In [1]:

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

```
Requirement already satisfied: transformers==2.4.0 in /usr/local/lib/python3.6/dist-packages (2.4.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from tra
nsformers==2.4.0) (2019.12.20)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.6/dist-packages (from transfo
rmers==2.4.0) (0.1.91)
Requirement already satisfied: tokenizers==0.0.11 in /usr/local/lib/python3.6/dist-packages (from tr
ansformers==2.4.0) (0.0.11)
Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from transformers==
2.4.0) (1.14.22)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (2.23.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (3.0.12)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (from transforme
rs==2.4.0) (4.41.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers==
2.4.0) (1.18.5)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.6/dist-packages (from transforme
rs==2.4.0) (0.0.43)
Requirement already satisfied: botocore<1.18.0,>=1.17.22 in /usr/local/lib/python3.6/dist-packages
(from boto3->transformers==2.4.0) (1.17.22)
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in /usr/local/lib/python3.6/dist-packages (f
rom boto3->transformers==2.4.0) (0.3.3)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.6/dist-packages (fro
m boto3->transformers==2.4.0) (0.10.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from req
uests->transformers==2.4.0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from re
quests->transformers==2.4.0) (2020.6.20)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests
->transformers==2.4.0) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/d
ist-packages (from requests->transformers==2.4.0) (1.24.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from sacremoses->trans
formers==2.4.0) (1.15.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from sacremoses->tr
ansformers==2.4.0) (0.16.0)
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from sacremoses->tra
nsformers==2.4.0) (7.1.2)
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/dist-packages (from
botocore<1.18.0,>=1.17.22->boto3->transformers==2.4.0) (0.15.2)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.6/dist-packages
(from botocore<1.18.0,>=1.17.22->boto3->transformers==2.4.0) (2.8.1)
```

In [2]:

```
# 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 [3]:

```
# 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')
```

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

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]:

```
# this function trims the tokens with length > 512 to match with the bert input.
def _trim_input(tokens, max_sequence_length=512):
    length = len(tokens)
    if length > max_sequence_length:
      tokens = tokens[:max_sequence_length-1]
    return tokens
```

In [9]:

```
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
MAX_SEQUENCE_LENGTH = 512

# function for tokenizing the input data for bert
def _convert_to_transformer_inputs(title, question, answer, tokenizer):
    question = f"{title} [SEP] {question}"
    question_tokens = tokenizer.tokenize(question)
    answer_tokens = tokenizer.tokenize(answer)
    question_tokens = _trim_input(question_tokens)
    answer_tokens = _trim_input(answer_tokens)
    ids_q = tokenizer.convert_tokens_to_ids(["[CLS]"] + question_tokens)
    ids_a = tokenizer.convert_tokens_to_ids(["[CLS]"] + answer_tokens)
    padded_ids_q = (ids_q + [tokenizer.pad_token_id] * (MAX_SEQUENCE_LENGTH - len(ids_q)))[:MAX_SEQUENCE_LENGTH
]
    padded_ids_a = (ids_a + [tokenizer.pad_token_id] * (MAX_SEQUENCE_LENGTH - len(ids_a)))[:MAX_SEQUENCE_LENGTH
]
    token_type_ids_q = ([0] * MAX_SEQUENCE_LENGTH)[:MAX_SEQUENCE_LENGTH]
    token_type_ids_a = ([0] * MAX_SEQUENCE_LENGTH)[:MAX_SEQUENCE_LENGTH]
    attention_mask_q = ([1] * len(ids_q) + [0] * (MAX_SEQUENCE_LENGTH - len(ids_q)))[:MAX_SEQUENCE_LENGTH]
    attention_mask_a = ([1] * len(ids_a) + [0] * (MAX_SEQUENCE_LENGTH - len(ids_a)))[:MAX_SEQUENCE_LENGTH]

    return (padded_ids_q, padded_ids_a, token_type_ids_q, token_type_ids_a, attention_mask_q, attention_mask_a)
```

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_q, input_token_type_ids_q, input_attention_masks_q = [], [], []
    input_ids_a, input_token_type_ids_a, input_attention_masks_a = [], [], []
    i=0
    for title, body, answer in zip(df["question_title"].values, df["question_body"].values, df["answer"].values
):
        values = _convert_to_transformer_inputs(title, body, answer, tokenizer)
        padded_ids_q, padded_ids_a, token_type_ids_q, token_type_ids_a, attention_mask_q, attention_mask_a = va
lues
        input_ids_q.append(padded_ids_q)
        input_ids_a.append(padded_ids_a)
        input_token_type_ids_q.append(token_type_ids_q)
        input_token_type_ids_a.append(token_type_ids_a)
        input_attention_masks_q.append(attention_mask_q)
        input_attention_masks_a.append(attention_mask_a)
        i+=1

    return (np.asarray(input_ids_q, dtype=np.int32),
            np.asarray(input_ids_a, dtype=np.int32),
            np.asarray(input_token_type_ids_q, dtype=np.int32),
            np.asarray(input_token_type_ids_a, dtype=np.int32),
            np.asarray(input_attention_masks_q, dtype=np.int32),
            np.asarray(input_attention_masks_a, dtype=np.int32))

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

In [11]:

```
# 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
roberta_config = RobertaConfig.from_pretrained('roberta-base', output_hidden_states=True)
roberta_model = TFRobertaModel.from_pretrained('roberta-base', config=roberta_config)

sequence_output, pooler_output, hidden_states = roberta_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)

x = Dropout(0.2)(x)

output = Dense(21, activation='sigmoid')(x)

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

In [12]:

```
model_q.summary()
```

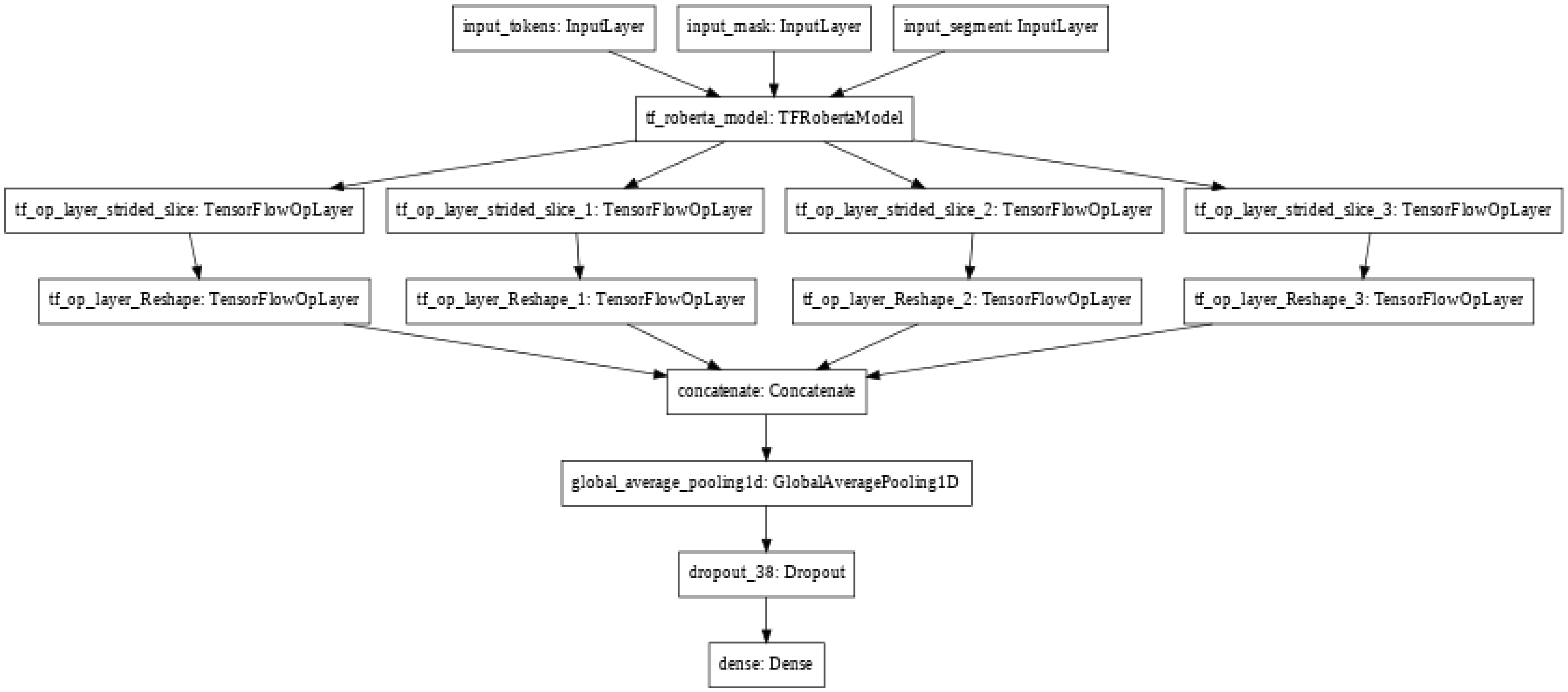
```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_tokens (InputLayer)       [(None, 512)]        0
__________________________________________________________________________________________________
input_mask (InputLayer)         [(None, 512)]        0
__________________________________________________________________________________________________
input_segment (InputLayer)      [(None, 512)]        0
__________________________________________________________________________________________________
tf_roberta_model (TFRobertaMode ((None, 512, 768), ( 124645632   input_tokens[0][0]
                                                                 input_mask[0][0]
                                                                 input_segment[0][0]
__________________________________________________________________________________________________
tf_op_layer_strided_slice (Tens [(None, 768)]        0           tf_roberta_model[0][14]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_1 (Te [(None, 768)]        0           tf_roberta_model[0][13]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_2 (Te [(None, 768)]        0           tf_roberta_model[0][12]
__________________________________________________________________________________________________
tf_op_layer_strided_slice_3 (Te [(None, 768)]        0           tf_roberta_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]
__________________________________________________________________________________________________
dropout_38 (Dropout)            (None, 3072)         0           global_average_pooling1d[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 21)           64533       dropout_38[0][0]
==================================================================================================
Total params: 124,710,165
Trainable params: 124,710,165
```

```
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [13]:

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

Out[13]:


input_tokens: InputLayer
input_mask: InputLayer
input_segment: InputLayer
tf_roberta_model: TFRobertaModel
tf_op_layer_strided_slice: TensorFlowOpLayer
tf_op_layer_strided_slice_1: TensorFlowOpLayer
tf_op_layer_strided_slice_2: TensorFlowOpLayer
tf_op_layer_strided_slice_3: TensorFlowOpLayer
tf_op_layer_Reshape: TensorFlowOpLayer
tf_op_layer_Reshape_1: TensorFlowOpLayer
tf_op_layer_Reshape_2: TensorFlowOpLayer
tf_op_layer_Reshape_3: TensorFlowOpLayer
concatenate: Concatenate
global_average_pooling1d: GlobalAveragePooling1D
dropout_38: Dropout
dense: Dense

In [14]:

```
# Test data
tokens_q, tokens_a, segments_q, segments_a, masks_q, masks_a = compute_input_arrays(X_test)
test_data_q = {'input_tokens': tokens_q,
             'input_mask': masks_q,
             'input_segment': segments_q}
```

In [15]:

```
# Train data
tokens_q, tokens_a, segments_q, segments_a, masks_q, masks_a = compute_input_arrays(X)
def generate_data(tr, cv):
  train_data_q = {'input_tokens': tokens_q[tr],
                'input_mask': masks_q[tr],
                'input_segment': segments_q[tr]}

  cv_data_q = {'input_tokens': tokens_q[cv],
             'input_mask': masks_q[cv],
             'input_segment': segments_q[cv]}

  return train_data_q, cv_data_q, y.values[tr,:21], y.values[cv,:21]
```

In [16]:

```
# 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 [17]:

```
# 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]:

```
from sklearn.model_selection import KFold
# Compiling and training the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00002)
model_q.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=metrics)
kf = KFold(n_splits=5, random_state=42)
for tr, cv in kf.split(np.arange(train.shape[0])):
  tr_data, cv_data, y_tr, y_cv = generate_data(tr, cv)
  model_q.fit(tr_data, y_tr, epochs=1, batch_size=4, validation_data=(cv_data, y_cv))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:296: FutureWarning: Setting
a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should le
ave random_state to its default (None), or set shuffle=True.
  FutureWarning
```

```
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/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)
```

```
1216/1216 [==============================] - 1342s 1s/step - loss: 0.4150 - rhos: 0.2774 - val_loss:
0.3813 - val_rhos: 0.4324
1216/1216 [==============================] - 1338s 1s/step - loss: 0.3815 - rhos: 0.3985 - val_loss:
0.3593 - val_rhos: 0.4781
1216/1216 [==============================] - 1341s 1s/step - loss: 0.3688 - rhos: 0.4400 - val_loss:
0.3560 - val_rhos: 0.5170
1216/1216 [==============================] - 1340s 1s/step - loss: 0.3615 - rhos: 0.4664 - val_loss:
0.3429 - val_rhos: 0.5349
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kern
el:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimizing the loss.
1216/1216 [==============================] - 1329s 1s/step - loss: 0.3542 - rhos: 0.4953 - val_loss:
0.3400 - val_rhos: 0.5674
```

In [26]:

```
model_q.save_weights("drive/My Drive/roberta_model_q.h5")
```

In [27]:

```
# complete train data
tokens_q, tokens_a, segments_q, segments_a, masks_q, masks_a = compute_input_arrays(X)
train_data_q = {'input_tokens': tokens_q,
                'input_mask': masks_q,
                'input_segment': segments_q}
```

In [28]:

```
# Predicting the train and test data labels
pred_q_test = model_q.predict(test_data_q)
pred_q_train = model_q.predict(train_data_q)

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

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

In [ ]:

```
```