Automated Discourse Identification in Argumentative Essays

Nikki Ratanapanichkich
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
nratan@umich.edu

Andrew Scheffer
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
drewskis@umich.edu

I. Problem Description

Automated Essay Scoring (AES) is an often misunderstood field of research that investigates how to systematically parse and score essays written in an educational setting. The goal of AES systems is to accurately and efficiently assess essay elements to provide students with quick and reliable feedback on their writing skills. This feedback can provide crucial information for students iterating on their writing skills, especially for those from disadvantaged backgrounds with limited access to a knowledgeable instructor or tutor. AES Systems require a breadth of natural language processing techniques in order to provide adequate feedback; these systems must measure spelling, coherence, factual validity, argument soundness, and even discourse structure. This complexity of developing accurate AES systems make it an active area of current research.

In recent years, the demand for AES systems have increased dramatically. Recently, initiatives have been launched worldwide to digitize and distribute anonymous student essays in an effort to make essay feedback and writing education more accessible to everyone, especially those from low-income, minority backgrounds, who lack sufficient resources to help them improve their writing abilities. It is beneficial to design fast, effective, and affordable solutions for automated grading of student-written essays in contrast to current AES systems which are very expensive. The development of AES technologies would decrease the educational barrier for students to improve their writing skills, and would allow them to iterate on their work in a fraction of the time.

In an attempt to fulfill the need for this high demand of affordable AES systems, this report presents a customized *Discourse Segmentation Model*, a critical component of the AES pipeline. *Discourse Segmentation* requires the accurate categorization of different argumentative elements in essays, which allows downstream AES systems to analyze essay structure, factual basis, etc. Our system takes argumentative essays as input and outputs a sequence of tokens indicating the corresponding argumentative element for each word of the essay. This task is challenging for many reasons. First, student essays are often riddled with misspelled words. Additionally, long essays present a challenge to language models that need to form contextual dependencies across large amounts of text. Our team uses data from the *PERSUADE Corpus* [3] in conjunction with an encoder transformer model to perform this sequencing task. This process was then evaluated using both qualitative and quantitative test metrics.

II. Related Work

In a similar vein to our topic, Taghipour and Ng were the first to explore a neural approach to Automated Essay Scoring. This approach takes a sequence of the one-hot vectors of all the words in a given essay as input to the model. The model first uses a convolution layer, used to extract n-gram level features, which captures the local textual dependencies between words in an n-gram. These features are then passed to a recurrent layer composed of a Long-Short Term Memory (LSTM) network [5].

We also took inspiration from work done with the SciBERT model, which was not used for student essays, but did conduct a similar task for scientific articles. SciBERT, based off of the BERT model, conducts scientific discourse tagging in order to improve the efficiency of parsing through scientific articles. SciBERT uses the BERT transformer which has a limited sized input and makes use of a moving-window approach in order to capture all the relevant context [1].

Our model improves upon this prior research by proposing a new model to use in place of BERT. In our project, we are instead using the *BigBird* transformer, because *BigBird* is more optimized for processing long input sequences. Since the data that we are dealing with contains essays that can be very long in length, *BigBird* allows us to avoid using the moving-window approach while still being able to capture all the relevant contextual information in the input.

III. Dataset and Preprocessing

Our team used a subset of the *PERSUADE Corpus* collected by the GSU Learning Agency Lab [3]. Exactly 15,594 student essays (at the 6th-12th grade level) were used for training and evaluation in this project. Each essay in the PERSUADE corpus was annotated by human raters for argumentative and discourse elements using a double-blind rating process. In an effort to characterize the quality of this dataset, our team focused on quantitative metrics of the essays.

Fig. 1 depicts the number of words present in all essays in the dataset, averaging around 421 words. Along with essay length, our team also visualized the total number of words categorized as each discourse tag, shown in Fig. 2. As expected, the *Evidence* tag is actually associated with more than 50% of all words in the argumentative essays! We also decided to determine the average Flesch reading ease score (70.77417), average number of words per sentence (21.0393), and the average number of sentences per essay (21.0393).

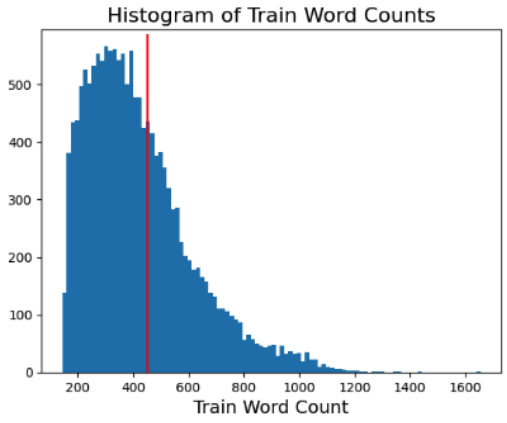

Fig. 1: Histogram of num words in student essays ($\mu = 421; \sigma = 191$)

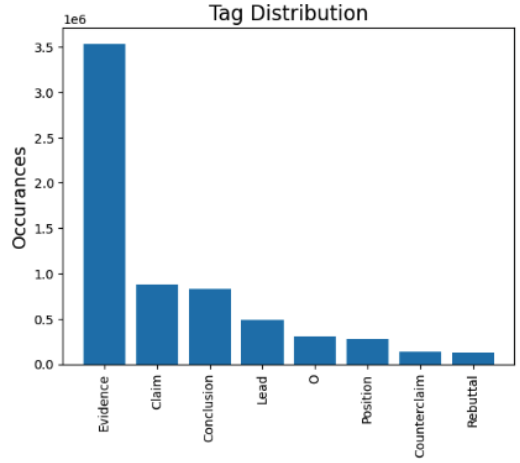

Fig. 2: Distribution of tag occurrences over all essays

The first major task of this project was converting the raw data from the *PERSUADE Corpus* to a format that was suitable for our task. Each element of the dataset consisted of a full student essay accompanied by a short list of the discourse elements of that essay, with corresponding word indices. For example, the short list could contain "Lead, 0 1, Claim, 2 3 4", meaning that words 0 and 1 in the essay are of the lead element, and words 2, 3, and 4 are of the claim element. This dataset was suitable for our task because it provided a corresponding discourse element for each word, which is also what our model intended to do.

Our team decided to convert these lists into tag sequences of the same length of the original essay, where each word corresponds to its own tag; that is, essay[i] would have the tag, tags[i], where i is a word index. Additionally, we expanded the amount of tags to include B-[tag] and I-[tag] for all discourse elements. The reason for this is to establish a reasonable baseline (discussed in Section V) and allow our model to infer relationships about the beginnings of discourse sequences. At the end of this preprocessing stage, we successfully assign all words in all essays their ground-truth tag/ID. A list of all 15 IDs/tags is shown in Table 1.

Table 1. All possible tags for discourse elements

ID	Discourse Tag
0	O
1	B-Lead
2	I-Lead
3	B-Position
4	I-Position
5	B-Claim
6	I-Claim
7	B-Counterclaim
8	I-Counterclaim
9	B-Rebuttal
10	I-Rebuttal
11	B-Evidence
12	I-Evidence
13	B-Concluding Statement
14	I-Concluding Statement

When we preprocess the essays, we deliberately avoid converting them to all lowercase characters, because words that begin with uppercase characters are more likely to be the beginnings of a sentence or discourse element. Therefore, we expect that keeping the input data text in its original case improves the performance of the word tagging as it is easier to identify the start of different discourse types in an essay. As a final preprocessing step, our team filters out extraneous and unreadable characters in the dataset that appear to be prevalent.

IV. Methodology

One popular application of Natural Language Processing is text-tagging, commonly used for tasks such as Part of Speech (POS) tagging and Named Entity Recognition (NER). We believe that the problem of identifying different components of student essays (i.e. leads, arguments, evidence, etc) may be mapped directly to a case-specific text-tagging problem. In this formulation, each word in an input text will be mapped to a corresponding tag listed in Table 1.

At a high level, our team proposes a three-step process for student essay component tagging:

1) Data preprocessing and data structure loading
2) Perform text tagging by fine tuning a pre-trained large transformer architecture, *BigBird*
3) Generate prediction strings for argumentative essays

A. Transformer Architecture

Transformers have become a popular choice for various Natural Language Processing (NLP) tasks, especially for text-tagging tasks such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and even sentiment analysis [7]. This popularity largely stems from their inherent ability to capture long-term dependencies and contexts in text. When compared to more traditional NLP techniques such as LSTM or GRU architectures, Transformers tend to outperform these models at capturing long-term dependencies [6]. This is largely due to how every word in an input sequence is processed together instead of sequentially, leaving little room for information loss.

We propose the use of the transformer architecture, *BigBird* [8], for our student-writing tagging task. We choose this specific transformer architecture for our task largely because of its focus on processing long sequences of input. This model uses a "sparse-attention mechanism" to combat the quadratic space/time complexity problem of subsequent models, connecting all input nodes to output nodes. This sparse-attention mechanism is largely based on random attention connections that allow the model to process input sequences up to 4096 tokens long (∼8x longer than previously possible by similar hardware). As seen in Fig. 1, this max input length should be perfectly sufficient for our task, given that the max essay length is less than 2000 words.

A diagram of our model architecture is shown below in Fig. 3, with data flowing from the bottom (the entire input essay text) to the top (word-based discourse tags).

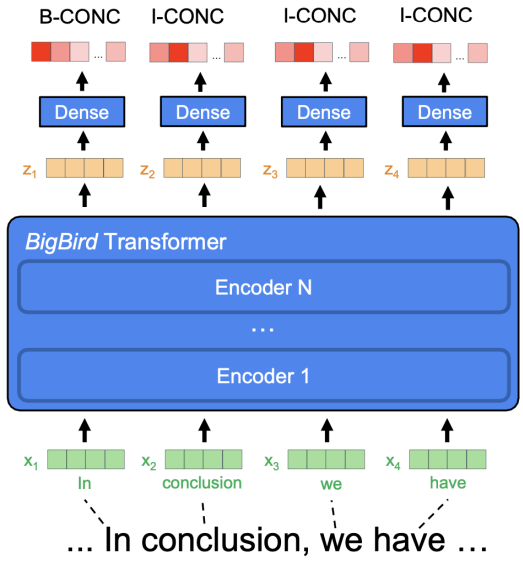

Fig. 3: Architecture of modified *BigBird* token classification model with customized head

B. Training Process

Fine-tuning the *BigBird* model for our given task was the major source of work over the past few weeks and required significant computational resources. As seen in in Fig. 3, the process starts by first tokenizing the input sequences of text into sub-word feature vectors that can be easily fed into the *BigBird* model. Our team found that sub-word features were ideal for our task because of their resilience to misspelled words and new vocabulary. This process is a pre-trained component of the architecture downloaded from HuggingFace. The tokenizer pads all token sequences to the same length and returns an attention mask which allows us to send batches of essays into the transformer even when the essays in each individual batch likely have different lengths.

Next, we download a BigBirdForTokenClassification model architecture with pre-trained weights that is in "sparse-attention" mode for large sequences. Additionally, we set the hyperparameter attention_prob_dropout = 0.1, set 12 attention heads, and define 12 hidden layers. This model also uses a position-embedding matrix to encode the relative position of each token in the sequence. In each training step, we pass the input-ids, attention-masks, and the corresponding labels to the *BigBird* model and update the gradients based on the cross-entropy loss of the output predictions. To compute the final accuracy of the model, we take the argmax of the predictions matrix (of size $(|W|, |C|)$ where $|W|$ is the number of words in the sequence and $|C|$ is the number of tag-classes). We only compute accuracy at the active labels in a batch (not labels that were padded). We also decided to use grad-norm gradient clipping as referenced in [4].

C. Baseline For Evaluation

As a baseline for our proposed method, our team redesigned a previously implemented NER tagger using Hidden Markov Models (HMMs). This model is more naive due to the Markov assumption used in the implementation of HMMs. We show that the context and attention offered in modern transformer architectures drastically outperforms the HMM technique in student essay component identification tasks.

V. Experiments

After a lot of effort, our team was able to successfully create and train both an HMM baseline and our modified *BigBird* transformer architecture described in IV. We split the dataset into training/validation/test sets in an 80/10/10 split. In this section, we will present and perform an analysis of our results.

A. Baseline

As described previously, our baseline for our word-tagging problem is a Hidden Markov Model trained on the corpus of data in the training set (roughly 12474 essays). This model then is tuned on the validation set (roughly 1558 essays) to find proper hyperparameters. In our research, we found that the values of $k_{initial} = 1$, $k_{transmission} = 1$, and $k_{emssion} = 0.5$, maximized our accuracy on the validation set. After hyperparameter tuning, the model was run on the test set, which yielded an overall accuracy of 0.567 over all tags (B-Claim, I-evidence, etc). Fig. 4 depicts the confusion matrix of the HMM model after training.

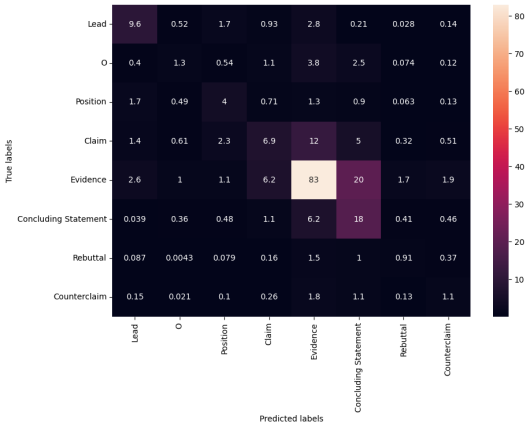

Fig. 4: Confusion Matrix of Trained Hidden Markov Model

As seen in this figure, the HMM did a decent job of identifying evidence-based words in the text. The main aspect of confusion was in the concluding statements. We believe that this confusion may stem from the fact that many conclusion statements reiterate the evidence presented in the rest of the paragraph. Additionally, one can see that the model is often incorrectly identifying claims vs. evidence statements on average. This is somewhat expected with the naive assumptions this model makes: basing the predictions of specific words largely based on the frequency count of those words in the corpus. Finally, this model seems to suffer at identifying the differences between claims/counterclaims and evidence/rebuttals as seen by the nearly uniform confusion matrix above for those sections. This shows that the model is not capturing the deeper meaning or themes of different sections of the text. Overall, this model is inherently flawed because it assumes the probability of word-tag pairs is directly correlated with their respective frequencies in the training set.

B. BigBird Model

Now we will go over the results of fine-tuning the *BigBird* transformer architecture to our word-tagging task. Our team used Google Colab to train our network over the training set (14034 essays) for a total of 5 epochs with 4 essays per batch. Using GPU acceleration and vectorized code, our team managed to get the runtime per epoch just under 1.5 hours, for a total training time of 7.3 hours. We used a decaying learning rate with the initial value of 2.5×10^{-4} for the first epoch that decreased by a factor of 10 every subsequent epoch. This was chosen to hopefully localize in on a more global-minimum of the loss function. Please refer to Fig. 5 for a history of the model's training loss and training accuracy over time.

Over 5 epochs, our training accuracy reached roughly 86.55% and the training loss reached just under 0.38. Our final accuracy on the test set (of 1560 essays) came out to be 0.7955, a significant improvement over our baseline. This accuracy metric was calculated by dividing the total number of correct predictions by the total number of predictions.

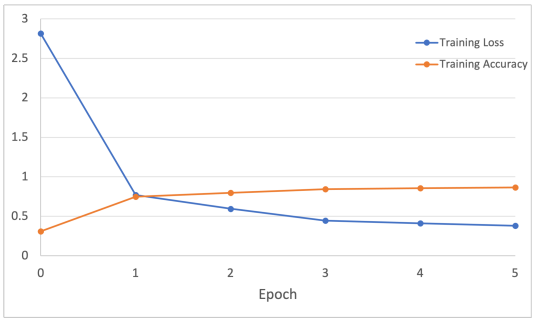

Fig. 5: Training Loss/Accuracy for *BigBird* over 5 epochs

Fig. 6, showcases the confusion matrix for our transformer-based architecture.

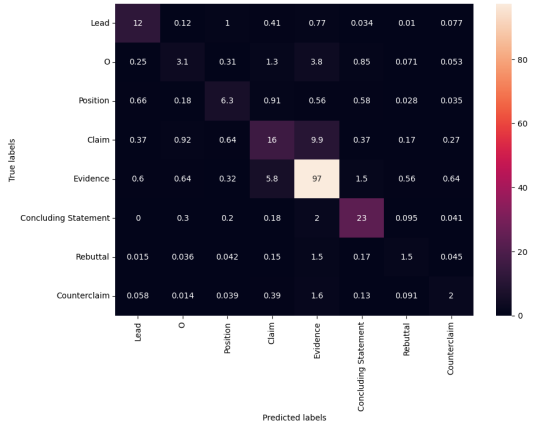

Fig. 6: Confusion Matrix of Trained *BigBird* Model

As seen in this confusion matrix, our transformer model significantly outperforms the baseline in all categories. Notably, this model does a significantly better job at localizing discourse elements that are based on location within a paragraph. For example, this model outperformed the baseline at predicting the Lead tags and Conclusion tags. This model still suffers slightly at differentiating between claim and evidence elements; however, unlike the baseline method, this model correctly predicts a "Claim" tags most often when the true label is "Claim."

C. Quantitative Results

Table 2 is a quantitative comparison between the baseline HMM model and the modified *BigBird* transformer model. Additionally, out of interest, we also compared both models to an "Only Evidence" model that only predicts the tag "Evidence" for every word in the essays. In this table, we are comparing using two metrics: the accuracy, and macro-F1 score. We chose to use macro-F1 because it will still reflect true model performance even when the classes are heavily imbalanced (in our case towards evidence tags). As seen in the table the *BigBird* Model significantly outperforms the HMM both in accuracy and macro-F1 score demonstrating that it is able to accurately distinguish between more classes

than the HMM. We presume that the HMM has a much lower macro-F1 score because of its direct reliance on biased training data. Here, we also can see that the accuracy of the HMM model barely outperforms the extremely naive "Only Evidence" predictions in overall accuracy, further supporting the use of the transformer architecture described in this paper.

Table 2. Comparison between HMM and *BigBird* models

	Test Accuracy	Test Macro F1-Score
Only Evidence	0.5287	0.0864
HMM	0.5674	0.3947
BigBird Model	0.7955	0.6645

D. Qualitative Results

Our team also wanted to investigate some specific case examples of where the Transformer architecture outperformed the Hidden Markov Model to get a better sense of what was being learned. To do this, we created a tool to visualize the tags on top of the original sentences. In Fig. 7, we compare two identical pieces of text run through both taggers.

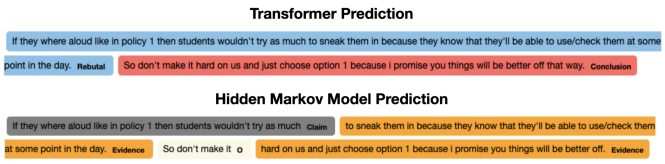

Fig. 7: Selected comparison of HMM and BigBird at essay conclusion

In this figure, the Transformer model can be seen creating a prediction that is much more reminiscent of the actual text. We can infer that the transformer is taking advantage of words such as "if … then" to deduce that the words belonging to the first sentence are likely part of a rebuttal. Interestingly, the transformer architecture also seems to be segmenting the text on a per-sentence basis (which is very representative of the dataset it was trained on).

Overall, as seen in the overall accuracy and confusion matrices, the transformer successfully incorporated context-clues to create an effective word-tagging system.

VI. Conclusion

Our team was able to perform automated discourse segmentation, a key component in AES systems, to a high degree of accuracy using a novel modified transformer-based architecture designed for long inputs. Our team believes that there are many opportunities for future improvements in this work. One potential improvement would be the inclusion of additional models into the pipeline (such as LongFormer [2]) to obtain more accurate predictions. More sophisticated post-processing techniques may be required to combine these different methods. One potential way to actually combine the models could be to add their corresponding word probabilities as a naive combination.

Additionally, if given more time, our team believes that it would be worthwhile to investigate performing beam-search on the returned probability matrix from the transformer. This would be more effective than taking an argmax over all possible tags. Since *BigBird* is a transformer, theoretically, this shouldn't help significantly; however, since *BigBird* is based on randomized connections, the context information may not be completely uniform, making beam-search potentially beneficial.

Overall, throughout this project, our team learned a significant amount about the implementation of NLP systems. Seemingly small tweaks (i.e. stripping newline characters, making all input lowercase, etc) can have a drastic impact on the performance of the system. Additionally, we learned that more complex models are significantly harder to interpret. It was clear to reason why the more simplistic HMM failed at some segmentation tasks, however, pinning down exactly went wrong in the transformer architecture was much harder to do. Finally, the effort to create a reasonable and simplistic baseline can provide a significant amount of value when developing NLP systems, especially when determining if your model is better than a naive solution.

References

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. pages 3615–3620, November 2019.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, Dec 2020.

[3] Scott A. Crossley, Perpetual Baffour, Yu Tian, Aigner Picou, Meg Benner, and Ulrich Boser. The persuasive essays for rating, selecting, and understanding argumentative and discourse elements (persuade) corpus 1.0. *Assessing Writing*, 54:100667, 2022.

[4] Nitish Keskar and Bryan McCann. Ctrl: A conditional transformer language model for controllable generation.

[5] Kaveh Taghipour and Hwee Tou Ng. A neural approach to automated essay scoring. pages 1882–1891, November 2016.

[6] Ashish Vaswan and Noam Shazeer. Attention is all you need. *31st Conference on Neural Information Processing Systems*, Dec 2017.

[7] Krzysztof Wrobel and Krzysztof Nowak3. Transformer-based part-of-speech tagging and lemmatization for latin. *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages*, page 193–197, Jun 2022.

[8] Manzil Zaheer and Guru Guruganesh. Big bird: Transformers for longer sequences. *34th Conference on Neural Information Processing Systems (*, Jan 2021.