Neural Networks for Automated Essay Grading

E4040.2018Fall.GGOP.report

Ariel Cohen Codar ac4391, Brian Midei bmm2172, Marko Mandic mm5305
Columbia University

Abstract

This project introduces a multi-layer perceptron (MLP) and a recurrent neural network (RNN) to generate grades for essays. The automation of the grading process for standardized testing essays can reduce the cost of labor and remove potential bias. Pretrained GLoVe word vectors are utilized to produce useful essay representations. The models presented in this paper manifests the ability to produce correct grading for essays ranging 100 to 500 words in length.

1. Introduction

Previous methods for automatic essay grading have largely focused on manual feature selection as a way of extracting key information from written texts in order to assess them accurately. Such attempts have resulted in divergence between the predicted grades and the true grades given by human judges. The difference between the automated and human assessment is measured using quadratic weighted kappa values (QWK). A QWK value of 1 represents a complete match in grading and is therefore the most desirable outcome, while a QWK of 0 is the least desirable, representing no improvement over random guessing. In 2006, Attali, et. al were able to achieve a QWK of 0.5 using traditional machine learning techniques. [2] In 2012, a Kaggle competition presented by The Hewlett Foundation to complete this task was dominated by models using predefined features scoring best QWK values in the range 0.81 to 0.83. [3][4]

Nguyen and Dery were among the first to use a neural network approach to tackle this problem. [5] They used a neural network architecture to obtain a QWK value of 0.94, a significant improvement over the previous work. The goal of this project is to reproduce the multi layer perceptron (MLP) and recurrent neural network (RNN) results presented in that paper as well as to experiment with different architectures that might further improve results.

2. Summary of the Original Paper

The paper submitted by Nguyen and Dery presents various neural network solutions for this task of automated essay grading. For word vector representation, they considered pre-trained GloVe word vectors, Term Frequency Inverse Document Frequency, and Trained Word Vectors. All three of these methods provide an element of context to the representation of each word. They found that the strongest word embedding was a GloVe embedding that was trained on the dataset of interest. This approach is the most desirable because relationships between words are trained using the precise dataset that the network will later attempt to learn. However, this approach is more time-intensive than simply using a pretrained model, and it also requires a great deal of data to accurately train the word embeddings.

Nguyen and Dery used multiple Artificial Neural Network (ANN) architectures to tackle the task of essay grading. They implemented a two layer MLP, a single layer Long Short-Term Memory (LSTM), a deep LSTM, and a bidirectional LSTM. Figure 1 shows the architecture choices from the original paper.

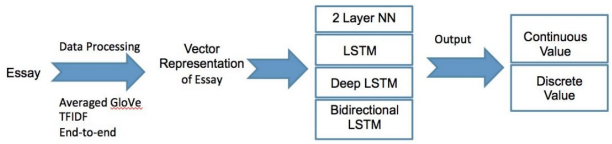

Figure 1: Original Paper Implementation

In addition to the multiple ANN architectures used, Nguyen and Dery experimented both with classification and regression. Interestingly, they found that a softmax cross entropy classification model performed better than a regression model.

All of the models presented in the paper were able to achieve a QWK value of at least 0.91. Surprisingly, the model with the best score was a two layer MLP that used trained word vectors and it reported a QWK value of 0.94. This result was important for two main reasons. First, it showed that an LSTM-based model is not necessarily better than a simple MLP, even for complicated sequential tasks such as processing and evaluating essays. This contradicts much of the prevailing wisdom in the field of neural networks that problems involving sequential data are best suited for RNNs. Second, this result demonstrated the power of neural networks and deep learning over traditional machine learning approaches, which were only able to achieve QWK values of 0.83 at best.

3. Methodology

We replicate the results the from Nguyen and Dery using pre-trained GloVe vectors and two different ANN architectures: a two layer MLP and a single layer RNN. For the RNN model we experiment with both LSTM and Gated Recurrent Unit (GRU) cells. Figure 2 shows the implementation used for this project. We decided to omit term frequency inverse document frequency as well as training our own GloVe word vectors due to our lack of knowledge in Natural Language Processing (NLP). We also omitted testing deep and bidirectional LSTM due to the complexity of these models.

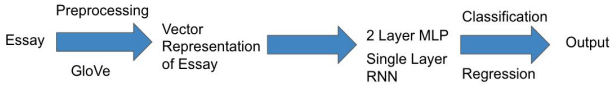

Figure 2: Our Implementation

3.1. Objectives and Technical Challenges

The objective of this report is to reproduce results detailed in the original paper and attempt to improve upon these results with newer neural network technologies such as GRU cells.

The first major challenge for this project is handling the data appropriately. The data is divided into 6 distinct sets, each of which contains essays responding to a particular prompt and a unique grading scale. Therefore the first issue is deciding whether to standardize the scores and group all the essays or to build a separate model for each essay set. Additionally, the data distribution of the labels is not even. Figure 3 shows the grade distribution for each set of essays.

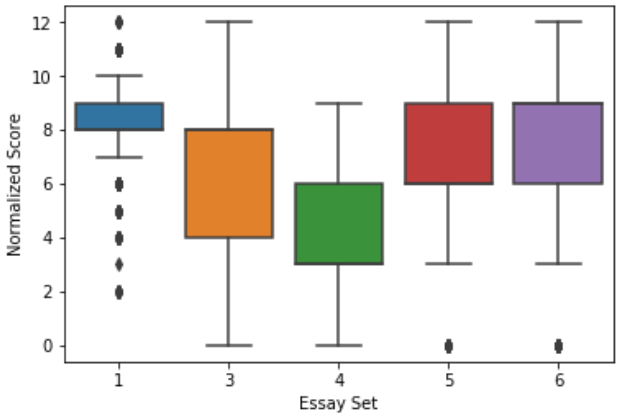

Figure 3: Boxplot of the Score Distribution

Most of the essay scores are close to the mean, meaning that essay scores near the extremes of the grading scale are underrepresented. Possible solutions to this problem include 1) generating more underrepresented data, 2) ignore some of the overrepresented data, or 3) use the unevenly distributed data unchanged.

A particular challenge with essays is that they are inherently variable length. Therefore, we must decide whether to 'pad' the essays, use some averaging methods, or implement models that take in variable length inputs.

Another major challenge is to find suitable word representations of the essays that can match the desired architecture. A one-hot encoding of each word in the essay is the most obvious, however this representation is quite memory inefficient, requiring a large (~400,000 word) dictionary of words. Additionally, one-hot encodings fail to capture context of a word.

A computational challenge present in many neural network projects is to obtain enough capacity and memory to run the models. In this case, an essay of length 400 words and a word vector embedding of size 200 is represented by a tensor of size (80,000,1). Even when running on a cloud computing instance with a GPU, this large of data can cause issues. During development, we must make a tradeoff between a larger word embedding and the memory constraints of the computing machine.

Finally, the main goal of this project is to test the performance of GRU cells compared to LSTM cells. Therefore, a key concern is creating RNN models that can switch between these two parameters easily and having them produce comparable results.

3.2. Problem Formulation and Design

One characteristic of the dataset is a large discrepancy in word length for the various essay sets. The essay sets are also the products of different question prompts to the students. We implement an individual model for each essay set to limit the scope of the network. Additionally, this allows us to use the actual grading for each essay set instead of normalizing the scores to a common scale.

To reduce the overrepresentation bias in the most common essay scores, we limit the number of essays from each class. Furthermore, we implement right padding to essays to a value that captures most of the essays but also reshapes them all to a uniform size. We utilize pre-trained GloVe word vectors without context vectors to generate vector representations of essays. Using pre-trained GloVe vectors for each word scales down the input of each word to a fixed-length word vector and also capturing the context for each word.

When running regression models, our task is to minimize the objective function Mean Squared Error:

$$\mathrm{J} = \frac{1}{N}\sum_{i}^{N}(\hat{y}^i - y^i)^2$$

For classification, we use Categorical Cross Entropy to assign probability of each essay belonging to a class (essay grade):

$$CE(y, \hat{y}) = -\sum_i y_i log(\hat{y}_i)$$

The architecture of the MLP is shown in Figure 4. For the activation function, we use ReLU as opposed to tanh, since ReLU has become much more popular since the writing of the paper. The number of units in the hidden layers is standardized to [1024, 256] but can be increased substantially without overfitting the data because the inputs are very large. We use an Adam Optimizer with batches of size 16.

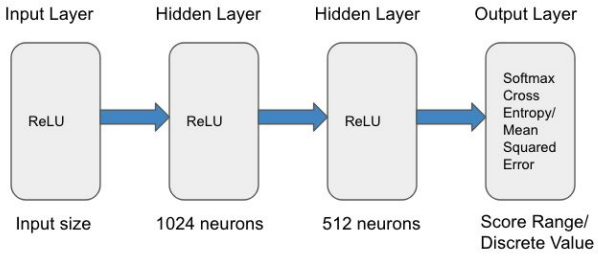

Figure 4: MLP Architecture

The architecture of the RNN model is shown in Figure 5. We implement the models using both LSTM and GRU cells. We simplify and use a single RNN layer with 128 units instead of using a layer with 50 and a layer with 32 units like the original paper states. We use an Adam Optimizer with batches of size 32.

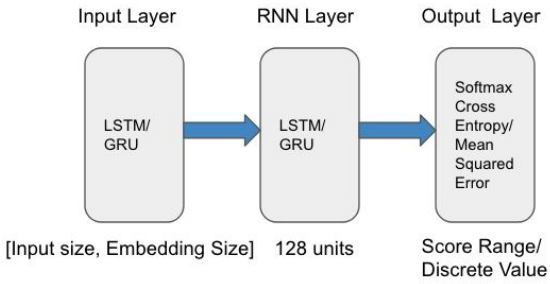

Figure 5: RNN Architecture

4. Implementation

This section will outline the data preparation techniques used and the design iterations of the MLP and RNN network architectures.

4.1. Data Preparation

The Kaggle dataset provided essays from 8 different essay prompts, referred to hereafter as "essay sets". The dataset also provided, amongst other things, the essay set ID and score received by the student. A major component of any NLP neural network is to preprocess the data into a form which the network can consume.

The first step of data preparation is to remove superfluous information from the dataset. Information such as traits of the graders who produced the scores is removed. Additionally, essays from sets 2, 7, and 8 are removed for the training purposes. These sets have unique characteristics such as inconsistent grading scales and variable question prompts. In order to limit the complexity of the network training, we remove these sets from the scope of this report. Therefore, only essays in sets 1, 3, 4, 5, and 6 are considered.

The next step of data preparation is to convert the text of each essay, represented as a string, to a numerical representation. This is done by first separating the essay string into individual words and punctuation. Next, stop words - articles, punctuations, and conjunctions - are removed because they carry little meaning in this context. After that, a pre-trained GloVe word embedding is used to convert each word to a vector of real numbers. A word embedding is preferred over a simple one-hot encoding for two reasons. The first is that the resulting input to the network is much smaller (A relatively small dictionary is 400,000 words, meaning the input to the network is a vector of size 400,000). More importantly, however, the word embedding is designed not only represent each word, but also to preserve important context for that word. For this report, embedding vectors of size 100 and 200 were chosen, however other sizes are available.

An additional consideration for a neural network training set is that the inputs need to be a standard size. In the case of the essays, they are obviously variable length. Therefore, it is necessary to 'pad' each essay with vectors of zeros such that the size of each essay was consistent. To do this, we calculate the mean essay length from each set, and remove any essays that were unusually long (essays lengths larger than one standard deviation above the mean). Then all essays are padded to the maximum length of the remaining essays. Note that this is done separately for each set, due to large discrepancies in length for the different sets. Shown in Figure 6 is a visualization of word counts for each set.

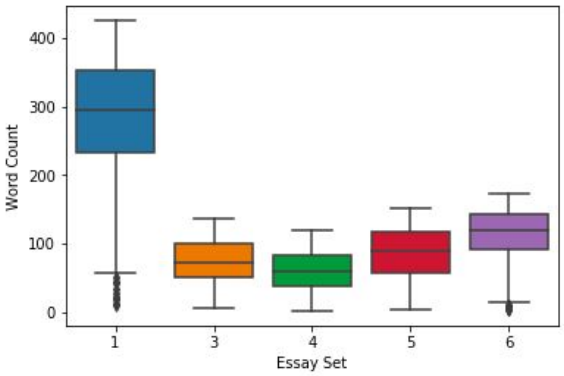

Figure 6: Word Counts

The final step in preprocessing is to separate the essays into training and testing sets. This is done using a 80% train, 20% test split. The resulting training set has 5970 essays while the testing set has 1493 essays.

4.2. Deep Learning Network

The original paper used both an MLP and LSTM to predict essay scores for student-written essays. This report primarily attempts to reproduce the results submitted by Huyenn et al. using the MLP and LSTM approaches. A secondary objective is to improve upon these results with the use of a GRU RNN, something that had not been proposed at the time of the original paper.

4.2.1. MLP

Considering the most simple neural network architecture, we began with an MLP to gain an understanding of the baseline performance we could attain with a neural network approach. Following a similar network architecture to Huyenn and Dery, we used a 2 layer network.

Initial attempts were made with smaller hidden dimensions of 256 and 64 for the first and second hidden layers, respectively. However, this network failed to decrease training or testing loss, ultimately performing about as well as a random number generator. A plot of training loss and validation loss over the training period is shown in Figure 7. It is clear that while the network is able to learn the training set well, it is unable to generalize to the validation set.

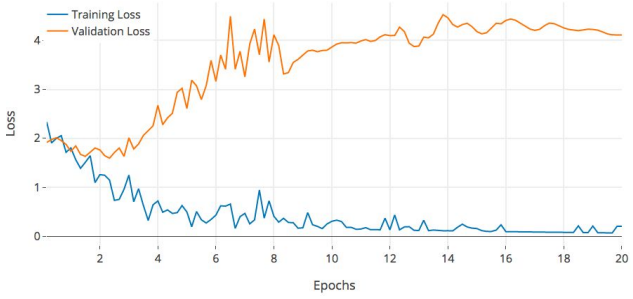

Figure 7: Poor MLP Training and Validation Loss

We reason that such a large input size (~42,000 input neurons) needs a larger capacity model to effectively learn the features of the dataset. After a few design iterations, the best results are obtained with hidden dimensions of 1024 and 256 neurons for the first and second hidden layers, respectively.

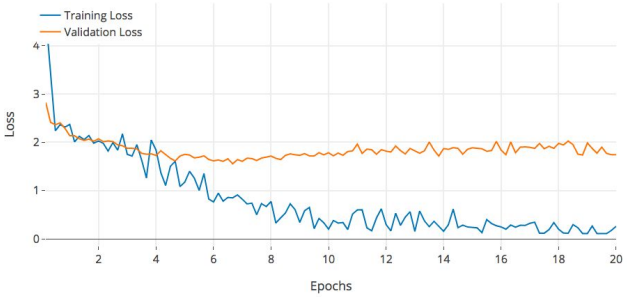

Figure 8: Improved MLP Training and Validation Loss

Another consideration for tuning the MLP is the choice of optimizer. We first tested using Stochastic Gradient Descent (SGD). This optimizer was effective, however recently SGD has decreased in popularity in favor of variable learning rate optimizers such as Adam. We had the best success in training time and accuracy using the Adam optimizer.

Due to a divergence of training loss and validation loss as the network training progressed, a few adjustments are made to improve the performance on the validation set. First, dropout on the hidden layers is implemented to improve generalization on the validation set. A keep probability of 0.8 proved effective in limiting the divergence of the training loss and validation loss. Additionally, early stopping is implemented by saving the model with the best performance against the validation set. In doing so, we are able to avoid a model that may have lower training loss, but does not generalize well to the validation set. A plot of training loss and validation loss with the final MLP network design is shown below. Note that this particular test was run with essay set 1, but similar results were obtained with all other essay sets.

4.2.2. LSTM

The other network architecture used by Huyen et al was the LSTM. This network architecture is attractive because of its ability to 'remember' previous states, thus making it a powerful tool for sequential data. The following shows the framework containing the gates of a typical LSTM cell:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

An LSTM architecture with 1 hidden layer and 128 LSTM cells was the initial choice. This model performed quite well. The training and validation loss is shown in Figure 9.

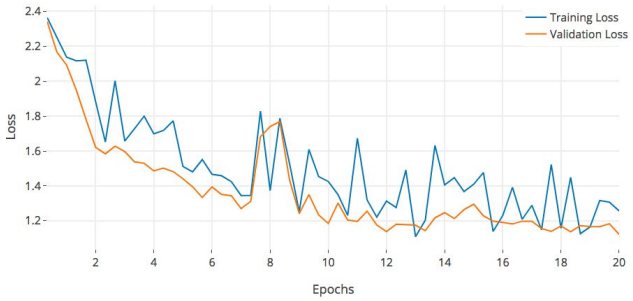

Figure 9: Initial LSTM Training and Validation Loss

Unlike the MLP, the initial untuned results of the LSTM model are satisfactory. After increasing the capacity of the model to 256 units, the performance improved further.

4.2.3. GRU

The final ANN architecture used was the GRU. Very similar in structure to the LSTM, the GRU only required a small change in implementation from LSTM cells to GRU cells in the RNN architecture. The following shows the framework containing the gates of a typical GRU cell:

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

The GRU was not discussed in the original paper because it is a more recent development in the field of neural networks. GRUs were first proposed by Chung et al. in 2014 [6]. This paper showed the effectiveness of GRUs as an alternative to LSTM cells that provided comparable results while maintaining a less complex inner structure, likely resulting in lower training time. Using the same network dimensions as our best LSTM, we are able to achieve similar training loss and validation loss curves, shown in Figure 10.

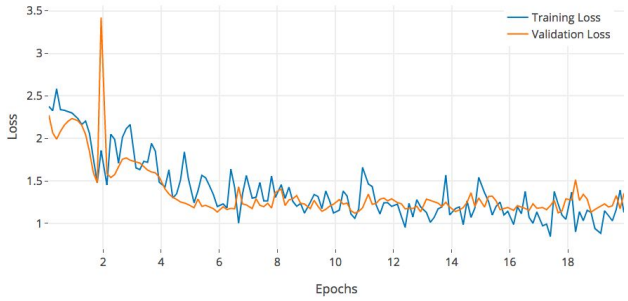

Figure 10: GRU Training and Validation Loss

5. Results

5.1. Project Results

Three ANN models were trained for each essay set: one MLP, one LSTM, and one GRU. Due to differences in network size, batch size, and network architecture, there were large variations in training time for these networks. After tuning the hyperparameters for each model, the following training times were obtained using a Tesla K80 GPU on a Google Cloud Compute instance.

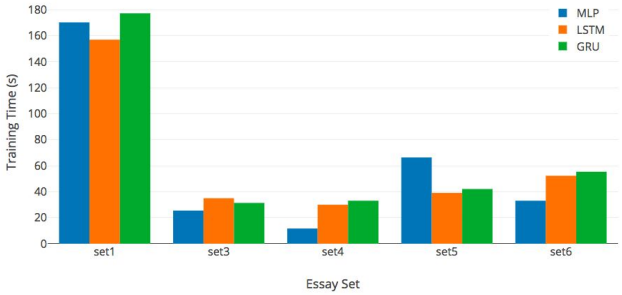

Figure 11: Training Times

From Figure 11, we can see that the training time for essays from Set 1 are considerably longer than the other sets. If we refer back to Figure 6, we can see that these essays are much larger in length, and therefore require a larger input layer to the neural network. All other essay sets have comparable training times. It should be noted that there is no appreciable difference in training times between LSTM networks and equivalently size GRU networks.

While training time is an important factor when considering a neural network solution, more important is the accuracy of the network. As stated previously, the metric for determining the usefulness of these networks is the QWK score, indicating the agreement between predicted scores and actual scores. Using the best models from each class, we can compare the QWK value obtained for each class from the three different network architectures.

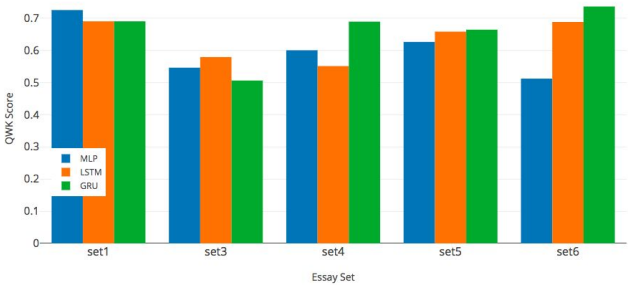

Figure 12: QWK Scores for each Model and Essay Set

As shown in Figure 12, All three of the models performed very well for Set 1, with QWK values of about 0.7. This is likely due to the large size of the grading scale

in Set 1. Therefore the accuracy of our model against random guessing is much better. The MLP using classification performed better than the LSTM and GRU in this essay class. A potential reason is that essays in Set 1 were an average of 420 words, much longer than the essays in the other sets. Perhaps the RNN models were unable to successfully train on such a long sequence. Using a deeper RNN or a bidirectional RNN such as those used by Huyen and Dery would have yielded better results in the case of Set 1.

In Sets 3-6, the RNN architectures typically provided better results than the MLP. This is possibly due to the shorter length of essays in these sets. The RNNs are better able to characterize the entire length of the sequences.

5.2. Comparison of Results

For all network types, we were unable to reproduce the results shown by Huyen and Dery. There are a few reasons for the discrepancy in QWK scores.

We now believe that the choice in word embedding is extremely important for the task of scoring essays. While a pretrained GloVe embedding was an improvement over a simple one-hot encoding, this embedding was originally trained on a Wikipedia training set. Training our own GloVe word embedding on the dataset of essays would have provided more meaningful word vectors for this specific task. This was the final implementation of Huyen and Dery, and it is certainly a cause of discrepancy between our QWK scores and theirs.

An additional source of discrepancy is the difference in model architectures. Our team did not have experience with deep RNNs or bidirectional RNNs and thus these were not implemented.

5.3. Discussion of Insights Gained

There are several possible reasons for the slight divergence in the results we produce and the original results. There is a lack of documentation in the paper regarding some significant parameters. Namely, it isn't clear whether they padded the essays and to what value they were padded. The number of neurons used for hidden layers in the MLP isn't stated. There is also no information regarding whether any manipulation or augmentation of the data was performed in order to make the data distribution more even. Most importantly, the paper mentions "Averaging pre-trained GloVe Vectors". This could mean that each vector representation of a word in the essay was averaged to obtain a single vector representing the whole essay. It could also mean that the word vector and context vector of each word were averaged, which is a common practice for non GloVe word representations, but GloVe doesn't have much documentation for this action. We may have completely ignored the context in which the words appear, thereby losing very important information. Finally, the dataset is acquired from Kaggle, but the competition the dataset is intended for is over, so we cannot run the validation or test set provided since they contain no labels. This means that our training dataset is essentially reduced by 15-20%, which can prove significant for final results.

6. Conclusion

This project reaches the goal of generating accurate grades for essays to a certain extent. The best models produce QWK values comparable if not better than those produced by traditional machine learning models on Kaggle. Due to a lack of time, our models were not well tuned enough to be equivalent to the ones produced in the original paper. However, there are still many valuable things to take away from this project. GRU cells proved to be a viable alternative to LSTM cells for RNN models. They achieved equivalent accuracy and trained in similar time compared to LSTM cells. However, they did not produce faster training as we were expecting. The MLP has a significant disadvantage compared to RNN models as it has no way of remembering sequential data. This has an even greater impact considering the fact that GloVe vectors don't contain context information. This means that when using simple models such as a 2 Layer NN, we must include context into the word vector embeddings. Finally, the distribution of the data plays a very important role in producing unbiased results, and uneven data such as the one used in this project must be manipulated intelligently to produce the best results.

There are several things we would like to try moving forward. First, we would like to do data augmentation to increase the number of underrepresented classes. This can be done by injecting noise or oversampling specific data, or even creating a generative model and producing new essays of a certain class. Second, we would like to experiment with different word vector representations, such as word2vec and train our own word vectors instead of using pre trained ones. Most importantly, we would like to utilize context vectors. Finally, we would like to develop more complex models, such as a deep LSTM or a bidirectional LSTM.

7. Acknowledgement

We thank Prof. Kostic and the class TA's for the advice given regarding the implementation of this project. Additionally, we'd like to The Hewlett foundation for providing the original dataset, Kaggle for hosting the dataset and competition, and Nguyen and Dery for pioneering the use of neural networks for this specific application.

8. References

[1]https://bitbucket.org/ecbm4040/2018_assignment2_mm5305/src/master/GGOP.project.ac4391.bmm2172.mm5305/

[2] Attali, Yigal, and Jill Burstein. ”Automated essay scoring with e-rater V. 2.” The Journal of Technology, Learning and Assessment 4.3 (2006).

[3] Kaggle. ”Develop an automated scoring algorithm for student-written essays.” (2012). https://www.kaggle.com/c/asap-aes

[4] Robertson, Stephen. ”Understanding inverse document frequency: on theoretical arguments for IDF.” Journal of documentation 60.5 (2004): 503-520.

[5] Nguyen, Dery. ”Neural Networks for Automated Essay Grading.” (2016)

[6] Chung, Gulchere, Cho, and Yoshua Bengio. ”Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.” (2014). arXiv:1412.3555

[7] Pennington, Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation.” (2014) https://nlp.stanford.edu/pubs/glove.pdf