cs60075_team2 at SemEval-2021 Task 1 : Lexical Complexity Prediction using Transformer-based Language Models pre-trained on various text corpora

Abhilash Nandy **Sayantan Adak** **Tanurima Halder** **Sai Mahesh Pokala**
20CS91R01,20CS91P01,20CS92P04,18ME10036
cs60075_team2
{nandyabhilash,sayantanadak.skni,haldertanurima,pokalasaimahesh}@gmail.com
Indian Institute of Technology, Kharagpur, India

Abstract

This paper describes the performance of the team cs60075_team2 at SemEval 2021 Task 1 - Lexical Complexity Prediction. The main contribution of this paper is to fine-tune transformer-based language models pre-trained on several text corpora, some being general (E.g., Wikipedia, BooksCorpus), some being the corpora from which the CompLex Dataset was extracted, and others being from other specific domains such as Finance, Law, etc. We perform ablation studies on selecting the transformer models and how their individual complexity scores are aggregated to get the resulting complexity scores. Our method achieves a best Pearson Correlation of 0.784 in sub-task 1 (single word) and 0.836 in sub-task 2 (multiple word expressions) The link to the GitHub Repository is https://github.com/abhi1nandy2/CS60075-Team-2-Task-1.

1 Individual Contributions

Abhilash Nandy (20CS91R01) - Idea, pre-training and implementation of approach for sub-task 1;
Sayantan Adak (20CS91P01) - Implementation of approach for sub-task 2;
Tanurima Halder (20CS92P04) - Pre-training corpus extraction and cleaning, and implementing baselines for sub-task 1;
Sai Mahesh Pokala (18ME10036) - Implementation of baselines for sub-task 2;

2 Introduction

Complex words hinder the readability of a text, as discussed in (William, 2004). To mitigate this problem, there is a necessity of lexical simplification (Leroy et al., 2013), and predicting the complexity of words is an integral part of this process.

Language Models learn the probability of co-occurrence of words in a corpus. They have been used for various sentence completion and text-based classification tasks. The first language models were n-gram Markov Models (Rabiner and Juang, 1986), which performed well for tasks that did not require very long-range dependencies. Then came RNNs (Cho et al., 2014a), LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014b), which were able to understand longer contexts, but struggled with long paragraphs due to the vanishing gradient problem. Transformers (Vaswani et al., 2017) were a task-agnostic solution that performed better due to the presence of Attention Layers between hidden layers of the neural network, which helped the layers of the neural network to look at the entire input at once. Transformers can perform very well on a broad suite of tasks by fine-tuning on a small number of task-specific samples. The intuition behind using such transformer-based language models for Lexical Complexity Prediction (LCP) was - transformer models pre-trained on different corpora would mimic annotators (of the CompLex Dataset (Shardlow et al., 2020)) having different backgrounds. Since the final score is an aggregation of the annotation scores given by annotators, we aggregate the various scores that are given as outputs by the transformer-based models fine-tuned on the CompLex Dataset.

The rest of the paper is organized as follows. Section 3 gives an overview of our solution approach, Section 4 talks about the corpora used for pre-training and the dataset used for fine-tuning for the LCP task, Section 5 discusses the experimental settings, baselines used, and a comparison and analysis of the results and Section 6 gives a conclusion.

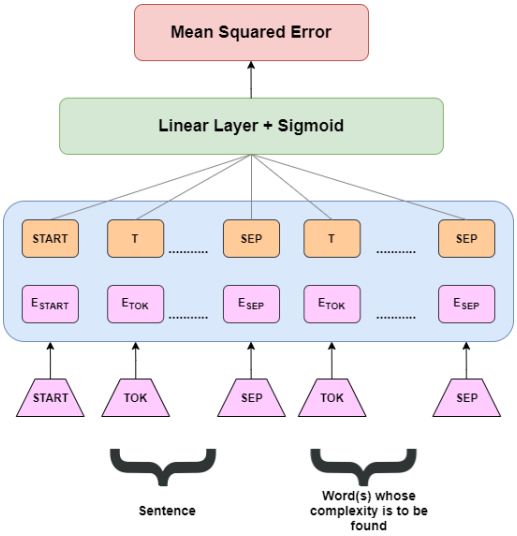

Figure 1: General Block Diagram of the Transformer

3 Solution Overview

3.1 Model Architecture

We use several transformer models. The general block diagram of such a model is shown in Fig. 1. The input to a model is the tokenized form of a sentence, and the tokenized form of the word/multi-word expression whose complexity score is to be predicted (separated by special tokens), and the target output is the complexity score. Each model consists of a transformer encoder, having the architecture of either BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), followed by a linear layer and a sigmoid activation layer so that the output is squashed in the range $(0, 1)$. Sigmoid Activation Function is applied, as the target complexity score is a value between 0 and 1. To compute loss for backpropagation, the mean squared error loss function is used, as the problem is, as such, a regression problem.

3.2 Pre-training the transformer on text corpora

In order to initialize the weights and the embeddings of the transformer encoder, it is pre-trained on large text corpora so that it has syntactic, lexical and semantic knowledge before fine-tuning on the task-specific data. This is done for two reasons - (1) To increase the rate of convergence towards the lexical complexity prediction task (2) To mimic an annotator from a particular background.

In order to pre-train a transformer, specific pre-training tasks are performed. If the transformer being used is RoBERTa, Masked Language Modelling (MLM) is performed, where 15% of all the tokens are randomly replaced by a $< MASK >$ token. Such a masked sentence is provided as input to the transformer language model, and a Softmax Layer activation Function is applied for the output corresponding to the masked token to find out the probabilities of various tokens in the vocabulary being in the place of the $< MASK >$ in the original, unmasked sentence. The target is the actual token that was masked. A cross-entropy loss function is used to calculate the loss that is backpropagated. In the case of the BERT Transformer, in addition to the MLM pre-training task, Next Sentence Prediction (NSP) Task is also performed. Two sentences are taken from the corpus, where either one sentence follows the other, or the two sentences are far apart. The output is either 1 corresponding to the sentences being adjacent to each other, and 0 being the case when they are far apart. Both the cases have the same number of samples while training. The output corresponding to the $START$ (here, $< CLS >$) token is passed through a linear layer to get a 2x1 shaped vector, which is then followed by a Softmax Layer, thus giving probabilities of whether the second sentence comes after the first one or not.

4 Data

4.1 Data used for pre-training

Since we require several transformer language models pre-trained on a wide variety of corpora, we make it a point that we have transformers pre-trained on text corpora from which the CompLex Dataset has been extracted. These corpora are - (1) World English Bible Translation (Christodouloupoulos and Steedman, 2015) (We used the data found in this link [1]) (2) English part of the Europian Parliament Proceedings from europarl (Koehn, 2005) (3) CRAFT corpus (Bada et al., 2012) of bio-medical domain. We pre-train three RoBERTa language models on these three corpora (initialized by weights from (Liu et al., 2019)) using the MLM pre-training task.

4.2 Data used for fine-tuning

For fine-tuning, we do not use any external data other than the datasets that have been provided for both the sub-tasks [2].

[1] https://www.kaggle.com/oswinrh/bible
[2] https://github.com/MMU-TDMLab/CompLex

5 Experiments and Results

5.1 Transformer Language Models used

We use the predictions from 9 transformer-based language models, 4 of which have a RoBERTa encoder, and the other 5 have a BERT-based encoder. 2 models are pre-trained on general domain corpora like Wikipedia and BooksCorpus, 2 models on biomedical and clinical data, 2 models on Europarl data, 1 on Bible, 1 on Financial data, and 1 on scientific papers. Also, 6 of the pre-trained transformer models were publicly available in the HuggingFace Models Catalog [3], while the other 3 were pre-trained by us on the three datasets from which CompLex Dataset is extracted, as mentioned in Section 4.1.

5.2 Training, validation and Test Sets

For each sub-task, the training and the test sets are the same as those provided for the competition. The trial data given for each sub-task is taken to be the validation data.

5.3 Hyperparameters

For pre-training, the RoBERTa transformer language model, a batch size of 16 is used and is trained up to 1 epoch. The rest of the parameters are the same as in (Liu et al., 2019).

When fine-tuning, irrespective of whether the model has a RoBERTa or a BERT Transformer encoder, the input sequence length is set to 256, with padding or truncation, as is the case. A learning rate of 2×10^{-5} is used with a batch size of 32, and a Weighted Adam Optimizer is used. The network is fine-tuned for 4 epochs. The Pearson Correlation on the validation data is calculated for every epoch, and the checkpoint giving the best Pearson Correlation is regarded as the best checkpoint, which would later be used for predicting outputs on the test data.

5.4 Methods of Aggregation used

In order to aggregate the complexity scores of a particular combination of models, we use the following two strategies - sample-wise average and sample-wise maximum across all transformer models. We then do the same across all permutations, see which combination gives the best test results and report it as the final result.

[3] https://huggingface.co/models

5.5 Baselines

We use XGBoost (Chen and Guestrin, 2016) to perform a boosting-based regression model, with an objective of squared error and other default parameters and hyperparameters over a set of features. The different baselines use different feature sets, which are as follows -

1. **xgb-A** - word length (sum of word lengths in the case of MWE), number of syllables and word frequency (from various text sources) (Speer et al., 2018) (average of word frequencies in the case of MWE) of the word/expression whose complexity is to be found, and the type of corpus of the sentence (either Bible, Europarl, or Biomedical).

2. **xgb-B** - Concatenation of features of xgb-A and the 50 and 100-dimensional GloVe (Pennington et al., 2014) word vectors of the word/expression whose complexity is to be found. For the expression, the sum of the GloVe Vectors of the individual words would be taken.

3. **xgb-C** - Concatenation of features of xgb-B and the probabilities of the word/expression whose complexity is to be found given the sentence with that word/expression that is masked, where the probabilities are predicted by different transformer-based masked language models pre-trained on different corpora.

   Note: The probability of the word/token given the masked sentence is approximated as the product of the probabilities of predicting each token, given other tokens of the sentence are masked. E.g., Given a sentence S - "I just love mowing the lawn with a lawn mower." Let's say one is required to find out the complexity of the expression - "lawn mower". First, 'lawn' is masked in S, and the probability to predict 'lawn' using the transformer model M is found, denoted by $P1$. Similarly, 'mower' is masked in S, and the probability to predict 'mower' using M is found, denoted by $P2$. Resultant feature value = $P1 * P2$

5.6 Results and Discussion

Table 1 compares the Pearson Correlations (higher the better) and mean squared errors (lower the better) of our best (according to Pearson Correlation)

	Single word		MWE	
APPROACH	**PC**	**MSE**	**PC**	**MSE**
xgb-A	0.718	0.0078	0.762	0.0103
xgb-B	0.741	0.0073	0.815	0.0083
xgb-C	0.744	0.0072	0.817	0.0082*
BERT-BASE-UNCASED	0.765	0.007*	0.791	0.009
BIBLE+EUROPARL+BIOMED (AVG.)	0.753	0.0075	0.798	0.0096
BIBLE+EUROPARL+BIOMED (MAX.)	0.751	0.0076	0.788	0.0092
BEST COMBINATION (AVG.)	**0.784**	**0.0066**	**0.836**	**0.0078**
BEST COMBINATION (MAX.)	0.774*	0.0071	0.819*	0.0091

Table 1: Comparing the Pearson Correlation (PC) and Mean Squared Error (MSE) of our methods and the baselines (The entries in **bold** are the best performing according to the respective column's metrics, while the ones with a * are the next best ones.)

aggregate results (for both average as well as maximum aggregation), some ablations, and the baselines for both the sub-tasks.

Based on the results, we can infer the following

-

1. **xgb-B** performs better than **xgb-A**, suggesting that, GloVe Word Vector features perform a vital role in complexity prediction, as they contain some contextual information regarding the word.

2. **xgb-C** performs the best among the baselines, as it also considers the probabilities of predicting the masked tokens whose complexity is found, adding to the contextual information.

3. Fine-tuning **BERT-BASE-UNCASED** transformer model for the LCP task performs better than the best baseline in case of sub-task 1, which could be attributed to the reason that, fine-tuning attention-based transformer models captures even more contextual information than the baselines.

4. Fine-tuning and aggregating RoBERTa Transformer models pre-trained on the three corpora from which the CompLex Dataset was extracted (**BIBLE+EUROPARL+BIOMED**), still gives better results than the baselines (except for **xgb-B** and **xgb-C** in case of sub-task 2), but performs inferior as compared to **BERT-BASE-UNCASED** model for single word sub-task, while performing almost similar in case of Multi-Word Expressions sub-task. Also, the average aggregation performs better than the maximum aggregation.

5. The combination of transformer models that gives the best results upon aggregation (**BEST COMBINATION**), consists of 3-4 different transformer models fine-tuned on the dataset, suggesting that, **transformer models pre-trained on domains related as well as unrelated to the dataset (such as Financial Data, Legal Data), are able to best mimic annotators coming from various backgrounds.** Even in this case, average aggregation performs better than maximum aggregation.

6. If we consider the evaluation metrics of Pearson Correlation (PC) and Mean Squared Error (MSE), it can be seen (especially in the single-word sub-task) that they are negatively correlated, as is expected.

6 Conclusion

We show that aggregating the results of various fine-tuned transformer models pre-trained on various corpora from different domains gives high Pearson Correlation and low mean squared errors compared to individual transformers and regression models using attributes such as hand-crafted features, word embeddings, transformer-based language model prediction probabilities, etc. This shows that transformer-based language models, each pre-trained on a different text corpus, can better imitate annotators of the dataset, who come from diverse backgrounds and prior knowledge.

References

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):1–20.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Gondy Leroy, David Kauchak, and Obay Mouradi. 2013. A user-study measuring the effects of lexical simplification and coherence enhancement on perceived and actual text difficulty. *International journal of medical informatics*, 82(8):717–730.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Lawrence Rabiner and Biinghwang Juang. 1986. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. Complex: A new corpus for lexical complexity predicition from likert scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*.

Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. Luminosoinsight/wordfreq: v2.2.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

HD William. 2004. The principles of readability. eric. *Online Submission*.

7 Submission Screenshots

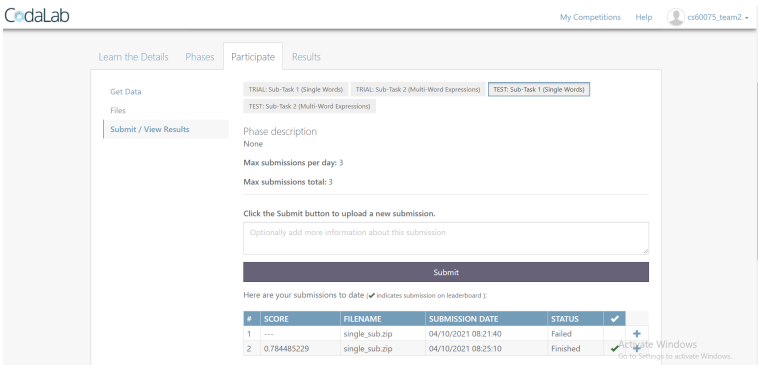

Figure 2: Sub-task 1 submission

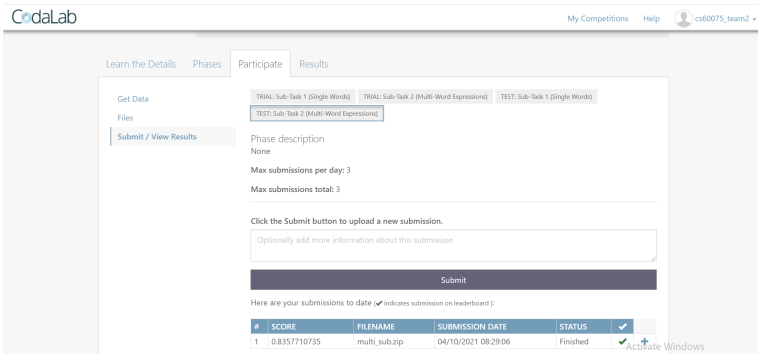

Figure 3: Sub-task 2 submission