Adapting Large Pretrained Language Models for Grammar Error Correction

Stav Yosef (ID. 316298876)

Submitted as final project report for the NLP course, IDC, 2022

1 Introduction

In today digital world, hundreds of millions of humans are writing and communicating in English on daily basis, big part of those people English is not their native language, writing fluently without any mistake is almost mission impossible and, some of them don't have confidence even to write one paragraph. Hence, the need for grammar error correction is rising drastically.
In this paper we propose how to fine-tune state of the art (SOTA) transformer [1], called T5 developed by Google [2], on grammatical error correction (GEC) corpus named JFLEG [3] which is a gold standard benchmark for developing and evaluating GEC systems. We have experimented 3 variations of T5, small, base, and, large. Each variation we tried different hyperparameters such as number of epochs, and, learning rate.

1.1 Related Works

Grammar error correction goes into two mainly channels, one, rule-based which is relies on set of grammar rules [4] and the second is statistics-based which are statistical models, for example, n-gram, and deep learning models [5]. The rule-based method has some advantages, grammar rules can be explained easily, rules can be added, edited, and deleted. The main disadvantage on such systems that it requires a professional expertise to create, watch the rules.
Most of the advantages and disadvantages in statistical model system are opposite to rule-based system. There is no longer need for professional expertise but to explain any rule becomes harder, especially in deep learning models. Advantages of statistical models that the training can be quick and better accuracy can be achieved with more compute power and data.

2 Background

2.1 Tokenization

Tokenization is the process of splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. Theses tokens could be words, numbers or punctuation marks.

2.1.1 T5 Tokenizer, Encoder and Decoder

The T5 family does not work with strings directly. Instead, it needs to tokenize (encode) the input text and works with integers. The model learns the statistical relationships among these tokens (integers) and excels at producing the next token in a sequence of tokens. The T5 tokenizer uses Byte Pair Encoding (BPE) [6], which is a method to address the Out Of Vocabulary (OOV) problem. BPE is a word segmentation algorithm that merges the most frequently occurring character or a sequence of characters. This tokenizer has a vocabulary size of 32,100 tokens, the T5 tokenizer takes a string and returns list of integers (Figure: 1). A helpful rule of thumb is that one token generally corresponds to $\sim$ 4 characters of text for common English text. This translates to roughly $\frac{3}{4}$ of a word (so 100 tokens $\sim=$ 75 words) .

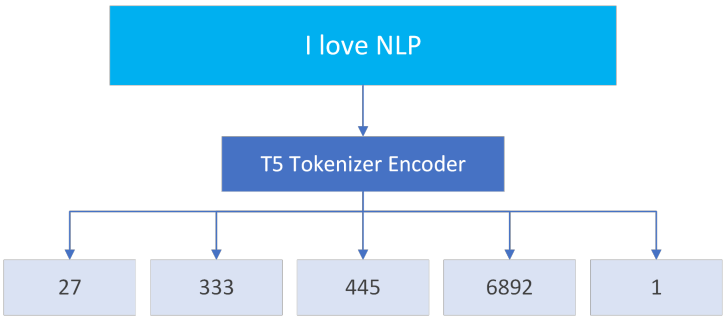

Figure 1: Encoding using T5 tokenizer.

The reverse procedure to encoding is referred to as decoding. T5 tokenizer, like most transformers, produces as an output a list of numbers known as tokens, which are then decoded to a list of strings (Figure: 2).

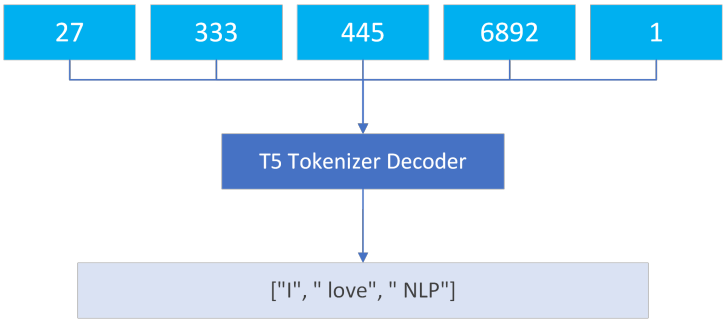

Figure 2: T5 tokenizer decoding example.

2.2 Attention

There have been several types of deep neural network architectures adapted for processing sequence data such as text, and currently the most popular approach is attention [1]. Suppose the following sentence is given as an input: "The boy didn't jump into the pool because he was scared.", what does the word "he" refers to? Our brain is able to automatically associate the word 'he' to the word "boy", and this is the role of the attention mechanism. There is extention to attention mechanism called Multi-Head Attention [1] that allows the model to learn information from different representation subspaces (Figure: 4).

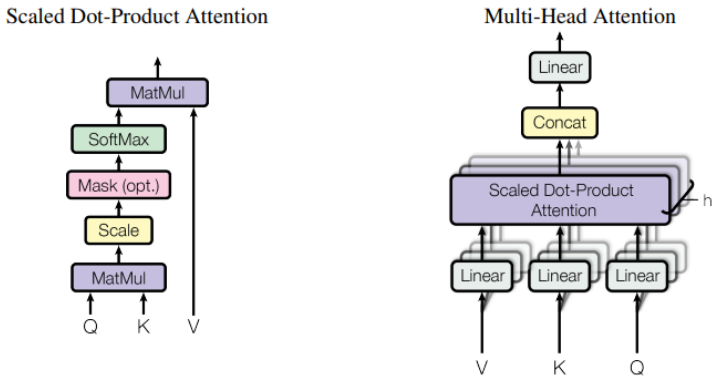

Figure 3: (Left) Scaled dot-product Attention. (Right) Multi-Head Attention consists of several attention layers running in parallel. From [1].

3 Solution

3.1 General approach

T5, or **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer, is a Transformer based architecture that uses a text-to-text approach. Every task including question answering, classification, translation, is cast as feeding the model text as input and training it to generate some target text. This allows for the use of the same model across our diverse set of tasks. Hence we can fine-tune this model to perform grammar error correction task.

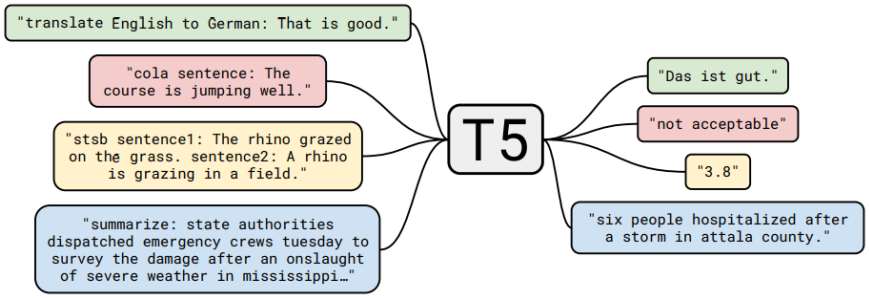

Figure 4: A diagram of our **t**ext-**t**o-**t**ext framework. Every task we consider including translation, question answering, and classification is cast as feeding our model text as input and training it to generate some target text. From [2]

Model	Parameters	Layers	d_{model}	n_{heads}	d_{ff}
Small	60M	6	512	8	2,048
Base	220M	12	768	12	3,072
Large	770M	24	1,024	16	4,096
3B	3B	24	1,024	32	16,384
11B	11B	24	1,024	128	65,536

Table 1: Architecture hyperparameters for the five model sizes. where d_{model} is the model dimension, n_{heads} is the number of heads. "The **f**eed **f**orward networks in each block consist of a dense layer with an output dimensionality d_{ff}" From [2].

We chose to generate predictions with beam search [7] over random sampling, top-k sampling, greedy, and etc. because beam search tends to select the best probable response which fits best for GEC problem.
JFLEG dataset contains 1511 examples in total, each example contains one sentence with grammar mistake and 4 corresponding corrected versions for the sentence. Our approach to solve GEC problem is by creating a training file from JFLEG dataset with explicit directions for GEC, the way we chose to do it is by prefix each incorrect sentence with "grammar: ". The idea behind this approach is that model's encoders will encode the incorrect sentence with the

prefix and the model's decoders will automatically decode the original sentence but without the mistakes. Example for the input: "grammar: So I think we can not live if old people could not find siences and tecnologies and they did not developped . " and the output can be four versions:

1. "So I think we would not be alive if our ancestors did not develop sciences and technologies . "
2. "So I think we could not live if older people did not develop science and technologies . "
3. "So I think we can not live if old people could not find science and technologies and they did not develop . "
4. "So I think we can not live if old people can not find the science and technology that has not been developed . "

3.2 Design

The code was written with Python 3.9.7 on PyCharm 2021.2.3. Our code based on PyTorch [8] and Transformers [9], each experiment took one minute up to one hour depends on the size of the model and number of epochs. The experiments run with one RTX3090 24GB, we faced technical difficulty when we tried to run T5 3b/11b and the program exceeds the GPU's memory and collapsed, in future work we need to try to solve it using BFLOAT16 [10] or with a framework like Microsoft's DeepSpeed [11] in order to reach T5 11b and bigger models in the future.

4 Experimental results

We split JFLEG dataset into 50% training set and 50% test set, 754 and 747 rows respectively. Each row is split into 4 examples, each sentence has 4 corresponding corrected versions, then, $|train_{set}| = 754 \cdot 4 = 3012$ samples and, $|test_{set}| = 747 \cdot 4 = 2988$. Our loss function is cross entropy which is great metric to measure the differences between two probability distributions, in GEC there isn't one close an answer, in fact there are a lot of correct options in different variations that the model can produce, hence cross entropy is standing out from other standard metrics. We run 27 experiments over 3 models' sizes, small, base and large as mentioned in Table 1, three epochs variations (1/2/3) and three learning rate variations, 5e-5, 5e-6 and, 5e-7. All three pretrained models, small, base, and large achieved same loss score, 1.269, 1.259, and 1.186 respectively. Each model predicted 13 sentences we chose to test out, for example the incorrect sentence: "The candidate promised not to rise taxes when elected ." and the correct sentence is: "The candidate promised not to **raise** taxes when elected ." We saw that T5-small models are performing very bad, almost without any correct answers. The results are not surprising, the average loss of T5-small models is 1.318, slightly different from pretrained version. Base

models are performing much better with average loss of 0.869 and large models are even better reaching loss of 0.414. In Table 2 there are all 27 experiments results, and in Figure 5 there is a graph that indicating healthy learning, T5-base, 2 epochs, and 5e-05 learning rate.

Model	Epochs	Learning rate	Loss	Runtime (seconds)
Small	1	5e-5	1.0104	11.9225
Small	1	5e-6	1.3531	11.6199
Small	1	5e-7	1.4805	11.463
Small	2	5e-5	0.5063	21.0984
Small	2	5e-6	0.9274	21.6269
Small	2	5e-7	1.1023	20.7302
Small	3	5e-5	1.3091	30.9812
Small	3	5e-6	1.7798	29.7692
Small	3	5e-7	2.0871	30.8042
Base	1	5e-5	0.5918	32.0862
Base	1	5e-6	0.933	30.3537
Base	1	5e-7	1.1581	29.7625
Base	2	5e-5	0.3273	60.9004
Base	2	5e-6	0.4873	62.1003
Base	2	5e-7	0.7831	60.5575
Base	3	5e-5	0.7972	88.9488
Base	3	5e-6	1.1595	91.5607
Base	3	5e-7	1.5829	87.6032
Large	1	5e-5	0.2096	885.8571
Large	1	5e-6	0.2188	893.971
Large	1	5e-7	0.8861	872.4621
Large	2	5e-5	0.0255	1778.2573
Large	2	5e-6	0.2161	1812.9781
Large	2	5e-7	0.3535	1921.8794
Large	3	5e-5	0.3982	3068.3926
Large	3	5e-6	0.5262	2777.7578
Large	3	5e-7	0.8918	2943.8923

Table 2: All 27 experiments results.

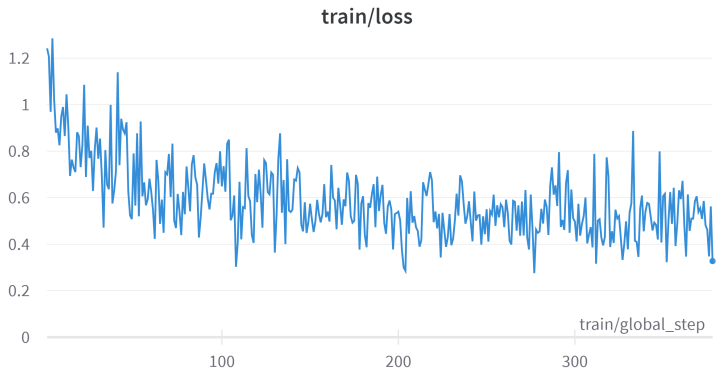

Figure 5: Loss graph through global training steps.

5 Discussion

Grammar error correction is becoming more and more relevant these days, COVID pandemic created a new wave of people that need to use English on their personal and business life, working online, ecommerce and etc. There are great services that provide us with GEC such as Microsoft's word, Grammarly and more. We found that GEC problem is not fully solved yet and probably have long way to go as the language is evolving on daily basis, the bigger the model the better results but it's coming with a tradeoff, small models can be train faster due to less parameters and batch size can be increase and vice versa large models takes longer, in our experiments we can see difference of x100 longer runtime (in Table 2) and that's translating directly to money. We found also that an approach like ours can solve the mistake, but the solution can change big part of the sentence even if the mistake is small like an order of 2 characters. Commercial solution must be a hybrid solution, rules-based and statistics-based systems. A big challenge in producing a deep neural network model is that it can't be explain easily and we had to do a lot of trial and errors experiments, for example, two similar learning rates can be huge different in the results. Transformers are great, surprising, and powerful, no expert in specific field can producing in short period of time an NLP system that's works well. We can't wait to see where Transformers will be in couple of years!

6 Code

https://github.com/stav95/CourseNaturalLanguageProcessing/tree/master/FinalProject

References

[1] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[2] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[3] Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. "JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 229–234. URL: https://aclanthology.org/E17-2037.

[4] Grigori Sidorov et al. "Rule-based System for Automatic Grammar Correction Using Syntactic N-grams for English Language Learning (L2)". In: (Aug. 2013), pp. 96–101.

[5] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. "LM-Critic: Language Models for Unsupervised Grammatical Error Correction". In: *CoRR* abs/2109.06822 (2021). arXiv: 2109.06822. URL: https://arxiv.org/abs/2109.06822.

[6] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *CoRR* abs/1508.07909 (2015). arXiv: 1508.07909. URL: http://arxiv.org/abs/1508.07909.

[7] Carnegie-Mellon University.Computer Science Dept. "Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University." In: (Apr. 2015). DOI: 10.1184/R1/6609821.v1. URL: https://kilthub.cmu.edu/articles/journal_contribution/Speech_understanding_systems_summary_of_results_of_the_five-year_research_effort_at_Carnegie-Mellon_University_/6609821.

[8] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: (2019). Ed. by H. Wallach et al., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[9] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: (Oct. 2020), pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[10] Dhiraj D. Kalamkar et al. "A Study of BFLOAT16 for Deep Learning Training". In: *CoRR* abs/1905.12322 (2019). arXiv: 1905.12322. URL: http://arxiv.org/abs/1905.12322.

[11] Samyam Rajbhandari et al. "ZeRO: Memory Optimization Towards Training A Trillion Parameter Models". In: *CoRR* abs/1910.02054 (2019). arXiv: 1910.02054. URL: http://arxiv.org/abs/1910.02054.