Automated Essay Grading and Inference using Linear and Deep Learning Models

Nishant Velagapudi, Beau Kramer, Pavan Kurapati
University of California, Berkeley, W266 - Natural Language Processing with Deep Learning
{nishray, kramerbeau, pkurapati}@berkeley.edu

Abstract

Free text responses to prompts are thought to quantify the level of subject mastery at the highest point of Bloom's taxonomy. Grading and providing feedback to text responses manually can be expensive and imprecise. We train models to automatically grade essays using Hewlett-Packard's Kaggle dataset titled "ASAP." We benchmark performance from a suite of linear models as well as CNN, LSTM, and feed-forward network, finding that stacked LSTM models provide the most predictive power. Random Forest models achieve a performance closest to the optimal score. Predictions are interpreted using the LIME package to generate useful feedback for writers.

1. Introduction

Practical attempts at automated essay scoring date to the 1990s when computers became powerful enough to perform the necessary computations. Peter Foltz and Thomas Landauer from the University of Colorado at Boulder leveraged Latent Semantic Analysis to begin development of the Intelligent Essay Assessor in the 1990s [1]. Around the same time, the Educational Testing Service developed its e-Rater system to score essays written for the GMAT [2]. Historically, the approach to this problem is to engineer hundreds of features that logically indicate text quality and model grades using those. By contrast, modelling essay grades through deep learning means that the features are discovered automatically, potentially trading transparency for performance.

Our first goal is to create/optimize essay grading algorithms that perform on par with the Kaggle competition leaderboard. Our second goal is to report and analyze features that are important to grading for feedback purposes: this will rely on linear modelling for transparency.

2a Data Processing & Output Metric Concerns

The Hewlett-Packard essay grading dataset contains 12976 essays. These essays are divided into 8 separate prompts. Each prompt has a distinct topic and a unique grading scale: these scales can range from 0-3 to 0-50. One of the key challenges of using the Hewlett-Packard essay grading dataset involved normalizing grading to leverage the entire set for network development.

Initial attempts at using this dataset involved either classifying on a prompt-by-prompt basis or using only 6 of the 8 prompts and normalizing all scores into a range of 0-12. Classification and regression architectures failed. This led us to normalize each score to a 0-1 range. We re-inflate and round the predictions to nearest integer before calculating metrics.

The standard measure for this task is the quadratic weighted Kappa, which quantifies the agreement between ordinal scores. The quadratic Kappa function includes weighting for "partial credit" on mismatched classes [3]. We divided the training data into an 80-20 train-test split and all metrics reported are predictions made on the test set. Unless otherwise noted, we pad/truncate sequence length to the 95th percentile of training sentence length. This is for processing speed: the maximum essay length is almost twice as many tokens as the 95^{th} percentile of length. We include other metrics to contextualize performance: RMSE is the most easily understandable metric to judge prediction quality (quantifying the average prediction error in this case). However, since the scale of predictions changes across datasets, we track the RMSE on the actual score as well as RMSE on normalized score.

3 Feature Engineering

There is a developed literature around typical features to engineer for this task [4], [5]. We attempted to engineer as many of these features as possible. We grouped our features into 6 broad categories and describe them in detail below.

3a. Length Features
These simple features capture the length of different components of a text. We created features to count the **essay length** in tokens, words, characters, and sentences. Prior work suggested that the **fourth root of character count** to be highly correlated with essay grade [5].

3b. Occurrence Features
These features count either the presence of or the count of certain tokens. For our purposes, we engineered features to count the number of **periods**, **commas**, **exclamation marks**, **question marks**, **semicolons**, and **colons**.

3c. Syntax Features
We utilized parse trees to engineer features that capture syntactic quality. Features include **sum**, **mean**, **max**, and **variance** of parse trees in an essay**.** We also captured the **average sentence length** and **variance of sentence lengths**. We refined these measures by counting the number of sentences of various lengths in words to capture **very short**, **short**, **medium**, **long**, and **very long** sentences.

3d. Lexical Features
We captured the lexical quality of an essay with several features. First, we counted the **vocabulary size** of each essay. To this, we added **counts of various word lengths**. We also calculated the **mean** and **variance of word length**. Finally, we computed the **type token ratio** which is a ratio between the count of unique tokens in an essay divided by the total tokens in an essay.

3e. Style Features
With our style features we attempted to address several elements of style. First, we computed **Yule's K** for each essay as a measure of its lexical richness [6]. Next, we counted the number of **connectives** in an essay from a fixed list to capture its flow. Finally, we engineered several features with **part of speech n-grams** to capture style and some grammatical errors. We computed a ratio comparing the number of unique part of speech n-grams to the total number of part of speech n-grams in an essay. We also computed the **mean tf/TF of part of speech n-grams**. (tf is the term frequency in a single essay and TF is the term frequency in the entire essay corpus). We create these features for tri- and four-grams. We expect a high tf/TF to be a grammatical and/or fluency error.

3f. Reading Scores
We finally included a set of 8 reading level metrics. This is a more direct approach to the problem as these metrics purport to capture the sophistication of a piece of text. These are computed using counts and ratios of syllables, word lengths, and sentence lengths in an essay. We included Flesch-Kincaid Grade Level, Flesch Reading Ease, Gunning Fog Index, Coleman Liau Index, Automated Readability Index, Lix, Gulpease Index, and Wiener Sach Textformel [7].

4 Linear Modelling

For the purpose of contrast with our neural networks, we used linear models with our engineered features. We used 3 models: linear regression, random forest (RF), and support vector regression (SVR). The RF and SVR models have large combinations of possible hyperparameters that can greatly change performance: for these models, we performed a grid search using 3-fold cross validation for model tuning.

Model	Spec	RMSE	Mean Quadratic Kappa	Accuracy

Linear Regression	C=1.0	2.02	0.5865	0.476
Random Forest	**500 trees, depth of 44, 3 samples/leaf**	**1.55**	**0.7072**	**0.531**
Support Vector Regression	C=0.1, gamma='scale'	1.87	0.5625	0.476

Table 1 - test set metrics for 3 linear model specifications. Random Forest is the best performer by a significant margin.

We observe that Random Forest provides the best performance across all metrics. This is followed by Support Vector Regression and Linear Regression. That said, the performance of the Linear Regression was surprising and speaks to the power of the engineered features.

Dataset	RMSE	Normalized RMSE	Quadratic Kappa	Accuracy
1	0.838	0.065	0.814	0.500
2	0.573	0.104	0.698	0.709
3	0.668	0.192	0.611	0.632
4	0.654	0.189	0.696	0.628
5	0.597	0.130	0.789	0.661
6	0.747	0.169	0.640	0.561
7	3.054	0.121	0.724	0.172
8	4.15	0.081	0.679	0.095

Table 2 – Random forest test set metrics split by prompt.

Including 8 distinct prompts in the training set means that validation needs to also be split out to the original 8 sets. We want to understand how the model is performing in each of the original segmentations of the data. We show this in Table 2 for the Random Forest model. We can see a comparable performance by Normalized RMSE for essay sets 2-7. Performance by quadratic Kappa is somewhat mixed with essay sets 1 and 5 having the strongest performance.

In Figure 1, we show the relative importance of the top 10 features. We observed that simple length and lexical features were relatively more important than more complex features. However, more complex features did contribute marginally. To meet our goal of incorporating interpretability, we utilized Local Interpretable Model-agnostic Explanations (LIME) [8]. Sample LIME output for our manually extracted features are shown in Figure 2. Hypothetical feedback for this essay would be to improve the writer's diction. Features corresponding to word choice, vocab size, mean word length, and variance of word length were all detractive to their score.

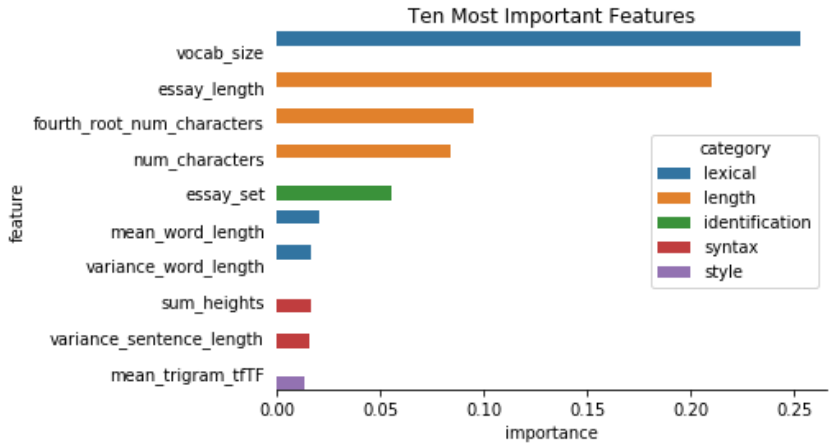

Figure 1 - Top 10 feature weights - labelled by name and colored by category of feature

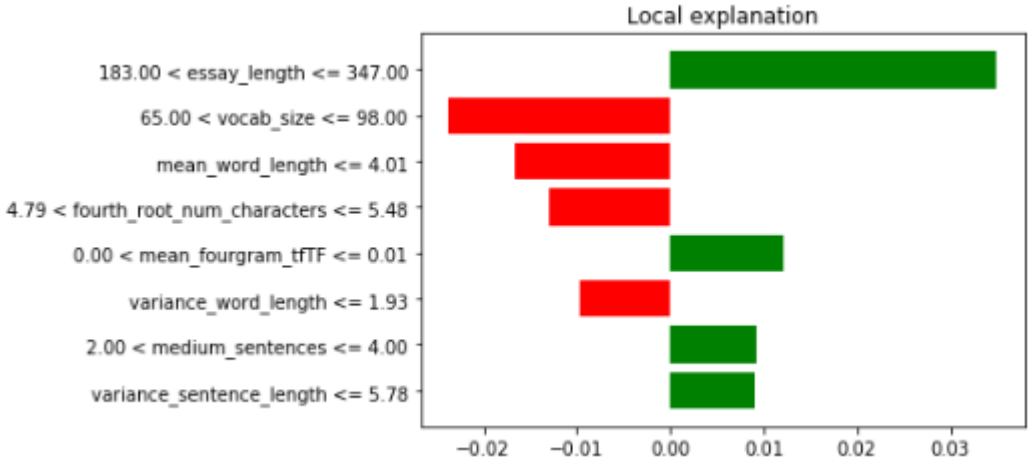

Figure 2 – Sample essay local explanation - contributors and detractors from score

5 Deep Learning for Whole Essay Scoring

We specified 3 types of networks for whole essay scoring. First, we used deep feedforward neural networks with varying numbers of nodes and hidden layers. Second, we evaluated a convolutional neural network (CNN) directly fed into an output layer again with varying numbers of convolutional layers. Finally, we used a recurrent neural network (RNN) with long short-term memory (LSTM) cells: we varied the number of LSTM layers stacked. We chose LSTM cells in our RNNs because long-range token interactions could potentially be significant in the essay architecture and such interactions are best captured by LSTM [9]. We evaluated three architectures for feedforward networks, RNNs, and CNNs in both a classification and regression paradigm (mean squared error loss for regression, categorical cross-entropy for classification). We found regression on normalized output scores to produce the best results for CNN and RNN architectures, while classification was better for feed-forward networks. In all further steps, we use regression (MSE loss). We used the Adam optimizer for faster convergence in all runs, with batch sizes of 500 and 35 total training epochs (as loss had universally reached a plateau by this point).

Model	Details	RMSE	Mean Quadratic Kappa	Accuracy	Trainable parameters
CNN	1 layer, 1 pooling	2.55	.585	0.439	105,409

	2 layers, 2 pooling	3.07	.512	0.411	138,035
	3 layers, 3 pooling	2.47	.570	0.421	105,165
RNN	[20]	1.86	0.651	0.478	25,701
	[20,20]	1.77	0.674	0.503	28,981
	[20,20,20]	1.89	0.687	0.520	32,261
	[32,32,32]	**1.72**	**0.708**	**0.532**	**59,297**
Feed Forward (class.)	100 Units	2.28	0.595	0.498	3,028,951
	[100, 50] Units	**2.40**	**0.596**	**0.484**	**1,534,601**
	[100, 50, 25] Units	2.49	0.543	0.437	786,176

Table 3 - Test set performances by neural network architecture. Recurrent architectures are most performant and have the least trainable parameters. Best in each class is bolded and the overall best model is in green.

5a. Feed-forward network performances:

We note that increasing the number of hidden layers while reducing the number of nodes per layer decreased the total trainable parameters since the final layer connections are reduced with the dimensionality of the last hidden layer. We can see comparable performances for each architecture, but the best performance is observed with two layers (50, 25 units) between embedding and output.

5b. Convolutional Neural Network Performances:

We note that complexity is maximized in the 1-layer CNN architecture - where the only convolutional layer has 75 different filters with a stride length of 5. Flattening this layer and fully connecting it an output layer thus generates significantly more trainable parameters than the alternative architecture which contain additional convolutional and pooling layers before fully connecting to a dense layer. The 1 convolution network had the best mean quadratic kappa score despite RMSE and accuracy scores worse than the 3-convolution network (possibly due to worse metrics in the essay prompts with a wider range of scores). This suggests that more credit was given for incorrect predictions in this network.

5c. Recurrent Neural Network Performances:

We observe that the best performance is given by 3 stacked LSTMs. The RNN architectures are favorable because there are significantly fewer trainable parameters than in the feedforward and CNN architectures. We observe the highest validation set scores using RNNs, likely due to the ability of LSTM cells to capture long range token interactions. We added an additional 3 layer stacked LSTM architecture to derive better performance by increasing the depth. Since the LSTM architectures had significantly fewer trainable parameters than others, added a second 3-layer stacked architecture with greater output dimensionality. We also observed an increased gap between training accuracy and validation accuracy and thus added a dropout layer to this more complex architecture. We also found this more complex stacked LSTM model was more performant without trimming input sequences.

5d. Inference distributed by dataset

As before with the Linear Models, we want to understand how the model is performing in each of the original segmentations of the data. We focus on our best performing model: the 3-layer stacked LSTM regression model with 32.

dataset	RMSE	Normalized RMSE	Quadratic Kappa	Accuracy
1	1.072	0.086	0.691	0.346
2	0.680	0.120	0.598	0.606
3	0.616	0.191	0.702	0.682
4	0.465	0.179	0.881	0.783
5	0.634	0.142	0.772	0.605
6	0.586	0.138	0.825	0.679
7	3.109	0.123	0.721	0.132
8	5.240	0.104	0.476	0.094

Table 4 - test set metrics split by prompt. We observe significantly worse performance in prompt 8.

We can see comparable performance on sets 2-6 in terms of actual RMSE as well as comparable performance between sets 1-7 by the quadratically weighted Kappa score. By all metrics, predictions for essay set 8 are poor.

6 Essay Scoring at a Sentence Level

We hypothesized that predicting scores at a sentence level could help address the gap in scoring by dataset. We divided each essay into individual sentences and tagged the sentences with the overall essay score, resulting in ~140k sentences in the training set. After scoring the sentences, we would merge the sentence scores back into one essay score. We used min, max and rounded mean functions to collapse sentence scores to essay scores. We observed that rounded mean functions were most performant. We also tried classifying the sequence of predictions to the overall score using a LSTM. Sequence classification performed far worse than simple functions to collapse sentence level scores.

Model	Details	RMSE	Mean Quadratic Kappa	Accuracy	Parameters
FF Neural Net	[50] units	2.14	0.319	0.361	2,625,101
	[50,50] units	2.13	0.333	0.380	2,627,651
	[50,50,50] units	2.20	0.318	0.358	2,630,201
GRU	Dim=32	1.994	0.396	0.413	32,001
	Dim=50	2.05	0.361	0.385	32,001
CNN+FF NN (1 layer)	kernel=5,FF Dim = 100	2.23	0.280	0.382	365,065

CNN + LSTM + FF NN	kernel=5, LSTM dim=100,FF NN dim = 100	2.08	0.362	0.395	172,265
Stacked LSTM	**3 layers: LSTM dimensions of 32**	**2.01**	**0.399**	**0.406**	**59,297**

Table 5 - test set metrics when predicting the score by sentence. We note that LSTMs yield the best performance, but the margin is much smaller.

We observe that 3 stacked LSTM layers outperform all other models in this problem set up. We observe that the test set metrics vary less by model in this approach, but the top results are not on par with essay-level classification. Dataset level metrics for the sentence level 3-layer stacked LSTM are below as well.

Dataset	RMSE	Normalized RMSE	Quadratic Kappa	Accuracy
1	1.369	0.110	0.253	0.308
2	0.785	0.142	0.349	0.532
3	0.775	0.259	0.253	0.407
4	0.616	0.257	0.756	0.627
5	0.823	0.203	0.451	0.469
6	0.809	0.215	0.515	0.494
7	4.119	0.163	0.279	0.082
8	5.164	0.102	0.335	0.051

Table 6 - test set metrics split by prompt when classifying individual sentence scores. We are unable to achieve uniform metrics through this approach.

We observe similar performance to the prompt level results at an entire essay level and we concluded that this approach does not help equalize model performance across various prompts.

7 Combining Deep Learning with Feature Engineering

We hypothesized that we could capture the encodings from our CNN and RNN networks and use them in tandem with the engineered features in Section 3 for maximal accuracy. Outputs from the best performing CNN and RNN architectures were concatenated with the new features and fed into a final fully connected layer before an output was predicted. The architecture is detailed below:

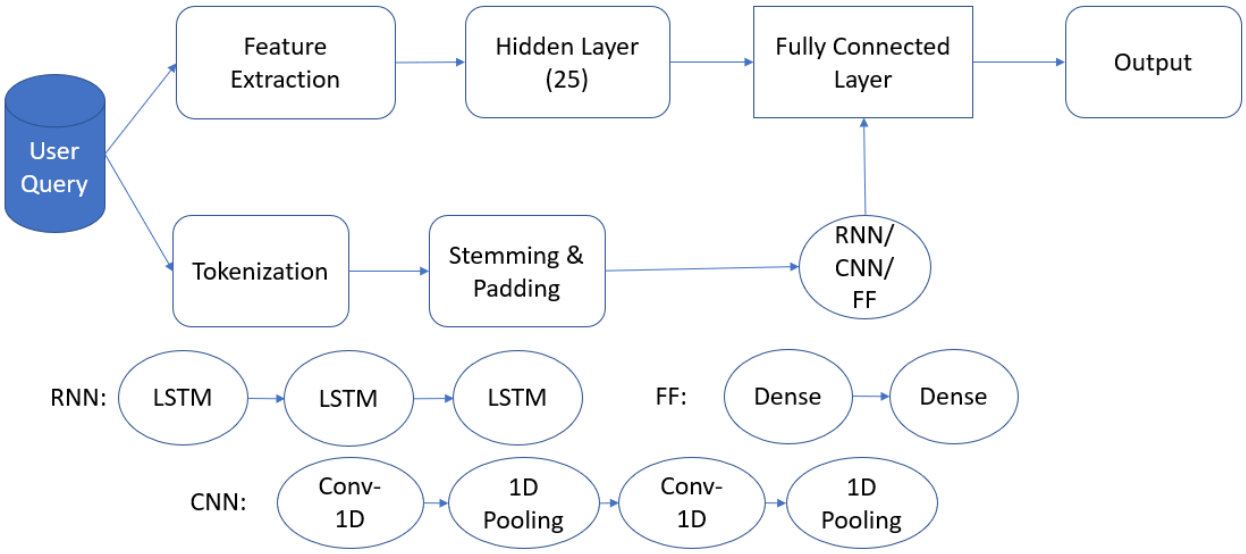

Figure 2 - Architecture diagram of combination of neural network architectures with manually extracted features

We observe that concatenating hidden layer outputs from the manually extracted features to the highest performing RNN architecture leads to nonsensical outputs. Concatenating second hidden layer features from feedforward networks led to a slight deterioration in performance. We note that the CNN and RNN architectures were regression networks, while the feed-forward network was set up in the classification paradigm (reflecting what was most successful for these architectures independently).

	RMSE	Mean Quadratic Kappa	Accuracy	Trainable Params
Feed-Forward	2.45	0.555	0.484	1,536,726
Single layer CNN	7.36	0.282	0.314	33,136
3-layer RNN	24.58	0.05	0.09	115,628

Table 7 - test set results when combining deep learning with manually defined features. We observe extremely poor performance.

We conclude that adding manually extracted features to the learned features worsens performance for each of the most performant architectures. The feed-forward network appears to have been worsened the least.

8 Error Investigation

We aimed to understand where the models were producing errors. First, we correlated the rater error in the test set between the random forest and 3-layer recurrent neural network errors. We find a Pearson correlation of .69 – suggesting that there is a reasonably strong, positive relationship between the errors made in the models.

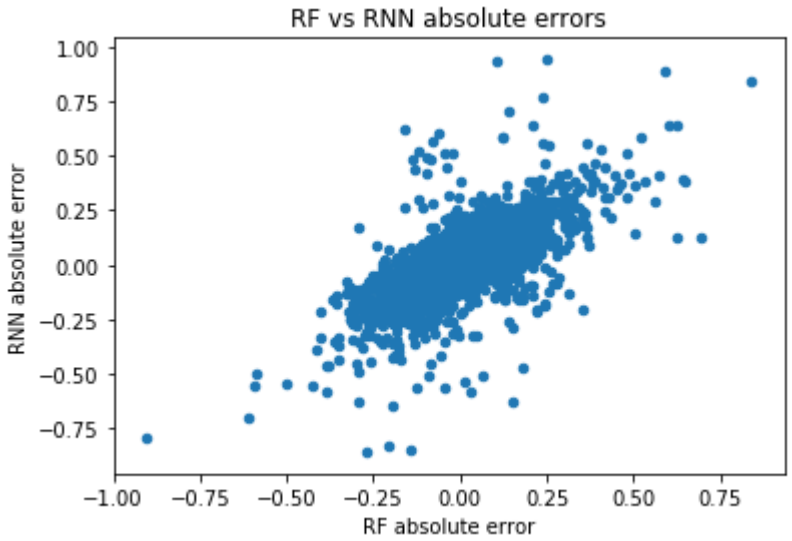

Figure 4 - plotting RF error against RNN error. We see a strong, positive trend between the two, correspond to the correlation of .69

We manually investigated interesting errors in predictions. First, we inspected the 5 cases where the two models disagreed the most. All 5 of these errors belong to essay set 4, which asks the student to explain why the author of an essay concludes with a certain paragraph. Three of the 5 investigated essays were longer: these scored well by the RNN. Two of these were well-written and had high marks, while one contained several errors and received no points. The other two essays were very short and received no points. These were predicted to have high grades by the RF model, reinforcing the observation that essay length is heavily relied on by the RF model.

We next read essays where the RF model errs most. The cases where the RF predicts a higher score generally did not address the prompt in the essay. They rambled about details but only addressed the topic indirectly. The RF model's reliance on length features seems to make it vulnerable to such exploits. We suspect there may be grading errors in this dataset: where RF committed the greatest errors, the essays were very short and mistake ridden but still received full marks.

Next, we examined the largest errors in the RNN. Once again, it appears that the essays are very well written, but are only tangentially related to the prompts themselves. There is one outlier in these five examples: the essay is very poorly written and is scored as such by the RNN. Despite several errors and a failure to address the prompt, the essay appears to have earned full marks. We again suspect grader error in this case.

Our sentence based deep learning models did not perform very well compared to the essay-based models. A quick analysis on the top 5 worst performing essays revealed that the model predicted "high" scores for these whereas the actual scores were very low. All the essays were extremely small in length implying that sentence-based models performs poorly for short length essays. Since the model grades at a sentence level, both the sentences got high grades resulting in higher mean score.

9 Conclusion

The HP ASAP Kaggle competition did not post the validation/test labels for us to compare the metrics for these datasets. Since the competition is officially closed, we can no longer submit our model to truly compare our results to the leaderboard. However, we found that the winning models had a mean quadratic weighted Kappa of around 0.75 [10], [11]. This was calculated by taking an average of the quadratic weighted Kappa of each individual essay set. Our best linear model using Random Forest with extracted features produced a mean quadratic Kappa score of 0.707. Our best non-linear model using 3-layered stacked LSTM produced 0.708. We find that these results are comparable, given that we held out 20% of the dataset for validation purposes. Kaggle competitors would have used all provided data for training and evaluated models on test data that is reserved in the competition.

Automated essay grading offers the ability to reduce the time cost in grading essays as well as the potential bias of having multiple human graders as evaluators. We have obtained promising results using both linear models and deep learning. Using packages like LIME, we can offer feedback to writers to improve their submissions if predictions are made with linear models. We believe that with additional work, these technologies can reduce the expense and inherent bias in the human grading of essays.

There are natural follow-ups to this work. First, expanding the dataset would likely yield significant reward in terms of validation accuracy. Second, our error investigations have shown clear patterns in errors for both the RNN and the RF models. Further work could directly target these clusters of errors. Finally, a greater parameter search and architecture exploration could yield better results than observed here.

11 References

[1] Landauer, Thomas K. Darrell Laham, and Peter Foltz. 2003. Automatic Essay Assessment. *Assessment in Education,* 10(3):295-308

[2] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. *Proceedings of the 17th International Conference on Computational Linguistics*, 206-210.

[3] Cohen, J. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220.

[4] Zesch, Torsten et al. 2015. Task-Independent Features for Automated Essay Grading. *BEA@NAACL-HLT* 224-232

[5] Chen, Hongbo and Ben He. 2013. Automated Essay Scoring by Maximizing Human-Machine Agreement. *EMNLP* 1741–1752

[6] Yule, G. U. 1944. *The Statistical Study of Literary Vocabulary*. Cambridge UniversityPress.

[7] Miltsakaki, E. and Audrey Trout. 2008. Real Time Web Text Classification and Analysis of Reading Difficulty. *The Third Workshop on Innovative Use of NLP for Building Educational Applications*, 89-97.

[8] Ribeiro, Marco Túlio et al. 2016. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier.” *HLT-NAACL Demos* Retrieved from https://arxiv.org/abs/1602.04938

[9] Chung J, Gulcehre C, Cho K., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Retrieved from https://arxiv.org/abs/1412.3555.

[10] Hamner, Ben 2011. Re: Congratulations and DC Conference. Retrieved from https://www.kaggle.com/c/asap-aes/discussion/1840

[11] Shermis, M. D., & Hamner, B. 2012. Contrasting State-of-the-Art Automated Scoring of Essays: Analysis. Retrieved from https://storage.googleapis.com/kaggle-forum-message-attachments/2422/NCME2012Paper3_29_12.pdf

Appendix:

Exploratory Analysis

We note that essay sets 7 and 8 have different prompts than the other 6 sets. The first 6 sets elicit responses to targeted questions or a passage of writing, essays 7 and 8 ask the student to tell a story (narrative writing). While the elements of good writing are potentially similar, we can see in the provided scoring keys for each essay prompt that prompts 7 and 8 have grading criteria that are not shared with the other sets (e.g. "was the story on topic"). We highlight two interesting features we found while exploring the data. We can see that although a longer essay, in words or number of sentences, does not guarantee a high score, it does coincide with fewer occurrences of low scores.

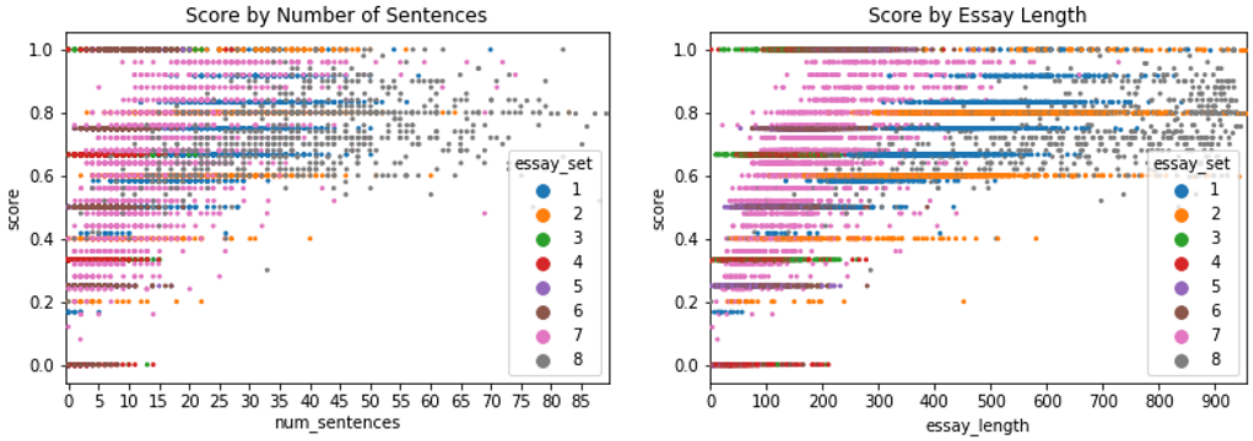

Figure A.1- Essay scores by number of sentences (left) and number of tokens (right). Very short essays exclusively score badly, but long essays are not guaranteed to score well.

GPU performance gain:

We trained all evaluated architectures on an nvidia-tesla-v100 GPU through gcloud. Training time comparisons are below:

Model	CPU			GPU		
	1 layer	2 layer	3 layer	1 layer	2 layer	3 layer
CNN	~23s	~24s	~24s	<1s	<1s	<1s
RNN	~30s	~50s	~70s	5s	~22s	~20s
Feed Forward	~7s	~10s	~10s	~0s	~0s	~0s

With this significant speedup across all architectures, we took our most performant models and allowed for greater training (100 epochs instead of 35 with a batch size of 1000 instead of 500. We observed no improvement in test set metrics with additional training.