Lexical Complexity Prediction

By: Mayank Raj , CWID- 20010945

1. Introduction:

The task of predicting lexical complexity score of a word in a sentence is crucial in NLP applications such as learning text, text simplification and Machine Translation. There are numerous studies that have been designed to calculate the level of word complexity using traditional machine learning and Deep Learning models. Deep learning models have proven to be more accurate than machine learning models for such tasks. Due to the advancements made in the field of transfer learning and pre-trained language models, it is now possible to fine tune a larger transformer-based model for such tasks. In this project I have calculated the score of word complexity between 0-1 in each sentence. I have used two transformer-based models (BERT and RoBERTa) followed by a multilayer neural network for this task.

2. Problem Formulation:

The problem that this project tries to address is the task of predicting lexical complexity score of a given word in a sentence based on a model that is trained on SemEval 2021 dataset. The task is predominantly a regression problem that can be solved using a supervised learning approach. The inputs to the model are a list of sentences and target words with corresponding lexical complexity score and output would be a complexity score between 0 and 1 where a value closer to 0 indicates low lexical complexity.

The sentence is tokenized in a list of sequence sub-words using the BERT tokenizer, which are further encoded into the form of integer-based tensors using the vocabulary of the tokenizer. The input vector stores a dictionary with keys being 'input_ids' and 'attention_mask' where the two tensors are concatenated together to form the input vector. The target vector consists of a list of floating numbers which denotes the corresponding lexical complexity scores which is then converted to a tensor.

The input vector is passed to the weights to two pre-trained transformer-based models namely BERT and RoBERTa separately. The output of the final layers of each of these models are then concatenated and then passed to a series of dense linear layers with appropriate activation functions which generates the desired final output.

3. Method:

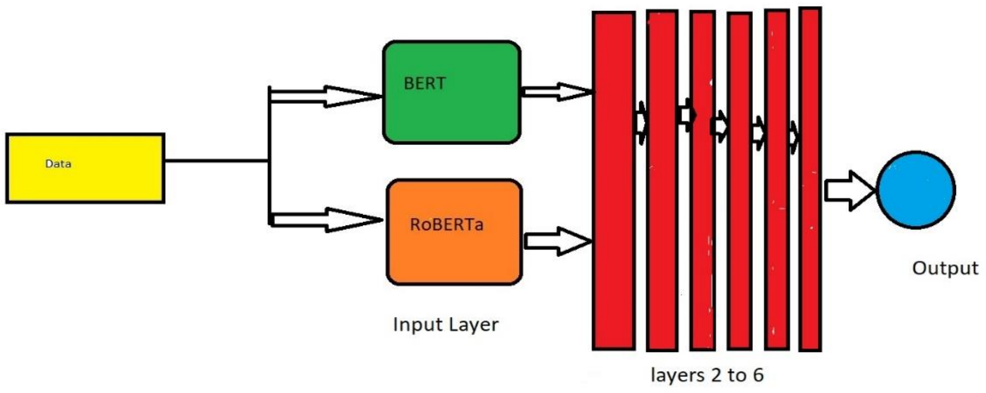
The entire process of training the model can be split into the following parts:

- Data Preparation and Pre-processing: The data was obtained from the official git repository where it was stored in the 'tsv' format. The data was then loaded as a pandas data-frame consisting of 5 fields namely: 'id', 'corpus', 'sentence', 'word' and 'complexity'.

  The sentences in the dataframe then had to be pre-processed. The preprocessing steps was based on the following approaches. These steps were performed using a function which was based on the 'nltk' library:
  - Removing Special characters - Special and non-alphanumeric characters from the sentences were removed using a regex.
  - Expanding Contraction – Short form of commonly used contractions were expanded.
  - Removing Stopwords -- Commonly used English stop-words were removed from the sentences.
  - Lemmatization and Stemming – Lemmatization and Stemming was performed on the words to improve the performance.

Then after preprocessing the dataframe, the sentences and the target words were encoded using the BERT tokenizer and an attention mask was generated. The tokens were padded to ensure that the shape of the input tensor is uniform.

- Model Design:
  Since this was fundamentally a regression problem that dealt with tokens, an attention based neural network was expected to have a better performance. So after carefully reviewing the architectures two pretrained transformer-based models were selected namely **BERT** and **RoBERTa**. The input vectors were separately passed to both these models and then the output of the final layer of these models were concatenated and passed to a series of linear layers as shown below.



Model Architecture

- Training and Inference:

  The loss function used for this model was ‘**mean absolute error’ (MAE)**. **L2** regularization was used to ensure the model deals with bias and variance. The linear layers have ‘**ReLU’** as activation function and the final output layer had linear activation function. The model also had two dropout layers to prevent overfitting. While training the ‘validation loss’ was monitored and early stopping mechanism was implemented using a callback function.

  The optimizer used for this function was **Adam** with learning rate of ‘**2e-4’**. The best set of hyper-parameters were searched manually after trying different combinations. Eventually after the final epoch, the model had a **training loss** of **0.1122** and a **validation loss** of **0.1319.** The loss for test data was 0.13 with ‘mean absolute error’ being **0.09** respectively.

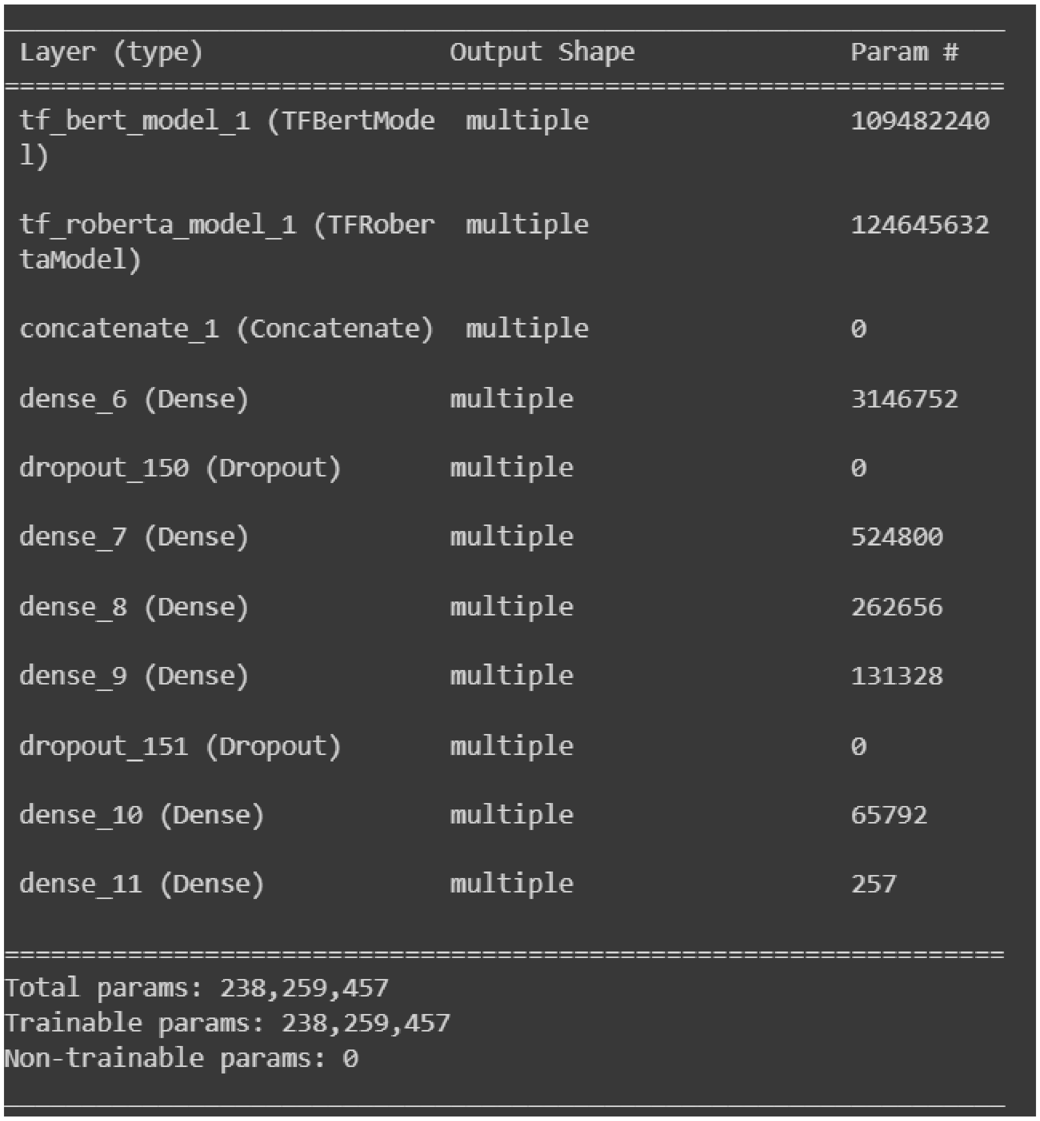
  An inference function was written to check the performance of the model on sample sentences from the test set and results for the same were shown.

4. Experiments:

The dataset which was used had over 7200 sentences and words taken from 3 different sources namely ‘bible’ (containing texts from bible), ‘europarl’ (containing texts from discussions in European Parliament) and ‘biomed’ (containing data from biomedical domain). Since the three texts were from three different contexts, it was essential to use a Transformer based neural network model. BERT and RoBERTa are two such models which have been trained on huge datasets and can be further trained on this dataset for this task. The results from these two models were then concatenated and passed to a series of 6 linear layers. The model architecture is shown below.

Since the dataset contained texts from different domains stemming and lemmatization were necessary steps to improve the performance. These steps were performed using the function based on the ‘nltk’ library.

.

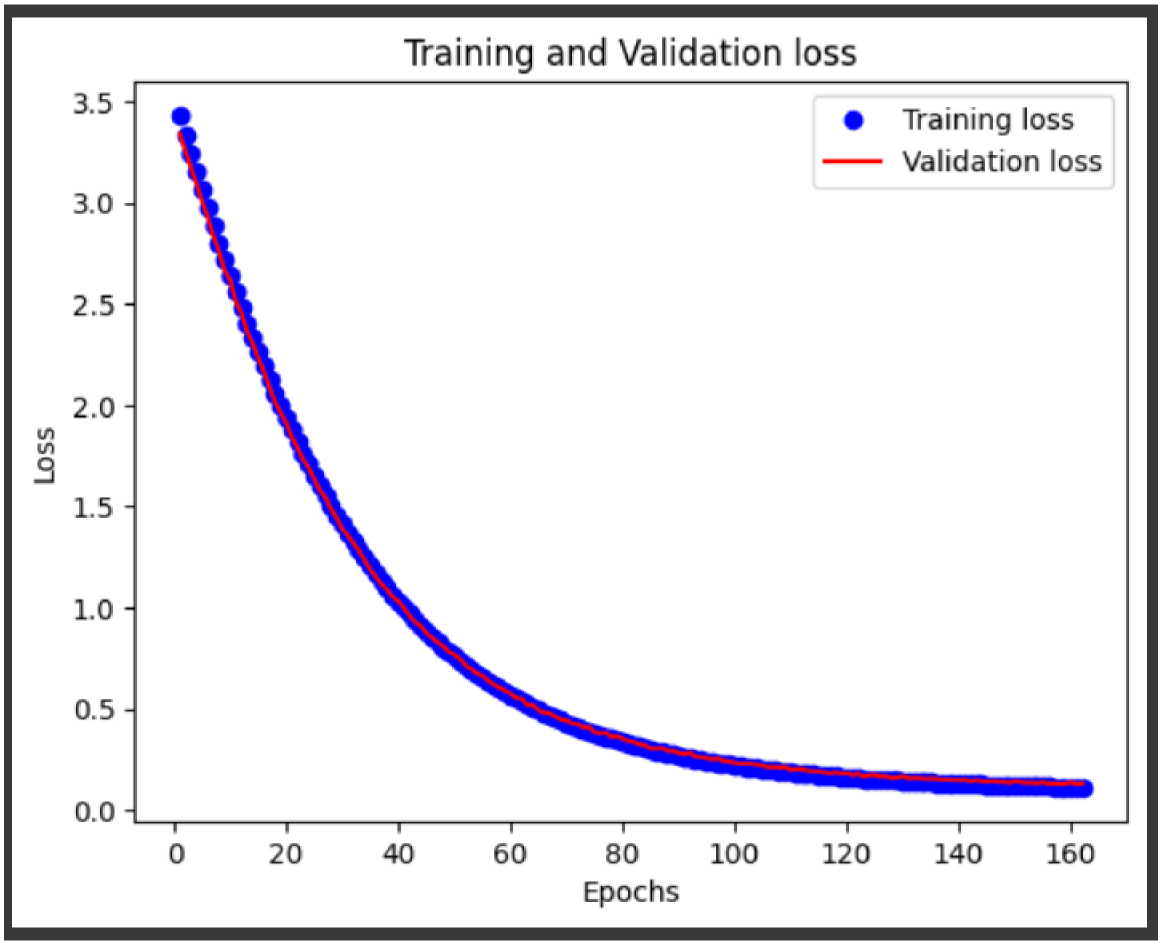


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 tf_bert_model_1 (TFBertMode  multiple                 109482240
 l)

 tf_roberta_model_1 (TFRober  multiple                 124645632
 taModel)

 concatenate_1 (Concatenate)  multiple                 0

 dense_6 (Dense)             multiple                  3146752

 dropout_150 (Dropout)       multiple                  0

 dense_7 (Dense)             multiple                  524800

 dense_8 (Dense)             multiple                  262656

 dense_9 (Dense)             multiple                  131328

 dropout_151 (Dropout)       multiple                  0

 dense_10 (Dense)            multiple                  65792

 dense_11 (Dense)            multiple                  257

=================================================================
Total params: 238,259,457
Trainable params: 238,259,457
Non-trainable params: 0
_________________________________________________________________
```

The model architecture is shown above. The dropout layer were used with a probability of **0.2** and L2 regularization was used with a value of **0.001** to address the concerns of bias and variance in the model.

The model was initially run for 250 epochs with **RMS prop** optimizer with a learning rate of **2e-5** and batch size of 32. The validation loss was monitored through a call back function to avoid overfitting.

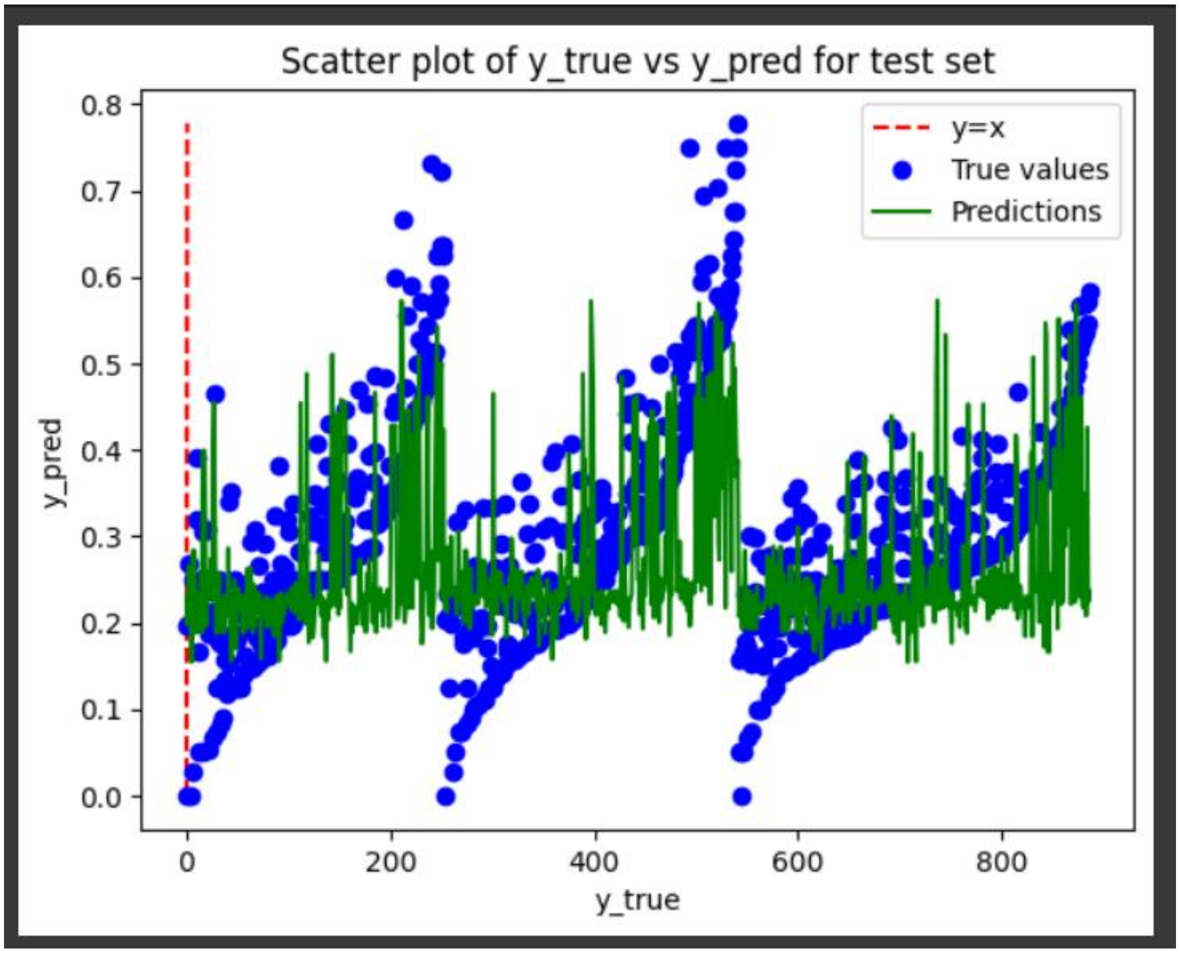
For training the model, the validation split was kept at 0.3 and metrics were mean absolute error and mean square error.



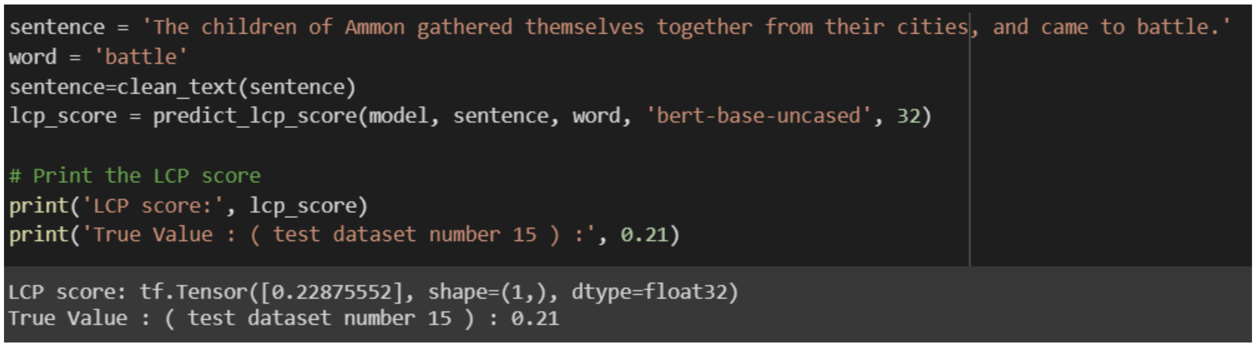
Plot of training and validation loss over the epochs

As observed from the plot of the training and validation loss, a gradual decline in both these metrics were observed throughout the epochs.

Finally the performance of the model was observed over the test data where the loss was reported to be **0.1316** and mean absolute error was **0.0947**.



Plot of true and predicted value for the test data



```
sentence = 'The children of Ammon gathered themselves together from their cities, and came to battle.'
word = 'battle'
sentence=clean_text(sentence)
lcp_score = predict_lcp_score(model, sentence, word, 'bert-base-uncased', 32)

# Print the LCP score
print('LCP score:', lcp_score)
print('True Value : ( test dataset number 15 ) :', 0.21)
```

```
LCP score: tf.Tensor([0.22875552], shape=(1,), dtype=float32)
True Value : ( test dataset number 15 ) : 0.21
```

Output for a sample sentence and word from the test dataset

Eventually the output for 10 randomly selected entries from test data were shown.

5. Conclusion:

Through this model it was demonstrated that transformer-based models (BERT and RoBERTa in this case) can be fine tuned to perform a regression task in the field of NLP. However, the performance can be further improved by using models which are trained on dataset from similar domains, which can then be fine tuned for such a task. The task can also be better performed using an ensemble of similar model after fine tuning them over the train dataset.