Neural Networks for Automated Essay Grading

Huyen Nguyen
Department of Computer Science
Stanford University
huyenn@stanford.edu

Lucio Dery
Department of Computer Science
Stanford University
ldery@stanford.edu

Abstract

The biggest obstacle to choosing constructed-response assessments over traditional multiple-choice assessments is the large cost and effort required for scoring. This project is an attempt to use different neural network architectures to build an accurate automated essay grading system to solve this problem.

1 Introduction

Attempts to build an automated essay grading system dated back to 1966 when Ellis B. Page proved on The Phi Delta Kappan that a computer could do as well as a single human judge [1].

Since then, much effort has been put into building the perfect system. Intelligent Essay Assessor (IEA), developed by Peter Foltz and Thomas Landauer, was first used to score essays for large undergraduate courses in 1994 [2]. The automated reader developed by the Educational Testing Service, e-Rater, used hundreds of manually defined features. It was trained on 64 different prompts and more than 25,000 essays on a 6-point scale from 1 to 6. Evaluated on the quadratic weighted kappa calculated between the automated scores for the essays and the resolved score for human raters on each set of essays, e-rater could only achieve the kappa score below 0.5. [3]

In 2012, the Hewlett Foundation sponsored a competition on Kaggle called the Automated Student Assessment Prize (ASAP). The competition also used quadratic weighted kappa to measure the similarity between the human scores and the automated scores. 154 teams participants attempted to predict. The winning team got the kappa score of 0.81407. [4] Later, a team at Carnegie Mellon University built a model using dense and sparse features, trained on the same dataset to achieve the kappa score of 0.833. [5]

All models so far have been built using predefined features without deep learning features. We want to build an automated essay grader using neural networks. This project aims at two things:

1. We want to apply what we have learned in the class to solve a practical problem.
2. We want to see how well neural networks perform compared to machine learning with predefined features.

2 Problem statement

Our aim is to build a model that can take in an essay and automatically outputs the grade of that essay. Within the scope of this project, we only work with essays written students in grade 7 to grade 10. The grading scale of the essay can vary. Our models are capable of acknowledging the difference in scale and outputting the corresponding grade.

There are two kinds of models we are building: regression models and classification models.

For regression models, our output will be a continuous value between 0 and 12. For classification models, our output will be a discrete value between 0 and 12.

2.1 Dataset

We use the dataset provided for Hewlett Foundation?s Automated Student Assessment Prize competition on Kaggle. Some characteristics of the dataset:

1. There are 8 different sets of essays, each generated from a single prompt. To fasten computation, we used only 6 sets.
2. Selected essays range from an average length of 150 to 550 words per response.
3. Each essay was hand-graded by two or three instructors.
4. There are 10686 samples in total. We used 10% for testing, the other 90% was used for both training and validation.
5. Each set has a different grading scale.

Essay Set	Essay Type	Domain	Score Range	Average Length	train	dev.	test	total
1	Persuasive/Narrative/Expository	-	2-12	350 words	1,284	321	178	1,783
2	Persuasive/Narrative/Expository	Writing Applications	1-6	350 words	1,296	324	180	1,800
		Language Conventions	1-4					
3	Source Dependent Responses	-	0-3	150 words	1,244	310	172	1,726
4	Source Dependent Responses	-	0-4	150 words	1,276	319	177	1,772
5	Source Dependent Responses	-	0-4	150 words	1,300	325	180	1,805
6	Source Dependent Responses	-	0-4	150 words	1,296	324	180	1,800

2.2 Expectations and Evaluation Metric

We evaluated our performance with the quadratic weighted kappa metric defined below:

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

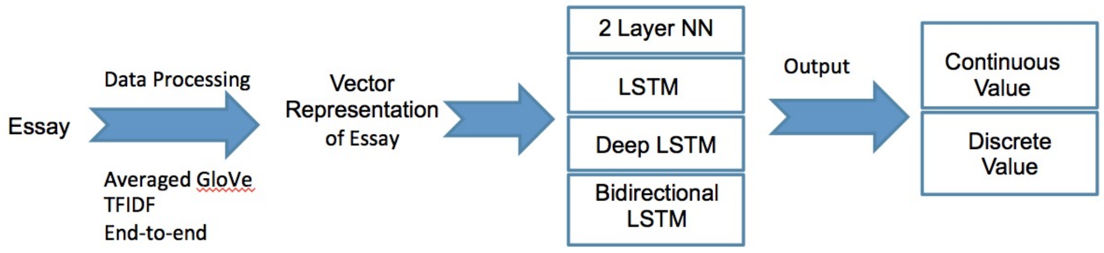
The Weight matrix W is calculated based on the difference between raters scores. Let the scores range from 1, 2 .. , N:

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

O_{ij} corresponds to the number of essays that received a rating i by Rater A and rating j by Rater B
E is the N-by-N histogram matrix of expected ratings calculated by taking the outer product of each rater's histogram of ratings, normalized such that E and O have the same sum.
The Quadratic Weighted Kappa (QWK) metric typically varies from 0 - only random agreement between raters - to 1 (complete agreement between raters). In the event that there is less agreement between the raters than expected by chance, this metric may go below 0.
Because we use the Kaggle dataset, our baseline is the kappa score of the winning team. We hoped to do at least as well as they did, with kappa score of 0.81407.

3 Technical Approach and Models

Our models include three steps: data processing, training the models with our loss functions and tuning of hyperparameters.



3.1 Vector Representation of Essays

We represent each essay as a vector. We use three different methods to vectorize the essays:

1. Using averaged pre-trained GLoVe word vectors [6]
   We take the GLoVe word vectors of all the word in an essay, average them to get the essay vector. We experimented with GLoVe word vectors of 50, 100, 200 and 300 dimensions.
2. Term Frequency Inverse Document Frequency (tf-idf)
   We counted the occurrence of each word in an essay and used term frequency inverse document frequency (tf-idf) [7] to take into account the importance of each word. Each document is represented by a tf-idf vector of dimensions equal to the size of the vocabulary. We then feed this representation as an input to an auxiliary layer in our networks.
3. Word vector training
   We trained the word vectors together with the weights. We initialized the word vectors to be the GLoVe word vectors of 300 dimensions.

3.2 Models

We had two framings of the problem. The first was to consider the problem as a regression problem. Here we our model outputs a score in the expected range of scores and our objective is to minimize the square error between the predicted scores and the actual scores. The objective in this case objective was Mean Squared Error.

Given a set of predictions $\hat{y}$ and the true grades y, we sought to minimize:

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

Let $h^{(out)}$ is the output of the layer before the readout layer, we have:

$$\hat{y} = \text{U}h^{(out)}$$

For the classification problem, we constructed a readout layer that predicted the probability of an essay having one of these scores. In this case, we used Categorical Cross Entropy [8] to minimize the distance between the predicted probability distribution and the actual probability distribution of the dataset.. Given a set of predictions $\hat{y}$ = softmax(θ) and the true grades y, represented as one hot vectors, we sought to minimize:

$$\text{CE}(\text{y}, \hat{y}) = -\sum_{i} y_i log(\hat{y}_i)$$

Let $h^{(out)}$ is the output of the layer before the readout layer, we have:

$$\hat{y} = \text{softmax}(W^{(out)}h^{(out)} + b^{(out)})$$

Below are the Neural Architechtures we implemented to solve the problem

1. Two Layer Feed Forward Neural Network [9]

$$h^{(1)} = \tanh(W^{(1)}x + b^{(1)})$$
$$h^{(out)} = \tanh(W^{(2)}h^{(1)} + b^{(2)})$$

   For our two layer Neural Network we experimented with different forms of input. We constructed an Essay vector for each essay which was obtained by averaging all the word vectors of the words in the essay. Essay vectors were constructed in two ways:

- Averaging pre-trained Glove Vectors
- Averaging word vectors after training them

Using this approach of creating Essay vectors constituted our bag of words model. These two models above did not reflect sequence trends that were present in our data.

2. Long Short Term Memory (LSTM) [10]
   An LSTM cell at time step t is defined as follows:

   Input gate : $i_t = \sigma(W^{(i)}x + U^{(i)}h^{(t-1)})$
   Forget gate: $f_t = \sigma(W^{(f)}x + U^{(f)}h^{(t-1)})$
   Output gate: $o_t = \sigma(W^{(o)}x + U^{(o)}h^{(t-1)})$
   New Memory Cell: $c'_t = tanh(W^{(c)}x + U^{(c)}h^{(t-1)})$
   Final Memory Cell: $c_t = f_t \circ c_{t-1} + i_t \circ c'_t$
   Final Hidden Cell: $h_t = o_t \circ c_t$

   We implemented a variable length simple LSTM with a scoring layer at the end. The LSTM receives a sequence of word vectors corresponding to the words of the essay and outputs a vector that encapsulated in the information contained in the essay. The scoring layer at the end converts this vector into a score in the required range. The bag of words model used in the feed forward networks described above has limited power. The rationale for Introducing LSTM based architecture therefore, was to capture sequence information in the dataset.
   We experimented with two LSTM models:

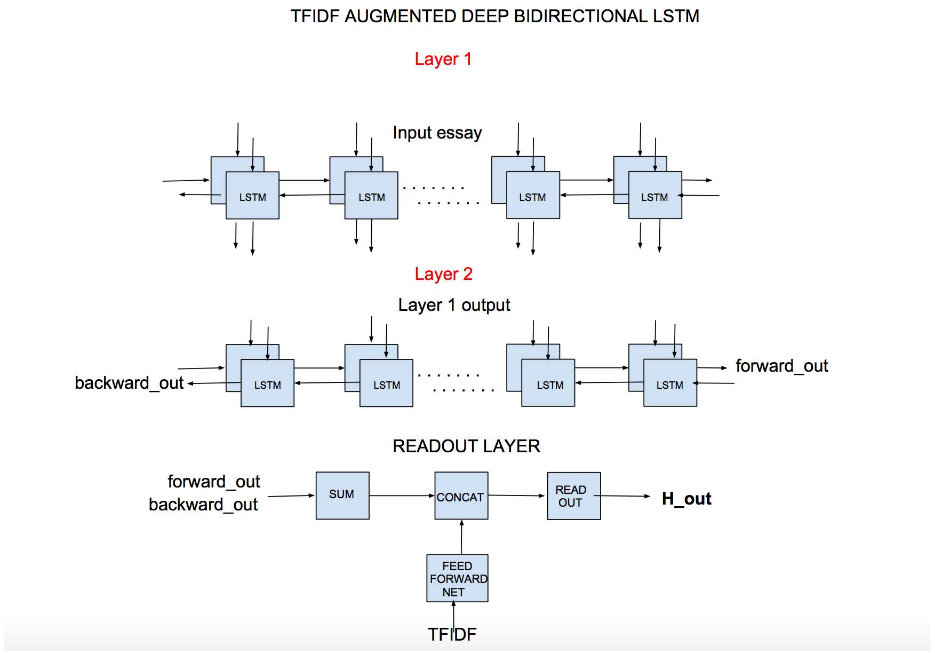
   - Single Layer LSTM with a timestep for each essay word
   - Deep LSTM consisting of three LSTM layers from (a), chained together, each with a timestep for a word in the essay

3. Bidirectional Long Short Term Memory
   We implemented a bidirectional Long Short Term Memory Architecture. This architecture traverses the essay in both forward and backward directions. It then sums up the output of the two. This output is then fed to a scoring layer to output a prediction. We posited that a bidirectional LSTM would give us improved performances over a simple LSTM because of the length of our essays.

   Our essays' average length range from 150-550 words in length, meaning we often have as many as 500 timesteps. To prevent information from the beginning of the essay from being significantly attenuated by the time it gets to the output, we decided to introduce another layer that reads the essay backwards, thus retaining more of the information encapsulated at the beginning of the essay. We also experimented with two models:

   - Single Layer Bidirectional LSTM
   - Deep Bidirectional LSTM architecture

   Our Deep Bidirectional LSTM architecture



4. Term Frequency Inverse Document Frequency (tf-idf) Augmentation As an augmentation, we decided to complement our current models with a layer, before the readout layer that receives the TIFIDF vector of each document as input. The TFIDF vectors were transformed to a lower dimension from the original |V| and subjected to a tanh nonlinearity. This vector was then concatenated with the output of the original model before a prediction is made in the scoring layer.

$$\text{tf-idf} = \tanh(W^{(T)}T + b^{(T)})$$
$$h^{(out)} = \mathrm{U}[h^{(f)}; tf - idf]$$

3.3 Hyperparameters Tuning

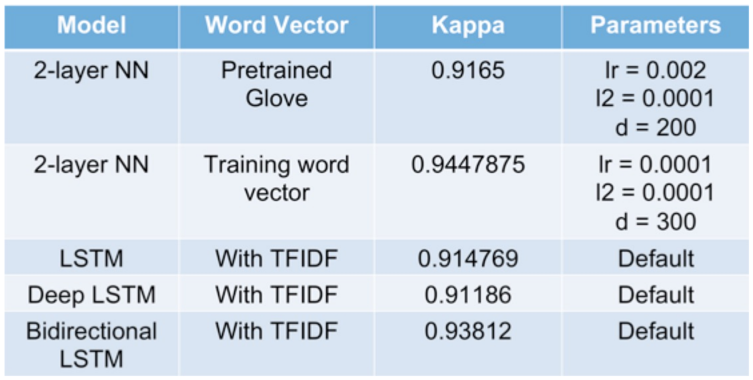
After building the models, we tuned the following hyperparameters to get the optimal results.

- Learning rate (lr)
- Regularization (l2)
- Dimension of essay vectors (50, 100, 200, 300)
- The percentage (fraction) of samples we used for training and validation

Unfortunately, since we started this project late due to a last minute change in project we did not have time to do a very extensive hyperparemeter search on the LSTM models, unlike the 2 layer neural network, because take more time to run.

4 Findings and Analysis

Our main goal going into this project was to outperform the best result on the dataset when it was presented in the 2012 Hewlett Foundation Automated Essay Scoring challenge. This was a quadratic weighted kappa score of 0.81407. We could not find any paper detailing the specifics of the algorithm used by the winning team. However, a paper released by researchers at CMU, which produced similar results, suggests that hand engineered features were used for solving the problem as opposed to a neural network architecture. This would be in line with the fact that the use of these architectures really became popular after 2012. It is therefore not surprising that even our baseline architecture, when fine tuned, was able to outperform the 2012 results as Neural Networks have



Model	Word Vector	Kappa	Parameters

2-layer NN	Pretrained Glove	0.9165	lr = 0.002
			l2 = 0.0001
			d = 200

2-layer NN	Training word vector	0.9447875	lr = 0.0001
			l2 = 0.0001
			d = 300

LSTM	With TFIDF	0.914769	Default
Deep LSTM	With TFIDF	0.91186	Default
Bidirectional LSTM	With TFIDF	0.93812	Default

expressive power than hand engineered features.

The best score we achieved was 0.9447875, using a 2 layer neural network that trains word vectors together with the weights with learning rate = 0.0001, regularization l2 = 0.0001. Word vectors were initialized to the GLoVe word vectors of dimension 300. Below is a summary of our results.

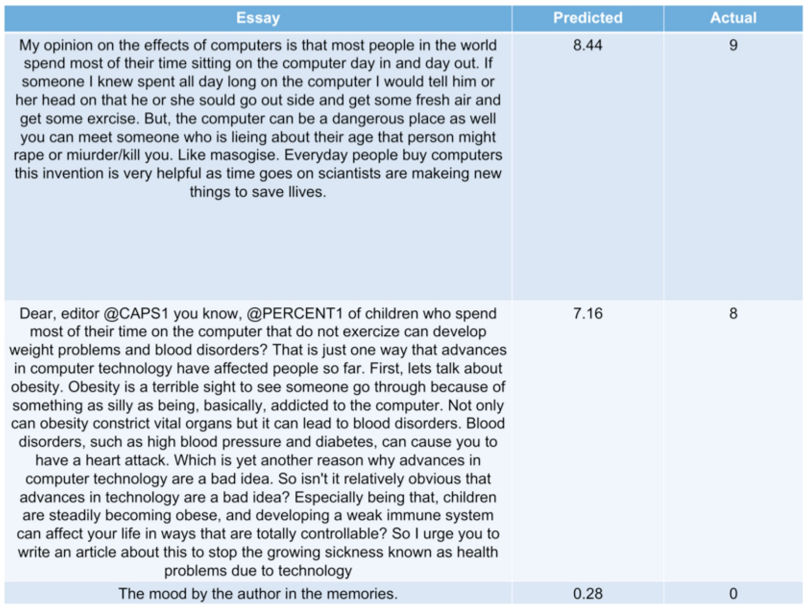
Baseline/Default parameters for LTSM-based models

- Word Vector Embedding dimension = 100
- Hidden Layer Dimensions = 50
- Learning Rate = 1e-4
- Batch Size = 128
- TFIDF Dimension = 500 (tf-idf vectors were reduced from |V| to this dimension)
- Regularization = 1e-3
- Output Dimension = 32 (output dimension of the Layer before the scoring/classification layer)

4.1 LSTM based models

Our LSTM Based approaches were trained on an Adam Optimizer using a batch size of 128. The embedding and hidden dimensions were standardized to 1x100 and 1x50 dimensions respectively for all for all experiments. Training was done for a maximum of 20 epochs with early stopping.

We started out with the Single LSTM framework trained on the mean square error objective and augmented with tf-idf features. The model was able to identify the different score ranges of the various essay types and adapt accordingly. The table below shows sample essays text with their predicted scores from the LSTM versus the actual scores.



Essay	Predicted	Actual
My opinion on the effects of computers is that most people in the world spend most of their time sitting on the computer day in and day out. If someone I knew spent all day long on the computer I would tell him or her head on that he or she sould go out side and get some fresh air and get some exrcise. But, the computer can be a dangerous place as well you can meet someone who is lieing about their age that person might rape or miurder/kill you. Like masogise. Everyday people buy computers this invention is very helpful as time goes on sciantists are makeing new things to save llives.	8.44	9
Dear, editor @CAPS1 you know, @PERCENT1 of children who spend most of their time on the computer that do not exercize can develop weight problems and blood disorders? That is just one way that advances in computer technology have affected people so far. First, lets talk about obesity. Obesity is a terrible sight to see someone go through because of something as silly as being, basically, addicted to the computer. Not only can obesity constrict vital organs but it can lead to blood disorders. Blood disorders, such as high blood pressure and diabetes, can cause you to have a heart attack. Which is yet another reason why advances in computer technology are a bad idea. So isn't it relatively obvious that advances in technology are a bad idea? Especially being that, children are steadily becoming obese, and developing a weak immune system can affect your life in ways that are totally controllable? So I urge you to write an article about this to stop the growing sickness known as health problems due to technology	7.16	8
The mood by the author in the memories.	0.28	0

Figure 1: Examples of good and bad essays

We were able to achieve a Quadratic Weighted Kappa (QWK) score of 0.914769 using the mean square error loss function. Updating the model from a regression to a classification model with cross entropy loss function lead to an improved score of 0.9337.

Encouraged by the success of our tuned single pass LSTM, we progressed to attempt a tf-idf augmented deep LSTM model. We stacked 3 of the LSTM modules from before together, using the output of one layer as the input to the next. Visualizing the hidden layer (produced by doing Principal component analysis on the 50 dimensional hidden layer outputs) just below the readout layer of the deep LSTM showed a very strong ability to discriminate between good and bad essays.

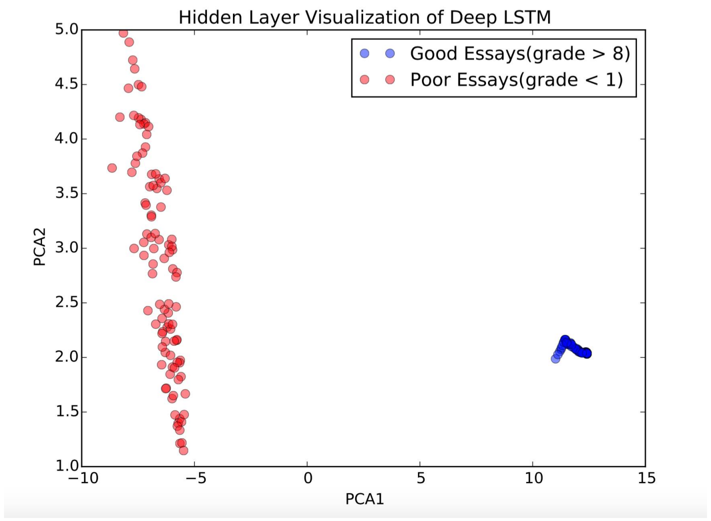


Figure 2: 100 essays that had a score of greater than 8 and 100 with a score less than 1

As can be seen, the good essays were tightly clustered together whilst the poor essays, though aggregated in a specific region are more spread out. This suggests that good essays have defining structure within themselves. Possible interpretations of this structure could be strength of logical progression of the essay, which is sequenced information that could easily be captured by the LSTM, and optimal essay length. If we then choose to interpret the x axis as the logical and cohesive progression of an essay and the y axis as length of the essay, then we can see that good essays have roughly the same length, not too short but also not too long, and very also have a high sense of logical progression (large x value). Poor essays on the other hand have very low strength of logical progression and vary wildly in length: from too short to too long.

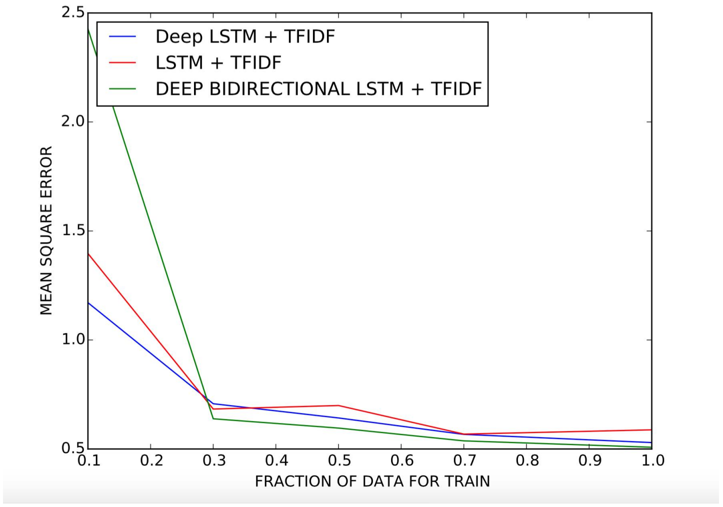
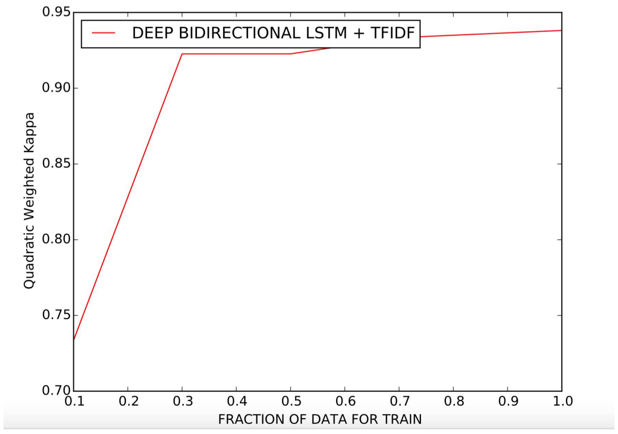


Figure 3: Performance of the 3 LSTM models in different data regimes

It can be seen that in low data regimes, the deep LSTM is the best performing model. In this regime, the deep bidirectional LSTM, due to the large number of parameters that have to be trained does not perform well. However, when we trained on the full train set, holding out 30% for cross validation, the Deep Bidirectional LSTM performed better, achieving the lowest mean square loss of 0.508. The graph below shows the QWK score of the deep Bidirectional LSTM in different data regimes. Using 0.1 of the train set results in a score less than the 2012 winning score, however, when the model trains on the full train set with 30% holdout cross validation, it achieves a quadratic weighted kappa score of 0.93812, the best amongst the LSTM based approaches. The graph below suggests that the model could conceivably be made to perform better with more data as a plateau has not yet been reached.



4.2 Two Layer Neural Network

As already established, our best results were achieved by a tuned simple two layer neural network. As expected, the neural network that trains word vector performs better than the one that uses pre-trained Glove vectors. Since the essays in the dataset were answers to a specific set of prompts, training the word vectors helped to capture the essence of the words in the domain of the essay prompts thus leading to better performance. Our performance also increased as the dimensionality of word vectors increased. The visualizations below summarize this information.

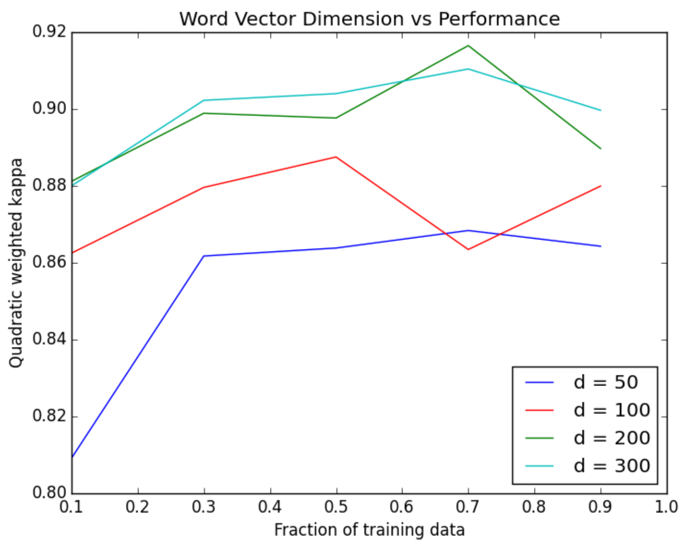


Figure 4: Two layered neural network with pretrained GLoVe word vectors

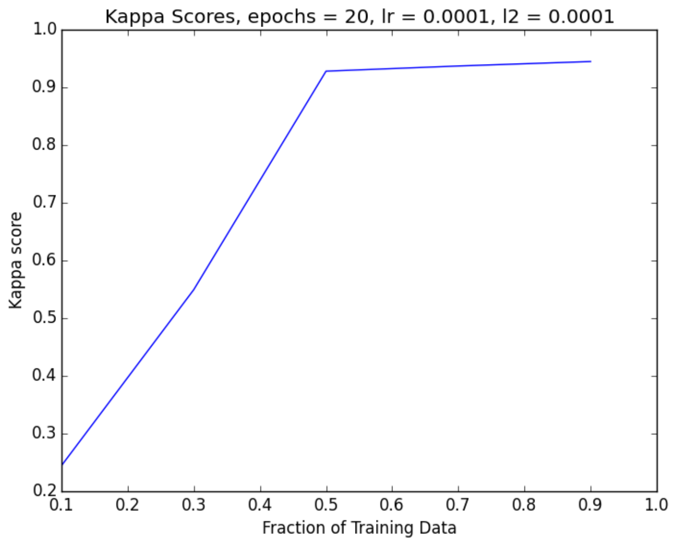


Figure 5: Two layered neural network with word vectors trained using backprop

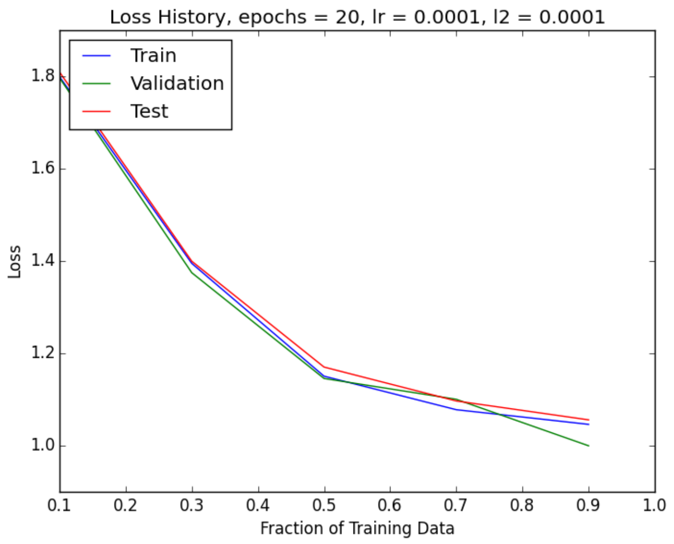


Figure 6: The corresponding cross entropy loss

5 Conclusion

Our models have greatly improved on upon the standard achieved during the Automatic Essay Grading Kaggle Competition. Our best neural network models achieved a score of 0.9447875 on the Quadratic Weighted Kappa metric as against the original best of 0.81407. This verifies the vast potential of neural network architectures in solving natural language processing problems. It was surprising that our simple neural network model using 300 dimensional Glove as initialization to the embedding layer was our most successful model. We believe that a more patient hyperparameter search with our LSTM based models could outperform this result. However, due to time constraints due to the last minute project change we were unable to find these optimal parameter set for the LSTM models which have a larger hyper parameter space.

There are many ideas moving forward. We would like to do a more extensive hyperparameter search on the deep LSTM models upon getting access to more computing resources and time. Trying out the models in Ensemble mode is also an extension we wish to try out in the near future.

We conceive continuing this project and possibly releasing an online Automatic Essay Grader to the general public at some point.

References

[1] Page, Ellis B. ”The imminence of... grading essays by computer.” The Phi Delta Kappan 47.5 (1966): 238-243..

[2] Foltz, Peter W., et al. ”Implementation and applications of the Intelligent Essay Assessor.” Handbook of automated essay evaluation (2013): 68-88.

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

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

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

[6] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. ”Glove: Global Vectors for Word Representation.” EMNLP. Vol. 14. 2014.

[7] Herman, Joan, and Robert Linn. ”On the Road to Assessing Deeper Learning: The Status of Smarter Balanced and PARCC Assessment Consortia. CRESST Report 823.” National Center for Research on Evaluation, Standards, and Student Testing (CRESST) (2013).

[8] De Boer, Pieter-Tjerk, et al. ”A tutorial on the cross-entropy method.” Annals of operations research 134.1 (2005): 19-67.

[9] Davidian, David. ”Feed-forward neural network.” U.S. Patent No. 5,438,646. 1 Aug. 1995.

[10] Hochreiter, Sepp, and Jrgen Schmidhuber. ”Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.