

In [2]:

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

Collecting transformers==2.4.0

Downloading <https://files.pythonhosted.org/packages/c6/38/c30b6a4b86705311c428a234ef752f6c4c4ffdd75422a829f1f2766136c3/transformers-2.4.0-py3-none-any.whl> (475kB)

|██| 481kB 3.5MB/s eta 0:00:01

Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (from transformers==2.4.0) (4.41.1)

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from transformers==2.4.0) (2.23.0)

Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from transformers==2.4.0) (2019.12.20)

Collecting tokenizers==0.0.11

Downloading https://files.pythonhosted.org/packages/5e/36/7af38d572c935f8e0462ec7b4f7a46d73a2b3b1a938f50a5e8132d5b2dc5/tokenizers-0.0.11-cp36-cp36m-manylinux1_x86_64.whl (3.1MB)

|██| 3.1MB 18.2MB/s

Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers==2.4.0) (1.18.5)

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/34/09d19aff26edcc8eb2a01bed8e98f13a1537005d31e95233fd48216eed10/sacremoses-0.0.43.tar.gz> (883kB)

|██| 890kB 30.9MB/s

Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from transformers==2.4.0) (1.14.22)

Collecting sentencepiece

Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a6ff9a5e9fe4f090f5d95ab341c53d28cbc58/sentencepiece-0.1.91-cp36-cp36m-manylinux1_x86_64.whl (1.1MB)

|██| 1.1MB 25.8MB/s

Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-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 requests->transformers==2.4.0) (2020.6.20)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->transformers==2.4.0) (3.0.4)

Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from sacremoses->transformers==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: 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: 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: 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=a7b9
fb8f110743d51016f7118fad6f109af70e7809d07f9771a9c53c2f6bbe2f
  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6fb5e19ceb42ce38d4
e58f45
Successfully built sacremoses
Installing collected packages: tokenizers, sacremoses, sentencepiece, transformers
Successfully installed sacremoses-0.0.43 sentencepiece-0.1.91 tokenizers-0.0.11 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 [5]:

```
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

Enter your authorization code:

.....

Mounted at /content/drive

In [6]:

```
# 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 [7]:

```
# # 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 [8]:

```
# 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 [9]:

```
# 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 [10]:

```
# 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 [12]:

```
tokenizer = XLNetTokenizer.from_pretrained('xlnet-base-cased')
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 [13]:

```

# 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 = values
        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 [16]:

```
# 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
xlnet_config = XLNetConfig.from_pretrained('xlnet-base-cased', output_hidden_states=True)
xlnet_model = TFXLNetModel.from_pretrained('xlnet-base-cased', config=albert_config)

sequence_output, hidden_states = xlnet_model([input_tokens, input_mask])
# 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], outputs=output)
```

In [17]:

```
model_q.summary()
```

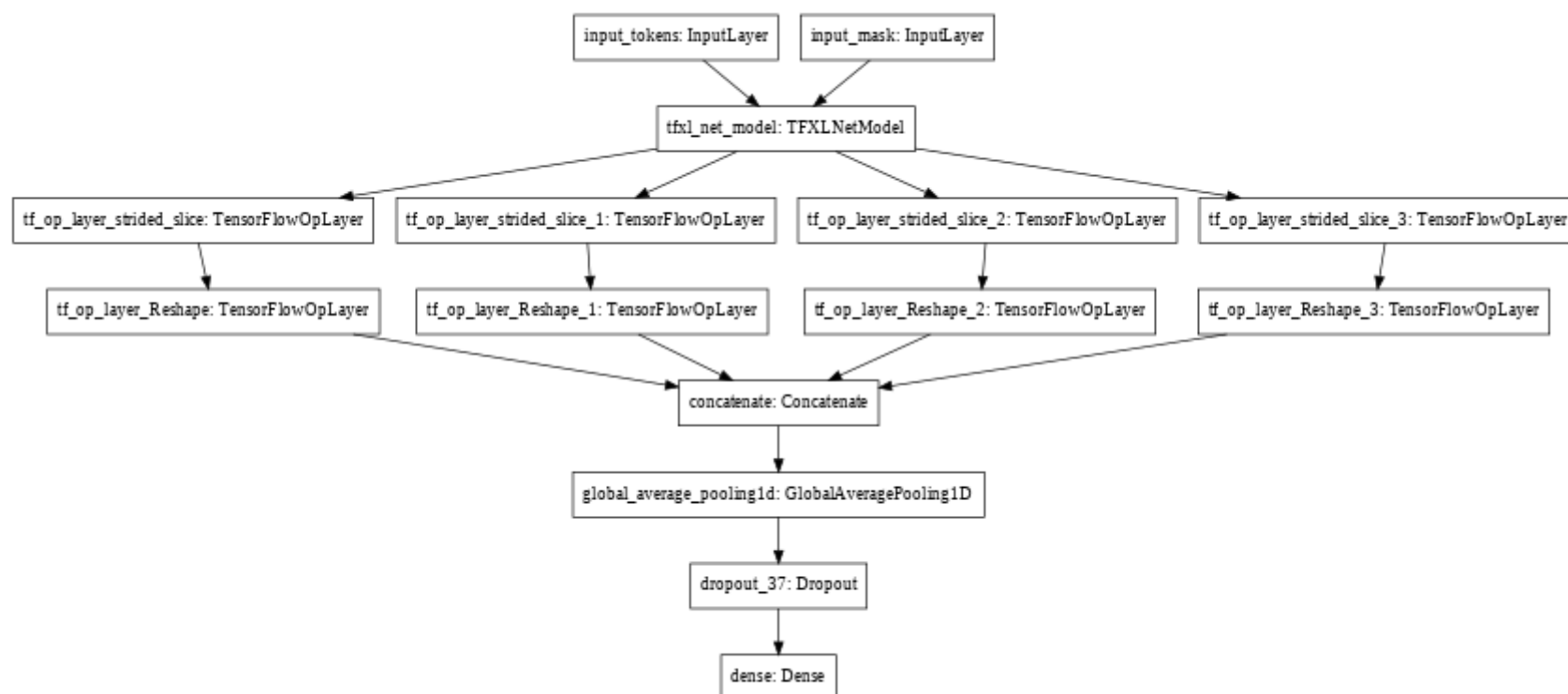
Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_tokens (InputLayer)	[(None, 512)]	0	
input_mask (InputLayer)	[(None, 512)]	0	
tfxl_net_model (TFXLNetModel)	((None, 512, 768), (116718336	input_tokens[0][0] input_mask[0][0]
tf_op_layer_strided_slice (Tens	[(None, 768)]	0	tfxl_net_model[0][13]
tf_op_layer_strided_slice_1 (Te	[(None, 768)]	0	tfxl_net_model[0][12]
tf_op_layer_strided_slice_2 (Te	[(None, 768)]	0	tfxl_net_model[0][11]
tf_op_layer_strided_slice_3 (Te	[(None, 768)]	0	tfxl_net_model[0][10]
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_37 (Dropout)	(None, 3072)	0	global_average_pooling1d[0][0]
dense (Dense)	(None, 21)	64533	dropout_37[0][0]
Total params: 116,782,869			
Trainable params: 116,782,869			
Non-trainable params: 0			

In [18]:

```
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[18]:



In [20]:

```
# 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}
```

In [21]:

```
# 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]}

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

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

In [22]:

```
# 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 [23]:

```
# 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 [24]:

```
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 leave random_state to its default (None), or set shuffle=True.
```

```
FutureWarning
```

```
WARNING:tensorflow:Gradients do not exist for variables ['tfxl_net_model/transformer/mask_emb:0', 'tfxl_net_model/transformer/layer_.0/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.0/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.5/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.5/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/r_s_bias:0', 'tfxl net model/transformer/layer . 7/rel attn/seg embed:0', 'tfxl net model/transformer/laye
```



```
r_.8/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.8/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/seg_embed:0'] when minimizing the loss.
```

```
WARNING:tensorflow:Gradients do not exist for variables ['tfxl_net_model/transformer/mask_emb:0', 'tfxl_net_model/transformer/layer_.0/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.0/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.5/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.5/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.8/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.8/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/seg_embed:0'] when minimizing the loss.
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2534: RuntimeWarning: invalid value encountered in true_divide
```

```
c /= stddev[:, None]
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/function_base.py:2535: RuntimeWarning: invalid value encountered in true_divide
```

```
c /= stddev[None, :]
```

```
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: invalid value encountered in greater
```

```
return (a < x) & (x < b)
```

```
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: RuntimeWarning: invalid value encountered in less
```

```
return (a < x) & (x < b)
```

```
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: RuntimeWarning: invalid value encountered in less_equal
```

```
cond2 = cond0 & (x <= _a)
```

```

1216/1216 [=====] - 2442s 2s/step - loss: 0.4779 - rhos: 0.2379 - val_loss:
0.3803 - val_rhos: 0.4304
1216/1216 [=====] - 2441s 2s/step - loss: 0.4186 - rhos: 0.3411 - val_loss:
0.3713 - val_rhos: 0.4790
1216/1216 [=====] - 2440s 2s/step - loss: 0.3966 - rhos: 0.3926 - val_loss:
0.3621 - val_rhos: 0.5109
1216/1216 [=====] - 2442s 2s/step - loss: 0.3820 - rhos: 0.4235 - val_loss:
0.3473 - val_rhos: 0.5250
WARNING:tensorflow:Gradients do not exist for variables ['tfxl_net_model/transformer/mask_emb:0', 't
fxl_net_model/transformer/layer_.0/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.0/rel_
attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/r_s_bias:0', 'tfxl_net_model/trans
former/layer_.1/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/r_s_bias:0',
'tfxl_net_model/transformer/layer_.2/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.3/r
el_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/seg_embed:0', 'tfxl_net_model/tr
ansformer/layer_.4/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/seg_embed:
0', 'tfxl_net_model/transformer/layer_.5/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.
5/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/r_s_bias:0', 'tfxl_net_mode
l/transformer/layer_.6/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/r_s_bi
as:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/laye
r_.8/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.8/rel_attn/seg_embed:0', 'tfxl_net_m
odel/transformer/layer_.9/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/seg_
embed:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/l
ayer_.10/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/r_s_bias:0', 'tfxl_
net_model/transformer/layer_.11/rel_attn/seg_embed:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables ['tfxl_net_model/transformer/mask_emb:0', 't
fxl_net_model/transformer/layer_.0/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.0/rel_
attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.1/rel_attn/r_s_bias:0', 'tfxl_net_model/trans
former/layer_.1/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.2/rel_attn/r_s_bias:0',
'tfxl_net_model/transformer/layer_.2/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.3/r
el_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.3/rel_attn/seg_embed:0', 'tfxl_net_model/tr
ansformer/layer_.4/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.4/rel_attn/seg_embed:
0', 'tfxl_net_model/transformer/layer_.5/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.
5/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.6/rel_attn/r_s_bias:0', 'tfxl_net_mode
l/transformer/layer_.6/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/r_s_bi
as:0', 'tfxl_net_model/transformer/layer_.7/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/laye
r_.8/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.8/rel_attn/seg_embed:0', 'tfxl_net_m
odel/transformer/layer_.9/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/layer_.9/rel_attn/seg_
embed:0', 'tfxl_net_model/transformer/layer_.10/rel_attn/r_s_bias:0', 'tfxl_net_model/transformer/l
ayer_.10/rel_attn/seg_embed:0', 'tfxl_net_model/transformer/layer_.11/rel_attn/r_s_bias:0', 'tfxl_
net_model/transformer/layer_.11/rel_attn/seg_embed:0'] when minimizing the loss.
1216/1216 [=====] - 2450s 2s/step - loss: 0.3662 - rhos: 0.4679 - val_loss:
0.3412 - val_rhos: 0.5685

```

In [25]:

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

In [27]:

```
# 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}
```

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('xlnet_pred_q_train.csv', index=False)

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

In []:

In []: