CS3264 Improving essay writing through AI

Jotham Wong Yi Shuen A0235410W*
Praveen Krishna A0235416J*
Wang Zichen A0239376R*

Abstract

Essay writing is an essential skill and a key component of Singapore's education system. This increased focus on writing has resulted in teachers spending longer hours grading, while students struggle with limited access to resources and feedback for improvement. In this project, we aim to reduce the educators' work load and provide students with more detailed feedback and engaging starts to their essays. For qualitative and quantitative AES, we propose using BERT-based tokenizers, RoBERTa, Common Crawl word embeddings, and LSTM+CNN models. For hook generation, we propose using DistilGPT2. To train our models, we use data from the Hewlett Foundation on Kaggle for AES and web-scraped essays for hook generation. Overall, our results show that DistilGPT2 performs best in hook generation; single-layer LSTM displayed optimal performance for qualitative AES; and RoBERTa for quantitative AES among existing transformer architectures.

Introduction

Automated essay scoring (AES) has been studied for more than 50 years due to its commercial and educational benefits for national exams such as the SAT and ACT. Research has primarily targeted the task of *holistic* scoring thanks to the abundance of manually annotated datasets, where the quality of an essay is summarised by a single numerical score. Although AES will allow educators to delegate essay grading to models, the output generated is not useful to students seeking areas to improve. The interpretability of a single score is limited in that students only know how they performed overall rather than in various components of the essay, such as vocabulary, syntax, cohesion, and so on. As a result, researchers have begun examining the more challenging issue of grading an essay based on aforementioned qualitative categories.

In Singapore, the competitive nature of our education scene has pushed students and teachers to further prioritise writing skills. Yet, with teachers overworked and students not receiving enough feedback to improve, neither party can progress. In this report, we explore the potential of AES to benefit both students and teachers. Additionally, we also investigate text generation via neural language models to further aid students in improving one of the most important aspects of an essay: the hook.

Problem Statement

We aim to build two tools that assist students and teachers in the field of essay writing and grading.

First, a system that takes a prompt and an essay as input and outputs 7 scores: overall grade, cohesion, syntax, vocabulary, phraseology, grammar and conventions. For this project, we restrict our system to essays written by students in grades 7 to 10 due to dataset availability. The grading scale of each essay varies and our system is capable of adapting to each scale. This task is henceforth referred to as "grader". We further split our system into the quantitative regression system (that deals with the single numerical score) and the qualitative regression system (that deals with the 6 qualitative features). Students who are in need of qualitative feedback can now grade their essays at their convenience and receive a breakdown of how they did while the tool can help to alleviate the burden on educators. With the Singapore government spending over $10 billion on education each year[1], the commercial value of AES is undeniable. Together with the social benefits it can provide students and teachers, AES has enormous potential in Singapore's education scene.

Second, a system that takes in a prompt and generates a introduction containing the key points for the essay. A good hook grabs the reader's attention, lays the groundwork for the thesis and persuades the reader to invest time and energy in reading the rest of their essay. Hence, a hook can serve as a good entry point for students to build upon and identify good points to use while still allowing them to derive educational value from completing the essay themselves. This task is henceforth referred to as "generator".

Datasets

Hewlett Foundation's ASAP Dataset For our grader, we use the Hewlett Foundation's Automated Student As-

*These authors contributed equally.

[1]https://www.trade.gov/market-intelligence/singapore-budget-2023

sessment Prize (ASAP) dataset released on Kaggle[2]. This dataset contains a total of 12977 essays. These are further split into 8 different sets grouped by a single prompt, each with a different grading scale. Only the final essay set (set 8) contains qualitative scores. Each essay then ranges from an average length of 95 to 600 words and is manually graded by two or three instructors.

GP Essay Dataset Due to a lack of pre-existing datasets, we used BeautifulSoup to scrape the Internet for sample essays. Essays that did not meet the minimum word count of 500 for an A Level General Paper essay are removed. This gives us approximately 60,000 essays to work with, with an essay length distribution of 785/1093/1534 for the 25th, 50th and 75th percentile respectively. In examining the topic distribution of our dataset, we observe that literature has the highest percentage (20.6%), followed by life, social, and social issues at 7% each. Health, business, history, science, and education receive approximately 4-5% each, while the remaining topics, such as culture, art, and philosophy, receive 1-3% each. We aim to release the dataset publicly in the future to benefit other students.

Methodology and Technical Approach

Evaluation Metric

Quantitative Grader We evaluate our quantitative grader's performance using the quadratic weighted kappa (QWK) metric defined below:

$$\kappa = 1 - \frac{\sum_{ij} W_{ij} O_{ij}}{\sum_{ij} W_{ij} E_{ij}}$$

The weight matrix W is calculated by obtaining the difference between the human grader's scores as follows:

$$W_{ij} = \frac{(i-j)^2}{(N-1)^2}$$

O_{ij} is the matrix that denotes the number of essays that received a score i from the first grader and a score j from the second grader while E is an $N \times N$ histogram matrix of expected scores obtained by an outer product of each grader's histogram of ratings, normalized so that E and O have the same sum. QWK ranges from -1 to 1 where 1 denotes perfect agreement between the graders, 0 denotes random agreement and -1 denotes perfect disagreement between the graders. We use QWK as this was the metric used in the Kaggle competition and this would allow us to compare the state of the art transformer models with previous attempts on the same problem.

Qualitative Grader We evaluate our qualitative grader with the mean column-wise root mean squared error (MCRMSE) defined below:

$$\textbf{MCRMSE} = \frac{1}{6}\sum_{j=1}^{6}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{i,j} - \widehat{Y}_{i,j})^2}$$

[2]https://www.kaggle.com/competitions/feedback-prize-english-language-learning/data

Each number from 1 to 6 that j takes on represents a different category that our grader is trying to score. In essence, the MCRMSE is used to calculate the average of all root mean square error within each column. A higher score would reflect poorer performance, and a score of 0 would reflect perfect agreement between the model and human graders. Given the easy-to-compute nature of MCRMSE, alongside its popularity amongst other papers aiming to solve similar problems, this evaluation metric was chosen for the purpose of this paper.

Generator We evaluate our generator using the Perplexity metric defined below:

$$\textbf{PP}(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, ...w_n)}}$$

Under the assumption that our scraped dataset consists of model essays, we want our model to assign high probabilities to sentences in our dataset. Intuitively, we can understand a model assigning high probability to a sentence in our test set as not being **perplexed** by it. This means that our model has a good understanding of how to generate a good thesis in response to a given essay prompt. The equation then simply corresponds to the inverse probability of the test set, normalized by the number of words in the test set, and a lower perplexity showcases how well our model has generalised to the text generation task.

Text-processing

Textual data first needs to be processed into a form suitable for machine learning algorithms. This text processing can be roughly divided into tokenization and feature extraction.

Tokenization Tokenization is the process of dividing raw text into atomic units known as tokens. These tokens can refer to words or characters separated by white space or punctuation. This involves converting sentences to list of words, removing punctuation, tags, stop words (such as "the", "is" that do not have specific semantic meaning) as well as stemming (standardizing words to their root form by reducing their inflections, such as studies to studi and studying to study) and lemmatization (obtaining the root form of the word, such as studies and studying to study). However, tokenization results in the out-of-vocabulary (OOV) problem. As models are trained on a fixed sized vocabulary set, words outside of this set are either ignored or replaced with an unknown token which reduces the model's generalizability and ability to deal with typos. Furthermore, as typos are par for the course among students, the sub-word tokenization technique is required. **Byte Pair Encoding** (BPE) is a data compression technique introduced by Philip Gage in 1994 which now finds use in tokenization to address the issue of open vocabulary (Gage 1994; Sennrich, Haddow, and Birch 2015). To train the tokenizer, a vocabulary of all characters is created. Starting from 1 character tokens, the tokenizer then repeatedly finds the most frequently occurring pair of tokens in the vocabulary and updates the set until the desired vocabulary size is reached. **WordPiece** tokenization (Wu et al. 2016) is an alternative to BPE where instead of choosing the

most frequent pair of tokens, a score is assigned to each pair of characters wherein pairs with lower individual frequencies are prioritized.

Feature Extraction Each token is represented by a n-dimensional vector where the aim is for semantically similar words to be near one another in the n-dimension vector space and for semantically distinct words to be further apart. There are primarily three ways to obtain these input representations, namely one-hot encoding, tf-idf and word embeddings.

However, our problem statement calls for particular attention to methods that are able to capture the **semantic meaning** of the individual words as many English words can typically be used interchangeably (students tend to use synonyms of commonly used words). Both one-hot encoding and tf-idf are unable to truly capture the semantic meaning and similarities between words. As such, our group only considered word embedding approaches. Each token is represented by a fixed sized dense vector and we obtain a word embedding matrix by randomly initializing these vectors and training them through some objective function with the end goal of having semantically similar words be closer. As our problem statement deals with essays which are typically more than 500 words long, ML architectures that can model contextual information and long-range dependencies would do better than architectures that cannot capture such properties. Transformer models make use of the self-attention mechanism (which can be roughly thought of as computing a weighted sum through key, query and value vectors which represent different aspect of the input data to capture contextual information) and are able to better model long range dependencies among the text compared to previous approaches such as Long Short Term Memory (LSTM) networks. This property makes Transformer architectures our best choice for the word embedding strategy.

Most state of the art **Transformer** architectures can be categorized based off their training strategy into masked-language models (MLM) and causal-language models (CLM). CLM are trained to predict the next token in a sequence of tokens, where the model can only attend to tokens on the left, and is thus unable to see future tokens, making it more appropriate for text generation. On the other hand, MLM takes into consideration both left and right context, allowing a holistic analysis of relationships between sentences. The model then randomly replaces a set percentage of tokens with '[MASK]' and is trained to predict these tokens. The bidirectional sequencing allows MLMs to learn stronger contextual representation for sentences, making it more appropriate for downstream tasks such as regression. In both cases, the model's output layer produces logits for all possible words in the model's vocabulary. Since the label is the next word to be predicted, the model can optimize its parameters using the cross entropy loss to maximize the probability of predicting the next word. Through this optimization, the model learns the underlying semantic nature of the words and these trained vector representations can be refined for downstream tasks such as grading essays through fine-tuning.

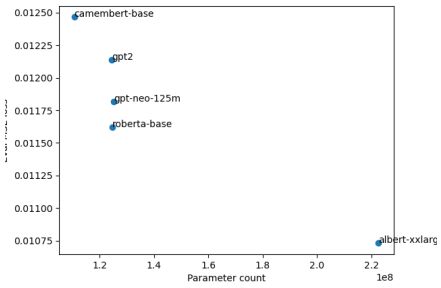

Figure 1: Eval mean squared error vs paramter count

Grader

The Grader architecture first prepends the prompt to the start of the essay. Prior approaches towards the AES problem only perform feature extraction on the essay and pass in an integer representing the identity of the essay prompt. Prepending the prompt should allow our model to generalize better compared to existing approaches as the model now has to process the prompt as well. This is followed by 2 parallel pipelines: the **quantitative** and **qualitative** grader pipelines. The quantitative grader outputs a single real number ranging from 0 to 1 which teachers and students can scale appropriately to the min and max essay mark to get a sense of the overall grade. The qualitative grader outputs 6 real numbers ranging from 0 to 5 for each category, which represent the scores for each of the qualitative features. These scores can then be used by students to improve their essay writing skills, as well as teachers to assist their grading.

Quantitative Grader

Due to the small size of the ASAP dataset (13,000 essays) in comparison to datasets used to train state of the art transformer architectures, we chose to fine-tune existing popular Transformer architectures that could fit on a 32GB RTX 6000 GPU instead of training our own transformer architecture from scratch. Owing to these limitations, our preliminary research led us to consider the following models: CamemBERT, GPT2, GPT-neo, RoBERTa and the ALBERT model. We prioritized MLMs as the literature suggests that MLMs tend to perform better than CLMs for downstream tasks but still chose 2 CLMs to evaluate this claim. We then ran experiments to determine the most cost effective model. This turned out to be the RoBERTa model as seen in **Figure 1**. It had the lowest eval MSE : parameter count ratio, least tFLOPS at 1.02e+16 and lowest training time among all competing models. Both plots of tFLOPS and training time were not pictured as they share the same conclusion as the parameter count plot. Furthermore, the claim that MLMs would strictly outperform CLMs in downstream tasks did not hold true as the GPT models performed similarly as the MLMs models but further experimentation should be performed to test this claim.

Explaining chosen model **Bidirectional Encoder Representations from Transformers** (BERT) is a masked-

language model (MLM) (Devlin et al. 2018) that uses WordPiece tokenization. The model is pre-trained to build deep bidirectional representations from unlabeled text, then fine-tuned with labelled data from downstream tasks. In pre-training, the model is trained on MLM and Next Sentence Prediction (NSP). In NSP, there is a 50% chance the next sentence is replaced with a random sentence from another text, which is then labelled IsNext or NotNext depending on whether it is replaced. **Robustly Optimised BERT approach** (RoBERTa) builds on BERT by using significantly larger mini-batches, training for a longer duration over more data and longer sequences, removing the NSP objective, employing dynamic masking as opposed to BERT's static masking (Liu et al. 2019) and finally employing byte level BPE, where the tokenizer examines byte sequences rather than character sequences. The paper finds that these changes allowed RoBERTa to outperform BERT$_{LARGE}$.

Practical implementation In the quantitative grader pipeline, we tokenize the input with a pre-trained RoBERTa tokenizer and feed it through a pre-trained RoBERTa model that was fine-tuned on the ASAP dataset to output a singular numerical score ranging from 0.0 to 1.0. We used Hugging Face's Seq2SeqClassificationModel and set the output layer to only use 1 head. By normalizing the minimum and maximum scores for each essay set and changing the training objective to the mean squared error function, this allows us to use the logits in the output layer as the model's predicted normalized score to act as a regression model.

Experimental Results After we determined the most cost effective transformer architecture to be RoBERTa, we tried improving the model through hyperparameter search on weight decay for the optimizer as well as the learning rate. The batch size was fixed at 16 and number of training epochs at 30 as it was observed through plotting the validation and training loss that the model started to overfit around this time. The experimental results are shown in **Figure 2**. By using the pretrained RoBERTa tokenizer, we were also able to use the pretrained word embeddings as inputs for simpler ML architectures such as the LSTM, Bidirectional LSTM and Convolutional Neural Networks (CNNs) to act as baselines with QWK as our evaluation metric. All baseline models were simple 2 layer neural networks followed by an output layer for the regression implemented using PyTorch. The LSTM+CNN+Attention model simply used the attention layer as the second last layer. We calculated the kappa score for each essay set (1-8) and obtained the overall kappa score through a weighted average based off set size. Our results supported the wisdom that transformers are "data hungry" as simple baselines such as the CNN-LSTM and LSTM-CNN-attention model outperformed the best performing RoBERTa model drastically, hinting that more manually labelled graded essays should be collated for Transformer approaches in AES to outperform their predecessors for now. Finally, our results showed that the fine-tuned RoBERTa model was robust to hyper parameters as tuning resulted in a kappa score of roughly 0.68. Future improvements to our model would mostly arise from more data collection of manually graded prompts and essays.

	model_name	1	2	3	4	5	6	7	8	kappa
0	LSTM	0.165	0.215	0.231	0.436	0.381	0.299	0.323	0.149	0.275
1	BiLSTM	0.226	0.276	0.239	0.502	0.375	0.412	0.361	0.188	0.322
2	CNN+LSTM	0.821	0.688	0.694	0.805	0.807	0.819	0.808	0.644	0.761
3	LSTM+CNN+att	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
4	lr=5e-05-wd=0.01	0.775	0.638	0.705	0.666	0.798	0.591	0.711	0.59	0.684
5	lr=0.001-wd=0.01	0.771	0.634	0.696	0.672	0.807	0.594	0.709	0.583	0.683
6	lr=0.0002-wd=0.01	0.780	0.636	0.698	0.669	0.801	0.595	0.713	0.583	0.684
7	lr=0.0001-wd=0.01	0.773	0.641	0.692	0.666	0.813	0.604	0.712	0.585	0.686
8	lr=2e-05-wd=0.01	0.786	0.655	0.698	0.663	0.803	0.577	0.695	0.592	0.684
9	lr=0.0005-wd=0.01	0.77	0.604	0.744	0.687	0.831	0.593	0.677	0.567	0.684
10	lr=0.002-wd=0.01	0.766	0.604	0.751	0.684	0.821	0.587	0.669	0.566	0.681
11	lr=1e-05-wd=0.001	0.754	0.624	0.744	0.688	0.823	0.597	0.694	0.528	0.682
12	lr=1e-05-wd=0.05	0.762	0.631	0.747	0.689	0.823	0.584	0.69	0.569	0.687
13	lr=0.005-wd=0.01	0.764	0.608	0.735	0.686	0.827	0.588	0.675	0.558	0.680
14	lr=1e-05-wd=0.005	0.771	0.602	0.742	0.682	0.829	0.583	0.675	0.566	0.681

Figure 2: Results for hyper parameter search on RoBERTa

Qualitative Grader

With our insights from our quantitative grader model training, we opted to use a simpler model for the qualitative grader as our available training data for the qualitative grading is a small subset of the ASAP dataset (only essay set 8) and transformer models simply would not outperform LSTM and CNNs.

We experimented with LSTM in combination with CNN to see if potential synergy with one another could potentially improve performance. The CNN would allow for the extraction of features at the local level (like grammar and vocabulary), while the LSTM would be able to better sense global features such as cohesion, given its ability to process sequential data.

Our qualitative grader model consists of a few parallel layers of CNN + LSTM, before utilizing a few fully connected Dense layers to converge to a final output layer with 6 dimensions as seen in **Figure 3**. Each dimension in the output layer corresponds to 1 of the 6 categories used for qualitative feedback as discussed earlier. Experiments were conducted on variants of this model to explore possible improvements in performance. For the purposes of this grader, we utilized 2 million word vectors trained on Common Crawl[3], released by Facebook's AI Research Lab, due to ease of use and ability to solve the Out-of-Vocabulary problem.

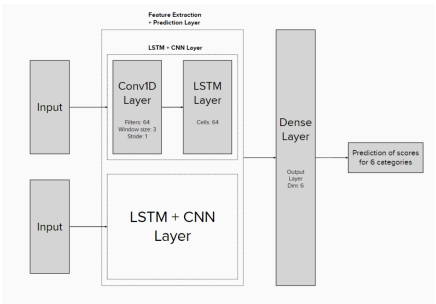

Figure 3: Qualitative Feedback LSTM + CNN Model

Experimental Results We aimed to optimize the LSTM + CNN model for qualitative essay scoring by varying the following parameters: *the number of parallel LSTM layers,*

[3]https://fasttext.cc/docs/en/english-vectors.html

the number of LSTM cells in each layer, whether a CNN was used, and the size of the CNN kernel.

Default Hyper-parameters Unless otherwise specified, all experiments used a learning rate of 0.001, batch size of 32, a convolution window size of 32, kernel stride of 1. The architecture used for the dense layer was [128, 64, 32, 16, 6] where 128 is the dimension of the input layer; 64, 32 and 16 the dimension of the hidden layers and 6 the output dimension for the 6 qualitative scores.

Experiments 1 and 2: Vary no. of LSTM layers and cells Separating the LSTM cells across different parallel LSTM layers may allow the grader to pick up on different features of the essay. On the other hand, increasing the number of LSTM cells in each layer may allow the grader to better remember contextual information within the essay. Therefore, we wanted to explore how varying these parameters may affect the model's performance.

Model number	LSTM cells per layer	MCRMSE		Model number	LSTM cells per layer	MCRMSE
1.1.1	16	0.41		1.1.2	32	0.40
1.2.1	32	0.40		1.2.2	64	0.38
1.3.1	64	0.38		1.3.2	128	0.39
1.4.1	128	0.39		1.4.2	256	0.37

Figure 4: Model hyper-parameters and performance

Models labelled 1.x.1 consisted of 2 layers of LSTM cells, while those labelled 1.x.2 consisted of 1 layer instead. Experiment 1 involved comparison between models labelled 1.x.1 and 1.x.2. Experiment 2 involved comparison within all models labelled 1.x.1 (and similarly within 1.x.2).

Experiment 3: Vary the presence of the CNN layer Since the CNN has an ability to enhance feature extraction and allow the model to factor in contextual information, we wanted to investigate whether the use of the CNN will enhance the model's performance. All the models used had a single CNN + LSTM layer. Models labelled 3.x.x had no CNN layer while the others did.

LSTM cells per layer	Model number	MCRMSE	Model number	MCRMSE
32	3.1.1	0.45	2.1.2	0.40
64	3.2.1	0.41	2.2.2	0.39
128	3.3.1	0.37	2.3.2	0.39

Figure 5: Model hyper-parameters and performance

Experiment 4: Vary window size of CNN An increase in the window size of the CNN should allow the CNN to absorb more context surrounding the essay, thus enabling it to better score the essay. To test this hypothesis, we varied the size of the window within the CNN layer. As with experiment 3, single layer models were used in this experiment, and the LSTM layer had 64 cells.

Model number	CNN window size	MCRMSE		Model number	CNN window size	MCRMSE
4.1	2	0.41		4.4	8	0.46
4.2	3	0.38		4.5	16	0.36
4.3	4	0.39		4.6	32	0.40

Figure 6: Model hyper-parameters and performance

Discussion Surprisingly, all attempts to optimize our models yielded no significant results. This was true even when analysing the RMSE within each category. This may be due to our limited data set, which does not allow the models enough data to exhibit the differences in capabilities. Nonetheless, it is interesting to note that for such small data sets, simple models such as a single layer of LSTM cells with no CNN, are able to exhibit similar levels of performance compared to more complicated models.

Generator

Due to the relatively small size of our GP essay dataset, we opted to fine-tune existing popular Transformer architectures that were around 100 million parameters that could fit on a 32GB RTX 6000 GPU. Furthermore, we opted to use CLMs instead of MLMs for the text generation task as CLMs are trained uni-directionally (in our scenario, on tokens on the left) and intuitively align with how humans typically write text. We chose CLMs for our experiments under similar constraints as our quantitative grader and ended up with four architectures: DistilGPT2, BART, META's OPT and OpenAI's GPT. We trained all models for 30 epochs using Hugging Face's Transformers library on our GP Dataset with a learning rate of 2e-5, a weight decay of 0.01. Since the goal is to train a thesis generator, we extracted the first paragraph from each row and prepended the prompt to the thesis. We then trained the model to output the next word in the thesis given the prompt as the starting sequence (by setting the attention mask to infinity for words in the thesis initially and no mask for words in the prompt). In this manner, we effectively end up with a model that can generate a good thesis given any essay prompt in line with our proposed application. As advised by Hugging Face, all pretrained transformer models were used alongside their corresponding tokenization strategy through the AutoTokenizer and AutoModelForCausalLM classes. The GP Dataset was split into a 90:10 training-validation split and we evaluated each model using the Perplexity metric. The results can be seen in **Figure 7**. Qualitative results were excluded from this report due to page limitations, they can be obtained by emailing Jotham for a copy of the code and model weights. The results show that DistilGPT2 performs the best with a validation perplexity of 2.023. Consequently, DistilGPT2 was used as the model for our gradio (a framework for sharing ML model demonstrations) demo.

Model	Parameter Count	Perplexity
DistilGPT2	82m	2.023
bart-base	96m	2.23
opt-125m	125m	58.28
openai-gpt	117m	4.86

Figure 7: Perplexity results

DistilGPT2 is a more lightweight and compressed version of GPT-2 which uses 82 million parameters instead of 124 million and is twice as fast compared to GPT-2 (Sanh

et al. 2019). It was trained using **knowledge distillation** wherein the student model (DistilGPT2) is trained to produce similar predictions as the teacher model (GPT-2).

Contributions and Reflections

All members had equal and substantial contribution towards the report.

Due to Jotham's prior experience in ML research and engineering competencies, he was the ideal member to work on quantitative regression and thesis generation. Jotham has gained an appreciation for the difficulties in Natural Language Processing, especially text generation. Many nights were spent understanding the theory behind text generation and how this simple idea of training a model to predict the next token given a sequence has resulted in technological marvels such as ChatGPT and the emergent abilities of large language models.

Praveen worked on the qualitative grader using LSTM + CNN architecture. Despite multiple attempts to optimize the grader (many of which were not displayed in this report), the experiments did not bear any fruit. Through this experience, Praveen has gained much insights into the inner workings of LSTM + CNN models, as well as proper methodologies to investigate the optimization of such models. Furthermore, Praveen had first-hand taste of the rigour and setbacks involved in such investigation. Nonetheless, he views this as a good base to continue exploring and experimenting with other architectures for the provision of qualitative feedback.

Zichen worked on data scraping for the generator due to prior experience with data collection. Data sourcing was difficult because many local essay sites had a small number of essays, ranging from 3 to 140. Zichen chose to scrape sites that offer essay writing services because of the larger amount of data they provide and the variety of topics. Some sites, however, had faulty HTML that BeautifulSoup could not handle and had to be handled by a custom parser. A total of 100,000 essays were scrapped, but 40% of them were removed due to repetition or incorrect content. Nonetheless, it was a valuable introduction to data scraping.

Limitations As we are undergraduate students with financial and time constraints, we resorted to LambdaLabs[4] and Linode[5] for their cloud GPUs. Three NVIDIA A10 Tensor Core GPUs and a 32GB RTX 6000 GPU were used for all model experiments. Consequently, there was a restriction on the transformer architectures that we could train due to memory issues.

Ethical Considerations

Educational Value With LMs such as ChatGPT, there is a concern that students may over rely on them to get answers. However, studies find that majority of students believe that this is unethical. Our suggestion poses even less of a threat since we only provide the hook and the student must complete the rest of the essay on their own, effectively stimulating thought and encouraging practice.

[4]https://lambdalabs.com/

[5]https://www.linode.com/

Educators Role As learning models become more sophisticated, there is a concern that educators will be rendered obsolete. This will not be the case. Similar to other models, our proposal will only provide some grading assistance to reduce the educator's workload. We can only provide scoring, so it is up to the educator to provide additional explicit feedback and they may choose to reject the scoring as well.

Bias A lack of variety in training data may lead to bias in our tools. OpenAI stated in the release of GPT-2 that highly represented topics produce better results 50% of the time(Radford et al. 2019). As essays tend to contain a degree of opinion, grading and generation may favour the popular opinion, penalising those who hold the minority opinion despite having a perfectly sound argument. This makes it difficult to identify novel ideas as well due to underrepresentation in data and, as a result, risk penalising and discouraging the student from exploring such ideas.

Conclusion

In conclusion, ML techniques such as LSTMs and Transformer architectures are promising approaches towards building automated essay scoring and thesis generation systems. By leveraging the power of NLP, we can develop systems that reduce the burden and stress on both our educators and students. However, there are still challenges to overcome, such as dealing with bias present in our datasets, and ensuring the ethical considerations of our system. With further research and experimentation, these challenges can be overcome and these systems can be extended beyond Singapore for the public good of our globalized world.

References

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gage, P. 1994. A new algorithm for data compression. *C Users Journal*, 12(2): 23–38.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Radford, A.; Wu, J.; Amodei, D.; Amodei, D.; Clark, J.; Brundage, M.; and Sutskever, I. 2019. Better language models and their implications. *OpenAI Blog https://openai.com/blog/better-language-models*, 1(2).

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.