Automated Essay Grading System (AEGS) Using Word Embedding

Hazem Alabiad, N20139094, Author
CMP711, Project Report
Hacettepe University, Computer Engineering Department

Abstract- **The task of human-based grading essays has been big trouble due to the long time it needs, subjectiveness in its nature, difficulty it involves, and focus it requires to minimize the possibility of errors as much as possible. As well as, the requirement of numerous teachers as in most of the situations, a single essay is being graded at least twice by distinct people. These reasons push us toward investigating and exploring more in this scope. We make use of the essays dataset obtained from Kaggle.com. These essays are divided into 8 sets based on context and grade of students arranging from 7 to 10. We first, clean the data and preprocess it using NLP techniques then, we convert the textual data into numeric by using the word-embedding approach. After that, we feed those input vectors into regression-based models to predict the grades. In the training and testing phases, we use the KFold methodology. Lastly, we evaluate the performance of the used models using many measurement matrices depending on the model. In the case of LSTM, we use Quadratic Weighted Kappa QWK to figure out the agreement between predicted and original grade.**

I. Introduction

Nowadays most educational institutes are moving to internet-based methods to perform exams; Especially, the English Language Examination which is taken by an enormous number of people around the world for many purposes. Automating the grading will be so beneficial as the exam takers do not have to wait for at least two weeks to get the results, and the educational institutes will save money and effort simply by enabling the machine-grader program which has been trained and designed to achieve the highest performance. We make use of Supervised Learning methods on the human-scored "labeled" essays. As a learning model, RNN seems to work very well in similar tasks.

II. Problem Statement

The main goal is to build a model that outputs the grade of a given essay.

A. Dataset

The dataset is provided by *Hewlett Foundation* and on Kaggle.com [5] in many formats, we preferred to use the *CSV* format. In this phase, we do Exploratory Data Analysis to comprehend our data and shape it in the best way to achieve the best results by the model.

In *Table 1*, we see general information about each essay sets. Each set has a different grading scale, and average essay length to guarantee a more robust model.

essay_set	grade_level	train_set_size	min_score	max _score
1	8	1783	2	12
2	10	1800	1	6
3	10	1726	0	3
4	10	1772	0	3
5	8	1805	0	4
6	10	1800	0	4
7	7	1569	0	30
8	10	723	0	60

Table 1: Essay Sets Description

In *Figure 1*, we see the frequency plot of each set with the average length of essays included:

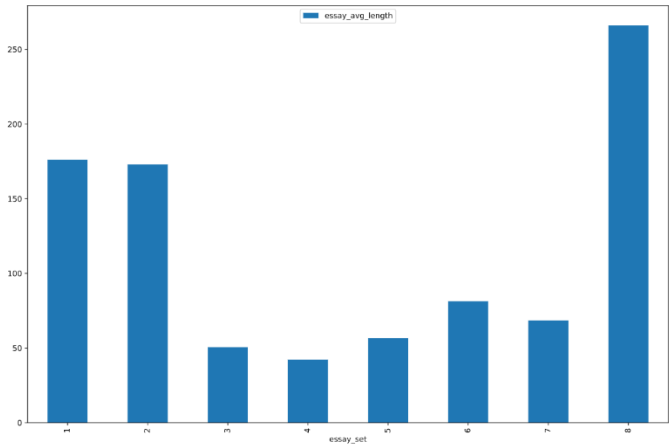


Figure 1: Essay Average Length of Each Set

In *Figure 2*, we see the frequency plot of the top 20 words in the dataset after cleaning the text and removing stop words.

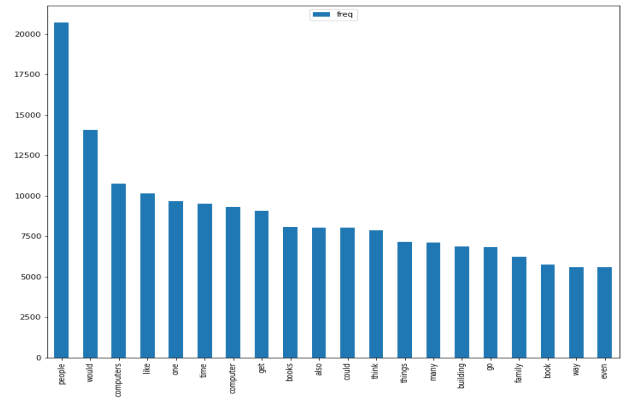


Figure 2: Top 20 Frequent Words in the Dataset

In *Figure 3*, we see the percentage of the top 10 scores:

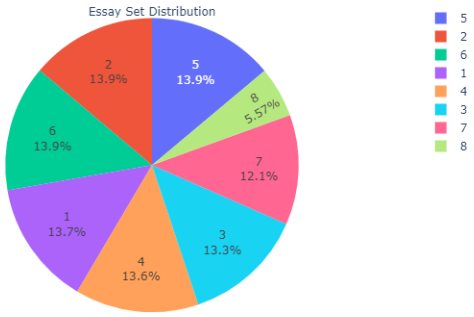
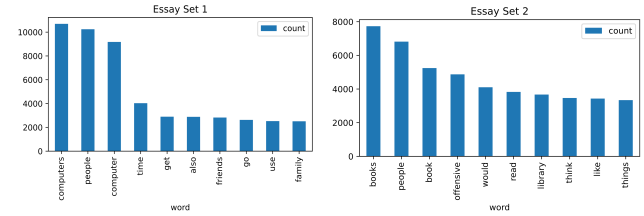


Figure 3: Top 10 Scores Distribution

In *Figure 4*, we see the histograms of the top 10 frequent words in each essay set:



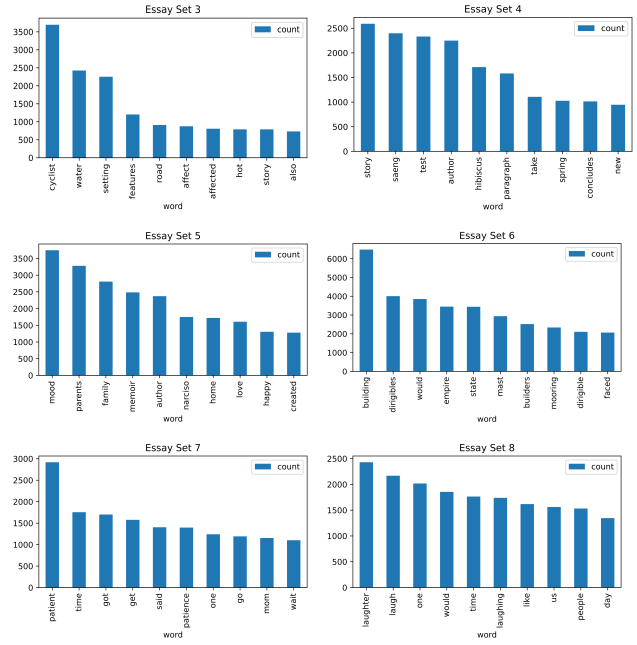


Figure 4: Top 10 Frequent Words in Each Essay Set

III. Technical Approach & Models

A. Methodology

Using the KFold approach with K splits, to avoid the overfitting problem [6], we split the dataset into train and test data K times, in each time we train and test then, we take the average of the accuracy in the K times.

We first get some insights and statistics about our data afterward, we act accordingly by dropping NA's, add columns "we use Pandas dataframes to process the data", remove others. Later, we preprocess our data by cleaning the text e.g. removing non-letter characters and eliminating stop words and tokenizing; Here we tend not to apply *Stemming* nor *Lemmatization* as in many research it is found that it does not improve the performance.

We employ the Word-Embedding technique to obtain 300 by using a pre-trained model *GoogleNews-vectors-negative300* [7] on *Google News Corpus* with around (3 billion words) which is mostly is going to give us better results than a model trained on our dataset.

In *figure 5*, we simply explain the whole project steps and phases:

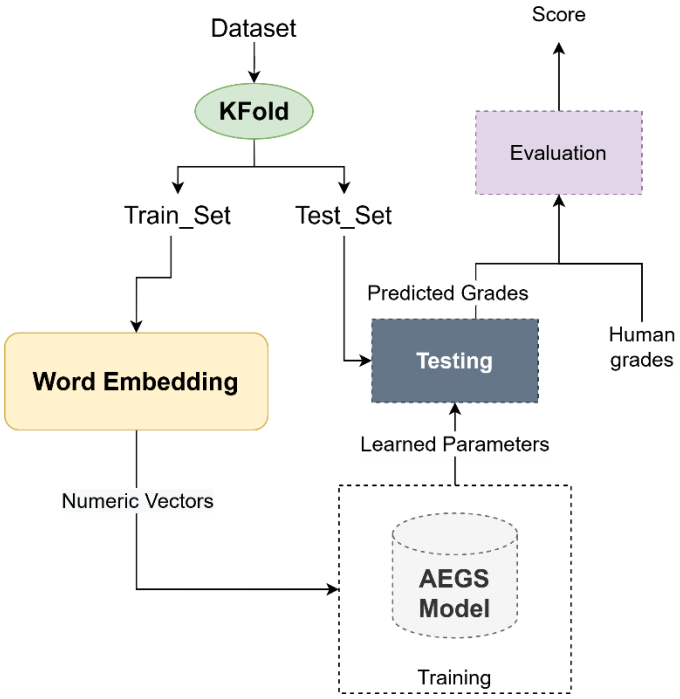


Figure 5: Project Overview Diagram

B. Tools & Libraries Used

We use Python 3.8.x, NLTK, SciKit-Learn, Keras, Tensorflow, VSCode, Jupyter, Matplotlib, Plotly, Pandas, Pickle, Numpy, re, and Gensim.

C. Learning Model

After reviewing related works, we conclude that LSTM is proved to work well with similar problems with an accuracy of ~ 0.72 using a W2V model that is trained on the dataset itself.

However, with the use of a pre-trained model, and some changes to model's parameters we improved the performance and acquired ~ 0.92 accuracy.

C.a. Model Components & general Architecture

LSTM: it is an RNN that has feedback connections, unlike traditional RNN.

We first need to point out that this is a regression problem where we input a numeric vector to our trained model then, it outputs the prediction as a numeric continuous value, we later round it to the nearest distinct value. We select the *mean squared value* (1) as the objective function to minimize the error between the predicted and actual score.

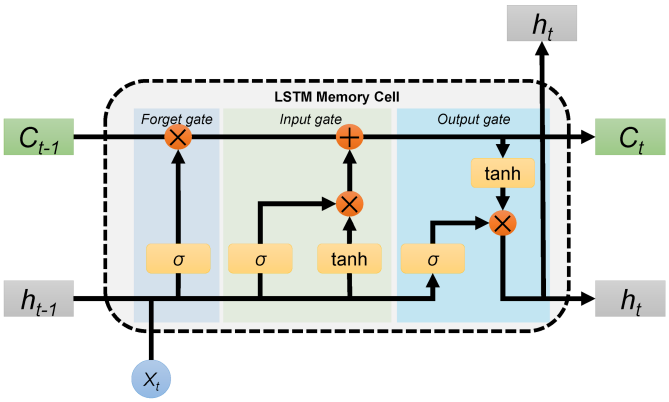


Figure 5: Architecture of LSTM Cell [9]

The cell state is regulated by three gates:

1- Forget gate (2): controls which elements of the previous cell state vector to be forgotten. Its output is a vector of the *sigmoid* function of range [0,1]. W_f and b_f define the set of trainable parameters for the forget gate.

2- Input gate (3): Decides which value to be updated. W_f and b_f define the set of trainable parameters. The output range [0,1].
Then, a *potential vector* of the cell state (4) is computed by combining *hidden previous state* h_{t-1} and *current input* x_t. It outputs values in the range [0,1]. Where W_C and b_C are the trainable parameters.
After that, we update the *old cell state* C_{t-1} into the *new cell state* C_t by using formula (5).

3- Output gate (6): Decides which to be output to the *sigmoid* function.
The *new hidden state* h_t is defined by both the output state and the *new cell state* C_t by the formula (7)

In figure 6, we see the overview architecture of the LSTM model:

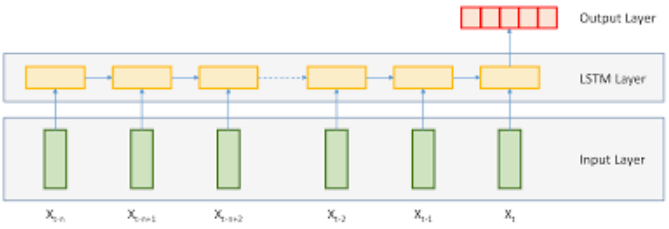


Figure 6: Architecture of LSTM model

D.b. Tuning for Hypermeters

It is the procedure of finding out the best parameters that minimize the loss function of the given data leading to the best and most accurate model. See (1).

In our case, we build an LSTM model of the following architecture:

Layer	**Output Shape**	**# of Parameters**
LSTM Layer	(None, 1, 300)	721200
LSTM Layer	(None, 64)	93440

Dropout Layer "used to reduce overfitting"	(None, 64)	0
Dense Layer (hidden layer) with *ReLU* (8) as the activation function	(None, 1)	65

We chose RMSProp (9) as our optimizer, and MAE as our metric which is used in monitoring the training of the model at the end of each epoch where information about the training including [ms/step, loss, [metrics]] is printed to console to track the training process.
We select our *batch_size* = 64 and epochs = 10, with all parameters above the model has a total number of trainable parameters 814,705. The output predictions are continuous data which we round into the nearest integer values to be comparable with the human scores.

IV. Evaluation

Utilizing KFold with K=5, the model took around ~ 8 hours. We could achieve an accuracy of ~ (0.92) with QWK which is a robust error metric used to measure agreement between the *predicted scores* and the *human scores* in every fold. Lately, we calculated the mean of the QWK score of each fold to have the average QWK score.
In the following table and *figure 7* we can see how much our model was accurate in predicting the grade. We used Mean Absulte Percentage Error (MAPE) (10)

	essay_set	error	accuracy
0	1	1.588235	98.411765
1	2	0.892351	99.107649
2	3	1.082857	98.917143
3	4	1.076923	98.923077
4	5	1.040698	98.959302
5	6	1.021164	98.978836
6	7	4.107463	95.892537
7	8	8.299270	91.700730

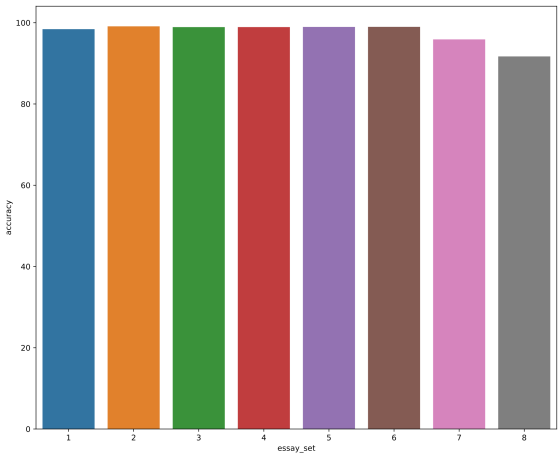


Figure 7: Accuracy in each essay set

V. Abbreviations and Acronyms

Abbreviation	Explanation
NLP	Natural Language Processing
CSV	Comma-separated Values
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
AEGS	Automated Essay Grading System
NLTK	Natural Language ToolKit
W2V	Word to Vector
QWK	Quadrant Weighted Kappa
DNN	Deep Neural Network
ReLU	Rectified Linear Activation
RMSProp	Root Mean Square Propagation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

VI. Equations

Equation reference	Explanation
1	$j = \frac{1}{N}\sum_{i}^{N}\left(\hat{y}_i - y_i\right)$
2	$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right)$
3	$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right)$
4	$\widetilde{C} = tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right)$
5	$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}$
6	$o_t = \sigma(Wo[ht - 1, xt] + bo)$
7	$h_t = o_t \cdot tanh\left(C_t\right)$

8	$y = max(0, x)$
9	$E\left[g^2\right]_t = 0.9E\left[g_2\right]_{t-1} + 0.1g_t^2$
10	$MAPE = \sum_{i}^{N} \frac{\lvert Error\ /\ Actual \rvert}{N}$

VII. Conclusion

Grading Essays using machines is a hot topic that is still investigated and pulling researchers' attention to invent better approaches to save time, effort, and money. We used the *Word Embedding* technique to capture the contextual meaning of the textual input data and convert them into numeric representations (vectors) that could be later fed to many regression-based models to train and predict the score of any given essay. We used QWK to measure the performance of our trained model against the actual scores.

VIII. Future Work

The performance could be increased by adding adding more features e.g. Readibility [11], Lexical Complixity, Grammar Errors, Content Analysis based on vocabulary. Also, another models could be investigated e.g. Random Forest Tree, Other Neural Network based models.

References

[1] Suresh, A. et Jha, M., *Automated Essay Grading using Natural Language Processing and Support Vector Machine*, International Journal of Computing and Technology, vol. 5, Issue 2, 2018, pp. 18-21

[2] Mahana, M. et Johns, M. et Apte, A., *Automated Essay Grading Using Machine Learning*, Stanford University, 2012, pp. 1-5

[3] Nguyen, H. et Dery, L., *Neural Networks for Automated Essay Grading*, Stanford University, pp. 1-10

[4] Taghipour, K. et Ng, H., *A Neural Approach to Automated Essay Scoring*, National University of Singapore, Association for Computational Linguistics, 2016, pp. 1882-1886

[5] https://www.kaggle.com/c/asap-aes/data

[6] https://en.wikipedia.org/wiki/Overfitting

[7] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[8] https://www.mdpi.com/2073-4441/12/1/175/htm

[9] CMP711 course note
https://web.cs.hacettepe.edu.tr/~ilyas/Courses/CMP711/

[10] https://readable.com/readability/