In [ ]:

```
!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: filelock in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (3.0.12)
Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from transformers==
2.4.0) (1.14.22)
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: sacremoses in /usr/local/lib/python3.6/dist-packages (from transforme
rs==2.4.0) (0.0.43)
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: sentencepiece in /usr/local/lib/python3.6/dist-packages (from transfo
rmers==2.4.0) (0.1.91)
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: requests in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (2.23.0)
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: 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: joblib in /usr/local/lib/python3.6/dist-packages (from sacremoses->tr
ansformers==2.4.0) (0.16.0)
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: click in /usr/local/lib/python3.6/dist-packages (from sacremoses->tra
nsformers==2.4.0) (7.1.2)
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: 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: 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: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests
->transformers==2.4.0) (2.10)
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)
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)
```

In [ ]:

```
# 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 [ ]:

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

In [ ]:

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

In [ ]:

```
# # 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 [ ]:

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

In [ ]:

```
# 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 [ ]:

```
# 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 [ ]:

```
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
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 [ ]:

```
# 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 [ ]:

```
# 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 [ ]:

```
# 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
albert_config = AlbertConfig.from_pretrained('albert-base-v2', output_hidden_states=True)
albert_model = TFAlbertModel.from_pretrained('albert-base-v2', config=albert_config)

sequence_output, pooler_output, hidden_states = albert_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(9, activation='sigmoid')(x)

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

In [ ]:

```
model_a.summary()
```

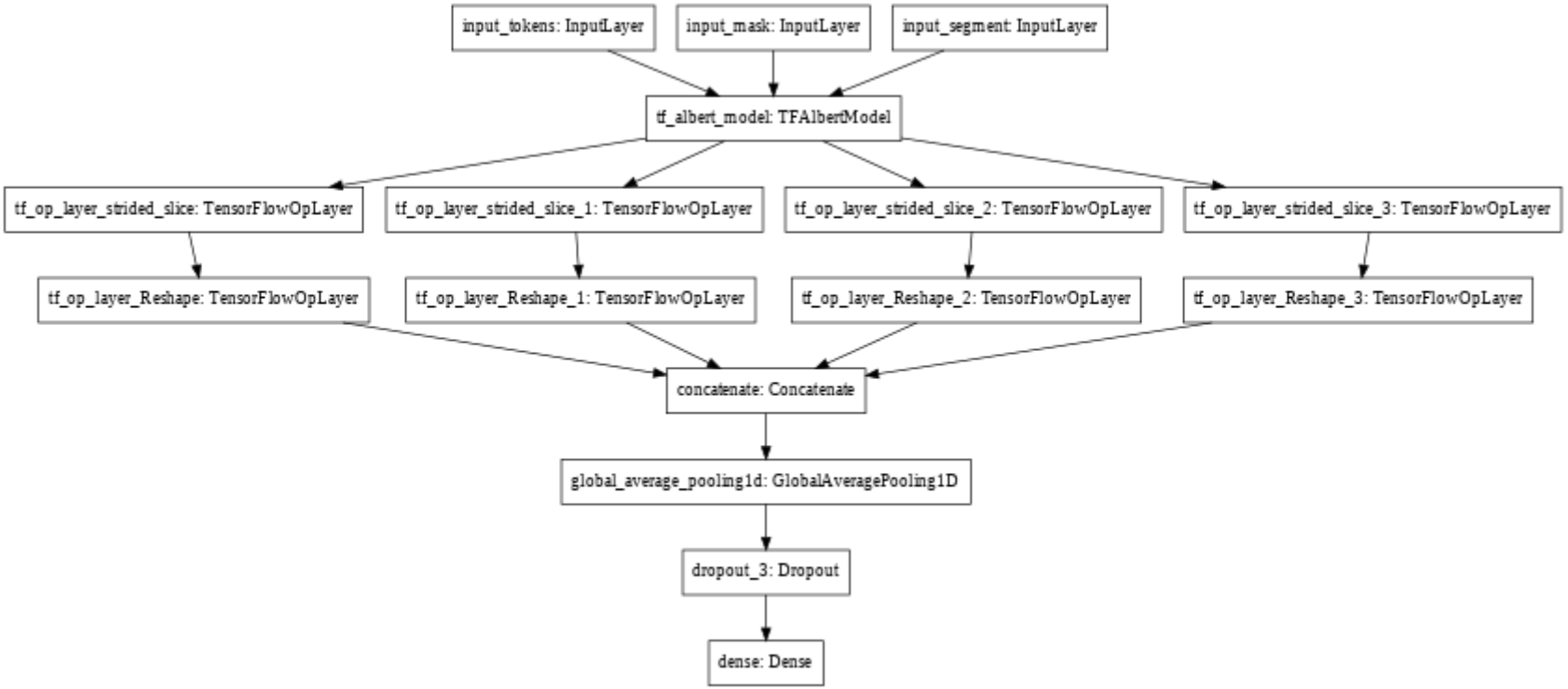
```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_tokens (InputLayer)    [(None, 512)]             0
_________________________________________________________________________________________________
input_mask (InputLayer)      [(None, 512)]             0
_________________________________________________________________________________________________
input_segment (InputLayer)   [(None, 512)]             0
_________________________________________________________________________________________________
tf_albert_model (TFAlbertModel) ((None, 512, 768), ( 11683584    input_tokens[0][0]
                                                                 input_mask[0][0]
                                                                 input_segment[0][0]
_________________________________________________________________________________________________
tf_op_layer_strided_slice (Tens [(None, 768)]         0           tf_albert_model[0][14]
_________________________________________________________________________________________________
tf_op_layer_strided_slice_1 (Te [(None, 768)]         0           tf_albert_model[0][13]
_________________________________________________________________________________________________
tf_op_layer_strided_slice_2 (Te [(None, 768)]         0           tf_albert_model[0][12]
_________________________________________________________________________________________________
tf_op_layer_strided_slice_3 (Te [(None, 768)]         0           tf_albert_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_3 (Dropout)          (None, 3072)              0           global_average_pooling1d[0][0]
_________________________________________________________________________________________________
dense (Dense)                (None, 9)                 27657       dropout_3[0][0]
=================================================================================================
Total params: 11,711,241
Trainable params: 11,711,241
```

```
Non-trainable params: 0
_________________________________________________________________________________________________
```

In [ ]:

```
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[ ]:



In [ ]:

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

In [ ]:

```
# Train data
tokens, masks, segments = compute_input_arrays(X)
def generate_data(tr, cv):
  train_data = {'input_tokens': tokens[tr],
                'input_mask': masks[tr],
                'input_segment': segments[tr]}

  cv_data = {'input_tokens': tokens[cv],
             'input_mask': masks[cv],
             'input_segment': segments[cv]}

  return train_data, cv_data, y.values[tr,21:], y.values[cv,21:]
```

In [ ]:

```
# 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 [ ]:

```
# 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 [ ]:

```
from sklearn.model_selection import KFold
# Compiling and training the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00002)
model_a.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_a.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_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
   2/1216 [..............................] - ETA: 9:49 - loss: 0.5978 - rhos: -0.1166

/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 [==============================] - 1245s 1s/step - loss: 0.4429 - rhos: 0.0757 - val_loss:
0.3963 - val_rhos: 0.2649
1216/1216 [==============================] - 1240s 1s/step - loss: 0.4163 - rhos: 0.1558 - val_loss:
0.3831 - val_rhos: 0.2479
1216/1216 [==============================] - 1240s 1s/step - loss: 0.4040 - rhos: 0.1918 - val_loss:
0.3749 - val_rhos: 0.3082
1216/1216 [==============================] - 1241s 1s/step - loss: 0.3944 - rhos: 0.2191 - val_loss:
0.3805 - val_rhos: 0.3012
WARNING:tensorflow:Gradients do not exist for variables ['tf_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tf_albert_model/pooler/kernel:0', 'tf_albe
rt_model/pooler/bias:0'] when minimizing the loss.
1216/1216 [==============================] - 1220s 1s/step - loss: 0.3829 - rhos: 0.2278 - val_loss:
0.3660 - val_rhos: 0.3438
```

In [ ]:

```
model_a.save_weights("drive/My Drive/albert_model_a.h5")
```

In [ ]:

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

In [ ]:

```
# 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('albert_pred_a_train.csv', index=False)

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

In [ ]: