```
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')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
! 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
2736	9d4b1e13a3	jen lol i know seems that the whole fabric of ...	i know seems that the whole fabric of our eart...	neutral	No	2	22
14160	aa04ff026b	i ca not get to bed	i ca not get to bed	neutral	No	0	5
20797	34fc124590	final thought for the day does deodorant reall...	final thought for the day does deodorant reall...	neutral	No	0	19
3315	3c4f26ae13	well no phone today we are waiting till june t...	good	positive	No	18	18
966	72bafdda6a	i was born there	i was born there	neutral	No	0	3

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

	textID	text	sentiment
1483	9932b9c033	if i wasnt workin in hours id be gettin ratars...	negative
2323	166332e48a	i am oh so very bored buut almost days til i l...	negative
1061	afa094db6a	got alot of runnin around to do today to get t...	positive
1280	990c3cf35f	too sick for rigging tomorrow	negative
1261	c69ccdfa56	i am very much in tune with your words today t...	positive

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

	textID	text	selected_text	sentiment	misspelled	start_indices	end_indices
6393	ddbce5f751	this is great i just found out that it is star...	amay the be it	positive	No	13	8
13668	8607d4de1a	dans public transport again and have decided i...	utter curse that	negative	No	16	14

```
train_df = train_df[train_df.end_indices>=train_df.start_indices]
```

```
train_df = train_df[train_df.sentiment!='neutral']
```

```
train_df.shape
```

```
(16351, 7)
```

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

```
X.shape
```

```
(16351, 5)
```

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

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

```
max length of sentence: 33
```

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
```

```
16351it [00:00, 599301.48it/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: ['oh fack u gave me police thats some seriouse curse' 'curse' 'negative' 9
 9]
o/p vector: [0. 0. 0. 0. 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.]
==================================================
Data: ['two macaroons go into a bar one says oh your a nut wow i need to get out more'
 'two' 'positive' 0 0]
o/p vector: [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: ['thanks for the follow raises hand i am' 'thanks for the follow'
 'positive' 0 3]
o/p vector: [1. 1. 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.]
==================================================
Data: ['officially misses quark huhuhu i had no heart hand partner and no one to dance with tonight come home so links'
 'no one to dance with tonight' 'negative' 11 16]
o/p vector: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0.]
==================================================
Data: ['well the one thing where you lay on the floor i could not do all of them i am very out of shape'
 'i am very out of shape' 'negative' 17 22]
o/p vector: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0.]
==================================================
```

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

```
((16351, 5), (16351, 34))
```

```
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
```

```
((13080, 5), (3271, 5), (13080, 34), (3271, 34))
```

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

```
((13080, 34, 1), (3271, 34, 1))
```

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

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

```
import os
if 'cc.en.300.bin' not in os.listdir():
  ! cp '/content/drive/My Drive/tweet-sentiment-extraction/cc.en.300.bin' .
```

```
from gensim.models import FastText
fasttext_model = FastText.load_fasttext_format("/content/cc.en.300.bin")
```

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

```
177017
```

```
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 15234
No of words with count less than 5 12977
No of words with count less than 2 11031
No of words with count of only 1 8988
```

```
#np.savetxt(r'/content/mytext.txt', x_train['text'].values, fmt='%s')
#! cat '/content/mytext.txt' | head -10
```

```
#!pip install fasttext
#import fasttext
#ft_model = fasttext.train_unsupervised('/content/mytext.txt', model='skipgram',dim=300,minCount=1,minn=1)
#ft_model.get_word_vector('flabbergasted').shape
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_text = Tokenizer(num_words=16000,lower=True,split=' ',filters='!"#$%&()*+,-./:;<=>?@[\\]^_{|}~\t\n',oov_token='oov')
tokenizer_text.fit_on_texts(X['text'].values)
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)
```

```
13080 3271
17397
{'oov': 1, 'i': 2, 'to': 3, 'the': 4, 'is': 5, 'a': 6, 'my': 7, 'you': 8, 'it': 9, 'and': 10, 'not': 11, 'for': 12, 'in': 13, 'am': 14, 'that': 15, 'have': 16, 'of': 17, 'so': 18, 'me': 1
```

```
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
'''from numpy import asarray
from numpy import zeros
embeddings_index = dict()
with open('/content/cc.en.300.vec') as f:
#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))'''
```

```
'from numpy import asarray\nfrom numpy import zeros\nembeddings_index = dict()\nwith open('/content/cc.en.300.vec') as f:\n#with open('/content/glove.6B.300d.txt') as f:\n  for line in f:\n    values = line.split()\n    word = values[0]\n    coefs = asarray(values[1:], dtype='float32')\n    embeddings_index[word] = coefs\n    \nprint('Loaded %s word vectors.' % len(embeddings_index))'
```

```
embedding_matrix = np.zeros((vocab_size_text, 300))
for word, i in tokenizer_text.word_index.items():
  try:
    embedding_vector = fasttext_model.wv.word_vec(word)
    embedding_matrix[i] = embedding_vector
  except:
    continue
print(embedding_matrix.shape)
```

```
(17397, 300)
```

```
max_length_text=33
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)
```

```
(13080, 33) (3271, 33)
```

```
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)
```

```
13080 3271
{'oov': 1, 'positive': 2, 'negative': 3}
4
```

```
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)
```

```
(13080, 1) (3271, 1)
```

```
embedding_matrix1 = np.zeros((vocab_size_sentiment, 300))
for word, i in tokenizer_sentiment.word_index.items():
  try:
    embedding_vector = fasttext_model.wv.word_vec(word)
    embedding_matrix1[i] = embedding_vector
  except:
    continue
print(embedding_matrix1.shape)
```

```
(4, 300)
```

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

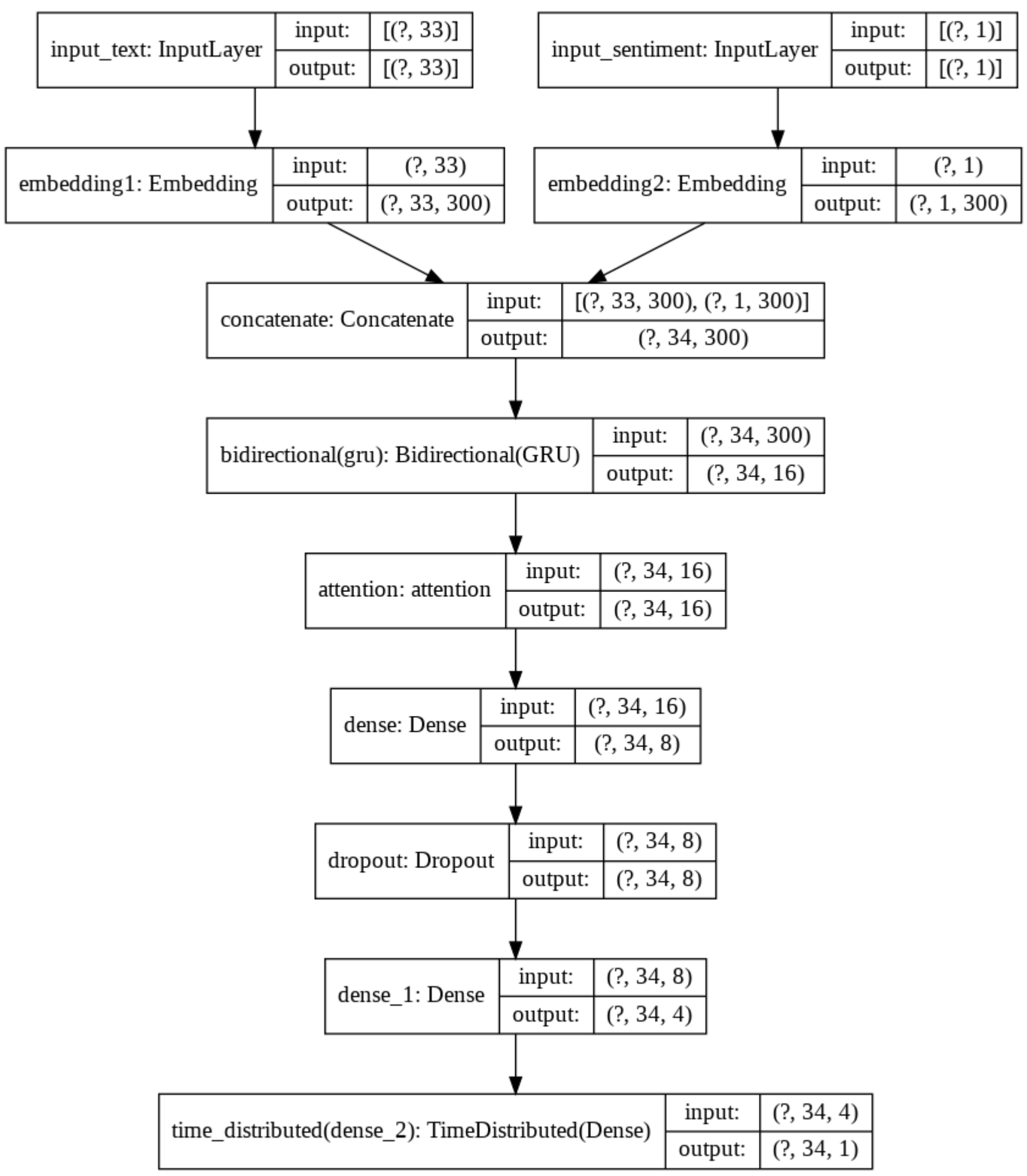
```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Embedding,Dense,Dropout,Concatenate,Flatten,TimeDistributed,Input,GRU,BatchNormalization,Bidirectional,SpatialDropout1D,LSTM,Layer
from tensorflow.keras.regularizers import l2
```

```
input1=Input(shape=(max_length_text,),name='input_text')
input2=Input(shape=(max_length_sentiment,),name='input_sentiment')

embed1 = Embedding(vocab_size_text,300,input_length=max_length_text,name='embedding1',\
                     trainable=False,mask_zero = True,embeddings_initializer=tf.constant_initializer(embedding_matrix))(input1)
embed2 = Embedding(vocab_size_sentiment,300,input_length=max_length_text,name='embedding2',\
                     trainable=False,mask_zero = True,embeddings_initializer=tf.constant_initializer(embedding_matrix1))(input2)
concat= Concatenate(axis=1)([embed1,embed2])
gru=Bidirectional(GRU(8,name='gru',return_sequences=True,dropout=0.4))(concat)


import tensorflow as tf
from tensorflow.keras.activations import  tanh
from tensorflow.keras.backend import dot
from tensorflow.keras.activations import softmax
from tensorflow.keras.backend import sum
class attention(tf.keras.layers.Layer):

    def __init__(self, return_sequences=True):
        self.return_sequences = return_sequences
        super().__init__()

    def build(self, input_shape):

        self.W=self.add_weight(name="att_weight", shape=(input_shape[-1],1),
                               initializer="normal")
        self.b=self.add_weight(name="att_bias", shape=(input_shape[1],1),
                               initializer="zeros")
```

```
        super(attention,self).build(input_shape)

    def call(self, x):

        e = tanh(dot(x,self.W)+self.b)
        a =  tf.nn.softmax(e, axis=1)
        output = x*a

        if self.return_sequences:
            return output

        return  tf.reduce_sum(output, axis=1)


#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',\
#                      mask_zero = True,embeddings_initializer=tf.constant_initializer(embedding_matrix))(concat)
#gru=Bidirectional(GRU(16,name='gru',return_sequences=True,dropout=0.4))(embed)


att = attention(return_sequences=True)(gru)
dense1 = Dense(8,activation='relu',kernel_regularizer=l2(0.0001))(att)
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, 33)]         0
__________________________________________________________________________________________________
input_sentiment (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________________________
embedding1 (Embedding)          (None, 33, 300)      5219100     input_text[0][0]
__________________________________________________________________________________________________
```

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

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

```
__________________________________________________________________________________________________
```

```
tf.keras.backend.clear_session()
```

```
dropout (Dropout)               (None, 34, 8)        0           dense[0][0]
```

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

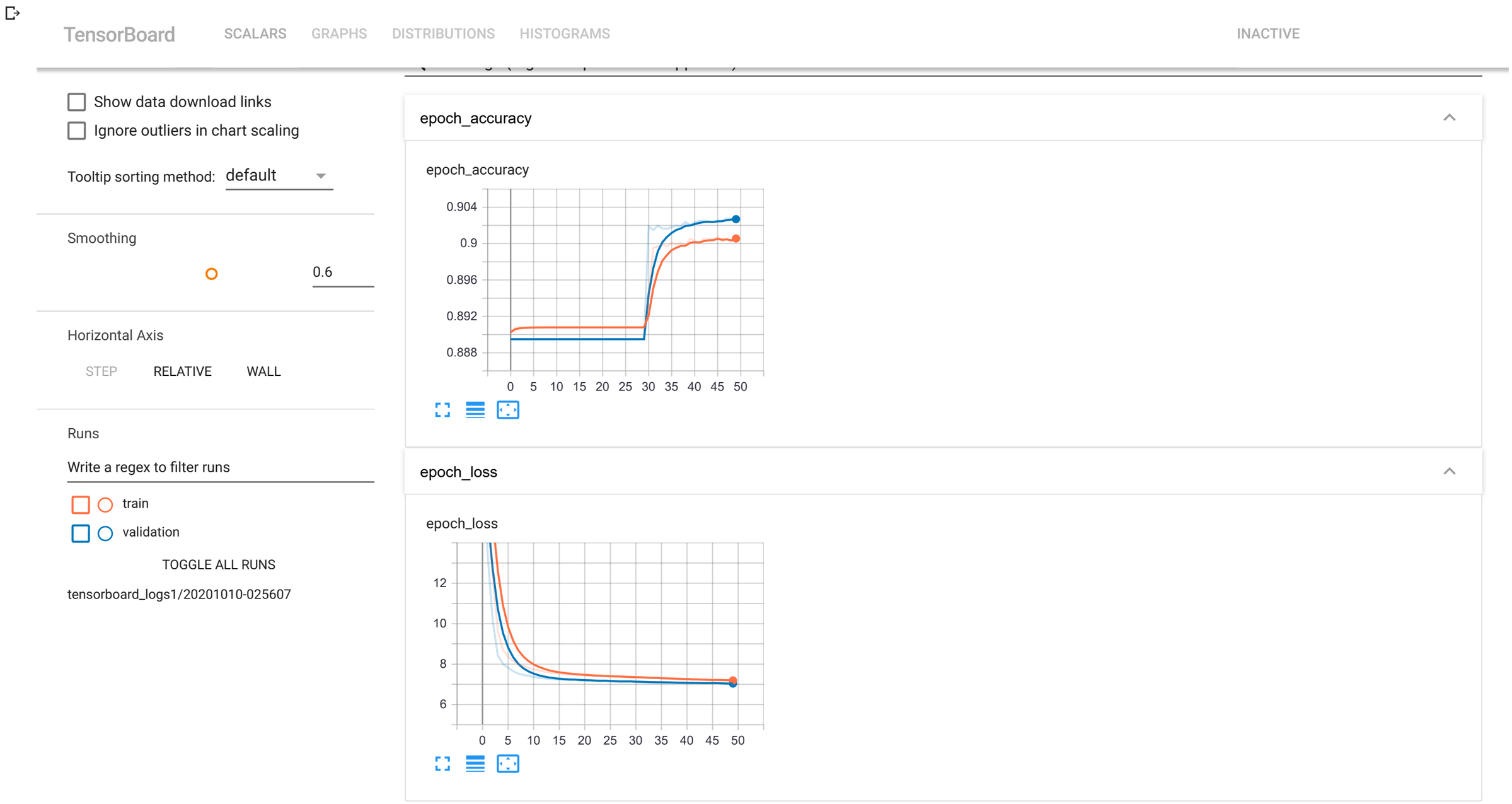
```
time_distributed (TimeDistribut (None, 34, 1)        5           dense_1[0][0]
```

```
%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=50,batch_size=128,validation_data=val_data,callbacks=callbacks)
```

```
WARNING:tensorflow:Model failed to serialize as JSON. Ignoring... Layer attention has arguments in `__init__` and therefore must override `get_config`.
Epoch 1/50
  1/103 [..............................] - ETA: 0s - loss: 23.5665 - accuracy: 0.8389WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/summary_ops_v2.py
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
  2/103 [..............................] - ETA: 18s - loss: 23.5518 - accuracy: 0.8622WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch ti
102/103 [============================>.] - ETA: 0s - loss: 21.4441 - accuracy: 0.8903
Epoch 00001: val_loss improved from inf to 18.45364, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 131ms/step - loss: 21.4115 - accuracy: 0.8903 - val_loss: 18.4536 - val_accuracy: 0.8895
Epoch 2/50
102/103 [============================>.] - ETA: 0s - loss: 16.0953 - accuracy: 0.8908
Epoch 00002: val_loss improved from 18.45364 to 13.44885, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 120ms/step - loss: 16.0703 - accuracy: 0.8908 - val_loss: 13.4488 - val_accuracy: 0.8895
Epoch 3/50
102/103 [============================>.] - ETA: 0s - loss: 12.3556 - accuracy: 0.8908
Epoch 00003: val_loss improved from 13.44885 to 10.13099, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 115ms/step - loss: 12.3362 - accuracy: 0.8908 - val_loss: 10.1310 - val_accuracy: 0.8895
Epoch 4/50
102/103 [============================>.] - ETA: 0s - loss: 9.6318 - accuracy: 0.8908
Epoch 00004: val_loss improved from 10.13099 to 8.42521, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 118ms/step - loss: 9.6173 - accuracy: 0.8908 - val_loss: 8.4252 - val_accuracy: 0.8895
Epoch 5/50
102/103 [============================>.] - ETA: 0s - loss: 8.7182 - accuracy: 0.8908
Epoch 00005: val_loss improved from 8.42521 to 8.01204, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 117ms/step - loss: 8.7049 - accuracy: 0.8908 - val_loss: 8.0120 - val_accuracy: 0.8895
Epoch 6/50
102/103 [============================>.] - ETA: 0s - loss: 8.3851 - accuracy: 0.8908
Epoch 00006: val_loss improved from 8.01204 to 7.82283, saving model to checkpt/model2.hdf5
103/103 [==============================] - 11s 103ms/step - loss: 8.3729 - accuracy: 0.8908 - val_loss: 7.8228 - val_accuracy: 0.8895
Epoch 7/50
102/103 [============================>.] - ETA: 0s - loss: 8.1768 - accuracy: 0.8908
Epoch 00007: val_loss improved from 7.82283 to 7.65312, saving model to checkpt/model2.hdf5
103/103 [==============================] - 10s 94ms/step - loss: 8.1646 - accuracy: 0.8908 - val_loss: 7.6531 - val_accuracy: 0.8895
Epoch 8/50
102/103 [============================>.] - ETA: 0s - loss: 8.0112 - accuracy: 0.8907
Epoch 00008: val_loss improved from 7.65312 to 7.52960, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 113ms/step - loss: 7.9989 - accuracy: 0.8908 - val_loss: 7.5296 - val_accuracy: 0.8895
Epoch 9/50
102/103 [============================>.] - ETA: 0s - loss: 7.9186 - accuracy: 0.8907
Epoch 00009: val_loss improved from 7.52960 to 7.46558, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 115ms/step - loss: 7.9065 - accuracy: 0.8908 - val_loss: 7.4656 - val_accuracy: 0.8895
Epoch 10/50
102/103 [============================>.] - ETA: 0s - loss: 7.8336 - accuracy: 0.8908
Epoch 00010: val_loss improved from 7.46558 to 7.41664, saving model to checkpt/model2.hdf5
103/103 [==============================] - 10s 99ms/step - loss: 7.8219 - accuracy: 0.8908 - val_loss: 7.4166 - val_accuracy: 0.8895
Epoch 11/50
102/103 [============================>.] - ETA: 0s - loss: 7.7533 - accuracy: 0.8909
Epoch 00011: val_loss improved from 7.41664 to 7.36623, saving model to checkpt/model2.hdf5
103/103 [==============================] - 10s 96ms/step - loss: 7.7423 - accuracy: 0.8908 - val_loss: 7.3662 - val_accuracy: 0.8895
Epoch 12/50
102/103 [============================>.] - ETA: 0s - loss: 7.7021 - accuracy: 0.8908
Epoch 00012: val_loss improved from 7.36623 to 7.32386, saving model to checkpt/model2.hdf5
103/103 [==============================] - 10s 94ms/step - loss: 7.6902 - accuracy: 0.8908 - val_loss: 7.3239 - val_accuracy: 0.8895
Epoch 13/50
102/103 [============================>.] - ETA: 0s - loss: 7.6458 - accuracy: 0.8907
Epoch 00013: val_loss improved from 7.32386 to 7.30067, saving model to checkpt/model2.hdf5
103/103 [==============================] - 11s 108ms/step - loss: 7.6341 - accuracy: 0.8908 - val_loss: 7.3007 - val_accuracy: 0.8895
Epoch 14/50
102/103 [============================>.] - ETA: 0s - loss: 7.5804 - accuracy: 0.8909
Epoch 00014: val_loss improved from 7.30067 to 7.27068, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 115ms/step - loss: 7.5701 - accuracy: 0.8908 - val_loss: 7.2707 - val_accuracy: 0.8895
Epoch 15/50
102/103 [============================>.] - ETA: 0s - loss: 7.5668 - accuracy: 0.8908
Epoch 00015: val_loss improved from 7.27068 to 7.25156, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 116ms/step - loss: 7.5553 - accuracy: 0.8908 - val_loss: 7.2516 - val_accuracy: 0.8895
Epoch 16/50
102/103 [============================>.] - ETA: 0s - loss: 7.5502 - accuracy: 0.8908
Epoch 00016: val_loss improved from 7.25156 to 7.23627, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 116ms/step - loss: 7.5390 - accuracy: 0.8908 - val_loss: 7.2363 - val_accuracy: 0.8895
Epoch 17/50
102/103 [============================>.] - ETA: 0s - loss: 7.5178 - accuracy: 0.8908
Epoch 00017: val_loss improved from 7.23627 to 7.22623, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 115ms/step - loss: 7.5068 - accuracy: 0.8908 - val_loss: 7.2262 - val_accuracy: 0.8895
Epoch 18/50
102/103 [============================>.] - ETA: 0s - loss: 7.5015 - accuracy: 0.8907
Epoch 00018: val_loss improved from 7.22623 to 7.21520, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 112ms/step - loss: 7.4898 - accuracy: 0.8908 - val_loss: 7.2152 - val_accuracy: 0.8895
Epoch 19/50
102/103 [============================>.] - ETA: 0s - loss: 7.4870 - accuracy: 0.8908
Epoch 00019: val_loss did not improve from 7.21520
103/103 [==============================] - 11s 112ms/step - loss: 7.4758 - accuracy: 0.8908 - val_loss: 7.2293 - val_accuracy: 0.8895
Epoch 20/50
102/103 [============================>.] - ETA: 0s - loss: 7.4668 - accuracy: 0.8908
Epoch 00020: val_loss improved from 7.21520 to 7.19120, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 115ms/step - loss: 7.4553 - accuracy: 0.8908 - val_loss: 7.1912 - val_accuracy: 0.8895
Epoch 21/50
102/103 [============================>.] - ETA: 0s - loss: 7.4587 - accuracy: 0.8907
Epoch 00021: val_loss improved from 7.19120 to 7.19000, saving model to checkpt/model2.hdf5
103/103 [==============================] - 11s 110ms/step - loss: 7.4468 - accuracy: 0.8908 - val_loss: 7.1900 - val_accuracy: 0.8895
Epoch 22/50
102/103 [============================>.] - ETA: 0s - loss: 7.4400 - accuracy: 0.8907
Epoch 00022: val_loss improved from 7.19000 to 7.18933, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 112ms/step - loss: 7.4284 - accuracy: 0.8908 - val_loss: 7.1893 - val_accuracy: 0.8895
Epoch 23/50
102/103 [============================>.] - ETA: 0s - loss: 7.4234 - accuracy: 0.8908
Epoch 00023: val_loss improved from 7.18933 to 7.17182, saving model to checkpt/model2.hdf5
103/103 [==============================] - 14s 137ms/step - loss: 7.4127 - accuracy: 0.8908 - val_loss: 7.1718 - val_accuracy: 0.8895
Epoch 24/50
102/103 [============================>.] - ETA: 0s - loss: 7.4248 - accuracy: 0.8909
Epoch 00024: val_loss improved from 7.17182 to 7.16440, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 123ms/step - loss: 7.4141 - accuracy: 0.8908 - val_loss: 7.1644 - val_accuracy: 0.8895
Epoch 25/50
102/103 [============================>.] - ETA: 0s - loss: 7.4063 - accuracy: 0.8907
Epoch 00025: val_loss did not improve from 7.16440
103/103 [==============================] - 11s 108ms/step - loss: 7.3947 - accuracy: 0.8908 - val_loss: 7.1827 - val_accuracy: 0.8895
Epoch 26/50
102/103 [============================>.] - ETA: 0s - loss: 7.3948 - accuracy: 0.8909
Epoch 00026: val_loss improved from 7.16440 to 7.14186, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 119ms/step - loss: 7.3841 - accuracy: 0.8908 - val_loss: 7.1419 - val_accuracy: 0.8895
Epoch 27/50
102/103 [============================>.] - ETA: 0s - loss: 7.3936 - accuracy: 0.8907
Epoch 00027: val_loss improved from 7.14186 to 7.13866, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 123ms/step - loss: 7.3821 - accuracy: 0.8908 - val_loss: 7.1387 - val_accuracy: 0.8895
Epoch 28/50
102/103 [============================>.] - ETA: 0s - loss: 7.3861 - accuracy: 0.8908
Epoch 00028: val_loss improved from 7.13866 to 7.13403, saving model to checkpt/model2.hdf5
103/103 [==============================] - 12s 119ms/step - loss: 7.3748 - accuracy: 0.8908 - val_loss: 7.1340 - val_accuracy: 0.8895
Epoch 29/50
102/103 [============================>.] - ETA: 0s - loss: 7.3689 - accuracy: 0.8907
Epoch 00029: val_loss did not improve from 7.13403
103/103 [==============================] - 13s 124ms/step - loss: 7.3575 - accuracy: 0.8908 - val_loss: 7.1442 - val_accuracy: 0.8895
Epoch 30/50
102/103 [============================>.] - ETA: 0s - loss: 7.3573 - accuracy: 0.8907
Epoch 00030: val_loss did not improve from 7.13403
103/103 [==============================] - 13s 124ms/step - loss: 7.3460 - accuracy: 0.8908 - val_loss: 7.1474 - val_accuracy: 0.8895
Epoch 31/50
102/103 [============================>.] - ETA: 0s - loss: 7.3558 - accuracy: 0.8941
```

```
Epoch 00031: val_loss improved from 7.13403 to 7.11304, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 122ms/step - loss: 7.3450 - accuracy: 0.8941 - val_loss: 7.1130 - val_accuracy: 0.9019
Epoch 32/50
102/103 [============================>.] - ETA: 0s - loss: 7.3510 - accuracy: 0.8996
Epoch 00032: val_loss improved from 7.11304 to 7.11153, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 124ms/step - loss: 7.3401 - accuracy: 0.8995 - val_loss: 7.1115 - val_accuracy: 0.9015
Epoch 33/50
102/103 [============================>.] - ETA: 0s - loss: 7.3295 - accuracy: 0.8997
Epoch 00033: val_loss improved from 7.11153 to 7.10574, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 126ms/step - loss: 7.3186 - accuracy: 0.8997 - val_loss: 7.1057 - val_accuracy: 0.9020
Epoch 34/50
102/103 [============================>.] - ETA: 0s - loss: 7.3121 - accuracy: 0.9000
Epoch 00034: val_loss improved from 7.10574 to 7.09552, saving model to checkpt/model2.hdf5
103/103 [==============================] - 13s 126ms/step - loss: 7.3014 - accuracy: 0.8999 - val_loss: 7.0955 - val_accuracy: 0.9017
Epoch 35/50
102/103 [============================>.] - ETA: 0s - loss: 7.3233 - accuracy: 0.8997
Epoch 00035: val_loss did not improve from 7.09552
```

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




```
Epoch 00048: val_loss improved from 7.03490 to 7.03030, saving model to checkpt/model2.hdf5
```

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

```
102/103 [============================>.] - ETA: 0s - loss: 7.2032 - accuracy: 0.9002
```

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

Inference

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

```
w = model.get_weights()
len(w)
```

```
16
```

```
names = [weight.name for layer in model.layers for weight in layer.weights]
len(names)
```

```
16
```

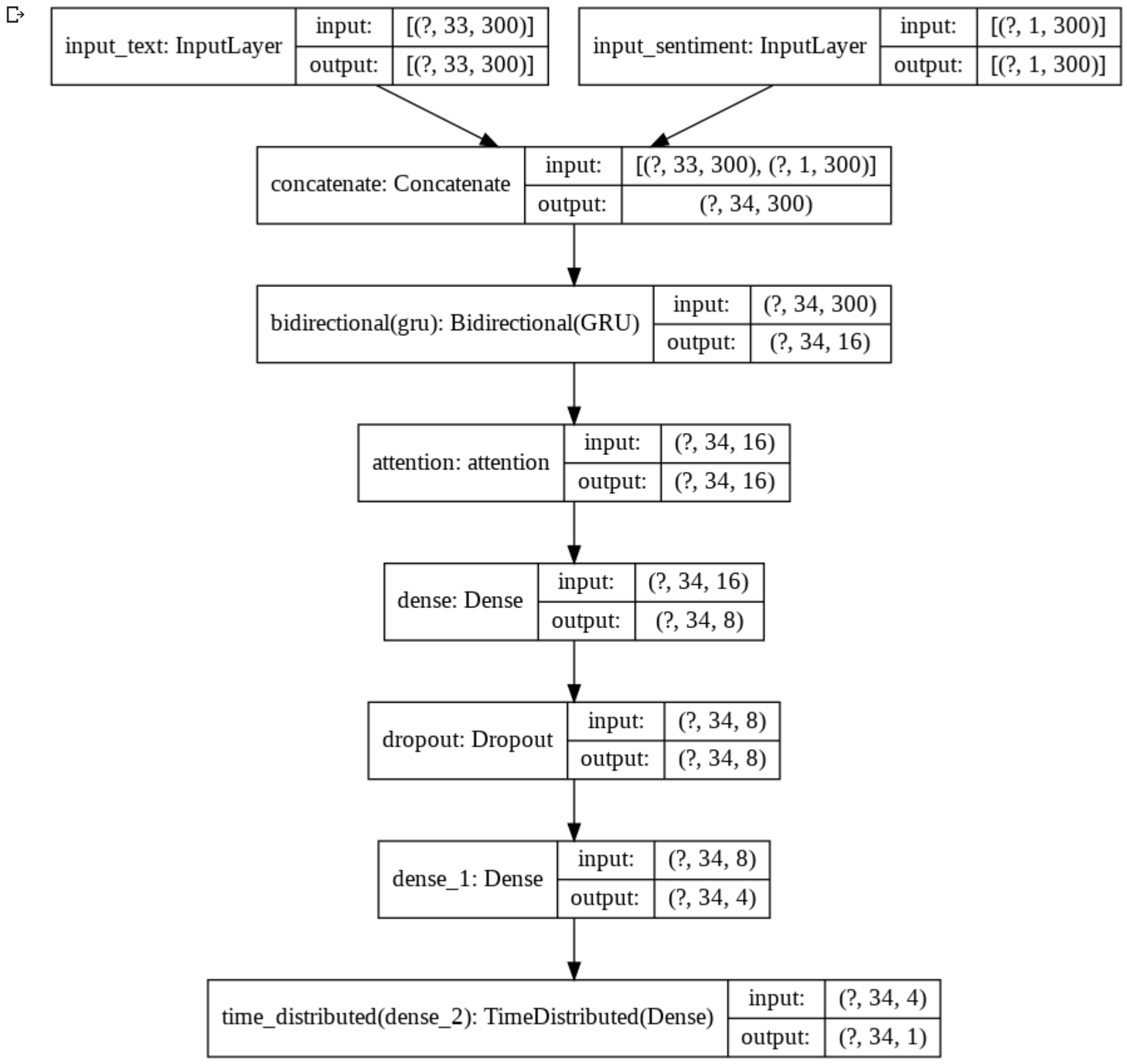
```
for layer,weight in zip(names,w):
  print(layer,weight.shape)
```

```
embedding1/embeddings:0 (17397, 300)
embedding2/embeddings:0 (4, 300)
bidirectional/forward_gru/gru_cell_1/kernel:0 (300, 24)
bidirectional/forward_gru/gru_cell_1/recurrent_kernel:0 (8, 24)
bidirectional/forward_gru/gru_cell_1/bias:0 (2, 24)
bidirectional/backward_gru/gru_cell_2/kernel:0 (300, 24)
bidirectional/backward_gru/gru_cell_2/recurrent_kernel:0 (8, 24)
bidirectional/backward_gru/gru_cell_2/bias:0 (2, 24)
attention/att_weight:0 (16, 1)
attention/att_bias:0 (34, 1)
dense/kernel:0 (16, 8)
dense/bias:0 (8,)
dense_1/kernel:0 (8, 4)
dense_1/bias:0 (4,)
time_distributed/kernel:0 (4, 1)
time_distributed/bias:0 (1,)
```

```
tf.keras.backend.clear_session()
```

```
from tensorflow.keras.models import Model
```

```
input1=Input(shape=(max_length_text,300),name='input_text')
input2=Input(shape=(max_length_sentiment,300),name='input_sentiment')

concat= Concatenate(axis=1)([input1,input2])
gru=Bidirectional(GRU(8,name='gru',return_sequences=True,dropout=0.4))(concat)
att = attention(return_sequences=True)(gru)
dense1 = Dense(8,activation='relu',kernel_regularizer=l2(0.0001))(att)
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)
import tensorflow as tf
tf.keras.utils.plot_model(model, 'Model1.png',show_shapes=True)
```




```
model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_text (InputLayer)         [(None, 33, 300)]    0
__________________________________________________________________________________________________
input_sentiment (InputLayer)    [(None, 1, 300)]     0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 34, 300)      0           input_text[0][0]
                                                                 input_sentiment[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 34, 16)       14880       concatenate[0][0]
__________________________________________________________________________________________________
attention (attention)           (None, 34, 16)       50          bidirectional[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 34, 8)        136         attention[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 34, 8)        0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 34, 4)        36          dropout[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 34, 1)        5           dense_1[0][0]
==================================================================================================
Total params: 15,107
Trainable params: 15,107
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
model.set_weights(w[2:])


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

	text	selected_text	sentiment
639	sorry to tweet about bgt but poor wonderful cr...	sorry	negative
9466	walah me still i am not getting the full idea	not getting the full idea	negative

```
from nltk.tokenize import WhitespaceTokenizer
def get_text_vectors(sent):
  vector = np.zeros(shape=(33,300))
  tk = WhitespaceTokenizer()
  sent = tk.tokenize(sent)
  for i in range(len(sent)):
    try:
      vec = fasttext_model.wv.word_vec(sent[i])
      vector[i] =  vec
    except:
      continue
  return vector
```

```
all_sent = (x_val['text'].values)

ip_text = np.zeros(shape=(len(all_sent),33,300))
for i in tqdm(range(len(all_sent))):
  res = get_text_vectors(all_sent[i])
  res = res.reshape((1,res.shape[0],res.shape[1]))
  ip_text[i] = res
```

```
100%|██████████| 3271/3271 [00:00<00:00, 10508.97it/s]
```

```
ip_text.shape
```

```
(3271, 33, 300)
```

```
def get_sentiment_vector(sentiment):
  return fasttext_model.wv.word_vec(sentiment)

all_sentiment = x_val['sentiment'].values

ip_sentiment = np.zeros(shape = (len(all_sentiment),1,300))
for i in tqdm(range(len(all_sentiment))):
  vec = get_sentiment_vector(all_sentiment[i])
  vec = vec.reshape((1,1,300))
  ip_sentiment[i] = vec

ip_sentiment.shape
```

```
100%|██████████| 3271/3271 [00:00<00:00, 177512.27it/s]
(3271, 1, 300)
```

```
val_data = (ip_text,ip_sentiment)


val_pred = model.predict(val_data)
val_pred = np.squeeze(val_pred)
val_pred = np.where(val_pred>0.2,1,0)
val_pred.shape
```

```
(3271, 34)
```

```
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%|██████████| 3271/3271 [00:00<00:00, 62916.19it/s]3271
```

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

```
x_val
```

	text	selected_text	sentiment	prediction
639	sorry to tweet about bgt but poor wonderful cr...	sorry	negative	[0, 1, 6, 7, 8, 9, 11, 12, 13, 14, 15]
9466	walah me still i am not getting the full idea	not getting the full idea	negative	[5, 6]
7105	yo wake your curse up and go to work go get th...	sick dont	negative	[0, 1, 2, 3, 13, 14, 15]
19196	thanks for sharing	thanks	positive	[0, 1, 2]
4717	it is an app to finally face the truth you lac...	you lack time	negative	[6, 7, 8, 9, 10, 11, 12, 13, 14, 17]
...	...	...	...	...
15914	oh noesss seniors last day however tickling wi...	totally worth it	positive	[0, 1, 2, 14, 15, 16, 17, 18]
23340	good luck tomorrow	good luck tomorrow	positive	[0, 1, 2]
12499	played with fontstruct links uploaded to dafon...	top	positive	[13]
21280	think i am gonna start writing a proper blog c...	i am gonna start writing a proper blog	positive	[11, 12, 13, 14]
18594	not to sound preachery or anything but my ipho...	amazing	positive	[0, 1, 2, 9, 10, 11]

3271 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_val[['text','prediction']].progress_apply(lambda x:get_pred_text(x),axis=1)
```

```
100%|| 3271/3271 [00:00<00:00, 26956.14it/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
8187	this is dumb i keep losing followers	this is dumb i keep losing followers	negative	[0, 1, 2, 3, 5]	this is dumb i losing
843	it just had to rain on me almost a perfect day...	it just had to rain on me	negative	[5, 6, 7, 8, 9, 10]	on me almost a perfect day
14300	c news was not as bad as i expected could have...	better	positive	[2, 3, 4, 5, 18]	was not as bad great
21638	omg that is awful wow our pyr figured out how ...	omg that is awful	negative	[0, 1, 2, 3, 4, 5, 17]	omg that is awful wow our crushing
8667	peep this remix from nothe wu dynasty remix ta...	delayed	negative	[10]	delayed

```
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%|| 3271/3271 [00:00<00:00, 31845.32it/s]
```

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

	text	selected_text	sentiment	prediction	pred_text	jaccard
12935	adam samberg new moon trailor good evening too...	too bad my cable is off as of friday	negative	[5, 6, 7, 8]	good evening too bad	0.181818
21439	so jealous see if you can get some dallas conc...	jealous	negative	[0, 1, 2]	so jealous see	0.333333
11408	i might have been a child but i was never one ...	apologised	negative	[1, 2, 3, 7, 8, 9, 11, 17]	might have been i was never of apologised	0.125000
6459	do not count on it	do not count on it	negative	[1]	not	0.200000
6442	happy mother is day to all the mothers	happy mother is day	positive	[0, 1, 2, 3]	happy mother is day	1.000000

```
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
```

```
((1661, 6), (1610, 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: 0.40192826596900244
Mean jaccard score for negative sentiment data 0.37148625313293876
```