In [ ]:

```
!pip install transformers==2.4.0
```

In [3]:

```
# importing necessary libraries
from typing import List, Tuple
import random
import html

import pandas as pd
import numpy as np
from sklearn.model_selection import GroupKFold, KFold
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import tensorflow as tf
import tensorflow.keras.backend as K
import os
from scipy.stats import spearmanr
from scipy.optimize import minimize
from math import floor, ceil
from transformers import *
from tensorflow.keras.layers import Flatten, Dense, Dropout, GlobalAveragePooling1D
from tensorflow.keras.models import Model
```

In [4]:

```
# fixing random seeds
seed = 13
random.seed(seed)
os.environ['PYTHONHASHSEED'] = str(seed)
np.random.seed(seed)
tf.random.set_seed(seed)
```

In [4]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk
8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3ao
ob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [5]:

```
# reading the data into dataframe using pandas
train = pd.read_csv('drive/My Drive/case_study_2/train.csv')
test = pd.read_csv('drive/My Drive/case_study_2/test.csv')
submission = pd.read_csv('drive/My Drive/case_study_2/sample_submission.csv')
```

In [6]:

```
# Selecting data for training and testing
y = train[train.columns[11:]] # storing the target values in y
X = train[['question_title', 'question_body', 'answer']]
X_test = test[['question_title', 'question_body', 'answer']]
```

In [7]:

```
# Cleaning the data
X.question_body = X.question_body.apply(html.unescape)
X.question_title = X.question_title.apply(html.unescape)
X.answer = X.answer.apply(html.unescape)

X_test.question_body = X_test.question_body.apply(html.unescape)
X_test.question_title = X_test.question_title.apply(html.unescape)
X_test.answer = X_test.answer.apply(html.unescape)
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py:5303: SettingWithCopyWarning:
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/indexi
ng.html#returning-a-view-versus-a-copy
  self[name] = value
```

In [8]:

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
MAX_SEQUENCE_LENGTH = 512

# this function trims the tokens with length > 512 to match with the bert input.
'''
In the function below, if the input sentence has the number of tokens > 512, the
sentence is trimmed down to 512. To trim the number of tokens, 256 tokens from
the start and 256 tokens from the end are kept and the remaining tokens are dropped.
Ex. suppose an answer has 700 tokens, to trim this down to 512, 256 tokens from the
beginning are taken and 256 tokens from the end are taken and concatenated to make
512 tokens. The remaining [700-(256+256) = 288] tokens that are in the middle of the
answer are dropped. The logic makes sense because in large texts, the beginning part
usually describes what the text is all about and the end part describes the conclusion
of the text. This is also closely related to the target features that we need to predict.
'''
def _trim_input(question_tokens, answer_tokens, max_sequence_length=512, q_max_len=254, a_max_len=254):
    q_len = len(question_tokens)
    a_len = len(answer_tokens)
    if q_len + a_len + 3 > max_sequence_length:
        if a_max_len <= a_len and q_max_len <= q_len:
            q_new_len_head = q_max_len//2
            question_tokens = question_tokens[:q_new_len_head] + question_tokens[-q_new_len_head:]
            a_new_len_head = a_max_len//2
            answer_tokens = answer_tokens[:a_new_len_head] + answer_tokens[-a_new_len_head:]
        elif q_len <= a_len and q_len < q_max_len:
            a_max_len = a_max_len + (q_max_len - q_len - 1)
            a_new_len_head = a_max_len//2
            answer_tokens = answer_tokens[:a_new_len_head] + answer_tokens[-a_new_len_head:]
        elif a_len < q_len:
            q_max_len = q_max_len + (a_max_len - a_len - 1)
            q_new_len_head = q_max_len//2
            question_tokens = question_tokens[:q_new_len_head] + question_tokens[-q_new_len_head:]

    return question_tokens, answer_tokens
```

In [9]:

```
# function for tokenizing the input data for bert
def _convert_to_transformer_inputs(title, question, answer, tokenizer, question_only=False):
    question = f"{title} [SEP] {question}"
    question_tokens = tokenizer.tokenize(question)
    if question_only:
        answer_tokens = []
    else:
        answer_tokens = tokenizer.tokenize(answer)
    question_tokens, answer_tokens = _trim_input(question_tokens, answer_tokens)
    ids = tokenizer.convert_tokens_to_ids(["[CLS]"] + question_tokens + ["[SEP]"] + answer_tokens + ["[SEP]"])
    padded_ids = ids + [tokenizer.pad_token_id] * (MAX_SEQUENCE_LENGTH - len(ids))
    token_type_ids = [0] * (1 + len(question_tokens) + 1) + [1] * (len(answer_tokens) + 1) + [0] * (MAX_SEQUENC
E_LENGTH - len(ids))
    attention_mask = [1] * len(ids) + [0] * (MAX_SEQUENCE_LENGTH - len(ids))
    return padded_ids, token_type_ids, attention_mask
```

In [10]:

```
# function for creating the input_ids, masks and segments for the bert input
def compute_input_arrays(df, question_only=False):
    input_ids, input_token_type_ids, input_attention_masks = [], [], []
    for title, body, answer in zip(df["question_title"].values, df["question_body"].values, df["answer"].values
):
        ids, type_ids, mask = _convert_to_transformer_inputs(title, body, answer, tokenizer, question_only=ques
tion_only)
        input_ids.append(ids)
        input_token_type_ids.append(type_ids)
        input_attention_masks.append(mask)
    return (
        np.asarray(input_ids, dtype=np.int32),
        np.asarray(input_attention_masks, dtype=np.int32),
        np.asarray(input_token_type_ids, dtype=np.int32)
    )


def compute_output_arrays(df):
    return np.asarray(df[output_categories])
```

In [6]:

```
# Creating the model
K.clear_session()
max_seq_length = 512

input_tokens = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_tokens")
input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_mask")
input_segment = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_segment")

#bert layer
bert_config = BertConfig.from_pretrained('bert-base-uncased', output_hidden_states=True)
bert_model = TFBertModel.from_pretrained('bert-base-uncased', config=bert_config)

sequence_output, pooler_output, hidden_states = bert_model([input_tokens,input_mask, input_segment])

# Last 4 hidden layers of bert
h12 = tf.reshape(hidden_states[-1][:,0],(-1,1,768))
h11 = tf.reshape(hidden_states[-2][:,0],(-1,1,768))
h10 = tf.reshape(hidden_states[-3][:,0],(-1,1,768))
h09 = tf.reshape(hidden_states[-4][:,0],(-1,1,768))
concat_hidden = tf.keras.layers.Concatenate(axis=2)([h12, h11, h10, h09])

x = GlobalAveragePooling1D()(concat_hidden)

output = Dense(9, activation='sigmoid')(x)

model_a = Model(inputs=[input_tokens, input_mask, input_segment], outputs=output)
```

In [7]:

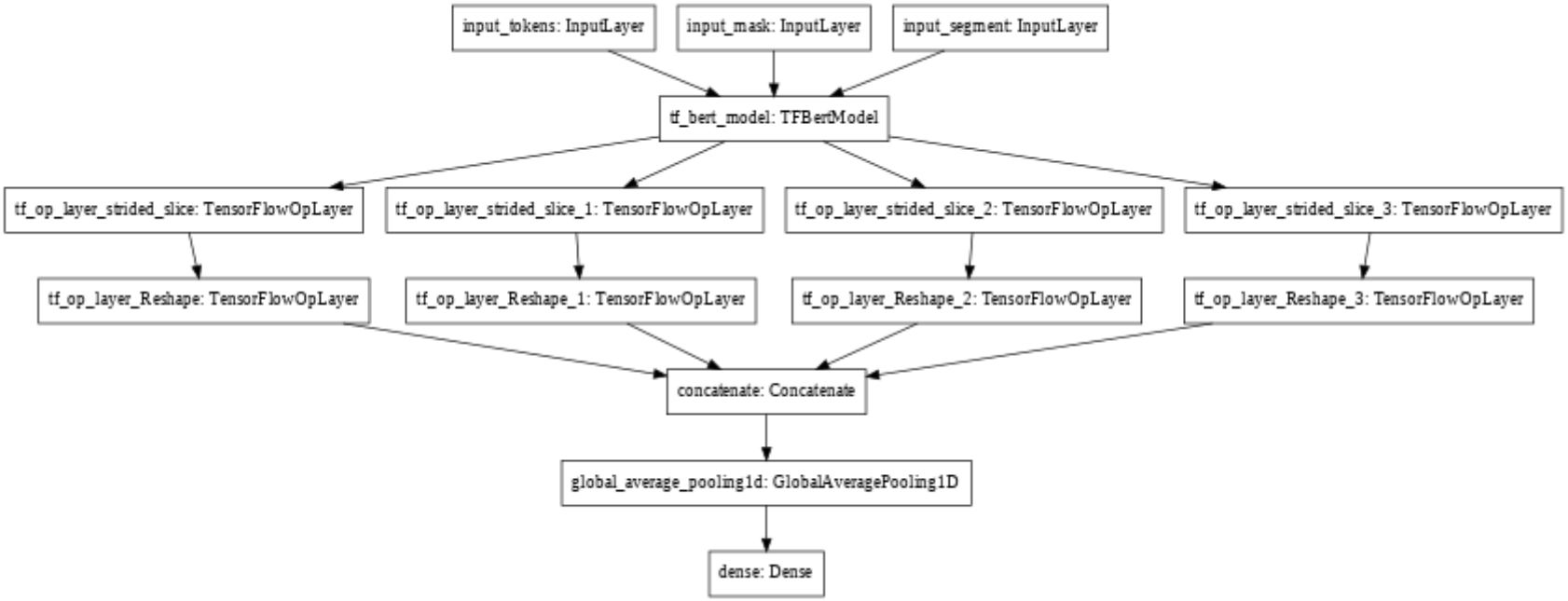
```
model_a.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_tokens (InputLayer)       [(None, 512)]        0
__________________________________________________________________________________________________
input_mask (InputLayer)         [(None, 512)]        0
__________________________________________________________________________________________________
input_segment (InputLayer)      [(None, 512)]        0
__________________________________________________________________________________________________
tf_bert_model (TFBertModel)     ((None, 512, 768), ( 109482240   input_tokens[0][0]
                                                                 input_mask[0][0]
                                                                 input_segment[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice (Tens [(None, 768)]        0           tf_bert_model[0][14]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_1 (Te [(None, 768)]        0           tf_bert_model[0][13]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_2 (Te [(None, 768)]        0           tf_bert_model[0][12]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_3 (Te [(None, 768)]        0           tf_bert_model[0][11]
__________________________________________________________________________________________________
tf_op_layer_Reshape (TensorFlow [(None, 1, 768)]     0           tf_op_layer_strided_slice[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_1 (TensorFl [(None, 1, 768)]     0           tf_op_layer_strided_slice_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_2 (TensorFl [(None, 1, 768)]     0           tf_op_layer_strided_slice_2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Reshape_3 (TensorFl [(None, 1, 768)]     0           tf_op_layer_strided_slice_3[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 1, 3072)      0           tf_op_layer_Reshape[0][0]
                                                                 tf_op_layer_Reshape_1[0][0]
                                                                 tf_op_layer_Reshape_2[0][0]
                                                                 tf_op_layer_Reshape_3[0][0]
__________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 3072)         0           concatenate[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 9)            27657       global_average_pooling1d[0][0]
==================================================================================================
Total params: 109,509,897
Trainable params: 109,509,897
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [8]:

```
tf.keras.utils.plot_model(
    model_a, to_file='model.png',
    show_shapes=False,
    show_layer_names=True,
    rankdir='TB',
    expand_nested=False, dpi=48
)
```

Out[8]:



In [14]:

```
# train_data
tokens, masks, segments = compute_input_arrays(X)
train_data = {'input_tokens': tokens,
              'input_mask': masks,
              'input_segment': segments}

# test data
tokens, masks, segments = compute_input_arrays(X_test)
test_data = {'input_tokens': tokens,
             'input_mask': masks,
             'input_segment': segments}
```

In [15]:

```
# Function to calculate the Spearman's rank correlation coefficient 'rhos' of actual and predicted data.
from scipy.stats import spearmanr
def compute_spearmanr_ignore_nan(trues, preds):
    rhos = []
    for tcol, pcol in zip(np.transpose(trues), np.transpose(preds)):
        rhos.append(spearmanr(tcol, pcol).correlation)
    return np.nanmean(rhos)
```

In [16]:

```
# Making the 'rhos' metric to tensorflow graph compatible.
def rhos(y, y_pred):
  return tf.py_function(compute_spearmanr_ignore_nan, (y, y_pred), tf.double)
metrics = [rhos]
```

In [18]:

```
# Compiling and training the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00002)
model_a.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=metrics)
for i in range(3):
  model_a.fit(train_data, y.values[:,21:], epochs=1, batch_size=4, validation_split=0.1)
  model_a.save_weights("drive/My Drive/model_a.h5")
```

```
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.


/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2534: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[:, None]
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2535: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[None, :]
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in greater
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in less
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: RuntimeWarning: in
valid value encountered in less_equal
  cond2 = cond0 & (x <= _a)

1368/1368 [==============================] - 1358s 993ms/step - loss: 0.3779 - rhos: 0.2838 - val_lo
ss: 0.3686 - val_rhos: 0.3447
1368/1368 [==============================] - 1361s 995ms/step - loss: 0.3521 - rhos: 0.3918 - val_lo
ss: 0.3670 - val_rhos: 0.3445
 352/1368 [======>.......................] - ETA: 16:14 - loss: 0.3470 - rhos: 0.4447
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-18-5c0c7aec2e6d> in <module>()
      2 model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=metrics)
      3 for i in range(3):
----> 4   model.fit(train_data, y.values[:,21:], epochs=1, batch_size=4, validation_split=0.1)
      5   model.save_weights("drive/My Drive/bert_case_study.h5")

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py in _method_wrapper
(self, *args, **kwargs)
     64   def _method_wrapper(self, *args, **kwargs):
     65     if not self._in_multi_worker_mode():  # pylint: disable=protected-access
---> 66       return method(self, *args, **kwargs)
     67
     68     # Running inside `run_distribute_coordinator` already.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py in fit(self, x, y,
batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sa
mple_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_fre
q, max_queue_size, workers, use_multiprocessing)
    846                 batch_size=batch_size):
    847               callbacks.on_train_batch_begin(step)
--> 848               tmp_logs = train_function(iterator)
    849               # Catch OutOfRangeError for Datasets of unknown size.
    850               # This blocks until the batch has finished executing.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py in __call__(self, *ar
gs, **kwds)
    578         xla_context.Exit()
    579     else:
--> 580       result = self._call(*args, **kwds)
    581
    582     if tracing_count == self._get_tracing_count():

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py in _call(self, *args,
**kwds)
    609       # In this case we have created variables on the first call, so we run the
    610       # defunned version which is guaranteed to never create variables.
--> 611       return self._stateless_fn(*args, **kwds)  # pylint: disable=not-callable
    612     elif self._stateful_fn is not None:
    613       # Release the lock early so that multiple threads can perform the call

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in __call__(self, *args,
```

```
 **kwargs)
   2418     with self._lock:
   2419       graph_function, args, kwargs = self._maybe_define_function(args, kwargs)
-> 2420     return graph_function._filtered_call(args, kwargs)  # pylint: disable=protected-access
   2421
   2422   @property

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in _filtered_call(self, a
rgs, kwargs)
   1663           if isinstance(t, (ops.Tensor,
   1664                             resource_variable_ops.BaseResourceVariable))),
-> 1665         self.captured_inputs)
   1666
   1667   def _call_flat(self, args, captured_inputs, cancellation_manager=None):

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in _call_flat(self, args,
captured_inputs, cancellation_manager)
   1744       # No tape is watching; skip to running the function.
   1745       return self._build_call_outputs(self._inference_function.call(
-> 1746           ctx, args, cancellation_manager=cancellation_manager))
   1747     forward_backward = self._select_forward_and_backward_functions(
   1748         args,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in call(self, ctx, args,
 cancellation_manager)
    596               inputs=args,
    597               attrs=attrs,
--> 598               ctx=ctx)
    599         else:
    600           outputs = execute.execute_with_cancellation(

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name,
 num_outputs, inputs, attrs, ctx, name)
     58     ctx.ensure_initialized()
     59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:
     62     if name is not None:

KeyboardInterrupt:
```

In [19]:

```
model_a.fit(train_data, y.values[:,21:], epochs=1, batch_size=4)
```

```
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0',
'tf_bert_model/bert/pooler/dense/bias:0'] when minimizing the loss.

/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2534: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[:, None]
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2535: RuntimeWarning: invalid valu
e encountered in true_divide
  c /= stddev[None, :]
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in greater
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: inv
alid value encountered in less
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: RuntimeWarning: in
valid value encountered in less_equal
  cond2 = cond0 & (x <= _a)

1520/1520 [==============================] - 1464s 963ms/step - loss: 0.3320 - rhos: 0.4967
```

Out[19]:

```
<tensorflow.python.keras.callbacks.History at 0x7f623418b358>
```

In [23]:

```
# Predicting the train and test data labels
pred_a_test = model_a.predict(test_data)
pred_a_train = model_a.predict(train_data)

# saving the predicted labels as dataframes
df = pd.DataFrame(pred_a_train, columns=y.columns[21:])
df.to_csv('pred_a_train.csv', index=False)

df = pd.DataFrame(pred_a_test, columns=y.columns[21:])
df.to_csv('pred_a_test.csv', index=False)
```

In [ ]: