Identifying Discourse Elements in Student Essays using Deep Learning

Sai Vineeth Kaza,[1] Rajesh Karumanchi,[2] Aishwarya Kurnutala[3]
Northeastern University[1,2,3]
kaza.s@northeastern.edu,[1] karumanchi.r@northeastern.edu,[2] kurnutala.a@northeastern.edu[3]

Abstract

Writing is a critical skill for success, but less than a third of high school seniors are proficient writers, and low-income, Black, and Hispanic students fare even worse with less than 15 percent demonstrating writing proficiency. Automated writing feedback tools exist, but they often fail to identify crucial writing structures and are inaccessible due to cost, particularly for under-serviced schools with a higher proportion of students of color and from low-income backgrounds. In response to this challenge [1], we have leveraged the largest dataset of student writing from a Feedback Prize Kaggle competition to develop innovative natural language processing techniques for segmenting and classifying argumentative and rhetorical elements in essays using deep learning algorithms. Our approach employs Exploratory Data Analysis (EDA) to understand data, Data Pre-processing to transform the text data, the use of Random Forest Classifier as the baseline model and DL algorithms like pre-trained BERT [7], variations of BERT [8] [11] and LongFormer [13] for the Named Entity Recognition task. These models are designed to identify various parts of the text with a discourse. We also conducted a comparative analysis of the BERT, RoBERTa, DeBERTa, LongFormer and Random Forest Classifier results using metrics such as F1 score, Precision and Recall. The pre-trained DeBERTa model emerged as the top performer, achieving an F1 score of 0.79.

Introduction

Writing proficiency is a crucial skill for academic success, but alarming statistics reveal that less than a third of high school seniors demonstrate proficient writing abilities, with even lower rates among low-income, Black, and Hispanic students. To address this pressing issue and help students improve their writing skills, automated feedback tools have emerged as potential solutions, offering personalized guidance and evaluations of student writing. However, the current landscape of automated writing feedback tools faces several limitations, such as their inability to accurately identify critical writing structures like thesis statements and supporting claims in essays. Moreover, many of these tools are proprietary and lack independent verification of their claims, making them inaccessible due to their prohibitive costs, especially for under-serviced schools serving a significant proportion of students from marginalized backgrounds. This creates a clear need for innovative approaches to democratize education and improve writing proficiency among students.

The main problem addressed in this project is to efficiently segment discourse elements and provide effective feedback. Data preprocessing is a critical step in preparing the dataset for our models. For the Random Forest approach using the TF-IDF matrix, we perform essential preprocessing steps, including stop word removal and stemming or lemmatization techniques. Similarly, for the BERT model, much of the preprocessing is handled by the AutoTokenizer class. This task poses several challenges, including class imbalance, gaps in the essays and majority of the text not classified into discourse types. To overcome these challenges, the project focuses on utilizing the BERT (Bi-directional Encoder Representations from Transformers) and its variations, which are more effective when it comes to long sequence data.

The dataset for this project is sourced from Georgia State University (GSU) as part of the Feedback Prize - Evaluating Student Writing [4] Kaggle competition. It consists of a comprehensive collection of 15,601 .txt files, each containing an essay response. Accompanying CSV files, such as 'train.csv' containing annotated discourse elements and their locations, facilitating the model training and evaluation process.

Our proposed solution revolves around the application of cutting-edge deep learning algorithms, particularly in the field of Named Entity Recognition (NER). Recent advancements in transformer-based architectures, such as BERT (Bi-directional Encoder Representations from Transformers) and Longformer, have shown promising results in NER tasks, making them suitable candidates for our discourse classification objectives. Additionally, we utilized the Random Forest Classifier as a baseline model for comparison.

The Longformer model, an extension of BERT, is specifically designed to handle larger input sequences while maintaining high quality [6]. Through hyperparameter tuning, model comparisons, and evaluation using diverse metrics, we gained valuable understanding regarding the efficacy of both BERT and Longformer in accurately segmenting and classifying argumentative and rhetorical elements in essays.

The project aims to achieve accurate segmentation and classification of discourse elements within student essays. By providing students with valuable feedback on their writing structures, we intend to enhance their overall writing skills and academic performance. To assess the performance, our team intends to compare the outcomes with a pre-trained BERT model and its variations. However, a potential concern lies in the training time and performance speed, which may be impacted by the size of the dataset.

The project responsibilities are distributed among the team members, with each member contributing expertise to various sub-tasks of the project. Collaboratively, we conducted Exploratory Data Analysis (EDA) and feature engineering, implemented the TF-IDF Random Forest Classifier baseline model, pre-trained BERT, variations of BERT, and Longformer models for discourse classification. And performed a comprehensive comparative analysis based on evaluation metrics like F1 score, precision, recall, and accuracy.

Northeastern University, DS5500 Capstone Project, 2023

Background

NER Task

Named Entity Recognition (NER) is an essential sub-task in the field of Natural Language Processing (NLP) that focuses on identifying and classifying named entities in text. Named entities can refer to persons, organizations, locations, date/time expressions, percentages, monetary values, and more. In a typical NER task, a model processes a piece of text and labels each word or sequence of words with the type of entity it represents, if any.

For example, in our case we need to identify Position, Claim, Evidence, Rebuttal, Counterclaim and Concluding Statement entities for a group of words/sentences in an essay.

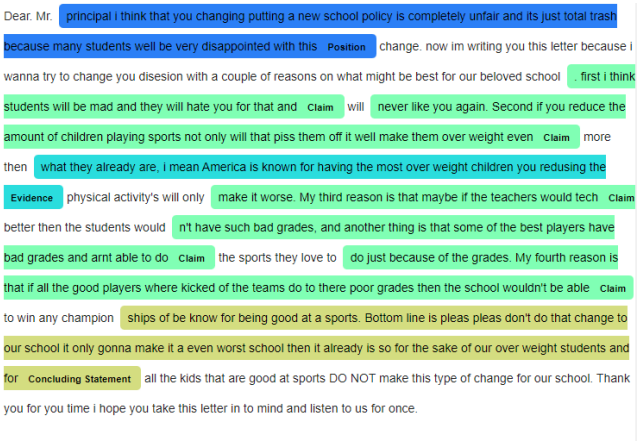


Fig 1: Glimpse of Named Entity Recognition

NER can be quite challenging due to the complexities and nuances of human language. For example, a term that is a named entity in one context might not be in another context. The sentence "I love Apple" could be referring to the fruit (not a named entity) or the technology company (a named entity). Additionally, many entity names can have different forms (e.g., "USA" vs. "United States"). While it was a complicated task thanks to new advancements in deep learning field like transformers. We have state of the art models like BERT (Bidirectional Encoder Representations from Transformers) and its variants. These models, pre-trained on large corpora and fine-tuned on specific NER tasks, have shown remarkable success in achieving high performance on NER tasks, owing to their ability to capture deep contextual relationships between words.

Hugging Face:

An open-source community that has become well-known in the natural language processing (NLP) and machine learning (ML) communities. It is primarily recognized for its contributions to NLP libraries and tools, making it easier for researchers and developers to work with state-of-the-art language models. Hugging face has something called Transformers library. This is an open-source library which provides an extensive collection of pre-trained transformer-based models, including BERT, GPT, RoBERTa, XLNet, etc. It also offers utilities for training, fine-tuning, and deploying these models on various NLP tasks, such as text classification, language generation, question-answering and named entity recognition. Thanks to the Hugging face community we were able to leverage pretrained BERT variations and their weights.

Related Work

We noticed people on Kaggle tried transformer-based approaches like BERT, DistilBERT, Bidirectional LSTMs etc. The notes from the competition winner explained that they achieved success by creating an ensemble of numerous transformer-based models like bigbird-roberta, gpt2-large, albert-xxlarge-v2 and lgb sentence prediction. Two key points we were able to leverage from the previous approaches are 1. Data set is imbalanced, 2. Sentences are too long. Using smote for RFC and resample library for other approaches we were to deal with class imbalance problem, and we also tried Longformer as it is more efficient on long texts. Finally, we decided to pursue a comparative analysis between RFC based NER model, BERT, RoBERTa, DeBERTa and LongFormer models. We also wanted to see how Google Natural Language API or OpenAI's GPT API will identify different discourse elements in the text. But, due to time constraints we were not able to do that.

Exploratory Data Analysis

During the exploratory data analysis, we conducted a comprehensive examination of the dataset to gain insights into the text characteristics and the discourse elements present in the essays. A crucial feature we investigated was the presence of prediction string, where it plays a significant role in the ground truth evaluation. It contains the word indices of the training sample and corresponds to the predicted discourse type for this sequence of words. Additionally, by identifying non-discourse text, such as titles, we ensured that our models focus on the relevant content when analyzing and providing feedback on students' writing. A sneak peak of the dataset can be seen in Fig 2.

	id	discourse_id	discourse_start	discourse_end	discourse_text	discourse_type	discourse_type_num	predictionstring
144270	AFEC37C2D43F	1617802869974	0	317	There has been at least one point in everyone'...	Lead	Lead 1	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18...
144271	AFEC37C2D43F	1617802880610	318	515	Because of this, sometimes, asking just one pe...	Position	Position 1	61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 7...
144272	AFEC37C2D43F	1617802885185	684	692	mistake,	Claim	Claim 1	120
144273	AFEC37C2D43F	1617802891320	693	710	misunderstanding,	Claim	Claim 2	121
144274	AFEC37C2D43F	1617802896937	714	724	misdeed.	Claim	Claim 3	123
144275	AFEC37C2D43F	1617802785480	725	1360	While mistakes and misunderstanding might have...	Evidence	Evidence 1	124 125 126 127 128 129 130 131 132 133 134 13...
144276	AFEC37C2D43F	1617802780393	1361	1471	The more similar iterations people give you, t...	Claim	Claim 4	234 235 236 237 238 239 240 241 242 243 244 24...
144277	AFEC37C2D43F	1617802727496	1472	1881	Misunderstandings are harder to avoid because ...	Evidence	Evidence 2	252 253 254 255 256 257 258 259 260 261 262 26...
144278	AFEC37C2D43F	1617802715474	1882	2019	The best thing to do in a situation like that ...	Claim	Claim 5	326 327 328 329 330 331 332 333 334 335 336 33...

Fig 2: Illustration of train.csv Dataset

Based on Fig 3, it is evident that "Claim" and "Evidence" are the most frequent discourse types, occurring around 50,000 and 45,000 times, respectively. In contrast, "Counterclaim" and "Rebuttal" are infrequent, with less than 5,000 instances each. The prevalence of "Claim" and "Evidence" suggests that essays often revolve around making claims supported by evidence. However, the imbalanced dataset could pose challenges for accurately classifying less frequent discourse types. Addressing this imbalance is crucial when developing automated writing feedback tools.

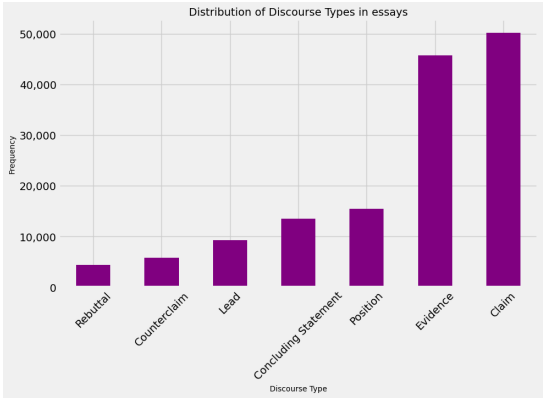


Fig 3: Distribution of Discourse types in essays

Fig 4 reveals the central role of "Evidence1," "Position1," and "Claim1" in essays, present in almost all of them, highlighting their fundamental importance in forming arguments. Most essays also include a Concluding Statement, demonstrating students' ability to summarize their main points effectively. However, it is surprising that around 40% of essays lack a Lead, indicating room for improvement in creating engaging introductions. Moreover, "Counterclaim" and "Rebuttal" appear infrequently, suggesting a need for students to consider alternative perspectives for well-rounded and persuasive writing. Addressing these areas can enhance students' writing skills and foster more balanced and impactful essays.

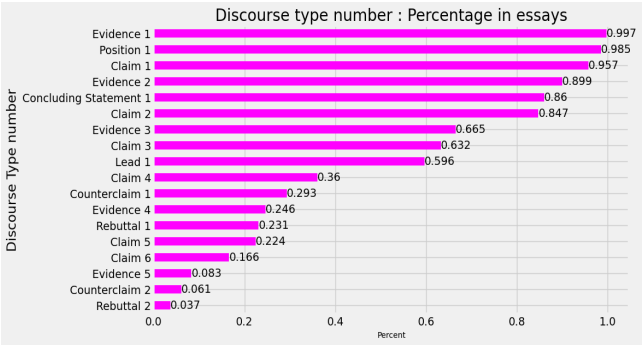


Fig 4: Discourse type number percentage

In Fig 5 the plot depicts the average start and end positions for all discourse types revealing interesting patterns in the placement of discourse elements within essays. Notably, when a Lead is present, it almost always occupies the first position in the essay. On the other hand, Lead2 is a rare occurrence, suggesting that students typically choose to use only one Lead, and it is usually positioned at the beginning of the essay to serve as the introduction.

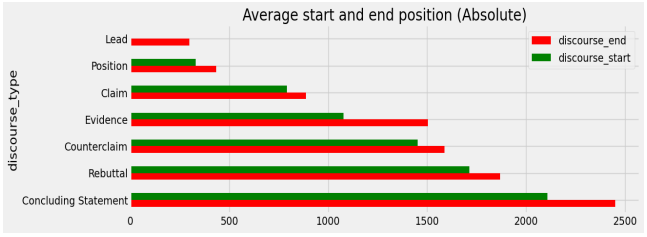


Fig 5: Average Start and end position

Gaps between Annotations

Fig 6 provides valuable insights into the quality and coherence of students' writing. The histogram of gap lengths, with outliers excluded, indicates that there are cases where significant portions of text remain unclassified (not assigned to any discourse type). While most essays may have relatively small gaps, there are instances where the gaps are much larger, indicating potential issues with the overall structure and content of the essays.

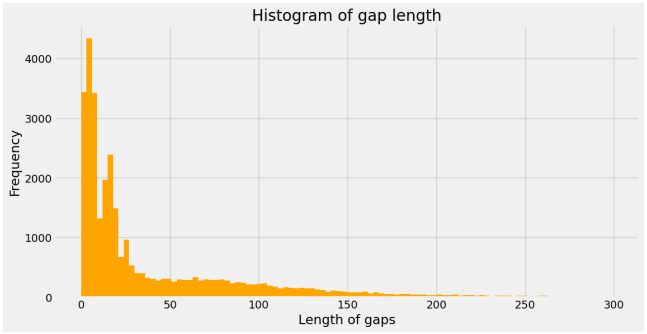


Fig 6: Histogram of gap length

Fig 7 Illustrates The presence of essays with large gaps, some even reaching around 90% of unclassified text, suggests that there are instances of poorly written or poorly organized essays. These essays may lack clear arguments, coherence, or meaningful content, leading to a large portion of the text not being categorized into any discourse type. One example, where the gap_end_length is 7348, stands out as the student copied and pasted the same text multiple times in the essay. This indicates a lack of originality and effort in constructing a coherent argument, resulting in a significant gap in discourse classification. Essays with large gaps and a lack of discourse classification pose challenges for providing meaningful and accurate feedback.

id	essay_length	Gap_length	Gap_end_length	%_not_classified
C278EDC82048	8015	13.0	7348.0	0.92
129497C3E0FC	3616	130.0	3173.0	0.91
F5EE08CB44B9	3022	62.0	2669.0	0.90
B7C17E1993BA	3569	1110.0	2060.0	0.89
F45B396E0A01	1865	1657.0	0.0	0.89

Fig 7: Illustration of bad essays

Project Description

The final workflow of our project is to perform comparative analysis on Random Forest Classifier (baseline), BERT and its variations and Longformer model

Random Forest Classifier using tfidf

Random forests, or random decision forests, are an ensemble learning method used for various tasks such as classification and regression. The idea behind random forests is to construct multiple decision trees during training and combine their outputs to make predictions. For classification tasks, the final prediction is determined by the majority vote of the individual trees, while for regression tasks, it is the average prediction of all the trees.

Random forests are an improvement over individual decision trees because they help overcome the overfitting problem often encountered with single decision trees. By building multiple trees and combining their predictions, random forests achieve better generalization performance on unseen data.

The architecture can be summarized as follows:

Bootstrap Sampling: Random Forest uses a technique called "Bootstrap Aggregating" or "Bagging." It involves creating multiple bootstrap samples (random samples with replacement) from the original training dataset.

Building the Trees on a Random Set of Features: For each decision tree in the Random Forest, only a random subset of features (predictors) is considered for splitting the nodes. This introduces further randomness and diversity into the model.

Decision Rule Creation: Each tree in the Random Forest is grown by applying decision rules to recursively split the data into subsets based on the selected features. The rules are typically based on metrics like Gini impurity or information gain for classification and mean squared error reduction for regression.

Bootstrap Aggregation (Bagging): After constructing multiple decision trees using different bootstrap samples and random subsets of features, the next step is to aggregate their predictions. During the training process, each decision tree is trained independently and does not influence the others.

Majority Voting and Final Class: For classification tasks, the Random Forest employs a majority voting mechanism to make the final prediction. Each tree in the forest makes a prediction, and the class that receives the most votes among all the trees becomes the final predicted class.

Overall, the Random Forest architecture provides a robust and accurate prediction model, less prone to overfitting, and capable of handling large datasets with high dimensionality.

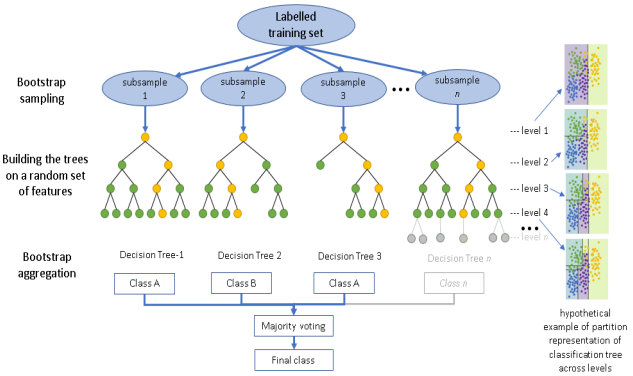


Fig 8: Random Forest Classifier

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical representation used in natural language processing to convert text data into a format that can be understood by machine learning algorithms like the Random Forest classifier. It helps to understand the importance of individual words in a document or a collection of documents (corpus) by calculating a score for each word.

Here's how TF-IDF works with the Random Forest classifier:

Text Preprocessing: Before applying TF-IDF, the text data undergoes preprocessing steps such as lowercasing, tokenization, removing punctuation, and stop word removal to prepare it for analysis.

Term Frequency (TF): The term frequency of a word is the number of times that word appears in a document divided by the total number of words in that document. It gives a measure of how frequently a word occurs within a specific document.

$$Term\ Frequency = \frac{number\ of\ instances\ of\ word\ w\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d}$$

Inverse Document Frequency (IDF): The inverse document frequency of a word is calculated as the total number of documents (N) in the corpus divided by the number of documents that contain the word (w).

$$IDF = log\left(\frac{total\ number\ of\ documents\ (N)\ in\ text\ corpus\ D}{number\ of\ documents\ containing\ w}\right)$$

The IDF score gives a measure of how important or rare a word is across the entire corpus. If a word appears in many documents, its IDF value will be lower, and if it appears in only a few documents, its IDF value will be higher.

TF-IDF Score: The TF-IDF score for each word in a document is obtained by multiplying its Term Frequency (TF) with its Inverse

Document Frequency (IDF). The resulting TF-IDF score reflects the importance of the word in the context of the specific document and the entire corpus.

$$TF - IDF = TF * IDF$$

Vectorization: After calculating the TF-IDF scores for all the words in the documents, the data is transformed into a numerical feature matrix where each row represents a document, and each column represents a word's TF-IDF score.

Training the Random Forest Classifier: The transformed TF-IDF feature matrix is then used as input to train the Random Forest classifier. The classifier learns from this data to identify patterns and relationships between the TF-IDF features and the corresponding class labels (categories).

By using TF-IDF, the Random Forest classifier can now understand the importance of specific words in each document and make more informed decisions when classifying new, unseen text data. However, TF-IDF may not capture some semantic and contextual nuances of language, which can be a limitation. Nonetheless, it remains a powerful and widely used technique for text classification tasks.

For the Random Forest Classifier, to overcome class imbalance, we upsampled 80% of the train data using SMOTE (Synthetic Minority Over-sampling Technique). As the provided Kaggle test data was limited, we split the train data into an 80% training set and a 20% test set. The train data had discourse elements tagged to full sentences or multiple sentences. So, for each essay we split it into individual sentences and assigned corresponding word indices and we transformed the classification model into a Named Entity Recognition model by classifying individual sentences and combining the results. This allowed us to effectively predict the discourse type for each discourse text.

Fig 9 displays the predicted class labels for an essay, where the discourse text is presented alongside the corresponding class labels.

Fig 9: Prediction of an essay using RFC

```
print("F1 Score for Overall:", f1_score_overall)

F1 Score for Lead: 0.5449526813880127
F1 Score for Claim: 0.6972356296621325
F1 Score for Evidence: 0.4764720374476472
F1 Score for Position: 0.8929475587703436
F1 Score for Rebuttal: 0.7110694183864915
F1 Score for Concluding_Statement: 0.5516680227827502
F1 Score for Counterclaim: 0.8021978021978022
F1 Score for Overall: 0.6723746286912459
```

Fig 10: F1 scores for RFC

BERT [Bidirectional Encoder Representations from Transformers]

A bidirectionally trained language model, like BERT, grasps language context and flow better than single-direction models. BERT employs a Transformer, an attention mechanism that captures contextual relations between words in text. The Transformer consists of an encoder and decoder, but since BERT focuses on language modeling, only the encoder is used

BERT utilizes two training methods:

1. Masked LM (MLM): Before inputting word sequences, 15% of the words are replaced with [MASK]. The model predicts the original values of these masked words based on context from non-masked words. This involves adding a classification layer, multiplying output vectors by the embedding matrix, and calculating word probabilities with softmax.

2. Next Sentence Prediction (NSP): The model learns to predict if the second sentence in a pair follows the first in the original document. Training includes pairs where the second sentence is subsequent and others where a random sentence is used. The [CLS] token output undergoes classification to determine the probability of sentence continuity using softmax.

BERT was trained on BooksCorpus (800M words) and English Wikipedia (2,500M words).

The configurations for BERT are:

```
[85] config

BertConfig {
  "_name_or_path": "bert-base-uncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.25.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

Fig 11: Parameters for BERT

And the training parameters are: TRAIN_SPLIT = 0.8, BATCH_SIZE = 8, EPOCHS = 4, SEQUENCE_LENGTH = 512

The loss values for BERT are:

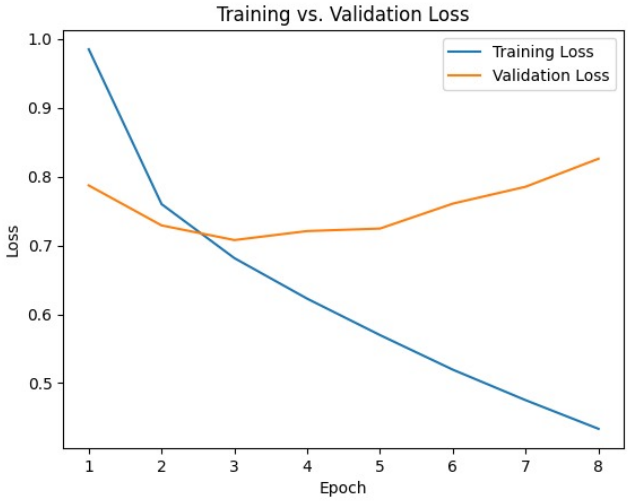


Fig 12: Training and validation loss for BERT

and the F-1 scores are

```
F1 Score for Lead: 0.9322740106186
F1 Score for Claim: 0.8146201512478
F1 Score for Evidence: 0.6577223732102
F1 Score for Position: 0.7250217469312
F1 Score for Rebuttal: 0.5218960217231
F1 Score for Concluding_Statement: 0.75559103161
F1 Score for Counterclaim: 0.6122082146132
F1 Score for Overall: 0.7561892376102
```

Fig 13: F1 scores for BERT

RoBERTa (Robustly Optimized BERT pre-training Approach)

While BERT is pre-trained on "Toronto BookCorpus" and "English Wikipedia datasets" i.e. as a total of 16 GB of data. RoBERTa was also trained on a. BOOKCORPUS (16GB) b. CC-NEWS (76GB) c. OPENWEBTEXT (38GB) d. STORIES (31GB) totalling aroung 160 GB

Other Key advancements are:

a. Dynamic Masking: RoBERTa uses dynamic masking, wherein for different Epochs different part of the sentences are masked. This makes the model more robust than BERT which uses static masking i.e. masking the same part of the sentence in each Epoch.

b. Remove NSP Task: As NSP was not so useful in pre-training BERT. RoBERTa Skips NSP step and Uses only MLM task for training.

c. Large Batch size: BERT uses a batch size of 256 with 1 million steps. RoBERTa used a batch size of 8,000 with 300,000 steps inorder to improve speed and performance.

The configurations for RoBERTa are:

```
[45] config

RobertaConfig {
  "_name_or_path": "roberta-base",
  "architectures": [
    "RobertaForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "transformers_version": "4.25.1",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 50265
}
```

Fig 14: Parameters for RoBERTa

And the training parameters are: TRAIN_SPLIT = 0.8, BATCH_SIZE = 8, EPOCHS = 4, SEQUENCE_LENGTH = 512

The loss values for RoBERTa are:

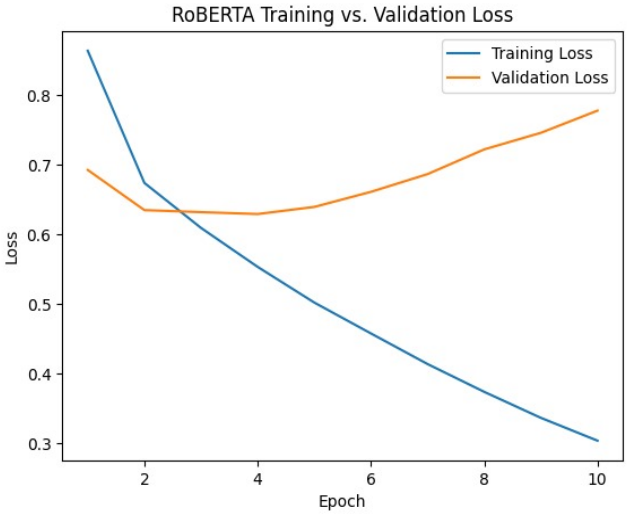


Fig 15: Training and validation loss for RoBERTa

And the F-1 scores are:

```
F1 Score for Lead: 0.9694299091159461
F1 Score for Claim: 0.8693667997718197
F1 Score for Evidence: 0.6102908277404921
F1 Score for Position: 0.8201771274547555
F1 Score for Rebuttal: 0.5889447236180905
F1 Score for Concluding_Statement: 0.7669858641130871
F1 Score for Counterclaim: 0.648936170212766
F1 Score for Overall: 0.7874539622193181
```

Fig 16: F1 scores for RoBERTa

DeBERTa (Decoding-enhanced BERT with disentangled attention):

DeBERTa is trained on was trained on Wikipedia (12GB), BookCorpus (6GB), OPENWEBTEXT (38GB), and STORIES (a subset of Common Crawl (Trinh & Le, 2018); 31GB) totaling around 78 GB. While it is trained only on half of the data compared to RoBERTa it performed consistently better on a wide range of NLP tasks (achieving improvements on MNLI by +0.9% , SQuAD v2.0 by +2.3%, RACE by +3.6% due to two key advancements

1.Distangled Attention:

In BERT, each word in the input layer is represented by a vector, which is the sum of its content embedding and position embedding. On the other hand, DeBERTa uses two vectors to represent each word—one for content and another for position. The attention weights between words are computed using disentangled matrices based on their contents and relative positions. This is driven by the understanding that attention weight between word pairs relies not just on their contents but also their relative positions.

Disentangled Attention in DeBERTa separates the different types of attention that each head learns. This disentanglement process allows the model to focus on different aspects of the input more explicitly. For example, some attention heads can focus on local context, while others attend to global context or long-range dependencies. This makes DeBERTa's attention mechanism more interpretable and provides better control over how the model allocates attention.

2. Enhanced Mask Decoder:

The BERT model incorporates absolute positions in the input layer. In DeBERTa, we incorporate them right after all the Transformer layers but before the softmax layer for masked token prediction. In this way, DeBERTa captures the relative positions in all the Transformer layers and only uses absolute positions as complementary information when decoding the masked words. Thus, we call DeBERTa's decoding component an Enhanced Mask Decoder (EMD).

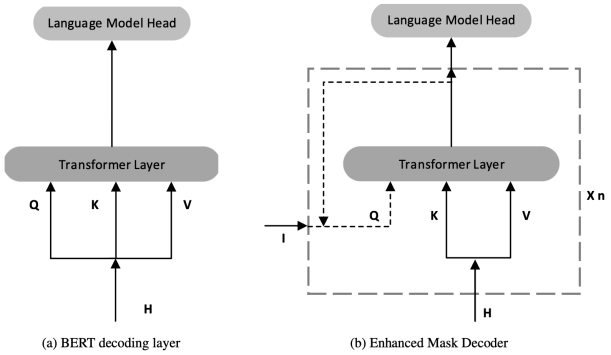


Fig 17: Enhanced Masked Decoder

The configurations for DeBERTa are

```
  "layer_norm_eps": 1e-07,
  "max_position_embeddings": 512,
  "max_relative_positions": -1,
  "model_type": "deberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_dropout": 0,
  "pooler_hidden_act": "gelu",
  "pooler_hidden_size": 768,
  "pos_att_type": [
    "c2p",
    "p2c"
  ],
  "position_biased_input": false,
  "relative_attention": true,
  "transformers_version": "4.31.0",
  "type_vocab_size": 0,
  "vocab_size": 50265
}
```

Fig 18: Parameters for DeBERTa

And the training parameters are: TRAIN_SPLIT = 0.8, BATCH_SIZE = 8, EPOCHS = 4, SEQUENCE_LENGTH = 512

The loss values for DeBERTa are:

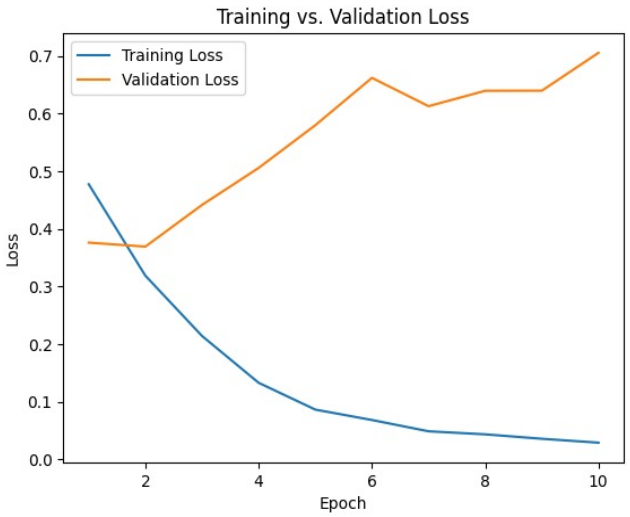


Fig 19: Training and validation loss for DeBERTa

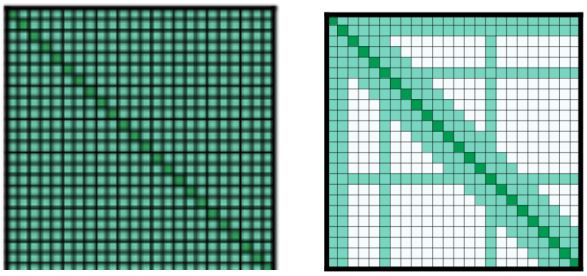
And the F-1 scores are:

```
F1 Score for Lead: 0.9622850804215197
F1 Score for Claim: 0.8515730553947114
F1 Score for Evidence: 0.6788522848034007
F1 Score for Position: 0.7850208044382802
F1 Score for Rebuttal: 0.5555555555555556
F1 Score for Concluding_Statement: 0.7996449966718437
F1 Score for Counterclaim: 0.6945668135095447
F1 Score for Overall: 0.7930654058313633
```

Fig 20: F1 scores for DeBERTa

Longformer

Longformer is a transformer-based language model designed to efficiently process long documents or sequences, making it a valuable tool for natural language processing tasks that involve extended texts. Unlike traditional transformer models, such as BERT, Longformer introduces a combination of global and local attention mechanisms to handle long-range dependencies effectively.



Full n^2 attention o(n^2) | **Global + Sliding window attention O(N*W + S*N*2)**

The key innovation in Longformer is the "Sliding Window Attention," which allows the model to attend only to a subset of tokens within a fixed-size window. This reduces the quadratic attention complexity associated with traditional self-attention mechanisms and makes Longformer more scalable for long sequences. By using Sliding Window Attention and other sparsity patterns, Longformer achieves linear complexity with respect to the sequence length, making it computationally more efficient for processing lengthy documents. As shown in the figure above by using sliding window attention similar to CNN process they were able to reduce quadaratic complexity to linear. In the above figure N = no of tokens, W = sliding window, S = no of super tokens.

Longformer's global attention mechanism enables it to capture the entire document's context, while the local attention focuses on the nearby tokens within the sliding window. This combination allows the model to effectively handle both short-range and long-range dependencies in the text

The loss value for longformer is:

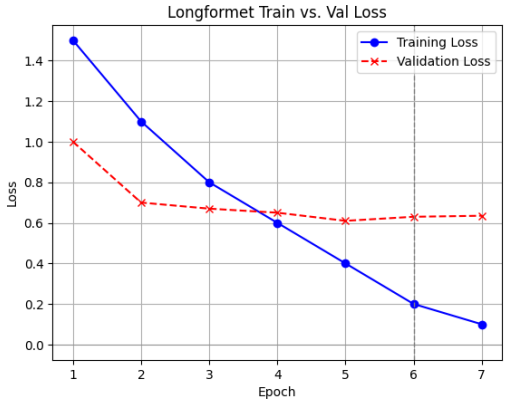


Fig 21: Training and validation loss for longformer

Result:

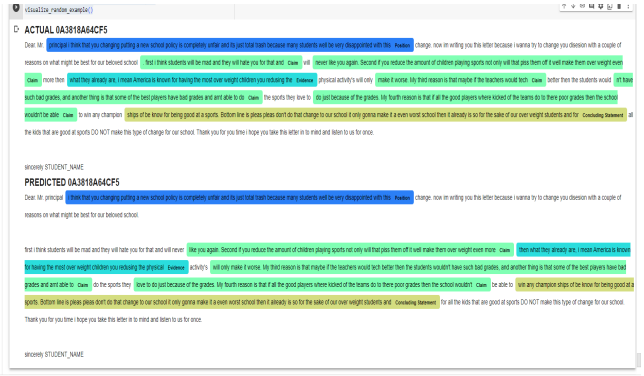


Fig 22: Actual vs Predicted

Each discourse element and its corresponding actual words indices were depicted on top while the predicted word indices are

highlighted in the bottom. We can also see the discourse name at the end of each highlighted text

Comparative Analysis & Conclusion

We are using F-1 as an evaluation metric as it is the evaluation metric used in Kaggle competition. We also compared models against precision, recall and runtime,

Apart from default pretrained model configs BERT and RoBERTa were trained on max_length = 512, batch_size = 8, learning_rate = 1e-05 with adam optimizer configurations. While DeBERTa takes up a bit more memory we reduced the batch_size to 4 instead of 8. For Longformer apart from default configurations we've set batch_size = 4, learning_rate = 0.25e-05.

All these models were ran on colab pro, with high ram and v100 GPU with 80% if the data for training and 20% for testing. All the F-1 results are shown below. (if the match between actual discourse word indices and predicted discourse word indices > 50% then it's a ture positive (TP), if its < 50% then its false positive, if there are word indices for a discourse element in actual text and if there are no word indices for it in predicted value, then it's a false negative)

To our surprise RFC based NER model had a decent F-1 score, reason being majority of the discourse texts has either a full sentence or a group of full sentences. And for RFC we classified sentences and merged common discourse sentences of an essay into one and if the 50% word indices match in an actual vs predicted discourse text in an essay, it's a true positive. Thus having 50% threshold and full sentence level discourse gave us better F-1 score. As for BERT models, they performed as expected DeBERTa achieving better scores than RoBERTa and BERT. Longformer didn't beat BERT models here. This is because Longformer models works best if there is relationship between sentences that are far apart from each other in a text. In our case there are relations between sentences that are close to each other. Hence BERT outperformed Longformer.

our final comparative analysis between the models is given below

	F1-Score	Precision	Recall	Runtime
Random Forest Classifier with NER task	0.672	0.698	0.647	0:25:00
BERT	0.756	0.861	0.651	1:32:23
RoBERTa	0.787	0.881	0.697	1:01:09
DeBERTa	0.793	0.885	0.706	2:40:52
Longformer	0.714	0.751	0.682	5:25:10

Table 1: Comparative analysis

The best F-1 score achieved in Kaggle was 0.75. We have better F-1 scores here because we tested out models on like 3k eassays whereas the submissions in Kaggle was tested against nearly 100k essays and since the submission window is closed we don't have access to those 100k essays.

Conclusion and Future work:

In this project we were able to see the power of deep learning models in NER task and had a chance to work with different transformer-based variations. We observed DeBERTa gave us the highest F-1 score. As part of our future work, we will focus on enhancing the F1 scores through the exploration of ensemble models and Advanced BERT models. We plan to implement advanced BERT models such as LayoutLM, Reformer, and Big Bird to enhance discourse element identification, as they have the potential to outperform the previously utilized BERT variations. Additionally, we will investigate ensemble approaches by combining predictions from various models, including BERT variations and the Random Forest classifier, to create a more robust and accurate prediction system. To address class imbalance issues and improve the model's generalization capabilities across different essay styles and topics, we plan to augment the dataset with synthetic examples or apply data augmentation techniques. Lastly, we plan to explore an ensemble of numerous transformer-based models like bigbird-roberta, gpt2-large, albert-xxlarge-v2 and lgb sentence prediction. These future directions aim to improve the performance of the automated writing feedback tool and provide more precise discourse element identification, thereby benefiting students and enhancing their writing proficiency.

GitHub Link & Instructions

GitHub Repository : DS5500 Code Base

Instructions to run the code:

1. Download dataset from Kaggle
2. Save all the contents in your local or google drive with name " feedback-prize-2021".
3. Check the data location paths and set it to your google drive path . Once these paths are updated everything else runs normally

REFERENCES

[1] https://www.kaggle.com/competitions/feedback-prize-2021/data

[2] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[3] https://handsonnlpmodelreview.quora.com/Maximizing-BERT-model-performance-An-approach-to-evaluate-a-pre-trained-BERT-model-to-increase-performance-Figure-1-T

[4] BERT paper https://arxiv.org/pdf/1810.04805.pdf

[5] https://towardsdatascience.com/exploring-bert-variants-albert-roberta-electra-642dfe51bc23#:~:text=Dynamic%20Masking%3A%20BERT%20uses%20static,makes%20the%20model%20more%20robust.

[6] RoBERTa paper https://arxiv.org/pdf/1907.11692.pdf

[7] BERT Pre-trained model: https://huggingface.co/docs/transformers/model_doc/bert
[8] RoBERTa Pre-trained model: https://huggingface.co/docs/transformers/model_doc/roberta#transformers.RobertaModel
[9] https://sh-tsang.medium.com/brief-review-deberta-decoding-enhanced-bert-with-disentangled-attention-f5cdb9a8bf0b
[10] https://medium.com/dair-ai/papers-explained-08-deberta-a808d9b2c52d
[11] DeBERTa Pre-trained model: https://huggingface.co/docs/transformers/model_doc/deberta
[12] DeBERTa Paper: https://arxiv.org/abs/2006.03654
[13] Longformer Pre-trained weights: https://huggingface.co/docs/transformers/model_doc/longformer
[14] https://medium.com/@tenzin_ngodup/simple-text-classification-using-random-forest-fe230be1e857
[15] https://towardsdatascience.com/longformer-the-long-document-transformer-cdfeefe81e89
[16] https://sh-tsang.medium.com/brief-review-longformer-the-long-document-transformer-8ab204d56613