Analyzing Student Writing Elements with Transfer Learning and Pre-trained Longformer

Justin Cheung, Air Singh, Adam Vogel, Phillip Yu

March 2022

Justin Cheung Contribution:

Wrote Transformers/Self-Attention and Longformer v.s. RoBERTa sections

Air Singh Contribution:

Wrote Transfer Learning section and Citations.

Adam Vogel Contribution:

Wrote Tokenizers and Masking sections.

Phillip Yu Contribution:

Developed the code base to fine-tune and predict with the open sourced pre-trained Longformer in PyTorch. Wrote Introduction, Methodology, Global/Local Attention, Implementation Details, and Results/Interpretations sections.

1 Introduction

We demonstrate transfer learning by analyzing writing elements from students grade 6 to 12 with a Pre-trained Longformer (Transformer variant). The task is to identify writing structures in essays, and we also conduct a literature review on similar Transformer variants. We find importance in this application as writing is a critical skill for success. "However, less than a third of high school seniors are proficient writers. Low-income, Black and Hispanic students fare even worse, with less than 15 percent demonstrating writing proficiency, according to the National Assessment of Educational Progress." Our model attempts to classify different paragraph structures to help students learn writing.

2 Methodology

The dataset is found on Kaggle, and it contains essays from U.S. students in grades 6 to 12, annotated by experts for elements commonly found in argumentative writing. The annotations include sentences that are classified as lead, position, claim, counterclaim, rebuttal, evidence, and concluding statements. These essays were written as part of the national and state standardized writing assessments from 2010 to 2020. We perform transfer learning with a Longformer architecture to classify sentences in our student essays into the outlined statements. Details on the model's architecture are in the sections below.

3 Tokenizers

Our input data is comprised of sentences. In order to use the transformer to make predictions based on these sentences, we must separate each sentence into their constituent components, which are individual words and elements of punctuation. Tokenizers allow us to take sentences as input, and output the individual words which comprise the sentence. Then, these tokens can be used as inputs for the transformer model.

4 Transformers and Self-Attention

In recent years, transformer-based models have been successful in solving many problems in NLP because of the attention mechanism. Applications include machine translation and sentiment analysis. In contrast, bidirectional LSTM/GRU may be unable to capture longer dependencies due to gradual loss of memory at various gates. Now, some of the technical details of transformers are highlighted below. See figure 1.

4.1 Embedding Layers and Sinusoidal Positional Encoding

This is to preprocess our input and output words. Transformers use learned word embeddings, i.e. iteratively update the weight matrix to maximise word probabilities during training. Also, since sequences are processed as a whole, we use $(PE_{pos,2dim}, PE_{pos,2dim+1}) = (\sin pos/10000^{2dim/dim\_model}, \cos pos/10000^{2dim/dim\_model})$ to encode word positions. Intuitively, it is an analogue to bitwise representation. This effectively preserves relative order, since the property of sines and cosines allows us to transform $\vec{PE}_{pos}$ to $\vec{PE}_{pos+k}$ with a linear map.

4.2 Encoders and Decoders

There are 6 stacks of identical layers in both. For encoder, there are two sub-layers: self-attention mechanism and feed forward neural network (FFNN). While for decoder, there are three: masked self-attention, encoder-decoder attention and FFNN. Residual connection, where residuals being zero matrices are much easier to learn than an identity mapping, is used to speed up convergence together with layer normalization.

4.3 Attention

An attention function takes query, key and value ($Q = YW_q, K = XW_k, V = XW_v$) as input. First, QK^T is computed to measure similarity in input and output matrices X,Y. It is then divided by $\sqrt{dim_k}$ to avoid vanishing gradient. It is then softmaxed and multiplied with V, such that words with high similarity with others get higher score. So, self-attention is a special case where X = Y, and a masked one ensures the word prediction depends only up to current information. Finally, multi-head attention is employed to allow joint attention under different representations that improves accuracy, and also efficient parallelized computation.

5 Global and Local Attention

To understand the Longformer variant we dive into the differences between global and local attention. At a high-level, global attention assumes that we compute attention across all tokens of our input sequence; whereas, in local attention we only consider a subset of tokens.

The primary drawback of global attention is the computational cost of attending to all words in the input sequence. Our essays range from less than 50 words to over 1,300 words, and the longer essays become computationally expensive and impractical. The local attention mechanism alleviates the some of the computational cost by selectively focusing on a small window or subset of the input sequence. This reduces the computational complexity from scaling quadratically with sequence length $O(n^2)$ to a mechanism that scales linearly with sequence length $O(n)$.

In more detail, the Longformer self-attention reduces the memory and time complexity of the query-key matrix multiplication operation from $O(n_s \times n_s)$ to $O(n_s \times w)$ where n_s is the sequence length and w is the average window size through a sliding window approach. With multiple stacked layers of local attention the architecture has the ability to build representations across the entire input, similar to CNNs. Ultimately, this gives Longformer the advantage of token classification for long sequences in our dataset of essays from students grades 6 to 12.

6 Masking

In the Transformer model we have implemented, masking allows the model to avoid 'cheating'. In our decoder, there exists several multi head attention blocks which produce attention scores on individual sets of queuries, keys, and value parameters (Q, K, V) in N different ways. Masking allows each attention block to preserve self-attention causality. In other words, the mask works by forcing the attention block to only take into account tokens at previous positions, while 'masking' tokens in future positions. This allows for attention calculations to be uncontaminated by information leaking from tokens in future positions.

For example, in the sentence "I have a car", which is comprised of four tokens, we can designate five values of X, where x_1 corresponds to the start of the sentence, and the remaining four values $(x_2, ..., x_5)$ correspond to the remaining tokens in the sentence. If our decoder wishes to output predictions for y_3 (which is predicting the next token in the sentence, x_4) we should only take into account those tokens x_1, x_2, x_3, since taking into account x_4 and x_5 will give the model future information about the sentence, which contaminates the prediction. This is most commonly accomplished by assigning weights for masked tokens values of zero, so that if we begin with five weights $W_1, ..., W_5$, we set W_4 and W_5 to zero in order to predict y_3.

7 Longformer v.s. RoBERTa

The baseline model of Longformer is a RoBERTa based model. To put it simply, a BERT model is a transformer that has two features in pretraining – Masked Language Modelling (predicting words of [MASK], 12% of randomly masked tokens) and Next Sentence Prediction (determining whether the second segment in an input sequence is a subsequent segment in the original document, see 3). It can then be finetuned to handle tasks such as Question Answering and Named Entity Recognition.

There are several differences between RoBERTa and BERT. For RoBERTa, the masking is done dynamically during training instead of preprocessing, such that masked words are chosen differently each time. In addition, RoBERTa is pretrained with blocks of contiguous full sentences from one or more documents, instead of word segments, without the NSP task. Byte Pair Encoding, a data compression technique, and higher training batch size are also used to optimize performance.

Longformer introduces the attention pattern to the pretrained RoBERTa model, as described in section 5. Dilation is added to two heads of the top multi-head attention layers. And, a higher w is used in top layers for learning higher-level context first. Moreover, to increase the maximum input length from 512 to 4096, Longformer minimally modifies the position encoding by stacking those produced by RoBERTa 8 times. Longformer outperforms state-of-the-art models on long documents in various tasks, like Document Classification and Coreference Resolution, while maintaining fewer parameters and memory requirements.

8 Transfer Learning

Transfer learning is when parts of a pre-trained model are used again in a new machine learning model. If the two models have to do similar things, then they can share knowledge. Transfer learning is a technique used in training models. The knowledge from training a past model is reused in a new task. The new task can be more specialized than the generalized case of pre-trained model.

Transfer learning means that every new task does not need to be trained from scratch. It allows for a model to be fine tuned rather than being trained from nothing. Training a model requires a lot of time and resources especially with labeling large data sets. By using transfer learning, it saves time and resources. With transfer learning, a model can be trained on a data set that is already labeled and transfer that learning onto a new, unlabeled data set.

9 Implementation Details

Given our chosen Longformer architecture, the specific implementation details are as follows. First, We download the open sourced pre-trained Longformer by Hugging Face, stored in a directory for further fine-tuning. We pre-process our training data of student essays to include five cross validation folds. This allows us to get an estimate of the test error by training on four folds and testing on one. We then tokenize our essays and assign a BIO (Beginning-Inside-Out) tag for each token along with their respective labels. Ultimately, this gives us five cross validation folds with 15 different labels on all our tokens.

After pre-processing our data, we use PyTorch to batch our training samples for model ingestion. We chose a batch size of two and a maximum token length of 1024, primarily due to memory constraints. For each cross validation we run six epochs, chosen due to time constraints, as each epoch takes takes an hour. We use three different learning rates decreasing across the six epochs. The five cross validation folds took approximately 30 hours with the entire training and testing process on a single NVIDA Quadro RTX 4000.

10 Results & Interpretation

To assess the model's performance we use a macro F1 score. For each sample, all ground truths and predictions for a given class are compared. If the overlap between the ground truth and prediction is greater than or equal to 0.5 than the prediction is considered a true positive. Any ground truths without a prediction is considered a false negative and any predictions without a ground truth are false positives. The final score is arrived by taking the macro F1-score across all classes in which we achieved 0.6136. Our results are shown in the graphs and tables section.

We should note that sentences labeled as rebuttal, counter-claim, and claim achieved the lowest F1 score. This is probably due to the difficulty of discriminating between the three classes. The subtleties of language make these three classes very difficult for a machine to distinguish as "counter-claim" counters "claim" and "rebuttal" in turn counters "counter-claim." It may be possible that our model is confused by these nuances of language, mistakenly classifying "counter-claim" as "claim" and "rebuttal" as "counter-claim."

We should also note that sentences labeled as lead and conclusion achieved the highest F1 score. This is probably due to the distinguishable "land-marks" of these sentences as the lead is nearly always at the start and the conclusion is at the end. Conclusion statements in this dataset also frequently start with "In conclusion," which is another factor that makes it easily distinguishable. The higher F1 scores of these labels are expected and understandable.

For this task we find that the Longformer provides a more advantageous architecture compared to a vanilla Transformer based on the different self-attention layers. The sliding window local attention in a Longformer enables larger input sequences, which is required for this analysis as we are classifying paragraph structures in student essays. Ultimately, this application of NLP is important as the open-sourced models will help students in lower income communities increase their writing proficiency.

Appendix: Graphs & Tables

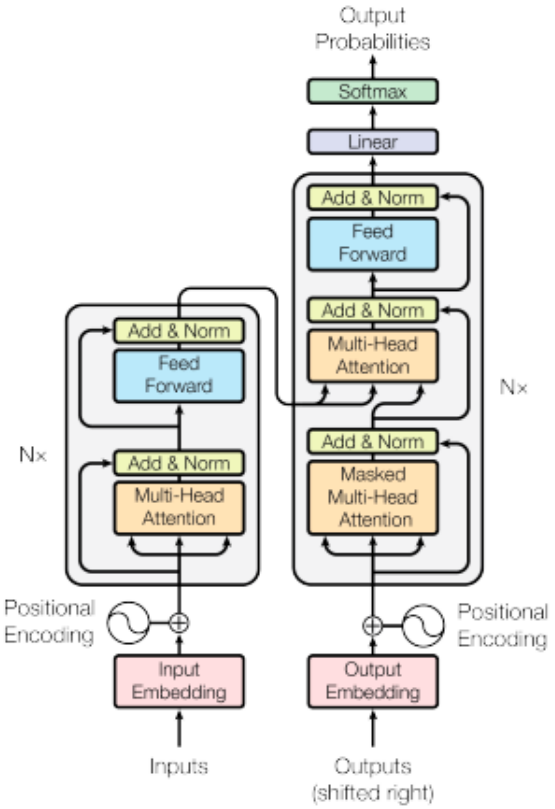


Figure 1: Architecture of a Transformer Model [15]

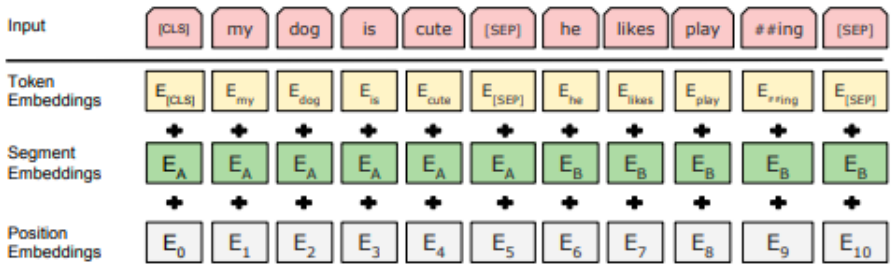
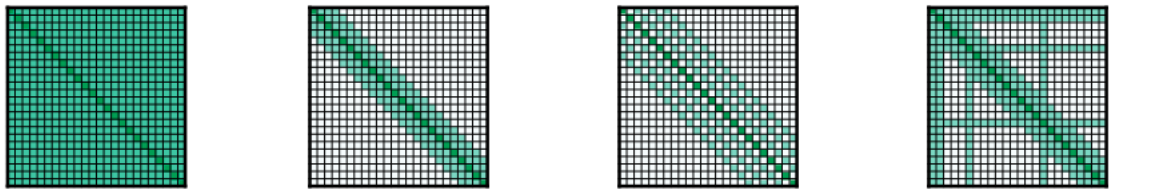


Figure 2: Input preprocessing of Next Sentence Prediction[3]



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 3: Different Attention Patterns in Longformer v.s. full attention[2]

Class	F1 Score
Lead	0.7816
Claim	0.5188
Evidence	0.6596
Counter-Claim	0.4931
Rebuttal	0.4038
Conclusion	0.7768
Position	0.6617
Overall	**0.6136**

Table 1: F1 Score by each section and overall macro F1 Score

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014). URL: https://arxiv.org/abs/1409.0473.

[2] Iz Beltagy, Matthew E Peters, and Arman Cohan. "Longformer: The long-document transformer". In: *arXiv preprint arXiv:2004.05150* (2020). URL: https://arxiv.org/abs/2004.05150.

[3] Jacob Devlin et al. *Bert: Pre-training of deep bidirectional Transformers for language understanding*. May 2019. URL: https://arxiv.org/abs/1810.04805?source=post_page.

[4] Dharti Dhami. *Learning word embeddings*. Dec. 2018. URL: https://medium.com/@dhartidhami/learning-word-embeddings-9f15533645b3.

[5] Rani Horev. *Bert explained: State of the art language model for NLP*. Nov. 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[6] Amirhossein Kazemnejad. *Transformer Architecture: The Positional Encoding*. 2019. URL: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/.

[7] Ria Kulshrestha. *Transformers*. Nov. 2020. URL: https://towardsdatascience.com/transformers-89034557de14.

[8] Yinhan Liu et al. *Roberta: A robustly optimized Bert pretraining approach*. July 2019. URL: https://arxiv.org/abs/1907.11692?_hsenc=p2ANqtz--dDpDrgqk_2fUksm5kxzhEIKRBlCGPx3dXqFFnJaMUxwqi1R8BCOQNQw3i7YkqFwG1Ujqj.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015). URL: https://arxiv.org/abs/1508.04025.

[10] Tomas Mikolov et al. *Efficient estimation of word representations in vector space*. Sept. 2013. URL: https://arxiv.org/abs/1301.3781.

[11] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural machine translation of rare words with Subword units*. June 2016. URL: https://arxiv.org/abs/1508.07909v5.

[12] Aastha Singh. *Evolving with bert: Introduction to Roberta*. July 2021. URL: https://medium.com/analytics-vidhya/evolving-with-bert-introduction-to-roberta-5174ec0e7c82.

[13] Matt Trotter. *Transfer learning for machine learning*. June 2021. URL: https://www.seldon.io/transfer-learning/.

[14] Jakob Uszkoreit. *Transformer: A novel neural network architecture for language understanding*. Aug. 2017. URL: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html.

[15] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017). URL: https://arxiv.org/abs/1706.03762.

[16] Felix Wu et al. "Pay less attention with lightweight and dynamic convolutions". In: *arXiv preprint arXiv:1901.10430* (2019). URL: https://arxiv.org/abs/1901.10430.

[17] Fuzhen Zhuang et al. "A comprehensive survey on transfer learning". In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76. URL: https://arxiv.org/abs/1911.02685.