Evaluating Student Writing

Apoorva Surendra Malemath
Northeastern University
malemath.a@northeastern.edu

Sravya Burugu
Northeastern University
burugu.s@northeastern.edu

1 Introduction

Writing has always been one of the most critical skills for success, as it the way to express ones thoughts and ideas. Even when it comes to school work, writing allows a student to express their understanding and individuality through it. And schools ask the students to submit their work in the form of essays because it allows the teacher to check if the student has understood the concepts and can gauge the student's critical thinking. However, according to a survey less than one third of high school seniors are proficient writers. Thus, one way to solve this problem and improve student's writing would be to design an automated feedback tool to evaluate student writing and provide personalized feedback.

There are various automated writing feedback technologies available at present, but each has its own set of limitations. Many tools either fail to identify writing patterns in essays, such as thesis statements and evidence for claims, or do so insufficiently. Moreover, the vast majority of the tools provided are proprietary, with algorithms and feature claims that cannot be independently verified. Furthermore, many of these writing tools are out of reach for few schools due to their high cost. In short, the subject of automated writing feedback is ready for innovation, which could aid in the democratization of education.

Thus, we aim to identify elements in student writing i.e. we segment text and classify argumentative and rhetorical elements i.e. predict human annotations in essays written by 6th-12th grade students.The projects will encompass a combination of Deep learning and NLP based models. The scope of the project is to make it easier for students to receive feedback on their writing and increase opportunities to improve writing outcomes. Virtual writing tutors and automated writing systems can leverage this project, and teachers may use them to reduce grading time. This project will allow any educational organization to better help young writers develop. This problem statement is hosted by Georgia State University on Kaggle. [1]

2 Related Work

Attention Is All You Need [2] explains how the transformers process an input sequence of words all at once, and they map relevant dependencies between words regardless of how far apart the words appear in the text. Thus, Transformers are highly parallelizable, and we can train large models at a faster rate, and use contextual clues to solve a lot of ambiguity issues that plague text.

BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding [3] is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

Language Models are Few-Shot Learners [4] models can perform various NLP tasks like question answering, textual entailment, text summarization etc. without any supervised training. These language models need very few to no examples to understand the tasks and perform equivalent

or even better than the state-of-the-art models trained in supervised fashion. The paper further explains how large language models develop pattern recognition and other skills using the text data they are trained on. Language models can recognize patterns in data which help them minimize the loss for language modeling task. When presented with few examples and/or a description of what it needs to do, the language models matches the pattern of the examples with what it had learnt in past for similar data and uses that knowledge to perform the tasks. This is a powerful capability of large language models which increases with the increase in the number of parameters of the model.

Attention-based Neural Text Segmentation [5] presents a supervised neural approach for text segmentation. Specifically, it talks about attention-based bidirectional LSTM model where sentence embeddings are learned using CNNs and the segments are predicted based on contextual information. Also, this model can automatically handle variable sized context information.

RoBERTa: A Robustly Optimized BERT Pretraining Approach [6] is a replication study of BERT pretraining that carefully measures the impact of many key hyperparameters and training data size. The issue with BERT is that, it is significantly undertrained. RoBERT for pretraining NLP systems is an improvement on BERT. Also RoBERT performs optimized training and takes lesser time during pre-training. RoBERT is built on language masking strategy used in BERT, wherein the system learns to predict intentionally hidden sections of text within which otherwise are unannotated language examples. It has modified key hyperparameters, and removes the next-sentence pretraining objective, and performs training with much larger mini-batches and learning rates.

Transformer-based models cannot process long sequences due to their self-attention operation, which scales quadratically with the sequence length. Longformer: The Long-Document Transformer [7] with an attention address this issue. The attention mechanism used in LongFormer scales linearly with sequence length. The long document transformer, accepts wider inputs unto 4096 tokens. Longformer's attention mechanism uses a drop-in replacement for the standard self-attention and combines a local windowed attention with a task motivated global attention. It employs self attention on both local and global context. LongFormer outperforms RoBERTa.

3 Dataset Overview

The dataset contains argumentative essays written by U.S students in grades 6-12. The essays were annotated by expert raters for elements commonly found in argumentative writing. Each text segment can be classified in either of the 7 classes i.e.

1. Lead
2. Position
3. Claim
4. Counterclaim
5. Rebuttal
6. Evidence
7. Concluding

The dataset consists of 1084 files i.e. each file has the contents of one essay. There exists supporting information i.e. the annotation for all the essays in the training set, and it consists of the attributes as mentioned in Table 1.

Table 1: Annotation Attributes

id	ID code for essay response
discourseId	ID code for discourse element
discourseStart	character position where discourse element begins in the essay response
discourseEnd	character position where discourse element ends in the essay response
discourseText	text of discourse element
discourseType	classification of discourse element
discourseTypeNum	enumerated class label of discourse element
predictionString	the word indices of the training sample, as required for predictions

From the above mentioned attributes from the annotation file there is a unique id assigned to each essay. There will be multiple discourse for the same essay. A discourse represents a segment in the essay, each discourse will have the start position and end position which represents where the segment begins and ends in the essay and the text representing the segment. discourse type is defined for each segment i.e. the class that the segment belongs to. There can be multiple instances of the same class in the essay i.e. for instance there can be multiple claims in the same essay.

4 Method

As mentioned in the above section, the dataset consists of an annotation file which comprises multiple discourse for each essay. Thus, to feed the data to the network we apply transformations. We first group the annotations by discourseType, discourseStart, discourseEnd and predictionString. Then we merge them along with raw text such that, one input now will now correspond to one essay. Further, we apply correction technique on the discourseType, which consist of the labels of the text segments i.e. we wish to differentiate the occurrences of the label. For instance, if in an essay there are 3 claims then originally we have 3 labels i.e. [claim, claim, claim], we now rename it to [claim1, claim2, claim3] so that they have distinct labels. The same process flow can be seen from the figure below.

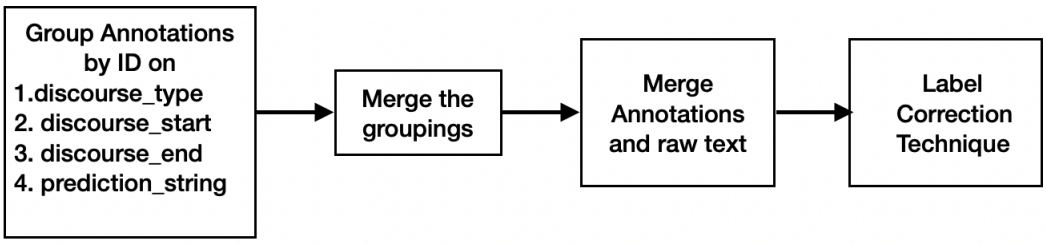

Figure 1: Prepocessing the raw data

Post pre-processing, we tokenize the data, and apply the data collection technique from the transformer using DataCollatorForTokenClassification. Data collectors form a batch by using a list of dataset elements as input. DataCollatorForTokenClassification dynamically pads the inputs and labels. We then apply the AutoModelForTokenClassification pre-trained model.

Our project follows the below pipeline,

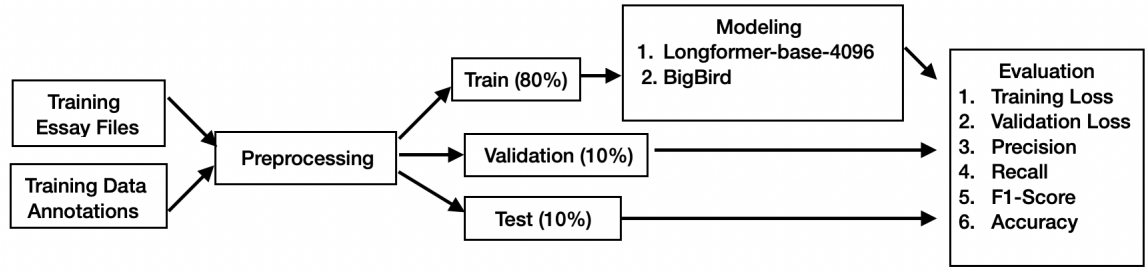

Figure 2: Modelling and Evaluation Pipleline

4.0.1 Longformer: The Long-Document Transformer

Our model is constructed using longformer-base-4096, this is a BERT-like model started from the RoBERTa and pretrained for long documents. It supports sequences of length up to 4096. Longformer uses a combination of a sliding window (local) attention and global attention. It is a transformer-based architecture that reformulates the self-attention computation to reduce the model complexity. [7]

- Longformer Self Attention : It applies self attention in both local and global context. And it takes individual values for query, key, and value for both local and global attention. The local attention is mainly used to build contexual representation, and the global attention helps build the full sequence representations for prediction. The local attention alone is not enough to learn task specific representation, thus we also compute global attention.
- Longformer Tokenizer : It is derived from RoBERTa tokenizer and uses byte-level Byte-Pair encoding.
- Longformer Model for token classification: It is a linear layer on top of of the hidden-states output, which is used for Named-Entity-Recognition task. It uses word embeds, position embedding and token type embedding.

We further discuss more on how attention is calculated:

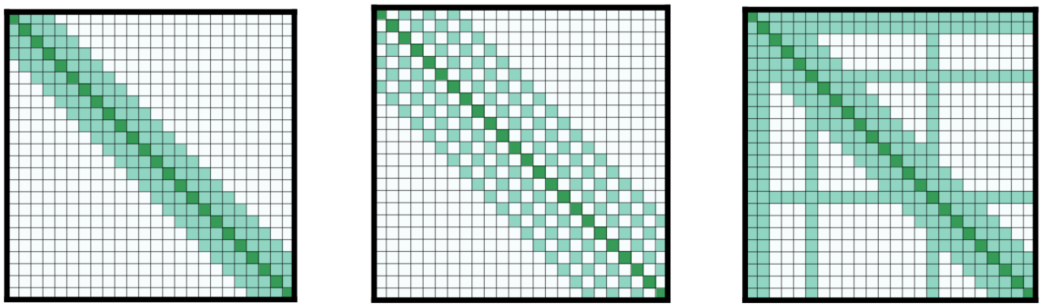

Figure 3: (a) Sliding Window (b) Dilated Sliding Window (c) Global Attention and Dilated Sliding Window Attention

- **Sliding Window:** Longformer uses fixed size window attention for each token. Large receptive fields are formed by stacking the multiple layers of the windowed attention. For fixed window size w, each token will consider $\frac{1}{2}w$ tokens on both the sides. Thus, the computation time is $O(n \times w)$, where n is length of the input sequence i.e. it scales linearly with input sequence length. The receptive field would be $l \times w$ where l is the number of layers in the transformer. This allows longformer to cover the entirety of the sequence.

- **Dilated Sliding Window:** In order to increase the receptive field further without increasing computation, the sliding window is dilated. Then the receptive field would be $l \times w \times d$, where d is the dialation. From the above equation, we can reach tens of thousands of tokens even for small value of d.
- **Global Attention:** A token for which global attention is calculated attends to all the tokens across the sequence, and all tokens in the sequence attend to it.
- **Linear Projections for Global Attention:** Given the linear projections Q, K, V, a tranformer computes attention scores as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

  Longformer uses 2 sets of projects one for local attention and for global attention.

The reason we use Longformer over traditional transformers is that, they can process long sequences. The traditional transformer based models cannot process long sequences due to the way self-attention is computed. It scales quadratically with sequence length. In order to address this issue, Longformer uses an attention mechanism that scales linearly with respect to the sequence length. Thus, it makes it easy for the Longformer to process documents with thousands of tokens. The attention mechanism is a replacement of the standard self-attention which combines both local windowed attention and global attention. This self-attention component enables the network to capture contextual information from the entire sequence.

Longformer outperforms RoBERTa on long document tasks. It uses multiple layers of attention to build contextual representation. Thus, does not require task specific architecture.

4.0.2 BigBird: Transformers for Longer Sequences

BigBird [8] runs on a sparse attention mechanism that allows it to overcome the quadratic dependency of BERT while preserving the properties of full-attention models. The sparse attention mechanism enables it to process sequences of length up to 8x more than what was possible with BERT. Attention mechanism is applied token by token, unlike BERT where the attention mechanism is applied to the entire input just once.

BERT works on a full self-attention mechanism. This leads to a quadratic growth of the computational and memory requirements for every new input token. The maximum input size is around 512 tokens which means this model cannot be used for larger inputs for tasks like large document summarization. This means that a large string has to be broken into smaller segments before applying them as input. This content fragmentation also causes a significant loss of context which makes its application limited. Thus, we use BigBird as it overcomes the issues observed in BERT when used for long documents.

- BigBird Model makes use of word embeddings, position embeddings and token type embeddings.
- BigBird consists of three important parts i.e.
  1. A set of global tokens g attend on all the parts of the sequence.
  2. All tokens attend to a set of local neighbors tokens w.
  3. All tokens attend to a set of random tokens r.

  Thus, this gives raise to high performing attention mechanism which can scale long sequence length i.e. up to $8x$. [9]
- BigBird relies on block sparse attention instead of normal attention as used in BERT. And it is more effecient than BERT.

We choose transformers such as LongFormer and BigBird over other BERT like models because they cannot process such long documents. Firstly, as discussed above the time require to compute self-attention would be high. Alternatively, to use BERT like models we can shorten the document up to 512 token which is the token limit for BERT like models. However, partitioning the data like that could result in loss of important cross-partition information.

In addition to using the pre-trained model and tokenizing the data, the preprocessing of the dataset and understanding how to use the data with the models was one of the crucial learning of the project. We tested our model with different tokenizers such as BERT, RoBERTa, RoBERTa-Large. We also observed that the classes are imbalanced, however directly balancing the classes might not be the best way to tackle this issue as we are dealing with the structure of the essay. Say every essay has multiple claims, then definitely total number of segments labeled as claim will be more. However, as this is the expected structure of the essay we cannot directly treat the imbalance. Instead we also tested the the models by dropping the essays which have more than 5 occurrences of the same class, we assume such essays to be outliers. We found 2858 essays to be outliers in our case. The model performance here did not significantly increase.

5 Experiments

5.1 Exploratory Data analysis using Annotations

Exploratory Data Analysis using the annotations will help reveal the dataset's underlying structure as well as trends and patterns that are not immediately apparent. It will also assist in deriving trustworthy conclusions from a large amount of data by carefully and deliberately looking at it via an analytical lens.

The below figure, we can see the distribution of the discourseType in the train dataset, it is observed that 'claim' has the highest count, followed by 'evidence'. And 'Rebutal' has the least count.

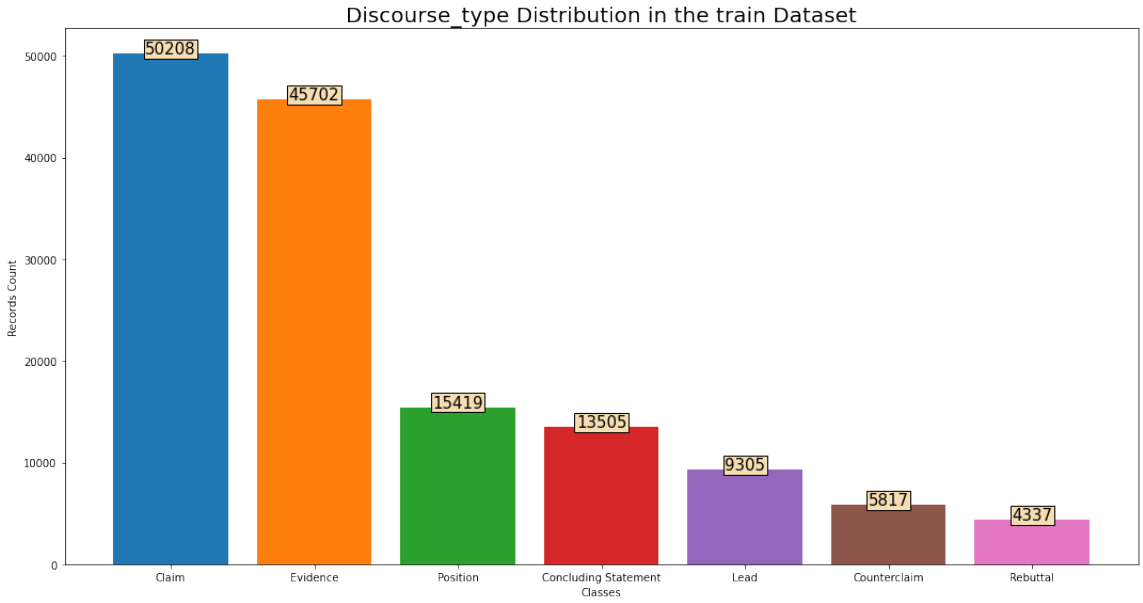

Figure 4: DiscourseType Distribution in the training data

From Fig 1, it can be noted that that most of the data is concentrated around Claim and Evidence class, causing class imbalance. Thus, this imbalance may lead to poor performance of the model.

To further understand the high values observed in Fig 1, we apply correction technique to the annotation files, as each easy can have multiple instances of the same class and we aim to capture the number of times the class has appeared in the same essay. For instance, if in an essay there are 3 claims then originally we have 3 labels i.e. [claim, claim, claim], we now rename it to [claim1, claim2, claim3].

We then plot and observe the distribution of the correct labels, the same can be seen in the figure below.

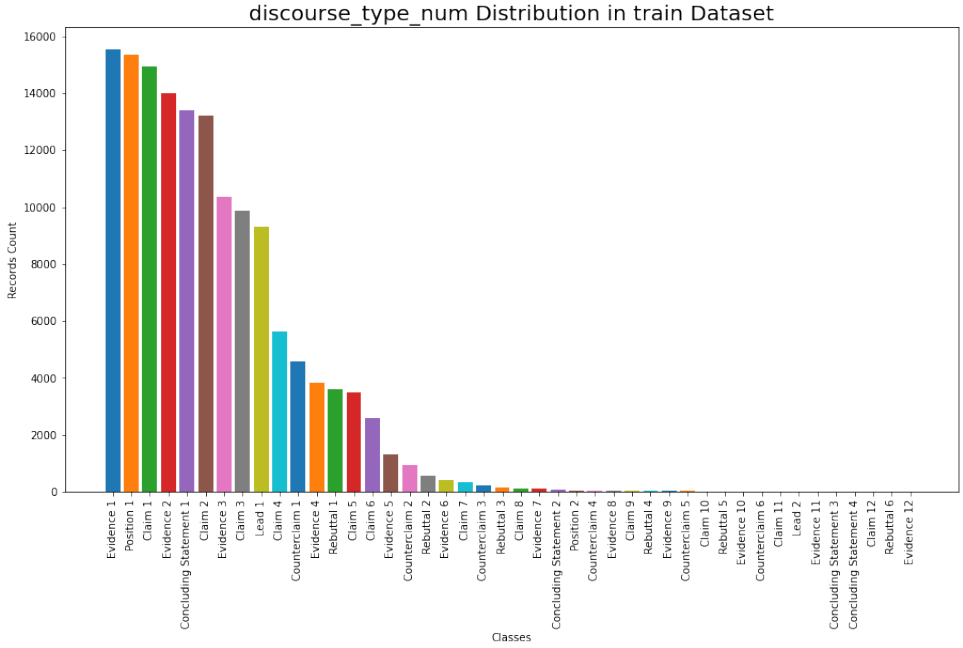

Figure 5: DiscourseTypeNum Distribution in the training data

From Fig 2, we observe the distribution of the corrected labels, and we note that there are up to 12 claims and 12 evidences in individual essay. As multiple instances of claims and evidences are present in each essay the total count for claim and evidence was very high. Also, not all classes are present in each essay. Thus, the classes are not balanced.

We further wish to understand the Text Length Frequency in a File. In the below Fig 3, it is clear that highest frequency is for the text files of length around 2000. Also, there exists fewer files with longer text length of around 6000-7000.

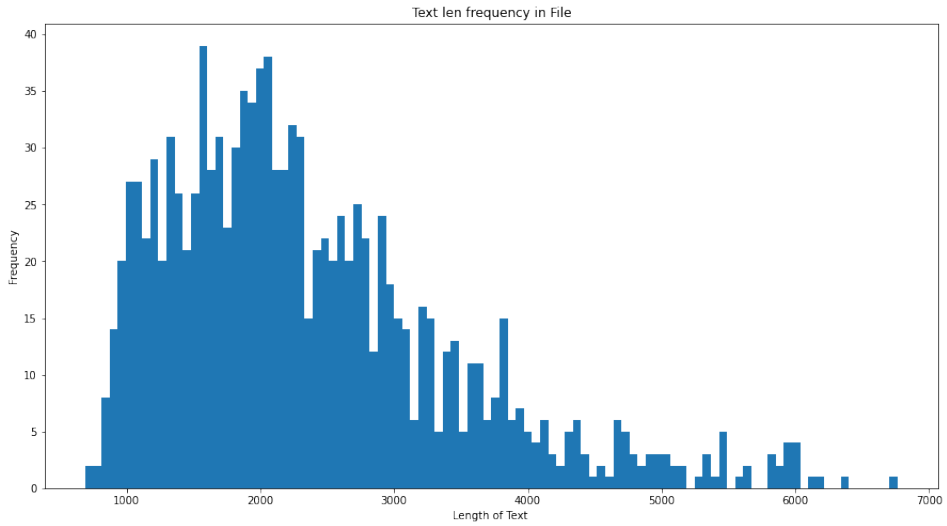

Figure 6: Text Length Frequency in a File

5.2 Evaluation Metrics

We use the following metrics to evaluate our model performance:

- **Training loss** indicates how well the model is fitting the training data
- **Validation loss** indicates how well the model fits new data.
- **Precision** is the ratio between the True Positives and all the Positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2}$$

- **Recall** is the measure of our model correctly identifying True Positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}$$

- **F1-Score** is the Harmonic mean of the Precision and Recall.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

- **Accuracy** is the ratio of the total number of correct predictions and the total number of predictions.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + TrueNegative + FalseNegative} \tag{5}$$

5.3 Train-Test split

We have a total of 1084 essays, and we use 80% of the data as our train data, 10% as the validation and 10% as the test data.

5.4 Hyperparameters

We use the below mentioned hyperparameters for our models.

Table 2: Longformer Hyperparameters

Maximum Length	1024
Stride	128
Minimum number of Tokens	6
Batch Size	4
Learning Rate	5e-5
Weight Decay	0.01
Gradient Accumulation Steps	8
Warm Up Ratio	0.1
Number of Epochs	5

5.5 Results

5.5.1 Longformer: The Long-Document Transformer

The below table summarises the results obtained on training with Longformer.

Table 3: Evaluation Metrics for Longformer

Epoch	Training Loss	Validation Loss	Precision	Recall	F1-Score	Accuracy
0	0.9516	0.6140	0.1719	0.3052	0.2199	0.8040
1	0.5780	0.5584	0.1889	0.3324	0.2409	0.8144
2	0.4916	0.5522	0.2276	0.3580	0.2783	0.8193
3	0.4248	0.5814	0.2292	0.3766	0.2850	0.8118
4	0.3707	0.5962	0.2297	0.3736	0.2844	0.8132

The below table summarizes the F-1 score per class using the LongFormer model.

Table 4: F-1 Scores for test data using LongFormer

Claim	0.5276
Concluding Statement	0.7075
Counterclaim	0.4695
Evidence	0.6525
Lead	0.7941
Position	0.6394
Rebuttal	0.3701

5.5.2 BigBird: Transformers for Longer Sequences

The below table summarises the results obtained on training with BigBird.

Table 5: Evaluation Metrics for BigBird

Epoch	Training Loss	Validation Loss	Precision	Recall	F1-Score	Accuracy
0	1.0197	0.6547	0.2275	0.3626	0.2796	0.7904
1	0.5944	0.5830	0.2968	0.4009	0.3411	0.8126
2	0.4969	0.5840	0.2778	0.4238	0.3356	0.8110
3	0.4164	0.5955	0.2831	0.4349	0.3430	0.8113
4	0.3574	0.6215	0.2827	0.4383	0.3437	0.8078

The below table summarizes the F-1 score per class using the BigBird model.

Table 6: F-1 Scores for test data using BigBird

Claim	0.5571
Concluding Statement	0.8041
Counterclaim	0.4946
Evidence	0.6915
Lead	0.7817
Position	0.6558
Rebuttal	0.3990

6 Conclusion

From the discussed model and pipeline, on experimentation we observed that RoBerta tokenizer performed the best. Looking at the evaluation metrics we observe that both LongFormer and BigBird have similar performance on the data set. We prefer LongFormer over BigBird, as BigBird additionally uses $16x$ more compute power.

In future we would like to explore more how to treat the class imbalance i.e. for instance assigning weights to the class such the frequently occurring class has lower weights and rarely occurring class has higher weight values. Improve the metrics by using combination of LongFormer and BigBird. Further build a ML pipeline using MLOps for better user experience.

References

[1] Feedback Prize - *Evaluating Student Writing*. https://www.kaggle.com/competitions/feedback-prize-2021/overview

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin , *Attention Is All You Need Need*

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

[4] Tom Brown, Ben Mann, Prafulla Dhariwal, Dario Amodei, Nick Ryder, Daniel M Ziegler, and Jeffrey Wu, *Language Models are Few-Shot Learners*

[5] Pinkesh Badjatiya, Litton J Kurisinkel, Manish Gupta, Vasudeva Varma, *Attention-based Neural Text Segmentation*

[6] Yinhan Liu Myle Ott Naman Goyal Jingfei Du Mandar Joshi Danqi Chen Omer Levy Mike Lewis Luke Zettlemoyer Veselin Stoyanov *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

[7] Iz Beltagy Matthew E. Peters Arman Cohan *Longformer: The Long-Document Transformer*

[8] Hugging Face - *BigBird*. https://huggingface.co/google/bigbird-roberta-large

[9] Manzil Zaheer Guru Guruganesh Avinava Dubey Joshua Ainslie Chris Alberti Santiago Ontanon Philip Pham Anirudh Ravula Qifan Wang Li Yang Amr Ahmed *Big Bird: Transformers for Longer Sequences*