Feedback Tool - Identifying Argumentative Essay Elements

Nitin Godi • Amir Jafari

Abstract Writing is one of the most important skills one can have in their armory to achieve significant levels of success. The best way to improve one's writing skills is by practicing writing essays. Students are often asked to write essays to improve their writing skills. According to the National Assessment of Educational Progress, more than two-thirds of high school seniors are not proficient writers. These students can improve their essay writing skills with the help of automated feedback tools. However, most of these tools are proprietary and often fail to recognize the writing structures, such as thesis statements and support for claims, in essays. This paper discusses the concepts used and models trained to create a tool to identify and distinguish various essay elements used in an essay. Since these elements vary in their word count and their position inside the essay, it would be challenging to distinguish the elements from one another accurately. Three different methods have been used to find the best method to distinguish essay elements accurately. These methods involve preprocessing the essays differently and using the preprocessed data to train different models such as the Naive-Bayes, Bidirectional LSTM (Long-Short Term Memory), and Roberta. The results show Roberta model performs the best, with an F1 score of about 0.62.

Keywords Essay elements classification, Naïve-Bayes model, Bidirectional LSTM model, Roberta model, Chunking, Sequence Tagging.

Nitin Godi
George Washington University
Washington, DC, USA
Email: nitingodi29@gwu.edu

Amir Hossein Jafari
George Washington University
Washington, DC, USA
Email: ajafari@gwu.edu

1 Introduction

Writing skills are essential to succeed in one's career. They are helpful in the application process for colleges and universities, job hunting, documentation of ideas and information, and the list continues. One way to improve writing skills is by writing essays. Writing essays helps one demonstrate an understanding of a particular topic, improve communication skills, cultivate critical thinking, develop persuasive skills, and learn to construct arguments. Hence, students in schools are made to write many essays.

Essay writing is an art. Despite having rules governing each essay writing style, no two essays are alike. An argumentative essay is one such essay style. It is a type of essay writing involving topic investigation, establishing a position, and collecting, generating, and evaluating evidence supporting the position. Some basic elements/ types of statements commonly used in argumentative essays are Lead, Position, Claim, Counter Claim, Rebuttal, Evidence, and Concluding Statement.

Many students lack writing skills. It is hard for their teachers to provide personalized feedback to all students since they outnumber the teachers significantly. One solution to this problem is automated feedback tools. These tools can provide personalized feedback to the students almost immediately. There are currently numerous automated feedback tools that provide feedback and help students improve their essay writing skills [6-10]. However, many of these tools focus mainly on grading the essays, while others often fail to identify structures such as introductory statements, thesis statements, concluding statements, and so on in essays or do not do it thoroughly.

Moreover, most of these tools are proprietary, making it very difficult for students from poor financial backgrounds to afford them. Hence, an open-source automated feedback tool that identifies all types of essay statements is required to help all students improve their writing skills. This paper discusses how a crude version of this open-source automated feedback system was developed.

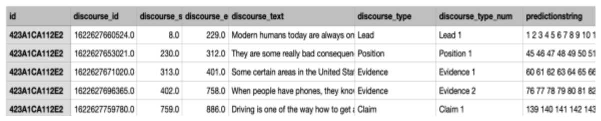
2 Related Works

Many works discussing the models and data handling techniques used in this process have been documented. Xu, S. et al. [1] discuss using the Multinomial Naive-Bayes classifier in multiclass text classification problems. This concept was used in the project to investigate its performance compared to neural networks. Papers from Zhai Z. et al. [2] and Huang Z. et al. [3] discussed the use of Bidirectional LSTM models along with Conditional Random Field (CRF) and Convolution Neural Networks (CNN) in Named Entity Recognition (NER) or Sequence Tagging. These papers formed the backbone of the neural network methods used in the project. The sequence tagging concept was used in both neural network methods, where one method trained Bidirectional LSTM and the other trained a Transformer model. This project uses all these concepts to find the best way to distinguish different kinds of essay statements used in an essay.

3 Data Description

The data used is open-source data obtained from a Kaggle competition, namely "Feedback Prize - Evaluating Student Writing.". It consists of argumentative essays from U.S. students in grades 6 to 12. Expert raters annotated the essays for elements commonly found in argumentative writing.
The training set consists of approximately 15.6 thousand individual essays in a folder of .txt files and a .csv file containing the annotated version of these essays. Some parts of these were not annotated. Figure 1 shows a snapshot of this file. It has the following features:

- id - ID code for essay response
- discourse_id - ID code for discourse element
- discourse_start - character position where discourse element begins in the essay response
- discourse_end - character position where discourse element ends in the essay response
- discourse_text - text of discourse element
- discourse_type - classification of discourse element
- discourse_type_num - enumerated class label of discourse element
- predictionstring - the word indices of the training sample, as required for predictions



id	discourse_id	discourse_s	discourse_e	discourse_text	discourse_type	discourse_type_num	predictionstring
423A1CA112E2	1622627660524.0	8.0	229.0	Modern humans today are always on	Lead	Lead 1	1 2 3 4 5 6 7 8 9 10 1
423A1CA112E2	1622627653021.0	230.0	312.0	They are some really bad consequen	Position	Position 1	45 46 47 48 49 50 51
423A1CA112E2	1622627671020.0	313.0	401.0	Some certain areas in the United Sta	Evidence	Evidence 1	60 61 62 63 64 65 66
423A1CA112E2	1622627696365.0	402.0	758.0	When people have phones, they kno	Evidence	Evidence 2	76 77 78 79 80 81 82
423A1CA112E2	1622627759780.0	759.0	886.0	Driving is one of the way how to get	Claim	Claim 1	139 140 141 142 143

Figure 1: Snapshot of data in train.csv file

4 Methodology

In an argumentative essay, the length and the position of the constituent structural elements vary. As a result, it is challenging to identify the essay portions of these structural components. Three different methods have been used to handle this problem.

4.1 Method 1

This method uses a simplistic approach. Figure 2 shows the steps involved in Method 1.

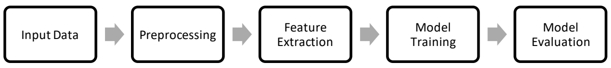


Figure 2: Steps in Method 1

In the preprocessing stage, the essays are broken down into sentences and assigned the corresponding discourse element type they represent. The sentences that do not belong to any discourse element type are assigned the "No Class" label. The data was now ready for feature extraction. TF-IDF (Term Frequency - Inverse Document Frequency) statistic was used for feature extraction. TF-IDF is a numerical statistic used to find the importance of a word to a document in a corpus. Term frequency is the relative frequency of term *t* to a document *d* (shown in Figure 3),

$$tf = \frac{total\ count\ of\ term\ t\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d}$$

Figure 3: Term Frequency formula

where the numerator is the number of times the term *t* occurs in document *d*, and the denominator is the total number of terms in document *d*.
The inverse document frequency measures how common or rare the term *t* is across all the documents. It is the logarithmic inverse fraction of the documents that contain the term *t* (shown in Figure 4),

$$idf = log\frac{total\ number\ of\ documents}{number\ of\ documents\ contating\ term\ t}$$

Figure 4: Inverse Document Frequency formula

where the numerator is the total number of documents, and the denominator is the number of documents where the term *t* appears. As shown in the figure below, TF-IDF is the product of term frequency and inverse document frequency.

$$tfidf = tf\ * idf$$

Figure 5: TF-IDF formula

The extracted features were fed to the Multinomial Naive-Bayes model in the training process. The Naive-Bayes model is a supervised machine learning algorithm/model based on Bayes's theorem, which can be used for both classification and regression problems. This model assumes that the features are independent of each other. This model is simple, easy to build, and can handle vast amounts of data. The Multinomial Naive-Bayes model is a multiclass Naive-Bayes model popularly used in Natural Language Processing. It calculates the likelihood of all labels for a sample, and the output is the label with the highest chance. Once the model was trained on the input data, it was saved and validated on the validation dataset.

4.2 Method 2

Another solution is chunking, which is the process of grouping words or phrases called chunks. Chunking extracts words/ phrases with similar characteristics from structured or unstructured text. With the help of chunking, the trained model can group chunks of text with similar characteristics and categorize them as one of the essay statement types.

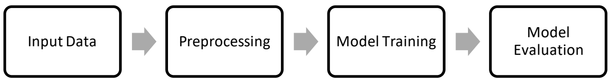


Figure 6: Steps in Methods 2 and 3

Figure 6 shows the steps in Method 2. In the preprocessing step, the essays were broken down into words. Each word was assigned the label of the discourse element type it belonged to in the IOB format. The IOB format stands for Inside, Outside, Beginning format. This format is one of the implementations of chunking. Figure 7 shows the use of the IOB format.

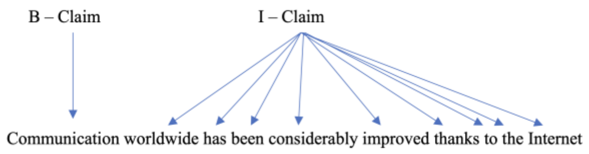

B – Claim
I – Claim
Communication worldwide has been considerably improved thanks to the Internet

Figure 7: IOB Formatting

Word at the beginning of a discourse element was assigned the prefix B- while the rest of the words belonging to that discourse element were assigned the prefix I-. The words that do not belong to any discourse element were assigned the tag O. A label mapper class was created to map the labels in IOB format to integers. A vocabulary mapper class was created to map the words from all the essays to ids obtained from the Roberta tokenizer. Now the data was ready for training.

The Bidirectional LSTM (Long-Short Term Memory) model was used in this method. The LSTM model is a Recurrent Neural Network (RNN) model capable of learning long-term dependencies in data. It is capable of processing input sequences of infinite length. LSTM models were developed to deal with the long-term dependency problem often encountered while training traditional RNN models. They have a chain-like structure with repeating modules of neural networks. There are four neural network layers inside each module, interacting in a particular way. The figure below shows the architecture of an LSTM cell.

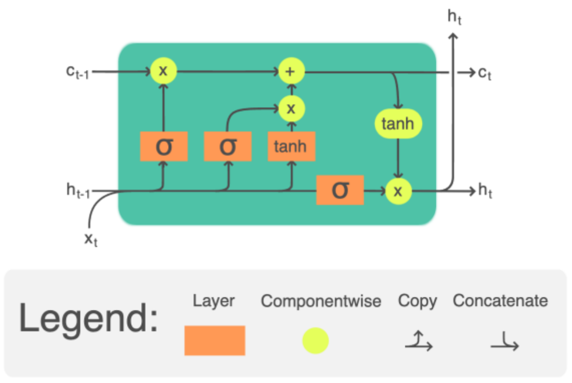


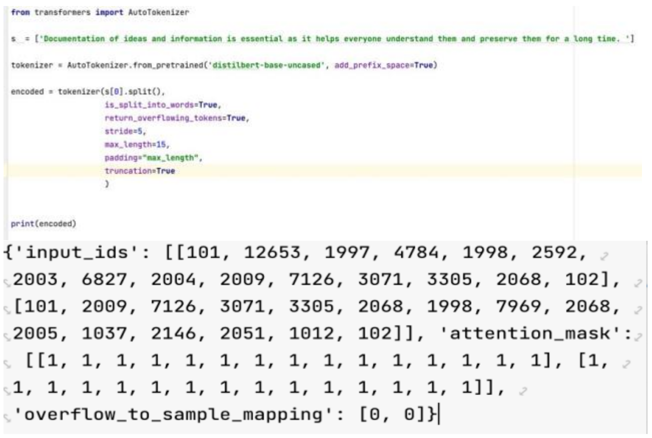
Figure 8: LSTM cell architecture [4]

The horizontal line running through the top of the figure represents the cell state. In a way, the cell state is like a conveyor belt. It runs through the entire chain with minor interactions. LSTMs also have gates that allow only specific information. The figure shows that these gates are composed of a sigmoid neural network layer and a pointwise multiplication operation. A Bidirectional LSTM is a model that consists of two LSTMs where one takes the sequences in the forward direction while the other takes the sequences in the backward direction. This increases the model's knowledge of the sequences.

Since LSTM models can process sequences of infinite length, entire essays were fed in batches to the model along with the labels for each word in the essay. The essays in every batch had different lengths. So, all the essays in a batch were padded to the length of the longest essay in the batch. The same was the case with the labels. The words were then converted to vectors of 100 dimensions using the pre-trained GloVe representations resulting in a 3-dimensional input. The first dimension was the number of essays in a batch; the second dimension was the length of the essays; the last dimension was the length of a word embedding. The embedded inputs were packed and fed to an LSTM layer. The outputs of the LSTM layer were unpacked and passed through a dropout layer before feeding them to a linear layer which acts as a classifier. The resulting output was a 3-dimensional tensor where the first, second, and third dimensions were batch size, length of the essay, and the total number of target labels, respectively. From the probabilities of each label, the label with maximum probability was assigned to the word. Cross entropy loss was used as the loss function, and stochastic gradient descent was used as the optimizer. This entire process continued until the model was trained on all the essays. After training, the model's performance was evaluated on the validation set.

4.3 Method 3

This method involves using Roberta, a Transformer model, and chunking for handling the essays and has the same steps as Method 2, as shown in the figure. The preprocessing step of this method was very similar to that of Method 2. The essays were loaded and spit into words, and each word was assigned a label in the IOB format. These words of each essay were tokenized using the RobertaTokenizer. Unlike the LSTM model, transformers cannot process sequences of any length. The maximum length of the sequences processed by this Roberta model was 512. The essays with lengths greater than the maximum length were truncated to a 512 length. The overflowing parts of the essays were stored as the next entry. To use the context in the training process, the last few tokens of the first part of the essay were repeated in the entry with the overflowing part. On the other hand, the essays with lengths lesser than the maximum length were padded. The code snippet in the figure below shows how the overflowing essays were handled.



```python
from transformers import AutoTokenizer

s_ = ['Documentation of ideas and information is essential as it helps everyone understand them and preserve them for a long time. ']

tokenizer = AutoTokenizer.from_pretrained('distilbert-base-uncased', add_prefix_space=True)

encoded = tokenizer(s[0].split(),
                    is_split_into_words=True,
                    return_overflowing_tokens=True,
                    stride=5,
                    max_length=15,
                    padding="max_length",
                    truncation=True
                    )


print(encoded)
```

```
{'input_ids': [[101, 12653, 1997, 4784, 1998, 2592,
2003, 6827, 2004, 2009, 7126, 3071, 3305, 2068, 102],
[101, 2009, 7126, 3071, 3305, 2068, 1998, 7969, 2068,
2005, 1037, 2146, 2051, 1012, 102]], 'attention_mask':
 [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]],
'overflow_to_sample_mapping': [0, 0]}
```

Figure 9: Code snippet explaining the concept of overflowing tokens.

In the example above, the sentence containing 20 words were passed through a tokenizer that returned tokenized inputs of length 15. Since the number of tokens in the sentence was more than the max length, the sentence was truncated to length 15, and the remaining tokens were put in the next entry. The first and last tags in both the entries were the start and end tags. The last five token ids of the first entry were also present at the beginning of the second entry due to the use of the stride parameter. Since the second entry falls short of the max length of 15, padding was added to the second entry.

After encoding the words, their corresponding labels were also encoded in the same way as the words. The resulting encoded data was the input data for the training process.
The Roberta model and the Adam optimizer were used in the training process. Roberta stands for Robustly Optimized BERT Pretraining Approach [5]. It is a BERT-based model with the same architecture as the BERT transformer but differs in pre-training. Roberta is trained for longer durations with bigger batches over more data; it does not use the next sentence prediction objective; it is trained on longer sequences; it uses a dynamic masking pattern instead of a static masking pattern.
The encoded word-ids and their labels were fed to the Roberta model, and the resulting output was loss and logits. The training accuracy was then calculated using only the active logits. This way, the Roberta model was trained on all the essays. After training the model, it was tested on the validation set to find the best performing model and save the model.

5 Results

To find the best model, the performances of these models must be evaluated using the same performance measure. The data used in this project has a lot of class imbalance due to which accuracy measure cannot be used. Instead, the macro F1 score was used to measure the performance of the models. As each method handled the essays differently, their validation processes were also different but calculated the macro F1 score.
In Method 1, which used the Naive-Bayes model, the labels for the sentences in the validation set essays were predicted. The predicted and actual labels were passed to a SkLearn function to calculate the macro F1 score.
In Method 2, the entire validation set essays were fed to the LSTM model to predict the labels. The prediction process was the same as the training process. The actual and predicted labels of the words of an essay were passed to a SkLearn function to calculate the macro F1 score.
In Method 3, the words with the same predicted label were grouped and recorded along with their word positions in their respective essays. The group of words or chunks with less than four words were removed. The actual and predicted chunks were then compared to calculate true positives, false positives, and false negatives. If the overlap of the actual chunk over the predicted chunk and the overlap of the predicted chunk over the actual chunk were >=0.5, then the prediction was considered a true positive. The unmatched actual chunks were considered false negatives, and the unmatched predicted chunks were considered false positives. The F1 score was calculated using the sums of true positives, false positives, and false negatives. This entire process was done for each class. The resulting F1 scores for all the classes were averaged, giving the macro F1 score for the model.
The table below shows the Macro F1 scores for all the models.

MODELS	**MACRO F1 SCORE**
Naïve-Bayes model	0.1489
LSTM model	0.4726
Roberta Model	0.6237

Table 1: Models performance table

From the performance table, it is seen that the Roberta model was the best.
Using these models, a crude version of the feedback tool was created. Figure 10 shows the process of identifying the essay elements in new essays.

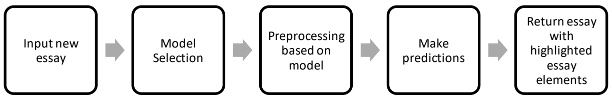

Input new essay
Model Selection
Preprocessing based on model
Make predictions
Return essay with highlighted essay elements

Figure 10: Process of identifying essay elements in feedback tool

The feedback tool accepts the given new essay and prompts to select the model to use for making predictions. If the Naïve-Bayes model is selected, the essay is broken down into sentences, and features are extracted using TF-IDF. Otherwise, the essay is broken down into words and tokenized according to the used neural network model. The resulting processed data is passed to the model to make predictions. From the predictions, the sentences or chunks of the essay representing different essay elements are highlighted in the final essay display.

6 Conclusion

This paper discussed the various methods used to train a model that accurately distinguishes different types of essay statements. The Naive-Bayes model performed poorly, while the neural networks models got decent F1 scores. Among the two neural network models, the Roberta model was the best. From the results of this project, it is evident that transformers can distinguish essay elements most accurately while using the chunking technique. In the future, different neural networks and transformers can be used to find the model with the best performance. Also, a one-source Graphical User Interface application using the best model can be developed with the sole purpose of assisting all students can be developed.

References

1. Xu, S., Li, Y., Wang, Z. (2017). Bayesian Multinomial Naïve Bayes Classifier to Text Classification. In: Park, J., Chen, SC., Raymond Choo, KK. (eds) Advanced Multimedia and Ubiquitous Engineering. FutureTech MUE 2017 2017. Lecture Notes in Electrical Engineering, vol 448. Springer, Singapore. https://doi.org/10.1007/978-981-10-5041-1_57
2. Zhai, Z., Nguyen, D. Q., & Verspoor, K. (2018). Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. arXiv preprint arXiv:1808.08450.
3. Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.
4. LSTM cell architecture. (2022, May 1). Wikipedia. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg
5. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized Bert pretraining approach. arXiv preprint arXiv:1907.11692.
6. Evgeny Chukharev-Hudilainen & Aysel Saricaoglu (2016) Causal discourse analyzer: improving automated feedback on academic ESL writing, Computer Assisted Language Learning, 29:3, 494-516, DOI: 10.1080/09588221.2014.991795
7. Burstein, J., Marcu, D., Andreyev, S., & Chodorow, M. (2001, July). Towards automatic classification of discourse elements in essays. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics* (pp. 98-105).
8. Dikli, S., & Bleyle, S. (2014). Automated essay scoring feedback for second language writers: How does it compare to instructor feedback?. *Assessing writing*, *22*, 1-17.
9. Woods, B., Adamson, D., Miel, S., & Mayfield, E. (2017, August). Formative essay feedback using predictive scoring models. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2071-2080).
10. Hussein, M. A., Hassan, H., & Nassef, M. (2019). Automated language essay scoring systems: A literature review. *PeerJ Computer Science*, *5*, e208.