Course Project – Task 1: Lexical Complexity Prediction (LCP)

Abhijeet Gusain (20016662)

Task Description

Lexical complexity plays a crucial role in reading comprehension. Predicting lexical complexity accurately can enable a system to better guide a user to an appropriate text, or tailor a text to their needs. NLP systems have been developed to simplify texts for second language learners, native speakers with low literacy levels, and people with reading disabilities.

The report presents my implementation of the given task.

Problem formulation

The problem given is a regression task. Input data contains three corpus set named bible, europarl, and Biomed. Each row has a sentence and a token with a labeled complexity score associated with a token with respect to the sentence. We have to represent to sentence and token into embedding of the same space to calculate the complexity score using regression. I have used an attention-based model for this task. Inspiration for using Bidirectional LSTM (BiLSTM) is from sentimental analysis as we need left-right embedding for predicting sentiment.

Approach:

Feature Selections:

Some of the features which I extracted are

1. Token length: The number of characters in the token

2. Number of Syllables: The number of syllables present in the target word(token).

3. Token position in the sentence

Pre-processing

For pre-processing of sentence and tokens these steps are performed:-

1. Convert to lowercase.
2. Remove stop words but care is taken if the word is a token itself.
3. Lemmatize:
4. Remove punctuations.

Embeddings are created using 300-dimensional Glove Embeddings for tokens and sentences. In the case of an unknown token, the mean of sentence embedding is given to the token embedding

Feature-extracted and pre-processed data are merged together for training purposes.

The training set is split into a train and validation set (80-20 ratio) for the model.

Attention Based Model

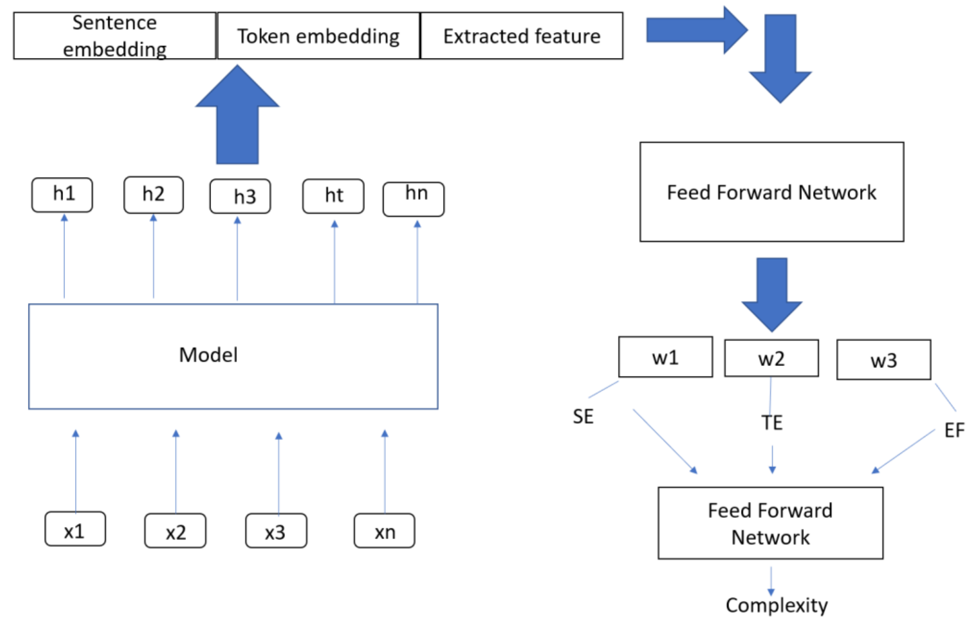
After pre-processed data, we have vectorized representation of words in sentences, tokens embeddings, and extracted features.

I used a bidirectional LSTM model to generate sentence representation and word representation. The start and end index of the word is known so word representation can be extracted from sentence representation.

We have three features sentence embeddings, token embedding, and extracted features. These are passed into MLP (feed-forward network with softmax activation) to get a three-dimensional weight matrix. Final embedding is calculated by multiplying each weight matrix by each of the features and fed to the Feedforward network to predict the complexity score.

*Final embedding = w1 * sentence embedding + w2 * token embedding + w3 * extracted features.*

This final embedding is fed to feed forward network for calculating prediction using the sigmoid activation function.



x1 x2 ,..., xn are words

h1, h2, h3,.. hn are hidden states for Bidirectional LSTM

w1 w2 w3 are weights from the Feed forward network

SE – Sentence Embedding

TE – Token Embedding

EF -- Extracted Features

Hyperparameters and Loss functions

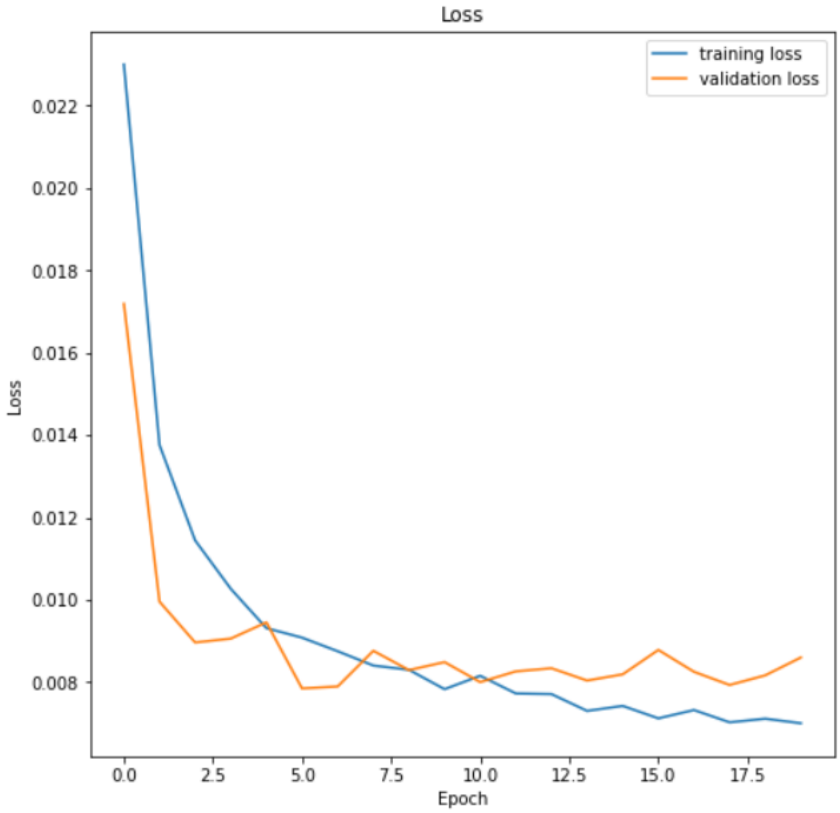
lstm_units = 20
hidden_size = 10
random_seed = 12
embedding_size = 300
dropout rate = 0.4
epochs =20
batch_size = 32
learning_rate = 0.005

The loss function used is Mean Squared Error and the optimizer used is the Adam optimizer

Results

Model	Training MAE	Validation MAE	Prediction MAE	Epoch
Attention Based BiLSTM	0.00698	0.008619	0.07118	20

Loss Curve



Method of Increasing Prediction Score

- Using Bert Based Model with data augmentation: By masking some words from the training sentence(excluding token), we can create more data for training. Then using the Bert-based model will give us a better score.
- Introducing new features like synonyms, hypernyms, word frequency, etc can improve the prediction score.

Conclusion

I showed that using word embedding from Glove Embedding and hand-crafted features we can implement an attention-based BiLSTM model for predicting the complexity score.