STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Report of Topic-1 : Lexical Complexity prediction

Name : Snehal Hosmani CWID : 20012653

1. Introduction

Lexical Complexity is a way to measure how hard it is to understand the context of a given word in any sentence. And this is a necessary measure in order to develop sophisticated state of the art models, which will translate languages or predict the next word in a sentence with high accuracy at a low cost.

It is quite challenging to get the lexical complexity of a word in a sentence as it depends on case to case. Firstly, for a Machine learning model to predict the complexity, we are required to provide it with a historic/annotated data from which it can learn. But the main challenge lies here, A word, for eg., 'discombobulate' which means to 'confuse someone', will definitely be harder to comprehend for someone who is in India or even america, but would be easier for a person with English major to understand. So basically who annotaes, or who decides how complex the word is matters. And once we have these various complexity scores from various people, how do we conclude which is the best value.

This task (1) was officially hosted at SemEval 2021. They provided the participants with an augmented (as in modified) version of CompLex, a multi-domain English dataset with sentences annotated by crowd sourcing using a 5-point Likert scale in 3 domains. The Likert scale has 5 values ranging from 1 to 5 (1 being Very easy and 5 being Very difficult) to comprehend. On the other hand, the data that we work with has complexity values from range 0 to 1 and hence augmented. Using this augmented data, model with neural networks are prepared in this project to predict the lexical complexity of the word in a sentence (0 to 1).

Once we have data like CompLex and we train them on Neural Netowrks, it becomes easier to develop NLP applications which cater to the specific genres. As the data provided is based on 3 genres(Biomed, Euro-parl, Bible), it opens up ways to create many NLP applications like Chatbots or Summarization tool specific to them.

2. Problem Formulation

Task type : Regression using Stacked Neural Networks- As we are predicting the complexity from the range 0 to 1.

The dataset provided has 4 column which are needed to develop the a model:

- Corpus : Source of the sentence and word (here: biomed,euro-parl,bible)

- Sentence
- Token
- Complexity score - range (0 to 1)

From these 4 column, the first three will constitute X/input variable and Y/output variable will be Complexity Score.

- Input: A sentence (str), a word (str), the source (str)
- Output: A score (float)

For example:

Input

Sentence : Behold, there came up out of the river seven cattle, sleek and fat, and they fed in the marsh grass.

Token : River

Source : Bible

Output : A score - 0 (which means it is easier to understand)

This is a Supervised Machine learning task and the the performance of the model developed will be evaluated on the test set using Mean Absolute Error.

$$\text{MAE} = 1/n(\sum_{1}^{n}|y\_test - y\_pred|)$$

3.Method

Data:

```
[ ] lcp_train.describe()
```

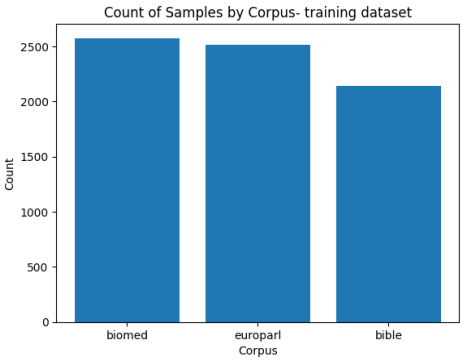
	complexity
count	7232.000000
mean	0.301485
std	0.133091
min	0.000000
25%	0.210526
50%	0.277778
75%	0.371250
max	0.861111

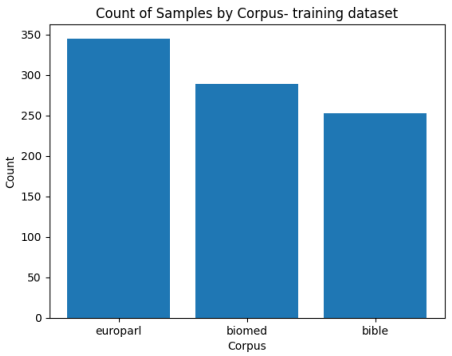
```
[ ] lcp_test.describe()
```

	complexity
count	887.000000
mean	0.298403
std	0.128126
min	0.000000
25%	0.214286
50%	0.277778
75%	0.360243
max	0.777778

We are provided with two tsv.txt files with inputs(sentece,token,corpus) and outputs(complexity) described in the previous section. The training data is of length - 7232 and testing - 887

The distribution of sentences according to genre is :





In both test and train dataset, we have less data w.rt to bible corpus. But overall there is more corpus related to bio-med. And that is the genre where most people face issues in comprehending words.

Pre-processing:

This project deals with non-numerical data, and the first and foremost step to do is pre-processing, that is creating a dictionary, converting into tokens and getting the input_ids which the model can make sense of.

-Sentence and Token:

I used Keras tokenizer to achieve the above. Main things I want to highlight in this part is

- I limited the vocabulary size to be just 10,000, this way the model need not worry about in-frequnt words.
- Calculated the maxim length of Sentence and the word - which will be utilized to give input to the model

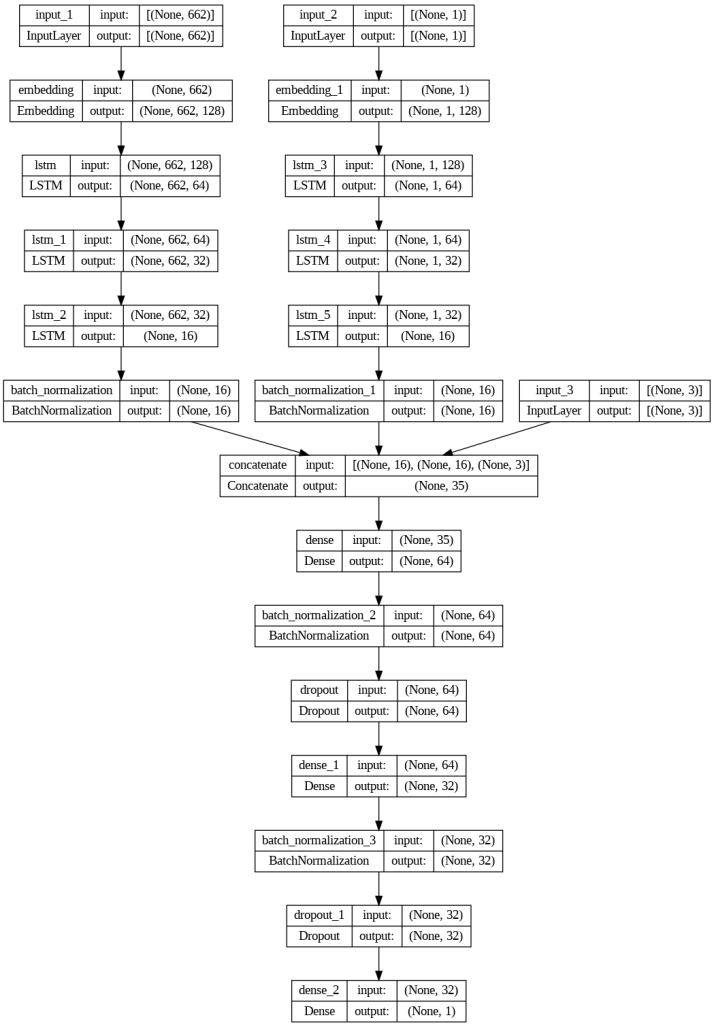
-Corpus

This is a categorical variable as it just has 3 values, for this I just one-hot encoded using keras one-hot encoder.

Model Architecture:

For this project, I developed only 1 model, which constitutes of Stacked LSTM networks with Dropouts and Batchnormalization in between. I chose stacked LSTM for the following reason:

1. As they allow for more complex modeling of sequential data as they make sure that the network captures more complex temporal patterns.
2. The extra layers help to learn higher representations of the input which interim helps to improve model performance.
3. As learnt during lectures, RNN's always face some issues with Vanishing gradient, so making use of Stacked LSTM's can help to tackle that problem.



As per the model plot above, you can see that I have

1. Create embeddings of both Sentence and token.
2. Then stacked 3 LSTM layes each for sentence and token and Batch Normalized them.
3. Next I concatanated the inputs that I received from Step 2 with one-hot encoding of the corpus.
4. Added Dense layers and dropouts until i got a single complexity score.

Methods like Dropout will be useful in this case as the data we have is of length ~7000 only and we don't want to risk over-fitting or under-fitting the model.

Training and Tuning:

I first trained and saved this base model(optimiser=adam,loss='mae') mentioned above and then tuned various parameters using Keras tuner like:

- Optimiser(Adam or RMSprop)
- Dropout value (0.2,0.5,0.7)
- Learning rate(`1e-2, 1e-3, 1e-4`)

As for loss, as it was mentioned to report the MAE score, that is what i monitored along with accuracy of the model at each epoch.

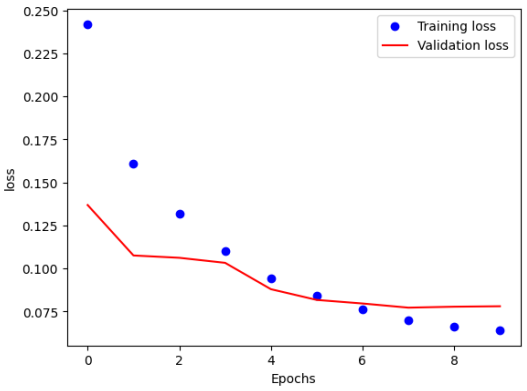
Epochs: To train the base model I set the epoch number to 10 and batch size to 32. But while Tuning I increased the number by setting batch size to 64 and epochs to 30.

Experiments:

Experimental setting:
Where - Google Colab
GPU used - Yes (A100)
Hardware : Allocated by Colab, CPU: 12.7 GB and GPU : 15GB
Software : Python, tensorflow, keras, numpy and pandas

Training and Model Parameters:
As mentioned, I have implemented one model architecture twice, once before tuning and once with tuned parameters. The objective was to evaluate the Mean absolute error on the test set. Below are my results:

Models with parameters	MAE
Base Stacked RNN(Adam(no lr),epochs=10,batchsize=32)	0.1100
Tuned Stacked RNN(RMSprop(lr=0.01),batchsize=64,epochs=30)	0.1072



And the above graph is a plot of base_models epochs vs loss. You can clearly see that validation loss is slightly higher than the training loss as it approcheas the last epoch. This indicated that the model implemented here has neither under-fit or over-fit.

Conclusion & Future Work:

The use of Stacked LSTM's to predict the complexity of a given word has given an MAE score of '0.107'. When I read the papers of people who competed in this task they all used Transformer models and got a MAE score of of around '0.06'. My MAE score is off by '0.04' units. Basically I wanted to implement using Conventional Neural Networks to check by how much will my score fall, as it is already proven that models like Transformers give state of the art results. I think the Stacked LSTM performance was pretty good compared to that of transformers as well. Adding more layers or maybe even using Bi-directinal LSTM's would help to improve the MAE score even further. But ultimaltely it all boils down to efficiency, even if I achieve similar results with LSTM's how efficient will it be when compared to transformers. How well (in terms of resources,efficiency) can both these LSTM's and transformers deal with big data, these are some of the questions that needs further exploration.

References:

1. https://sites.google.com/view/lcpsharedtask2021
2. 'CompLex: A New Corpus for Lexical Complexity Prediction from Likert Scale Data ', https://arxiv.org/pdf/2003.07008.pdf
3. 'OCHADAI-KYOTO at SemEval-2021 Task 1: Enhancing Model Generalization and Robustness for Lexical Complexity Prediction'https://aclanthology.org/2021.semeval-1.2.pdf
4. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer
5. https://keras.io/keras_tuner/