```
import pandas as pd
import numpy as np
from tqdm import tqdm
from tqdm.notebook import tqdm_notebook
tqdm_notebook.pandas()
import warnings
warnings.filterwarnings('ignore')
```

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

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

```
! cp '/content/drive/My Drive/tweet-sentiment-extraction/preprocessed_train.csv' .
! cp '/content/drive/My Drive/tweet-sentiment-extraction/preprocessed_test.csv' .
```

```
train_df = pd.read_csv('preprocessed_train.csv')
test_df = pd.read_csv('preprocessed_test.csv')
```

```
train_df.shape,test_df.shape
```

```
((27469, 7), (3534, 3))
```

```
train_df.sample(5)
```

	textID	text	selected_text	sentiment	misspelled	start_indices	end_indices
14276	22d7ee60ad	boo you can come over and we`ll watch telenove...	boo you can come over and we`ll watch telenove...	neutral	No	0	13
17125	d0c8149986	sicky sicky sucks on such a lovely day	sicky sicky sucks on such a lovely day	neutral	No	0	7
15146	6e8a9b822b	just got back from seeing star strek	just got back from seeing star strek	neutral	No	0	6
3235	fc96e25ebd	back to my interesting emails	back to my interesting emails	positive	No	0	4
7907	542d1f7b68	beating heat with tea try some masala chaas	beating heat with tea try some masala chaas	neutral	No	0	7

```
test_df.sample(5)
```

	textID	text	sentiment
3290	edef02e047	wow that`s a big list lol i would be happy if ...	neutral
2444	07b995a175	storming outside	neutral
1951	a29a76a54c	so sad i have to pay	negative
3263	dec8c3dac3	happy mothers day mamma	positive
2758	850681c6b2	is feeling sick oh well i reckon those people ...	negative

```
train_df[train_df.end_indices<train_df.start_indices ]
```

textID	text	selected_text	sentiment	misspelled	start_indices	end_indices

```
X = train_df[['text','selected_text','sentiment','start_indices','end_indices']]
```

```
lens=[]
for each in X.text.values:
  lens.append(len(each.split()))
```

```
print('max length of sentence:',max(lens))
```

```
max length of sentence: 32
```

For each input text, we are gonna create a output vector in such a way that, the words which are part of selected text will be given a value of 1 and others will be given a value of 0

Example : text ---------> 'I am not happy with the kind of service'

selected_text--> 'not happy'

output -------> 0 0 1 1 0 0 0 0 0

Since the max length of input sentences are 32, output vector will be a 32 dimensional vector

```
Y = np.zeros((X.shape[0],max(lens)+1))
for i,each in tqdm(enumerate(X.values)):
  start = each[3]
  end = each[4]
  Y[i][start:end+1] = 1
```

```
27469it [00:00, 525454.87it/s]
```

```
#Cross checking whether the code has worked correctly.
import random
for _ in range(5):
  x = random.randint(0,train_df.shape[0])
  print('Data:',X.values[x])
  print('o/p vector:',Y[x])
  print('='*50)
```

```
Data: ['awesome effort this w e even if u didnt win good luck at tassie'
 'awesome effort' 'positive' 0 1]
o/p vector: [1. 1. 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.]
==================================================
Data: ['links widescreen laptop rotation comics awesome' 'awesome' 'positive' 5
 5]
o/p vector: [0. 0. 0. 0. 0. 1. 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.]
```

```
X.shape,Y.shape
```

```
((27469, 5), (27469, 33))

Data: ['my brother is planning on moving to vietnam and staying there forever'
```

```
from sklearn.model_selection import train_test_split
x_train,x_val,y_train,y_val= train_test_split(X,Y,test_size=0.20,random_state=42)
x_train.shape,x_val.shape,y_train.shape,y_val.shape
```

```
((21975, 5), (5494, 5), (21975, 33), (5494, 33))

 'yay mom bought me the sakura bodyshop lotion' 'neutral' 0 7]
```

```
y_train=np.expand_dims(y_train,-1)
y_val = np.expand_dims(y_val,-1)
y_train.shape,y_val.shape
```

```
((21975, 33, 1), (5494, 33, 1))
```

```
train_text = x_train['text'].values
val_text = x_val['text'].values
```

```
import os
if  'glove.6B.300d.txt' not in os.listdir('/content/'):
  ! cp '/content/drive/My Drive/tweet-sentiment-extraction/glove.6B.300d.txt' .
```

```
words_all = []
for each in train_text:
  words_all.extend(each.split())
len(words_all)
```

```
278997
```

```
from collections import Counter
a = Counter(words_all)
vals = list(a.values())
print('Total No.of values',len(vals))
count = len([i for i in vals if i<=5])
print('No of words with count less than 5',count)
count = len([i for i in vals if i<=2])
print('No of words with count less than 2',count)
count = len([i for i in vals if i<2])
print('No of words with count of only 1',count)
```

```
Total No.of values 21654
No of words with count less than 5 18358
No of words with count less than 2 15680
No of words with count of only 1 12766
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_text = Tokenizer(num_words = 15000, lower=True,split=' ',filters='!"#$%&()*+,-./:;<=>?@[\\]^_{|}~\t\n',oov_token='oov')
tokenizer_text.fit_on_texts(train_text)
train_text=tokenizer_text.texts_to_sequences(train_text)
val_text=tokenizer_text.texts_to_sequences(val_text)
print(len(train_text),len(val_text))
vocab_size_text=len(tokenizer_text.word_index)+1
print(vocab_size_text)
print(tokenizer_text.word_index)
```

```
21975 5494
21656
{'oov': 1, 'i': 2, 'to': 3, 'the': 4, 'a': 5, 'my': 6, 'and': 7, 'you': 8, 'it': 9, 'is': 10, 'in': 11, 'for': 12, 'of': 13, 'on': 14, 'me': 15, 'so': 16, 'have': 17, 'that': 18, 'but': 1
```

```
#Check the max index value(No.of unique words)
max=0
for each in train_text:
  for x in each:
    if x>=max:
      max=x
print('Max index',max)
```

```
Max index 14999
```

```
max_length_text=32
from tensorflow.keras.preprocessing.sequence import pad_sequences
train_text = pad_sequences(train_text,maxlen=max_length_text,padding='post')
val_text = pad_sequences(val_text,maxlen=max_length_text,padding='post')
print(train_text.shape,val_text.shape)
```

```
(21975, 32) (5494, 32)
```

```
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
from numpy import asarray
from numpy import zeros
embeddings_index = dict()
with open('/content/glove.6B.300d.txt') as f:
  for line in f:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs

print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

```
embedding_matrix = zeros((vocab_size_text, 300))
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

print(embedding_matrix.shape)
```

```
(21656, 300)
```

```
train_sentiment = x_train['sentiment'].values
val_sentiment = x_val['sentiment'].values
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_sentiment = Tokenizer(lower=True,split=' ',filters='!"#$%&()*+,-./:;<=>?@[\\]^_{|}~\t\n',oov_token='oov')
tokenizer_sentiment.fit_on_texts(train_sentiment)
train_sentiment=tokenizer_sentiment.texts_to_sequences(train_sentiment)
val_sentiment=tokenizer_sentiment.texts_to_sequences(val_sentiment)
print(len(train_sentiment),len(val_sentiment))
print(tokenizer_sentiment.word_index)
vocab_size_sentiment=len(tokenizer_sentiment.word_index)+1
print(vocab_size_sentiment)
```

```
21975 5494
{'oov': 1, 'neutral': 2, 'positive': 3, 'negative': 4}
5
```

```
max_length_sentiment=1
from tensorflow.keras.preprocessing.sequence import pad_sequences
train_sentiment = pad_sequences(train_sentiment,maxlen=max_length_sentiment,padding='post')
val_sentiment = pad_sequences(val_sentiment,maxlen=max_length_sentiment,padding='post')
print(train_sentiment.shape,val_sentiment.shape)
```

```
(21975, 1) (5494, 1)
```

```
#https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/
```

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Embedding,Dense,Dropout,Concatenate,Flatten,TimeDistributed,Input,GRU,BatchNormalization,Bidirectional,SpatialDropout1D,LSTM,Layer
from tensorflow.keras.regularizers import l2
```

```
batch_size=128
```

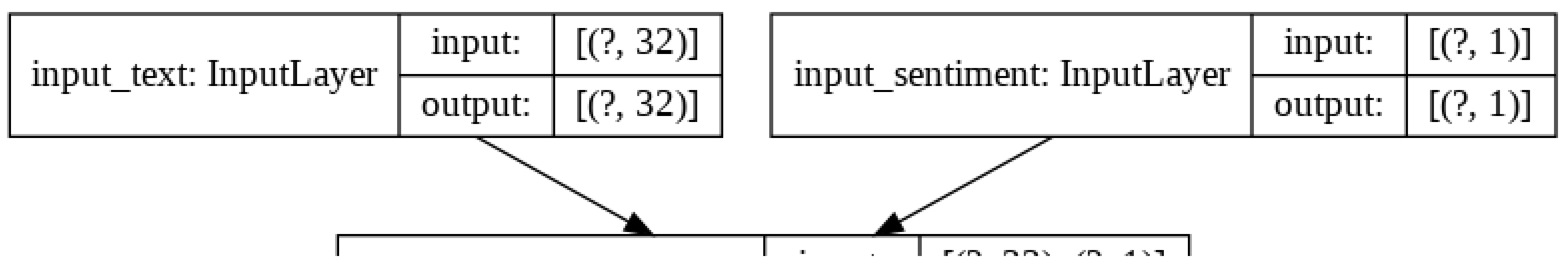
```
input1=Input(shape=(max_length_text,),name='input_text')
input2=Input(shape=(max_length_sentiment,),name='input_sentiment')
concat= Concatenate()([input1,input2])
embed = Embedding(vocab_size_text,300,input_length=max_length_text,name='embedding',\
                      trainable=False,mask_zero = True,embeddings_initializer=tf.constant_initializer(embedding_matrix))(concat)
gru=Bidirectional(GRU(16,name='gru',return_sequences=True,dropout=0.4))(embed)
```

```
dense1 = Dense(8,activation='relu',kernel_regularizer=l2(0.0001))(gru)
dp = Dropout(0.5)(dense1)
dense1 = Dense(4,activation='relu',kernel_regularizer=l2(0.0001))(dp)
output=TimeDistributed(Dense(1,activation='sigmoid'))(dense1)
```

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

```
for each in model.layers:
  if(type(each) == tf.keras.layers.Embedding):
    each.trainable = False
```

```
import tensorflow as tf
tf.keras.utils.plot_model(model, 'Model.png',show_shapes=True)
```



```
model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_text (InputLayer)         [(None, 32)]         0
__________________________________________________________________________________________________
input_sentiment (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 33)           0           input_text[0][0]
                                                                 input_sentiment[0][0]
__________________________________________________________________________________________________
embedding (Embedding)           (None, 33, 300)      6496800     concatenate[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 33, 32)       30528       embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 33, 8)        264         bidirectional[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 33, 8)        0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 33, 4)        36          dropout[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 33, 1)        5           dense_1[0][0]
==================================================================================================
Total params: 6,527,633
Trainable params: 30,833
Non-trainable params: 6,496,800
__________________________________________________________________________________________________
```

```
input_data = (train_text,train_sentiment)
output_data = y_train

val = (val_text,val_sentiment)
output_val = y_val
val_data = (val,output_val)
```

```
! rm -r '/content/checkpt'
! rm -r '/content/tensorboard_logs1'
```

```
rm: cannot remove '/content/checkpt': No such file or directory
rm: cannot remove '/content/tensorboard_logs1': No such file or directory
```

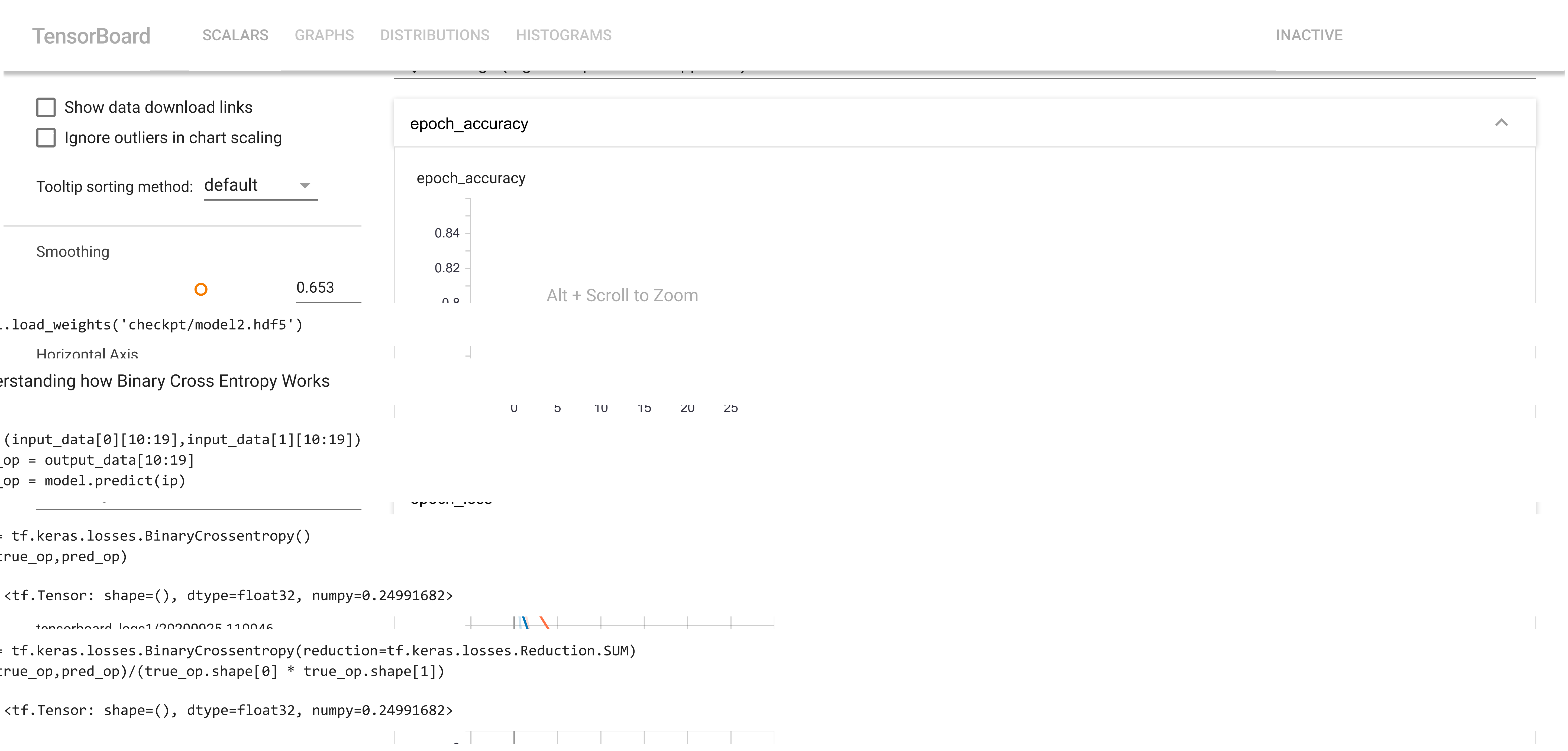
```
%load_ext tensorboard
import datetime
import os
log_dir= os.path.join("tensorboard_logs1" , datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True)
! mkdir  'checkpt'
file_path = os.path.join('checkpt/model2.hdf5')
checkpt_save = tf.keras.callbacks.ModelCheckpoint(filepath=file_path,save_weights_only=True,monitor='val_loss',save_best_only=True,verbose=1)
callbacks=[tensorboard_callback,checkpt_save]
```

```
def my_loss(true,pred):
  #print(true.shape,pred.shape)

  loss_obj = tf.keras.losses.BinaryCrossentropy(reduction=tf.keras.losses.Reduction.SUM)
  loss = loss_obj (true,pred)
  return loss/128 #batch size


#loss_fn = tf.keras.losses.BinaryCrossentropy()
model.compile(optimizer='adam',loss=my_loss,metrics=['accuracy'])
```

```
model.fit(input_data,output_data,epochs=30,batch_size=128,validation_data=val_data,callbacks=callbacks)
```

```
=============================] - 17s 98ms/step - loss: 2.5171 - accuracy: 0.8077 - val_loss: 2.1096 - val_accuracy: 0.8450
0
=============================] - ETA: 0s - loss: 2.5131 - accuracy: 0.8068
7: val_loss did not improve from 2.10965
=============================] - 16s 95ms/step - loss: 2.5131 - accuracy: 0.8068 - val_loss: 2.1280 - val_accuracy: 0.8435
0
=============================] - ETA: 0s - loss: 2.5006 - accuracy: 0.8085
8: val_loss did not improve from 2.10965
=============================] - 16s 92ms/step - loss: 2.5006 - accuracy: 0.8085 - val_loss: 2.1272 - val_accuracy: 0.8436
0
=============================] - ETA: 0s - loss: 2.4877 - accuracy: 0.8082
9: val_loss improved from 2.10965 to 2.10628, saving model to checkpt/model2.hdf5
=============================] - 16s 91ms/step - loss: 2.4877 - accuracy: 0.8082 - val_loss: 2.1063 - val_accuracy: 0.8443
0
=============================] - ETA: 0s - loss: 2.4798 - accuracy: 0.8088
0: val_loss improved from 2.10628 to 2.10341, saving model to checkpt/model2.hdf5
=============================] - 16s 91ms/step - loss: 2.4798 - accuracy: 0.8088 - val_loss: 2.1034 - val_accuracy: 0.8448
0
=============================] - ETA: 0s - loss: 2.4822 - accuracy: 0.8082
1: val_loss did not improve from 2.10341
=============================] - 16s 91ms/step - loss: 2.4822 - accuracy: 0.8082 - val_loss: 2.1370 - val_accuracy: 0.8433
0
=============================] - ETA: 0s - loss: 2.4703 - accuracy: 0.8085
2: val_loss did not improve from 2.10341
=============================] - 16s 91ms/step - loss: 2.4703 - accuracy: 0.8085 - val_loss: 2.1241 - val_accuracy: 0.8446
0
=============================] - ETA: 0s - loss: 2.4649 - accuracy: 0.8091
3: val_loss did not improve from 2.10341
=============================] - 16s 91ms/step - loss: 2.4649 - accuracy: 0.8091 - val_loss: 2.1322 - val_accuracy: 0.8447
0
=============================] - ETA: 0s - loss: 2.4587 - accuracy: 0.8094
4: val_loss did not improve from 2.10341
=============================] - 16s 92ms/step - loss: 2.4587 - accuracy: 0.8094 - val_loss: 2.1468 - val_accuracy: 0.8447
0
=============================] - ETA: 0s - loss: 2.4480 - accuracy: 0.8093
5: val_loss did not improve from 2.10341
=============================] - 17s 100ms/step - loss: 2.4480 - accuracy: 0.8093 - val_loss: 2.1246 - val_accuracy: 0.8439
0
=============================] - ETA: 0s - loss: 2.4433 - accuracy: 0.8094
6: val_loss did not improve from 2.10341
=============================] - 16s 92ms/step - loss: 2.4433 - accuracy: 0.8094 - val_loss: 2.1979 - val_accuracy: 0.8454
0
=============================] - ETA: 0s - loss: 2.4368 - accuracy: 0.8104
7: val_loss did not improve from 2.10341
=============================] - 16s 91ms/step - loss: 2.4368 - accuracy: 0.8104 - val_loss: 2.1298 - val_accuracy: 0.8453
0
=============================] - ETA: 0s - loss: 2.4244 - accuracy: 0.8111
8: val_loss did not improve from 2.10341
=============================] - 15s 90ms/step - loss: 2.4244 - accuracy: 0.8111 - val_loss: 2.1865 - val_accuracy: 0.8429
0
=============================] - ETA: 0s - loss: 2.4201 - accuracy: 0.8103
9: val_loss did not improve from 2.10341
=============================] - 15s 90ms/step - loss: 2.4201 - accuracy: 0.8103 - val_loss: 2.1750 - val_accuracy: 0.8440
0
=============================] - ETA: 0s - loss: 2.4166 - accuracy: 0.8112
0: val_loss did not improve from 2.10341
=============================] - 15s 90ms/step - loss: 2.4166 - accuracy: 0.8112 - val_loss: 2.1396 - val_accuracy: 0.8444
w.python.keras.callbacks.History at 0x7fa3cc7ec0f0>
```

```
tf.keras.backend.clear_session()
%tensorboard --logdir $log_dir --port 0
```



```
model.load_weights('checkpt/model2.hdf5')
```

Understanding how Binary Cross Entropy Works

```
ip = (input_data[0][10:19],input_data[1][10:19])
true_op = output_data[10:19]
pred_op = model.predict(ip)
```

```
bce = tf.keras.losses.BinaryCrossentropy()
bce(true_op,pred_op)
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=0.24991682>
```

```
bce = tf.keras.losses.BinaryCrossentropy(reduction=tf.keras.losses.Reduction.SUM)
bce(true_op,pred_op)/(true_op.shape[0] * true_op.shape[1])
```

```
<tf.Tensor: shape=(), dtype=float32, numpy=0.24991682>
```

```
#input_data = (train_text,train_sentiment)
#output_data = y_train
#val = (val_text,val_sentiment)
#output_val = y_val
#val_data = (val,output_val)
```

For Training data

```
x_train = x_train[['text','selected_text','sentiment']]
x_train.shape
```

```
(21975, 3)
```

```
train_pred = model.predict(input_data)
train_pred = np.squeeze(train_pred)
train_pred = np.round(train_pred)
train_pred.shape
```

```
(21975, 33)
```

```
pred_output = []
for each in tqdm(train_pred):
  indices=[]
  for x in range(len(each)):
    if each[x] == 1:
      indices.append(x)
    else:
      continue

  indices = np.array(indices)
  pred_output.append(indices)

print(len(pred_output))
```

```
100%|██████████| 21975/21975 [00:01<00:00, 12411.92it/s]21975
```

```
x_train['prediction'] = pred_output
```

```
x_train
```

	text	selected_text	sentiment	prediction
22716	i really want a shish kebab going to have to s...	i really want a shish kebab going to have to s...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
1231	lol i thought we was suppose to guess curse i ...	lol i thought we was suppose to guess curse i ...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
531	today was a lovely day i had fun with and this...	lovely	positive	[3, 7]
17252	hey mate fancy finden you on hea	hey mate fancy finden you on hea	neutral	[0, 1, 2, 3, 4, 5, 6]
6334	home should be in the bed but i`m just super d...	super duper excited	positive	[10, 11]
...	...	...	...	...
21575	you have my vote want to see gino in drag plea...	you have my vote want to see gino in drag plea...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
5390	awake good midday	good	positive	[1]
860	links fire and urban at rock challenge	links fire and urban at rock challenge	neutral	[0, 1, 2, 3, 4, 5, 6]
15795	u witch im upstate in a curse hick dry county ...	u witch im upstate in a curse hick dry county ...	negative	[]
23654	awww we can do that and then go to chick fila	awww we can do that and then go to chick fila	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

21975 rows × 4 columns

```
def get_pred_text(x):
  pred = []
  text = x[0].split()
  indices = x[1]
  l = len(text)
  for each in indices:
    if each < l:
      pred.append(text[each])

  return pred
```

```
pred_text= x_train[['text','prediction']].progress_apply(lambda x:get_pred_text(x),axis=1)
```

```
100%                                    21975/21975 [00:01<00:00, 13913.93it/s]
```

```
x_train['pred_text'] = pred_text
x_train['pred_text'] = x_train['pred_text'].apply(lambda x: ' '.join(x))
x_train
```

	text	selected_text	sentiment	prediction	pred_text
22716	i really want a shish kebab going to have to s...	i really want a shish kebab going to have to s...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]	i really want a shish kebab going to have to s...
1231	lol i thought we was suppose to guess curse i ...	lol i thought we was suppose to guess curse i ...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]	lol i thought we was suppose to guess curse i ...
531	today was a lovely day i had fun with and this...	lovely	positive	[3, 7]	lovely fun
17252	hey mate fancy finden you on hea	hey mate fancy finden you on hea	neutral	[0, 1, 2, 3, 4, 5, 6]	hey mate fancy finden you on hea
6334	home should be in the bed but i`m just super d...	super duper excited	positive	[10, 11]	duper excited
...	...	...	...	...	...
21575	you have my vote want to see gino in drag plea...	you have my vote want to see gino in drag plea...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]	you have my vote want to see gino in drag plea...
5390	awake good midday	good	positive	[1]	good
860	links fire and urban at rock challenge	links fire and urban at rock challenge	neutral	[0, 1, 2, 3, 4, 5, 6]	links fire and urban at rock challenge
15795	u witch im upstate in a curse hick dry county ...	u witch im upstate in a curse hick dry county ...	negative	[]	
23654	awww we can do that and then go to chick fila	awww we can do that and then go to chick fila	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	awww we can do that and then go to chick fila

21975 rows × 5 columns

```
def jaccard_score(x):
    str1, str2 = str(x[0]),str(x[1])
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    return float(len(c)) / (len(a) + len(b) - len(c))
```

```
x_train['jaccard'] = x_train[['selected_text','pred_text']].progress_apply(jaccard_score,axis=1)
```

```
100%                                    21975/21975 [00:01<00:00, 17414.45it/s]
```

```
pos_data = x_train[x_train['sentiment'] == 'positive']
neg_data = x_train[x_train['sentiment'] == 'negative']
neu_data = x_train[x_train['sentiment'] =='neutral']
pos_data.shape,neg_data.shape,neu_data.shape
```

```
((6863, 6), (6240, 6), (8872, 6))
```

Jaccard scores for training data

```
print('Mean jaccard score for positive sentiment data:', np.mean(pos_data['jaccard']))
print('Mean jaccard score for negative sentiment data', np.mean(neg_data['jaccard']))
print('Mean jaccard score for neutral sentiment data', np.mean(neu_data['jaccard']))
```

```
Mean jaccard score for positive sentiment data: 0.4442868339615516
Mean jaccard score for negative sentiment data 0.41351098573474393
Mean jaccard score for neutral sentiment data 0.9838572677764533
```

For Validation data

```
x_val = x_val[['text','selected_text','sentiment']]
x_val.shape
```

```
(5494, 3)
```

```
val_pred = model.predict(val_data)
val_pred = np.squeeze(val_pred)
val_pred = np.round(val_pred)
val_pred.shape
```

```
(5494, 33)
```

```
val_pred_output = []
for each in tqdm(val_pred):
  indices=[]
  for x in range(len(each)):
    if each[x] == 1:
      indices.append(x)
    else:
      continue

  indices = np.array(indices)
  val_pred_output.append(indices)

print(len(val_pred_output))
```

```
100%|██████████| 5494/5494 [00:00<00:00, 11548.67it/s]5494
```

```
x_val['prediction'] = val_pred_output
```

```
x_val
```

	text	selected_text	sentiment	prediction
5875	i`m off for tonight good night everyone	good	positive	[4]
21879	i think my wireless router is dieing	dieing	negative	[4, 5, 6]
3308	so bored nothing to do	bored	negative	[0, 1]
23187	phillies gamee with mama for mothers day	phillies gamee with mama for mothers day	neutral	[0, 1, 2, 3, 4, 5, 6]
18229	sanctuarysunday yay for sanctuary i may watch ...	sanctuarysunday yay for sanctuary i may watch ...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
...	...	...	...	...
6814	has a gym day and is hoping to enjoy the last ...	enjoy	positive	[6, 8]
6165	haha not always just a day trip for a friend b...	haha not always just a day trip for a friend b...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...

```
def get_pred_text(x):
  pred = []
  text = x[0].split()
  indices = x[1]
  l = len(text)
  for each in indices:
    if each < l:
      pred.append(text[each])

  return pred
```

```
pred_text= x_val[['text','prediction']].progress_apply(lambda x:get_pred_text(x),axis=1)
```

```
100%                                    5494/5494 [00:00<00:00, 14096.74it/s]
```

```
x_val['pred_text'] = pred_text
x_val['pred_text'] = x_val['pred_text'].apply(lambda x: ' '.join(x))
x_val.sample(5)
```

	text	selected_text	sentiment	prediction	pred_text
7691	praying the rosary with my family	praying the rosary with my family	neutral	[0, 1, 2, 3, 4, 5]	praying the rosary with my family
23112	were you roaming same thing happen to me so i ...	were you roaming same thing happen to me so i ...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]	were you roaming same thing happen to me so i ...
24752	u have a lot but the bad thing is we r gonna h...	u have a lot but the bad thing	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	u have a lot but the bad thing is we r gonna h...
27072	can`t go to bed an am sooooo tired	tired	negative	[6, 7]	sooooo tired
16792	i think i hate you i didnt really want to but ...	you make it hard for me	negative	[3]	hate

```
def jaccard_score(x):
    str1, str2 = str(x[0]),str(x[1])
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    return float(len(c)) / (len(a) + len(b) - len(c))
```

```
x_val['jaccard'] = x_val[['selected_text','pred_text']].progress_apply(jaccard_score,axis=1)
```

```
100%                                    5494/5494 [00:00<00:00, 13613.96it/s]
```

```
x_val.sample(5)
```

	text	selected_text	sentiment	prediction	pred_text	jaccard
17316	had a blast this weekend with my sweet girls i...	awesome	positive	[13]	awesome	1.000000
11091	you can put a saucepan full of water on the co...	indian style scrabbled are the best	positive	[18, 19, 20, 21]	scrabbled are the best	0.666667
16953	is feeling sad i so dont do goodbye`z	sad	negative	[1, 2]	feeling sad	0.500000
4528	links i really love this picture	i really love this picture	positive	[2, 3, 4, 5]	really love this picture	0.800000
16570	got my grubby paws on a live recording of para...	better	positive	[]		0.000000

```
pos_data = x_val[x_val['sentiment'] == 'positive']
neg_data = x_val[x_val['sentiment'] == 'negative']
neu_data = x_val[x_val['sentiment'] =='neutral']
pos_data.shape,neg_data.shape,neu_data.shape
```

```
((1712, 6), (1538, 6), (2244, 6))
```

Jaccard scores for validation data

```
print('Mean jaccard score for positive sentiment data:', np.mean(pos_data['jaccard']))
print('Mean jaccard score for negative sentiment data', np.mean(neg_data['jaccard']))
print('Mean jaccard score for neutral sentiment data', np.mean(neu_data['jaccard']))
```

```
Mean jaccard score for positive sentiment data: 0.41748941470238576
Mean jaccard score for negative sentiment data 0.38164663464367576
Mean jaccard score for neutral sentiment data 0.9815631303707222
```

Analysis of Validation data

```
x_val['text_len'] = x_val['text'].progress_apply(lambda x: len(x.split()))
x_val['seltext_len'] = x_val['selected_text'].progress_apply(lambda x: len(x.split()))
x_val['diff'] = x_val['text_len'] -  x_val['seltext_len']
```

```
100%                                    5494/5494 [00:00<00:00, 83445.31it/s]

100%                                    5494/5494 [00:00<00:00, 56815.20it/s]
```

```
x_val
```

	text	selected_text	sentiment	prediction	pred_text	jaccard	text_len	seltext_len	diff
5875	i`m off for tonight good night everyone	good	positive	[4]	good	1.000000	7	1	6
21879	i think my wireless router is dieing	dieing	negative	[4, 5, 6]	router is dieing	0.333333	7	1	6
3308	so bored nothing to do	bored	negative	[0, 1]	so bored	0.500000	5	1	4
23187	phillies gamee with mama for mothers day	phillies gamee with mama for mothers day	neutral	[0, 1, 2, 3, 4, 5, 6]	phillies gamee with mama for mothers day	1.000000	7	7	0
18229	sanctuarysunday yay for sanctuary i may watch ...	sanctuarysunday yay for sanctuary i may watch ...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	sanctuarysunday yay for sanctuary i may watch ...	1.000000	10	10	0
...	...	...	...	...	...	...	...	...	...
6814	has a gym day and is hoping to enjoy the last ...	enjoy	positive	[6, 8]	hoping enjoy	0.500000	15	1	14
6165	haha not always just a day trip for a friend b...	haha not always just a day trip for a friend b...	neutral	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	haha not always just a day trip for a friend b...	1.000000	22	22	0
11238	beautiful monday morning so happy links	beautiful monday morning so happy	positive	[0, 4]	beautiful happy	0.400000	6	5	1
16970	finally watched the rest of the guild season i...	i always feel so sorry	negative	[10, 11, 12, 14]	feel so sorry hopeful	0.500000	22	5	17
16118	lol too bad he`s taken	bad	negative	[2]	bad	1.000000	5	1	4

```
x_val[x_val.sentiment =='neutral']['diff'].mean(),x_val[x_val.sentiment =='neutral']['diff'].median()
```

```
(0.27540106951871657, 0.0)
```

```
print(x_val[x_val.sentiment =='positive']['diff'].mean(),x_val[x_val.sentiment =='positive']['diff'].median())
```

```
9.392523364485982 9.0
```

```
print(x_val[x_val.sentiment =='negative']['diff'].mean(),x_val[x_val.sentiment =='negative']['diff'].median())
```

```
9.527958387516255 9.0
```

Inferences

- If you calculate the difference between the number of words in the text and selected_text columns for **Neutral** data, the mean and median values suggests that there is almost no difference in the no.of words in text and selected_text and the model performs well in this scenario
- From the above, the difference between number of words in text and selected_text columns for the Positive and Negative data are around 9.3 and 9.5 , which suggests that **as the difference between text and selected_text increases**, the model doesn't seem to perform better.