# Tweet Sentiment Extraction

## Problem Statement:

With all of the tweets circulating every second it is hard to tell whether the sentiment behind a specific tweet will impact a company, or a person's, brand for being viral (positive), or devastate profit because it strikes a negative tone. Capturing sentiment in language is important in these times where decisions and reactions are created and updated in seconds. But, which words actually lead to the sentiment description? In this competition we will need to pick out the part of the tweet (word or phrase) that reflects the sentiment. The main objective of this study is to figure out which phrase or word determines the sentiment of the tweet.

## Metric:

Jaccard Score. As we need to find the correct selected text we shall use jaccard score which calculates intersection over union. This metric is suggested by kaggle.

# Code

```
!pip install -q kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle competitions download -c tweet-sentiment-extraction
```

```
Downloading tweet-sentiment-extraction.zip to /content
  0% 0.00/1.39M [00:00<?, ?B/s]
100% 1.39M/1.39M [00:00<00:00, 132MB/s]
```

```
import zipfile
with zipfile.ZipFile('/content/tweet-sentiment-extraction.zip', 'r') as zipref:
  zipref.extract('train.csv')
  zipref.extract('test.csv')
  zipref.extract('sample_submission.csv')
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
train_data=pd.read_csv('/content/train.csv')
```

```
train_data.head(5)
```

| | textID | text | selected_text | sentiment |
|---|---|---|---|---|
| 0 | cb774db0d1 | I`d have responded, if I were going | I`d have responded, if I were going | neutral |
| 1 | 549e992a42 | Sooo SAD I will miss you here in San Diego!!! | Sooo SAD | negative |
| 2 | 088c60f138 | my boss is bullying me... | bullying me | negative |
| 3 | 9642c003ef | what interview! leave me alone | leave me alone | negative |
| 4 | 358bd9e861 | Sons of ****, why couldn`t they put them on t... | Sons of ****, | negative |

# Exploratory Data Analysis

[ ] ↳ *43 cells hidden*

# Preprocessing

```
from bs4 import BeautifulSoup
import re
def preprocess(sentence):
    sentence=sentence.replace('****',"curse")# changing bad words(marked as **** in text) to 'curse'
    sentence=' '.join(e.lower() for e in sentence.split())
    return sentence.strip()
```

```
train_data['preprocessed_texts']=train_data['text'].apply(lambda x: preprocess(x))
train_data['preprocessed_sel_texts']=train_data['selected_text'].apply(lambda x: preprocess(x))
```

Now we need to output for training the model. Here our output would be in the form of array with values 1 and 0. The array will be of length of the maximum length of the sentences and value 1 will be assigned to the particular index position for which that particular text is in selecetd text.

eg:

text: He is a good boy.

selected text: good boy

Output_array: [0,0,0,1,1]

To start with it is necessary to exract the index position of the words. There were lot of complications in extracting it.

eg:

text: This is awesome

selected text: s awesome

text: pod...sad...i have

selected text ..sad

Below code is used to run these complications.

```
import re
def crct_start_ind_1(x):
  '''
  This function is used to retrive the text wherein the selected text is have few first words and rest missing in the real text.
  Eg: Text: Jealously
      Selceted text: Jealous
  Returns: Jealously
      In this case the function returns the word Jealously beacause with the exact word we can find the index position in the text.

                                          (OR)

      Text: gonna
      Selected text: onna
  Returns: gonna
  '''
  tex=x[0]#List of words in text
  sel_word=x[1]#string: selected text
  len_sel_word=len(sel_word)
  for wrd in tex:
    if sel_word == wrd[:len_sel_word]:# This finds the words like Jealous
      wrd_in_tex=wrd
      break
    elif sel_word == wrd[-len_sel_word:]:# This finds for the words like onna
      wrd_in_tex=wrd
      break
  return wrd_in_tex

def crct_start_ind_2(x):
  """
  This function is used to retrive the text wherein the selected text has words in between the words in the text.
  eg: Text: pod...sad...i
      Selected text: ..sad

  returns: pod...sad..i

  """
  tex=x[0]
  sel_word=x[1]
  for wrd in tex:
    if re.search(sel_word,wrd) != None:
      wrd_in_tex=wrd
      break
  return wrd_in_tex


def start_indices(x):
    text_str=x[0]
    sel_text_str=x[1]
    prepro_text=x[2]
    text_list=text_str.split()#splitting the text
    sel_text_list=sel_text_str.split()#spliting selected text
    prepro_text_list=prepro_text.split()#spliting preprocessed text #no need
    end=sel_text_list[0]
    try:
      #Finds whether the first word of selected text is in text.
      index=text_list.index(end)
    except:
      if len(sel_text_list)==1:
        try:
          #if the 1st word is not in selected text and number of words in selected text is 1 then find the word using the crct_start_ind_1 function
          end_new=crct_start_ind_1((text_list,end))
          index=text_list.index(end_new)
        except:
          #There are some words like in text 'pod...sad...i' but selected text it is '..sad'. The number of abnormal data like these are 15 data points. We can neglect those data points and give
          index=0
      else:
        try:
          #There are some words like in the text 'this awesome' but selected text it is 's awesome'. Here we neglect those 's' and start from the other word.
          end=sel_text_list[1]
          index=text_list.index(end)
        except:
          #There are some words like in the text 'this awesome' but selected text it is 's awes'. Here we neglect 's' and take 'awes' and retrive 'awesome' from original text.
          end_new=crct_start_ind_1((text_list,end))
          index=text_list.index(end_new)
    return index
def end_indices(x):
    text_str=x[0]
    sel_text_str=x[1]
    start_indices=x[2]
    text_list = text_str.split()
    sel_text_list = sel_text_str.split()
    end = sel_text_list[-1]
    try:
        #if the end word is in the text in then we directly retrive the index of that word and declare it as end index
        index = text_list.index(end,start_indices)
    except:
        #if the end word is not proper then we obtain the end index by adding the length of the words in selected text to the starting index to get the exactl end index
        index = start_indices+(len(sel_text_list)-1)
    return index



train_data['start_index']=train_data[['text','selected_text','preprocessed_texts']].apply(lambda x: start_indices(x),axis=1)
train_data['end_index']=train_data[['text','selected_text','start_index']].apply(lambda x: end_indices(x),axis=1)


complete_data=train_data[['text','selected_text','preprocessed_texts','sentiment','start_index','end_index']]


from sklearn.model_selection import train_test_split
train,test=train_test_split(complete_data,stratify=complete_data['sentiment'],test_size=0.2,random_state=42)


print(train.shape)
print(test.shape)

     (21984, 6)
     (5496, 6)


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

     Mounted at /content/drive
```

```
train.to_csv('/content/drive/MyDrive/case_study_2_new/train_pos_neg_neu.csv')
test.to_csv('/content/drive/MyDrive/case_study_2_new/test_pos_neg_neu.csv')
```

```
train
```

| | text | selected_text | preprocessed_texts | sentiment | start_index | end_index |
|---|---|---|---|---|---|---|
| **16595** | Press `Ctrl` on bottom right. It`s there. KY | Press `Ctrl` on bottom right. It`s there. KY | press `ctrl` on bottom right. it`s there. ky | neutral | 0 | 7 |
| **15039** | ah remember the days when you`d sleep in until... | loser | ah remember the days when you`d sleep in until... | negative | 21 | 21 |
| **1804** | my momma is comin 2night ! 2morrow tennis day... | yuppie | my momma is comin 2night ! 2morrow tennis day ... | positive | 11 | 11 |
| **7302** | I do that all the time | I do that all the time | i do that all the time | neutral | 0 | 5 |
| **27217** | We don`t feel too comfortable using it. It`s... | . It`s not awful, but pretty icky | we don`t feel too comfortable using it. it`s n... | negative | 7 | 13 |
| **...** | ... | ... | ... | ... | ... | ... |
| **25183** | Should be drank with sugar and milk, not coff... | Should be drank with sugar and milk, not coffe... | should be drank with sugar and milk, not coffe... | neutral | 0 | 14 |
| **7595** | Thinks she`s getting sick..... | Thinks she`s getting sick..... | thinks she`s getting sick..... | negative | 0 | 3 |
| **16318** | Get Up, You are NOT old! What did you do?! =O | Get Up, You are NOT old! What did you do?! =O | get up, you are not old! what did you do?! =o | neutral | 0 | 10 |
| **7399** | Ha Ha thanks Tom! I`m such a loser! Hopefully... | Ha Ha thanks Tom! I`m such a loser! Hopefully ... | ha ha thanks tom! i`m such a loser! hopefully ... | neutral | 0 | 23 |
| **21790** | they can`t be in their carriers anymore? | they can`t be in their carriers anymore? | they can`t be in their carriers anymore? | neutral | 0 | 6 |

21984 rows × 6 columns

```
X_train=train
X_test=test
```

```
sent_len=[]
for i in X_train['preprocessed_texts']:
  sent_len.append(len(i.split()))
```

```
max_sent_len=max(sent_len)
```

```
print(max_sent_len)
```

```
33
```

Here the maximum length of the sentence is 33.

```
train_data=train_data.reset_index(drop=True)
```

```
X_train=X_train.reset_index(drop=True)
X_test=X_test.reset_index(drop=True)
```

Now let's create the output array with the start and end index that has been defined.

```
y_train=np.zeros((len(X_train),max_sent_len))
for i in range(len(X_train)):
  s_ind=X_train['start_index'][i]
  e_ind=X_train['end_index'][i]
  y_train[i][s_ind:e_ind+1]=1

y_test=np.zeros((len(X_test),max_sent_len))
for i in range(len(X_test)):
  s_ind=X_test['start_index'][i]
  e_ind=X_test['end_index'][i]
  y_test[i][s_ind:e_ind+1]=1


print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(21984, 6)
(21984, 33)
(5496, 6)
(5496, 33)
```

## Tokenizing text

Now let's tokenizer the given text.

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_text = Tokenizer(lower=True,split=' ',oov_token='oov',filters='')
tokenizer_text.fit_on_texts(X_train['preprocessed_texts'])
train_text=tokenizer_text.texts_to_sequences(X_train['preprocessed_texts'])
test_text=tokenizer_text.texts_to_sequences(X_test['preprocessed_texts'])
print(len(train_text),len(test_text))
vocab_size_text=len(tokenizer_text.word_index)+1
print(vocab_size_text)
```

```
21984 5496
38689
```

```
len_of_texts=[]
for i in range(len(train_text)):
  len_of_texts.append(len(train_text[i]))
max_length=max(len_of_texts)
print("Maximum length is ",max_length)
```

```
Maximum length is  33
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
train_text = pad_sequences(train_text,maxlen=max_length,padding='post')
test_text = pad_sequences(test_text,maxlen=max_length,padding='post')
print(train_text.shape,test_text.shape)
```

```
(21984, 33) (5496, 33)
```

Importing the Glove Vectors for embedding

```
!wget --header="Host: cdn-lfs.huggingface.co" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36" --header="Ac
```

```
--2022-10-26 06:13:24--  https://cdn-lfs.huggingface.co/stanfordnlp/glove/3123e7f5c3f6a30095d413b12fc3284bbf717acd2a9bed63d1c7c13bf5223352?response-content-disposition=attachment%3B%20file
Resolving cdn-lfs.huggingface.co (cdn-lfs.huggingface.co)... 13.227.254.47, 13.227.254.33, 13.227.254.52, ...
Connecting to cdn-lfs.huggingface.co (cdn-lfs.huggingface.co)|13.227.254.47|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1520408741 (1.4G) [application/zip]
Saving to: ‘glove.twitter.27B.zip’

glove.twitter.27B.z 100%[===================>]   1.42G  81.0MB/s    in 18s

2022-10-26 06:13:42 (80.6 MB/s) - ‘glove.twitter.27B.zip’ saved [1520408741/1520408741]
```

```
import zipfile
with zipfile.ZipFile('/content/glove.twitter.27B.zip', 'r') as zipref:
  zipref.extractall('/content/')
  zipref.close()
```

```
embeddings_index = dict()
f = open('/content/glove.twitter.27B.200d.txt')
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 1193515 word vectors.
```

```
len(embeddings_index['fan'])
```

```
200
```

The embedding matrix is a 200 dimensional matrix.

```
# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size_text, 200))
for word, i in tokenizer_text.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
len(embedding_matrix)
```

```
38689
```

```
tokenizer_sentiment = Tokenizer(lower=True,split=' ',oov_token='oov')
tokenizer_sentiment.fit_on_texts(X_train['sentiment'])
train_sentiment=tokenizer_sentiment.texts_to_sequences(X_train['sentiment'])
test_sentiment=tokenizer_sentiment.texts_to_sequences(X_test['sentiment'])
print(len(train_sentiment),len(test_sentiment))
vocab_size_sentiment=len(tokenizer_sentiment.word_index)+1
print(vocab_size_sentiment)
```

```
21984 5496
5
```

```
tokenizer_sentiment.word_index
```

```
{'oov': 1, 'neutral': 2, 'positive': 3, 'negative': 4}
```

```
len_of_sentiment=[]
for i in range(len(train_sentiment)):
  len_of_sentiment.append(len(train_sentiment[i]))
max_length_sentiment=max(len_of_sentiment)
print("Maximum length for sentiment is ",max_length_sentiment)
```

```
Maximum length for sentiment is  1
```

```
embedding_matrix_sentiment = np.zeros((vocab_size_sentiment, 200))
for word, i in tokenizer_sentiment.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix_sentiment[i] = embedding_vector
```

```
len(embedding_matrix_sentiment)
```

```
5
```

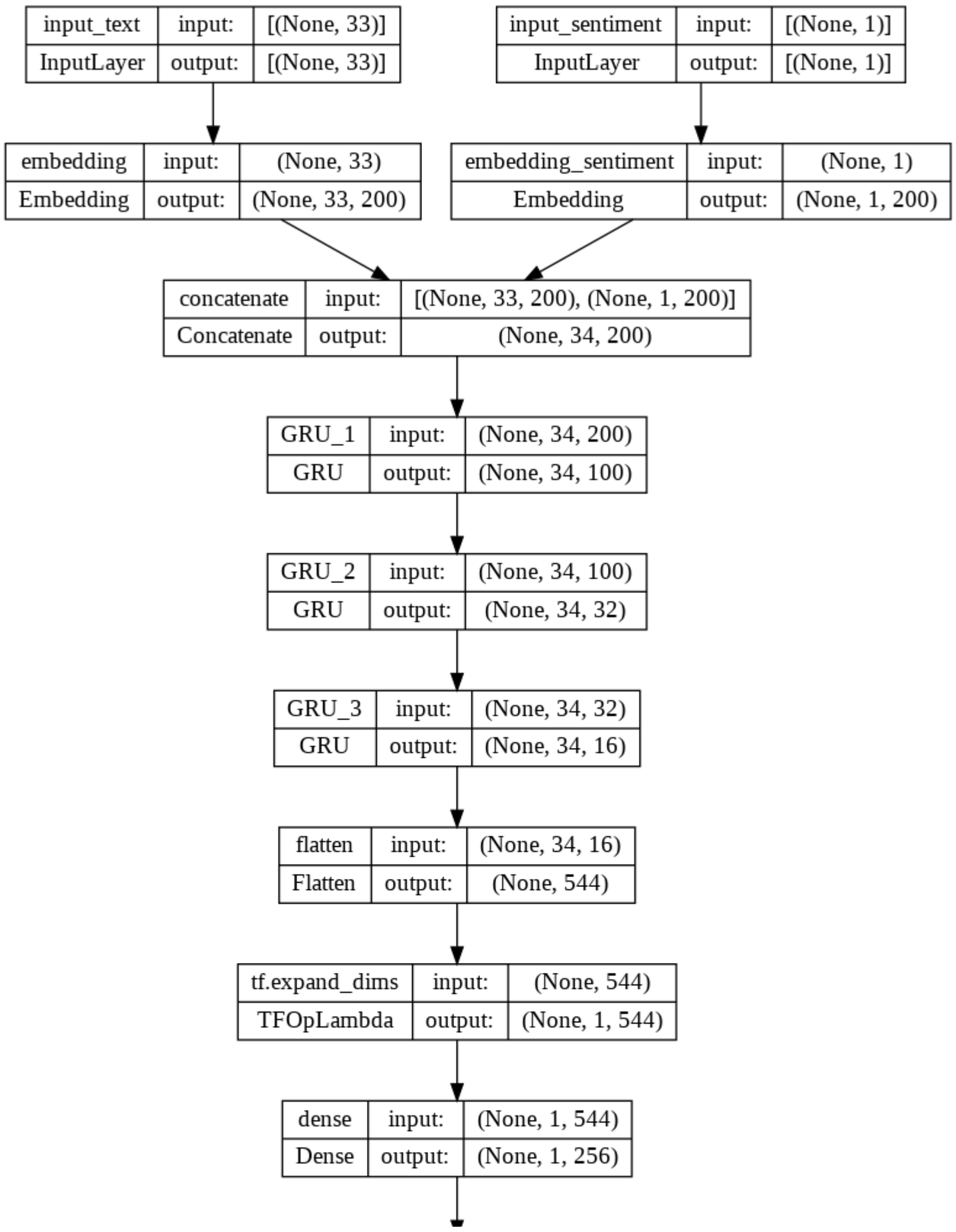
```
max_length
```

```
33
```

## Modelling

```
import tensorflow as tf
import keras
from tensorflow.keras.layers import Input,Embedding,GRU,Dense,Flatten,Concatenate,Dropout,LayerNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras import Model

tf.keras.backend.clear_session()
input1=Input(shape=(max_length,),name='input_text')
embed = Embedding(vocab_size_text,200,input_length=max_length,name='embedding',
                      trainable=False,weights=[embedding_matrix])(input1)

input2=Input(shape=(max_length_sentiment,),name='input_sentiment')
```

```
embed2=Embedding(vocab_size_sentiment,200,input_length=max_length_sentiment,trainable=False,
                 weights=[embedding_matrix_sentiment],name='embedding_sentiment')(input2)
concat1=Concatenate(axis=1)([embed,embed2])
gru_1=GRU(100,name='GRU_1',return_sequences=True)(concat1)
gru_2=GRU(32,name='GRU_2',return_sequences=True)(gru_1)
gru_3=GRU(16,name='GRU_3',return_sequences=True)(gru_2)
f1=Flatten()(gru_3)

f1=tf.expand_dims(f1,1)
dense2=Dense(256,activation='relu',kernel_regularizer=l2(0.0001))(f1)
drop1 = Dropout(0.2)(dense2)
ln1= LayerNormalization()(drop1)
dense3=Dense(128,activation='relu',kernel_regularizer=l2(0.0001))(ln1)
drop2 = Dropout(0.2)(dense3)
ln2= LayerNormalization()(drop2)
dense4=Dense(64,activation='relu',kernel_regularizer=l2(0.0001))(ln1)
output=Dense(33,activation='sigmoid',name='output')(dense4)

model=Model(inputs=[input1,input2],outputs=[output])



model1=Model(inputs=[input1,input2],outputs=[output])


tf.keras.utils.plot_model(model,show_shapes=True)
```




```
!pip install git+https://github.com/qubvel/segmentation_models
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-r8mbd3jc
  Running command git clone -q https://github.com/qubvel/segmentation_models /tmp/pip-req-build-r8mbd3jc
  Running command git submodule update --init --recursive -q
Collecting keras_applications<=1.0.8,>=1.0.7
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 7.0 MB/s
Collecting image-classifiers==1.0.0
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting efficientnet==1.0.0
```

```
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages (from efficientnet==1.0.0->segmentation-models==1.0.1) (0.18.3)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentation-models==1.0.1) (3.1.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentation-models==1.0.1) (1.21.6)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py->keras_applications<=1.0.8,>=1.0.7->segmentation-models==1.0.1) (1.5.2)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (7.1.2)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (2.6.3)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (2.9.0)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (3.2.2)
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (1.7.3)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (2021.11.2)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (1.3.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmen
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1) (0.11.
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models==1.0.1
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentati
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-mod
Building wheels for collected packages: segmentation-models
  Building wheel for segmentation-models (setup.py) ... done
  Created wheel for segmentation-models: filename=segmentation_models-1.0.1-py3-none-any.whl size=33810 sha256=40f7011bde749902dbfc169f5b2eb8bcbc059b8d539da856f9ad7dc5a65a3ebc
  Stored in directory: /tmp/pip-ephem-wheel-cache-6tw89nl8/wheels/02/cd/18/61c0bbb8766acfec68f9d20618886b7b38dfeeb95865b6ba00
Successfully built segmentation-models
Installing collected packages: keras-applications, image-classifiers, efficientnet, segmentation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-1.0.8 segmentation-models-1.0.1
```

```
%load_ext tensorboard
tf.config.run_functions_eagerly(True)
import datetime
import os
import segmentation_models as sm
import math
from tensorflow.keras.callbacks import LearningRateScheduler
'''
def step_decay(epoch):
    initial_lrate = 0.0001
    drop = 0.1
    epochs_drop = 3
    lrate = initial_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
    return lrate

lrate = LearningRateScheduler(step_decay)
'''

focal_loss=sm.losses.DiceLoss(per_image=True)
iou_score=sm.metrics.IOUScore(threshold=0.5)

log_dir= "/content/drive/MyDrive/case_study_2_new/base_model_TBlog1"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True)
callbacks=[tensorboard_callback,
          tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor=0.1,patience=2,verbose=1,mode='min',min_lr=0.00001),
                    tf.keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/case_study_2_new/base_model_modelckpt1',monitor='val_loss',verbose=1,
                                                       save_best_only=True,save_weights_only=True)
]
model.compile(optimizer=tf.keras.optimizers.Adam(0.001),loss=focal_loss,metrics=[iou_score])
```

```
     The tensorboard extension is already loaded. To reload it, use:
       %reload_ext tensorboard
```

```
import warnings
warnings.filterwarnings('ignore')


train_input=(np.array(train_text),np.array(train_sentiment))
train_output=y_train[:,np.newaxis,:]
test_input=(np.array(test_text),np.array(test_sentiment))
test_output=y_test[:,np.newaxis,:]
validation=(test_input,test_output)
model.fit(train_input,
          train_output,epochs=40,
          validation_data=validation,callbacks=callbacks)
```

```
     Epoch 1/40
     687/687 [==============================] - ETA: 0s - loss: 0.4070 - iou_score: 0.5193
     Epoch 1: val_loss improved from inf to 0.36999, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 56s 81ms/step - loss: 0.4070 - iou_score: 0.5193 - val_loss: 0.3700 - val_iou_score: 0.5404 - lr: 0.0010
     Epoch 2/40
     687/687 [==============================] - ETA: 0s - loss: 0.3744 - iou_score: 0.5351
     Epoch 2: val_loss improved from 0.36999 to 0.36246, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 55s 80ms/step - loss: 0.3744 - iou_score: 0.5351 - val_loss: 0.3625 - val_iou_score: 0.5415 - lr: 0.0010
     Epoch 3/40
     687/687 [==============================] - ETA: 0s - loss: 0.3691 - iou_score: 0.5363
     Epoch 3: val_loss did not improve from 0.36246
     687/687 [==============================] - 50s 73ms/step - loss: 0.3691 - iou_score: 0.5363 - val_loss: 0.3630 - val_iou_score: 0.5400 - lr: 0.0010
     Epoch 4/40
     687/687 [==============================] - ETA: 0s - loss: 0.3679 - iou_score: 0.5397
     Epoch 4: val_loss improved from 0.36246 to 0.35877, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 51s 74ms/step - loss: 0.3679 - iou_score: 0.5397 - val_loss: 0.3588 - val_iou_score: 0.5436 - lr: 0.0010
     Epoch 5/40
     687/687 [==============================] - ETA: 0s - loss: 0.3653 - iou_score: 0.5459
     Epoch 5: val_loss improved from 0.35877 to 0.35606, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 50s 73ms/step - loss: 0.3653 - iou_score: 0.5459 - val_loss: 0.3561 - val_iou_score: 0.5685 - lr: 0.0010
     Epoch 6/40
     687/687 [==============================] - ETA: 0s - loss: 0.3611 - iou_score: 0.5552
     Epoch 6: val_loss improved from 0.35606 to 0.35117, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 50s 72ms/step - loss: 0.3611 - iou_score: 0.5552 - val_loss: 0.3512 - val_iou_score: 0.5706 - lr: 0.0010
     Epoch 7/40
     687/687 [==============================] - ETA: 0s - loss: 0.3556 - iou_score: 0.5700
     Epoch 7: val_loss improved from 0.35117 to 0.34864, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 50s 72ms/step - loss: 0.3556 - iou_score: 0.5700 - val_loss: 0.3486 - val_iou_score: 0.5629 - lr: 0.0010
     Epoch 8/40
     687/687 [==============================] - ETA: 0s - loss: 0.3511 - iou_score: 0.5787
     Epoch 8: val_loss improved from 0.34864 to 0.34667, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 51s 74ms/step - loss: 0.3511 - iou_score: 0.5787 - val_loss: 0.3467 - val_iou_score: 0.5736 - lr: 0.0010
     Epoch 9/40
     687/687 [==============================] - ETA: 0s - loss: 0.3483 - iou_score: 0.5855
     Epoch 9: val_loss improved from 0.34667 to 0.34613, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 50s 73ms/step - loss: 0.3483 - iou_score: 0.5855 - val_loss: 0.3461 - val_iou_score: 0.6020 - lr: 0.0010
     Epoch 10/40
     687/687 [==============================] - ETA: 0s - loss: 0.3439 - iou_score: 0.5998
     Epoch 10: val_loss improved from 0.34613 to 0.34303, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
     687/687 [==============================] - 54s 79ms/step - loss: 0.3439 - iou_score: 0.5998 - val_loss: 0.3430 - val_iou_score: 0.5975 - lr: 0.0010
     Epoch 11/40
     687/687 [==============================] - ETA: 0s - loss: 0.3405 - iou_score: 0.6111
     Epoch 11: val_loss did not improve from 0.34303
     687/687 [==============================] - 51s 74ms/step - loss: 0.3405 - iou_score: 0.6111 - val_loss: 0.3433 - val_iou_score: 0.6109 - lr: 0.0010
     Epoch 12/40
```

```
687/687 [==============================] - ETA: 0s - loss: 0.3372 - iou_score: 0.6182
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 12: val_loss did not improve from 0.34303
687/687 [==============================] - 54s 79ms/step - loss: 0.3372 - iou_score: 0.6182 - val_loss: 0.3469 - val_iou_score: 0.5962 - lr: 0.0010
Epoch 13/40
687/687 [==============================] - ETA: 0s - loss: 0.3311 - iou_score: 0.6255
Epoch 13: val_loss improved from 0.34303 to 0.34061, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
687/687 [==============================] - 50s 72ms/step - loss: 0.3311 - iou_score: 0.6255 - val_loss: 0.3406 - val_iou_score: 0.6148 - lr: 1.0000e-04
Epoch 14/40
687/687 [==============================] - ETA: 0s - loss: 0.3265 - iou_score: 0.6348
Epoch 14: val_loss improved from 0.34061 to 0.33922, saving model to /content/drive/MyDrive/case_study_2_new/base_model_modelckpt1
687/687 [==============================] - 50s 73ms/step - loss: 0.3265 - iou_score: 0.6348 - val_loss: 0.3392 - val_iou_score: 0.6153 - lr: 1.0000e-04
```

```
model.save('/content/drive/MyDrive/case_study_2_new/base_model_weights/dl_model.h5')
```

```
model1.load_weights('/content/drive/MyDrive/case_study_2_new/base_model_weights/dl_model.h5')
```

```
X_train[['text','selected_text','start_index','end_index']]
```

| | text | selected_text | start_index | end_index |
|---|---|---|---|---|
| **0** | Press `Ctrl` on bottom right. It`s there. KY | Press `Ctrl` on bottom right. It`s there. KY | 0 | 7 |
| **1** | ah remember the days when you`d sleep in until... | loser | 21 | 21 |
| **2** | my momma is comin 2night ! 2morrow tennis day... | yuppie | 11 | 11 |
| **3** | I do that all the time | I do that all the time | 0 | 5 |
| **4** | We don`t feel too comfortable using it. It`s... | . It`s not awful, but pretty icky | 7 | 13 |
| **...** | ... | ... | ... | ... |
| **21979** | Should be drank with sugar and milk, not coff... | Should be drank with sugar and milk, not coffe... | 0 | 14 |
| **21980** | Thinks she`s getting sick..... | Thinks she`s getting sick..... | 0 | 3 |
| **21981** | Get Up, You are NOT old! What did you do?! =O | Get Up, You are NOT old! What did you do?! =O | 0 | 10 |
| **21982** | Ha Ha thanks Tom! I`m such a loser! Hopefully... | Ha Ha thanks Tom! I`m such a loser! Hopefully ... | 0 | 23 |
| **21983** | they can`t be in their carriers anymore? | they can`t be in their carriers anymore? | 0 | 6 |

21984 rows × 4 columns

## Saving necessary files

```
import joblib
joblib.dump(tokenizer_text,'/content/drive/MyDrive/case_study_2_new/base_model_tokenizer/tokenizer_text.pkl')
joblib.dump(tokenizer_sentiment,'/content/drive/MyDrive/case_study_2_new/base_model_tokenizer/tokenizer_sentiment.pkl')
```

```
['/content/drive/MyDrive/case_study_2_new/base_model_tokenizer/tokenizer_sentiment.pkl']
```

```
import joblib
tokenizer_text=joblib.load('/content/drive/MyDrive/case_study_2_new/base_model_tokenizer/tokenizer_text.pkl')
tokenizer_sentiment=joblib.load('/content/drive/MyDrive/case_study_2_new/base_model_tokenizer/tokenizer_sentiment.pkl')
```

## Jaccard score

```
train_prediction=model.predict((np.array(train_text),np.array(train_sentiment)))
test_prediction=model.predict((np.array(test_text),np.array(test_sentiment)))
```

```
687/687 [==============================] - 16s 23ms/step
172/172 [==============================] - 4s 21ms/step
```

```
tr_p=np.squeeze(train_prediction,1)
te_p=np.squeeze(test_prediction,1)
```

```
tr_p.shape
```

```
(21984, 33)
```

```
def prob_to_binary(x,threshold=0.5):
  lst=[]
  for i in x:
    if i>=threshold:
      lst.append(1)
    else:
      lst.append(0)
  return lst
```

```
tr_pred=[]
for j in tr_p:
  tr_pred.append(prob_to_binary(j))
tr_pred=np.array(tr_pred)
```

```
te_pred=[]
for j in te_p:
  te_pred.append(prob_to_binary(j))
te_pred=np.array(te_pred)
```

```
def pred_text(x):
    pred_array=x[0]
    text=x[1]
    text_list=x[1].split()
    max_len_list=len(text_list)
    indices=np.where(pred_array==1)[0]
    indices=[ind for ind in indices if ind<max_len_list]
    pred_text_list=np.array(text_list)[indices]
    pred_text=' '.join(pred_text_list)
    return pred_text
```

```
train_pred_text=[]
for i in range(len(tr_pred)):
  txt=X_train['text'].iloc[i]
  pred_arr=tr_pred[i]
```

```
    pred_txt=pred_text((pred_arr,txt))
    train_pred_text.append(pred_txt)

test_pred_text=[]
for i in range(len(te_pred)):
    txt=X_test['text'].iloc[i]
    pred_arr=te_pred[i]
    pred_txt=pred_text((pred_arr,txt))
    test_pred_text.append(pred_txt)


X_train['pred_text']=train_pred_text
X_test['pred_text']=test_pred_text


def jaccard(x):
    str1=x[0]
    str2=x[1]
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    if (len(a)==0) & (len(b)==0):
        return 0.5
    c = a.intersection(b)

    return float(len(c)) / (len(a) + len(b) - len(c))


X_train
```

| | text | selected_text | preprocessed_texts | sentiment | start_index | end_index | pred_text |
|---|---|---|---|---|---|---|---|
| **0** | Press `Ctrl` on bottom right. It`s there. KY | Press `Ctrl` on bottom right. It`s there. KY | press `ctrl` on bottom right. it`s there. ky | neutral | 0 | 7 | Press `Ctrl` on bottom right. It`s there. KY |
| **1** | ah remember the days when you`d sleep in until... | loser | ah remember the days when you`d sleep in until... | negative | 21 | 21 | ah remember god i feel like a loser |
| **2** | my momma is comin 2night ! 2morrow tennis day... | yuppie | my momma is comin 2night ! 2morrow tennis day ... | positive | 11 | 11 | my momma p?nar yuppie ! |
| **3** | I do that all the time | I do that all the time | i do that all the time | neutral | 0 | 5 | I do that all the time |
| **4** | We don`t feel too comfortable using it. It`s... | . It`s not awful, but pretty icky | we don`t feel too comfortable using it. it`s n... | negative | 7 | 13 | We don`t not awful, but pretty icky. Scurrying |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **21979** | Should be drank with sugar and milk, not coff... | Should be drank with sugar and milk, not coffe... | should be drank with sugar and milk, not coffe... | neutral | 0 | 14 | Should be drank with sugar and milk, not coffe... |
| **21980** | Thinks she`s getting sick..... | Thinks she`s getting sick..... | thinks she`s getting sick..... | negative | 0 | 3 | Thinks she`s getting sick..... |
| **21981** | Get Up, You are NOT old! What did you do?! =O | Get Up, You are NOT old! What did you do?! =O | get up, you are not old! what did you do?! =o | neutral | 0 | 10 | Get Up, You are NOT old! What did you do?! =O |
| **21982** | Ha Ha thanks Tom! I`m such a loser! Hopefully... | Ha Ha thanks Tom! I`m such a loser! Hopefully ... | ha ha thanks tom! i`m such a loser! hopefully ... | neutral | 0 | 23 | Ha Ha thanks Tom! I`m such a loser! Hopefully ... |
| **21983** | they can`t be in their carriers anymore? | they can`t be in their carriers anymore? | they can`t be in their carriers anymore? | neutral | 0 | 6 | they can`t be in their carriers anymore? |

21984 rows × 7 columns

```
X_train['jaccard_score']=X_train[['selected_text','pred_text']].apply(lambda x: jaccard(x),axis=1)


X_test['jaccard_score']=X_test[['selected_text','pred_text']].apply(lambda x: jaccard(x),axis=1)


jacc_pos_neg_train=X_train[X_train['sentiment']!='neutral']['jaccard_score']
jacc_pos_neg_test=X_test[X_test['sentiment']!='neutral']['jaccard_score']
print('Jaccard score of positive and negative sentences for train data:',np.array(jacc_pos_neg_train).mean())
print('Jaccard score of positive and negative sentences for test data:',np.array(jacc_pos_neg_test).mean())
```

```
Jaccard score of positive and negative sentences for train data: 0.4170387329309261
Jaccard score of positive and negative sentences for test data: 0.35966433494944866
```

```
jacc_neut_train=X_train[X_train['sentiment']=='neutral']['jaccard_score']
jacc_neut_test=X_test[X_test['sentiment']=='neutral']['jaccard_score']
print('Jaccard score of neutral sentences for train data:',np.array(jacc_neut_train).mean())
print('Jaccard score of neutral sentences for test data:',np.array(jacc_neut_test).mean())
```

```
Jaccard score of neutral sentences for train data: 0.9594261469409443
Jaccard score of neutral sentences for test data: 0.9623107955588975
```

```
print("Overall Train Jaccard Score",np.array(X_train['jaccard_score']).mean())
print("Overall Test Jaccard Score",np.array(X_test['jaccard_score']).mean())
```

```
Overall Train Jaccard Score 0.6364707589591785
Overall Test Jaccard Score 0.6034203542243404
```

```
%load_ext tensorboard
%tensorboard --logdir /content/drive/MyDrive/case_study_2_new/base_model_TBlog1
```

Here the overall Jaccard score is 60.34 for test dataset.