In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import pickle
import re
from scipy.stats import spearmanr
from tqdm import tqdm
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

```
Using TensorFlow backend.
```

In [6]:

```
train = pd.read_pickle('./train')
train.head()
```

Out[6]:

	qa_id	question_title	question_body	question_user_name	question
0	0	what am i losing when using extension tubes in...	after playing around with macro photography on...	ysap	https://photo.stackexchange.con
1	1	what is the distinction between a city and a s...	i am trying to understand what kinds of places...	russellpierce	https://rpg.stackexchange.con
2	2	maximum protusion length for through hole comp...	i am working on a pcb that has through hole co...	Joe Baker	https://electronics.stackexchange.com/
3	3	can an affidavit be used in beit din	an affidavit from what i understand is basical...	Scimonster	https://judaism.stackexchange.con
4	5	how do you make a binary image in photoshop	i am trying to make a binary image i want more...	leigero	https://graphicdesign.stackexchange.(

In [3]:

```
test = pd.read_pickle('./test')
test.head()
```

Out[3]:

	qa_id	question_title	question_body	question_user_name	que
0	39	will leaving corpses lying around upset my pri...	i see questionsinformation online about how to...	Dylan	https://gaming.stackexchange.
1	46	url link to feature image in the portfolio	i am new to wordpress i have issue with featur...	Anu	https://wordpress.stackexchange.
2	70	is accuracy recoil or bullet spread affected b...	to experiment i started a bot game toggled inv...	Konsta	https://gaming.stackexchange.
3	132	suddenly got an io error from my external hdd	i have used my raspberry pi as a torrent serve...	robbannn	https://raspberrypi.stackexchange.
4	200	passenger name flight booking passenger only...	i have bought delhi london return flights for ...	Amit	https://travel.stackexchange.

In [7]:

```
X = pd.DataFrame()
for i in test.columns:
    X[i] = train[i]
```

In [8]:

```
X.columns
```

Out[8]:

```
Index(['qa_id', 'question_title', 'question_body', 'question_user_name',
       'question_user_page', 'answer', 'answer_user_name', 'answer_user_pag
e',
       'url', 'category', 'host'],
      dtype='object')
```

In [9]:

```
features = ['question_title','question_body','answer']
X[features]
```

Out[9]:

	question_title	question_body	answer
0	what am i losing when using extension tubes in...	after playing around with macro photography on...	i just got extension tubes so here is the skin...
1	what is the distinction between a city and a s...	i am trying to understand what kinds of places...	it might be helpful to look into the definitio...
2	maximum protusion length for through hole comp...	i am working on a pcb that has through hole co...	do you even need grooves we make several pro...
3	can an affidavit be used in beit din	an affidavit from what i understand is basical...	sending an affidavit it is a dispute between...
4	how do you make a binary image in photoshop	i am trying to make a binary image i want more...	check out image trace in adobe illustrator i...
...	...	...	...
6074	using a ski helmet for winter biking	i am curious if anyone uses a skiing helmet fo...	if you are thinking about wearing a ski helmet...
6075	adjustment to road bike brakes for high grade ...	i have a road bike with a front brake that wea...	you can replace the pads as stated elsewhere...
6076	suppress file truncated messages when using tail	i am tailing a log file using tail f messages...	maybe help if can be fixes origin of this erro...
6077	when should a supervisor be a co author	what are people is views on this to be speci...	as a non mathematician i am somewhat mystified...
6078	why are there so many different types of screw...	newbie question why is it that there is a baz...	first i really like eric is answer for practic...

6079 rows × 3 columns

In [10]:

```
y = pd.DataFrame()
for i in train.columns:
    if i not in X.columns:
        y[i] = train[i]
```

In [11]:

```
y.columns
```

Out[11]:

```
Index(['question_asker_intent_understanding', 'question_body_critical',
       'question_conversational', 'question_expect_short_answer',
       'question_fact_seeking', 'question_has_commonly_accepted_answer',
       'question_interestingness_others', 'question_interestingness_self',
       'question_multi_intent', 'question_not_really_a_question',
       'question_opinion_seeking', 'question_type_choice',
       'question_type_compare', 'question_type_consequence',
       'question_type_definition', 'question_type_entity',
       'question_type_instructions', 'question_type_procedure',
       'question_type_reason_explanation', 'question_type_spelling',
       'question_well_written', 'answer_helpful',
       'answer_level_of_information', 'answer_plausible', 'answer_relevanc
e',
       'answer_satisfaction', 'answer_type_instructions',
       'answer_type_procedure', 'answer_type_reason_explanation',
       'answer_well_written'],
      dtype='object')
```

In [12]:

```
words = train['question_title'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage question_title text contains words less than : {}".format(i,words[
```

```
91 percentage question_title text contains words less than : 16
92 percentage question_title text contains words less than : 17
93 percentage question_title text contains words less than : 17
94 percentage question_title text contains words less than : 17
95 percentage question_title text contains words less than : 18
96 percentage question_title text contains words less than : 19
97 percentage question_title text contains words less than : 20
98 percentage question_title text contains words less than : 21
99 percentage question_title text contains words less than : 23
```

In [13]:

```
words = train['question_body'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage question_body text contains words less than : {}".format(i,words[r
```

```
91 percentage question_body text contains words less than : 434
92 percentage question_body text contains words less than : 460
93 percentage question_body text contains words less than : 508
94 percentage question_body text contains words less than : 557
95 percentage question_body text contains words less than : 613
96 percentage question_body text contains words less than : 720
97 percentage question_body text contains words less than : 883
98 percentage question_body text contains words less than : 1072
99 percentage question_body text contains words less than : 1489
```

In [14]:

```
words = train['answer'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage answer text contains words less than : {}".format(i,words[round((l
```

```
91 percentage answer text contains words less than : 416
92 percentage answer text contains words less than : 442
93 percentage answer text contains words less than : 473
94 percentage answer text contains words less than : 509
95 percentage answer text contains words less than : 568
96 percentage answer text contains words less than : 638
97 percentage answer text contains words less than : 724
98 percentage answer text contains words less than : 843
99 percentage answer text contains words less than : 1237
```

In [15]:

```
X_train, X_cv, y_train, y_cv = train_test_split(X[features], y.values, test_size=0.20)
```

In [16]:

```
import tensorflow as tf
import keras
import os
import random as rn
import tensorflow_hub as hub
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, Flatten,concatenate,Input, Dropout,Batc
from tensorflow.keras import backend as K
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint,Callback
np.random.seed(42)

tf.random.set_seed(20)

rn.seed(12)
```

In [272]:

```
tf.keras.backend.clear_session()

max_seq_length = 25

input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input

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

segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_

bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])

bert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_output)
```

In [273]:

```
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
```

In [ ]:

```
!pip install sentencepiece
```

```
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a7
8cca907a6ff9a5e9fe4f090f5d95ab341c53d28cbc58/sentencepiece-0.1.91-cp36-cp36m
-manylinux1_x86_64.whl (https://files.pythonhosted.org/packages/d4/a4/d0a884
c4300004a78cca907a6ff9a5e9fe4f090f5d95ab341c53d28cbc58/sentencepiece-0.1.91-
cp36-cp36m-manylinux1_x86_64.whl) (1.1MB)
     |████████████████████████████████| 1.1MB 5.2MB/s eta 0:00:01
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.91
```

In [ ]:

```
import tokenization
```

In [274]:

```
tokenizer = tokenization.FullTokenizer(vocab_file,do_lower_case)
```

In [275]:

```
X_train_qt_tokens = np.zeros(shape=(X_train.shape[0],25))
X_train_qt_mask = np.zeros(shape=(X_train.shape[0],25))
X_train_qt_segment = np.zeros(shape=(X_train.shape[0],25))
X_cv_qt_tokens = np.zeros(shape=(X_cv.shape[0],25))
X_cv_qt_mask = np.zeros(shape=(X_cv.shape[0],25))
X_cv_qt_segment = np.zeros(shape=(X_cv.shape[0],25))
```

In [276]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][0])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_qt_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_qt_segment[i] = np.array([0]*max_seq_length)
```

In [277]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][0])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_qt_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_qt_segment[i] = np.array([0]*max_seq_length)
```

In [278]:

```
X_train_qt_pooled_output = bert_model.predict([X_train_qt_tokens.astype('int32'), X_train_q
X_cv_qt_pooled_output = bert_model.predict([X_cv_qt_tokens.astype('int32'), X_cv_qt_mask.as
```

In [279]:

```
tf.keras.backend.clear_session()

max_seq_length = 512

input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input

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

segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_

bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])

bert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_output)
```

In [280]:

```
#getting Vocab file
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()

tokenizer = tokenization.FullTokenizer(vocab_file,do_lower_case)
```

In [281]:

```
X_train_q_tokens = np.zeros(shape=(X_train.shape[0],512))
X_train_q_mask = np.zeros(shape=(X_train.shape[0],512))
X_train_q_segment = np.zeros(shape=(X_train.shape[0],512))
X_cv_q_tokens = np.zeros(shape=(X_cv.shape[0],512))
X_cv_q_mask = np.zeros(shape=(X_cv.shape[0],512))
X_cv_q_segment = np.zeros(shape=(X_cv.shape[0],512))
```

In [282]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][1])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_q_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_q_segment[i] = np.array([0]*max_seq_length)
```

In [283]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][1])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_q_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_q_segment[i] = np.array([0]*max_seq_length)
```

In [284]:

```
X_train_q_pooled_output = bert_model.predict([X_train_q_tokens.astype('int32'), X_train_q_m
X_cv_q_pooled_output = bert_model.predict([X_cv_q_tokens.astype('int32'), X_cv_q_mask.astyp
```

In [285]:

```
X_train_a_tokens = np.zeros(shape=(X_train.shape[0],512))
X_train_a_mask = np.zeros(shape=(X_train.shape[0],512))
X_train_a_segment = np.zeros(shape=(X_train.shape[0],512))
X_cv_a_tokens = np.zeros(shape=(X_cv.shape[0],512))
X_cv_a_mask = np.zeros(shape=(X_cv.shape[0],512))
X_cv_a_segment = np.zeros(shape=(X_cv.shape[0],512))
```

In [286]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][2])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_a_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_a_segment[i] = np.array([0]*max_seq_length)
```

In [287]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][2])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_a_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_a_segment[i] = np.array([0]*max_seq_length)
```

In [288]:

```
X_train_a_pooled_output = bert_model.predict([X_train_a_tokens.astype('int32'), X_train_a_m
X_cv_a_pooled_output = bert_model.predict([X_cv_a_tokens.astype('int32'), X_cv_a_mask.astyp
```

In [45]:

```
class SpearmanCallback(Callback):
    def __init__(self, validation_data, patience, model_name):
        self.x_val = validation_data[0]
        self.y_val = validation_data[1]
        self.patience = patience
        self.value = -1
        self.bad_epochs = 0
        self.model_name = model_name

    def on_epoch_end(self, epoch, logs={}):
        y_pred_val = self.model.predict(self.x_val)
        rho_val = np.mean([spearmanr(self.y_val[:, ind], y_pred_val[:, ind]+ np.random.norm
        if rho_val >= self.value:
            self.value = rho_val
            self.bad_epochs = 0
            self.model.save_weights(self.model_name)
        else:
            self.bad_epochs += 1
        if self.bad_epochs >= self.patience:
            print("early stopping Threshold")
            self.model.stop_training = True
        print('\nbad: {}'.format(self.bad_epochs))
        print('\nval_spearman-corr: %s' % (str(round(rho_val, 4))), end=100*' '+'\n')
        return rho_val
```

Bert Model

In [339]:

```
K.clear_session()
input_1 = Input(shape=(X_train_qt_pooled_output.shape[1]))
qt = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(s

input_2 = Input(shape=(X_train_q_pooled_output.shape[1]))
q = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(se

input_3 = Input(shape=(X_train_a_pooled_output.shape[1]))
a = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(se

concat = concatenate([qt, q, a])

output = Dense(30, activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_uni

model_bert = Model(inputs=[input_1, input_2, input_3], outputs=output)

model_bert.summary()
```

```
Model: "model"
____________________________________________________________________________
______________________
Layer (type)                    Output Shape         Param #     Connected t
o
============================================================================
======================
input_1 (InputLayer)            [(None, 768)]        0
____________________________________________________________________________
______________________
input_2 (InputLayer)            [(None, 768)]        0
____________________________________________________________________________
______________________
input_3 (InputLayer)            [(None, 768)]        0
____________________________________________________________________________
______________________
dense (Dense)                   (None, 66)           50754       input_1[0]
[0]
____________________________________________________________________________
______________________
dense_1 (Dense)                 (None, 66)           50754       input_2[0]
[0]
____________________________________________________________________________
______________________
dense_2 (Dense)                 (None, 66)           50754       input_3[0]
[0]
____________________________________________________________________________
______________________
concatenate (Concatenate)       (None, 198)          0           dense[0][0]
                                                                 dense_1[0]
[0]
                                                                 dense_2[0]
[0]
____________________________________________________________________________
______________________
dense_3 (Dense)                 (None, 30)           5970        concatenate
[0][0]
============================================================================
======================
Total params: 158,232
Trainable params: 158,232
```

```
Non-trainable params: 0
________________________________________________________________________________
```

In [340]:

```
!rm -rf model_bert
```

In [341]:

```
train_data = [X_train_qt_pooled_output,X_train_q_pooled_output,X_train_a_pooled_output]
cv_data = [X_cv_qt_pooled_output,X_cv_q_pooled_output,X_cv_a_pooled_output]
```

In [342]:

```
model_bert.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='binary_crossentropy')
```

In [343]:

```
log_dir_1 = os.path.join('model_bert')

tensorboard_callback1 = tf.keras.callbacks.TensorBoard(log_dir=log_dir_1, histogram_freq=1,
                                                      write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.
```

In [344]:

```
model_bert.fit(train_data, y_train, batch_size=32, epochs=100, verbose=1,
           validation_data=(cv_data, y_cv),
           callbacks=[SpearmanCallback(validation_data=(cv_data, y_cv),patience=5,
                       model_name='best_weight_bert.h5'),tensorboard_callback1])
```

```
Epoch 1/100
146/152 [===========================>..] - ETA: 0s - loss: 0.4271
bad: 0

val_spearman-corr: 0.2001
152/152 [==============================] - 1s 7ms/step - loss: 0.4265 - va
l_loss: 0.4121
Epoch 2/100
146/152 [===========================>..] - ETA: 0s - loss: 0.4084
bad: 0

val_spearman-corr: 0.2522
152/152 [==============================] - 1s 6ms/step - loss: 0.4082 - va
l_loss: 0.4062
Epoch 3/100
143/152 [===========================>..] - ETA: 0s - loss: 0.4032
bad: 0

val_spearman-corr: 0.2674
```

In [345]:

```
%%time
model_bert.load_weights("best_weight_bert.h5")
y_pred_val = model_bert.predict(cv_data)
sp = np.mean([spearmanr(y_cv[:, ind], y_pred_val[:, ind]).correlation for ind in range(y_pr
```

```
CPU times: user 136 ms, sys: 14.6 ms, total: 151 ms
Wall time: 130 ms
```

In [346]:

```
print("spearman {}".format(sp))
```

```
spearman 0.34591226798195923
```

In [347]:

```
%load_ext tensorboard
%tensorboard --logdir model_bert
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6008 (pid 14705), started 0:26:01 ago. (Use '!ki
ll 14705' to kill it.)

<IPython.core.display.Javascript object>
```

Bert Model with convolution1D

In [365]:

```
K.clear_session()
input_1 = Input(shape=(X_train_qt_pooled_output.shape[1],1))
con1D_1 = Conv1D(10, 20, activation='relu',name = 'con1D_1',kernel_initializer=tf.keras.ini
con1D_4 = Conv1D(20, 15, activation='relu',name = 'con1D_4',kernel_initializer=tf.keras.ini
con1D_5 = Conv1D(30, 10, activation='relu',name = 'con1D_5',kernel_initializer=tf.keras.ini
maxpool1 = MaxPooling1D(pool_size=3,name = 'maxpool1')(con1D_5)
qt = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=2

input_2 = Input(shape=(X_train_q_pooled_output.shape[1],1))
con1D_2 = Conv1D(10, 20, activation='relu',name = 'con1D_2',kernel_initializer=tf.keras.ini
con1D_6 = Conv1D(20, 15, activation='relu',name = 'con1D_6',kernel_initializer=tf.keras.ini
con1D_7 = Conv1D(30, 10, activation='relu',name = 'con1D_7',kernel_initializer=tf.keras.ini
maxpool2 = MaxPooling1D(pool_size=3,name = 'maxpool2')(con1D_7)
q = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20

input_3 = Input(shape=(X_train_a_pooled_output.shape[1],1))
con1D_3 = Conv1D(10, 20, activation='relu',name = 'con1D_3',kernel_initializer=tf.keras.ini
con1D_8 = Conv1D(20, 15, activation='relu',name = 'con1D_8',kernel_initializer=tf.keras.ini
con1D_9 = Conv1D(30, 10, activation='relu',name = 'con1D_9',kernel_initializer=tf.keras.ini
maxpool3 = MaxPooling1D(pool_size=3,name = 'maxpool3')(con1D_9)
a = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20

concat = concatenate([qt,q,a])
flat = Flatten()(concat)

output = Dense(30, activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_uni

model = Model(inputs=[input_1, input_2, input_3], outputs=output)

model.summary()
```

```
Model: "model"
__________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================
========================
input_1 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________
________________________
input_2 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________
________________________
input_3 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________
________________________
con1D_1 (Conv1D)                (None, 749, 10)      210         input_1
[0][0]
__________________________________________________________________________
________________________
con1D_2 (Conv1D)                (None, 749, 10)      210         input_2
[0][0]
__________________________________________________________________________
________________________
con1D_3 (Conv1D)                (None, 749, 10)      210         input_3
[0][0]
__________________________________________________________________________
```

```
________________________________
con1D_4 (Conv1D)                 (None, 735, 20)     3020        con1D_1
[0][0]

__________________________________________________________________________
________________________________
con1D_6 (Conv1D)                 (None, 735, 20)     3020        con1D_2
[0][0]

__________________________________________________________________________
________________________________
con1D_8 (Conv1D)                 (None, 735, 20)     3020        con1D_3
[0][0]

__________________________________________________________________________
________________________________
con1D_5 (Conv1D)                 (None, 726, 30)     6030        con1D_4
[0][0]

__________________________________________________________________________
________________________________
con1D_7 (Conv1D)                 (None, 726, 30)     6030        con1D_6
[0][0]

__________________________________________________________________________
________________________________
con1D_9 (Conv1D)                 (None, 726, 30)     6030        con1D_8
[0][0]

__________________________________________________________________________
________________________________
maxpool1 (MaxPooling1D)          (None, 242, 30)     0           con1D_5
[0][0]

__________________________________________________________________________
________________________________
maxpool2 (MaxPooling1D)          (None, 242, 30)     0           con1D_7
[0][0]

__________________________________________________________________________
________________________________
maxpool3 (MaxPooling1D)          (None, 242, 30)     0           con1D_9
[0][0]

__________________________________________________________________________
________________________________
dense (Dense)                    (None, 242, 66)     2046        maxpool1
[0][0]

__________________________________________________________________________
________________________________
dense_1 (Dense)                  (None, 242, 66)     2046        maxpool2
[0][0]

__________________________________________________________________________
________________________________
dense_2 (Dense)                  (None, 242, 66)     2046        maxpool3
[0][0]

__________________________________________________________________________
________________________________
concatenate (Concatenate)        (None, 242, 198)    0           dense[0]
[0]
                                                                 dense_1
[0][0]
                                                                 dense_2
[0][0]

__________________________________________________________________________
________________________________
flatten (Flatten)                (None, 47916)       0           concatena
te[0][0]

__________________________________________________________________________
________________________________
```

```
dense_3 (Dense)                  (None, 30)           1437510     flatten
[0][0]
==========================================================================
========================
Total params: 1,471,428
Trainable params: 1,471,428
Non-trainable params: 0

__________________________________________________________________________

________________________
```

In [363]:

```
!rm -rf model
```

In [366]:

```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='binary_crossentropy')
```

In [367]:

```
log_dir_2 = os.path.join('model')

tensorboard_callback2 = tf.keras.callbacks.TensorBoard(log_dir=log_dir_2, histogram_freq=1,
                                                      write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.
```

In [368]:

```
model.fit(train_data, y_train, batch_size=32, epochs=100, verbose=1,
           validation_data=(cv_data, y_cv),
           callbacks=[SpearmanCallback(validation_data=(cv_data, y_cv),patience=5,
                        model_name='best_weight_bert_2.h5'),tensorboard_callback2])
```

```
Epoch 1/100
150/152 [============================>.] - ETA: 0s - loss: 0.4176
bad: 0

val_spearman-corr: 0.2691
152/152 [==============================] - 5s 36ms/step - loss: 0.4174 - val
_loss: 0.4056
Epoch 2/100
152/152 [==============================] - ETA: 0s - loss: 0.3998
bad: 0

val_spearman-corr: 0.2991
152/152 [==============================] - 5s 32ms/step - loss: 0.3998 - val
_loss: 0.3993
Epoch 3/100
151/152 [============================>.] - ETA: 0s - loss: 0.3932
bad: 0

val_spearman-corr: 0.309
152/152 [==============================] - 5s 33ms/step - loss: 0.3931 - val
_loss: 0.3944
Epoch 4/100
151/152 [============================>.] - ETA: 0s - loss: 0.3876
bad: 0

val_spearman-corr: 0.3134
152/152 [==============================] - 5s 32ms/step - loss: 0.3877 - val
_loss: 0.3942
Epoch 5/100
151/152 [============================>.] - ETA: 0s - loss: 0.3837
bad: 0

val_spearman-corr: 0.3212
152/152 [==============================] - 5s 32ms/step - loss: 0.3837 - val
_loss: 0.3927
Epoch 6/100
151/152 [============================>.] - ETA: 0s - loss: 0.3797
bad: 0

val_spearman-corr: 0.3231
152/152 [==============================] - 5s 33ms/step - loss: 0.3798 - val
_loss: 0.3906
Epoch 7/100
152/152 [==============================] - ETA: 0s - loss: 0.3758
bad: 1

val_spearman-corr: 0.3218
152/152 [==============================] - 5s 32ms/step - loss: 0.3758 - val
_loss: 0.3909
Epoch 8/100
152/152 [==============================] - ETA: 0s - loss: 0.3711
bad: 0

val_spearman-corr: 0.3283
```

```
152/152 [==============================] - 5s 32ms/step - loss: 0.3711 - val
_loss: 0.3923
Epoch 9/100
151/152 [============================>.] - ETA: 0s - loss: 0.3673
bad: 0

val_spearman-corr: 0.3315
152/152 [==============================] - 5s 32ms/step - loss: 0.3673 - val
_loss: 0.3919
Epoch 10/100
151/152 [============================>.] - ETA: 0s - loss: 0.3631
bad: 1

val_spearman-corr: 0.3304
152/152 [==============================] - 5s 33ms/step - loss: 0.3632 - val
_loss: 0.3951
Epoch 11/100
151/152 [============================>.] - ETA: 0s - loss: 0.3589
bad: 2

val_spearman-corr: 0.3261
152/152 [==============================] - 5s 33ms/step - loss: 0.3590 - val
_loss: 0.3895
Epoch 12/100
152/152 [==============================] - ETA: 0s - loss: 0.3545
bad: 3

val_spearman-corr: 0.3258
152/152 [==============================] - 5s 32ms/step - loss: 0.3545 - val
_loss: 0.3927
Epoch 13/100
151/152 [============================>.] - ETA: 0s - loss: 0.3495
bad: 4

val_spearman-corr: 0.3217
152/152 [==============================] - 5s 32ms/step - loss: 0.3497 - val
_loss: 0.3931
Epoch 14/100
151/152 [============================>.] - ETA: 0s - loss: 0.3450early stopp
ing Threshold

bad: 5

val_spearman-corr: 0.3241
152/152 [==============================] - 5s 32ms/step - loss: 0.3451 - val
_loss: 0.3943
```

Out[368]:

```
<tensorflow.python.keras.callbacks.History at 0x7f42fce0a320>
```

In [369]:

```
%%time
model.load_weights("best_weight_bert_2.h5")
y_pred_val = model.predict(cv_data)
sp = np.mean([spearmanr(y_cv[:, ind], y_pred_val[:, ind]).correlation for ind in range(y_pr
```

```
CPU times: user 371 ms, sys: 84.4 ms, total: 455 ms
Wall time: 375 ms
```

In [370]:

```
print("spearman {}".format(sp))
```

```
spearman 0.33145174574938835
```

In [371]:

```
%load_ext tensorboard
%tensorboard --logdir model
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6009 (pid 16299), started 0:17:59 ago. (Use '!ki
ll 16299' to kill it.)

<IPython.core.display.Javascript object>
```

In [1]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model","spearman"]
x.add_row(["base model", 0.28504])
x.add_row(["bert model", 0.34591])
x.add_row(["bert model conv1D", 0.33145])

print(x)
```

```
+-------------------+----------+
|       Model       | spearman |
+-------------------+----------+
|     base model    | 0.28504  |
|     bert model    | 0.34591  |
| bert model conv1D | 0.33145  |
+-------------------+----------+
```

Albert

In [372]:

```
tf.keras.backend.clear_session()

max_seq_length = 25
input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input

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

segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_

albert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/albert_en_base/1", trainable=Tr

pooled_output, sequence_output = albert_layer([input_word_ids, input_mask, segment_ids])

albert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_outpu
```

In [19]:

```
!pip install sentencepiece
```

```
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a7
8cca907a6ff9a5e9fe4f090f5d95ab341c53d28cbc58/sentencepiece-0.1.91-cp36-cp36m
-manylinux1_x86_64.whl (https://files.pythonhosted.org/packages/d4/a4/d0a884
c4300004a78cca907a6ff9a5e9fe4f090f5d95ab341c53d28cbc58/sentencepiece-0.1.91-
cp36-cp36m-manylinux1_x86_64.whl) (1.1MB)
     |████████████████████████████████| 1.1MB 6.6MB/s eta 0:00:01
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.91
```

In [20]:

```
import tokenization
```

In [373]:

```
sp_model_file = albert_layer.resolved_object.sp_model_file.asset_path.numpy()
tokenizer = tokenization.FullSentencePieceTokenizer(sp_model_file)
```

In [374]:

```
X_train_qt_tokens = np.zeros(shape=(X_train.shape[0],25))
X_train_qt_mask = np.zeros(shape=(X_train.shape[0],25))
X_train_qt_segment = np.zeros(shape=(X_train.shape[0],25))
X_cv_qt_tokens = np.zeros(shape=(X_cv.shape[0],25))
X_cv_qt_mask = np.zeros(shape=(X_cv.shape[0],25))
X_cv_qt_segment = np.zeros(shape=(X_cv.shape[0],25))
```

In [375]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][0])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_qt_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_qt_segment[i] = np.array([0]*max_seq_length)
```

In [376]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][0])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_qt_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_qt_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_qt_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_qt_segment[i] = np.array([0]*max_seq_length)
```

In [377]:

```
X_train_qt_pooled_output = albert_model.predict([X_train_qt_tokens.astype('int32'), X_train
X_cv_qt_pooled_output = albert_model.predict([X_cv_qt_tokens.astype('int32'), X_cv_qt_mask.
```

In [378]:

```
tf.keras.backend.clear_session()

max_seq_length = 512
input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input

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

segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_

albert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/albert_en_base/1", trainable=Tr

pooled_output, sequence_output = albert_layer([input_word_ids, input_mask, segment_ids])

albert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_outpu
```

In [379]:

```
sp_model_file = albert_layer.resolved_object.sp_model_file.asset_path.numpy()
tokenizer = tokenization.FullSentencePieceTokenizer(sp_model_file)
```

In [380]:

```
X_train_q_tokens = np.zeros(shape=(X_train.shape[0],512))
X_train_q_mask = np.zeros(shape=(X_train.shape[0],512))
X_train_q_segment = np.zeros(shape=(X_train.shape[0],512))
X_cv_q_tokens = np.zeros(shape=(X_cv.shape[0],512))
X_cv_q_mask = np.zeros(shape=(X_cv.shape[0],512))
X_cv_q_segment = np.zeros(shape=(X_cv.shape[0],512))
```

In [381]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][1])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_q_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_q_segment[i] = np.array([0]*max_seq_length)
```

In [382]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][1])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_q_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_q_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_q_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_q_segment[i] = np.array([0]*max_seq_length)
```

In [383]:

```
X_train_q_pooled_output = albert_model.predict([X_train_q_tokens.astype('int32'), X_train_q
X_cv_q_pooled_output = albert_model.predict([X_cv_q_tokens.astype('int32'), X_cv_q_mask.ast
```

In [384]:

```
X_train_a_tokens = np.zeros(shape=(X_train.shape[0],512))
X_train_a_mask = np.zeros(shape=(X_train.shape[0],512))
X_train_a_segment = np.zeros(shape=(X_train.shape[0],512))
X_cv_a_tokens = np.zeros(shape=(X_cv.shape[0],512))
X_cv_a_mask = np.zeros(shape=(X_cv.shape[0],512))
X_cv_a_segment = np.zeros(shape=(X_cv.shape[0],512))
```

In [385]:

```
for i in range(X_train.shape[0]):
    tokens = tokenizer.tokenize(X_train.values[i][2])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_train_a_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_train_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_train_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_train_a_segment[i] = np.array([0]*max_seq_length)
```

In [386]:

```
for i in range(X_cv.shape[0]):
    tokens = tokenizer.tokenize(X_cv.values[i][2])
    if len(tokens) >= max_seq_length-2:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        X_cv_a_segment[i] = np.array([0]*max_seq_length)
    else:
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_cv_a_mask[i] = np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens)))
        tokens = tokens + ['[PAD]']*(max_seq_length-len(tokens))
        X_cv_a_tokens[i] = np.array(tokenizer.convert_tokens_to_ids(tokens))
        X_cv_a_segment[i] = np.array([0]*max_seq_length)
```

In [387]:

```
X_train_a_pooled_output = albert_model.predict([X_train_a_tokens.astype('int32'), X_train_a
X_cv_a_pooled_output = albert_model.predict([X_cv_a_tokens.astype('int32'), X_cv_a_mask.ast
```

In [388]:

```
train_data = [X_train_qt_pooled_output,X_train_q_pooled_output,X_train_a_pooled_output]
cv_data = [X_cv_qt_pooled_output,X_cv_q_pooled_output,X_cv_a_pooled_output]
```

Albert Model

In [389]:

```
K.clear_session()
input_1 = Input(shape=(X_train_qt_pooled_output.shape[1],))
qt = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(s

input_2 = Input(shape=(X_train_q_pooled_output.shape[1],))
q = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(se

input_3 = Input(shape=(X_train_a_pooled_output.shape[1],))
a = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.glorot_uniform(se

concat = concatenate([qt, q, a])

output = Dense(30, activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_uni

model_albert = Model(inputs=[input_1, input_2, input_3], outputs=output)

model_albert.summary()
```

```
Model: "model"
__________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================
========================
input_1 (InputLayer)            [(None, 768)]        0
__________________________________________________________________________
________________________
input_2 (InputLayer)            [(None, 768)]        0
__________________________________________________________________________
________________________
input_3 (InputLayer)            [(None, 768)]        0
__________________________________________________________________________
________________________
dense (Dense)                   (None, 66)           50754       input_1
[0][0]
__________________________________________________________________________
________________________
dense_1 (Dense)                 (None, 66)           50754       input_2
[0][0]
__________________________________________________________________________
________________________
dense_2 (Dense)                 (None, 66)           50754       input_3
[0][0]
__________________________________________________________________________
________________________
concatenate (Concatenate)       (None, 198)          0           dense[0]
[0]
                                                                 dense_1
[0][0]
                                                                 dense_2
[0][0]
__________________________________________________________________________
________________________
dense_3 (Dense)                 (None, 30)           5970        concatena
te[0][0]
==========================================================================
========================
Total params: 158,232
Trainable params: 158,232
Non-trainable params: 0
________________________________________________________________________________
_________________________________
```

In [390]:

```
!rm -rf model_albert
```

In [391]:

```
model_albert.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='binary_crossentropy
```

In [393]:

```
log_dir_3 = os.path.join('model_albert')

tensorboard_callback3 = tf.keras.callbacks.TensorBoard(log_dir=log_dir_3, histogram_freq=1,
                                                      write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.
```

In [394]:

```
model_albert.fit(train_data, y_train, batch_size=32, epochs=100, verbose=1,
           validation_data=(cv_data, y_cv),
           callbacks=[SpearmanCallback(validation_data=(cv_data, y_cv),patience=5,
                      model_name='best_weight_bert_3.h5'),tensorboard_callback3])
```

```
Epoch 1/100
  2/152 [..............................] - ETA: 27s - loss: 0.7772WARNING:
tensorflow:Method (on_train_batch_end) is slow compared to the batch updat
e (0.184489). Check your callbacks.

WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the bat
ch update (0.184489). Check your callbacks.
```

In [395]:

```
%%time
model_albert.load_weights("best_weight_bert_3.h5")
y_pred_val = model_albert.predict(cv_data)
sp = np.mean([spearmanr(y_cv[:, ind], y_pred_val[:, ind]).correlation for ind in range(y_pr
```

```
CPU times: user 143 ms, sys: 8.43 ms, total: 152 ms
Wall time: 128 ms
```

In [396]:

```
print("spearman {}".format(sp))
```

```
spearman 0.32268370571748045
```

In [397]:

```
%load_ext tensorboard
%tensorboard --logdir model_albert
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6007 (pid 1609), started 3:30:36 ago. (Use '!kil
l 1609' to kill it.)

<IPython.core.display.Javascript object>
```

Albert Model with convolution1D

In [401]:

```
K.clear_session()
input_1 = Input(shape=(X_train_qt_pooled_output.shape[1],1))
con1D_1 = Conv1D(10, 20, activation='relu',name = 'con1D_1',kernel_initializer=tf.keras.ini
con1D_4 = Conv1D(20, 15, activation='relu',name = 'con1D_4',kernel_initializer=tf.keras.ini
con1D_5 = Conv1D(30, 10, activation='relu',name = 'con1D_5',kernel_initializer=tf.keras.ini
maxpool1 = MaxPooling1D(pool_size=3,name = 'maxpool1')(con1D_5)
qt = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=2

input_2 = Input(shape=(X_train_q_pooled_output.shape[1],1))
con1D_2 = Conv1D(10, 20, activation='relu',name = 'con1D_2',kernel_initializer=tf.keras.ini
con1D_6 = Conv1D(20, 15, activation='relu',name = 'con1D_6',kernel_initializer=tf.keras.ini
con1D_7 = Conv1D(30, 10, activation='relu',name = 'con1D_7',kernel_initializer=tf.keras.ini
maxpool2 = MaxPooling1D(pool_size=3,name = 'maxpool2')(con1D_7)
q = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20

input_3 = Input(shape=(X_train_a_pooled_output.shape[1],1))
con1D_3 = Conv1D(10, 20, activation='relu',name = 'con1D_3',kernel_initializer=tf.keras.ini
con1D_8 = Conv1D(20, 15, activation='relu',name = 'con1D_8',kernel_initializer=tf.keras.ini
con1D_9 = Conv1D(30, 10, activation='relu',name = 'con1D_9',kernel_initializer=tf.keras.ini
maxpool3 = MaxPooling1D(pool_size=3,name = 'maxpool3')(con1D_9)
a = Dense(66,activation = 'relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20

concat = concatenate([qt,q,a])
flat = Flatten()(concat)

output = Dense(30, activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_uni

conv_model = Model(inputs=[input_1, input_2, input_3], outputs=output)

conv_model.summary()
```

```
Model: "model"
__________________________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================================
========================
input_1 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________________________
________________________
input_2 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________________________
________________________
input_3 (InputLayer)            [(None, 768, 1)]     0
__________________________________________________________________________________________
________________________
con1D_1 (Conv1D)                (None, 749, 10)      210         input_1
[0][0]
__________________________________________________________________________________________
________________________
con1D_2 (Conv1D)                (None, 749, 10)      210         input_2
[0][0]
__________________________________________________________________________________________
________________________
con1D_3 (Conv1D)                (None, 749, 10)      210         input_3
[0][0]
```

```
__________________________________________________________________________________________________
________________________
con1D_4 (Conv1D)                (None, 735, 20)      3020        con1D_1
[0][0]
__________________________________________________________________________________________________
________________________
con1D_6 (Conv1D)                (None, 735, 20)      3020        con1D_2
[0][0]
__________________________________________________________________________________________________
________________________
con1D_8 (Conv1D)                (None, 735, 20)      3020        con1D_3
[0][0]
__________________________________________________________________________________________________
________________________
con1D_5 (Conv1D)                (None, 726, 30)      6030        con1D_4
[0][0]
__________________________________________________________________________________________________
________________________
con1D_7 (Conv1D)                (None, 726, 30)      6030        con1D_6
[0][0]
__________________________________________________________________________________________________
________________________
con1D_9 (Conv1D)                (None, 726, 30)      6030        con1D_8
[0][0]
__________________________________________________________________________________________________
________________________
maxpool1 (MaxPooling1D)         (None, 242, 30)      0           con1D_5
[0][0]
__________________________________________________________________________________________________
________________________
maxpool2 (MaxPooling1D)         (None, 242, 30)      0           con1D_7
[0][0]
__________________________________________________________________________________________________
________________________
maxpool3 (MaxPooling1D)         (None, 242, 30)      0           con1D_9
[0][0]
__________________________________________________________________________________________________
________________________
dense (Dense)                   (None, 242, 66)      2046        maxpool1
[0][0]
__________________________________________________________________________________________________
________________________
dense_1 (Dense)                 (None, 242, 66)      2046        maxpool2
[0][0]
__________________________________________________________________________________________________
________________________
dense_2 (Dense)                 (None, 242, 66)      2046        maxpool3
[0][0]
__________________________________________________________________________________________________
________________________
concatenate (Concatenate)       (None, 242, 198)     0           dense[0]
[0]
                                                                 dense_1
[0][0]
                                                                 dense_2
[0][0]
__________________________________________________________________________________________________
________________________
flatten (Flatten)               (None, 47916)        0           concatena
te[0][0]
__________________________________________________________________________________________________
```

```
________________________
dense_3 (Dense)                 (None, 30)           1437510     flatten
[0][0]
==========================================================================
========================
Total params: 1,471,428
Trainable params: 1,471,428
Non-trainable params: 0

__________________________________________________________________________

________________________
```

In [400]:

```
!rm -rf conv_model
```

In [402]:

```
conv_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='binary_crossentropy')
```

In [403]:

```
log_dir_4 = os.path.join('conv_model')

tensorboard_callback4 = tf.keras.callbacks.TensorBoard(log_dir=log_dir_4, histogram_freq=1,
                                                      write_graph=True,write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `
TensorBoard` Callback.
```

In [404]:

```
conv_model.fit(train_data, y_train, batch_size=32, epochs=100, verbose=1,
          validation_data=(cv_data, y_cv),
          callbacks=[SpearmanCallback(validation_data=(cv_data, y_cv),patience=5,
                      model_name='best_weight_bert_4.h5'),tensorboard_callback4])
```

```
Epoch 1/100
152/152 [==============================] - ETA: 0s - loss: 0.4144
bad: 0

val_spearman-corr: 0.2736
152/152 [==============================] - 5s 36ms/step - loss: 0.4144 - v
al_loss: 0.4033
Epoch 2/100
152/152 [==============================] - ETA: 0s - loss: 0.3979
bad: 0

val_spearman-corr: 0.3016
152/152 [==============================] - 5s 32ms/step - loss: 0.3979 - v
al_loss: 0.3989
Epoch 3/100
151/152 [============================>.] - ETA: 0s - loss: 0.3907
bad: 0

val_spearman-corr: 0.3125
152/152 [==============================] - 5s 32ms/step - loss: 0.3906 - v
al_loss: 0.3953
Epoch 4/100
152/152 [==============================] - ETA: 0s - loss: 0.3840
bad: 0

val_spearman-corr: 0.3188
152/152 [==============================] - 5s 32ms/step - loss: 0.3840 - v
al_loss: 0.3933
Epoch 5/100
152/152 [==============================] - ETA: 0s - loss: 0.3771
bad: 0

val_spearman-corr: 0.3219
152/152 [==============================] - 5s 32ms/step - loss: 0.3771 - v
al_loss: 0.3927
Epoch 6/100
151/152 [============================>.] - ETA: 0s - loss: 0.3701
bad: 1

val_spearman-corr: 0.32
152/152 [==============================] - 5s 32ms/step - loss: 0.3702 - v
al_loss: 0.3920
Epoch 7/100
152/152 [==============================] - ETA: 0s - loss: 0.3631
bad: 2

val_spearman-corr: 0.3137
152/152 [==============================] - 5s 32ms/step - loss: 0.3631 - v
al_loss: 0.3946
Epoch 8/100
151/152 [============================>.] - ETA: 0s - loss: 0.3557
bad: 3

val_spearman-corr: 0.3158
```

```
152/152 [==============================] - 5s 31ms/step - loss: 0.3556 - v
al_loss: 0.3989
Epoch 9/100
151/152 [============================>.] - ETA: 0s - loss: 0.3492
bad: 4

val_spearman-corr: 0.3148
152/152 [==============================] - 5s 32ms/step - loss: 0.3493 - v
al_loss: 0.3970
Epoch 10/100
151/152 [============================>.] - ETA: 0s - loss: 0.3420early sto
pping Threshold

bad: 5

val_spearman-corr: 0.313
152/152 [==============================] - 5s 32ms/step - loss: 0.3422 - v
al_loss: 0.4032
```

Out[404]:

```
<tensorflow.python.keras.callbacks.History at 0x7f4338c25828>
```

In [405]:

```
%%time
conv_model.load_weights("best_weight_bert_4.h5")
y_pred_val = conv_model.predict(cv_data)
sp = np.mean([spearmanr(y_cv[:, ind], y_pred_val[:, ind]).correlation for ind in range(y_pr
```

```
CPU times: user 369 ms, sys: 87.7 ms, total: 457 ms
Wall time: 375 ms
```

In [406]:

```
print("spearman {}".format(sp))
```

```
spearman 0.32188323203277386
```

In [407]:

```
%load_ext tensorboard
%tensorboard --logdir conv_model
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
<IPython.core.display.Javascript object>
```

In [2]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model","spearman"]
x.add_row(["base model", 0.28504])
x.add_row(["bert model", 0.34591])
x.add_row(["bert model conv1D", 0.33145])
x.add_row(["Albert model", 0.32268])
x.add_row(["Albert model conv1D", 0.32188])

print(x)
```

```
+---------------------+----------+
|        Model        | spearman |
+---------------------+----------+
|      base model     | 0.28504  |
|      bert model     | 0.34591  |
|  bert model conv1D  | 0.33145  |
|     Albert model    | 0.32268  |
| Albert model conv1D | 0.32188  |
+---------------------+----------+
```

In [ ]:

```
```