A Project Report

on

TWEET EMOTION RECOGNITION

*carried out as part of the **Mini Project DS3130** Submitted*

by

Perumalla Thushara Meher Siva Mani
209302296

Mehak Jain
209309069

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

MANIPAL UNIVERSITY
JAIPUR

School of Computing and Information Technology
Department of Information Technology

MANIPAL UNIVERSITY JAIPUR
JAIPUR-303007
RAJASTHAN, INDIA

November 2022

CERTIFICATE

Date: 24/11/2022

This is to certify that the minor project titled **TWEET EMOTION RECOGNITION** is a record of the bonafide work done by **Perumalla Thushara Meher Siva Mani**(Reg. No:209302296) and **Mehak Jain**(Reg. No:209309069) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Data Science of Manipal University Jaipur, during the academic year 2022-23.

Dr.Sweta Sharma

Project Guide, Department of Information Technology
Assistant Professor

Manipal University Jaipur

ABSTRACT

Social media has recently emerged as a premier method to disseminate information online. Millions of individuals communicate their thoughts, personal experiences, and social ideals through these online networks. Therefore, we explore the potential of social media to predict human emotions of Anger, Joy, Sadness, etc. We use an inbuilt dataset from hugging face.com and online portal offering pre-processed datasets by taking huge data from various platforms such as Kaggle.

We use Recurrent neural network algorithm to build the model for analysis. Recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. Recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence.

While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions. Our model could produce up to 80% accuracy upon testing the model with the pre-defined testing dataset.

LIST OF FIGURES

Table of Contents

1. Introduction

1.1 Motivation

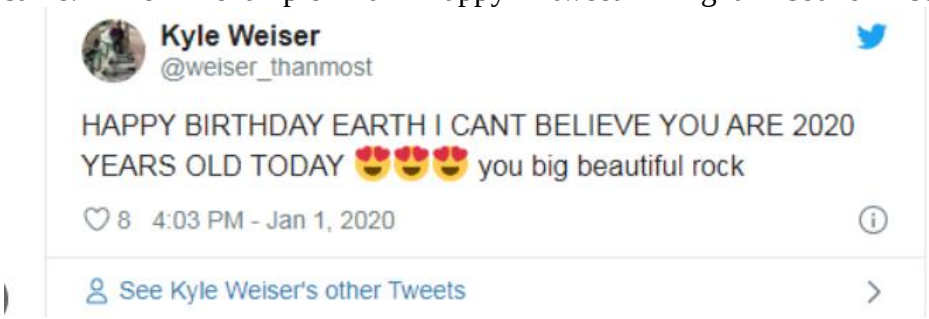
Feeling which is so called emotion is one kind of communication medium in our day to day life. Emotion can be articulated by the physical expression, facial expression, writing, speaking, behaviour, etc. In our technology-based era, having a social media account is very common. In today's world, people desire to express their emotions and thoughts from side to side different social media like Facebook, Twitter, Instagram and so on. Here millions of people in their everyday lives share their views, opinions and also their emotions or a particular thing through social media [1]. This huge amount of statistics containing emotions creates an opportunity for researchers for their work. So, detecting emotions by analyzing social media's text is a big challenge for researchers. Among all social media, Twitter has become a popular and interesting research area because of its short text. Twitter is a 'microblogging' structure that allows users to send and receive short posts called tweets. Tweets can be up to 140 characters long and can include links to relevant websites and resources [2]. The goal of emotion detection from text is to categorize the actual emotion polarity. It can be taken as a text classification problem. The dataset which we have used in this work contains "positive" and "negative" polarity of user's tweets

Analyzing the content of Tweets has become an increasingly more popular method to understand and make predictions about human social behaviors. Given the frequency with which Twitter is used by the broad population, it is a very rich source of data that can be used to analyze a variety of these behaviors.

1.2 Problem statement

Twitter – a platform where people tend to share their daily thoughts in multiple topics all over the world might also be a toxic place for some. Some people share information but some share their thoughts and all these tweets have some emotion attached to them.

The aim of this project is to identify such emotions and classify the tweets according to the same. For example a happy tweet might sound something like this-



Whereas a sad tweet might sound like this-

We humans deal with a lot of emotions every day and this project aims to classify those emotions mentioned in the tweets of any person.

1.3 Objectives

- Tokenization of tweets taken from a huge dataset
- Converting text labels to numeric vectors
- Using padding and truncating Different tweets are of different lengths.
- Using RNN with one embedding layer, two bi-directional LSTM layers and one dense layer in the model
- Classifying test set tweets to their emotions

2. Background Detail

2.1 Literature Review

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.[2]

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, convolutional neural networks and Transformers have been applied to fields including computer vision, speech recognition and natural language processing. Neural networks have been used for implementing language models since the early 2000s.

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

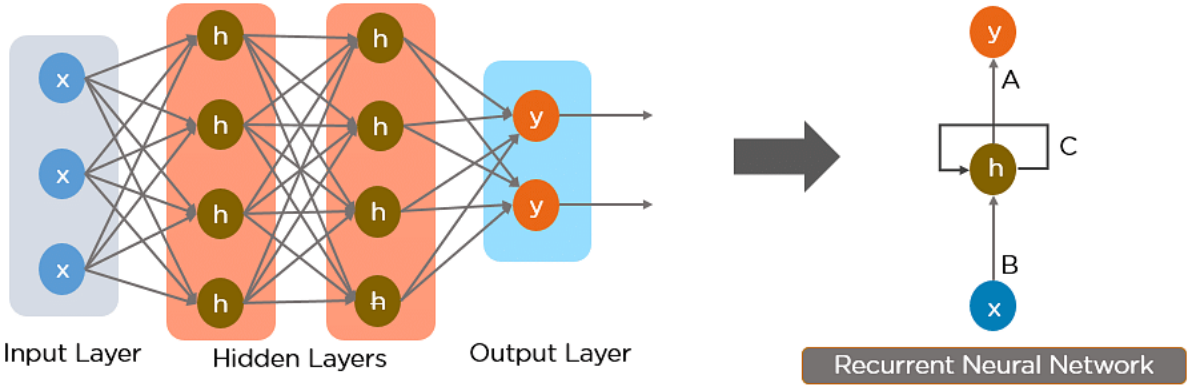
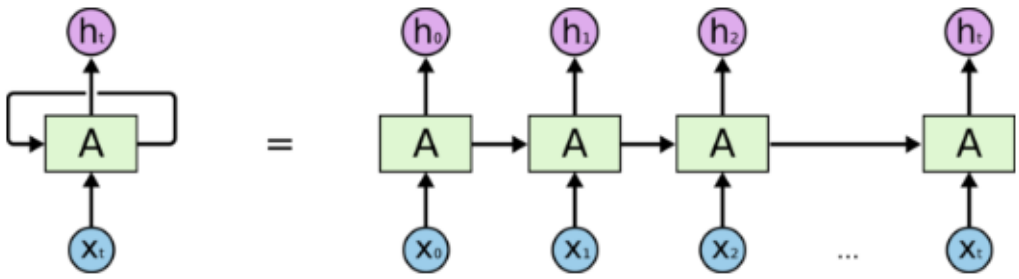


Fig 2.1 Recurrent Neural Network Structure



An unrolled recurrent neural network.

Advantages of Recurrent Neural Network

1. **RNN** can model sequence of data so that each sample can be assumed to be dependent on previous ones
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood.

LSTM helped to improve machine translation and language modeling.

Layers in the deep learning model can be considered as the architecture of the model. There can be various types of layers that can be used in the models. All of these different layers have their own importance based on their features. Like we use LSTM layers mostly in the time series analysis or in the NLP problems.

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here.

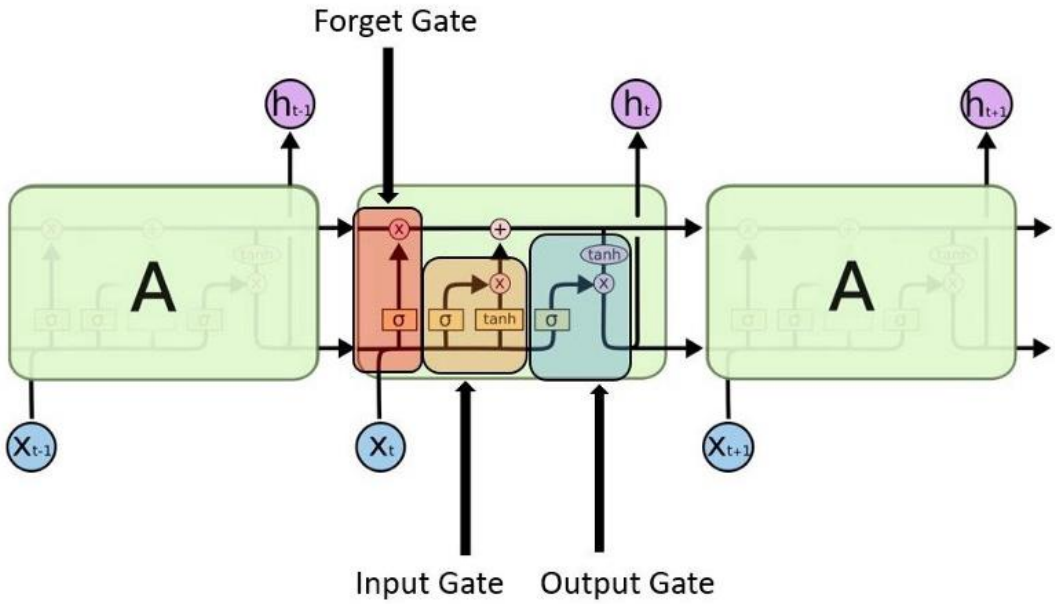


Fig 2.2 Structure of LSTM with Gates

In an LSTM network, three gates are present:

Input gate -discover which value from input should be used to modify the memory.

Forget gate — discover what details to be discarded from the block. It is decided by the sigmoid function.

Output gate — the input and the memory of the block is used to decide the output.

Metrics: -

In order to be able to evaluate the performance of each algorithm, several metrics are defined accordingly, and are discussed briefly in this section.

Confusion Matrix: It is very informative performance measures for classification tasks. Ci,j an element of matrix tells how many of items with label i are classified as label j. Ideally we are looking for diagonal Confusion matrix where no item is miss-classified. The matrix in Figure 1 is a good representation for our binary classification. Positive (P) represents toxic label and n (negative) represents non-toxic label.

Accuracy: This metric measures how many of the comments are labelled correctly. However, in our data set, where most of comments are not toxic, regardless of performance of model, a high accuracy was achieved.

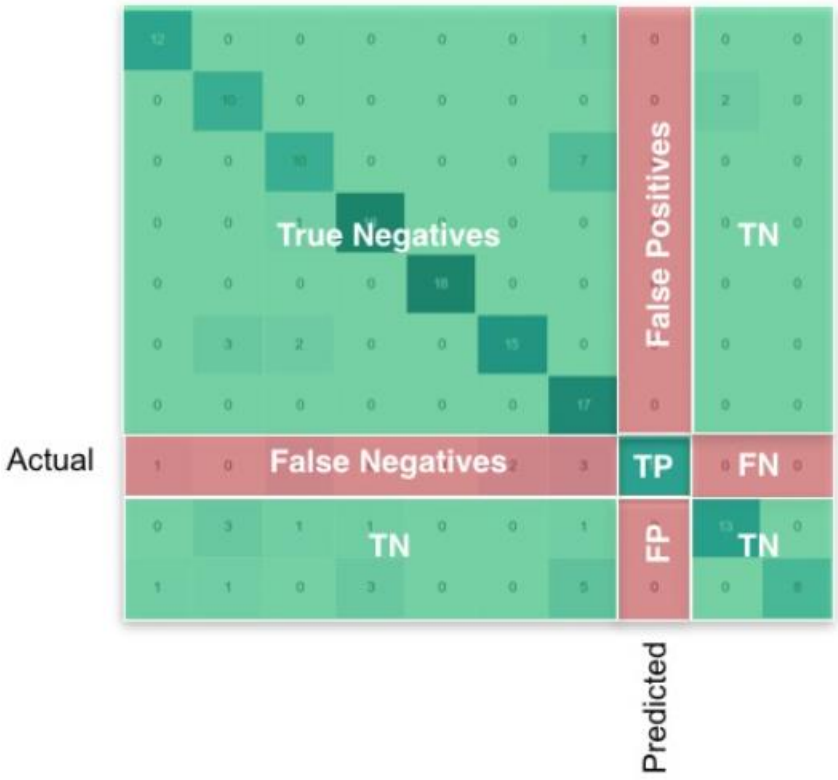
Accuracy = (TP + TN)/(TP + TN + FP + FN)

Precision and Recall: Precision and recall in were designed to measure the model performance in its ability to correctly classify the toxic comments. Precision explains what fraction of toxic classified comments are truly toxic, and Recall measures what fraction of toxic comments are labelled correctly

Precision = TP/(TP + FN), Recall = TP/(TP + TN)

F1 Score: Both Precision and Recall are important for checking the performance of the model. However, implementing a more advanced metric that combines both Precision and Recall together is quite informative and applicable. In this equation, setting $\beta = 1$ leads equation to return harmonic mean of Precision and Recall.

F1-Score = (Percision * Recall)/(Precision + Recall)



3. System Design & Methodology

3.1. System Architecture (Flow Chart)

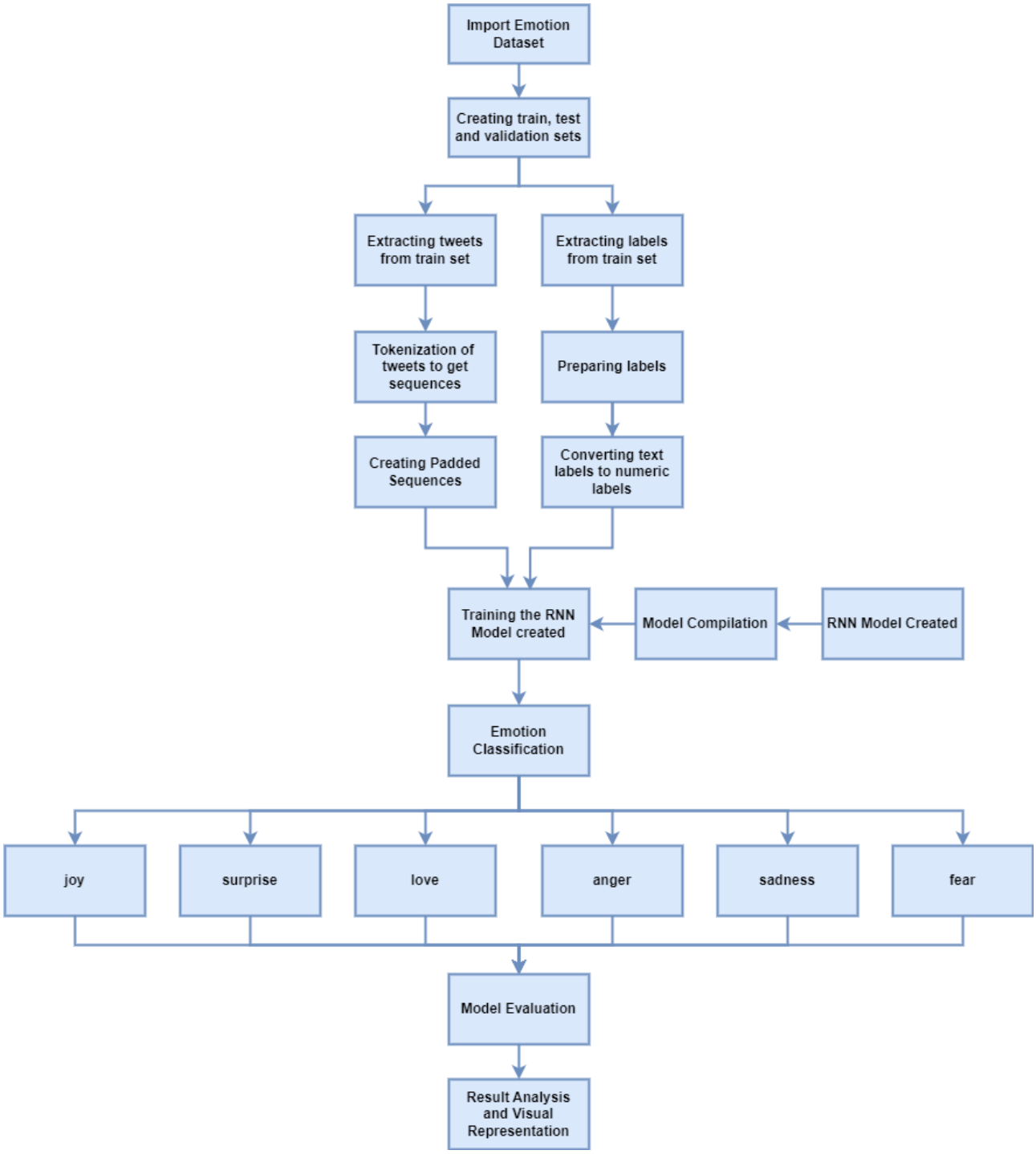


Fig 3.1 Flow Chart

3.2. Development Environment

3.2.1 Hardware Requirement

- I3 processor or higher
- 8GB RAM or higher
- 200GB ROM or higher

3.2.2 Software Requirement

- Windows 7 or higher
- Python 3.10 or higher
- Any python supported IDE-Jupyter Notebook, Google colab,etc.,

3.3. Methodology:

Emotion analysis is the process of identifying and analyzing the underlying emotions expressed in textual data. As this is a text classification problem we can use any of the classifiers like Naïve Bayes, Random Forest, XGBoost, etc., and Neural Network models like ANN and RNN. We are using Recurrent Neural Networks(RNN) in this project to reach our ultimate goal to classify tweets into six main emotions such as joy, surprise, love, angry, sadness, and fear.

RNN is used in this project because it is better suited to analyzing temporal, sequential data, such as text or videos.

Dataset:-We have selected the emotion dataset from the hugging faces NLP package as the dataset for this project. It is because the motion dataset has approximately 20000 tweets with emotion labels which is a very huge dataset that is appropriate for performing nlp tasks without bias. Neural Networks too need huge data to work without bias.

Data Splits

name	train	validation	test
default	16000	2000	2000
emotion	16000	2000	2000

```
In [5]: #Displaying the dataset
        dataset

Out[5]: {'train': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 1600
        0),
         'validation': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 2
        000),
         'test': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 2000)}
```

Fig 3.2 Overview of Emotion Dataset

Tokenization: -Tokenization is breaking the raw text into small chunks. The current problem is all about text. The model cannot interpret raw text. So, Tokenization breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the model for the NLP. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

- We tokenized the tweets using Tokenizer (). We had tokenized 10000 most frequent words using parameter num_words=10000 and rest are tokenized as <UNK>(unknown)

Padding and Truncating: - Batched inputs are often different lengths, so they can't be converted to fixed-size tensors. Padding and truncation are strategies for dealing with this problem, to create rectangular tensors from batches of varying lengths. Padding adds a

special padding token to ensure shorter sequences will have the same length as either the longest sequence in a batch or the maximum length accepted by the model. Truncation works in the other direction by truncating long sequences.

- After calculating the lengths of tweets, we get to know that all the tweets are of different lengths. But in order to create a model for them they need to be in a fixed input shape. So, we have done padding and truncating to get them into a fixed input shape.

RNN Model: - Recurrent Neural Network (RNN) is a type of Neural Network where the output from previous step is fed as input to the current step**.**

- RNN is used in this project because it is better suited to analyzing temporal, sequential data, such as text.

Long Short-Term Memory Network (LSTM): - To solve the problem of Vanishing and Exploding Gradients in a Deep Recurrent Neural Network, many variations were developed. One of the most famous of them is the Long Short Term Memory Network **(**LSTM). In concept, an LSTM recurrent unit tries to "remember" all the past knowledge that the network is seen so far and to "forget" irrelevant data.

- We have used two Bi-directional LSTM layers in the RNN model we created.

The model is then trained and then evaluated using metrics.

4. Implementation and Result

4.1. Modules of Implemented Project

There are 8 major modules in this project

4.1.1 Setup and Imports
4.1.2 Importing Data and Data Exploration
4.1.3 Tokenization of Tweets
4.1.4 Padding and Truncating Sequences
4.1.5 Preparing the labels
4.1.6 Creating the model
4.1.7 Training the model
4.1.8 Evaluating the model

4.2. Implementation Detail

4.2.1 Setup and Imports

- Installing Hugging Face's nlp package
- Importing Libraries such as tensorflow, numpy, matplotlib and nlp

4.2.2 Importing Data

- Import the Tweet Emotion dataset
- Displaying the dataset
- Create train, validation and test sets

4.2.3 Tokenization of Tweets

- Import inbuilt Tokenizer from text preprocessing module from tensorflow
- Creating sequences by tokenization of tweets using Tokenizer() for the 10000 most frequently used words

```
In [18]: #Creating an object tokenizer(We can keep any name) to use Tokenizer() method
         #We specified num_words=10000 in Tokenizer() means we only use ten thousand most frequent unique words in corpus to be tokenized
         #oov_token means out of vocabulary words
         #We specified oov_token='<UNK>' in Tokenizer() means rest of words other than 10000 most frequent words will be specified as out
         #These out of vocabulary words have the same token given as specified string '<UNK>' (unknown)
         tokenizer = Tokenizer(num_words=10000, oov_token='<UNK>')
         #Using the created object tokenizer to convert the corpus tweets_train having tweets of train set to sparse matrix stored in obje
         count_matrix=tokenizer.fit_on_texts(tweets_train)
```

Fig 4.1 Code snippet of tokenization

4.2.4 Padding and Truncating Sequences

- Calculate length of the tweets in train set 'tweets_train' and visualize them
- Creating a function 'get_sequences' to get padded sequences.

```
def get_sequences(tweets):
    sequences = tokenizer.texts_to_sequences(tweets)
    # maxlen=50 means highest lenth allowed is 50
    #Post truncating is used then the tweets length>50 will be truncated/chopped to length 50 at from the end
    #Post padding is used means the tweets will be padded zero at the end
    padded_sequences = pad_sequences(sequences, truncating='post', maxlen=50, padding='post')
    return padded_sequences
```

Fig 4.2 Code snippet of padded sequences

4.2.5 Preparing the labels

- Creating a set containing labels of train set

- Display and visualize the labels of train set
- Creating a class to index dictionary with keys as labels and values as index
- Creating a index to class dictionary with keys as index and values as class
- Creating a lambda function to convert text labels into numerical labels

4.2.6 Creating the model

- Creating a RNN model using sequential class with one Embedding, two LSTM and one Dense Layers

```
In [41]: #Sequential() class is used to create a sequential model 'model_created'
#A list of layers are passed inside Sequential() class to create a sequential model 'model_created'
model_created = tf.keras.models.Sequential([
    #Embedding layer with inp_dim=10000,output_dim=16 and input_length/max length of sequence=50
    tf.keras.layers.Embedding(10000, 16, input_length=50),
    #Bi-directional LSTM Layer which gives output for every time step as return_sequences=True
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20, return_sequences=True)),
    #Ouput from previous Bi-directional LSTM Layer is fed to this LSTM Layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),
    #Dense Layer to change the dimensions of output
    #to make it easy for model to define relationship b/w values of data the model is working on
    #Output size specified as 6 dimensions and activation function as 'softmax'
    #As this is a multi class classification problem, it is best to use 'softmax' as activation function
    #We need only six classes(Emotions:-joy,love,surprise,...).So reduce dimensions to six
    tf.keras.layers.Dense(6, activation='softmax')
])
```

Fig 4.3 Code snippet of RNN model creation

- Compiling the created model

```
In [42]: #compile() method of keras is used to compile the already created model 'model_created'
model_created.compile(
    #compile() method has 3 main arguments loss,optimizer and metrics
    #As there is no one-hot encoded data it is better to use 'sparse_categorical_crossentropy'
    loss='sparse_categorical_crossentropy',
    #'adam' optimizer is best for sparse and noisy data
    optimizer='adam',
    metrics=['accuracy']
)
```

Fig 4.4 Code snippet of Compilation of model

- Display model summary

4.2.7 Training the model

- Preparing the validation set by previously created functions like get_sequences, names_to_ids to get padded sequences and to convert text labels to numeric labels for validation set
- Training the model

```
In [47]: #training the model using fit()
model_trained = model_created.fit(
    #Training the model using train set's padded sequences and converted numeric values of labels/classes
    # X_train=padded_train_seq and y_train=labels_train_num
    padded_train_seq,labels_train_num,
    #Validation set used to tune the parameters of a classifier(number of hidden units,epochs,etc,.) in a neural network.
    # X_val=padded_val_seq and y_val=labels_val_num
    validation_data=(padded_val_seq,labels_val_num),
    #Number of epochs to run=20
    epochs=20,
    #callbacks specified with monitor='val_accuracy' and patience=2
    # It means if there is no improvement in val_accuracy for continuous 2 epochs then the training of model will be stopped
    # even before 20 epochs completed
    callbacks=[
        tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)
    ]
)
```

Fig 4.5 Code snippet of model training

4.2.8 Evaluating the model

- Visualizing training history
- Preparing the test set by previously created functions like get_sequences, names_to_ids to get padded sequences and to convert text labels to numeric labels for test set
- Evaluating the trained model by feeding test data
- Predicting labels of test set by using predict() method on model

```
In [67]: #Using predict() to get predicted data
         #Here 'test data' X_test=padded_test_seq and the 'test data label prediction' y_pred=labels_predicted
         labels_predicted = np.argmax(model_created.predict(padded_test_seq), axis=-1)

         #Shape of
         labels_predicted.shape, labels_test_num.shape

         63/63 [==============================] - 1s 18ms/step

Out[67]: ((2000,), (2000,))
```

Fig 4.6 Code snippet of predicted labels

- Plot confusion matrix
- Calculate classification report

4.3 Results and Discussion

4.3.1 Observations from lengths of tweets

- Most of the tweets have their lengths between 10 and 20
- There are very few tweets of length more than 50
- Padding and truncating of tweets is done to maintain the length of all the tweets at 50.

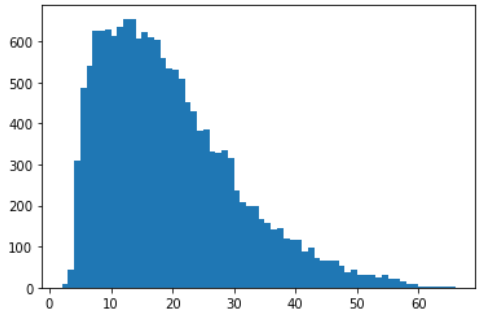


Fig 4.7 Visualizing length of all tweets

4.3.2 The labels of train set

- From the above histogram we can say that sadness and joy are the major classes
- love,anger and suprise are minor classes
- There is a class imbalance which csn be solved by many methods like undersampling and oversampling.

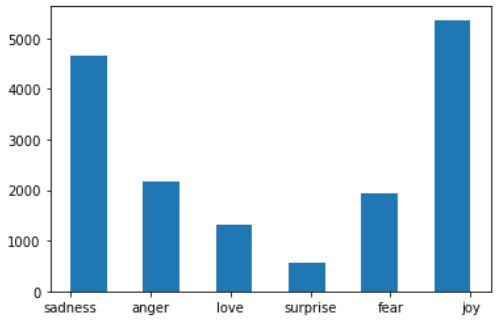


Fig 4.8 Labels of the train set

4.3.3 Displaying an example padded sequence

```
In [25]: # Displaying 0th index element of all the padded sequences in padded_train_seq
         padded_train_seq[0]

Out[25]: array([  2, 139,   3, 679,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0])
```

Fig 4.9 Visualizing a padded sequence

4.3.4 Summary of the created model

- We have one embedding layer,two bi-directional LSTM layers and one dense layer in the model
- The first bidirectional LSTM layer has output shape(None,50,40) which means it is giving output of all sequences not only the last sequence

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 50, 16)            160000

 bidirectional (Bidirectiona  (None, 50, 40)           5920
 l)

 bidirectional_1 (Bidirectio  (None, 40)               9760
 nal)

 dense (Dense)               (None, 6)                 246

=================================================================
Total params: 175,926
Trainable params: 175,926
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.10 Summary of RNN model created

4.3.5 Observations from the training are:-

- Accuracy of train set:98.62%
- Validation accuracy or accuracy on validation set:89.15%
- The val_accuracy of 9^{th},10^{th} and 11^{th} epochs are 0.9045,0.9000 and 0.8915 respectively
- The val_accuracy is not improving for continuous two epochs(patience=2) in 10^{th} and 11^{th} epochs but decreasing after 9^{th} epoch
- So the model training stopped after 11^{th} epoch.

```
Epoch 1/20
500/500 [==============================] - 44s 63ms/step - loss: 1.3056 - accuracy: 0.4853 - val_loss: 0.7612 - val_accuracy:
0.7220
Epoch 2/20
500/500 [==============================] - 29s 58ms/step - loss: 0.5482 - accuracy: 0.8066 - val_loss: 0.6400 - val_accuracy:
0.7925
Epoch 3/20
500/500 [==============================] - 27s 55ms/step - loss: 0.2694 - accuracy: 0.9148 - val_loss: 0.4056 - val_accuracy:
0.8745
Epoch 4/20
500/500 [==============================] - 28s 56ms/step - loss: 0.1568 - accuracy: 0.9498 - val_loss: 0.3818 - val_accuracy:
0.8830
Epoch 5/20
500/500 [==============================] - 28s 56ms/step - loss: 0.1283 - accuracy: 0.9606 - val_loss: 0.3670 - val_accuracy:
0.8860
Epoch 6/20
500/500 [==============================] - 28s 55ms/step - loss: 0.1016 - accuracy: 0.9682 - val_loss: 0.5103 - val_accuracy:
0.8745
Epoch 7/20
500/500 [==============================] - 31s 62ms/step - loss: 0.0963 - accuracy: 0.9713 - val_loss: 0.3429 - val_accuracy:
0.8955
Epoch 8/20
500/500 [==============================] - 28s 56ms/step - loss: 0.0721 - accuracy: 0.9759 - val_loss: 0.3743 - val_accuracy:
0.8985
Epoch 9/20
500/500 [==============================] - 28s 57ms/step - loss: 0.0621 - accuracy: 0.9805 - val_loss: 0.3812 - val_accuracy:
0.9045
Epoch 10/20
500/500 [==============================] - 27s 54ms/step - loss: 0.0524 - accuracy: 0.9837 - val_loss: 0.3989 - val_accuracy:
0.9000
Epoch 11/20
500/500 [==============================] - 27s 54ms/step - loss: 0.0424 - accuracy: 0.9862 - val_loss: 0.4407 - val_accuracy:
0.8915
```

Fig 4.11 Epochs of training

4.3.6 Accuracy and loss of training and validation sets

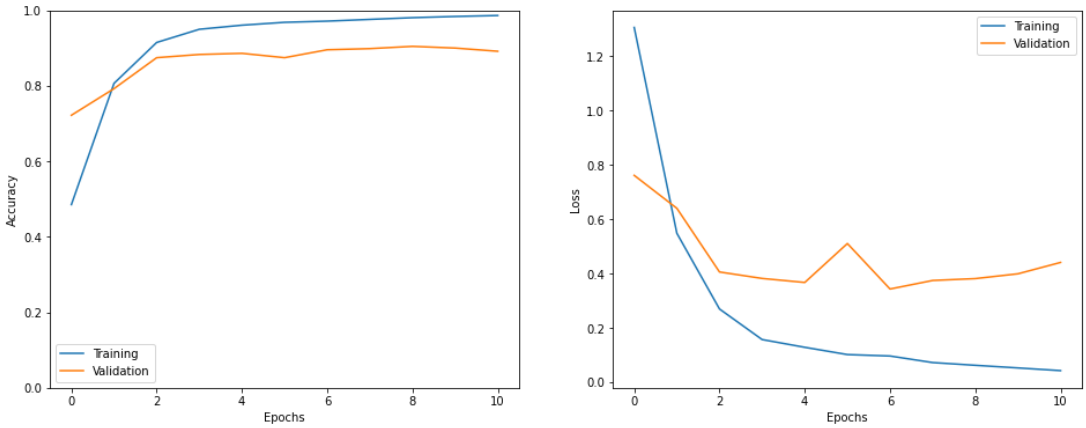


Fig 4.12 Accuracy and loss of training and validation sets

4.3.1 Confusion Matrix of y_test=labels_test_num and y_pred=labels_predicted

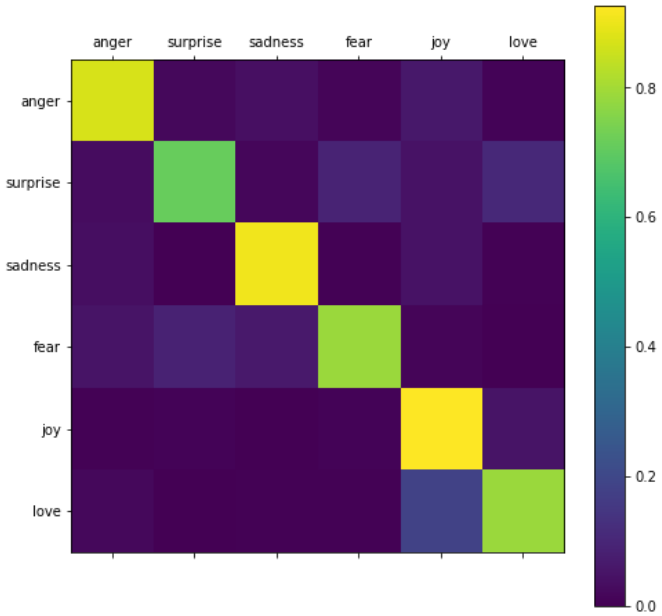


Fig 4.13 Confusion matrix

4.3.2 Explanation of Classification report

- We have six classes with labels-0,1,2,3,4 and 5 which originally represent anger, surprise, sadness, fear, joy, and love respectively.
- Overall Accuracy is 88%

	precision	recall	f1-score	support
0	0.85	0.87	0.86	275
1	0.60	0.71	0.65	66
2	0.95	0.91	0.93	581
3	0.90	0.79	0.84	224
4	0.89	0.93	0.91	695
5	0.73	0.79	0.76	159
accuracy			0.88	2000
macro avg	0.82	0.83	0.82	2000
weighted avg	0.88	0.88	0.88	2000

Fig 4.14 Classification report of all classes/labels

4.4 Month-wise wise plan of work (Progress Chart/Time Line Chart)

- 4th September 2022- Deciding on topics of projects
- 11th September 2022- First initial draft of project
- 11th to 17th September 2022- Data Exploration
- 1st to 7th October 2022- Data preprocessing
- 8th to 15th October 2022- RNN Model
- 16th to 26th October 2022- RNN Model Evaluation
- 3rd to 10th November 2022- Results and Analysis
- 21st to 23rd November 2022- Preparation of project report and presentation

5. Conclusion and Future Plan

We implemented Emotion analysis on the Emotion dataset to classify the tweets into six main emotions such as joy, surprise, love, angry, sadness, and fear using. This technique proves to be efficient for emotion prediction. We have applied Recurrent Neural Network(RNN) on the preprocessed tweets and labels. Recurrent Neural Network(RNN) applied on the emotion dataset stands out with overall accuracy 88%.

We used python as a language in this project to do all the necessary steps like loading dataset, data exploration, pre-processing data, tokenization, padding, truncating, splittingg of dataset into train, test and validation sets, building RNN model and evaluation of models, etc.,

There are still some limitations of this project as we used 10000 most frequent words only to be tokenized which may contain some repeated unimportant words. So the accuracy may somewhat drop due to this. We can solve this problem to some extent using algorithms like TFIDF(Term Frequency Inverse Document Frequency). TFIDF calculates the importance of words used in document. We can use TFIDF to consider the most important words instead of taking frequent words.

Benefits of Emotion analysis is that it helps companies monitor product and brand sentiment from customer reviews and feedback. It also helps them understand the needs of customers. It can be used on all the social media platforms like Twitter, Facebook and Instagram, etc., by their management companies to filter hate speech, abusive words and racist comments to take action against them. It can also be used by any company providing services or involved in direct product sales like Tata Cliq, Amazon, Flipkart, etc., to classify their customer reviews into emotions using Emotion Analysis. With emotion recognition, we can get more granular data, bringing the real thing that the user was feeling, at the moment of that specific post.

6. References

Web

[1] Emotion Dataset, https://huggingface.co/datasets/emotion

[2] Tokenization, https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4

[3] Tokenizer Class arguments, https://medium.com/analytics-vidhya/understanding-nlp-keras-tokenizer-class-arguments-with-example-551c100f0cbd#:~:text=num_words%20is%20nothing%20but%20your,word_index)%20%2B%201%22.

[4] Padding and Truncating, https://huggingface.co/docs/transformers/pad_truncation

[5] Types of Padding and Truncating and ways to implement them, https://stackoverflow.com/questions/42943291/what-does-keras-io-preprocessing-sequence-pad-sequences-do

[6] Recurrent Neural Network, https://www.tutorialspoint.com/microsoft_cognitive_toolkit/microsoft_cognitive_toolkit_recurrent_neural_network.htm

[7] Keras Model creation and compilation https://www.tutorialspoint.com/keras/keras_model_compilation.htm

[8] Keras Model Evaluation and Prediction https://www.tutorialspoint.com/keras/keras_model_evaluation_and_prediction.htm

[9] Implementation of RNN and LSTM https://www.tutorialspoint.com/keras/keras_time_series_prediction_using_lstm_rnn.htm

7. Acknowledgement

We, Perumalla Thushara Meher Siva Mani and Mehak Jain, the students of BTech in Data Science and Engineering - 2020 batch (first year first semester) would like to thank all of our teachers and Manipal University Jaipur for providing us this platform. We would like to thank our mentor Mr.Rahul Saxena who helped us to clear any doubt what so ever we had to encounter while making this project.

This is a wonderful opportunity for us students to show our creativity and to get ourself involve in the work that we love and desire to do. This has enable us to learn and have fun at same time. We will try our best to perform as per the expectation put upon us by our parents, family members, teachers, Manipal University Jaipur and the whole nation.

Lastly, I would like to thank my Family for the tremendous amount of support they offered me while making this Project. I also had the invaluable support of my friends and colleagues which helped me in completing this Project.