Capturing Emotions: The Art of Speech Recognition and Sentiment Analysis

Krishna Chaitanya Pulipati
Data science and Analytics
Georgia State University
Atlanta,Georgia
kpulipati2@student.gsu.edu

***Abstract*—** The objective of this project is to explore the potential of using the MELD dataset to develop machine learning models for accurately detecting customer emotions in call centers and chatbots, with the aim of improving customer satisfaction and increasing sales.

The focus will be on understanding the various techniques and algorithms used for emotion detection and sentiment analysis, as well as exploring the challenges involved in applying these technologies in real-worldscenarios.

Keywords—MELD dataset

Dataset link- https://www.kaggle.com/datasets/zaber666/meld-dataset

I. INTRODUCTION (*HEADING 1*)

Sentiment analysis has become an increasingly important tool for businesses and organizations to understand customer feedback and improve their products and services. The objective is to analyze the emotions expressed by characters in the popular **TV show FRIENDS**, with the aim of identifying patterns and insights that can be used to target specific demographics in marketing campaigns. The goal is to develop effective and reliable models for emotion detection that can be applied in a wide range of business contexts and to explore new ways in which emotion detection can be used to enhance customer engagement and understanding. For example, if a large number of customers express frustration or dissatisfaction during calls in call centres, the company can take steps to address these issues and improve customer satisfaction.

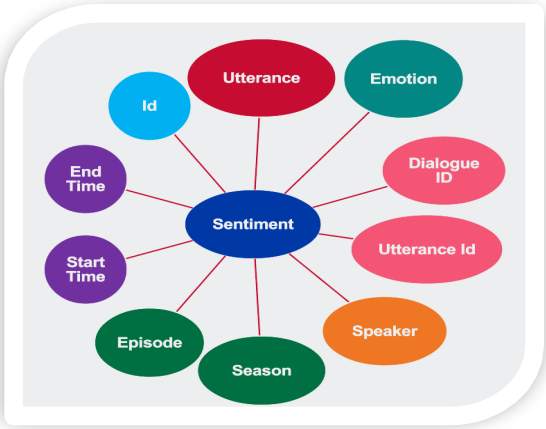
II. DATA DESCRIPTION:

The MELD dataset contains more than **1400 dialogues from which 13000 utterances** extracted or deduced from the popular TV series FRIENDS. Each utterance in dialogue has been labeled with one of seven emotions: **Anger, Disgust, Sadness, Joy, Neutral, Surprise, or Fear**. This means that each line of dialogue uttered by a character in the TV show has been classified according to the emotion expressed by the character.

Additionally, the dataset provides sentiment annotations for each utterance, which include a positive, negative, or neutral sentiment. This means each line of dialogue has also been labeled according to overall sentiment expressed by the character in the line. The dataset contains several columns of information for each utterance including:

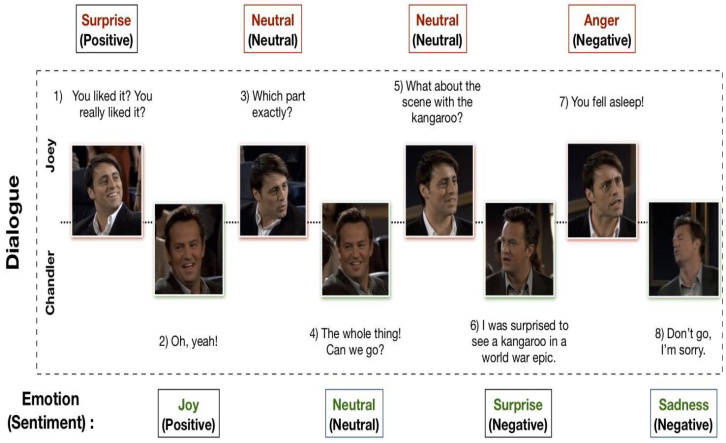
Column Name	**Description**
Utterance	The text of the utterance spoken by a character in the TV series
Speaker	The name of the character who spoke the utterance
Emotion	The emotion expressed in the utterance, which can be one of Anger, Disgust, Sadness, Joy, Neutral, Surprise, or Fear
Sentiment	The sentiment expressed in the utterance, which can be one of Positive, Negative, or Neutral
Dialogue_ID	The ID of the dialogue to which the utterance belongs
Utterance_ID	The ID of the utterance within the dialogue
Season	The season in which the dialogue appears
Episode	The episode in which the dialogue appears
StartTime	The start time of the utterance within the episode
EndTime	The end time of the utterance within the episode

This information allows for detailed analysis of the emotions and sentiments expressed by the characters throughout the TV series and can be used for a wide range of applications, including sentiment analysis, emotion detection, and targeted marketing.

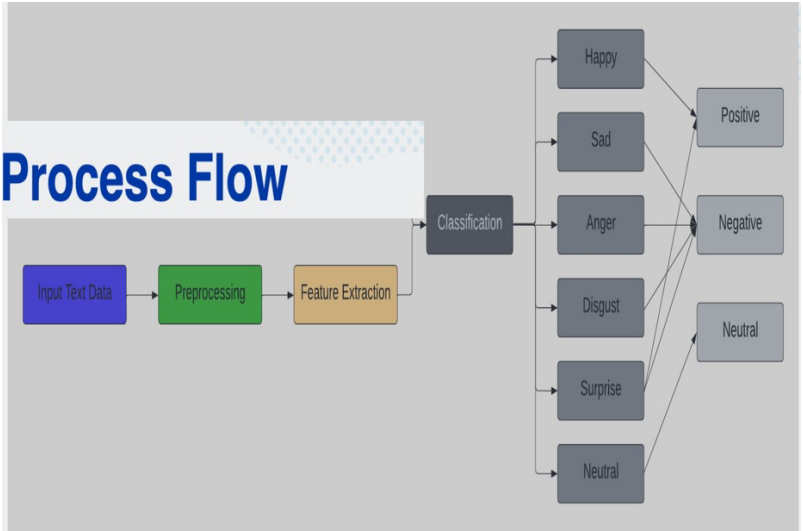


In the above dataset, we can comprehend the **sentiment** expressed in each utterance is the target variable, while the remaining columns (SrNo, Utterance, Speaker, Emotion, Dialogue_ID, Utterance_ID, Season, Episode, StartTime, and EndTime) represent independent variables or features.

Example Dialogue & Utterances :



III. PROCESS AND WORKFLOW OF THE PROJECT



The workflow for emotion and sentiment classification using both audio and text input data can be broken down into the following steps:

7.. **Data Collection and Preprocessing:** The first step involves collecting both the audio and text input data from the MELD dataset. The collected data is then preprocessed to remove any noise, irrelevant information, or inconsistencies in the data.

7.. **Feature Extraction from Audio Data:** In this step, features are extracted from the audio input data using techniques such as Mel-Frequency Cepstral Coefficients (MFCCs), which capture the spectral characteristics of the audio signals. These features are used to train the emotion classification model.

7.. **Feature Extraction from Text Data:** In this step, features are extracted from the text input data using techniques such as word embeddings or bag-of-words models. These features are used to train the sentiment classification model.

7.. **Training Classification Models**: In this step, separate classification models are trained on the audio and text data using supervised learning algorithms such as Support Vector Machines (SVMs), Random Forests, or Neural Networks. The models are trained to classify the emotions expressed in the audio data and the sentiment expressed in the text data.

7.. **Testing and Validation:** Once the classification models are trained, they are tested on a separate set of validation data to measure their accuracy and effectiveness. This step helps to identify any issues with the models and fine-tune the hyperparameters of the classification algorithms.

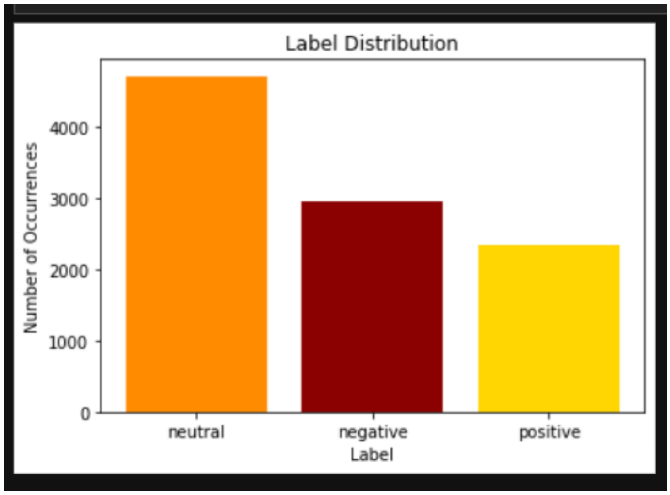
7.. **Integration of Audio and Text Models:** In the

final step, the classification models for audio and text data are integrated to provide a combined output that predicts both the emotion expressed in the audio data and the sentiment expressed in the text data.

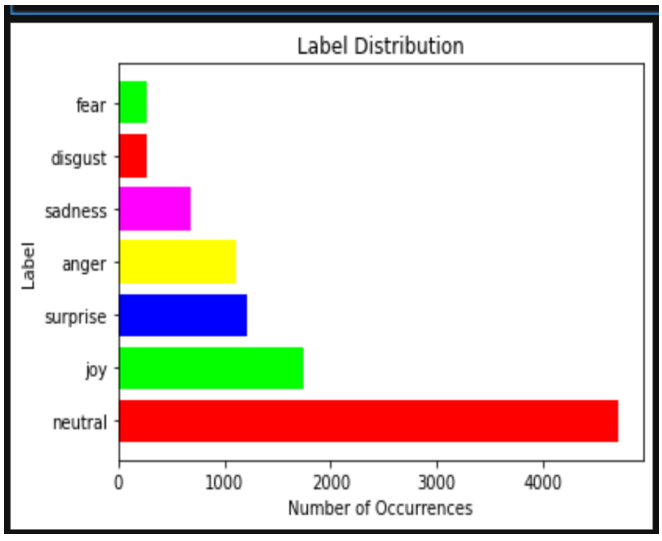
7.. **Overall,** this workflow involves the use of machine learning techniques to extract features from audio and text data, train separate classification models, and integrate them to provide a combined output for emotion and sentiment classification.

IV. DATA INSIGHTS:

Label Distribution:



Sentiment Plot

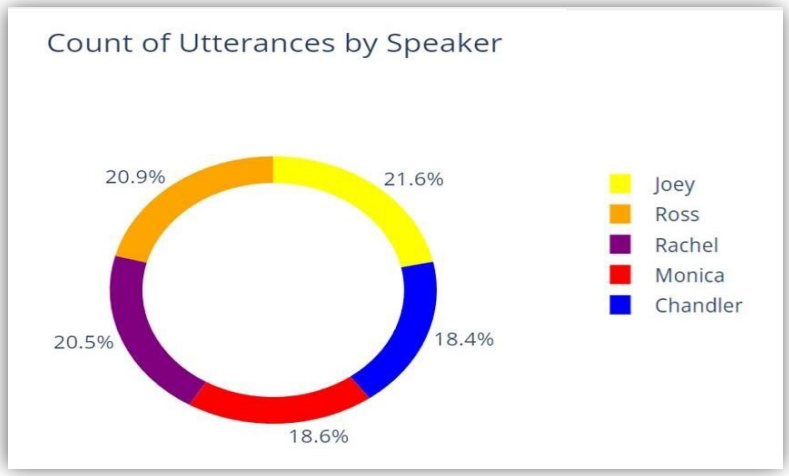


Emotion Analysis plot

From the Sentiment plot it can be clearly observed that the sentiment Neutral has the majority class and the sentiment positive has the minority class.

From the Emotion Analysis plot it can be deduced that the Neutral emotion is the majority class and and the emotion Fear is the minority class.

The number of occurrences of neutral words is the highest and positive words are lowest in both the Sentiment and Emotion Label Distribution.



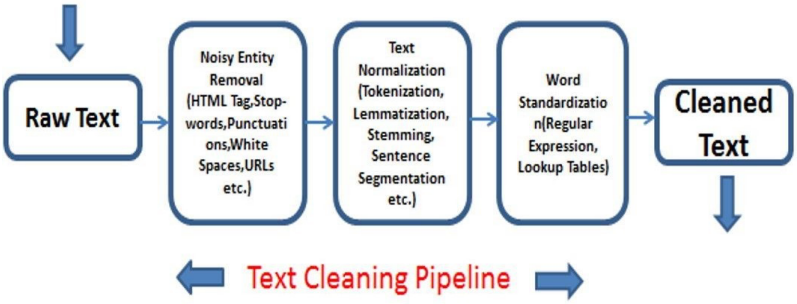
Speaker Utterance Analysis:

- Character Joey spoke **most** utterances.
- Character Monica spoke **least** utterances in the above dataset.

Word Cloud:

The word cloud represents that "Oh", “Yeah", "Well", “Know" and "okay" were more repeated words.

V. DATA PRE-PROCESSING FOR TEXT DATA:



The Workflow for the Data Pre-processing can be broken down into following Steps:

1. **Removing Punctuation:** Punctuation marks such as commas, periods, question marks, and exclamation marks can interfere with the text analysis, so it's essential to remove them.
2. **Lowercasing:** To ensure consistency in the text data, all text should be converted to lowercase.
3. **Removing Stop Words:** Stop words such as "the", "and," & "a" are common words that do not carry any significant meaning in the text, so it's recommended to remove them.
4. **Tokenization:** The process of dividing text data into smaller units such as words or phrases is called tokenization. It's an essential step in text analysis as it breaks down the text data into manageable pieces.
5. **Stemming or Lemmatization:** Stemming or lemmatization is the process of reducing words to their base form. For example, the words "running," "runner," and "run" can all be reduced to "run."
6. **Removing Numbers and Symbols:** Numbers and symbols can interfere with text analysis, so they should be removed.
7. **Spell Checking:** Spelling errors can impact the accuracy of the analysis, so it's recommended to use spell-checking tools to correct any errors.
8. **Removing HTML Tags:** If the text data has been scraped from websites, it may contain HTML tags, which should be removed to avoid any issues during analysis.

Overall, data preprocessing and cleaning on text data involve several steps that aim to standardize the text data and remove any noise or irrelevant information that can affect the accuracy of the analysis.

Code Snippet for Text Data Preprocessing:

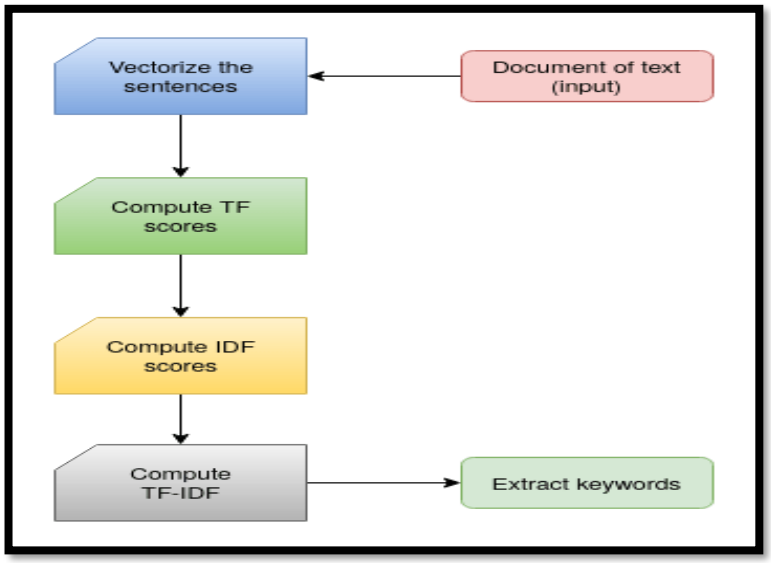
```
def clean_text(text):
    text = re.sub(r'http\S+', '', text) # remove URLs
    text = re.sub(r'@[A-Za-z0-9]+', '', text) # remove mentions
    text = re.sub(r'[^\w\s]', '', text) # remove special characters
    text = text.lower() # convert to lowercase
    stop_words = set(stopwords.words('english')) #remove stopwords
    tokens = text.split()
    filtered_tokens = [token for token in tokens if token not in stop_words]
    text = ' '.join(filtered_tokens)
    return text

df['clean_text'] = df['Utterance'].apply(lambda x: clean_text(x))
```

Feature Extraction – Text Input to Vectorization:

Feature extraction is the process of transforming raw text data into a format that can be easily understood and used by machine learning algorithms. In other words, it involves selecting and transforming relevant aspects of the text data into a set of numeric features that can be used to train a machine-learning model.

We have explored state-of-the-art techniques and implemented the below processes.

TF-IDF Vectorization:

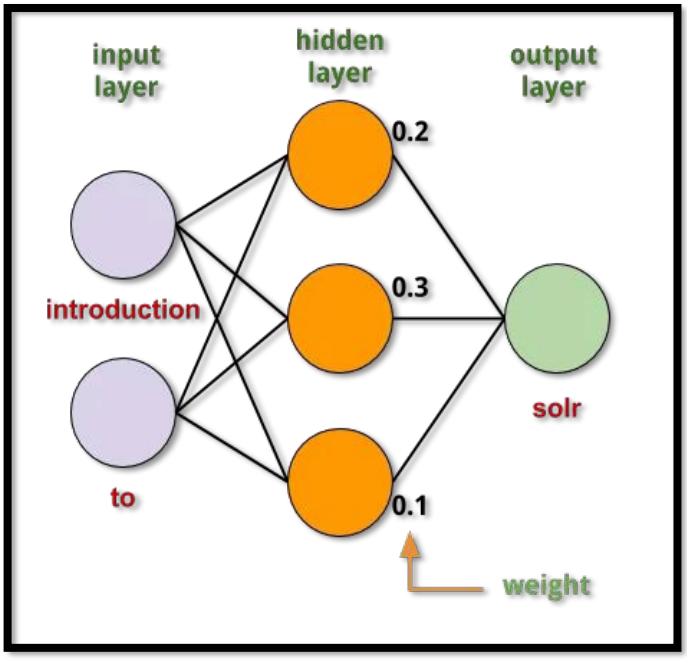


Here are the steps involved in the above TF-IDF vectorization process:

1. **Tokenization**: The text data is split into smaller units such as words or phrases.
2. **Text Cleaning:** The text data is cleaned and preprocessed by removing stop words, punctuation marks, and other irrelevant information.
3. **Term Frequency (TF) Calculation**: For each word in the text data, the number of times it appears in each document is counted. This count is referred to as the term frequency.
4. **Inverse Document Frequency (IDF) Calculation**: The inverse document frequency is calculated as the logarithm of the total number of documents divided by the number of documents in which the wordappears.
5. **TF-IDF Calculation**: The TF-IDF value for each word is calculated as the product of its term frequencyand inverse document frequency.
6. **Vectorization**: The TF-IDF values are then converted into a numeric vector representation for each document, where each element of the vector corresponds to the TF-IDF value of a particular word inthe vocabulary.

The resulting TF-IDF vectors can be used as features in machine learning algorithms to perform tasks such as text classification, sentiment analysis, and information retrieval. The advantage of TF-IDF vectorization is that it gives more weight to words that are important in a particular document but not frequent in the entire corpus, making it more effective in capturing the unique characteristics of each document.

Word2Vec Vectorizer:



The **advantage of Word2Vec is that it captures the meaning and relationships between words** in a way that traditional bag-of-words models cannot. By generating dense vector representations for each word, Word2Vec is able to capture the context and meaning of words in a more nuanced and sophisticated way, making it more effective in capturing the semantic properties of text data.

Here are the steps involved in the Word2Vec process:

1. **Corpus Preparation**: The text data is preprocessed by removing stop words, punctuation marks, and other irrelevant information.
2. **Tokenization**: The text data is split into smaller units such as words or phrases.
3. **Building Vocabulary:** The unique words in the corpus are identified and a vocabulary is created. Each word is assigned a unique index in the vocabulary.
4. **Training the Model:** Word2Vec is a neural network model that takes a large corpus of text data as input and generates a vector representation for each word in the vocabulary. The model is trained on the corpus by predicting the context words (words that appear near the target word) given a target word.
5. **Generating Word Embeddings:** The output of the Word2Vec model is a set of high-dimensional vector representations, or embeddings, for each word in the vocabulary. Each element of the vector corresponds to a learned feature of the word that captures its semantic and syntactic properties.
6. **Using Word Embeddings:** The resulting embeddings can be used as features in ML algorithms to perform tasks such as text classification, sentiment analysis, and information retrieval. They can also be used to measure semantic similarity between words or to perform operations such as word analogies.

Code Snippet for the above Feature Extraction Processes:

```
# Apply count vectorization and tf-idf transformation
# to the training and testing data
cv = CountVectorizer()
X_train_counts = cv.fit_transform(X_train)
tfidf = TfidfTransformer()
X_train_tfidf = tfidf.fit_transform(X_train_counts)
X_test_counts = cv.transform(X_test)
X_test_tfidf = tfidf.transform(X_test_counts)
```

TF-IDF Vectorization

```python
# Download pretrained word2vec model
wv = api.load('word2vec-google-news-300')

import numpy as np

def create_word_vectors(text):
    vectors = []
    for word in nltk.word_tokenize(text):
        if word in wv:
            vectors.append(wv[word])
    if not vectors:
        vectors.append([0] * 300)
    return np.mean(vectors, axis=0)


data['WordVectors'] = data['Utterance'].apply(create_word_vectors
```

Word2Vec Vectorizer

VI. MODELLING:

Modeling on-Text Data:

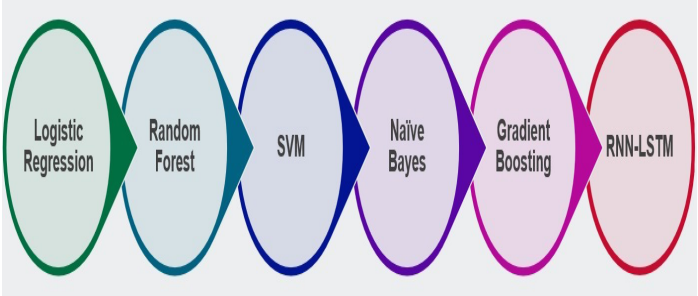
Label Encoder :

Before proceeding to the data modeling, we performed label encoding of the target Variable Sentiment. It is used to convert the categorical labels in the 'Emotion' column of the data frame to numerical values. This is necessary as ML algorithms can only work with numerical data.

Train Test Split

The data has been split into train and test with an 80:20 ratio and random state = 42.

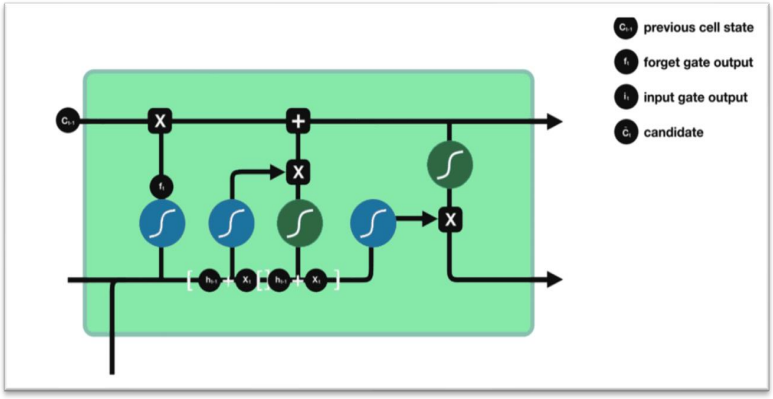
Now below are the ML and DL models implemented to classify the text data:



We have used both the Stat Models of ML and the RNN-Based LSTM model to compare the results. We have done this with the below primary assumption:

The LSTM architecture is particularly effective in handling long sequences of data because it is able to selectively remember or forget information from the past based on the current input. This allows it to capture long-term dependencies and make more accurate predictions on sequential data.

RNN-LSTM Architecture:



Here are the key components of the LSTM architecture:

1. **Cell State**: This is the main memory component of the LSTM. It is a vector that stores the informationfrom previous time steps.
2. **Input Gate:** This gate controls the flow of new input into the cell state. It decides which newinformation should be added to the cell state and which should be discarded.
3. **Forget Gate**: This gate controls the flow of old information out of the cell state. It decides whichinformation should be forgotten and which should be kept.
4. **Output Gate**: This gate controls the output from the cell state. It decides which information shouldbe passed on to the next time step.
5. **Hidden State**: This is the output of the LSTM at each time step. It is calculated based on the input,the previous hidden state, and the current cell state.

During the training process, the LSTM learns to update the cell state and the hidden state based on the input data and the previous state. The forget gate, input gate, and output gate are all learned through the training process.

The trained Model Summary:

```
model.summary()
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (None, 256)               264192

 dropout_10 (Dropout)        (None, 256)               0

 dense_10 (Dense)            (None, 128)               32896

 dropout_11 (Dropout)        (None, 128)               0

 dense_11 (Dense)            (None, 3)                 387

=================================================================
Total params: 297,475
Trainable params: 297,475
Non-trainable params: 0
_________________________________________________________________
```

The above model summary depicts a detailed overview of the architecture used:

1. **Layer type**: This indicates the type of layer as LSTM, and dense
2. **convolutional. Output shape**: Shape of the output tensor for the final layer is (3).
3. **Param #**: Number of trainable parameters in the Model are 297,475.
4. **Activation**: The activation function used for the layers is ReLU.

Example : Evaluating the Model on the Emotion class:

```
print("Accuracy:", accuracy)
print("Classification report:\n", report)
```

```
63/63 [==============================] - 2s 38ms/step - loss: 1.0440 -
Accuracy: 0.7615115115115115
```

```
Accuracy: 0.7615115115115115
Classification report:
              precision    recall  f1-score   support

       anger       0.56      0.61      0.59       226
     disgust       0.67      0.61      0.62        55
        fear       0.78      0.58      0.66        44
         joy       0.81      0.70      0.75       332
     neutral       0.92      0.91      0.91       985
     sadness       0.63      0.58      0.60       129
    surprise       0.60      0.66      0.62       227

    accuracy                           0.76      1998
   macro avg       0.61      0.71      0.66      1998
weighted avg       0.71      0.76      0.71      1998
```

VII. RESULTS

Model Comparison of Stat Models vs RNN:

Algorithm	Accuracy (TF-IDF)	Accuracy (Word2vec)
Logistic Regression	57.298	62.121
Random Forest	66.194	75.334
Support Vector Classifier	58.462	68.483
Naïve Bayes	60.667	63.604
Gradient Boosting Classifier	**72.071**	73.459
BERT	69.183	
RNN - LSTM (Long Short-Term Memory)	71.412	**76.151**

The algorithms listed in the table include Logistic Regression, Random Forest, Support Vector Classifier, Naïve Bayes, Gradient Boosting Classifier, BERT, and RNN-LSTM (Recurrent Neural Network with Long Short-TermMemory).

Each of these algorithms has been trained on the same dataset and evaluated using the same metrics, allowing for a fair comparison of their performance.

Based on the Results, the **RNN-LSTM algorithm has the highest accuracy score of 76.151% using the Word2vec technique**, followed by the **Random Forest algorithm with an accuracy score of 75.334%**. Both algorithms have outperformed the other algorithms listed in the table. However, it should be noted that the performance of an algorithm can depend heavily on the nature of the dataset and the specific task at hand.

VIII. CONCLUSION:

- Text Data: LSTM with Relu activation function : 76.15 accuracy.
- By implementing all the models we can conclude that: the Gradient Boosting classifier has highest accuracy of 72.071% with TF-IDF vectorizer and the LSTM with Relu activation function gave the overall highest accuracy of 76.151% with Word2vec vectorizer.
- The Word2vec vectorizer gave us the highest accuracy because it considers the semantic distance between the words in a text to the the text into vectors.

REFERENCES

[1] Patil, K.J.; Zope, P.H.; Suralkar, S.R. Emotion Detection From Speech Using Mfcc and Gmm. Int. J. Eng. Res. Technol. (IJERT) **2012**, 1, 9.

[2] Hassan, A.; Damper, R.I. Multi-class and hierarchical SVMs for emotion recognition. In Proceedings of the INTERSPEECH 2010, Makuhari, Japan, 26–30 September 2010; pp. 2354–2357.

[3] Lin, Y.L.; Wei, G. Speech emotion recognition based on HMM and SVM. In Proceedings of the 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 August 2005; Volume 8, pp. 4898–4901.

[4] Nicholson, J.; Takahashi, K.; Nakatsu, R. Emotion Recognition in Speech Using Neural Networks. In Proceedings of the 6th International Conference on Neural Information Processing (ICONIP '99), Perth, Australia, 16–20 November 1999.

[5] Schüller, B.; Rigoll, G.; Lang, M. Speech emotion recognition combining acoustic features and linguistic information in a hybrid support vector machine-belief network architecture. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, QC, Canada, 17–21 May 2004.

`