Automated Essay Grading (AES) with Transformer Embeddings

Suhas Maddali
Masters in Data Science
maddali.s@northeastern.edu

Venkatesh Desai
Masters in Robotics
desai.ven@northeastern.edu

Abstract

Automated Essay Scoring (AES) systems are gaining in popularity due to the time-consuming and inconsistent nature of manual grading. While conventional approaches use various hand-engineered features given to machine learning models, there is a growing popularity and trend towards transformer architectures and deep learning models. Furthermore, they reduce the time taken to hand-engineer features while also improving accuracy. Therefore, our method proposes the use of transformer embeddings with various deep learning and machine learning models for the task of automated essay scoring.

1 Introduction

This report focuses on the task of Automated Essay Scoring (AES), which is a technology designed to automatically evaluate student essays. Writing is an important skill for students, but evaluating essays can be time-consuming and difficult for teachers. AES systems were introduced to alleviate this workload and improve the feedback cycle in the teaching-learning process. AES saves time and money, improves evaluation quality, and reduces bias and human errors. Traditionally, AES systems rely on hand-crafted features designed by experts to measure the intrinsic characteristics of writing. However, recent developments in machine learning (ML) and natural language processing (NLP) have opened the door to new approaches for AES tasks. In this project, we implement AES using transformer-based pre-trained large language models like Bidirectional Encoder Representations from Transformers (BERT), its distilled version DistilBERT, DeBERTa, and GPT-2. These transformers are used to obtain word embeddings which are then trained on CNN, LSTM, BiLSTM and machine learning models like logistic regression, Random Forest Classifier, Adaboost Classifier, K Neighbors Classifier and Support Vector Classifier. We use these transformers from the open-source Hugging Face library to find the word embeddings of the pre-processed data. To accomplish this task, we use the popular Kaggle Dataset - Automated Student Assessment Prize (ASAP), which includes essays written by students in grades 7-10. The dataset contains around 12,948 essays divided into 8 sets, such as Persuasive Essays, Source-based Essays, Narrative Essays, Expository Essays, etc. These essays have been graded by two human evaluators, and an aggregated score feature is used as the target variable for this project.

2 Related Works

The Automated Essay System is comprised of two primary components: the Embedding block and the training block. The system utilizes various models, including CNN, RNN, LSTM, and regression algorithms, as well as machine learning algorithms such as the Random Forest Classifier, Adaboost Classifier, K Neighbors Classifier, and Support Vector Classifier. The Embedding block utilizes a wide range of embeddings, from Bag of Words and TF-IDF to n-grams, glove, word2vec, and transformer-based models such as BERT, ROBERTA, XLNET, and GPT-2. After reviewing several papers, it appears that the progress of AES systems can be categorized into three phases: traditional, Neural Network or Deep Learning, and Pre-trained Large Language Models.

2.1 Traditional

The earliest AES system, Project Essay Grading (PEG), utilized regression methods to predict essay scores using surface linguistic features. Traditional AES systems focused primarily on holistic scoring, which involved extracting predefined surface linguistic features such as morphology, syntax, and semantics, and using them in learning algorithms like linear regression, support vector regression, logistic regression, and Bayesian network classification. Support vector machines were developed to focus on handcrafted linguistic features for automated essay scoring. Meanwhile, open-source engines such as EASE ([13]) treated the essay scoring problem as a regression problem, attempting to find the ground truth scores and predicted scores using a complex regression loss. Traditional AES typically employs regression or ranking systems with intricate handcrafted features for essay rating, such as those used by Yannakoudakis et al. [18] and Chen and He [2]. These features rely on the prior knowledge of linguists and can achieve good performance even with small amounts of data. However, one limitation of traditional approaches is that they often rely on manually extracted features, with deep-level linguistic information being overlooked.

2.2 Deep Learning

In recent times, the neural network approach has brought about a significant change in the AES tasks. This approach eliminates the need for manual feature extraction and allows for the automatic learning of semantic features. Researchers have started to focus on setting up AES models without handcrafted features. Taghipour and Ng [14] proposed a neural network approach based on LSTM for AES tasks, which achieved better results than previous studies. The model used the raw text's word sequence as input and used the convolutional layer to extract features. Dong and Zhang [4] used a CNN model to automatically learn syntactic and semantic features without external preprocessing. Later, Dong et al. [6] demonstrated how the attention mechanism could improve AES accuracy using concentration at the word and sentence levels. Therefore, deep neural networks such as LSTM or CNN can learn complex features of essays automatically, making AES an end-to-end task. Even though the neural network approach has shown promising results, it requires training on a large collection of training data. A combination of traditional and deep neural network approaches can provide better results, as seen in studies by Jin et al. [8] and Dasgupta et al. [3]. However, ensemble methods still require handcrafted features, which can be time-consuming and require significant effort by researchers.

2.3 Pre-trained Language Models

Pre-training for AES involves using a pre-trained language model as the initial essay representation module and then fine-tuning the model on the essay training set. While pre-trained methods have achieved state-of-the-art performance in most NLP tasks, they have not shown a clear advantage over other deep learning methods for AES (such as those used by Dong et al. [6]) in most cases, despite the efforts of researchers such as Uto et al. [15], Rodriguez et al. [12], and Mayfield and Black [10]. The only two pretraining approaches that have surpassed other deep learning methods in AES, to our knowledge, are the work from Cao et al. [1] and Yang et al. [17], who achieved improvement mainly through training optimization. Cao et al. [1] used two self-supervised tasks and domain adversarial training, while Yang et al. [17] combined regression and ranking to train their model. While many researchers have focused on pre-trained language models such as BERT, DistilBERT, ROBERTA, and XLNET, there are relatively few papers on models like GPT-2 and other recently released models that have been used for word embedding tasks.

3 Methods

A large number of research papers concentrated on fine-tuned large language models for the task of automated essay scoring. However, this approach can be computationally expensive with a minimal increase in performance [10]. This paper mentions that fine-tuning BERT or other large language models (LLMs) can give good results but with significant computational costs. Instead, the paper emphasizes the importance of using the embeddings from these pre-trained models (LLMs) to make classification.

In addition to this, the paper also highlights the importance of using a light weight version of BERT known as distilBERT. By using this large language model, it was proven by the paper that the parameter size is reduced by 40 percent to 66 million parameters and reduced model inference times to 60 percent in most cases. This is done with the help of distillation method in which smaller "student" network is trained to reproduce the behavior of a pretrained "teacher" network.

By going through this research, we focused our attention mainly on using the embeddings from large language models and training them on large set of ML models to make the predictions on essay scores. In this way, we got an opportunity to explore the strengths and weaknesses of various language model embeddings and their representations in determining the scores of each of the essay prompts.

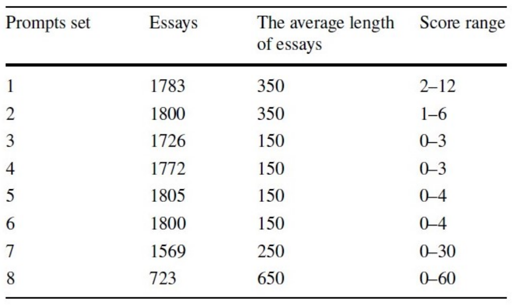
Since there were 8 different essay prompts with different score ranges, it was important to use models separately for each of the prompts to get a good final predictive model with highest quadratic weighted kappa (QWK) values. According to [7] and [12], the emphasis is laid on using large language models for essay scoring. By following the research from these papers, we were primarily interested in exploring the usage of these models.

Since the text was available in a series of rows in our dataframe, we had to perform tokenization to process it [9]. Tokenization is an important part of processing a text sequence before it is fed to a transformer for classification. The type of tokenization can be either word based, character based, or subword-based depending on the type of large language model (LLM) used. Word-based tokenization is the starting point for most of the language models. There are some language models, however, that use character-based tokenization. There are other that rely on subword tokenization which is a combination of the above two mentioned approaches. Therefore, the type of tokenization used is mainly depending on choosing the type of language model for the task of essay scoring.

The large language models (LLMs) that we used in our project are BERT, DeBERTa, distilBERT, and GPT-2. The reason for initially starting with BERT and distilBERT was inspired by [12], [16], [10] . In addition to this, we pursued other models such as GPT-2 and DeBERTa due to their popularity of dealing with textual information in various Kaggle competitions and challenges.

After getting embeddings from large language models, we also hand-engineered a few set of features such as essay length and average word lengths to determine the essay scores. Furthermore, we also used popular vectorization techniques such as bag-of-words (BOW) and term-frequency inverse document frequency (TFIDF) to generate additional features. After getting all of these features, we merged them with the embeddings generated by language models. However, this resulted in a high-dimensional representation with sparse vectors. Therefore, principal component analysis (PCA) was used to reduce the dimensionality while preserving information, leading to a massive decrease in the training times by machine learning and deep learning models.

After getting the embeddings from each of these models along with vectorized and hand-engineered features, we were able to get rich representations of text. After performing this step, we used them as features given to machine learning and deep learning models to perform the task of classification and scoring of essays. The machine learning models used in our approach include logistic regression,



Prompts set	Essays	The average length of essays	Score range
1	1783	350	2–12
2	1800	350	1–6
3	1726	150	0–3
4	1772	150	0–3
5	1805	150	0–4
6	1800	150	0–4
7	1569	250	0–30
8	723	650	0–60

Figure 1: prompts

random forests, adaboost classifier, k nearest neighbors, and support vector classifier. The deep learning models used were CNNs, LSTMs and BiLSTMs for the task of classification.

It is to be noted that we performed k-fold cross validation with the k value being equal to 5. In order words, we divided our data into 5 parts and used each part as the cross-validation data while training the remaining data to observe and note the performance. After using the 5 folds, we used the maximum values of QWK scores to be the final classifier performance. While some papers focus on taking the average QWK scores across all the folds, others rely on using the maximum value obtained as a result of performing k-fold cross validation. We decided to follow the latter approach as it was more convenient and indicative of the performance of classifiers.

By following these approaches, we were able to experiment with a large number of models starting with embeddings of essays using various large language models along with using a set of machine learning and deep learning models to make the final classification of essays. This resulted in gaining results for a large number of models along with the best ones for each prompt based on their quadratic weighted kappa (QWK) scores respectively.

4 Experiments

4.1 ASAP Dataset

For demonstrating the validity of the proposed model, we used ASAP dataset that is a Kaggle competition dataset. ASAP dataset is sponsored by the William and Flora Hewlett Foundation (Hewlett Foundation) in 2012. There are eight sets of essays from different topics and genres. Topics 1 and 2 are argumentative essays requiring writers to state their opinions on a specific topic. Topics 3 to 6 are response essays where writers are expected to read an extract and respond according to the material. Topic 7 and 8 are narrative essays where writers tell a story based on a particular situation. The scoring range varies from each topic. At least two human evaluators grade all essays. The average length of each topic is different, ranging from 150 to 650 words. Table 1 summarizes the prompts and genre of the ASAP dataset.

4.2 Pre-Processing

To prepare the raw essays of the ASAP dataset for analysis, pre-processing techniques such as stop word removal, stemming, lemmatization, and grammar correction are applied. Stop words, which are frequently used but do not contribute to the meaning of the text, are eliminated without affecting its overall meaning. Stemming involves reducing words to their root form, while lemmatization reduces words to their base form based on their part of speech. Both techniques help in standardizing the text and reducing the number of unique words that need to be processed. Additionally, grammar correction using tools such as spellchecker library is used to further refine the data.

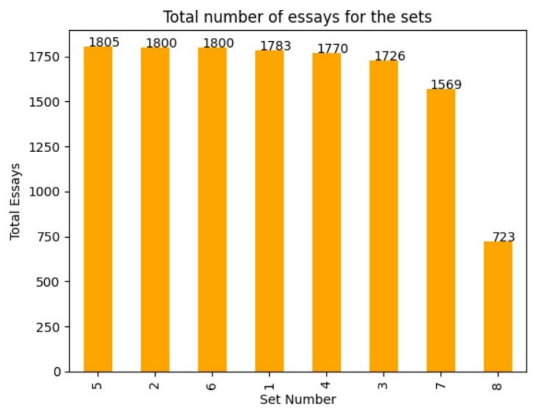


Figure 2: Total number of essays per set

4.3 Implementation Details

As the ASAP dataset was obtained from a Kaggle Competition, we did not have access to a separate test set. Thus, we divided the dataset into 80% training set and 20% test set for model evaluation. To ensure robustness in our results, we employed 5-fold cross-validation, where 60% of the data was allocated for training, 20% for validation, and 20% for testing in each fold. We used the validation data to select the optimal model. To obtain word embeddings for our pre-processed data, we utilized pre-trained models such as BERT, DistilBERT, DeBERTa, and GPT-2 from the Hugging Face library, and obtained the respective model weights and tokenizers. The vector dimension for BERT and DistilBERT is 768, while it is 1024 for DeBERTa and GPT-2. Finally, we employed deep learning and machine learning models for training with the help of GPU's available on the discovery.

The LSTM/Bi-LSTM architecture consists of two layers: an LSTM layer and a Dense layer. The first LSTM or Bidirectional LSTM layer has 400 number of units and the second LSTM layer has 128 number of units. The input sequence undergoes processing with a dropout rate and recurrent dropout rate of 0.2. The output of the Dense layer is activated using the relu function and comprises a single unit. During the training phase, mean squared error is used as the loss function, and the optimizer is rmsprop. Furthermore, the performance of the model is evaluated using mean absolute error as the metric.

The CNN architecture consists of two layers: The first convolutional layer consists of 64 filters with a kernel size of 3, stride of 1, and ReLU activation function. The output of the first layer is then passed through a MaxPooling1D layer with a pool size of 2, which reduces the size of the output by half. The output from the MaxPooling layer is then fed into the second convolutional layer, which comprises of 128 filters with the same kernel size, stride, and activation function as the first layer. The output of the second convolutional layer is then passed through a GlobalMaxPooling1D layer. Finally, the output tensor is passed through a fully connected Dense layer with an activation function of ReLU.

4.4 Evaluation Metric

Quadratic Weighted Kappa (QWK) by McHugh [11] is used to quantify the consistency between raters' scores and predicted scores. The ASAP competition treats QWK as the golden standard for evaluation. Other experiments using the ASAP dataset (e.g. F. Dong et al. [5]) also adopt this evaluation as an indicator of performance. Since we use the ASAP dataset for evaluation in our experiments, we thus regard QWK as the evaluation standard. This indicator usually varies from 0 (no consistency between raters) to 1 (perfect consistency between raters). For details about the QWK formula, please refer to Zhao et al. [19].

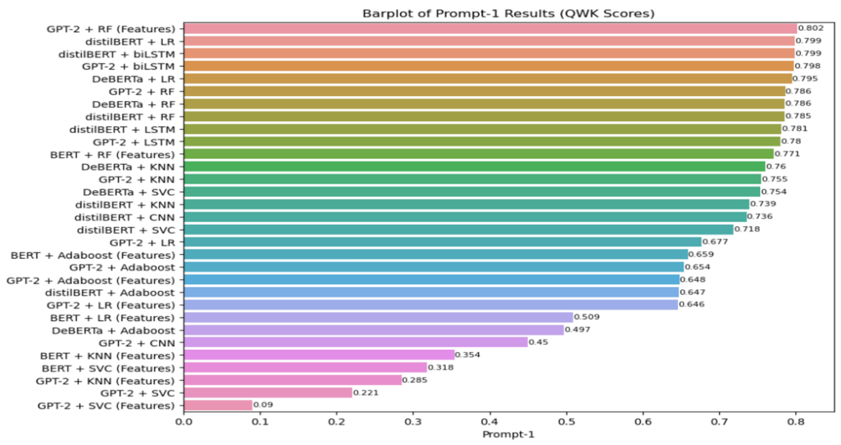


Figure 3: Models Performance for Prompt-1

$$QWK = 1 - \frac{W_{i,j}O_{I,j}}{W_{i,j}E_{i,j}} \tag{1}$$

4.5 Results

After following the list of models in our approach, we could see from the figure that GPT-2 + RF along with hand-engineered and vectorized features provided with the best quadratic weighted kappa (QWK) scores. Despite using the same transformer embeddings (GPT-2), SVC along with features showed far less superior performance. This shows that despite having the same set of features, a list of machine learning models might not always lead to results that are similar with one another. Here is the link to the plots that demonstrate our model results for the remaining prompts: **https://tinyurl.com/y3v63c4n**

The second prompt has a score range between 1-6 and the total set of essays are 1800. The average length of the essays is about 350 words each. By experimenting with a wide number of models, we found that GPT-2 representations along with random forest used as the final classification model was performing the best on the QWK scores. The performance of distilBERT with logistic regression was worthy to mention as it also captures good representations from the text, leading to better essay scoring performance with a good reduction in computation.

The third prompt is relatively short having an average length of 150 words. The score range of the essays are in the range of 0-3 and the total essays in the dataset were 1726. This time, BERT representation with random forest along with vectorized and hand-engineered features was performing the best. This shows that BERT is capable of capturing important representations from shorter texts. It is noteworthy to also see the performance of GPT-2 model with logistic regression as it comes close to the performance of BERT for this prompt.

The fourth prompt is also relatively short and it follows consistent grading range between 0-3. The total number of essays in the dataset are 1772 respectively. The average essay length is similar to the third prompt of 150 words. After experimenting with different models, we found that GPT-2 with adaboost classifier was performing the best. DeBERTa with logistic regression was also also performing quite close to the former mentioned model. This is insightful as DeBERTa is capable to capturing long-term dependencies between words for tasks such as question answering, natural language inference, and text classification.

The fifth prompt consist of score ranges between 0-4 which is different from prompt-3 and prompt-4. However, the average number of words is 150 which is similar to these prompts. The total number of essays were 1805 in the dataset. The best performing model in terms of QWK scores was DeBERTa

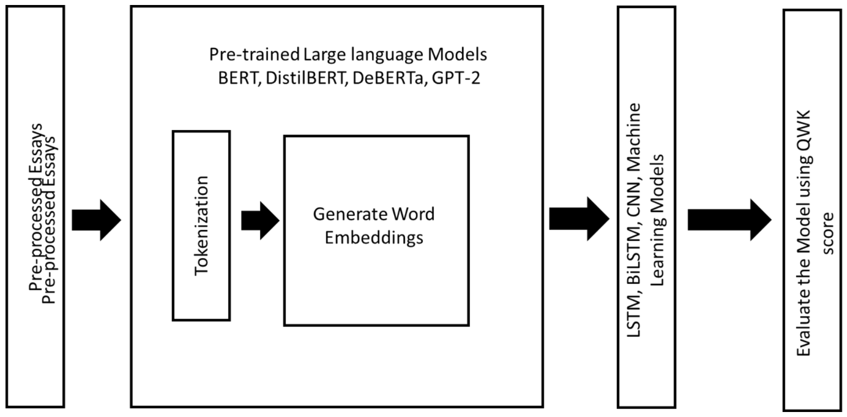
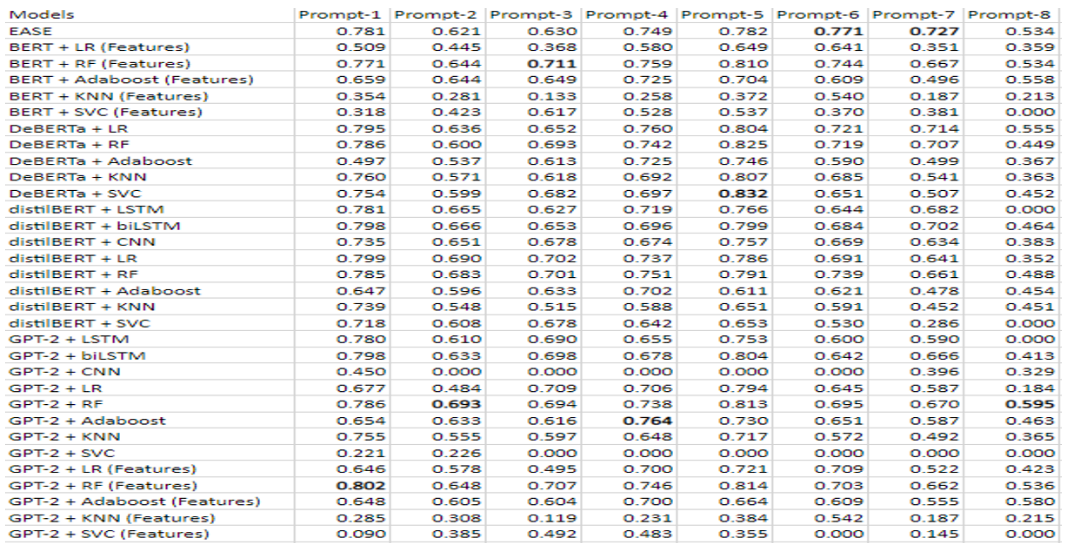

Pre-processed Essays
Pre-processed Essays
Pre-trained Large language Models
BERT, DistilBERT, DeBERTa, GPT-2
Tokenization
Generate Word Embeddings
LSTM, BiLSTM, CNN, Machine Learning Models
Evaluate the Model using QWK score

Figure 4: Architecture Design



Models	Prompt-1	Prompt-2	Prompt-3	Prompt-4	Prompt-5	Prompt-6	Prompt-7	Prompt-8
EASE	0.781	0.621	0.630	0.749	0.782	**0.771**	**0.727**	0.534
BERT + LR (Features)	0.509	0.445	0.368	0.580	0.649	0.641	0.351	0.359
BERT + RF (Features)	0.771	0.644	**0.711**	0.759	0.810	0.744	0.667	0.534
BERT + Adaboost (Features)	0.659	0.644	0.649	0.725	0.704	0.609	0.496	0.558
BERT + KNN (Features)	0.354	0.281	0.133	0.258	0.372	0.540	0.187	0.213
BERT + SVC (Features)	0.318	0.423	0.617	0.528	0.537	0.370	0.381	0.000
DeBERTa + LR	0.795	0.636	0.652	0.760	0.804	0.721	0.714	0.555
DeBERTa + RF	0.786	0.600	0.693	0.742	0.825	0.719	0.707	0.449
DeBERTa + Adaboost	0.497	0.537	0.613	0.725	0.746	0.590	0.499	0.367
DeBERTa + KNN	0.760	0.571	0.618	0.692	0.807	0.685	0.541	0.363
DeBERTa + SVC	0.754	0.599	0.682	0.697	**0.832**	0.651	0.507	0.452
distilBERT + LSTM	0.781	0.665	0.627	0.719	0.766	0.644	0.682	0.000
distilBERT + biLSTM	0.798	0.666	0.653	0.696	0.799	0.684	0.702	0.464
distilBERT + CNN	0.735	0.651	0.678	0.674	0.757	0.669	0.634	0.383
distilBERT + LR	0.799	0.690	0.702	0.737	0.786	0.691	0.641	0.352
distilBERT + RF	0.785	0.683	0.701	0.751	0.791	0.739	0.661	0.488
distilBERT + Adaboost	0.647	0.596	0.633	0.702	0.611	0.621	0.478	0.454
distilBERT + KNN	0.739	0.548	0.515	0.588	0.651	0.591	0.452	0.451
distilBERT + SVC	0.718	0.608	0.678	0.642	0.653	0.530	0.286	0.000
GPT-2 + LSTM	0.780	0.610	0.690	0.655	0.753	0.600	0.590	0.000
GPT-2 + biLSTM	0.798	0.633	0.698	0.678	0.804	0.642	0.666	0.413
GPT-2 + CNN	0.450	0.000	0.000	0.000	0.000	0.000	0.396	0.329
GPT-2 + LR	0.677	0.484	0.709	0.706	0.794	0.645	0.587	0.184
GPT-2 + RF	0.786	**0.693**	0.694	0.738	0.813	0.695	0.670	**0.595**
GPT-2 + Adaboost	0.654	0.633	0.616	**0.764**	0.730	0.651	0.587	0.463
GPT-2 + KNN	0.755	0.555	0.597	0.648	0.717	0.572	0.492	0.365
GPT-2 + SVC	0.221	0.226	0.000	0.000	0.000	0.000	0.000	0.000
GPT-2 + LR (Features)	0.646	0.578	0.495	0.700	0.721	0.709	0.522	0.423
GPT-2 + RF (Features)	**0.802**	0.648	0.707	0.746	0.814	0.703	0.662	0.536
GPT-2 + Adaboost (Features)	0.648	0.605	0.604	0.700	0.664	0.609	0.555	0.580
GPT-2 + KNN (Features)	0.285	0.308	0.119	0.231	0.384	0.542	0.187	0.215
GPT-2 + SVC (Features)	0.090	0.385	0.492	0.483	0.355	0.000	0.145	0.000

Figure 5: Models Performance with Baseline Model

with support vector classifier. Due to the usage of diverse set of models and embeddings, various models performed differently for each of the prompts. The performance of GPT-2 is also exceptional for a large number of essay prompts.

In the sixth prompt, the score is between 0-4 which is similar to prompt-5. The average number of words per essay is 150. The total essays considered for training and testing are 1800. In this prompt, BERT used with random forest along with hand-engineered features was performing the best on QWK score. DistilBERT with random forest also does a great job of competing with BERT. The total number of computations reduced significantly as compared to using BERT.

The seventh prompt consists of scoring range between 0-30 which is unlike other prompts we have seen so far. Therefore, it can be interesting to analyze the performance of the models for this task. The average number of words per essay is about 250 and the dataset consists of 1569 essays. In this prompt, DeBERTa with logistic regression was performing the best. We know that logistic regression is a simple classifier. Therefore, most of the learnings from the text came from DeBERTa embeddings. This highlights the strength of using DeBERTa for essay scoring tasks for certain types of texts.

Finally, the eight prompt consists of scoring range between 0-60. The average number of words in the essays are 650 and the total number of essays are 723. The scoring for this problem can be tough as there is a higher likelihood of not accurately classifying the scores by models as there are a large number of categories. Despite the complexity, we see GPT-2 with random forest is having the highest kappa score and it is significantly better than other models. This shows that GPT-2 is a viable choice for essay scoring as it is giving great results for the majority of essay prompts.

Github Link - https://github.com/suhasmaddali/English-Language-Learning-Prediction-with-AI-and-Machine-Learning/tree/main

References

[1] Yue Cao, Hanqi Jin, Xiaojun Wan, and Zhiwei Yu. Domain-adaptive neural automated essay scoring. In *SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information*, pages 1011–1020, 2020.

[2] Hongbo Chen and Ben He. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752, 2013.

[3] Tirthankar Dasgupta, Abir Naskar, Lipika Dey, and Rupsa Saha. Augmenting textual qualitative features in deep convolution recurrent neural network for automatic essay scoring. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 93–102, 2018.

[4] Fei Dong and Yue Zhang. Automatic features for essay scoring–an empirical study. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1072–1077, 2016.

[5] Fei Dong, Yue Zhang, and Jie Yang. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st conference on computational natural language learning (CoNLL 2017)*, pages 153–162, 2017.

[6] Fei Dong, Yue Zhang, and Jun Yang. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, 2017.

[7] Nigel Fernandez, Aritra Ghosh, Naiming Liu, Zichao Wang, Benoît Choffin, Richard Baraniuk, and Andrew Lan. Automated scoring for reading comprehension via in-context bert tuning. In *Artificial Intelligence in Education: 23rd International Conference, AIED 2022, Durham, UK, July 27–31, 2022, Proceedings, Part I*, pages 691–697. Springer, 2022.

[8] Cancan Jin, Ben He, Kai Hui, and Le Sun. Tdnn: A two-stage deep neural network for promptindependent automated essay scoring. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1088–1097, 2018.

[9] Christian Mayer and Sabrina Ludwig. Automated essay scoring using transformer models. 2022.

[10] Elijah Mayfield and Alan W Black. Should you fine-tune bert for automated essay scoring? In *Proceedings of the 15th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162, 2020.

[11] Mary Louise McHugh. Interrater reliability: The kappa statistic. *Biochem. Med.*, 22(3):276–282, 2012.

[12] Pedro Uria Rodriguez, Amir Jafari, and Christopher M Ormerod. Language models and automated essay scoring. *arXiv preprint arXiv:1911.05267*, 2019.

[13] Lawrence M Rudner and Tahung Liang. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2), 2002.

[14] Kaveh Taghipour and Hwee Tou Ng. A neural approach to automated essay scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1882–1891, 2016.

[15] Masaki Uto, Yikuan Xie, and Maomi Ueno. Neural automated essay scoring incorporating handcrafted features. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6077–6088, 2020.

[16] Yongjie Wang, Chuan Wang, Ruobing Li, and Hui Lin. On the use of bert for automated essay scoring: Joint learning of multi-scale essay representation. *arXiv preprint arXiv:2205.03835*, 2022.

[17] Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Youzheng Wu, and Xiaodong He. Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1560–1569, 2020.

[18] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 180–189, 2011.

[19] Siyuan Zhao, Yaqiong Zhang, Xiaolu Xiong, Anthony Botelho, and Neil Heffernan. A memory-augmented neural model for automated grading. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*, pages 189–192, 2017.