. Techniques such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are commonly employed. BoW represents each document as a vector, with each element signifying the frequency of a particular word. The assigns weights to words based on their frequency in the document and across the entire corpus, emphasizing words’ importance while considering their prevalence across documents. Furthermore, word embeddings, like Word2Vec and GloVe, translate words into dense vector representations, capturing semantic relationships and contextual nuances.

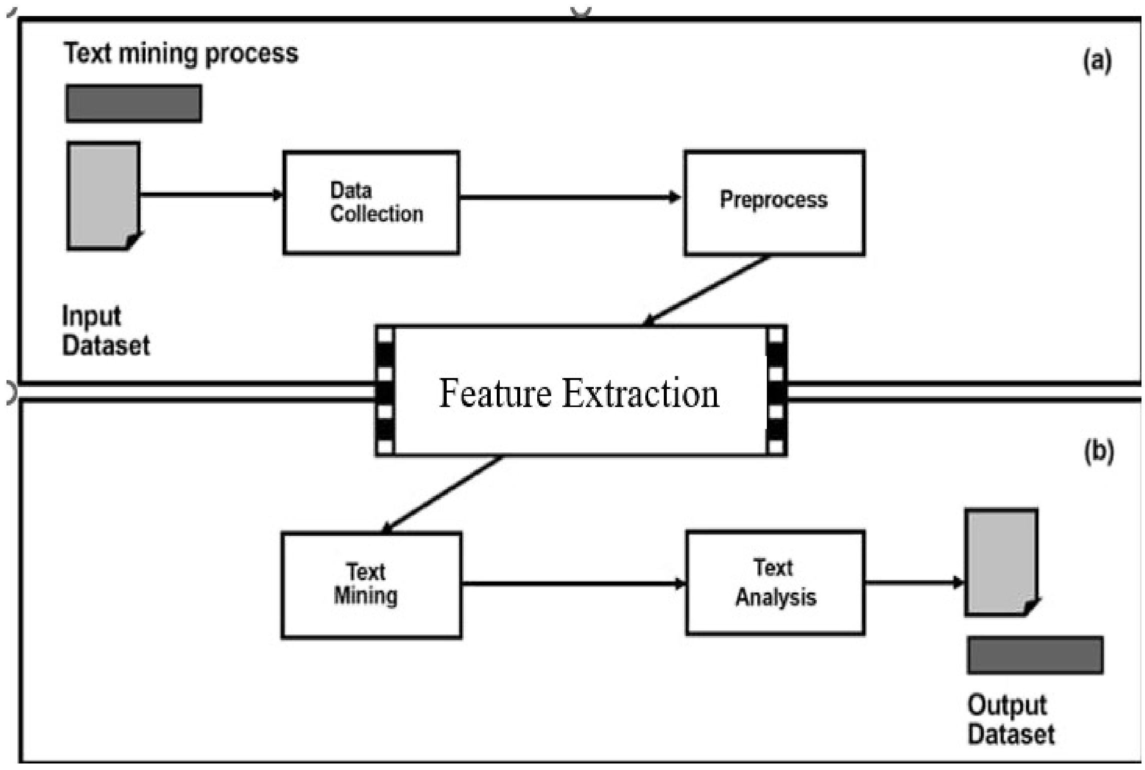


Figure 4.5: **Data Preprocessing**

In tandem with feature extraction, label encoding ensures the model comprehends and predicts emotions by translating emotion labels (e.g., sadness, joy) into numerical formats. Finally, the dataset is partitioned into training, validation, and test sets, enabling robust evaluation of the model's performance. These meticulously crafted data processing and feature extraction stages lay the groundwork for a sophisticated emotion classification system, poised to provide insightful analyses of textual emotions through advanced natural language processing techniques.

4.4.2 Model Creation

The utilization of Recurrent Neural Networks (RNNs), particularly the Long Short-Term Memory (LSTM) architecture, presents a formidable approach for emotion classification tasks. LSTMs are specifically engineered to address the limitations of traditional RNNs, such as the vanishing gradient problem, by incorporating memory cells equipped with gating mechanisms. These mechanisms allow LSTMs to retain relevant information over extended sequences, making them well-suited for analyzing textual data with long-term dependencies, including emotional expressions.

In the LSTM architecture, multiple LSTM layers are stacked together to form a deep neural network capable of learning complex patterns and representations from sequential data. Each LSTM layer consists of memory cells interconnected through gates, including input, forget, and output gates.

These gates regulate the flow of information, enabling the network to selectively retain or discard information based on its relevance to the current context. By processing text data through these layers, the LSTM architecture can effectively capture the nuanced temporal dynamics inherent in emotional expressions.

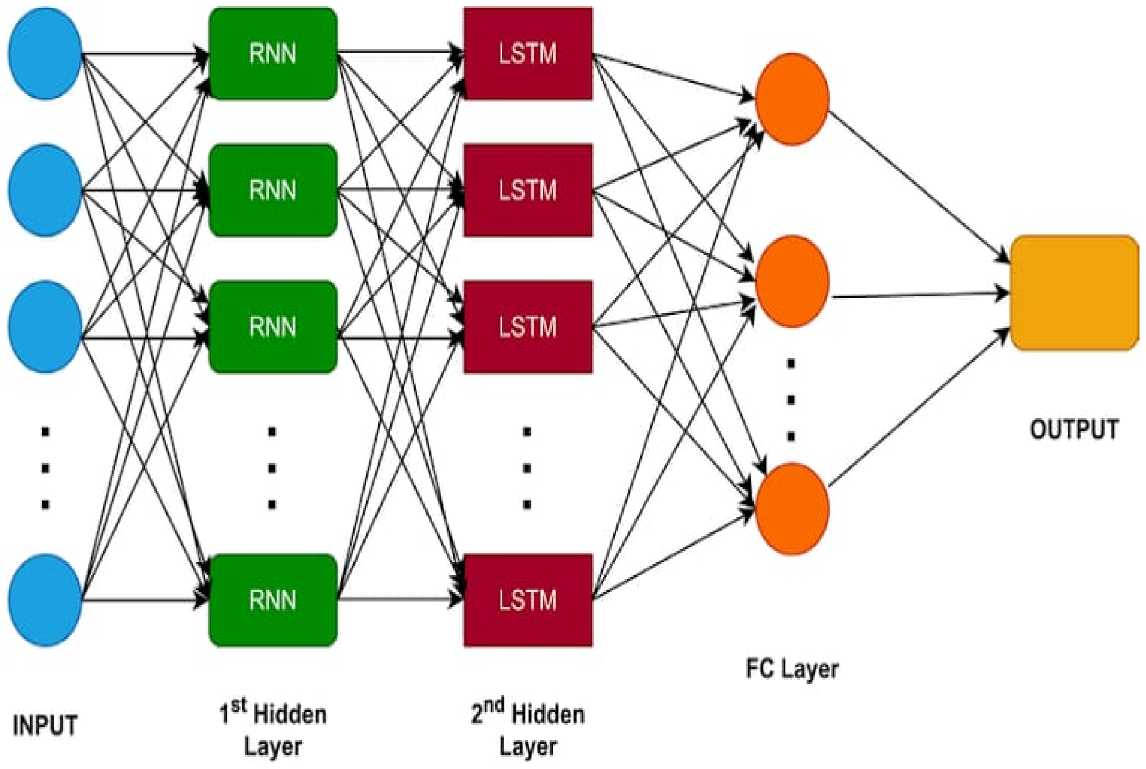


Figure 4.6: **Model Creation**

The LSTM architecture is augmented with additional components, such as dropout layers, to enhance its generalization capability and mitigate overfitting. Dropout layers randomly deactivate a fraction of neurons during training, forcing the model to learn more robust and generalized representations of the data. Additionally, the output layer of the LSTM network employs the softmax activation function to generate a probability distribution over predefined emotion classes, enabling the model to classify input text into distinct emotional categories with confidence. Through the integration of LSTM architecture, the proposed system achieves enhanced accuracy and robustness in emotion classification, empowering users to gain deeper insights into the emotional content of textual data.

4.4.3 Model Integration

To seamlessly integrate the LSTM model into the Flask framework, we first load the pre-trained model into the application environment. This model, constructed using the LSTM architecture, serves as the core component responsible for predicting the emotions conveyed in textual data. Loading the model ensures that it is readily available and operational within the Flask application, ready to process incoming text inputs and classify the associated emotions accurately.

A robust text preprocessing function named text preprocess. This function plays a crucial role in preparing the input text data for emotion classification. It performs fundamental preprocessing tasks such as tokenization, cleaning, and optional removal of stopwords. By applying these preprocessing steps to the input text, we ensure that the text is appropriately formatted and structured before being passed to the LSTM model for emotion classification, thereby enhancing the accuracy and reliability of the classification process.

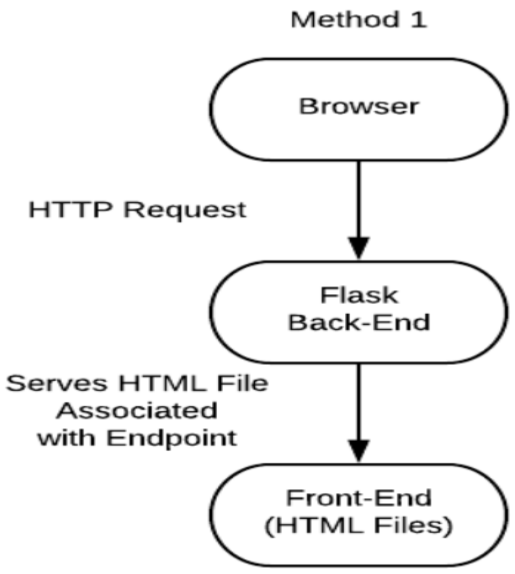


Figure 4.7: **Model Integration**

To create a prediction function named predict, which orchestrates the emotion prediction process using the loaded LSTM model. This function accepts preprocessed text as input, tokenizes it, converts it into sequences, pads the sequences to ensure uniform length, and subsequently feeds them into the LSTM model for emotion classification. Leveraging the predictive capabilities of the LSTM model, the predict function accurately predicts the emotions expressed in the text and returns the predicted emotions to the user. By seamlessly integrating this prediction function into the Flask framework, users can experience real-time emotion classification of their text inputs through the web interface with efficiency and effectiveness.

4.5 Steps to execute/run/implement the project

4.5.1 Python Installation

- **Install Python:** Download and install Python from the official website (https://www.python.org/).
- **Create Virtual Environment:** Use the venv module or conda environment to create a virtual environment for the project.
- **Install Libraries:** Use pip to install the required libraries. For example: pip install pandas nltk matplotlib seaborn numpy gensim tensorflow flask keras.
- **Download Dataset:** Obtain Dataset: Search for publicly available datasets containing text samples annotated with emotions. Websites like Kaggle (https://www.kaggle.com/) or UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/index.php) may have suitable datasets.

4.5.2 Training Model

- **Train Model:**Select a deep learning model architecture suitable for emotion classification, such as a bidirectional LSTM network.
- **Word Embeddings:**Generate word embeddings using pre-trained models like Word2Vec or GloVe to represent words as dense vectors.
- **Model Training:**Train the selected model using the preprocessed text data and corresponding emotion labels.
- **Choose Framework:**Select a web development framework such as Flask or Django for building the web application.

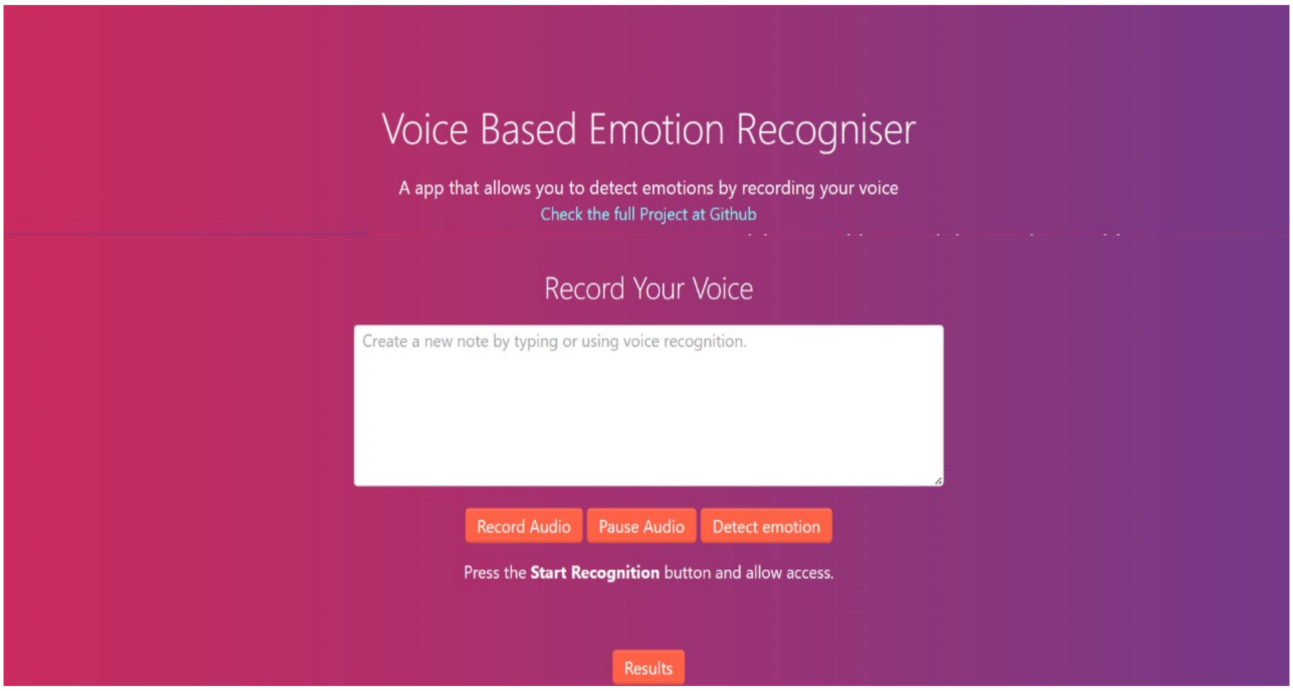
Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

The input for this implementation of the project consists of audio files. The file hierarchy if presented in the Figure.



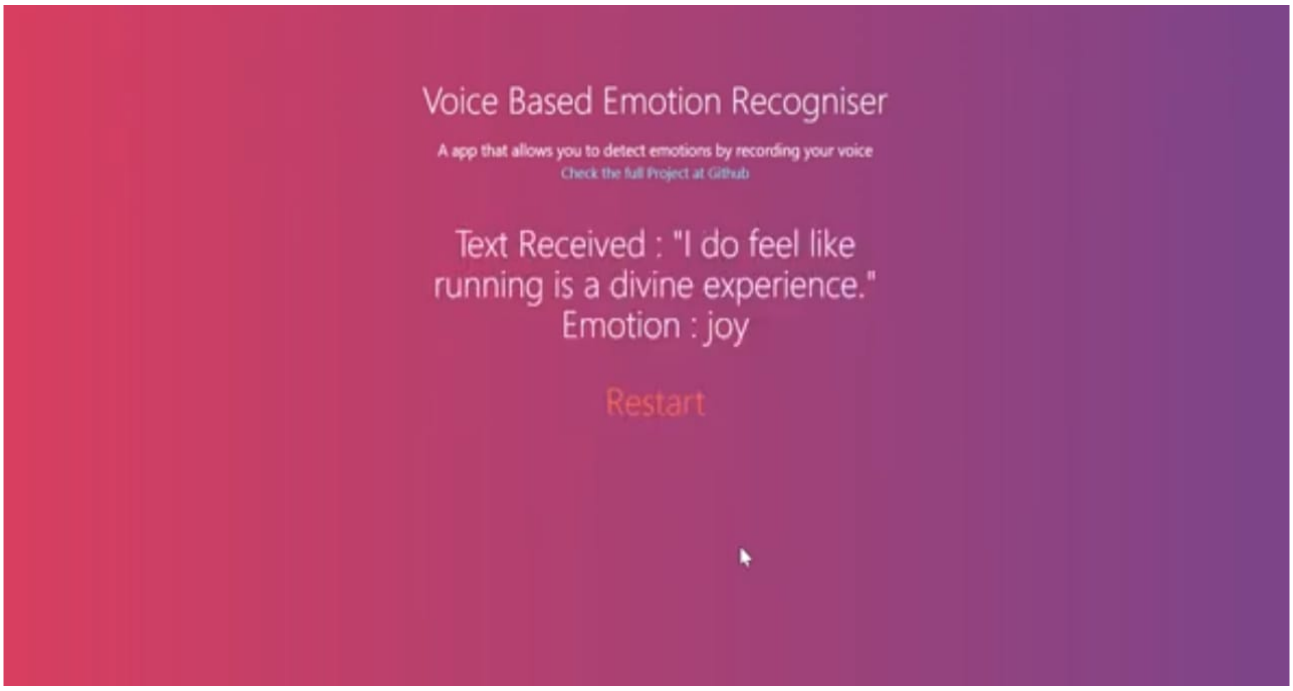
Voice Based Emotion Recogniser
A app that allows you to detect emotions by recording your voice
Check the full Project at Github
Record Your Voice
Create a new note by typing or using voice recognition.
Record Audio
Pause Audio
Detect emotion
Press the Start Recognition button and allow access.
Results

Figure 5.1: **Input Design**

In figure 5.1 shows that the speech data is the primary modality used for emotion recognition. It includes audio recordings of people speaking with different emotional states such as happiness, sadness, anger, and so on. The data can be collected from various sources, such as interviews, movies, or conversations, and should cover a diverse range of emotions.

5.1.2 Output Design

The output design of the implementation consists of the Emotion's name, the location where he is identified. The sample output is depicted in Figure.


Voice Based Emotion Recogniser

A app that allows you to detect emotions by recording your voice

Check the full Project at Github

Text Received : "I do feel like running is a divine experience."
Emotion : joy

Restart

Figure 5.2: **Output Design**

In figure 5.2 shows the accuracy of the speech emotion recognition with deep learning using fusion can be explain ability of the deep learning model to correctly to predict the emotional state based on there speech data and other modelities such as physiological signals and voices.

5.2 Testing

A crucial stage in the creation of any software system is testing. Strict testing procedures have been used to verify the system's overall performance, accuracy, and reliability. Various important segments make up the testing process.

- **Unit Testing:** The foundation of the Threat Detection System lies in its machine-learning models, each responsible for detecting specific types of network intrusions. Unit testing is conducted on these models individually, verifying that they produce accurate predictions and classifications. This ensures that each component operates as intended before integration.
- **Integration Testing:** When integrating machine learning algorithms with a web framework such as Flask, integration testing is critical to the system. It looks at how various parts work together to make sure everything communicates well.
- **System Testing:** A critical stage of the software development life cycle, system testing evaluates the functionality of the complete integrated system. compared to unit testing, which concentrates on individual modules or components, system testing confirms that all of the parts function as a

complete unit. This stage is crucial for determining if the program satisfies the requirements and performs as intended across a range of circumstances.

- The test cases considered for this project are listed below.

Table 5.1: **Test Cases Table**

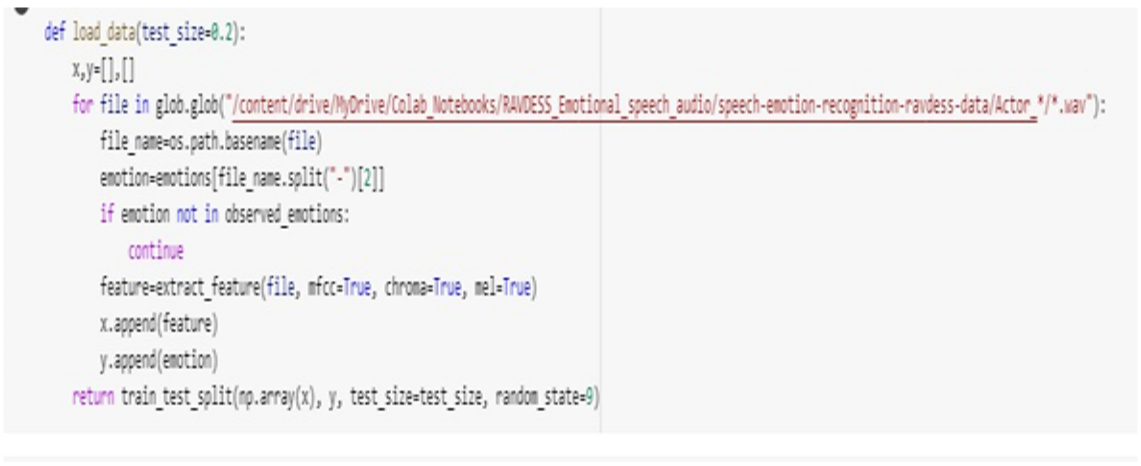
S. No	**Test Case**	**Provided Input**	**Expected Output**	**Actual Output**
1	Input validation with null values	Null values or no input is provided	Value Error	Value Error is displayed
2	Speech Analysis by Model	Providing the text input	RNN Model Predicted Output	RNN Model Predictions
3	Data analysis Integration	Graphs and Images	Frame Format	Frame Format

5.3 Types of Testing

5.3.1 Unit Testing

Testing the Machine Learning Models.

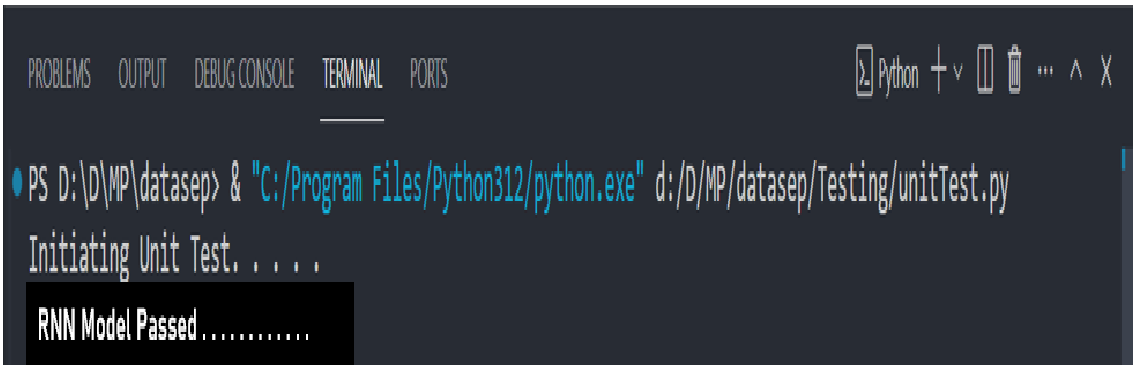
Input



```
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("/content/drive/MyDrive/Colab_Notebooks/RAVDESS_Emotional_speech_audio/speech-emotion-recognition-ravdess-data/Actor_*/*.wav"):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            continue
        feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

Figure 5.3: **Unit Test Input**

Test Result



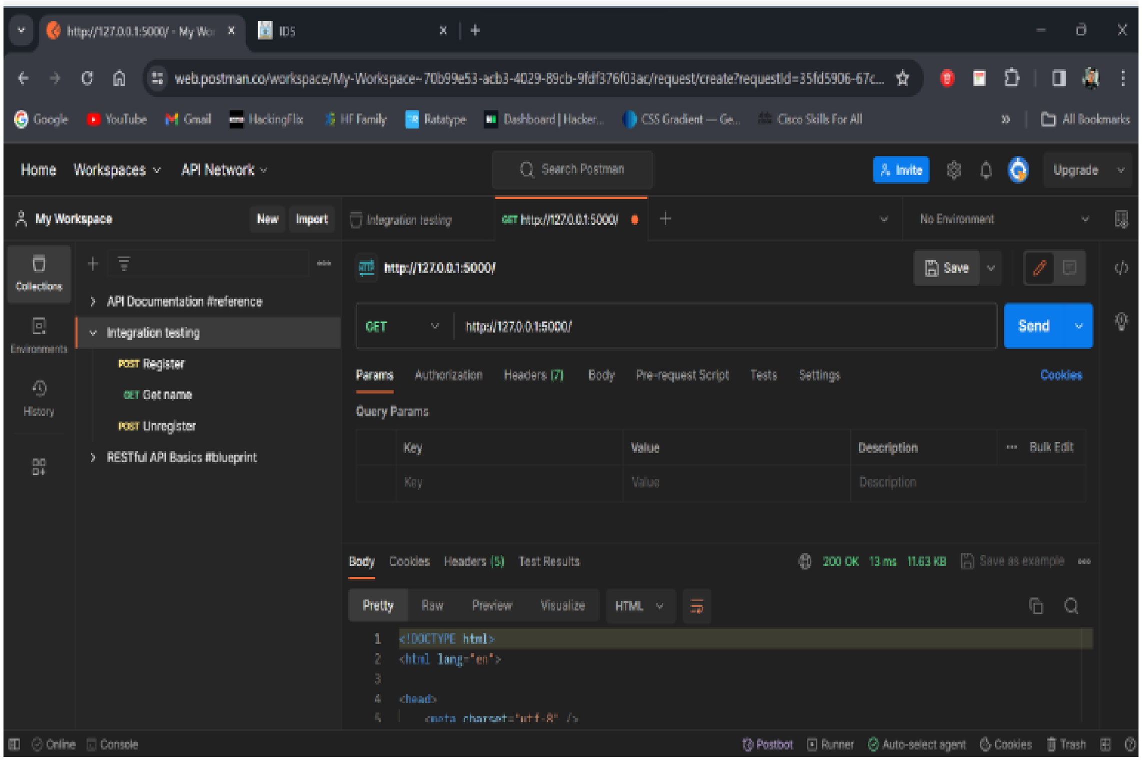
```
PS D:\D\MP\datasep> & "C:/Program Files/Python312/python.exe" d:/D/MP/datasep/Testing/unitTest.py
Initiating Unit Test. . . . .
RNN Model Passed............
```

Figure 5.4: **Unit Test Result**

5.3.2 Integration Testing

- The Integration test is carried out by the PostmanAPI.

Test Result



web.postman.co/workspace/My-Workspace~70b99e53-acb3-4029-89cb-9fdf376f03ac/request/create?requestId=35fd5906-67c...

GET http://127.0.0.1:5000/

200 OK 13 ms 11.63 KB

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
```

Figure 5.5: **Integration Test Result**

5.3.3 System Testing

The goal of system testing is to ensure that all components of a software application work together seamlessly to achieve the specified requirements and functionality. All the modules present in this project are tested together.

Input

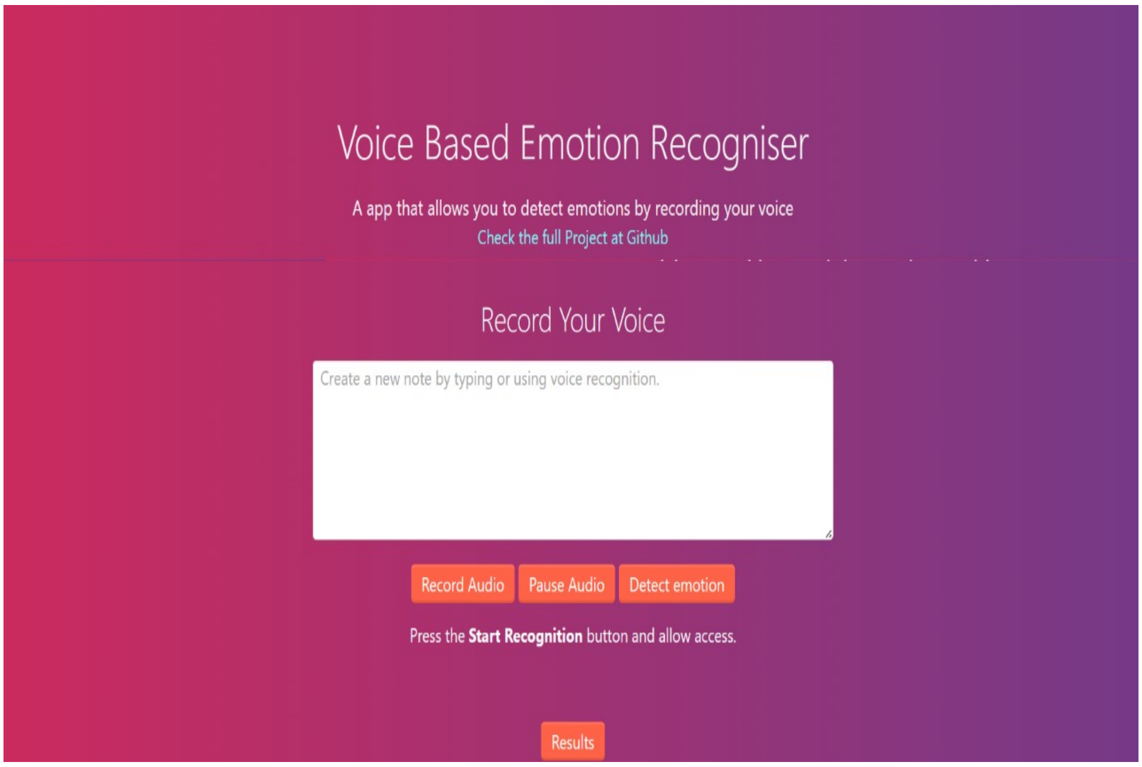
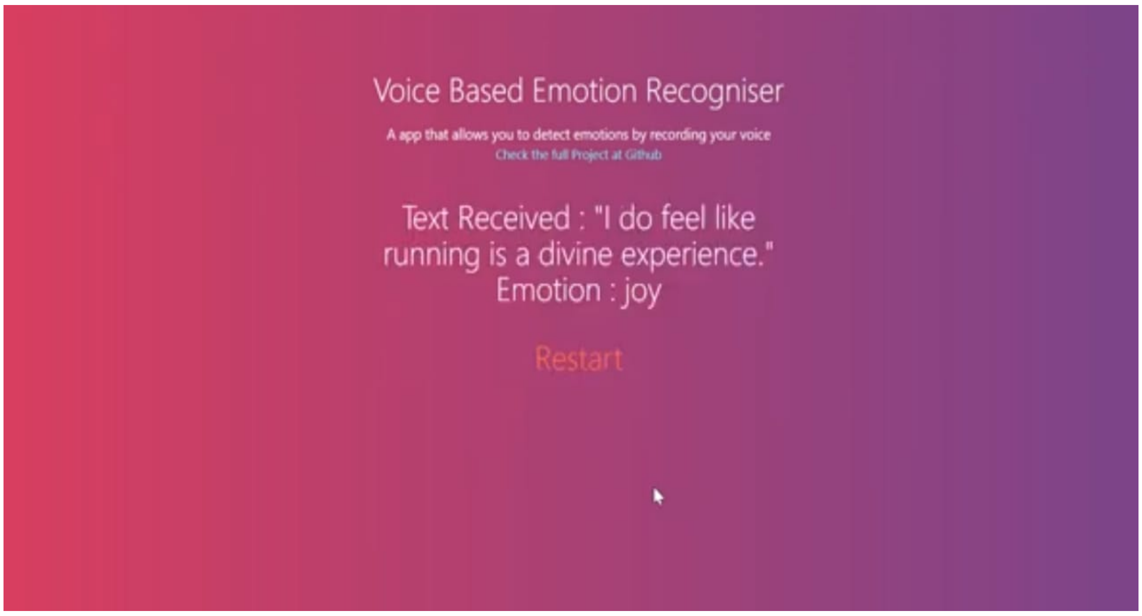

Voice Based Emotion Recogniser
A app that allows you to detect emotions by recording your voice
Check the full Project at Github
Record Your Voice
Create a new note by typing or using voice recognition.
Record Audio
Pause Audio
Detect emotion
Press the Start Recognition button and allow access.
Results

Figure 5.6: **Unit Test Result**

Test Result


Voice Based Emotion Recogniser
A app that allows you to detect emotions by recording your voice
Check the full Project at Github
Restart

Text Received : "I do feel like running is a divine experience."
Emotion : joy

Figure 5.7: **System Test Result**

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The efficiency of the proposed Emotion Classification Web Application is paramount in ensuring its effectiveness and user satisfaction. Firstly, accuracy stands as a cornerstone of its efficiency, as it directly impacts the reliability of emotion classification results. A highly accurate system provides users with confidence in the emotional insights derived from their textual inputs, facilitating informed decision-making and deeper understanding of communication nuances. Moreover, accuracy is crucial for applications in domains such as mental health assessment or sentiment analysis, where precise emotion classification is essential for drawing meaningful conclusions and providing appropriate interventions or recommendations.

The speed and responsiveness of the system significantly contribute to its efficiency. A fast and responsive application ensures a seamless user experience, allowing users to obtain emotion predictions swiftly without experiencing delays or lags. This is particularly important in real-time scenarios, such as social media monitoring or customer feedback analysis, where timely insights are crucial for timely responses or interventions. Additionally, a system's scalability and resource utilization play a vital role in its efficiency. A well-designed system should be able to handle increasing user traffic and data volumes gracefully, without sacrificing performance or incurring excessive resource costs. By efficiently utilizing computational resources and optimizing processing workflows, the system can maintain high throughput and responsiveness even under heavy loads, ensuring continued reliability and user satisfaction.

6.2 Comparison of Existing and Proposed System

The comparison between the existing system and the proposed Emotion Classification Web Application reveals significant advancements and improvements in several key aspects. In terms of accuracy, the existing system often relies on simplistic rule-based approaches or basic sentiment analysis techniques, which may result in limited accuracy and reliability. In contrast, the proposed system utilizes advanced NLP techniques and deep learning models, such as bidirectional LSTM networks, coupled with pre-trained word embeddings, to achieve higher accuracy and precision in emotion classification. By leveraging deep learning algorithms, the proposed system can capture subtle nuances in textual data and provide more accurate predictions of emotions expressed within.

Regarding speed and responsiveness, the existing system may suffer from performance bottlenecks or processing delays, especially when dealing with large volumes of data or complex text inputs. In contrast, the proposed web application offers real-time emotion classification capabilities, allowing users to obtain instant insights into the emotional content of their textual inputs. By leveraging efficient algorithms and optimized processing workflows, the proposed system delivers fast and responsive performance, ensuring a smooth user experience and timely results.

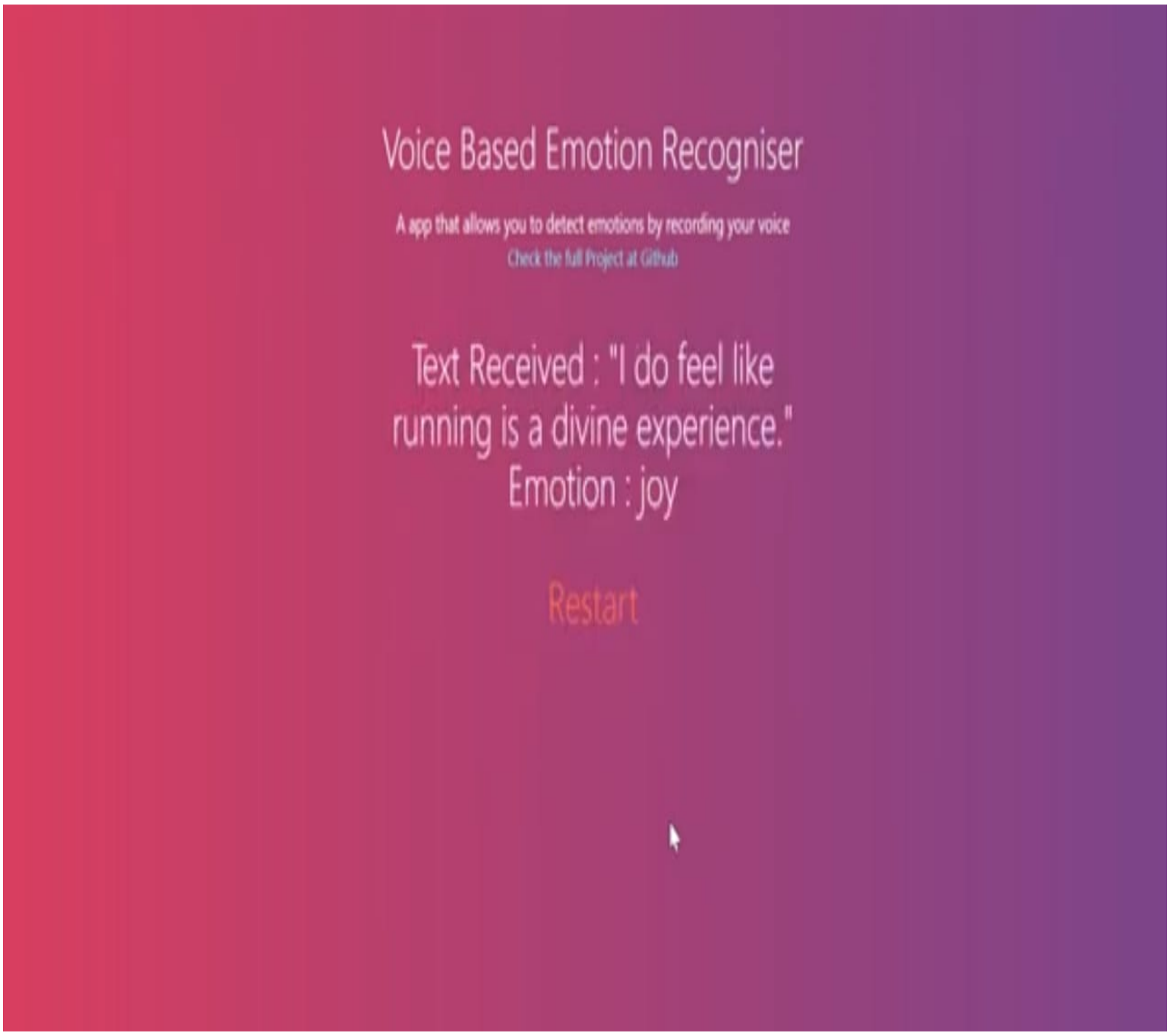
Furthermore, in terms of scalability and flexibility, the existing system may lack the capability to scale effectively to accommodate increasing user demands or data volumes. In contrast, the proposed web application is designed with scalability in mind, leveraging cloud-based infrastructure and modern web technologies to seamlessly handle growing user traffic and data inputs. Additionally, the proposed system offers greater flexibility and customization options, allowing users to tailor the application to their specific needs and preferences through customizable features and settings.

Overall, the comparison between the existing system and the proposed Emotion Classification Web Application highlights significant advancements in accuracy, speed, scalability, and flexibility. By leveraging state-of-the-art NLP techniques and deep learning models, the proposed system offers enhanced capabilities for analyzing and understanding emotions expressed in textual data, providing users with valuable insights and facilitating more meaningful interactions across various domains.

6.3 Sample Code

```
import pandas as pd
from nltk.corpus import stopwords
from flask import Flask, redirect, url_for, request, render_template, session
from tensorflow import keras
from nltk.tokenize import word_tokenize
import re
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import numpy as np
import pickle
from flask_session import Session

# Initialize Flask app
app = Flask(_name_)
SESSION_TYPE = 'filesystem'
app.config.from_object(_name_)
Session(app)

# Text preprocessing function
def text_preprocess(text, stop_words=False):
    # Define stopwords
    STOPWORDS = set(stopwords.words('english'))
    # Clean text from non-words
    text = re.sub(r'\W+', ' ', text).lower()
    # Tokenize the text
    tokens = word_tokenize(text.lower())
    if stop_words:
        # Remove stopwords
        tokens = [token for token in tokens if token not in STOPWORDS]
    return tokens

# Function to predict emotions
def predict(texts, model):
    # Load tokenizer
    with open('tokenizer.pickle', 'rb') as handle:
        tokenizer = pickle.load(handle)
    # Preprocess text
    texts_prepr = [text_preprocess(t) for t in texts]
    sequences = tokenizer.texts_to_sequences(texts_prepr)
    pad = pad_sequences(sequences, maxlen=35)
    # Make predictions
    predictions = model.predict(pad)
    labels = np.argmax(predictions, axis=1)
    # Map labels to emotions
    emotions_to_labels = {'anger': 0, 'love': 1, 'fear': 2, 'joy': 3, 'sadness': 4, 'surprise': 5}
    labels_to_emotions = {j: i for i, j in emotions_to_labels.items()}
    # Print results
    for i, lbl in enumerate(labels):
        if i == 0:
            return labels_to_emotions[lbl]

# Route for home page
@app.route('/', methods=['GET'])
def index():
    session['sentence'] = ""
    return render_template('index.html')

# Route for index page
@app.route('/index', methods=['GET'])
def index2():
    return render_template('index.html')

# Route for final result
@app.route('/finalresult', methods=['POST', "GET"])
def finalres():
    result = ""
    fungive = ['gg']
    fungive[0] = session['sentence']
    model = keras.models.load_model('model.h5')
    record = predict(fungive, model)
    session.pop('sentence', None)
    return render_template("finalresult.html", record=record, update=fungive[0])

# Route for recording
@app.route('/record', methods=['POST', "GET"])
def record():
    record = True
    transcript = ""
    if request.method == "POST":
        information = request.data
        text = information.decode("utf-8")
        session['sentence'] = text
        record = False
        return render_template("result.html", record=record)
    else:
        return render_template("result.html", record=record)

# Run the app
if _name_ == '_main_':
    app.run(debug=True)

write your code here
main code
```

Output


Voice Based Emotion Recogniser

A app that allows you to detect emotions by recording your voice

Check the full Project at Github

Text Received : "I do feel like running is a divine experience."

Emotion : joy

Restart

Figure 6.1: **Output for Emotional Speech Analysis**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The Emotion Classification Web Application presents a significant leap forward in the domain of emotion analysis from textual data, offering a substantial improvement over existing systems. Through the integration of advanced NLP techniques and deep learning models, the proposed system achieves heightened accuracy and precision in emotion classification. Moreover, its real-time capabilities ensure swift and responsive performance, enabling users to obtain instant insights into the emotional content of their textual inputs. Additionally, the system's scalability and flexibility make it well-suited for handling growing user demands and data volumes, while also offering customization options to meet diverse user needs and preferences.The accuracy of present model is more than 85%. Overall, the Emotion Classification Web Application represents a powerful tool for analyzing and understanding emotions expressed in textual data, with significant implications for various domains including mental health assessment, sentiment analysis, and customer feedback analysis.

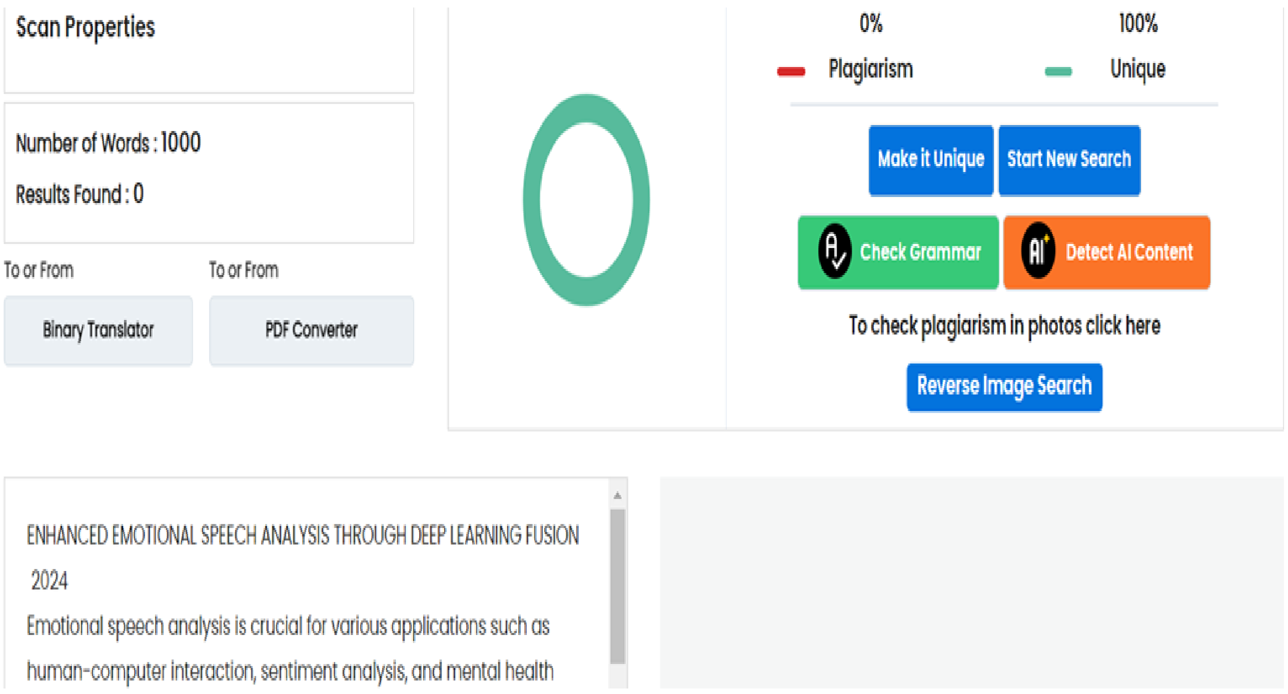
7.2 Future Enhancements

Several avenues for future enhancement of the Emotion Classification Web Application could be explored to further augment its capabilities and usability. One potential area for improvement is the incorporation of multimodal inputs, such as images or audio, to capture emotions expressed through different modalities. Additionally, enhancing the system's interpretability by providing explanations or visualizations of emotion classification results could improve user trust and understanding.

Furthermore, expanding the application's language support to include languages other than English would broaden its applicability and user base. Moreover, ongoing refinement of the deep learning models and continuous training on diverse datasets could further enhance the system's accuracy and generalization capabilities. Lastly, integrating feedback mechanisms to gather user input and preferences for iterative improvements would ensure the application remains relevant and effective in meeting user needs over time. Overall, these future enhancements hold the potential to elevate the Emotion Classification Web Application to new heights of functionality and utility, solidifying its position as a valuable tool for emotion analysis in various domains.

Chapter 8

PLAGIARISM REPORT


Scan Properties

Number of Words : 1000

Results Found : 0

To or From | To or From

Binary Translator | PDF Converter

0% 🟥 Plagiarism

100% 🟩 Unique

Make it Unique | Start New Search

Check Grammar | Detect AI Content

To check plagiarism in photos click here

Reverse Image Search

ENHANCED EMOTIONAL SPEECH ANALYSIS THROUGH DEEP LEARNING FUSION

2024

Emotional speech analysis is crucial for various applications such as human-computer interaction, sentiment analysis, and mental health

Figure 8.1: **Plagiarism Report**

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code

app.py

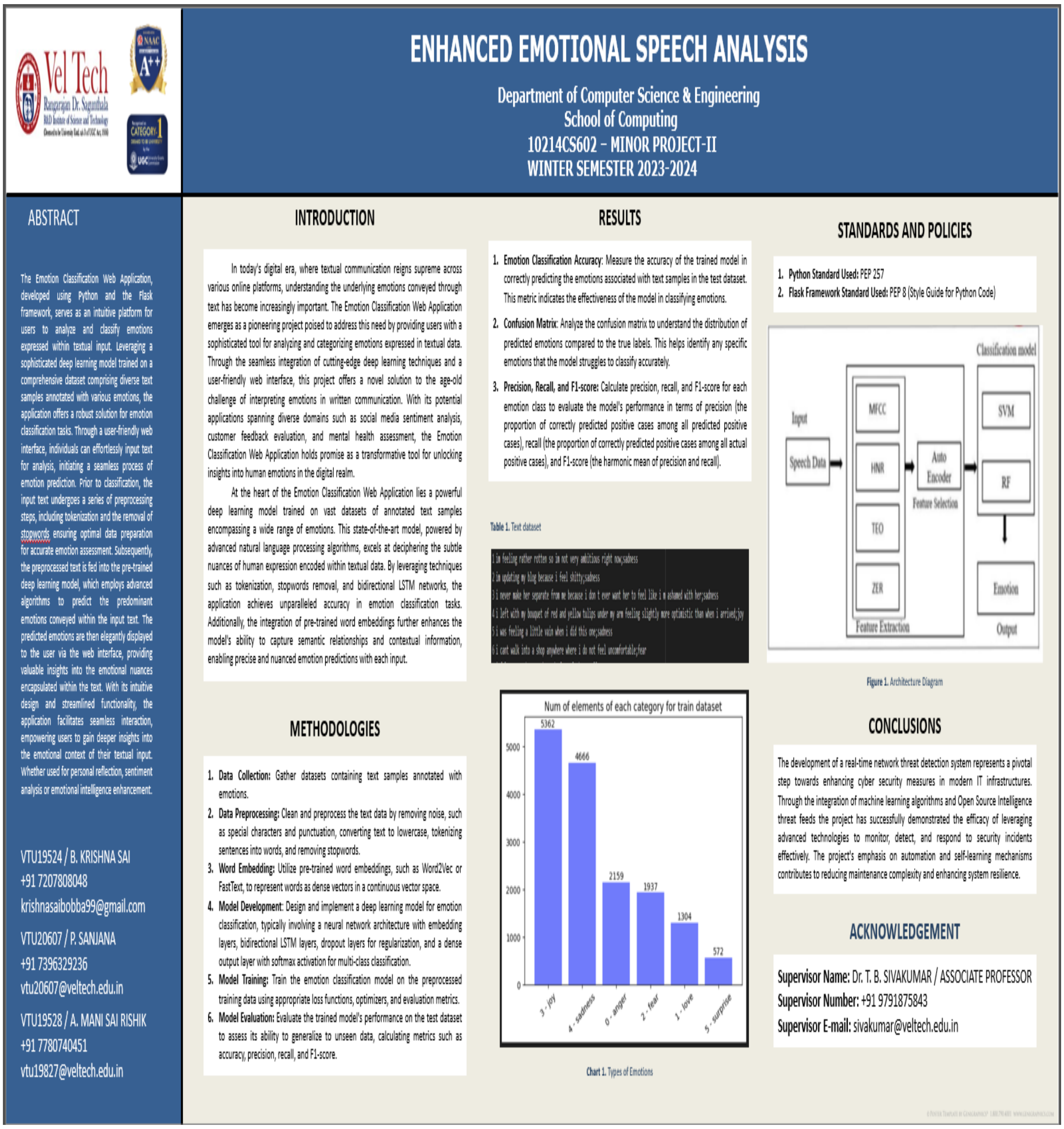
```python
import pandas as pd
from nltk.corpus import stopwords
from flask import Flask, redirect, url_for, request, render_template, session
from tensorflow import keras
from nltk.tokenize import word_tokenize
import re
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import numpy as np
import pickle
from flask_session import Session

# Initialize Flask app
app = Flask(__name__)
SESSION_TYPE = 'filesystem'
app.config.from_object(__name__)
Session(app)

# Text preprocessing function
def text_preprocess(text, stop_words=False):
    # Define stopwords
    STOPWORDS = set(stopwords.words('english'))
    # Clean text from non-words
    text = re.sub(r'\W+', ' ', text).lower()
    # Tokenize the text
    tokens = word_tokenize(text.lower())
    if stop_words:
        # Remove stopwords
        tokens = [token for token in tokens if token not in STOPWORDS]
    return tokens

# Function to predict emotions
def predict(texts, model):
    # Load tokenizer
    with open('tokenizer.pickle', 'rb') as handle:
        tokenizer = pickle.load(handle)
    # Preprocess text
    texts_prepr = [text_preprocess(t) for t in texts]
    sequences = tokenizer.texts_to_sequences(texts_prepr)
    pad = pad_sequences(sequences, maxlen=35)
    # Make predictions
    predictions = model.predict(pad)
    labels = np.argmax(predictions, axis=1)
    # Map labels to emotions
    emotions_to_labels = {'anger': 0, 'love': 1, 'fear': 2, 'joy': 3, 'sadness': 4, 'surprise': 5}
    labels_to_emotions = {j: i for i, j in emotions_to_labels.items()}
    # Print results
    for i, lbl in enumerate(labels):
        if i == 0:
            return labels_to_emotions[lbl]

# Route for home page
@app.route('/', methods=['GET'])
def index():
    session['sentence'] = ""
    return render_template('index.html')

# Route for index page
@app.route('/index', methods=['GET'])
def index2():
    return render_template('index.html')

# Route for final result
@app.route('/finalresult', methods=['POST', "GET"])
def finalres():
    result = ""
    fungive = ['gg']
    fungive[0] = session['sentence']
    model = keras.models.load_model('model.h5')
    record = predict(fungive, model)
    session.pop('sentence', None)
    return render_template("finalresult.html", record=record, update=fungive[0])

# Route for recording
@app.route('/record', methods=['POST', "GET"])
def record():
    record = True
    transcript = ""
    if request.method == "POST":
        information = request.data
        text = information.decode("utf-8")
        session['sentence'] = text
        record = False
        return render_template("result.html", record=record)
    else:
        return render_template("result.html", record=record)

# Run the app
if __name__ == '__main__':
    app.run(debug=True)

write your code here
```

Model creation

```
import re  # Regex
import pandas as pd  # Tables
import matplotlib.pyplot as plt  # Plots
import seaborn as sns  # Plots
import numpy as np  # Operations with arrays and matrices
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import nltk
import gensim.downloader as api
from gensim.models import Word2Vec, KeyedVectors
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from keras.models import load_model

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

# Read datasets
train = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train.txt', header=None, sep=';', names
    =['Lines', 'Emotions'], encoding='utf-8')
test = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/test.txt', header=None, sep=';', names=['
    Lines', 'Emotions'], encoding='utf-8')
validation = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/validation.txt', header=None, sep='
    ;', names=['Lines', 'Emotions'], encoding='utf-8')

# Add a column with encoded emotions
emotions_to_labels = {'anger': 0, 'love': 1, 'fear': 2, 'joy': 3, 'sadness': 4, 'surprise': 5}
train['Labels'] = train['Emotions'].replace(emotions_to_labels)
test['Labels'] = test['Emotions'].replace(emotions_to_labels)
validation['Labels'] = validation['Emotions'].replace(emotions_to_labels)

# Function to visualize labels distribution
def visualize_labels_distribution(df, title='the'):
    '''
    Accepts a dataframe with 'Emotions' column and dataset title (e.g., 'train')
    Creates a bar chart with the number of elements of each category
    Returns nothing
    '''
    # Create a pandas series with labels and their counts
    num_labels = df['Emotions'].value_counts()

    # Number of unique categories
    x_barchart = range(df['Emotions'].nunique())
    # List of labels
    x_barchart_labels = [str(emotions_to_labels[emotion]) + ' - ' + emotion for emotion in list(
        num_labels.index)]
    # List of counts
    y_barchart = list(num_labels.values)

    # Create bar chart
    plt.figure(figsize=(6, 5))
    plt.bar(x_barchart, y_barchart, color='#707bfb')

    # Add number of elements for each category on plot as text
    for index, data in enumerate(y_barchart):
        plt.text(x=index, y=data + max(y_barchart) / 100, s='{}'.format(data), fontdict=dict(
            fontsize=10), ha='center')

    plt.xticks(x_barchart, x_barchart_labels, rotation=40)
    plt.title('Num of elements of each category for {} dataset'.format(title))
    plt.tight_layout()
    print('There are {} records in the dataset.\n'.format(len(df.index)))
    plt.show()

# Visualize labels distribution for each dataset
visualize_labels_distribution(train, 'train')
visualize_labels_distribution(test, 'test')
visualize_labels_distribution(validation, 'val')

# Text preprocessing function
def text_preprocess(text, stop_words=False):
    '''
    Accepts text (a single string) and a parameter of preprocessing
    Returns preprocessed text
    '''
    STOPWORDS = set(stopwords.words('english'))

    # Clean text from non-words
    text = re.sub(r'\W+', ' ', text).lower()
    # Tokenize the text
    tokens = word_tokenize(text)

    if stop_words:
        # Delete stop_words
        tokens = [token for token in tokens if token not in STOPWORDS]

    return tokens

# Display data before and after preprocessing
print('Before:')
print(train.head())

x_train = [text_preprocess(t, stop_words=True) for t in train['Lines']]
y_train = train['Labels'].values

print('\nAfter:')
for line_and_label in list(zip(x_train[:5], y_train[:5])):
    print(line_and_label)

# Process test and validation data
x_test = [text_preprocess(t, stop_words=True) for t in test['Lines']]
y_test = test['Labels'].values
x_validation = [text_preprocess(t, stop_words=True) for t in validation['Lines']]
y_validation = validation['Labels'].values

# Load pre-trained models
model_wiki = api.load('fasttext-wiki-news-subwords-300')
model_w2v = Word2Vec(x_train + x_test + x_validation, min_count=2, vector_size=300).wv

DICT_SIZE = 15000

# Create a dictionary with most used words
tokenizer = Tokenizer(num_words=DICT_SIZE)
total = x_train + x_test + x_validation
tokenizer.fit_on_texts(total)

# Determine maximum sentence length
x_train_max_len = max([len(i) for i in x_train])
x_test_max_len = max([len(i) for i in x_test])
x_validation_max_len = max([len(i) for i in x_validation])
MAX_LEN = max(x_train_max_len, x_test_max_len, x_validation_max_len)

# Replace words with their indexes and pad indexes
X_train = tokenizer.texts_to_sequences(x_train)
X_train_pad = pad_sequences(X_train, maxlen=MAX_LEN)
X_test = tokenizer.texts_to_sequences(x_test)
X_test_pad = pad_sequences(X_test, maxlen=MAX_LEN)
X_val = tokenizer.texts_to_sequences(x_validation)
X_val_pad = pad_sequences(X_val, maxlen=MAX_LEN)

# Function to create weight matrix
def create_weight_matrix(model, second_model=None, tokenizer=None, DICT_SIZE=None):
    if tokenizer is None or DICT_SIZE is None:
```

```python
        raise ValueError("Please provide tokenizer and DICT_SIZE parameters.")

    vector_size = model.vector_size
    w_matrix = np.zeros((DICT_SIZE, vector_size))
    skipped_words = []

    for word, index in tokenizer.word_index.items():
        if index < DICT_SIZE:
            if word in model:
                if second_model is not None and word in second_model:
                    w_matrix[index] = second_model[word][:vector_size]  # Resize vector if needed
                else:
                    w_matrix[index] = model[word][:vector_size]  # Resize vector if needed
            else:
                if second_model is not None and word in second_model:
                    w_matrix[index] = second_model[word][:vector_size]  # Resize vector if needed
                else:
                    skipped_words.append(word)
    print(f'{len(skipped_words)} words were skipped. Some of them:')
    print(skipped_words[:50])
    return w_matrix

# Create weight matrix
weight_matrix = create_weight_matrix(model_wiki, second_model=model_w2v, tokenizer=tokenizer,
    DICT_SIZE=DICT_SIZE)

# Initialize early stopping
stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)

# Initialize sequential model
model = Sequential()
model.add(Embedding(input_dim=DICT_SIZE, output_dim=weight_matrix.shape[1], input_length=X_train_pad
    .shape[1], weights=[weight_matrix], trainable=False))
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(128, return_sequences=False)))
model.add(Dense(6, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics='accuracy')

# Train the model
history = model.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_validation), batch_size=8,
    epochs=100, callbacks=stop)

# Save the model
model.save('/content/drive/MyDrive/Colab Notebooks/model.h5')
```

9.2 Poster Presentation



ENHANCED EMOTIONAL SPEECH ANALYSIS

Department of Computer Science & Engineering
School of Computing
10214CS602 – MINOR PROJECT-II
WINTER SEMESTER 2023-2024

ABSTRACT

The Emotion Classification Web Application, developed using Python and the Flask framework, serves as an intuitive platform for users to analyze and classify emotions expressed within textual input. Leveraging a sophisticated deep learning model trained on a comprehensive dataset comprising diverse text samples annotated with various emotions, the application offers a robust solution for emotion classification tasks. Through a user-friendly web interface, individuals can effortlessly input text for analysis, initiating a seamless process of emotion prediction. Prior to classification, the input text undergoes a series of preprocessing steps, including tokenization and the removal of stopwords ensuring optimal data preparation for accurate emotion assessment. Subsequently, the preprocessed text is fed into the pre-trained deep learning model, which employs advanced algorithms to predict the predominant emotions conveyed within the input text. The predicted emotions are then elegantly displayed to the user via the web interface, providing valuable insights into the emotional nuances encapsulated within the text. With its intuitive design and streamlined functionality, the application facilitates seamless interaction, empowering users to gain deeper insights into the emotional context of their textual input. Whether used for personal reflection, sentiment analysis or emotional intelligence enhancement.

VTU19524 / B. KRISHNA SAI
+91 7207808048
krishnasaibobba99@gmail.com

VTU20607 / P. SANJANA
+91 7396329236
vtu20607@veltech.edu.in

VTU19528 / A. MANI SAI RISHIK
+91 7780740451
vtu19827@veltech.edu.in

INTRODUCTION

In today's digital era, where textual communication reigns supreme across various online platforms, understanding the underlying emotions conveyed through text has become increasingly important. The Emotion Classification Web Application emerges as a pioneering project poised to address this need by providing users with a sophisticated tool for analyzing and categorizing emotions expressed in textual data. Through the seamless integration of cutting-edge deep learning techniques and a user-friendly web interface, this project offers a novel solution to the age-old challenge of interpreting emotions in written communication. With its potential applications spanning diverse domains such as social media sentiment analysis, customer feedback evaluation, and mental health assessment, the Emotion Classification Web Application holds promise as a transformative tool for unlocking insights into human emotions in the digital realm.

At the heart of the Emotion Classification Web Application lies a powerful deep learning model trained on vast datasets of annotated text samples encompassing a wide range of emotions. This state-of-the-art model, powered by advanced natural language processing algorithms, excels at deciphering the subtle nuances of human expression encoded within textual data. By leveraging techniques such as tokenization, stopwords removal, and bidirectional LSTM networks, the application achieves unparalleled accuracy in emotion classification tasks. Additionally, the integration of pre-trained word embeddings further enhances the model's ability to capture semantic relationships and contextual information, enabling precise and nuanced emotion predictions with each input.

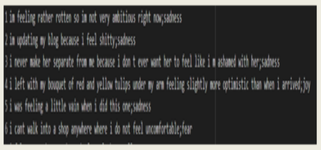
METHODOLOGIES

1. **Data Collection:** Gather datasets containing text samples annotated with emotions.
2. **Data Preprocessing:** Clean and preprocess the text data by removing noise, such as special characters and punctuation, converting text to lowercase, tokenizing sentences into words, and removing stopwords.
3. **Word Embedding:** Utilize pre-trained word embeddings, such as Word2Vec or FastText, to represent words as dense vectors in a continuous vector space.
4. **Model Development**: Design and implement a deep learning model for emotion classification, typically involving a neural network architecture with embedding layers, bidirectional LSTM layers, dropout layers for regularization, and a dense output layer with softmax activation for multi-class classification.
5. **Model Training:** Train the emotion classification model on the preprocessed training data using appropriate loss functions, optimizers, and evaluation metrics.
6. **Model Evaluation:** Evaluate the trained model's performance on the test dataset to assess its ability to generalize to unseen data, calculating metrics such as accuracy, precision, recall, and F1-score.

RESULTS

1. **Emotion Classification Accuracy**: Measure the accuracy of the trained model in correctly predicting the emotions associated with text samples in the test dataset. This metric indicates the effectiveness of the model in classifying emotions.
2. **Confusion Matrix**: Analyze the confusion matrix to understand the distribution of predicted emotions compared to the true labels. This helps identify any specific emotions that the model struggles to classify accurately.
3. **Precision, Recall, and F1-score:** Calculate precision, recall, and F1-score for each emotion class to evaluate the model's performance in terms of precision (the proportion of correctly predicted positive cases among all predicted positive cases), recall (the proportion of correctly predicted positive cases among all actual positive cases), and F1-score (the harmonic mean of precision and recall).

Table 1. Text dataset



```
1 im feeling rather rotten so im not very ambitious right now;sadness
2 im updating my blog because i feel shitty;sadness
3 i never make her separate from me because i don t ever want her to feel like i m ashamed with her;sadness
4 i left with my bouquet of red and yellow tulips under my arm feeling slightly more optimistic than when i arrived;joy
5 i was feeling a little vain when i did this one;sadness
6 i cant walk into a shop anywhere where i do not feel uncomfortable;fear
```

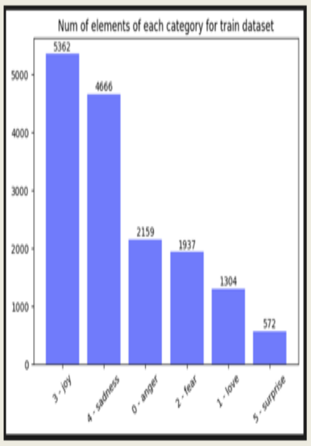


Chart 1. Types of Emotions

STANDARDS AND POLICIES

1. **Python Standard Used:** PEP 257
2. **Flask Framework Standard Used:** PEP 8 (Style Guide for Python Code)

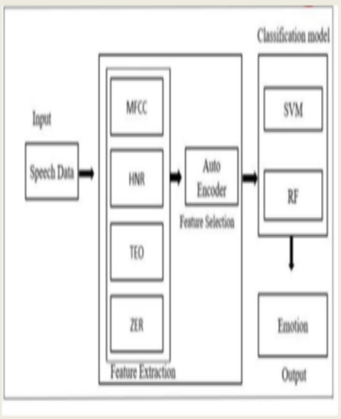


Figure 1. Architecture Diagram

CONCLUSIONS

The development of a real-time network threat detection system represents a pivotal step towards enhancing cyber security measures in modern IT infrastructures. Through the integration of machine learning algorithms and Open Source Intelligence threat feeds the project has successfully demonstrated the efficacy of leveraging advanced technologies to monitor, detect, and respond to security incidents effectively. The project's emphasis on automation and self-learning mechanisms contributes to reducing maintenance complexity and enhancing system resilience.

ACKNOWLEDGEMENT

Supervisor Name: Dr. T. B. SIVAKUMAR / ASSOCIATE PROFESSOR
Supervisor Number: +91 9791875843
Supervisor E-mail: sivakumar@veltech.edu.in

Figure 9.1: **Poster Presentation**

References

[1] Farashi S, Bashirian S, Jenabi E, Razjouyan K. Effectiveness of virtual reality and computerized training programs for enhancing emotion recognition in people with autism spectrum disorder: a systematic review and meta-analysis. International Journal of Developmental Disabilities. 2024 Jan 2;70(1):110-26.

[2] Nossier, S. A., Wall, J., Moniri, M., Glackin, C., Cannings, N. (2024). An experimental analysis of deep learning architectures for supervised speech enhancement. Electronics, 10(1), 17.

[3] Peng, Ziqiao, et al. "Emotalk: Speech-driven emotional disentanglement for 3d face animation." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

[4] Zhang, X., Xiao, H. (2023). Enhancing speech emotion recognition with the Improved Weighted Average Support Vector method. Biomedical Signal Processing and Control, 93, 106140.

[5] Lu, Cheng, Wenming Zheng, Hailun Lian, Yuan Zong, Chuangao Tang, Sunan Li, and Yan Zhao. "Speech emotion recognition via an attentive time–frequency neural network." IEEE Transactions on Computational Social Systems (2022).

[6] Mehrish, Ambuj, Navonil Majumder, Rishabh Bharadwaj, Rada Mihalcea, and Soujanya Poria. "A review of deep learning techniques for speech processing." Information Fusion (2022): 101869.

[7] Bhangale, Kishor Barasu, and Mohanaprasad Kothandaraman. "Survey of deep learning paradigms for speech processing." Wireless Personal Communications 125, no. 2 (2022): 1913-1949.

[8] Gao, Ruohan, and Kristen Grauman. "Visualvoice: Audio-visual speech separation with cross-modal consistency." In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15490-15500. IEEE, 2021.

[9] Fahad MS, Ranjan A, Yadav J, Deepak A. A survey of speech emotion recognition in natural environment. Digital signal processing. 2021 Mar 1;110:102951.

[10] Dangol R, Alsadoon A, Prasad PW, Seher I, Alsadoon OH. Speech emotion recognition Using-Convolutional neural network and long-short TermMemory. Multimedia Tools and Applications. 2020 Nov;79(43):32917-34.

[11] Lv, S., Hu, Y., Zhang, S., Xie, L. (2021). Dccrn+: Channel-wise subband dccrn with snr estimation for speech enhancement. arXiv preprint arXiv:2106.08672.

[12] Triantafyllopoulos, A., Keren, G., Wagner, J., Steiner, I., Schuller, B. (2019). Towards robust speech emotion recognition using deep residual networks for speech enhancement.

[13] Lalitha, S., Tripathi, S., Gupta, D. (2019). Enhanced speech emotion detection using deep neural networks. International Journal of Speech Technology, 22, 497-510.

width=!,height=!,page=-