Data Science Program

Capstone Report - Spring 2022

Feedback Prize – Evaluating Student Writing

Nitin Godi

supervised by

Amir Jafari

ABSTRACT

Writing is one of the most important skills one can have in their armory to achieve significant levels of success. Even so, more than two-thirds of high school seniors are not proficient writers, according to the National Assessment of Educational Progress. These students can improve their 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. The goal for this project would be to create a tool to identify the structural elements of 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. The concept of chunking and the NER technique takes care of this issue.

CONTENTS

INTRODUCTION

Documentation of ideas and information is essential as it helps everyone understand them and preserve them for a long time. Hence, writing skills are one of the most crucial skills to succeed. However, many students lack writing skills. It is hard for their teachers to provide personalized feedback to all students since they outnumber the teachers by a large margin. One solution to this problem is automated feedback tools. These tools can provide personalized feedback to the students almost immediately. Nonetheless, these feedback tools cannot recognize the structural components such as thesis statement and concluding statement in an essay. Moreover, most of these tools are proprietary, making it hard for students from poorer financial backgrounds to use them.

This project aims to address the issues mentioned earlier by creating an open-source automated feedback tool to identify structural components of an argumentative essay to help students improve their writing skills.

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. One 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 structural components of the essay. This model uses chunks of texts and tags representing structural components in the training process.

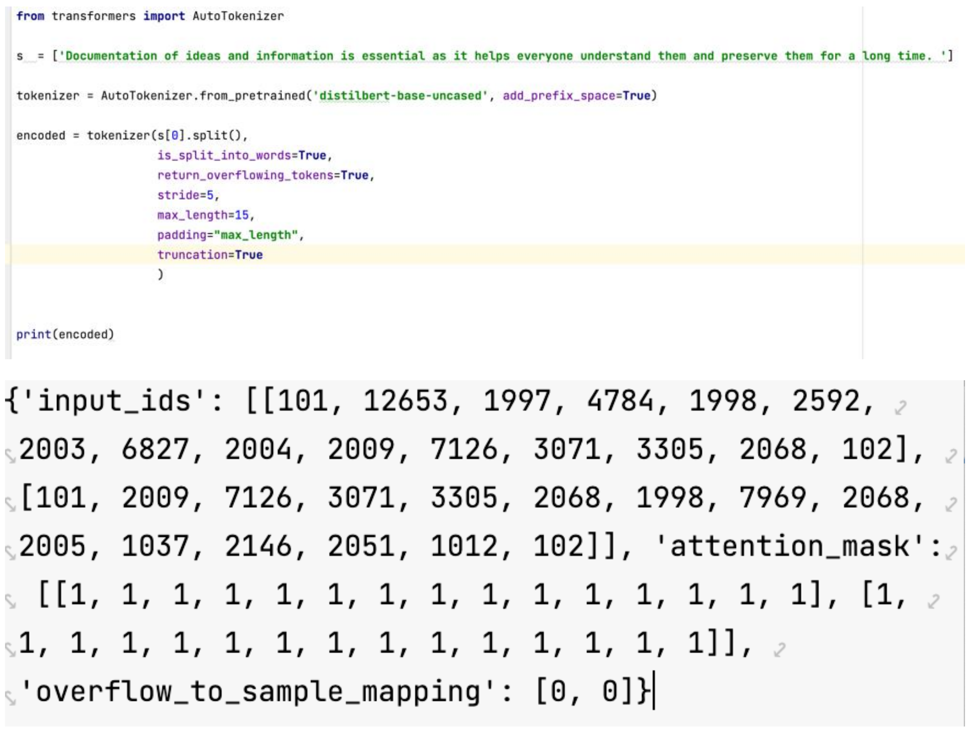
This project uses the data from the Kaggle competition, namely Feedback Prize - Evaluating Student Writing. The dataset consists of argumentative essays from U.S. students in grades 6-12. Expert raters annotated the essays for elements commonly found in argumentative writing. The annotations are documented in a CSV file consisting of the id of the essay, id of the discourse element, start and end character positions of the discourse element, the text of discourse element, discourse type, and the indices of the words that belong to the discourse element.

PREPROCESSING

When the data is downloaded from Kaggle, the essays are stored in text files. All the essays are extracted from the text files and stored in a data frame along with their ids. Each essay is split into words and stored in the data frame. Each word is assigned its corresponding discourse element tag in the IOB format. The IOB format stands for Inside, Outside, Beginning format. At the beginning of a discourse element, the word is assigned the prefix B- while the rest of the words belonging to that discourse element are assigned the prefix I-. The words that do not belong to any discourse element have the tag O. These tags are encoded, and mapping dictionaries are created to map the labels and their encoded values. This dataset is split into 90%-10% training and validation sets.

TOKENIZATION

This project uses transformers for the tokenization and modeling processes. The column containing the essays split into words is passed to the tokenizer. Other parameters such as max length = 512, truncation = True, padding = 'max_length', stride = 200 and return_overflowing_tokens = True are also passed. The max length parameter is passed because most transformers can take a fixed finite number of inputs. The number of tokens for many essays exceeds 512; the overflowing tokens need to be stored and used in the training process. The stride is used to include the last few tokens from the overflowing entry to use the context to predict the discourse type of the overflowing token accurately. The code snippet below shows an example of how overflowing entries are 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 1: Code snippet explaining the concept of overflowing tokens.

In the example above, the sentence containing 20 words is passed through a tokenizer that returns tokenized inputs of length 15. Since the number of tokens in the sentence is more than the max length, the sentence is truncated at length 15, and the remaining tokens are put in the next entry. The first and last tags in both the entries are the start and end tags. The last five token ids of the first entry are 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 is added to the second entry.

Once the essays are tokenized, the encoded labels are added to the tokenized data along with the word ids.

TRAINING

This project uses five transformers to find the best transformer that accurately identifies the discourse elements.

MODELS

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language model developed by Google in 2018. It is a bidirectional transformer pre-trained using masked language modeling objective and next sentence predictions on a large corpus comprising Toronto Book Corpus and Wikipedia. It is a state-of-the-art model that achieves high scores in most NLP tasks without substantial task-specific architecture modifications. It is efficient at handling Natural Language Understanding tasks but not so in the case of text generation tasks. BERT has 12 encoders and 12 attention heads. It uses three kinds of embedding to compute the input representations: token embeddings, segment embeddings, and position embeddings.

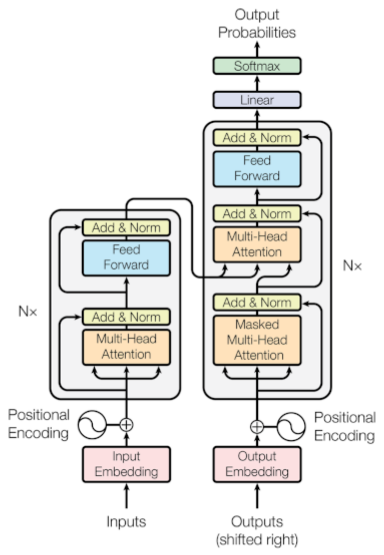


Figure 2: Architecture of BERT model.

DistilBERT

DistilBERT is the distilled version of the BERT model. It is pre-trained by distilling the BERT model. It is a small, fast, cheap, and lightweight transformer with 40% fewer parameters than BERT, running 60% faster while preserving over 95% of BERT's performance as measured on the GLUE benchmark. DistilBERT does not have token-type embeddings, pooler, and retains half of the layers from BERT.

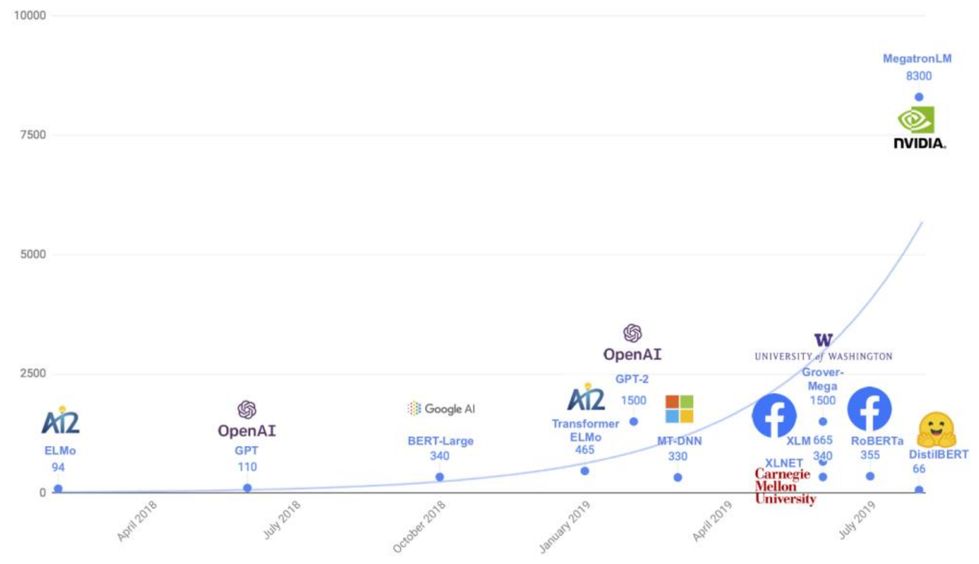


Figure 3: Different transformers with their parameter size in millions.

ROBERTA

RoBERTa stands for Robustly Optimized BERT Pretraining Approach. It is a BERT-based model with the same architecture as BERT 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 dynamic masking pattern instead of static masking pattern.

	BERT	**RoBERT**	**DistilBERT**
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	5% degradation from BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation

Table 1: Difference between BERT, RoBERTa, and DistilBERT

ELECTRA

ELECTRA stands for Efficiently Learning an Encoder that Classifies Token Replacements Accurately. It is a pre-training approach that trains two models: the generator and the discriminator. The generator is trained as a masked language model as it replaces tokens in a sequence. The discriminator tries to identify the tokens that have been replaced. Some tokens are replaced by [MASK] token in BERT, and the model is trained to predict the original token. In ELECTRA, few tokens are replaced by plausible alternatives generated by the generator, and the discriminator is trained to predict whether each token is replaced or not.

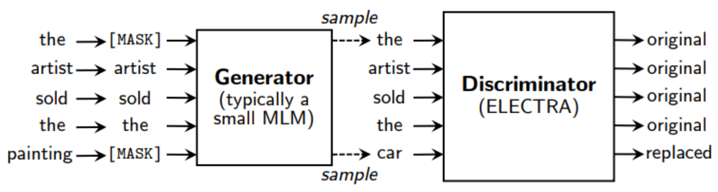


Figure 4: Example of ELECTRA training using a sample

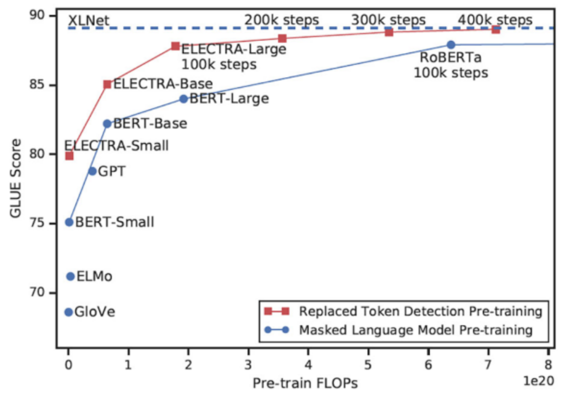


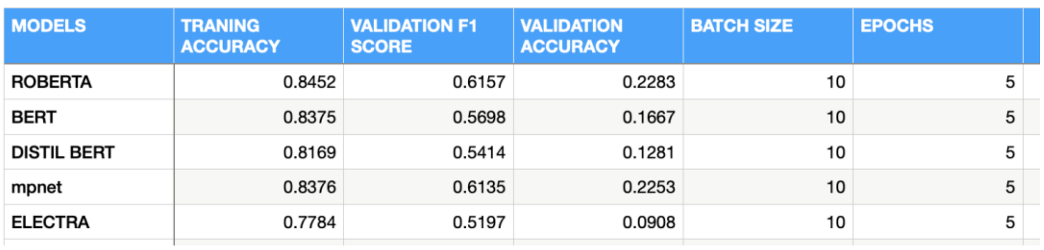
Figure 5: Comparison of ELECTRA model's performance with other transformer models.

MPNet

MPNet is a model that adopts a unique pre-training approach, named masked and permuted language modeling, that inherits the advantages of masked language modeling (MLM) and permuted language modeling (PLM) for NLU. The MLM approach adopted by BERT neglects the dependency among the predicted token. XLNet's PLM approach addresses this issue. However, it does not leverage a sentence's complete position information, which leads to position discrepancies between pre-training and fine-tuning. MPNet manages to inherit the advantages of the two methods leaving out their limitations.

RESULTS

In the validation phase, the resulting output logits were processed and converted to label ids. Predicted label ids were mapped to labels for each essay in the validation set. Chunks of length greater than seven with the same label were stored in a data frame along with the essay id and the ids of words belonging to the chunk. This process is repeated for all the chunks in the validation set. The word ids of the predicted chunks are compared with the word ids of the ground truth. If the overlap percentage in both directions is greater than 50%, it is considered true positive. Any unmatched ground truths are considered false negatives, and any unmatched predictions are considered false positives. True negatives are computed by taking the set difference between the complete set of matched and unmated ground truths and the set union of true positives, false positives, and false negatives. The F1 score and accuracy of the models were calculated using these values. The performance of RoBERTa and MPNet is identical and are the best models among the five models. They are followed by BERT, Distil BERT, and ELECTRA in that order.



MODELS	TRANING ACCURACY	VALIDATION F1 SCORE	VALIDATION ACCURACY	BATCH SIZE	EPOCHS
ROBERTA	0.8452	0.6157	0.2283	10	5
BERT	0.8375	0.5698	0.1667	10	5
DISTIL BERT	0.8169	0.5414	0.1281	10	5
mpnet	0.8376	0.6135	0.2253	10	5
ELECTRA	0.7784	0.5197	0.0908	10	5

Table 2: Performances of different models.