# I - Problem Statement

- **Given** - A Data Set of Fake and Real news.
- **Objective** - To develop a solution which detects if a given news is Fake or Real.
- **Methodology used** - We try to pose the problem as a text classification problem and build a deep learning model for achieving the objective.

### Different Models

While there multiple types of models which could be used for building the solution for Text Classification. Some Examples are -

- **1D- Conv Net** - Yes, CNNs could also be used for text. **Advantage**: They are faster to ttrain. In fact given proper label, a CNN model could achieve decent accuracy **Disdvantage**: They fail to capture long term dependencies in Text, and doesnt not capture sequential information in text.
- **RNN based models (LSTM, GRU) ** - **Advantage** : They focus on considering words T each time step, encoding them by some non-linear calculation and then taking a decision. In simple words, they can capture sequential nature of a Text.
- *Disadvantage** - Slower to train, focus is more on sequential nature, and less on attention mechanism.
- **Transformer based Models(BERT, GPT2)**- Transformer based models are a breaktrhough in NLP, and tend to model. Which leverages multiple Transformer units, and a multi-headed attention mechanism. The advantage is that they focus only on attention mechanism. Thus we obtain a model, which is can used in context heavy applications.

# II - StandAlone BERT Model -

- For our solution we will be using BERT model to develop Fake News or Real News Classification Solution.
- We achieved an accuracy of 95+ % on test set, and a remarkable AUC by a standalone BERT Model. More improvements could be done with better tuning, and training for longer time. In cloud settings like Google Cloud(with larger GPUs) or AWS infra. But improvement is a continuous process :)

- We build an MVP with BERT Stand-alone model
- We can Also use, BERT (as an embedder) + LSTM model to build this solution.




Above figure shows the Kind of model we will be building in developing our solution.

# III - Coding Environment

- I chose Google Colab for two reasons primarily - Firstly, To Document the process step by step. Seoncdly, other than this I wanted to leverage the free GPU available in Google Colab.
- The avaialble GPU helped me train faster, compared to my own PC.(well, you might notice that even the the resource, got exhausted after 2 epochs, but we created a checkpoint to save the model)
- In production as well, GPU powered AWS frameweorks such as AWS Sagemkaer and Google GPU cloud infra are useful in training models and deploying quicker.
- For coding environment we can develop models using Keras or Tensorflow. Depending upon the level of control we want on model creation, we can work with that TF versions and utilities such as core, and functional APIs.
- For more control and inner workings of
  model, TF core comes is usually very useful, and for quicker prototyping we can go for Keras or TF2.0 which is keras style.
- I chose to go with Tensorflow 2.0 for BERT TF.2.0 model for classification.

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

# IV - Data Preprocessing and Exploration

## Loading Data

Our data are json files stored on web, therefore we download it and convert it into pandas dataframe

```
import urllib
import json

def load_convert_data(url):
    """
    Downloads the json file from net and convert into pandas dataframe format.
    """
    with urllib.request.urlopen(url) as url:
        df = json.loads(url.read().decode())
        df = pd.DataFrame.from_dict(df)

    return df
```

Given:

There are 4 files:

- training - set of real news.
- testing - set of real news.
- training - set of fake news.
- testing - set of fake news

```
# Real news data
real_train = load_convert_data("https://storage.googleapis.com/public-resources/dataset/real_
real_test = load_convert_data("https://storage.googleapis.com/public-resources/dataset/real_t


# Fake news data
fake_train = load_convert_data("https://storage.googleapis.com/public-resources/dataset/fake_
fake_test = load_convert_data("https://storage.googleapis.com/public-resources/dataset/fake_t


# quick look on real news training data
real_train.head()
```

| | url | title | |
|---|---|---|---|
| **0** | https://www.thetimes.co.uk/edition/scotland/sc... | Scots GPs told not to meet fever patients as f... | Scots GPs told |
| **1** | https://www.bbc.com/news/world-africa-52103799 | Coronavirus : Fighting al - Shabab propaganda ... | Coronavirus: F |
| **2** | https://www.thetimes.co.uk/edition/business/en... | Engineer fears China virus impact | Engineer imp |
| | | Coronavirus : South Korean PM | Here’s a s |

**Observation** : -

- We can see that there are 800 rows and 3 columns for real news in training set, we will only use the 'text' column for modeling (for simplicity sake).
- In case if the model doesnt perform well, we can use multiple features like url as well.
- However, we acheived good performance by using text data alone.

```
# Quick look on Fake news training data
fake_train.head()
```

| | url | title | |
|---|---|---|---|
| **0** | https://nabd.com/s/71539812-b7228b/%D9%86%D8%B... | Online Facts New conspiracy theory: #Bel_Gates... | Roger Stone sugg Monday that B |
| **1** | https://shamra.sy/news/article/8eb73454931e6d1... | Revolutionary Guards: Corona could be an Ameri... | Source\nRuss \|\nIranian Revolut |
| **2** | https://sudanewsnow.com/19800/ | Yellow skin is the host environment of the vir | Sudan news nov sources sudanews |

**Observations** :-

- Training set of Fake news contain also 800 rows.
- So we can see that the number of real news and fake news are same in our dataset.
- It won't be an imbalanced classification problem.

## General Data Preprocessing

Next we label our data where real news are labeled as 0 (negative) and fake news are labeled as 1 (positive). The reason we label fake news as positive is that the main purpose of the modeling is to detect fake news.

```
real_train['label'] = 0
```

```
real_test['label'] = 0
fake_train['label'] = 1
fake_test['label'] = 1

train = pd.concat([real_train, fake_train], ignore_index=True)
test = pd.concat([real_test, fake_test], ignore_index=True)
```

We then remove non alphanumeric characters as well as converting to all lower case from the text.

```
import re
def clean_txt(text):
    text = re.sub("'", "", text)
    text = re.sub("(\\W)+", " ", text)
    text = text.lower()
    return text

train['text'] = train['text'].apply(clean_txt)
test['text'] = test['text'].apply(clean_txt)
```

## Plotting Data

### Word Count histogram

[ ] ↳ *7 cells hidden*

### Word Cloud

[ ] ↳ *6 cells hidden*

### TopK Word Proportion

[ ] ↳ *4 cells hidden*

# V - Modeling

For this project, we use BERT as our modeling algorithm.

## Splitting Data to Train/Validation

First we like to split our training set into training and validation set with a ratio of 8:2, this way we

```
from sklearn.model_selection import train_test_split
train, val = train_test_split(train, test_size=0.2, random_state=35)
```

## Long Document Preprocessing

Since BERT algorithm can only accept sentence length up to 512 words, we need to preprocess our data (long news) in order to feed in to the algorithm. To do so, we follow the idea from this paper and segment each of the text into multiple subtext of no longer than 150 words. The subtexts will have some overlapping, specifically, the last 30 words for first subtext will be the first 30 words of the second subtext.

```
def get_split(text):
    """
    Split each news text to subtexts no longer than 150 words.
    """
    l_total = []
    l_parcial = []
    if len(text.split())//120 >0:
        n = len(text.split())//120
    else:
        n = 1
    for w in range(n):
        if w == 0:
            l_parcial = text.split()[:150]
            l_total.append(" ".join(l_parcial))
        else:
            l_parcial = text.split()[w*120:w*120 + 150]
            l_total.append(" ".join(l_parcial))
    return l_total

train['text_split'] = train['text'].apply(get_split)
val['text_split'] = val['text'].apply(get_split)
test['text_split'] = test['text'].apply(get_split)
```

```
train['text_split'][1]
```

```
['coronavirus fighting al shabab propaganda in somalia muslim clerics in conflict
hit somalia are moving to the front line of the battle against coronavirus in a bid
to counter the propaganda of militant islamists writes bbc somalia analyst mary
harper the al qaeda linked al shabab group has warned muslims to beware of
infectious diseases such as coronavirus which it says are spread by the crusader
forces who have invaded the country and the disbelieving countries that support them
the militants control much of southern and central somalia and have been a powerful
force for well over a decade although there have only been a handful of confirmed
cases of coronavirus in somalia so far the authorities are deeply worried that if
```

```
 'the disease takes hold they will be unable to cope koranic schools shut about 30
years of conflict have devastated health facilities hundreds of thousands of
displaced people',
 'that if the disease takes hold they will be unable to cope koranic schools shut
about 30 years of conflict have devastated health facilities hundreds of thousands
of displaced people are crowded into camps with limited access to soap and water and
no way of practising social distancing the message from al shabab only adds to the
governments concerns especially as it could oppose medical help from international
aid agencies just as it resisted most food aid during the 2010 2012 famine when more
than 250 000 people were estimated to have died 12 3 millionpopulation 6 3 millionat
risk of hunger 3 1 millionneed health assistance 2 7 millionneed water and
sanitation support 2 6 millionhomeless because of instability somalias mainstream
muslim clerics have now decided to fight back koranic schools known as madrassas
which nearly every child in somalia attends have been closed to help stop the
spread',
 'somalias mainstream muslim clerics have now decided to fight back koranic schools
known as madrassas which nearly every child in somalia attends have been closed to
help stop the spread of the virus the ministries of religious affairs health and
information have been working with muslim leaders to transform teachers and mosque
imams into what has been dubbed as an anti corona army they will stand with
loudspeakers at every crossroads every gathering place to spread the word on how to
prevent the spread of covid 19 says koshin abdi hashi deputy co ordinator for the
prevention and countering of violent extremism in the office of the prime minister
they will drive around in vehicles mounted with loudspeakers and transmit messages
from the minarets of mosques he adds mohamed ali ibrahim a university professor and
senior adviser in the ministry of religious affairs says the messages will include
religious elements',
 'messages from the minarets of mosques he adds mohamed ali ibrahim a university
professor and senior adviser in the ministry of religious affairs says the messages
will include religious elements we will talk about hand washing and social
distancing but also speak of how islam encourages cleanliness including ablutions
before prayers and how there were dangerous diseases at the time of the prophet
muhammad islamic teachings on hygiene cleanliness is half of faith wash hands before
and after eating wash hands after going to the toilet wash hands face and feet
before each of the five daily prayers bathe before main weekly prayer on fridays
wash a person after death some clerics say it is fine if this cannot be done in
current circumstances the anti corona army will monitor people making sure they wash
their hands thoroughly and regularly with soap and water and maintain a safe
distance from',
 'done in current circumstances the anti corona army will monitor people making sure
they wash their hands thoroughly and regularly with soap and water and maintain a
safe distance from one another given that so many somalis live in overcrowded camps
and other dwellings this will not always be possible hand washing facilities have
been set up in some of the camps but as mr hashi says social distancing is an
abstract issue some believe covid 19 is a divine punishment imposed on china for its
treatment of the muslim uighurs osman aden dubow deputy minister of religious
affairs says we have to use the sheikhs religious leaders and koranic teachers to
fight this disease as they are the most respected community in somalia people trust
them more than anybody else and will not only listen to what they say but follow
```

As we can see from above example, a piece of long document is splitted into list of multiple subtexts. Next, we augument our original data into a larger dataset where each row contains a piece of subtext and its corresponding label and index.

```
def data_augumentation(df, df_name):
    """
    Create a new dataframe from the original one because now one text may contain multiple su
    Text correspond to subtexts from original text, while index correspond to its index of or
    """
    text_l = []
    label_l = []
    index_l = []
    for idx,row in df.iterrows():
      for l in row['text_split']:
        text_l.append(l)
        label_l.append(row['label'])
        index_l.append(idx)
    new_df = pd.DataFrame({'text':text_l, 'label':label_l, 'index':index_l})
    print("The " + df_name +" set now has " + str(len(new_df)) + ' subtexts extracted from '
    return new_df

train_df = data_augumentation(train, df_name = 'training')
val_df = data_augumentation(val, df_name  = 'validation')
test_df = data_augumentation(test, df_name = 'testing')
```

```
The training set now has 5621 subtexts extracted from 1280 texts.
The validation set now has 1305 subtexts extracted from 320 texts.
The testing set now has 1568 subtexts extracted from 400 texts.
```

## Building BERT Model

```
!pip install bert-for-tf2
import math
import os
from tqdm import tqdm
import tensorflow as tf
from tensorflow import keras
import bert
from bert import BertModelLayer
from bert.loader import StockBertConfig, map_stock_config_to_params, load_stock_weights
from bert.tokenization.bert_tokenization import FullTokenizer
```

```
Collecting bert-for-tf2
  Downloading https://files.pythonhosted.org/packages/35/5c/6439134ecd17b33fe0396fb0b7d6
     |████████████████████████████████| 40kB 3.9MB/s
Collecting py-params>=0.9.6
  Downloading https://files.pythonhosted.org/packages/a4/bf/c1c70d5315a8677310ea10a41cf0
Collecting params-flow>=0.8.0
  Downloading https://files.pythonhosted.org/packages/a9/95/ff49f5ebd501f142a6f0aaf42bc1
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from par
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from para
Building wheels for collected packages: bert-for-tf2, py-params, params-flow
  Building wheel for bert-for-tf2 (setup.py) ... done
  Created wheel for bert-for-tf2: filename=bert_for_tf2-0.14.4-cp36-none-any.whl size=30
```

```
  Stored in directory: /root/.cache/pip/wheels/cf/3f/4d/79d7735015a5f523648df90d871ce8e8
  Building wheel for py-params (setup.py) ... done
  Created wheel for py-params: filename=py_params-0.9.7-cp36-none-any.whl size=7302 sha2
  Stored in directory: /root/.cache/pip/wheels/67/f5/19/b461849a50aefdf4bab47c4756596e82
  Building wheel for params-flow (setup.py) ... done
  Created wheel for params-flow: filename=params_flow-0.8.2-cp36-none-any.whl size=19473
  Stored in directory: /root/.cache/pip/wheels/08/c8/7f/81c86b9ff2b86e2c477e3914175be03e
Successfully built bert-for-tf2 py-params params-flow
Installing collected packages: py-params, params-flow, bert-for-tf2
Successfully installed bert-for-tf2-0.14.4 params-flow-0.8.2 py-params-0.9.7
```

First we like to load the pretrained weight of BERT and finetune it. The source of pretrained weights is called uncased_L-12_H-768_A-12. Since Because tf.train.load_checkpoint limitation requiring list permissions on the google storage bucket, we perform a tweak below to copy the pre-trained BERT weights locally.

```
%%time

bert_ckpt_dir="gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/"
bert_ckpt_file = bert_ckpt_dir + "bert_model.ckpt"
bert_config_file = bert_ckpt_dir + "bert_config.json"

bert_model_dir="2018_10_18"
bert_model_name="uncased_L-12_H-768_A-12"

!mkdir -p .model .model/$bert_model_name

for fname in ["bert_config.json", "vocab.txt", "bert_model.ckpt.meta", "bert_model.ckpt.index
  cmd = f"gsutil cp gs://bert_models/{bert_model_dir}/{bert_model_name}/{fname} .model/{bert_
  !$cmd

!ls -la .model .model/$bert_model_name

bert_ckpt_dir = os.path.join(".model/",bert_model_name)
bert_ckpt_file = os.path.join(bert_ckpt_dir, "bert_model.ckpt")
bert_config_file = os.path.join(bert_ckpt_dir, "bert_config.json")
```

```
Copying gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/bert_config.json...
/ [1 files][  313.0 B/  313.0 B]
Operation completed over 1 objects/313.0 B.
Copying gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/vocab.txt...
/ [1 files][226.1 KiB/226.1 KiB]
Operation completed over 1 objects/226.1 KiB.
Copying gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/bert_model.ckpt.meta...
/ [1 files][883.0 KiB/883.0 KiB]
Operation completed over 1 objects/883.0 KiB.
Copying gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/bert_model.ckpt.index...
/ [1 files][  8.3 KiB/  8.3 KiB]
Operation completed over 1 objects/8.3 KiB.
```

```
Copying gs://bert_models/2018_10_18/uncased_L-12_H-768_A-12/bert_model.ckpt.data-00000-o
- [1 files][420.0 MiB/420.0 MiB]
Operation completed over 1 objects/420.0 MiB.
.model:
total 12
drwxr-xr-x 3 root root 4096 Jul 12 23:20 .
drwxr-xr-x 1 root root 4096 Jul 12 23:20 ..
drwxr-xr-x 2 root root 4096 Jul 12 23:23 uncased_L-12_H-768_A-12

.model/uncased_L-12_H-768_A-12:
total 431244
drwxr-xr-x 2 root root      4096 Jul 12 23:23 .
drwxr-xr-x 3 root root      4096 Jul 12 23:20 ..
-rw-r--r-- 1 root root       313 Jul 12 23:22 bert_config.json
-rw-r--r-- 1 root root 440425712 Jul 12 23:23 bert_model.ckpt.data-00000-of-00001
-rw-r--r-- 1 root root      8528 Jul 12 23:22 bert_model.ckpt.index
-rw-r--r-- 1 root root    904243 Jul 12 23:22 bert_model.ckpt.meta
-rw-r--r-- 1 root root    231508 Jul 12 23:22 vocab.txt
CPU times: user 105 ms, sys: 63.1 ms, total: 168 ms
Wall time: 22.6 s
```

Next we preprocess our original text into input features BERT can read. The process is basically tokenizing and coverting our original text into token ids that can be read by the algorithm. The words are tokenized base on the vocabulary dictionary it pretrained on(about 30,000 words), and unknown words are breaken down into smaller words contained in the dictionary. Maximum sequence length are also specified so we can pad all sequence into the same length. Note: The final sequence length would be larger than specified since BERT tokenizer will break unknown words into multiple small known words.

```
class FakeNewsData:
    """
    Preprocessing text into BERT features.

    max_seq_len: Maximum sequence length specified
    tokenizer: BERT tokenizer
    """
    DATA_COLUMN = "text"
    LABEL_COLUMN = "label"

    def __init__(self, tokenizer, train, validation, test, max_seq_len = 150):
        self.tokenizer = tokenizer
        self.max_seq_len = max_seq_len
        ((self.train_x, self.train_y),
         (self.val_x, self.val_y),
         (self.test_x, self.test_y)) = map(self._prepare, [train, validation, test])

        ((self.train_x, self.train_x_token_types),
         (self.val_x, self.val_x_token_types),
```

```
            (self.test_x, self.test_x_token_types)) = map(self._pad,
                                                          [self.train_x, self.val_x, self.test_x

        def _prepare(self, df):
            """
            Add start and end token for each sequence, and convert the text to tokenids.
            """
            x, y = [], []
            with tqdm(total=df.shape[0], unit_scale=True) as pbar:
                for ndx, row in df.iterrows():
                    text, label = row[FakeNewsData.DATA_COLUMN], row[FakeNewsData.LABEL_COLUMN]
                    tokens = self.tokenizer.tokenize(text)
                    tokens = ["[CLS]"] + tokens + ["[SEP]"]
                    token_ids = self.tokenizer.convert_tokens_to_ids(tokens)
                    self.max_seq_len = max(self.max_seq_len, len(token_ids))
                    x.append(token_ids)
                    y.append(int(label))
                    pbar.update()
            return np.array(x), np.array(y)

        def _pad(self, ids):
            """
            Pad each sequence to the specified max sequence length with [0]
            """
            x, t = [], []
            token_type_ids = [0] * self.max_seq_len
            for input_ids in ids:
                input_ids = input_ids[:min(len(input_ids), self.max_seq_len - 2)]
                input_ids = input_ids + [0] * (self.max_seq_len - len(input_ids))
                x.append(np.array(input_ids))
                t.append(token_type_ids)
            return np.array(x), np.array(t)


    %%time
    tokenizer = FullTokenizer(vocab_file=os.path.join(bert_ckpt_dir, "vocab.txt"))
    data = FakeNewsData(tokenizer,
                        train = train_df,
                        validation = val_df,
                        test = test_df,
                        max_seq_len= 150)

         100%|██████████| 5.62k/5.62k [00:13<00:00, 418it/s]
         100%|██████████| 1.30k/1.30k [00:03<00:00, 421it/s]
         100%|██████████| 1.57k/1.57k [00:03<00:00, 418it/s]
         CPU times: user 20.8 s, sys: 110 ms, total: 20.9 s
         Wall time: 20.9 s


    def create_model(max_seq_len,lr = 1e-5):
      """
      Creates a BERT classification model.
```

```
    The model architecutre is raw input -> BERT input -> drop out layer to prevent overfitting

    max_seq_len: the maximum sequence length
    lr: learning rate of optimizer
    """



    # create the bert layer
    with tf.io.gfile.GFile(bert_config_file, "r") as reader:
        bc = StockBertConfig.from_json_string(reader.read())
        bert_params = map_stock_config_to_params(bc)
        bert = BertModelLayer.from_params(bert_params, name="bert")

    input_ids = keras.layers.Input(shape=(max_seq_len,), dtype='int32', name="input_ids")
    output = bert(input_ids)

    print("bert shape", output.shape)
    cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(output)
    # Dropout layer
    cls_out = keras.layers.Dropout(0.8)(cls_out)
    # Dense layer with probibility output
    logits = keras.layers.Dense(units=2, activation="softmax")(cls_out)

    model = keras.Model(inputs=input_ids, outputs=logits)
    model.build(input_shape=(None, max_seq_len))

    # load the pre-trained model weights
    load_stock_weights(bert, bert_ckpt_file)

    model.compile(optimizer=keras.optimizers.Adam(learning_rate = lr),
                  loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=[keras.metrics.SparseCategoricalAccuracy(name="acc")])

    model.summary()

    return model

model = create_model(max_seq_len = data.max_seq_len, lr = 1e-5)

    bert shape (None, 319, 768)
    Done loading 196 BERT weights from: .model/uncased_L-12_H-768_A-12/bert_model.ckpt into
    Unused weights from checkpoint:
            bert/embeddings/token_type_embeddings
            bert/pooler/dense/bias
            bert/pooler/dense/kernel
            cls/predictions/output_bias
            cls/predictions/transform/LayerNorm/beta
            cls/predictions/transform/LayerNorm/gamma
            cls/predictions/transform/dense/bias
            cls/predictions/transform/dense/kernel
            cls/seq_relationship/output_bias
            cls/seq_relationship/output_weights
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 319)]             0
_________________________________________________________________
bert (BertModelLayer)        (None, 319, 768)          108890112
_________________________________________________________________
lambda (Lambda)              (None, 768)               0
_________________________________________________________________
dropout (Dropout)            (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 2)                 1538
=================================================================
Total params: 108,891,650
Trainable params: 108,891,650
Non-trainable params: 0
_________________________________________________________________
```

## Model Training

```
import datetime
OUTPUT_DIR = '/bert_news'
print('***** Model output directory: {} *****'.format(OUTPUT_DIR))

log_dir = ".log/bert_news/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%s")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir)

def model_fitting(max_epoch = 5, patience = 1):


    model.fit(x=data.train_x, y=data.train_y,
              validation_data = (data.val_x,data.val_y),
              batch_size=16,
              shuffle=True,
              epochs=max_epoch,
              callbacks=[keras.callbacks.EarlyStopping(patience=patience, restore_best_weight
                         tensorboard_callback])
    return model

model = model_fitting(max_epoch = 5, patience = 1)
# Save the optimal weights for future usage
model.save_weights('bert_news.h5', overwrite=True)
```

```
***** Model output directory: /bert_news *****
Epoch 1/5
352/352 [==============================] - 543s 2s/step - loss: 0.5591 - acc: 0.7315 - \
Epoch 2/5
352/352 [==============================] - 544s 2s/step - loss: 0.4074 - acc: 0.9018 - \
Epoch 3/5
```

```
352/352 [==============================] - 539s 2s/step - loss: 0.3818 - acc: 0.9295 - v
Epoch 4/5
352/352 [==============================] - 541s 2s/step - loss: 0.3611 - acc: 0.9516 - v
```

```
# Download the model checkpoint
# OPTIONAL STEP - if we want to use them for prediction somewhere
from google.colab import files
files.download('bert_news.h5')
```

## Model Evaluation

After model is done training, we evaluate on training set, validation set, and test set. The metric used is accuracy.

```
%%time
# model = create_model(max_seq_len = data.max_seq_len, lr = 1e-5)
model.load_weights("bert_news.h5")

_, train_acc = model.evaluate(data.train_x, data.train_y)
_, val_acc = model.evaluate(data.val_x, data.val_y)
_, test_acc = model.evaluate(data.test_x, data.test_y)

print("train acc: ", train_acc)
print("validation acc: ", val_acc)
print("test acc: ", test_acc)
```

```
176/176 [==============================] - 147s 837ms/step - loss: 0.3516 - acc: 0.9596
41/41 [==============================] - 34s 835ms/step - loss: 0.3578 - acc: 0.9540
49/49 [==============================] - 41s 827ms/step - loss: 0.3658 - acc: 0.9439
train acc:  0.9596157073974609
validation acc:  0.954023003578186
test acc:  0.9438775777816772
CPU times: user 46.6 s, sys: 25.2 s, total: 1min 11s
Wall time: 3min 46s
```

**Evaluation Observations**:

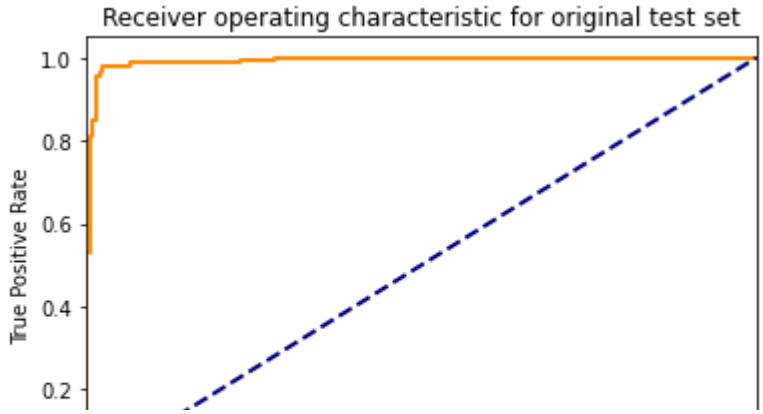
We can see that we have about ~ aapprox 96.0 % accuracy for training set and about 95.40 % accuracy for our validation set. This seems to be good, however, there are two more things we should do.

1. The prediction is on the **augumented** test set, but we care more about the prediction on **original** text. Therefore, for each original text, we should average the probabily of each

subtexts and obtain final prediction for that piece of news.

2. Other metrics such as AUC to get a more thorough evaluation of our model.

```
from sklearn.metrics import roc_curve, auc, accuracy_score, precision_score, recall_score, f1
# model = create_model(max_seq_len = data.max_seq_len, lr = 1e-5)
model.load_weights("bert_news.h5")

# predict on test set
predictions = model.predict(data.test_x)
predictions = predictions[:,1]
test_df['pred'] = predictions
# average the prediction to become final prediction of original test set
test['avg_pred'] = test_df.groupby(['index'])['pred'].mean()

# plot ROC curve
fpr, tpr, _ = roc_curve(test['label'], test['avg_pred'])
roc_auc = auc(fpr, tpr)

fig = plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (auc = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for original test set')
plt.legend(loc="lower right")
plt.show()

acc = accuracy_score(test['label'], test['avg_pred'] > 0.5)
precision = precision_score(test['label'], test['avg_pred'] > 0.5)
recall = recall_score(test['label'], test['avg_pred'] > 0.5)
f1 = f1_score(test['label'], test['avg_pred']  > 0.5)

print('Orignal test accuracy is ', acc)
print('Orignal test auc is ', roc_auc)
print('Orignal test precision is ', precision)
print('Orignal test recall is ', recall)
print('Orignal test f1 score is ', f1)
```




# VI - Predicting on a New Text- "NEWS"

The goal of building model is to predict new coming instances. Below is a function to load the model checkpoint and predict result of input document. Please run everything before Model Training section in the notebook.

HOW TO PREDICT ?-

I - via SAVED MODEL (If you dont want to train model again)

- If you wish to predict from the already saved model,
- Please keep the model ('bert.h5') in mounted google drive
- Replace the path of the mounted model in SAVED_MODEL variable value in the below code snippet.

II - New Trained Model (If you ran all code blocks)

- If you have again run all the code blocks.
- Please simply replace the SAVED_MODEL value with 'bert.h5'
- Because after training the model will be saved with that name in current path.

```
# As per the steps above :)

SAVED_MODEL_PATH  = "bert_news.h5"


# create model and load previous weights
model = create_model(max_seq_len = data.max_seq_len)
model.load_weights(SAVED_MODEL_PATH)
```

```
bert shape (None, 319, 768)
Done loading 196 BERT weights from: .model/uncased_L-12_H-768_A-12/bert_model.ckpt into
Unused weights from checkpoint:
        bert/embeddings/token_type_embeddings
        bert/pooler/dense/bias
        bert/pooler/dense/kernel
```

```
        cls/predictions/output_bias
        cls/predictions/transform/LayerNorm/beta
        cls/predictions/transform/LayerNorm/gamma
        cls/predictions/transform/dense/bias
        cls/predictions/transform/dense/kernel
        cls/seq_relationship/output_bias
        cls/seq_relationship/output_weights
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 319)]             0
_________________________________________________________________
bert (BertModelLayer)        (None, 319, 768)          108890112
_________________________________________________________________
lambda_5 (Lambda)            (None, 768)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 768)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 1538
=================================================================
Total params: 108,891,650
Trainable params: 108,891,650
Non-trainable params: 0
_________________________________________________________________
```

```
def predict_new(doc, model):
    """
    Predict new document using the trained model.

    doc: input document in format of a string
    """

    # clean the text
    doc = clean_txt(doc)
    # split the string text into list of subtexts
    doc = get_split(doc)
    # tokenize the subtexts as well as padding
    tokenizer = FullTokenizer(vocab_file=os.path.join(bert_ckpt_dir, "vocab.txt"))
    pred_tokens = map(tokenizer.tokenize, doc)
    pred_tokens = map(lambda tok: ["[CLS]"] + tok + ["[SEP]"], pred_tokens)
    pred_token_ids = list(map(tokenizer.convert_tokens_to_ids, pred_tokens))
    pred_token_ids = map(lambda tids: tids +[0]*(data.max_seq_len-len(tids)),pred_token_ids)
    pred_token_ids = np.array(list(pred_token_ids))

    # create model and load previous weights
    # model = create_model(max_seq_len = data.max_seq_len)
    # model.load_weights()

    # predict the subtexts and average the prediction
    predictions = model.predict(pred_token_ids)
```

```
    predictions = predictions[:,1]
    avg_pred = predictions.mean()
    if avg_pred > 0.5:
      doc_label = 'fake'
    else:
      doc_label = 'Real'

    return doc_label, avg_pred



# Run an example text from original test set
fake_test = load_convert_data("https://storage.googleapis.com/public-resources/dataset/fake_t
doc = fake_test['text'][7]
print('----------------------NEWS ----------------------')
print(doc)
doc_label, avg_pred = predict_new(doc, model)
print()
print('---------------- PREDICTION RESULTS --------------')
print('The predicted probability of news being FAKE is ', avg_pred)
print('CLASSIFICATION : The predicted label of news is ', doc_label.upper())
```

```
----------------------NEWS ----------------------
Elena Malysheva, the famous Russian presenter and presenter, has shed some nutrients tha
World - Europe
During one of her televised medical programs, Malysheva said: "Food products that contai
The doctor indicated that a person can get this important component from different foods
According to Malysheva, the elderly with diabetes are among the most at risk of corona,
The doctor advised those suffering from cardiovascular diseases and cancer patients to a
She also stressed the need to drink more water, especially for the elderly, and she expl

---------------- PREDICTION RESULTS --------------
The predicted probability of news being FAKE is  0.9999819
CLASSIFICATION : The predicted label of news is  FAKE
```

Colab paid products - Cancel contracts here

✓ 1s completed at 11:59 PM