```
#import libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers as L
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.optimizers import Adam

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
import plotly.figure_factory as ff
import seaborn as sns


import numpy as np
import pandas as pd

import nltk
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import TweetTokenizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

import re
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#dataset
train = pd.read_csv('Corona_NLP_train.csv',encoding='latin1')
test = pd.read_csv('Corona_NLP_test.csv',encoding='latin1')

train.head()
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive

```
INDEX = np.random.randint(1,train.shape[0])
print(INDEX)
print(train['OriginalTweet'][INDEX])
print(train['Sentiment'][INDEX])
```

```
1109
US Food Industry Scrambles To Resupply Stores Amid Apocalyptic Surge In Demand | Zero Hedge https://t.co/Pw9QbGcE1j #coronavirus #C
Extremely Negative
```

```
#dataset size
print('Examples in train data: {}'.format(len(train)))
print('Examples in test data: {}'.format(len(test)))
```

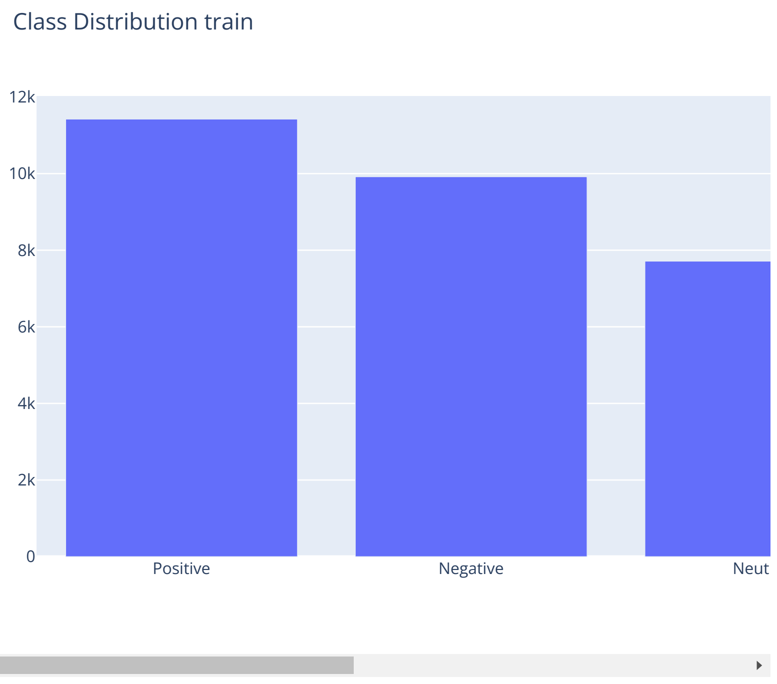
```
Examples in train data: 41157
Examples in test data: 3798
```

```
#missing value
train.isna().sum()
```

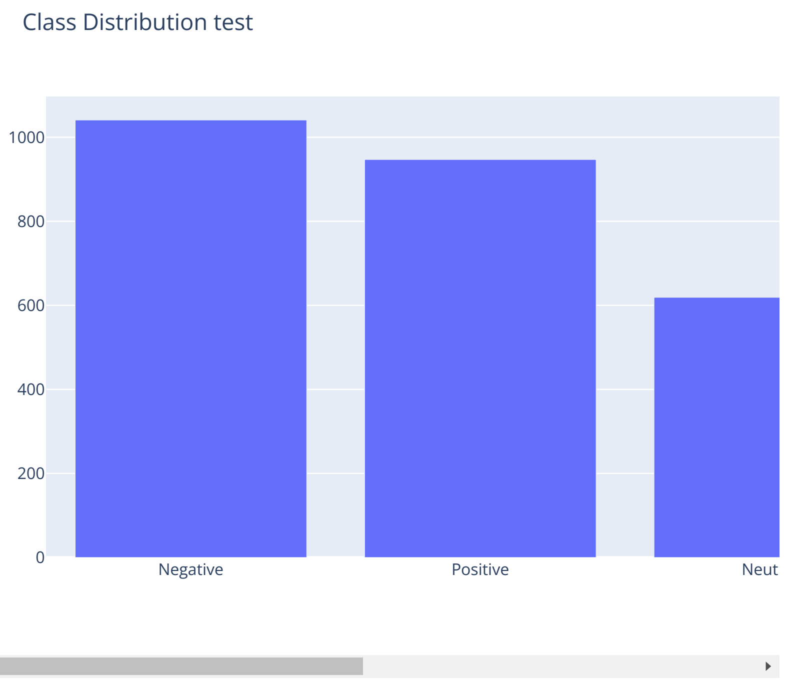
```
UserName           0
ScreenName         0
Location        8590
TweetAt            0
OriginalTweet      0
Sentiment          0
dtype: int64
```

```
#class distribution
dist_train = train['Sentiment'].value_counts()
dist_test = test['Sentiment'].value_counts()

def ditribution_plot(x,y,name):
    fig = go.Figure([
        go.Bar(x=x, y=y)
    ])

    fig.update_layout(title_text=name)
    fig.show()
```

```
ditribution_plot(x= dist_train.index, y= dist_train.values, name= 'Class Distribution train')
```



```
ditribution_plot(x= dist_test.index, y= dist_test.values, name= 'Class Distribution test')
```



```
#data preprocessing
X = train['OriginalTweet'].copy()
y = train['Sentiment'].copy()
```

```python
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```python
#cleaning data
def data_cleaner(tweet):

    # remove urls
    tweet = re.sub(r'http\S+', ' ', tweet)

    # remove html tags
    tweet = re.sub(r'<.*?>',' ', tweet)

    # remove digits
    tweet = re.sub(r'\d+',' ', tweet)

    # remove hashtags
    tweet = re.sub(r'#\w+',' ', tweet)

    # remove mentions
    tweet = re.sub(r'@\w+',' ', tweet)

    #remove non words character+ underscore
    tweet = re.sub(r'[\W_]+',' ',tweet)

    #removing stop words
    tweet = tweet.lower().split()
    tweet = " ".join([word for word in tweet if not word in stop_words])

    return tweet


stop_words = stopwords.words('english')

X_cleaned = X.apply(data_cleaner)
X_cleaned.head()
```

```
0
1    advice talk neighbours family exchange phone n...
2    coronavirus australia woolworths give elderly ...
3    food stock one empty please panic enough food ...
4    ready go supermarket outbreak paranoid food st...
Name: OriginalTweet, dtype: object
```

```python
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yours
```

```python
#tokenizing
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_cleaned)

X = tokenizer.texts_to_sequences(X_cleaned)

vocab_size = len(tokenizer.word_index)+1

print("Vocabulary size: {}".format(vocab_size))
print("\nExample:\n")
print("Sentence:\n{}".format(X_cleaned[6]))
print("\nAfter tokenizing :\n{}".format(X[6]))

X = pad_sequences(X, padding='post')
print("\nAfter padding :\n{}".format(X[6]))
```

```
Vocabulary size: 33750

Example:

Sentence:
cashier grocery store sharing insights prove credibility commented civics class know talking

After tokenizing :
[1075, 6, 3, 1167, 682, 2766, 9513, 9514, 17742, 1391, 53, 811]

After padding :
[ 1075     6     3  1167   682  2766  9513  9514 17742  1391    53   811
     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]
```

```
#feature encoding
encoding = {'Extremely Negative': 0,
            'Negative': 0,
            'Neutral': 1,
            'Positive':2,
            'Extremely Positive': 2
           }

labels = ['Negative', 'Neutral', 'Positive']


y.replace(encoding, inplace=True)
```

```
#model building
tf.keras.backend.clear_session()

# hyper parameters
EPOCHS = 2
BATCH_SIZE = 32
embedding_dim = 16
units = 256

model = tf.keras.Sequential([
    L.Embedding(vocab_size, embedding_dim, input_length=X.shape[1]),
    L.Bidirectional(L.LSTM(units,return_sequences=True)),
    L.GlobalMaxPool1D(),
    L.Dense(256, activation="relu"),
    L.Dropout(0.7),
    L.Dense(128, activation="relu"),
    L.Dropout(0.3),
    L.Dense(3)
])


model.compile(loss=SparseCategoricalCrossentropy(from_logits=True),
              optimizer='adam',metrics=['accuracy']
             )

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 47, 16)            540000

 bidirectional (Bidirectiona  (None, 47, 512)          559104
 l)

 global_max_pooling1d (Globa  (None, 512)              0
 lMaxPooling1D)

 dense (Dense)               (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 3)                 387

=================================================================
Total params: 1,263,715
Trainable params: 1,263,715
Non-trainable params: 0
_________________________________________________________________
```

```
X[1]
```

```
array([  385,   736,  2381,   167,  2502,   715,  1084,   932,   319,
         273,   715,  1084,  2381,  1063,  3366,  3043,  4363,   368,
          12,    11,  2127, 17740,  2901,   101,   898,  2714,   139,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0], dtype=int32)
```

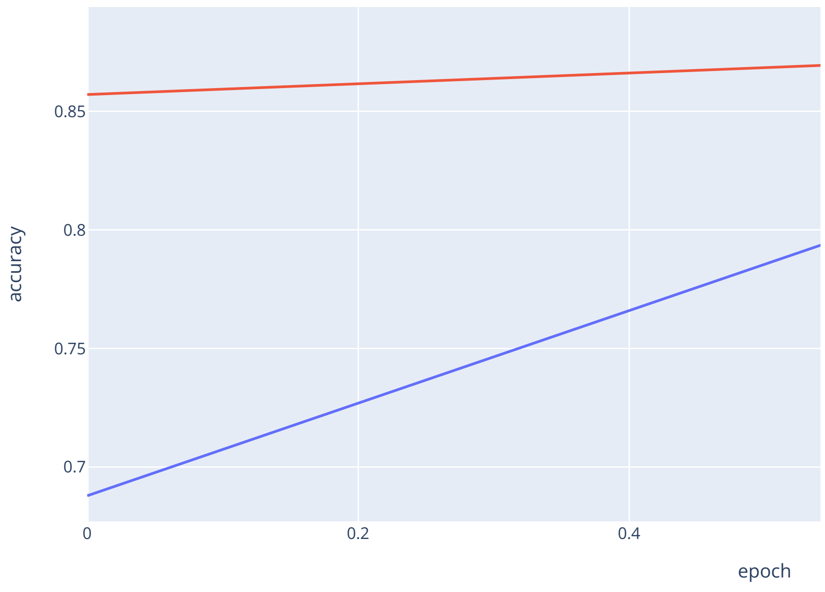
```
X.shape[1]
```

```
47
```

```
#training
history = model.fit(X, y, epochs=EPOCHS, validation_split=0.12, batch_size=BATCH_SIZE)
```

```
Epoch 1/2
1132/1132 [==============================] - 16s 12ms/step - loss: 0.7151 - accuracy: 0.6879 - val_loss: 0.4062 - val_accuracy: 0.8
Epoch 2/2
1132/1132 [==============================] - 13s 11ms/step - loss: 0.3582 - accuracy: 0.8831 - val_loss: 0.3576 - val_accuracy: 0.8
```

```
fig = px.line(
    history.history, y=['accuracy', 'val_accuracy'],
    labels={'index': 'epoch', 'value': 'accuracy'}
)

fig.show()
```



```
fig = px.line(
    history.history, y=['loss', 'val_loss'],
    labels={'index': 'epoch', 'value': 'loss'}
)

fig.show()
```

```
#Preprocessing test data
X_test = test['OriginalTweet'].copy()
y_test = test['Sentiment'].copy()

X_test = X_test.apply(data_cleaner)

X_test = tokenizer.texts_to_sequences(X_test)

X_test = pad_sequences(X_test, padding='post',maxlen=47)

y_test.replace(encoding, inplace=True)
```

```
X_test.shape
```

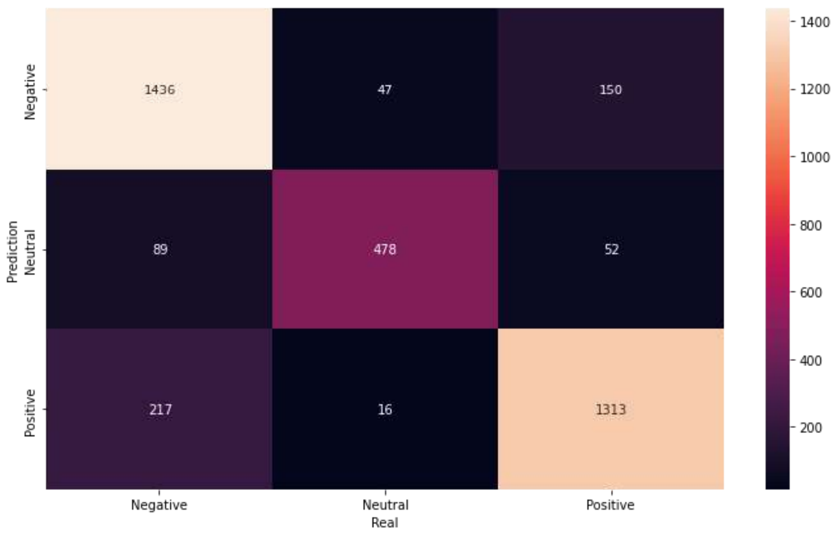
```
(3798, 47)
```

```
#predict data test
predict_x=model.predict(X_test)
classes_x=np.argmax(predict_x,axis=1)
```

```
loss, acc = model.evaluate(X_test,y_test,verbose=0)
print('Test loss: {}'.format(loss))
print('Test Accuracy: {}'.format(acc))
```

```
Test loss: 0.4198286235332489
Test Accuracy: 0.8496577143669128
```

```
conf = confusion_matrix(y_test, classes_x)

cm = pd.DataFrame(
    conf, index = [i for i in labels],
    columns = [i for i in labels]
)

plt.figure(figsize = (12,7))
sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('Real')
plt.ylabel('Prediction')
plt.show()
```



```
test['OriginalTweet']
```

```
0       TRENDING: New Yorkers encounter empty supermar...
1       When I couldn't find hand sanitizer at Fred Me...
2       Find out how you can protect yourself and love...
3       #Panic buying hits #NewYork City as anxious sh...
4       #toiletpaper #dunnypaper #coronavirus #coronav...
                              ...
3793    Meanwhile In A Supermarket in Israel -- People...
3794    Did you panic buy a lot of non-perishable item...
3795    Asst Prof of Economics @cconces was on @NBCPhi...
3796    Gov need to do somethings instead of biar je r...
3797    I and @ForestandPaper members are committed to...
Name: OriginalTweet, Length: 3798, dtype: object
```

```python
text = [input()]
text = tokenizer.texts_to_sequences(text)
text = pad_sequences(text, padding='post',maxlen=47)
```

```
#Finland is boosting its #COVID-19 #vaccination campaign in preparation for the new #COVID19 #BA.5 variant first detected in #South
```

```python
pred = model.predict(text)
output = np.argmax(pred)

if output==0:
    print('Hasil klasifikasi : negative')
elif output==1:
    print('Hasil klasifikasi : neutral')
else:
    print('Hasil klasifikasi : positive')
```

```
Hasil klasifikasi : negative
```