Amazon Fine Food Review Sentiment Analysis.

In this notebook, You will do amazon review classification with BERT.[Download data from this (https://www.kaggle.com/snap/amazon-fine-food-reviews/data) link]

It contains 5 parts as below. Detailed instrctions are given in the each cell. please read every comment we have written.

1. Preprocessing
2. Creating a BERT model from the Tensorflow HUB.
3. Tokenization
4. Getting the pretrained embedding Vector for a given review from the BERT.
5. Using the embedding data apply NN and classify the reviews.

In [1]:

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

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

In [ ]:

```
!pip install pyunpack
!pip install patool
```

```
Collecting pyunpack
  Downloading https://files.pythonhosted.org/packages/83/29/020436b1d8e96e5f
26fa282b9c3c13a3b456a36b9ea2edc87c5fed008369/pyunpack-0.2.2-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/83/29/020436b1d8e96e5f26fa282
b9c3c13a3b456a36b9ea2edc87c5fed008369/pyunpack-0.2.2-py2.py3-none-any.whl)
Collecting entrypoint2
  Downloading https://files.pythonhosted.org/packages/8a/b0/8ef4b1d8be02448d
164c52466530059d7f57218655d21309a0c4236d7454/entrypoint2-0.2.4-py3-none-any.
whl (https://files.pythonhosted.org/packages/8a/b0/8ef4b1d8be02448d164c52466
530059d7f57218655d21309a0c4236d7454/entrypoint2-0.2.4-py3-none-any.whl)
Collecting easyprocess
  Downloading https://files.pythonhosted.org/packages/48/3c/75573613641c90c6
d094059ac28adb748560d99bd27ee6f80cce398f404e/EasyProcess-0.3-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/48/3c/75573613641c90c6d094059
ac28adb748560d99bd27ee6f80cce398f404e/EasyProcess-0.3-py2.py3-none-any.whl)
Installing collected packages: entrypoint2, easyprocess, pyunpack
Successfully installed easyprocess-0.3 entrypoint2-0.2.4 pyunpack-0.2.2
Collecting patool
  Downloading https://files.pythonhosted.org/packages/43/94/52243ddff508780d
d2d8110964320ab4851134a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.wh
l (https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d81109643
20ab4851134a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl) (77kB)
     |████████████████████████████████| 81kB 2.4MB/s
Installing collected packages: patool
Successfully installed patool-1.12
```

In [ ]:

```
from pyunpack import Archive
Archive('/content/drive/MyDrive/Sem_seg/Reviews.rar').extractall('/content')
```

In [2]:

```
#All imports
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras.models import Model
from tqdm import tqdm_notebook
from sklearn.model_selection import train_test_split
from bs4 import BeautifulSoup
import re
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
import pickle
import os
import glob
from tensorflow.keras.layers import Input, Dense, Activation, Dropout
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras import Input
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import LearningRateScheduler, ReduceLROnPlateau, EarlyStopp
import shutil
import datetime
from wordcloud import WordCloud, STOPWORDS
```

In [ ]:

```
tf.test.gpu_device_name()
```

Out[5]:

```
'/device:GPU:0'
```

Grader function 1

In [ ]:

```
def grader_tf_version():
    assert((tf.__version__)>'2')
    return True
grader_tf_version()
```

Out[6]:

```
True
```

Part-1: Preprocessing

In [ ]:

```
#Read the dataset - Amazon fine food reviews
reviews = pd.read_csv(r"Reviews.csv")
#check the info of the dataset
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

In [ ]:

```
# Get only 2 columns - Text, Score
reviews = reviews[['Text', 'Score']]
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Text    568454 non-null  object
 1   Score   568454 non-null  int64
dtypes: int64(1), object(1)
memory usage: 8.7+ MB
```

In [ ]:

```
# Checking for NAN values.
reviews.isnull().sum(axis = 0)
```

Out[4]:

```
Text     0
Score    0
dtype: int64
```

In [ ]:

```
# Converting as Binary Classification Problem.
#if score> 3, set score = 1
#if score<=2, set score = 0
#if score == 3, remove the rows.
reviews = reviews[reviews["Score"]!=3]
upd = lambda row: 1 if row > 3 else 0
reviews
```

Out[5]:

	Text	Score
0	I have bought several of the Vitality canned d...	5
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	4
3	If you are looking for the secret ingredient i...	2
4	Great taffy at a great price. There was a wid...	5
...	...	...
568449	Great for sesame chicken..this is a good if no...	5
568450	I'm disappointed with the flavor. The chocolat...	2
568451	These stars are small, so you can give 10-15 o...	5
568452	These are the BEST treats for training and rew...	5
568453	I am very satisfied ,product is as advertised,...	5

525814 rows × 2 columns

In [ ]:

```
reviews['Score'] = reviews['Score'].map(upd)
reviews
```

Out[6]:

	Text	Score
0	I have bought several of the Vitality canned d...	1
1	Product arrived labeled as Jumbo Salted Peanut...	0
2	This is a confection that has been around a fe...	1
3	If you are looking for the secret ingredient i...	0
4	Great taffy at a great price. There was a wid...	1
...	...	...
568449	Great for sesame chicken..this is a good if no...	1
568450	I'm disappointed with the flavor. The chocolat...	0
568451	These stars are small, so you can give 10-15 o...	1
568452	These are the BEST treats for training and rew...	1
568453	I am very satisfied ,product is as advertised,...	1

525814 rows × 2 columns

Grader function 2

In [ ]:

```
def grader_reviews():
    temp_shape = (reviews.shape == (525814, 2)) and (reviews.Score.value_counts()[1]==44377
    assert(temp_shape == True)
    return True
grader_reviews()
```

Out[12]:

```
True
```

In [ ]:

```
# Limiting the data by using the reviews of length less than 50 and sampling 1,00,000 recor
def get_wordlen(x):
    return len(x.split())
reviews['len'] = reviews.Text.apply(get_wordlen)
reviews = reviews[reviews.len<50]
reviews = reviews.sample(n=100000, random_state=30)
reviews.shape
```

Out[7]:

```
(100000, 3)
```

In [ ]:

```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100000 entries, 64117 to 19261
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   Text    100000 non-null  object
 1   Score   100000 non-null  int64 
 2   len     100000 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 3.1+ MB
```

In [ ]:

```
reviews.reset_index(drop = True, inplace = True)
reviews
```

Out[9]:

	Text	Score	len
0	The tea was of great quality and it tasted lik...	1	30
1	My cat loves this. The pellets are nice and s...	1	31
2	Great product. Does not completely get rid of ...	1	41
3	This gum is my favorite! I would advise every...	1	27
4	I also found out about this product because of...	1	22
...	...	...	...
99995	Using this coffee and a stove top espresso mak...	1	39
99996	THE TASTE OF THIS M&M IS THE BEST. I USED IT I...	1	28
99997	Excellent Tea. I enjoy a cup every now and the...	1	21
99998	These oatmeal cookies have a great spice taste...	1	23
99999	This is the best coffee ever! I will never dri...	1	28

100000 rows × 3 columns

In [ ]:

```
reviews[reviews['Score']==0]
```

Out[10]:

	Text	Score	len
8	Never received product. Contacted seller askin...	0	42
10	For a shipping charge of over $8, you'd think ...	0	44
15	Despite the picture, and that it shows up when...	0	24
20	I thought I would love this...I love coconut. ...	0	36
21	These candy bars were old and stale. I should...	0	33
...	...	...	...
99972	I switched my standing order to Meal Bars (10 ...	0	42
99982	This blend is bitter and does taste earthly l...	0	21
99983	The tropical fruit in this drink just didn't w...	0	30
99986	I bought some last week and I thought it was r...	0	43
99992	I'm usually impressed with this brand's stevia...	0	43

12996 rows × 3 columns

In [ ]:

```
# Remove HTML from the Text column and save in the Text column only
# Split the data into train and test data(20%) with Stratify sampling, random state 33
# Plot bar graphs of y_train and y_test
```

In [ ]:

```
rev = []
for sentence in reviews.Text.values:
    t = BeautifulSoup(sentence, "lxml").get_text()
    rev.append(t.strip())
```

In [ ]:

```
reviews['Text'] = rev
reviews
```

Out[12]:

	Text	Score	len
0	The tea was of great quality and it tasted lik...	1	30
1	My cat loves this. The pellets are nice and s...	1	31
2	Great product. Does not completely get rid of ...	1	41
3	This gum is my favorite! I would advise every...	1	27
4	I also found out about this product because of...	1	22
...	...	...	...
99995	Using this coffee and a stove top espresso mak...	1	39
99996	THE TASTE OF THIS M&M IS THE BEST. I USED IT I...	1	28
99997	Excellent Tea. I enjoy a cup every now and the...	1	21
99998	These oatmeal cookies have a great spice taste...	1	23
99999	This is the best coffee ever! I will never dri...	1	28

100000 rows × 3 columns

In [ ]:

```
#saving to disk. if we need, we can load preprocessed data directly.
reviews.to_csv('preprocessed.csv', index=False)
```

In [28]:

```
Positive_Reviews = reviews[reviews['Score']==1]
print(Positive_Reviews.shape)
#Positive_Reviews
```

```
(87004, 3)
```

In [29]:

```
Negative_Reviews = reviews[reviews['Score']==0]
print(Negative_Reviews.shape)
#Negative_Reviews
```

```
(12996, 3)
```

In [30]:

```
Positive_Reviews['Text'] = Positive_Reviews['Text'].apply(lambda x: x.lower())
Positive_Reviews
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  """Entry point for launching an IPython kernel.
```

Out[30]:

	Text	Score	len
0	the tea was of great quality and it tasted lik...	1	30
1	my cat loves this. the pellets are nice and s...	1	31
2	great product. does not completely get rid of ...	1	41
3	this gum is my favorite! i would advise every...	1	27
4	i also found out about this product because of...	1	22
...	...	...	...
99995	using this coffee and a stove top espresso mak...	1	39
99996	the taste of this m&m is the best. i used it i...	1	28
99997	excellent tea. i enjoy a cup every now and the...	1	21
99998	these oatmeal cookies have a great spice taste...	1	23
99999	this is the best coffee ever! i will never dri...	1	28

87004 rows × 3 columns

In [31]:

```
Negative_Reviews['Text'] = Negative_Reviews['Text'].apply(lambda x: x.lower())
Negative_Reviews
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  """Entry point for launching an IPython kernel.
```

Out[31]:

	Text	Score	len
8	never received product. contacted seller askin...	0	42
10	for a shipping charge of over $8, you'd think ...	0	44
15	despite the picture, and that it shows up when...	0	24
20	i thought i would love this...i love coconut. ...	0	36
21	these candy bars were old and stale. i should...	0	33
...	...	...	...
99972	i switched my standing order to meal bars (10 ...	0	42
99982	this blend is bitter and does taste earthly l...	0	21
99983	the tropical fruit in this drink just didn't w...	0	30
99986	i bought some last week and i thought it was r...	0	43
99992	i'm usually impressed with this brand's stevia...	0	43

12996 rows × 3 columns

In [32]:

```
comment_words = ''
for texts in Positive_Reviews['Text'].values:
    texts = str(texts)
    comment_words += " " + texts + " "
```

In [33]:

```
# Word Cloud visualisation of Positive reviews.
stopwords_ = set(STOPWORDS)
wordcloud = WordCloud(width = 800, height = 800, background_color = 'white', stopwords = st

plt.figure(figsize = (10, 10), facecolor = None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

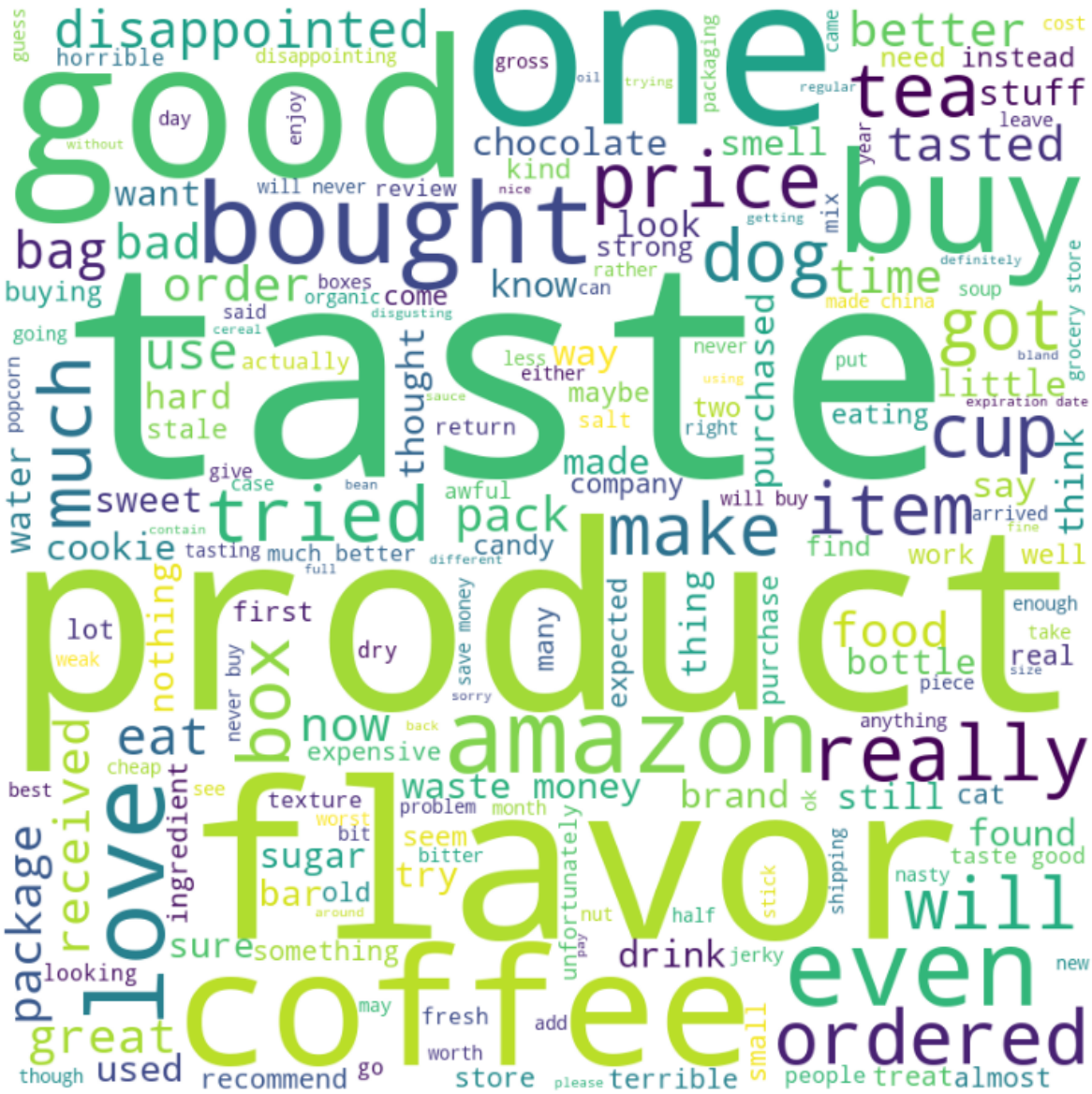
In [37]:

```
comment_words = ''
for texts in Negative_Reviews['Text'].values:
    texts = str(texts)
    comment_words += " " + texts + " "
```

In [38]:

```
# Word Cloud visualisation of Negative reviews.
wordcloud = WordCloud(width = 800, height = 800, background_color = 'white', stopwords = st

plt.figure(figsize = (10, 10), facecolor = None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```



In [39]:

```
del Positive_Reviews
del Negative_Reviews
```

In [ ]:

```
x_train, x_val, y_train, y_val = train_test_split(reviews.Text, reviews.Score, test_size =
```

In [ ]:

```
np.bincount(y_train.values)
```

Out[21]:

```
array([10397, 69603])
```

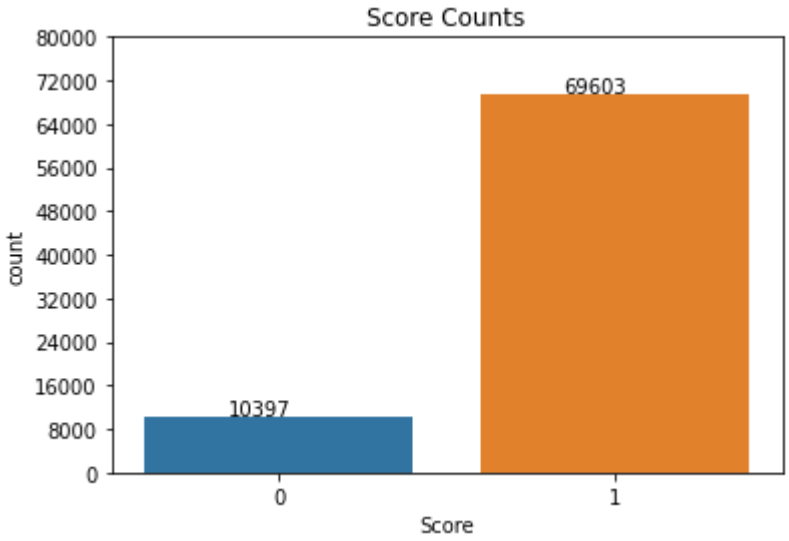
In [ ]:

```
# Count of Train Scores.
total = len(y_train)
print(Counter(y_train))
ax = sns.countplot(y_train)
for p in ax.patches:
  ax.annotate('{}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+5))

ax.yaxis.set_ticks(np.linspace(0, total, 11))
plt.title("Score Counts")
plt.show()
```

```
Counter({1: 69603, 0: 10397})

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



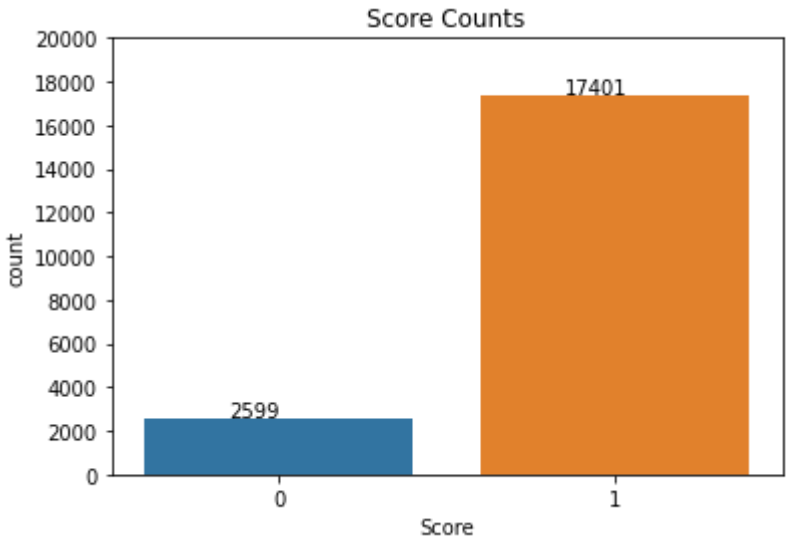
In [ ]:

```
# Count of Val Scores.
total = len(y_val)
print(Counter(y_val))
ax = sns.countplot(y_val)
for p in ax.patches:
  ax.annotate('{}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+5))

ax.yaxis.set_ticks(np.linspace(0, total, 11))
plt.title("Score Counts")
plt.show()
```

```
Counter({1: 17401, 0: 2599})

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



From above, it is seen that classes are imbalanced.

Part-2: Creating BERT Model

If you want to know more about BERT, You can watch live sessions on Transformers and BERt. we will strongly recommend you to read Transformers (https://jalammar.github.io/illustrated-transformer/), BERT Paper (https://arxiv.org/abs/1810.04805) and, This blog (https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/).

For this assignment, we are using BERT uncased Base model (https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1). It uses **L=12 hidden layers (i.e., Transformer blocks), a hidden size of H=768, and A=12 attention heads.**

In [ ]:

```
## Loading the Pretrained Model from tensorflow HUB
tf.keras.backend.clear_session()

# maximum length of a seq in the data we have, for now i am making it as 55. You can change
max_seq_length = 55

#BERT takes 3 inputs

#this is input words. Sequence of words represented as integers
input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input

#mask vector if you are padding anything
input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_mas

#segment vectors. If you are giving only one sentence for the classification, total seg vec
#If you are giving two sentenced with [sep] token separated, first seq segment vectors are
#second seq segment vector are 1's
segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_
```

In [ ]:

```
#bert layer
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
```

In [ ]:

```
#Bert model
#We are using only pooled output not sequence out.
#If you want to know about those, please read https://www.kaggle.com/questions-and-answers/
bert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_output)
```

In [ ]:

```
bert_model.summary()
```

```
Model: "model"
__________________________________________________________________________________
_____________________
Layer (type)                    Output Shape         Param #     Connected t
o
==================================================================================
=====================
input_word_ids (InputLayer)     [(None, 55)]         0
__________________________________________________________________________________
_____________________
input_mask (InputLayer)         [(None, 55)]         0
__________________________________________________________________________________
_____________________
segment_ids (InputLayer)        [(None, 55)]         0
__________________________________________________________________________________
_____________________
keras_layer (KerasLayer)        [(None, 768), (None, 109482241   input_word_
ids[0][0]
                                                                 input_mask
[0][0]
                                                                 segment_ids
[0][0]
==================================================================================
=====================
Total params: 109,482,241
Trainable params: 0
Non-trainable params: 109,482,241
__________________________________________________________________________________
_____________________
```

In [ ]:

```
bert_model.output
```

Out[31]:

```
<KerasTensor: shape=(None, 768) dtype=float32 (created by layer 'keras_laye
r')>
```

Part-3: Tokenization

In [ ]:

```
#Getting Vocab file
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
```

In [ ]:

```
!pip install sentencepiece
```

```
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/f5/99/e0808cb947ba10f5
75839c43e8fafc9cc44e4a7a2c8f79c60db48220a577/sentencepiece-0.1.95-cp37-cp37m
-manylinux2014_x86_64.whl (https://files.pythonhosted.org/packages/f5/99/e08
08cb947ba10f575839c43e8fafc9cc44e4a7a2c8f79c60db48220a577/sentencepiece-0.1.
95-cp37-cp37m-manylinux2014_x86_64.whl) (1.2MB)
     |████████████████████████████████| 1.2MB 2.2MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.95
```

We are`using tokenizer given by **Tensorflow.** **(https://github.com/tensorflow/models/tree/master/official/nlp/bert)**

In [ ]:

```
#import tokenization -  tokenization.py file
import tokenization
```

In [ ]:

```
# Create tokenizer " Instantiate FullTokenizer"
# Name must be "tokenizer"
# The FullTokenizer takes two parameters 1. vocab_file and 2. do_lower_case
# We have created these in the above cell ex: FullTokenizer(vocab_file, do_lower_case )
# Please check the "tokenization.py" file the complete implementation
```

In [ ]:

```
tokenizer = tokenization.FullTokenizer(vocab_file, do_lower_case)
```

Grader function 3

In [ ]:

```
#it has to give no error
def grader_tokenize(tokenizer):
    out = False
    try:
        out=('[CLS]' in tokenizer.vocab) and ('[SEP]' in tokenizer.vocab)
    except:
        out = False
    assert(out==True)
    return out
grader_tokenize(tokenizer)
```

Out[37]:

```
True
```

In [ ]:

```
# Create train and test tokens (X_train_tokens, X_test_tokens) from (X_train, X_test) using

# add '[CLS]' at start of the Tokens and '[SEP]' at the end of the tokens.

# maximum number of tokens is 55(We already given this to BERT layer above) so shape is (No

# if it is less than 55, add '[PAD]' token else truncate the tokens length.(similar to padd

# Based on padding, create the mask for Train and Test ( 1 for real token, 0 for '[PAD]'),
# it will also same shape as input tokens (None, 55) save those in X_train_mask, X_test_mas

# Create a segment input for train and test. We are using only one sentence so all zeros. T

# type of all the above arrays should be numpy arrays

# after execution of this cell, you have to get
# X_train_tokens, X_train_mask, X_train_segment
# X_test_tokens, X_test_mask, X_test_segment
```

In [ ]:

```
def Tokens_to_Ids(tokens, tokenizer, max_seq_length):

    if len(tokens) > (max_seq_length-2):
        tokens = tokens[:max_seq_length-2]
        tokens = ["[CLS]"] + tokens + ["[SEP]"]
        token_2_ids = tokenizer.convert_tokens_to_ids(tokens)
        return np.array(token_2_ids)
    else:
        tokens = ["[CLS]"] + tokens + ["[SEP]"]
        tokens = tokens + ["[PAD]"] * (max_seq_length - len(tokens))
        token_2_ids = tokenizer.convert_tokens_to_ids(tokens)
        return np.array(token_2_ids)
```

In [ ]:

```
def Masks(tokens, max_seq_length):
    #tokens = ["[CLS]"] + tokens + ["[SEP]"] --> len(tokens) + 2
    if (len(tokens) + 2) > (max_seq_length):
        mask = [1] * max_seq_length
        return np.array(mask)
    else:
        mask = [1] * (len(tokens) + 2) + [0] * (max_seq_length - (len(tokens) + 2))
        return np.array(mask)
```

In [ ]:

```
# For Train tokens.
X_train = []
X_train_tokens = []
X_train_mask = []

for item in (tqdm_notebook(x_train.values)):
    tokens = tokenizer.tokenize(item)
    X_train.append(tokens)

    T2I = Tokens_to_Ids(tokens, tokenizer, max_seq_length)
    X_train_tokens.append(T2I)

    mask = Masks(tokens, max_seq_length)
    X_train_mask.append(mask)

X_train = np.array(X_train)
X_train_tokens = np.array(X_train_tokens)
X_train_mask = np.array(X_train_mask)
X_train_segment = np.zeros((len(x_train), max_seq_length))
X_train_tokens.shape, X_train_mask.shape, X_train_segment.shape
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: TqdmDeprecat
ionWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  """

HBox(children=(FloatProgress(value=0.0, max=80000.0), HTML(value='')))



/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a
list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shape
s) is deprecated. If you meant to do this, you must specify 'dtype=object' w
hen creating the ndarray
  from ipykernel import kernelapp as app
```

Out[41]:

```
((80000, 55), (80000, 55), (80000, 55))
```

In [ ]:

```
# For Val tokens.
X_val = []
X_val_tokens = []
X_val_mask = []

for item in (tqdm_notebook(x_val.values)):
    tokens = tokenizer.tokenize(item)
    X_val.append(tokens)

    T2I = Tokens_to_Ids(tokens, tokenizer, max_seq_length)
    X_val_tokens.append(T2I)

    mask = Masks(tokens, max_seq_length)
    X_val_mask.append(mask)

X_val = np.array(X_val)
X_val_tokens = np.array(X_val_tokens)
X_val_mask = np.array(X_val_mask)
X_val_segment = np.zeros((len(x_val), max_seq_length))
X_val_tokens.shape, X_val_mask.shape, X_val_segment.shape
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: TqdmDeprecat
ionWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  """

HBox(children=(FloatProgress(value=0.0, max=20000.0), HTML(value='')))



/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a
list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shape
s) is deprecated. If you meant to do this, you must specify 'dtype=object' w
hen creating the ndarray
  from ipykernel import kernelapp as app
```

Out[42]:

```
((20000, 55), (20000, 55), (20000, 55))
```

Example

```
print("original sentance : \n", np.array(X_train.values[0].split()))
print("number of words: ", len(X_train.values[0].split()))
print('='*50)
tokens = tokenizer.tokenize(X_train.values[0])
# we need to do this "tokens = tokens[0:(max_seq_length-2)]" only when our len(tokens) is more than "max_seq_length - 2"
# we will consider only the tokens from 0 to max_seq_length-2
# if our len(tokens) are < max_seq_length-2, we don't need to do this
tokens = tokens[0:(max_seq_length-2)]
# we are doing that so that we can include the tokens [CLS] and [SEP] and make the whole sequence length == max_seq_length
tokens = ['[CLS]',*tokens,'[SEP]']
print("tokens are: \n", np.array(tokens))
print('='*50)
print("number of tokens :",len(tokens))
print("tokens replaced with the positional encoding :\n",np.array(tokenizer.convert_tokens_to_ids(tokens)))
print('='*50)
print("the mask array is : ", np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens))))
print('='*50)
print("the segment array is :",np.array([0]*max_seq_length))
print('='*50)
```

```
original sentance :
 ['I' 'had' 'never' 'tried' 'this' 'brand' 'before,' 'so' 'I' 'was'
 'worried' 'about' 'the' 'quality.' 'It' 'tasted' 'great.' 'A' 'very'
 'nice' 'smooth' 'rich' 'full' 'flavor.' 'Its' 'my' 'new' 'favoret.']
number of words:  28
==================================================
tokens are:
 ['[CLS]' 'i' 'had' 'never' 'tried' 'this' 'brand' 'before' ',' 'so' 'i'
 'was' 'worried' 'about' 'the' 'quality' '.' 'it' 'tasted' 'great' '.' 'a'
 'very' 'nice' 'smooth' 'rich' 'full' 'flavor' '.' 'its' 'my' 'new'
 'favor' '##et' '.' '[SEP]']
==================================================
number of tokens : 36
tokens replaced with the positional encoding :
 [  101  1045  2018  2196  2699  2023  4435  2077  1010  2061  1045  2001
  5191  2055  1996  3737  1012  2009 12595  2307  1012  1037  2200  3835
  5744  4138  2440 14894  1012  2049  2026  2047  5684  3388  1012   102]
==================================================
the mask array is :  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
==================================================
the segment array is : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
==================================================
```

In [ ]:

```
##save all your results to disk so that, no need to run all again.
pickle.dump((X_train, X_train_tokens, X_train_mask, X_train_segment, y_train),open('train_d
pickle.dump((X_val, X_val_tokens, X_val_mask, X_val_segment, y_val),open('val_data.pkl','wb
```

In [ ]:

```
#you can load from disk
X_train, X_train_tokens, X_train_mask, X_train_segment, y_train = pickle.load(open("train_d
X_test, X_test_tokens, X_test_mask, X_test_segment, y_test = pickle.load(open("val_data.pkl
```

Grader function 4

In [ ]:

```
def grader_alltokens_train():
    out = False

    if type(X_train_tokens) == np.ndarray:

        temp_shapes = (X_train_tokens.shape[1]==max_seq_length) and (X_train_mask.shape[1]=
        (X_train_segment.shape[1]==max_seq_length)

        segment_temp = not np.any(X_train_segment)

        mask_temp = np.sum(X_train_mask==0) == np.sum(X_train_tokens==0)

        no_cls = np.sum(X_train_tokens==tokenizer.vocab['[CLS]'])==X_train_tokens.shape[0]

        no_sep = np.sum(X_train_tokens==tokenizer.vocab['[SEP]'])==X_train_tokens.shape[0]

        out = temp_shapes and segment_temp and mask_temp and no_cls and no_sep

    else:
        print('Type of all above token arrays should be numpy array not list')
        out = False
    assert(out==True)
    return out

grader_alltokens_train()
```

Out[47]:

```
True
```

Grader function 5

In [ ]:

```
def grader_alltokens_test():
    out = False
    if type(X_test_tokens) == np.ndarray:

        temp_shapes = (X_test_tokens.shape[1]==max_seq_length) and (X_test_mask.shape[1]==m
        (X_test_segment.shape[1]==max_seq_length)

        segment_temp = not np.any(X_test_segment)

        mask_temp = np.sum(X_test_mask==0) == np.sum(X_test_tokens==0)

        no_cls = np.sum(X_test_tokens==tokenizer.vocab['[CLS]'])==X_test_tokens.shape[0]

        no_sep = np.sum(X_test_tokens==tokenizer.vocab['[SEP]'])==X_test_tokens.shape[0]

        out = temp_shapes and segment_temp and mask_temp and no_cls and no_sep

    else:
        print('Type of all above token arrays should be numpy array not list')
        out = False
    assert(out==True)
    return out
grader_alltokens_test()
```

Out[48]:

```
True
```

Part-4: Getting Embeddings from BERT Model

We already created the BERT model in the part-2 and input data in the part-3. We will utlize those two and will get the embeddings for each sentence in the Train and test data.

In [ ]:

```
bert_model.input
```

Out[49]:

```
[<KerasTensor: shape=(None, 55) dtype=int32 (created by layer 'input_word_id
s')>,
 <KerasTensor: shape=(None, 55) dtype=int32 (created by layer 'input_mask')
>,
 <KerasTensor: shape=(None, 55) dtype=int32 (created by layer 'segment_ids')
>]
```

In [ ]:

```
bert_model.output
```

Out[50]:

```
<KerasTensor: shape=(None, 768) dtype=float32 (created by layer 'keras_laye
r')>
```

In [ ]:

```
# get the train output, BERT model will give one output so save in
# X_train_pooled_output
X_train_pooled_output=bert_model.predict([X_train_tokens,X_train_mask,X_train_segment])
```

In [ ]:

```
X_train_pooled_output.shape
```

Out[52]:

```
(80000, 768)
```

In [ ]:

```
# get the test output, BERT model will give one output so save in
# X_test_pooled_output
X_test_pooled_output=bert_model.predict([X_test_tokens,X_test_mask,X_test_segment])
```

In [ ]:

```
X_test_pooled_output.shape
```

Out[54]:

```
(20000, 768)
```

In [ ]:

```
##save all your results to disk so that, no need to run all again.
pickle.dump((X_train_pooled_output, X_test_pooled_output),open('final_output.pkl','wb'))
```

In [ ]:

```
#X_train_pooled_output, X_test_pooled_output= pickle.load(open('final_output.pkl', 'rb'))
```

Grader function 6

In [ ]:

```
#now we have X_train_pooled_output, y_train
#X_test_pooled_ouput, y_test

#please use this grader to evaluate
def greader_output():
    assert(X_train_pooled_output.shape[1]==768)
    assert(len(y_train)==len(X_train_pooled_output))
    assert(X_test_pooled_output.shape[1]==768)
    assert(len(y_test)==len(X_test_pooled_output))
    assert(len(y_train.shape)==1)
    assert(len(X_train_pooled_output.shape)==2)
    assert(len(y_test.shape)==1)
    assert(len(X_test_pooled_output.shape)==2)
    return True
greader_output()
```

Out[56]:

```
True
```

Part-5: Training a NN with 768 features

Create a NN and train the NN.

1. **You have to use AUC as metric.**
2. You can use any architecture you want.
3. You have to use tensorboard to log all your metrics and Losses.
4. Print the loss and metric at every epoch.
5. You have to submit without overfitting and underfitting.

In [ ]:

```
test = 'model_save/*.hdf5'
r = glob.glob(test)
for i in r:
    os.remove(i)
```

In [ ]:

```
filepath="model_save/model-{epoch:02d}-{val_auc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_auc',  verbose=1, save_best_on
```

In [ ]:

```
%load_ext tensorboard
shutil.rmtree("./logs/", ignore_errors = True )
```

In [ ]:

```
log_dir="logs\\fit\\" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, wri
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.
```

In [ ]:

```
earlystop = EarlyStopping(monitor='val_loss', patience=5, verbose=1, restore_best_weights=T
```

In [ ]:

```
callbacks = [checkpoint, tensorboard_callback, earlystop]
```

In [ ]:

```
##create an NN
tf.keras.backend.clear_session()
model = Sequential()
model.add(Dense(256,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(se
model.add(Dropout(0.4))
model.add(Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(see
model.add(Dense(32,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(see
model.add(Dense(16,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(see
model.add(Dropout(0.2))
model.add(Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_norm
```

In [ ]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
FC1 (Dense)                  (None, 256)               196864
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
FC3 (Dense)                  (None, 64)                16448
_________________________________________________________________
FC4 (Dense)                  (None, 32)                2080
_________________________________________________________________
FC5 (Dense)                  (None, 16)                528
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
Out_layer (Dense)            (None, 1)                 17
=================================================================
Total params: 215,937
Trainable params: 215,937
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate= 0.001), loss = 'binary_cr
```

In [ ]:

```
model.fit(X_train_pooled_output, y_train, epochs=40, batch_size=256, verbose=1, validation_
```

```
Epoch 1/40
313/313 [==============================] - 3s 7ms/step - loss: 0.3290 - auc:
0.7708 - val_loss: 0.2262 - val_auc: 0.9191

Epoch 00001: val_auc improved from -inf to 0.91914, saving model to model_sa
ve/model-01-0.9191.hdf5
Epoch 2/40
313/313 [==============================] - 2s 5ms/step - loss: 0.2441 - auc:
0.8989 - val_loss: 0.2058 - val_auc: 0.9345

Epoch 00002: val_auc improved from 0.91914 to 0.93452, saving model to model
_save/model-02-0.9345.hdf5
Epoch 3/40
313/313 [==============================] - 1s 4ms/step - loss: 0.2354 - auc:
0.9079 - val_loss: 0.2124 - val_auc: 0.9320

Epoch 00003: val_auc did not improve from 0.93452
Epoch 4/40
313/313 [==============================] - 1s 5ms/step - loss: 0.2401 - auc:
0.9041 - val_loss: 0.2057 - val_auc: 0.9372

Epoch 00004: val_auc improved from 0.93452 to 0.93720, saving model to model
_save/model-04-0.9372.hdf5
Epoch 5/40
313/313 [==============================] - 1s 4ms/step - loss: 0.2553 - auc:
0.8879 - val_loss: 0.2134 - val_auc: 0.9383

Epoch 00005: val_auc improved from 0.93720 to 0.93828, saving model to model
_save/model-05-0.9383.hdf5
Epoch 6/40
313/313 [==============================] - 2s 5ms/step - loss: 0.2367 - auc:
0.9065 - val_loss: 0.2052 - val_auc: 0.9413

Epoch 00006: val_auc improved from 0.93828 to 0.94126, saving model to model
_save/model-06-0.9413.hdf5
Epoch 7/40
313/313 [==============================] - 1s 4ms/step - loss: 0.2370 - auc:
0.9064 - val_loss: 0.2181 - val_auc: 0.9410

Epoch 00007: val_auc did not improve from 0.94126
Epoch 8/40
313/313 [==============================] - 2s 5ms/step - loss: 0.2280 - auc:
0.9149 - val_loss: 0.2390 - val_auc: 0.9414

Epoch 00008: val_auc improved from 0.94126 to 0.94135, saving model to model
_save/model-08-0.9414.hdf5
Epoch 9/40
313/313 [==============================] - 1s 5ms/step - loss: 0.2298 - auc:
0.9142 - val_loss: 0.2198 - val_auc: 0.9446

Epoch 00009: val_auc improved from 0.94135 to 0.94456, saving model to model
_save/model-09-0.9446.hdf5
Epoch 10/40
313/313 [==============================] - 1s 5ms/step - loss: 0.2264 - auc:
0.9174 - val_loss: 0.2246 - val_auc: 0.9393
```

```
Epoch 00010: val_auc did not improve from 0.94456
Epoch 11/40
313/313 [==============================] - 1s 5ms/step - loss: 0.2261 - auc:
0.9165 - val_loss: 0.2191 - val_auc: 0.9416

Epoch 00011: val_auc did not improve from 0.94456
Restoring model weights from the end of the best epoch.
Epoch 00011: early stopping
```

Out[179]:

```
<tensorflow.python.keras.callbacks.History at 0x7f0035fa72d0>
```

In [ ]:

```
model.evaluate(X_test_pooled_output, y_val, batch_size = 256)
```

```
79/79 [==============================] - 0s 3ms/step - loss: 0.2052 - auc:
0.9413
```

Out[180]:

```
[0.20523276925086975, 0.9412624835968018]
```

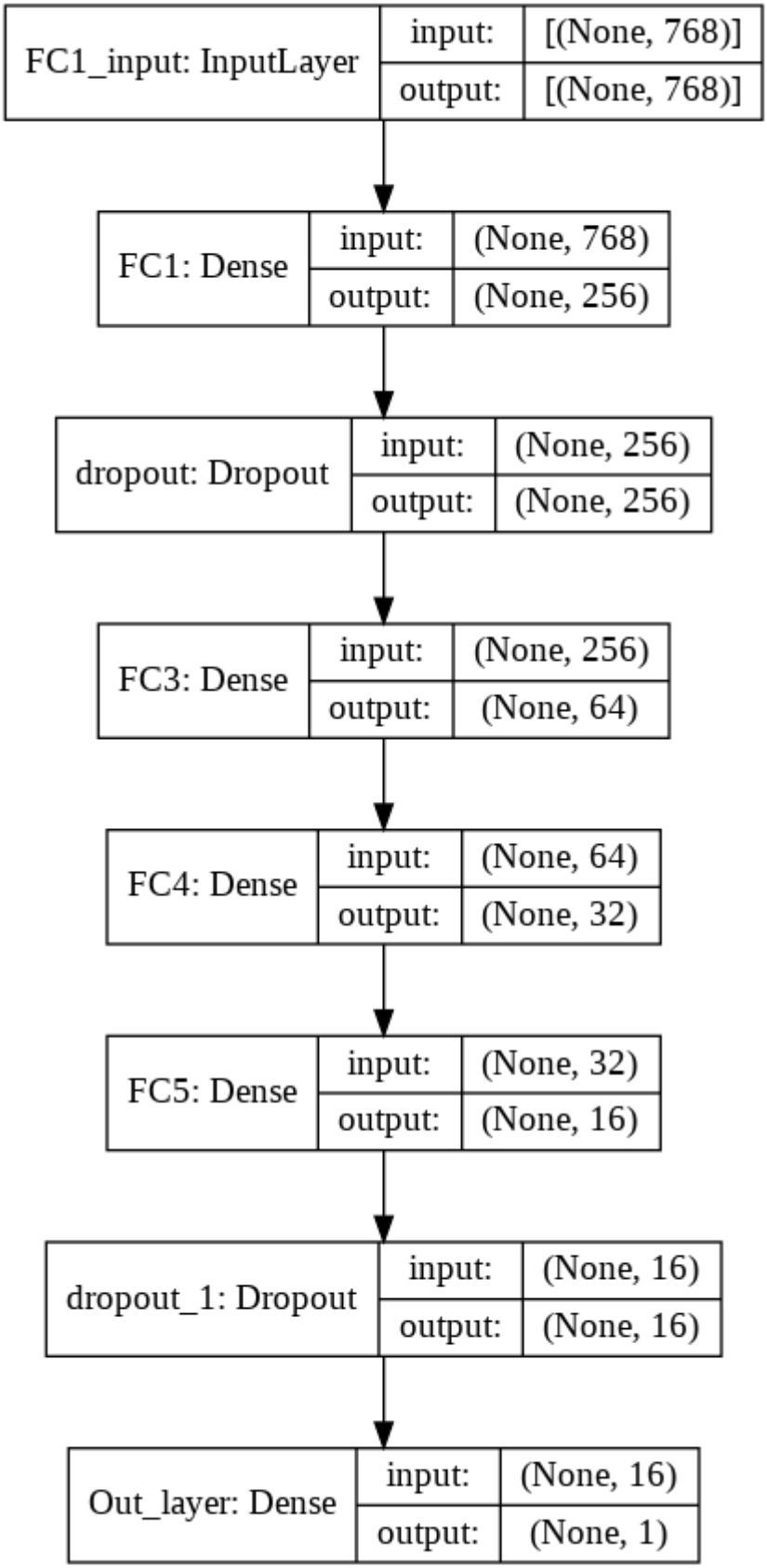
In [ ]:

```
model.save('Model_final.h5')
```

In [ ]:

```
from tensorflow.keras.utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

Out[182]:



In [ ]:

```
%tensorboard --logdir logs\\fit\\20210610-063902
```

```
<IPython.core.display.Javascript object>
```

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

train

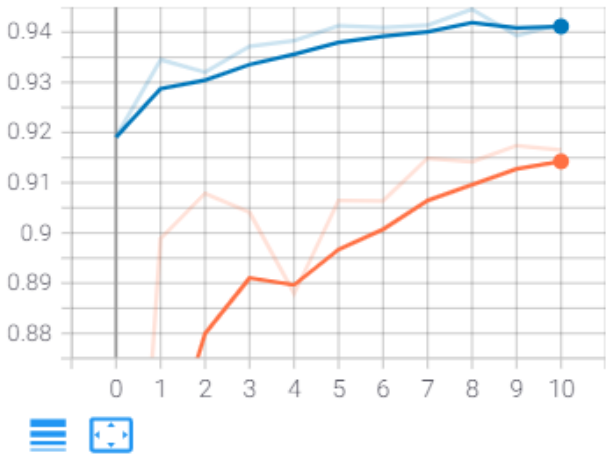
validation

TOGGLE ALL RUNS

logs\fit\20210610-063902

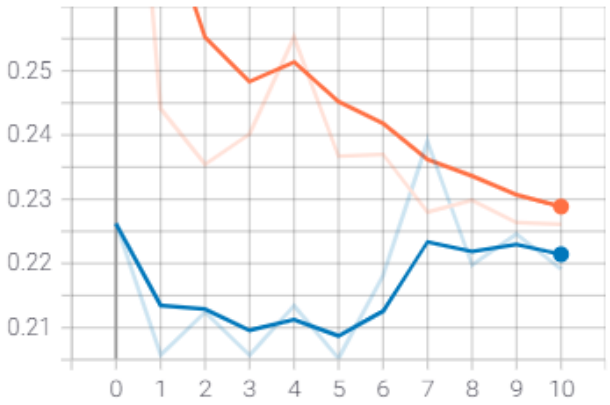
epoch_auc

epoch_auc
tag: epoch_auc



epoch_loss

epoch_loss
tag: epoch_loss



In [ ]:

```
def Pipe(reviews):
    max_seq_length = 55
    rev = []
    for sentence in reviews.Text.values:
        t = BeautifulSoup(sentence, "lxml").get_text()
        rev.append(t.strip())

    #getting Vocab file
    vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
    do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()

    tokenizer = tokenization.FullTokenizer(vocab_file, do_lower_case)

    X_test = []
    X_test_tokens = []
    X_test_mask = []

    for item in (tqdm_notebook(rev)):
        tokens = tokenizer.tokenize(item)
        X_test.append(tokens)

        T2I = Tokens_to_Ids(tokens, tokenizer, max_seq_length)
        X_test_tokens.append(T2I)

        mask = Masks(tokens, max_seq_length)
        X_test_mask.append(mask)

    X_test = np.array(X_test)
    X_test_tokens = np.array(X_test_tokens)
    X_test_mask = np.array(X_test_mask)
    X_test_segment = np.zeros((len(X_test), max_seq_length))
    X_test_tokens.shape, X_test_mask.shape, X_train_segment.shape

    X_test_pooled_output = bert_model.predict([X_test_tokens,X_test_mask,X_test_segment])

    return X_test_pooled_output
```

Summary

1. The task is to learn how to use pre-trained BERT from tensorflow hub and use it for Sentiment Analysis.
2. BERT model

   - Uncased Base Model.
   - 12 Encoder Layers(Stacks).
   - 12 Attention Heads.
   - Hidden units of 768 dim.

3. The given review scores are modified suitably for Biary Classification Task.

4. Output of BERT model which is **Sentence Embeddings** is used for the classification of the reviews using NN.
5. By using this Sentence Embeddings and NN, **0.9416 AUC** is achieved on Val Data.