Group Details

1. Dharmik Shetty - I073
2. Manan Shah - I075
3. Aniruddh Kulkarni - I081

A.1. Aim of the Project/ Problem/Statement

Tweet emotion recognition using Natural Language Processing with Tensorflow

Description of the Project

In this project we try to predict the emotion of twitter posts using NLP Techniques with Tensorflow library. The dataset that we used in the project, **Emotion dataset**, was an inbuilt dataset in the NLP Library. Emotion *is a dataset of English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise.* The packages that we used in our project include tensorflow, numpy, matplotlib, nlp and random. We have performed 10 tasks in order to predict whether the given tweet can be classified as *anger, fear, joy, love, sadness, and surprise,* and will give a short overview of each task a follows:

1. Task 1: We loaded the required libraries as well as packages that were required in the project and we did some Data Exploration, to check how our data actually looks. The dataset that we used in the project, **Emotion dataset**, was an inbuilt dataset in the NLP Library. Here's a link to the dataset 'emotion' dataset from NLP Library. We plotted subplots of Epochs on the X-axis versus Accuracy on the Y-axis and Epochs on the X-axis versus Loss on the Y-axis. We have also plotted a confusion matrix which will help us later whether we have predicted the right emotion for the given tweet

Code Snippet

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import nlp
import random


def show_history(h):
    epochs_trained = len(h.history['loss'])
    plt.figure(figsize=(16, 6))

    plt.subplot(1, 2, 1)
    plt.plot(range(0, epochs_trained),
h.history.get('accuracy'), label='Training')
    plt.plot(range(0, epochs_trained),
h.history.get('val_accuracy'), label='Validation')
    plt.ylim([0., 1.])
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(range(0, epochs_trained), h.history.get('loss'),
label='Training')
    plt.plot(range(0, epochs_trained),
h.history.get('val_loss'), label='Validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

```
def show_confusion_matrix(y_true, y_pred, classes):
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y_true, y_pred, normalize='true')

    plt.figure(figsize=(8, 8))
    sp = plt.subplot(1, 1, 1)
    ctx = sp.matshow(cm)
    plt.xticks(list(range(0, 6)), labels=classes)
    plt.yticks(list(range(0, 6)), labels=classes)
    plt.colorbar(ctx)
    plt.show()
```

Task 3: Importing our data that is *emotion* dataset from the NLP library. We tried to divide our dataset into two features: text and *label* which helps us to tokenize the words in the next step.

Code Snippets:

```
dataset=nlp.load_dataset('emotion')
```

```
dataset
```

```
train=dataset['train']
val=dataset['validation']
test=dataset['test']
```

```
type(train)
```

```
def get_tweet(data):
 tweets=[x['text'] for x in data]
 labels=[x['label'] for x in data]
 return tweets,labels
```

```
tweets,labels=get_tweet(train)
```

```
tweets[90],labels[90]
```

```
('i was ready to meet mom in the airport and feel her ever supportive
arms around me',
 'love')
```

```
tweets[46],labels[46]
```

```
('i lost my special mind but don t worry i m still sane i just wanted
you to feel what i felt while reading this book i don t know how many
times it was said that sam was special but i can guarantee you it was
many more times than what i used in that paragraph did i tell you she
was special',
 'joy')
```

```
tweets[5],labels[5]
```

```
('ive been feeling a little burdened lately wasnt sure why that was',
 'sadness')
```

Task 4: Tokenizer

In this task we tokenized the tweets by using the inbuilt function *Tokenizer* from the *tensorflow's keras* module.

```
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer=Tokenizer(num_words=10000,oov_token='<UNK>')
tokenizer.fit_on_texts(tweets)

tokenizer.texts_to_sequences([tweets[5]])
```

[[73, 48, 8, 7, 56, 521, 319, 328, 158, 161, 9, 20]]

Task 5: Padding and Truncating Sequences

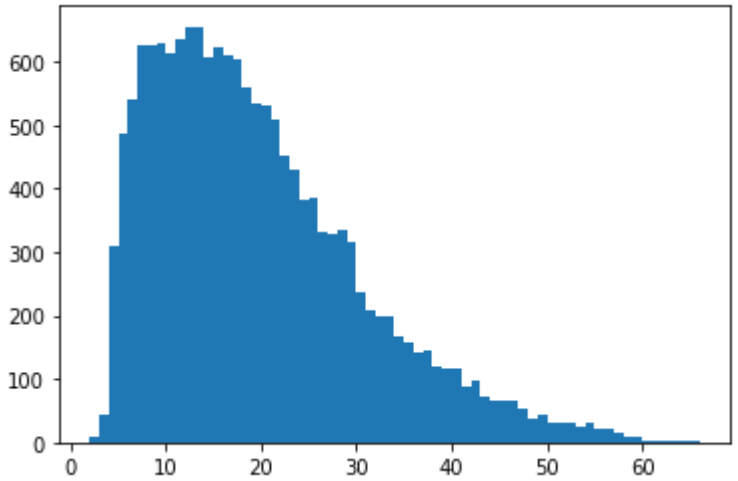
The two main things we did here was that we first checked the length of the tweets and created a padded sequence. We tried to split each word from the sentence and tried to get all the words in a List. Then we plotted histplot, the first step is to “bin” the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval, the histplot was plotted for *length of tweets on the Y-axis* versus bins on the X–axis. We tried to take bins as an interval of 10 and divided each word present in the list by 10 and plotted it on the Y-axis.

Then we performed padding on the words, here padding refers to inserting non-informative characters into a string, we insert 0’s on either side until the max length of the string is equal to 50. We have kept the maximum length of the string to be 50, which means that we add required number of zeros on either sides until the max length of the string becomes 50.

For strings whose length is more than 50, they are truncated to length 50 here truncated refers to shorten the length of the string until the max length is 50.

Code Snippets

```
lengths=[len(t.split(' ')) for t in tweets]
plt.hist(lengths,bins=len(set(lengths)))
plt.show
```




```
from tensorflow.keras.preprocessing.sequence import
pad_sequences

maxlen=50

def get_sequences(tokenizer,tweets):
 sequences=tokenizer.texts_to_sequences(tweets)

padded=pad_sequences(sequences,truncating='post',padding='post
',maxlen=maxlen)
 return padded

padded_train_seq=get_sequences(tokenizer,tweets)
```

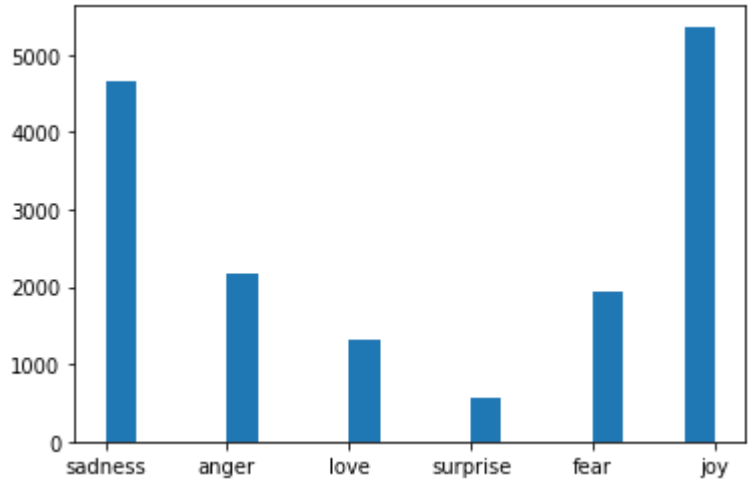
```
padded_train_seq[0]
```

```
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],
dtype=int32)
```

```
classes=set(labels)
print(classes)
```

```
{'surprise', 'fear', 'anger', 'joy', 'sadness', 'love'}
```

```
plt.hist(labels,bins=20)
plt.show()
```



Then we make a function called get sequence where we pass two variables tokenizer and tweets which helps us perform padding and truncating on the length of the string. We assign random values to each word which helps us in predicting the emotion later

```
def get_sequences(tokenizer,tweets):
 sequences=tokenizer.texts_to_sequences(tweets)

padded=pad_sequences(sequences,truncating='post',padding='post
',maxlen=maxlen)
 return padded
```

```
padded_train_seq=get_sequences(tokenizer,tweets)
```

```
padded_train_seq[0]
```

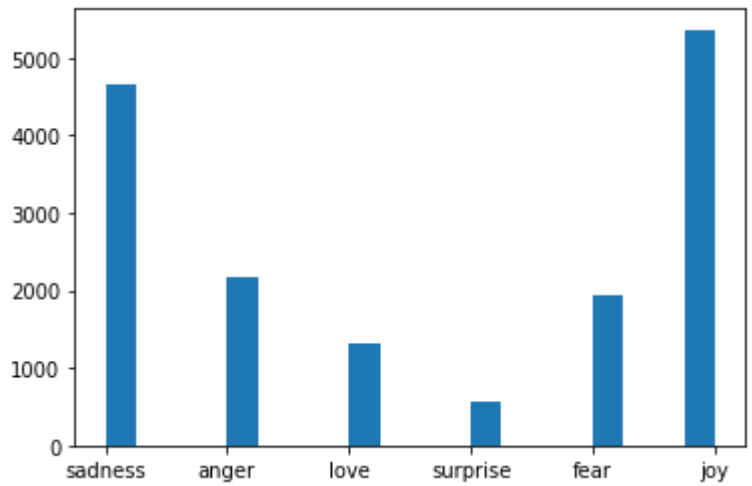
```
array([  2, 139,   3, 679,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0],dtype=int32)
```

Now we print the classes present in the column called labels, which gives us all the emotions given in the emotion dataset that is *anger, fear, joy, love, sadness, and surprise.* Here we try to plot the histogram by keeping the value of the bins as 20 in between each class present in the label column on the X-axis versus number of words present under that class(a*nger, fear, joy, love, sadness, and surprise)* on the Y-axis.

```
classes=set(labels)
print(classes)
```

```
{'surprise', 'fear', 'anger', 'joy', 'sadness', 'love'}
```

```
plt.hist(labels,bins=20)
plt.show()
```



Then we assign ids starting from 0 to each class under labels column as follows and store it in a dictionary, then we make a numpy array for all the classes present in the labels column

```
class_to_index=dict((c,i) for i,c in enumerate(classes))
index_to_class=dict((v,k) for k,v in class_to_index.items())
```

```
class_to_index
```

{'surprise': 0, 'love': 1, 'joy': 2, 'anger': 3, 'sadness': 4, 'fear': 5}

```
class_to_index.items()
```

dict_items([('surprise', 0), ('love', 1), ('joy', 2), ('anger', 3), ('sadness', 4), ('fear', 5)])

```
index_to_class
```

{0: 'surprise', 1: 'love', 2: 'joy', 3: 'anger', 4: 'sadness', 5: 'fear'}

```
names_to_ids= lambda labels:np.array([class_to_index.get(x)
for x in labels])
```

```
names_to_ids
```

```
np.array([class_to_index.get(x) for x in labels])
```

array([4, 4, 3, ..., 2, 3, 4])

```
train_labels=names_to_ids(labels)
len(train_labels)
```

```
16000
```

Task 7: Creating the model

We make a Neural Network model

Task 8: Training the model

Here we prepare a validation set and train or Neural Network model.

We make call variables val_tweets, val_labels and get_tweets