



In [1]:

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

```
Collecting transformers==2.4.0
  Downloading https://files.pythonhosted.org/packages/c6/38/c30b6a4b86705311c428a234ef752f6c4c4ffdd7
5422a829f1f2766136c3/transformers-2.4.0-py3-none-any.whl (475kB)
    |████████████████████████████████████████| 481kB 5.6MB/s eta 0:00:01
Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from transformers==
2.4.0) (1.14.22)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from tra
nsformers==2.4.0) (2019.12.20)
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a6ff9a5e9fe4f090f
5d95ab341c53d28cbc58/sentencepiece-0.1.91-cp36-cp36m-manylinux1_x86_64.whl (1.1MB)
    |████████████████████████████████████████| 1.1MB 23.4MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers==
2.4.0) (1.18.5)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (2.23.0)
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: filelock in /usr/local/lib/python3.6/dist-packages (from transformers
==2.4.0) (3.0.12)
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/7d/34/09d19aff26edcc8eb2a01bed8e98f13a1537005d
31e95233fd48216eed10/sacremoses-0.0.43.tar.gz (883kB)
    |████████████████████████████████████████| 890kB 38.4MB/s
Collecting tokenizers==0.0.11
  Downloading https://files.pythonhosted.org/packages/5e/36/7af38d572c935f8e0462ec7b4f7a46d73a2b3b1a
938f50a5e8132d5b2dc5/tokenizers-0.0.11-cp36-cp36m-manylinux1_x86_64.whl (3.1MB)
    |████████████████████████████████████████| 3.1MB 39.9MB/s
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: 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: 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: 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: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests
->transformers==2.4.0) (2.10)
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: 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: 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->transformers==2.4.0) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from sacremoses->transformers==2.4.0) (0.16.0)
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)
Building wheels for collected packages: sacremoses
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.43-cp36-none-any.whl size=893260 sha256=67eb145533f6d92a69db2d33cb5df3b59002d18dfabc537f4e28ec28358a420a
  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6fb5e19ceb42ce38d4e58f45
Successfully built sacremoses
Installing collected packages: sentencepiece, sacremoses, tokenizers, transformers
Successfully installed sacremoses-0.0.43 sentencepiece-0.1.91 tokenizers-0.0.11 transformers-2.4.0

```

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')
```

Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect\\_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aob&response\\_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:

.....

Mounted at /content/drive

In [14]:

```
# 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 [15]:

```
# 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 [16]:

```
# 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/indexing.html#returning-a-view-versus-a-copy
    self[name] = value
```

In [17]:

```

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 [18]:

```
# 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_SEQUENCE_LENGTH - len(ids))
    attention_mask = [1] * len(ids) + [0] * (MAX_SEQUENCE_LENGTH - len(ids))
    return padded_ids, token_type_ids, attention_mask
```

In [19]:

```
# 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=question_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 [20]:

```
# 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(30, activation='sigmoid')(x)

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

In [21]:

```
model_qa.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, 30)	92190	global_average_pooling1d[0][0]

Total params: 109,574,430

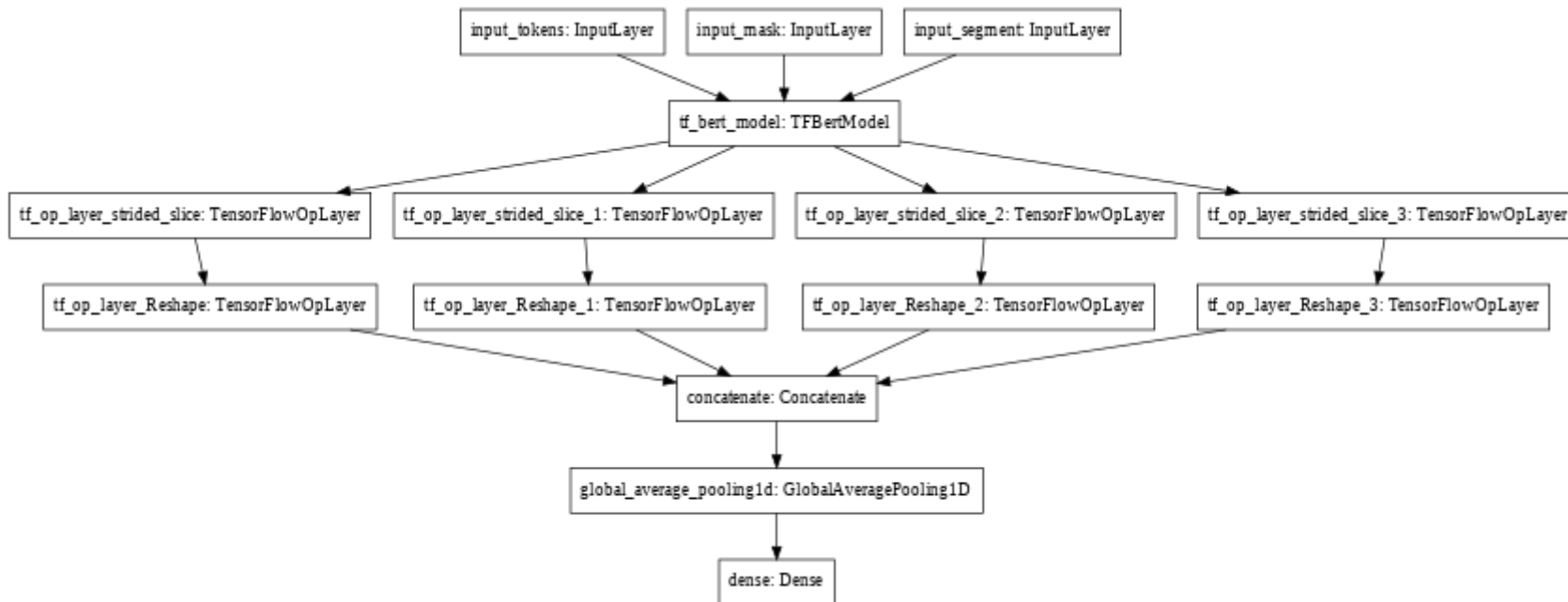
Trainable params: 109,574,430

Non-trainable params: 0

In [22]:

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

Out[22]:



In [23]:

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

In [24]:

```
# 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], y.values[cv]
```

In [25]:

```
# 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 [26]:

```
# 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 [27]:

```
from sklearn.model_selection import KFold
# Compiling and training the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00002)
model_qa.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_qa.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_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.
```

```
1/1216 [.....] - ETA: 0s - loss: 0.7140 - rhos: -0.0374
```

```
/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 [=====] - 411s 338ms/step - loss: 0.3891 - rhos: 0.3139 - val_loss: 0.3753 - val_rhos: 0.3891
1216/1216 [=====] - 409s 336ms/step - loss: 0.3640 - rhos: 0.4196 - val_loss: 0.3561 - val_rhos: 0.4483
1216/1216 [=====] - 409s 337ms/step - loss: 0.3514 - rhos: 0.4684 - val_loss: 0.3458 - val_rhos: 0.5107
1216/1216 [=====] - 409s 336ms/step - loss: 0.3406 - rhos: 0.5074 - val_loss: 0.3274 - val_rhos: 0.5496
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.
1216/1216 [=====] - 390s 321ms/step - loss: 0.3291 - rhos: 0.5508 - val_loss: 0.3160 - val_rhos: 0.6111

```

In [28]:

```
model_qa.save_weights("drive/My Drive/bert_model_qa.h5")
```

In [29]:

```

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

```

In [30]:

```

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

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

df = pd.DataFrame(pred_a_test, columns=y.columns)
df.to_csv('bert_pred_qa_test.csv', index=False)

```



In [ ]: