```
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, 5), (3534, 3))
```

```
train_df.head()
```

	textID	text	selected_text	sentiment	misspelled
0	cb774db0d1	i`d have responded if i were going	i`d have responded if i were going	neutral	No
1	549e992a42	sooo sad i will miss you here in san diego	sooo sad	negative	No
2	088c60f138	my boss is bullying me	bullying me	negative	No
3	9642c003ef	what interview leave me alone	leave me alone	negative	No
4	358bd9e861	sons of curse why couldn`t they put them on th...	sons of curse	negative	No

```
test_df.head()
```

	textID	text	sentiment
0	f87dea47db	last session of the day links	neutral
1	96d74cb729	shanghai is also really exciting precisely sky...	positive
2	eee518ae67	recession hit veronique branquinho she has to ...	negative
3	01082688c6	happy bday	positive
4	33987a8ee5	links i like it	positive

```
def find_start_indices(x):
    text,sel_text  = x[0],x[1]
    text = text.split()
    sel_text = sel_text.split()
    end = sel_text[0]
    index = text.index(end)
    return index
```

```
train_df['start_indices'] = train_df[['text','selected_text']].progress_apply(lambda x:find_start_indices(x),axis=1)
```

```
100%  27469/27469 [00:02<00:00, 10496.39it/s]
```

```
def find_end_indices(x):
    text,sel_text,start_indices  = x[0],x[1],x[2]
    text = text.split()
    sel_text = sel_text.split()
    end = sel_text[-1]
    try:
        index = text.index(end,start_indices)
    except:
        index = text.index(end)

    return index
```

```
train_df['end_indices'] = train_df[['text','selected_text','start_indices']].progress_apply(lambda x:find_end_indices(x),axis=1)
```

```
100%  27469/27469 [00:01<00:00, 18221.61it/s]
```

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

```
(23, 7)
```

We have 23 rows where the start indices are greater than end incides. we can drop these rows

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

```
train_df.shape
```

```
(27446, 7)
```

```
train_df.sample(15)
```

	textID	text	selected_text	sentiment	misspelled	start_indices	end_indices
3364	0f4354b370	entering twitter `lurk` mode time to lock the ...	entering twitter `lurk` mode time to lock the ...	neutral	No	0	13
11804	01818a5856	thanking god for after elton for allowing me t...	thanking god	positive	No	0	1
3400	6dfe561162	get down tonight links the bridesmaids moms br...	moms bride i rockin` the reception	positive	No	6	11
14177	56382a16f4	i have just been to see the jonas brothers mov...	i have just been to see the jonas brothers mov...	positive	No	0	19
4305	8a7feebc4b	boring what ugh come back to of then	boring	negative	No	0	0
2619	fa67ea8b66	morning bank holiday monday and the sun has go...	morning bank holiday monday and the sun has go...	neutral	No	0	20
11618	47dae4cfaf	i wish she knew what she puts me through she s...	i wish she knew what she puts me through she s...	positive	No	0	27
14899	3333672c6e	beanz curse sorry to hear that tina	sorry	negative	No	2	2
14320	dbade951f3	more days with slow internet	slow internet	negative	No	3	4
2149	55276d9a72	well another family was chosen for the child w...	well another family was chosen for the child w...	neutral	No	0	16
3245	67d77a93b5	oh curse happy late birthday my txt didn`t send	oh curse happy late birthday my txt didn`t send	neutral	No	0	8
21035	5bea8d1365	i will have spent my allowance at cybernet exp...	flo bttr	positive	No	22	23
22352	f5bbbdc257	try installing twibble which is a java based a...	try installing twibble which is a java based a...	neutral	No	0	14
392	5fcf7bd80c	my sunburn is peeling	my sunburn is peeling	negative	No	0	3

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

```
X.shape,y.shape
```

```
((27446, 2), (27446, 2))
```

```
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
```

```
((21956, 2), (5490, 2), (21956, 2), (5490, 2))
```

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

```
#import os
#os.listdir('/content/')
```

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

```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_text = Tokenizer(lower=True,split=' ',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)
```

```
21956 5490
20901
```

```
max_length_text=40
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)
```

```
(21956, 40) (5490, 40)
```

```
#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)
```

```
(20901, 300)
```

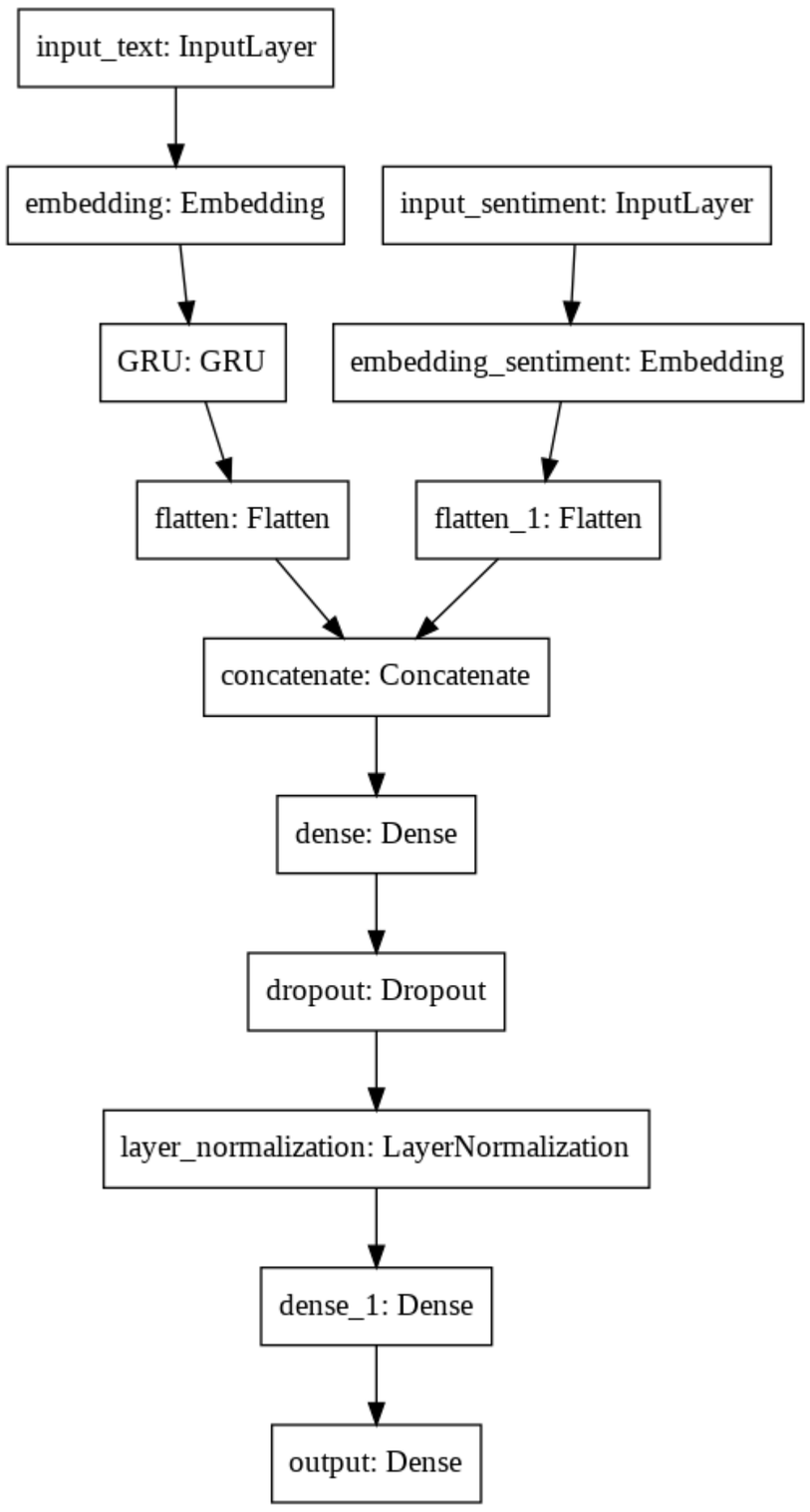
```
train_sentiment = x_train['sentiment'].values
val_sentiment = x_val['sentiment'].values


from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_sentiment = Tokenizer(lower=True,split=' ',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)
```

```
21956 5490
{'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)
```

```
(21956, 1) (5490, 1)
```

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


input1=Input(shape=(max_length_text,),name='input_text')
embed = Embedding(vocab_size_text,300,input_length=max_length_text,name='embedding',\
                      trainable=False,embeddings_initializer=tf.constant_initializer(embedding_matrix))(input1)
gru=GRU(32,name='GRU',return_sequences=True)(embed)
f1=Flatten()(gru)


input2=Input(shape=(max_length_sentiment,),name='input_sentiment')
embed2=Embedding(vocab_size_sentiment,10,input_length=max_length_sentiment,name='embedding_sentiment')(input2)
f2=Flatten()(embed2)


concat1=Concatenate(axis=1)([f1,f2])
dense1=Dense(16,activation='relu',kernel_regularizer=l2(0.0001))(concat1)
drop1 = Dropout(0.6)(dense1)
ln= LayerNormalization()(drop1)
dense2=Dense(8,activation='relu',kernel_regularizer=l2(0.0001))(ln)
output=Dense(2,name='output')(dense2)


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


import tensorflow
tensorflow.keras.utils.plot_model(model)
```




```
model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_text (InputLayer)         [(None, 40)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 40, 300)      6270300     input_text[0][0]
__________________________________________________________________________________________________
input_sentiment (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________________________
GRU (GRU)                       (None, 40, 32)       32064       embedding[0][0]
__________________________________________________________________________________________________
embedding_sentiment (Embedding) (None, 1, 10)        50          input_sentiment[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 1280)         0           GRU[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 10)           0           embedding_sentiment[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 1290)         0           flatten[0][0]
                                                                 flatten_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 16)           20656       concatenate[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 16)           0           dense[0][0]
__________________________________________________________________________________________________
layer_normalization (LayerNorma (None, 16)           32          dropout[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 8)            136         layer_normalization[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 2)            18          dense_1[0][0]
==================================================================================================
Total params: 6,323,256
Trainable params: 52,956
Non-trainable params: 6,270,300
```

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

val = (val_text,val_sentiment)
output_val = y_val.values
val_data = (val,output_val)


%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)
callbacks=[tensorboard_callback]


model.compile(optimizer='adam',loss="mse",metrics=["mae"])
```

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Mean Squared Error can be defined as the average squared difference between the actual and predicted values

```
#how MSE works for Multi-output model
true_values = np.array([[1,2,3],[3,4.8,5],[6,6.0,8],[6.0,9,20]])
pred_values = np.array([[5,0,7],[5.0,0,5],[0,0,7],[5,0,8]])

mse = tensorflow.keras.losses.MeanSquaredError()
print(mse(true_values,pred_values))
```

```
tf.Tensor(30.170001983642578, shape=(), dtype=float64)
```

```
print(np.square(true_values - pred_values))
f = ((np.square(true_values - pred_values)).flatten())
print(f)
print(np.sum(f))
print(np.sum(f) /len(f))
print(np.mean(np.square(true_values - pred_values)))
```

```
[[ 16.     4.    16.  ]
 [  4.    23.04   0.  ]
 [ 36.    36.     1.  ]
 [  1.    81.   144.  ]]
[ 16.     4.    16.     4.    23.04   0.    36.    36.     1.     1.
  81.   144.  ]
362.03999999999996
30.169999999999998
30.169999999999998
```

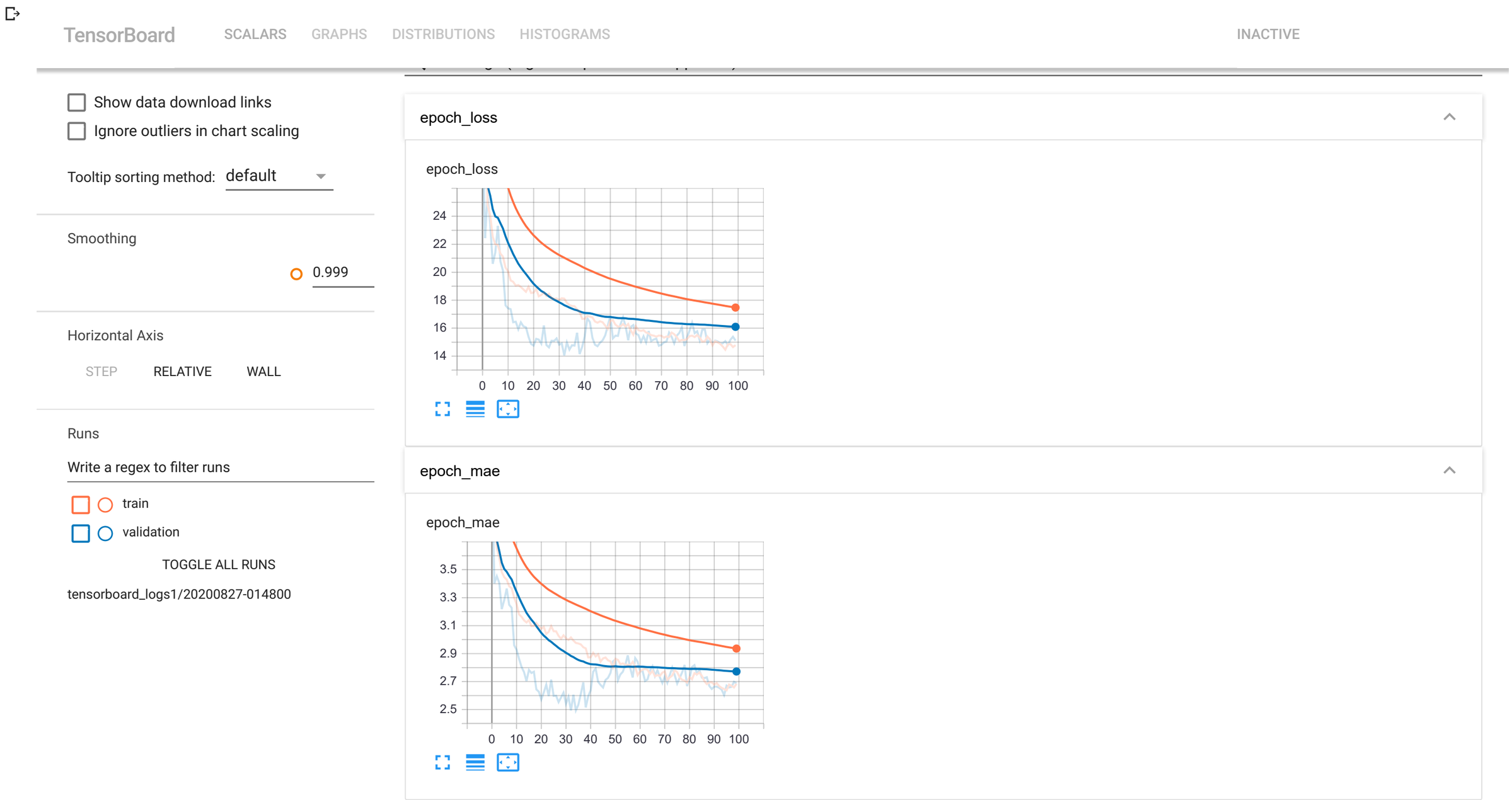
So MSE works as follows fr multi dim. data

1. Calculate diff bt. true and pred values (true-pred)
2. Square those values
3. Calculate their mean values

```
model.fit(input_data,output_data,epochs=100,batch_size=128,validation_data=val_data,validation_batch_size=64,callbacks=callbacks)
```

```
172/172 [==============================] - 2s 12ms/step - loss: 15.4590 - mae: 2.7723 - val_loss: 14.8113 - val_mae: 2.7379
Epoch 72/100
172/172 [==============================] - 2s 12ms/step - loss: 15.3246 - mae: 2.7123 - val_loss: 14.9917 - val_mae: 2.6910
Epoch 73/100
172/172 [==============================] - 2s 12ms/step - loss: 15.4067 - mae: 2.7296 - val_loss: 15.0482 - val_mae: 2.7469
Epoch 74/100
172/172 [==============================] - 2s 12ms/step - loss: 15.3665 - mae: 2.7435 - val_loss: 15.6847 - val_mae: 2.8002
Epoch 75/100
172/172 [==============================] - 2s 12ms/step - loss: 15.6848 - mae: 2.7815 - val_loss: 15.3758 - val_mae: 2.7460
Epoch 76/100
172/172 [==============================] - 2s 12ms/step - loss: 15.5691 - mae: 2.7675 - val_loss: 14.9145 - val_mae: 2.6863
Epoch 77/100
172/172 [==============================] - 2s 12ms/step - loss: 15.3603 - mae: 2.7322 - val_loss: 15.4585 - val_mae: 2.7261
Epoch 78/100
172/172 [==============================] - 2s 12ms/step - loss: 15.0835 - mae: 2.7126 - val_loss: 15.6637 - val_mae: 2.7656
Epoch 79/100
172/172 [==============================] - 2s 12ms/step - loss: 15.1579 - mae: 2.7072 - val_loss: 16.0651 - val_mae: 2.8191
Epoch 80/100
172/172 [==============================] - 2s 12ms/step - loss: 15.1409 - mae: 2.6925 - val_loss: 14.7226 - val_mae: 2.6677
Epoch 81/100
172/172 [==============================] - 2s 12ms/step - loss: 15.2170 - mae: 2.7321 - val_loss: 15.3244 - val_mae: 2.7241
Epoch 82/100
172/172 [==============================] - 2s 12ms/step - loss: 15.4321 - mae: 2.7163 - val_loss: 15.8263 - val_mae: 2.8047
Epoch 83/100
172/172 [==============================] - 2s 12ms/step - loss: 15.4616 - mae: 2.7606 - val_loss: 16.3908 - val_mae: 2.8176
Epoch 84/100
172/172 [==============================] - 2s 12ms/step - loss: 15.3662 - mae: 2.7512 - val_loss: 15.6441 - val_mae: 2.7553
Epoch 85/100
172/172 [==============================] - 2s 12ms/step - loss: 15.5090 - mae: 2.7650 - val_loss: 15.7794 - val_mae: 2.7827
Epoch 86/100
172/172 [==============================] - 2s 12ms/step - loss: 15.3290 - mae: 2.7408 - val_loss: 15.1325 - val_mae: 2.7286
Epoch 87/100
172/172 [==============================] - 2s 12ms/step - loss: 15.0429 - mae: 2.7032 - val_loss: 15.9004 - val_mae: 2.7248
Epoch 88/100
172/172 [==============================] - 2s 12ms/step - loss: 15.0687 - mae: 2.6821 - val_loss: 15.8993 - val_mae: 2.7309
Epoch 89/100
172/172 [==============================] - 2s 12ms/step - loss: 15.4016 - mae: 2.6942 - val_loss: 14.9278 - val_mae: 2.6691
Epoch 90/100
172/172 [==============================] - 2s 12ms/step - loss: 15.1860 - mae: 2.6954 - val_loss: 15.2008 - val_mae: 2.6473
Epoch 91/100
172/172 [==============================] - 2s 12ms/step - loss: 14.7745 - mae: 2.6787 - val_loss: 14.8905 - val_mae: 2.6638
Epoch 92/100
172/172 [==============================] - 2s 12ms/step - loss: 15.0676 - mae: 2.6899 - val_loss: 15.1120 - val_mae: 2.6638
Epoch 93/100
172/172 [==============================] - 2s 12ms/step - loss: 14.9853 - mae: 2.6679 - val_loss: 15.0567 - val_mae: 2.6561
Epoch 94/100
172/172 [==============================] - 2s 12ms/step - loss: 14.8819 - mae: 2.6550 - val_loss: 14.9146 - val_mae: 2.6423
Epoch 95/100
172/172 [==============================] - 2s 12ms/step - loss: 14.6877 - mae: 2.6468 - val_loss: 14.9181 - val_mae: 2.6031
Epoch 96/100
172/172 [==============================] - 2s 12ms/step - loss: 14.4594 - mae: 2.6374 - val_loss: 15.0722 - val_mae: 2.6712
Epoch 97/100
172/172 [==============================] - 2s 12ms/step - loss: 14.8516 - mae: 2.6794 - val_loss: 14.8016 - val_mae: 2.6581
Epoch 98/100
172/172 [==============================] - 2s 12ms/step - loss: 14.8619 - mae: 2.6697 - val_loss: 15.1715 - val_mae: 2.6627
Epoch 99/100
172/172 [==============================] - 2s 12ms/step - loss: 14.6475 - mae: 2.6581 - val_loss: 15.3865 - val_mae: 2.6984
Epoch 100/100
172/172 [==============================] - 2s 12ms/step - loss: 14.7851 - mae: 2.6842 - val_loss: 15.1120 - val_mae: 2.6871
<tensorflow.python.keras.callbacks.History at 0x7fce10094438>
```

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



Compare MSE of this model with a random model

```
pred_values = model.predict(input_data)
true_values = output_data
```

```
mse = tensorflow.keras.losses.MeanSquaredError()
print(mse(true_values,pred_values))
```

```
tf.Tensor(12.821518, shape=(), dtype=float32)
```

+ Code + Text

```
pred = np.mean(true_values,axis=0)
print(pred)
```

```
[2.7855256 8.6957096]
```

```
pred = np.tile(pred,(true_values.shape[0],1))
print(pred)
print(pred.shape)
```

```
[[2.7855256 8.6957096]
 [2.7855256 8.6957096]
 [2.7855256 8.6957096]
 ...
 [2.7855256 8.6957096]
 [2.7855256 8.6957096]
 [2.7855256 8.6957096]]
(21956, 2)
```

```
#MSE value for random model (mean value as start and end indices)
print(mse(true_values,pred))
```

```
tf.Tensor(34.11643981933594, shape=(), dtype=float64)
```

```
print(train_df['start_indices'].value_counts()[:1])
print(train_df['end_indices'].value_counts()[:1])
```

```
0    16188
Name: start_indices, dtype: int64
3    2171
Name: end_indices, dtype: int64
```

```
pred = [0,3]
pred = np.tile(pred,(true_values.shape[0],1))
print(pred)
print(pred.shape)
```

```
[[0 3]
 [0 3]
 [0 3]
 ...
 [0 3]
 [0 3]
 [0 3]]
(21956, 2)
```

```
#MSE value for random model (most occurred value as start and end indicesss)
print(mse(true_values,pred))
```

```
tf.Tensor(53, shape=(), dtype=int64)
```

```
test_df
```

	textID	text	sentiment
0	f87dea47db	last session of the day links	neutral
1	96d74cb729	shanghai is also really exciting precisely sky...	positive
2	eee518ae67	recession hit veronique branquinho she has to ...	negative
3	01082688c6	happy bday	positive
4	33987a8ee5	links i like it	positive
...	...	...	...
3529	e5f0e6ef4b	its at am im very tired but i can`t sleep but ...	negative
3530	416863ce47	all alone in this old house again thanks for t...	positive
3531	6332da480c	i know what you mean my little dog is sinking ...	negative
3532	df1baec676	sutra what is your next youtube video gonna be...	positive
3533	469e15c5a8	links omgssh ang cute ng bby	positive

3534 rows × 3 columns

```
test_text = test_df['text'].values
test_text=tokenizer_text.texts_to_sequences(test_text)
test_text = pad_sequences(test_text,maxlen=max_length_text,padding='post')
test_text.shape
```

```
(3534, 40)
```

```
test_sentiment = test_df['sentiment'].values
test_sentiment=tokenizer_sentiment.texts_to_sequences(test_sentiment)
test_sentiment = pad_sequences(test_sentiment,maxlen=max_length_sentiment,padding='post')
test_sentiment.shape
```

```
(3534, 1)
```

```
results  = model.predict([test_text,test_sentiment])
```

```
results.shape
```

```
(3534, 2)
```

```
results = np.round(results)
results
```

```
array([[ 2.,  7.],
       [ 2.,  5.],
       [12., 16.],
       ...,
       [ 7., 13.],
       [ 7., 13.],
       [ 1.,  5.]], dtype=float32)
```

```
test_df['start'],test_df['end'] = (abs(results[:,0])),(abs(results[:,1]))
```

```
test_df['start'] = test_df['start'].astype('int')
test_df['end'] = test_df['end'].astype('int')
```

```
test_df
```