

AUTOMATIC ESSAY GRADING SYSTEM

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

ALEKHYA NARAVAJHULA

1602-18-733-067

DULUGUNTI SAI HARITHA

1602-18-733-095

Under the guidance of

Dr. V. Sireesha

Assistant Professor



Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering



DECLARATION BY THE CANDIDATE

We, **ALEKHYA NARAVAJHULA** and **SAI HARITHA DULUGUNTI**, bearing hall ticket numbers, **1602-18-733-067** and **1602-18-733-095** hereby declare that the project report entitled “**Automatic Essay Grading System**” under the guidance of **DR. V. SIREESHA**, ASSISTANT PROFESSOR, Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering in Computer Science & Engineering**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Alekhyia Naravajhula ,

1602-18-733-067

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering



DECLARATION BY THE CANDIDATE

I, **ALEKHYA NARAVAJHULA**, bearing hall ticket number, **1602-18-733-067**, hereby declare that the project report entitled “**Automatic Essay Grading System**”, under the guidance of **DR. V. SIREESHA**, ASSISTANT PROFESSOR, Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering in Computer Science & Engineering**.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Alekhya Naravajhula,
1602-18-733-067

Sai Haritha Dulugunti,
1602-18-733-95

Vasavi College of Engineering (Autonomous)
(Affiliated to Osmania University)
Hyderabad-500 031
Department of Computer Science & Engineering



BONAFIDE CERTIFICATE

This is to certify that the project entitled “**Automatic Essay Grading System**” being submitted by **ALEKHYA NARAVAJHULA** and **SAI HARITHA DULUGUNTI**, bearing **1602-18-733-067** and **1602-18-733-095** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

Dr. V. Sireesha,
Assistant Professor ,
Internal Guide,

Dr. T. Adilakshmi,
Professor &HOD,
Dept. of CSE,

ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Dr. V.Sireesha ma'am**, Department of Computer Science and Engineering, whose guidance was unforgettable and filled with innovative ideas as well as her constructive suggestions has made the presentation of my major project a grand success.

We are thankful to **Dr. T. Adilakshmi**, Head of Department (CSE), **Vasavi College of Engineering** for the help during our course work.

Finally we express our gratitude to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete our project work successfully.

ABSTRACT

Essays are important for testing students' academic scores, creativity, and being able to remember what they studied, but grading them manually is really expensive and time-consuming for a large number of essays. This project aims to implement and train neural networks to assess and grade essays automatically. The human grades given to the essays should be matched with grades generated from our automatic essay grading system consistently with minimum error. Automated Essay Grading can be used for evaluating essays written according to specific prompts or specific topics. It is the process of automating scoring system of essays without any human intervention and using computer programs. This system is most beneficial for educators since it helps reducing manual work and saves a lot of time. It not only saves a lot of time but also speeds up the process of learning feedback. We used deep neural networks in our system instead of traditional machine learning models.

TABLE OF CONTENTS

List of Figures.....	
List of Tables.....	
1. Introduction.....	1
1.1 Overview.....	3
1.2 Motivation.....	3
1.3 Problem Definition.....	4
1.4 Scope.....	5
2. Literature Survey.....	6
2.1 Existing Systems.....	6
2.2 Related Work.....	10
2.3 Tools and Technology.....	13
3. System Analysis.....	14
3.1 Dataset.....	14
3.2 Features.....	18
3.2 Word Embeddings.....	20
4. System Design.....	21
4.1 Proposed Methodology.....	22
4.2 System Architecture.....	35
5. Pseudo Algorithm.....	37
6. Results.....	55
7. Testing.....	56
8. Conclusion and Future Work.....	58
9. References.....	59

LIST OF FIGURES

Figure 1. Automatic Essay Grading.....	2
Figure 2. AES Explained.....	4
Figure 3. C-Rater.....	7
Figure 4. E-Rater.....	9
Figure 5. History of Essay Grading.....	11
Figure 6. Dataset Statistics.....	15
Figure 7. Training Set.....	16
Figure 8. Training, Test, Validation Division.....	17
Figure 9. Features.....	19
Figure 10. GloVe Embeddings.....	20
Figure 11. Kappa Score Formula.....	21
Figure 12. Cohen Kappa Score Statistics.....	22
Figure 13. Feed Forward Neural Network.....	25
Figure 14. Formula for Neural Network.....	27
Figure 15. Recurrent Neural Network.....	31
Figure 16. Formulae for RNN.....	33
Figure 17. LSTM Network.....	34
Figure 18. LSTM Cell.....	35
Figure 19. LSTM Equations.....	37
Figure 20. System Architecture.....	38
Figure 21. Test Set.....	58

LIST OF TABLES

Table 1. Represents the output for the algorithms.....	56
--	----

.

1. INTRODUCTION

Imagine if there are some thousands of essays written from a university, school, or competition and there are less number of human graders then it's nearly impossible to evaluate such many essays in less time. The cost and the bias can be minimized if there exists an automated system for grading the essays.

Essays help students enhance their creativity, expand knowledge, and to express their opinions and ideas. Essays are best suited over true/false or multiple-choice questions and answers which allows for testing students' thinking skills, and remembering power, thoughts, and ideas. The automated essay scoring system allows a faster and easier method for evaluating essays. It is the process of automating the scoring system of essays without any human intervention and using computer programs. In our Automated essay grading system, we used certain models and algorithms to grade the essays which match with that the human graded scores with minimum error. We used two methods, word embeddings and feature extraction to evaluate the essay quality. We used Feed Forward neural networks and Long Short-Term Memory to train the essays and validate them. We also made a comparative study and analysed the accuracies of each model and algorithm we used. We implemented a system that grades the essays by considering proxies and aspects of how human graders consider while evaluating essays.



Fig 1. Automatic Essay Grading

1.1. OVERVIEW

Assessments are included in almost every school's syllabus, essays being one of them to hone the skills. Essays are the best way to express ideas and are better than the straightforward questions asked by teachers to evaluate the performance of the student. Essays provide a platform to test and enhance the skills of the students therefore used for assessing students. It is of vital importance to use automated systems to make the process efficient and allow faster grading of the essays.

1.2. MOTIVATION

There exists large number of essays and even the cost can increase. The cost of grading can be reduced and even the bias can be diminished by the automatic grading system. This system is a boon to many universities and schools that are in need of an effective scoring system to evaluate and grade essays. The grading standard can also be unified among different schools. The potential for the usage of this system will increase as it keeps receiving more number of essays thereby increasing the accuracy by a significant amount.

1.3. PROBLEM STATEMENT

The problem statement is Automatic essay grading system and this is used to grade the essay and give score in a given range. Each essay consists of properties called as features and there is no limit in the existence of these features. We have proposed a wide variety of features for our project and extracted these features. There exists dependency between the style of essay, length of essay, topic of the essay and the features that have been extracted from the essay. Manual essay scoring is a tough procedure to human scorers. Due to the COVID-19 epidemic there was a tremendous shift in teaching methodology from offline classes to online mode of education.



Fig 2. AES Explained

1.4. PROJECT SCOPE

Automating the process of grading essays is our aim and making work easy for teachers and evaluators. Teachers take lot of amount and work to grade the scores manually which is huge amount of work when there are a greater number of essays. So, to address this problem we must look for an alternative such as an automated process for scoring essays by software programs.

One has to consider various factors and metrics while grading an essay considering both style and content of the essay. Content and style are both important for an essay content is useful for checking if the essay is conveying the meaning and topic relevance and style is important for checking fluency, vocabulary, grammar structure and many other aspects of writing essay. So, creating an automated essay grading system considering the above aspects is necessary for accurate and right way of grading the essays. So, the system is expected to consider above aspects of evaluating the essay.

2. LITERATURE SURVEY

2.1. EXISTING SYSTEMS

Existing systems describe the current models that used for commercial or research purposes in the real world. They are as follows

2.1.1. C-RATER

C-Rater is an automatic scoring engine that has been advanced to attain responses to content-based essay solution questions. It isn't always surely a string-matching program, as an alternative it uses predicate argument structure, pronominal reference, morphological evaluation and synonyms to assign complete or partial credit score to a brief solution query. Crater has been utilized in studies: National Assessment for Educational Progress (NAEP) and a state-wide evaluation in Indiana. In each study, C-Rater agreed with human graders approximately 84% of the time. C-Rater's purpose is to map the student's reaction onto the version, and in so doing to demonstrate the correctness of the reaction or, failing that, its incorrectness or inadequacy.

The version is built with the aid of using hand however the mapping is completely automated. Because a version is required, the query have to have a unmarried accurate solution or a variety of correct answers. This approach that C-Rater isn't always designed to attain open-ended questions, together with ones that ask for examples taken from private revel in, or for an opinion, or for progressive methods to resolving a conflict. To rating essay solution responses, the scoring

engine have to be capable of apprehend whilst a concept is expressed and whilst it isn't always.

We think about the set of accurate responses as being paraphrases of the precise solution, and of the C-Rater scoring engine as a paraphrase recognizer that identifies the participants of this set. It is critical to be aware that C-Rater is not surely matching words, the paraphrases have to obey syntactic constraints. A query may be scored with the aid of using C-Rater if there may be a finite variety of standards that fulfill it. Thus, an open-ended query inquiring for an opinion or for examples from the student's very own revel in isn't always a query for C-Rater.

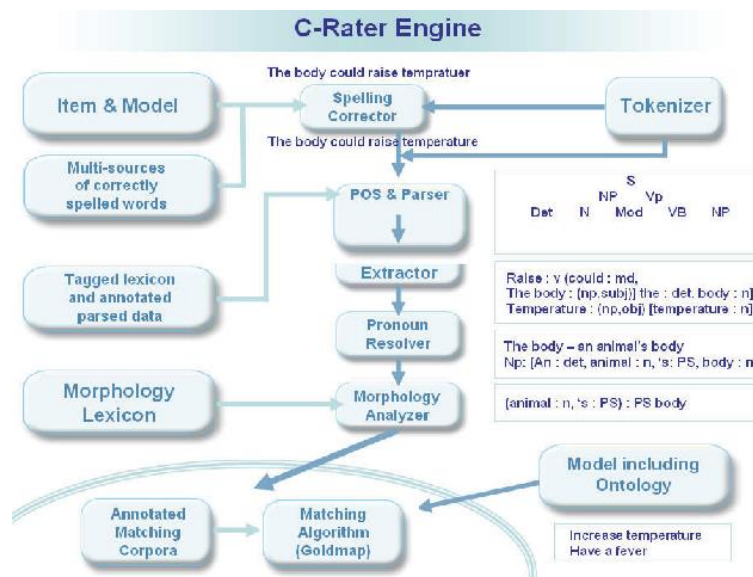


Fig 3. C-Rater

2.1.2. E-RATER

E-Rater became evolved via way of means of Burstein and others. E-Rater makes use of the NLP device for parsing all sentences with inside the essay. Essays are compared against a set of human-graded essays to determine their worth. With E-Rater, an essay that stays on topic, has a strong, cohesive, and well-prepared argument structure, and presents a variety of phrases and grammatical structure will receive a rating on a six-factor scale at the better end of the scale.

Software is designed to pick out functions with inside the textual content that replicate writing features laid out and is presently completed via way of means of 5 foremost complete modules. Three modules identify functions that can be utilised like a scoring manual standards for an essay's syntactic diversity, conceptual enterprise, and language use. To choose functions for essay grading, a fourth impartial module is used. Many other available systems are significantly more difficult and require more education than E-Rater.

e-Rater construct

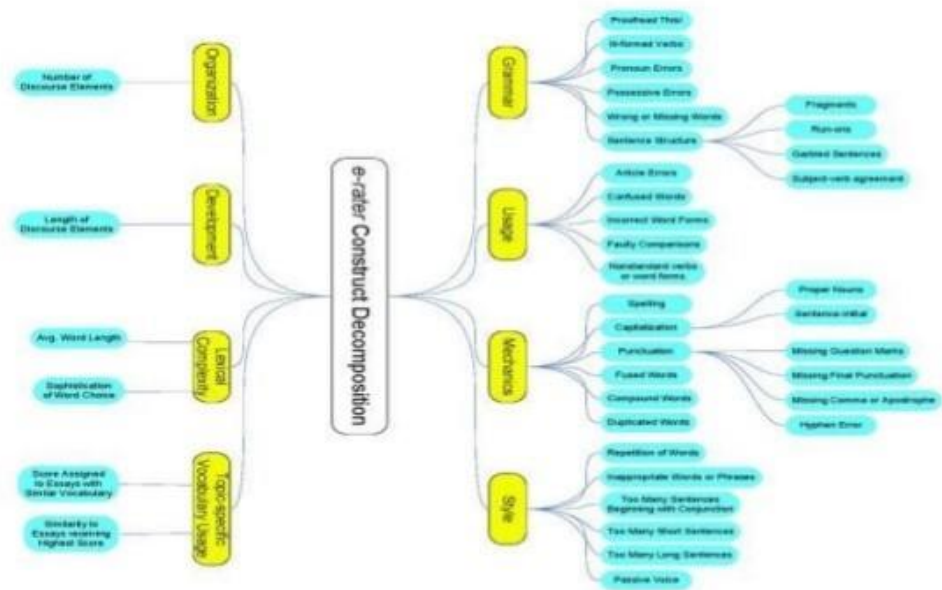


Fig 4. E-Rater

2.2. RELATED WORK

It has been a long while since the research on AES started. The primary machine named Project Essay Grade (PEG) [3] for the instructional evaluation turned into visible. Intelligent Essay Assessor (IEA) [4] uses a set of rules to produce semantic matrices and calculates the similarity among the matrices. This is believed to be near how the human grader grades the essays besides the contextual references. Since deep getting to know turned into added into language processing, an increasing number of researchers have finished associated research.

To perform sentiment evaluation of brief writings, Cicero Nogueira dos Santos [7] suggested a deep convolutional neural community that specialises in certain stages of evaluation, from character-degree to sentence-degree facts. Taghipour and colleagues (Taghipour et al.) are a group of researchers who have [8]. For automated essay grading, they combined neural networks and found that these technologies perform better than topologies that rely heavily on hand-crafted functions. [9]. According to Huygen and Lucio's paper [2], using a 2-layer neural community to train phrase vectors simultaneously, Deep Learning techniques achieved a common Kappa rating of 0.9447. We recommend this implementation, which was influenced by the aforementioned work and took its technique from it.

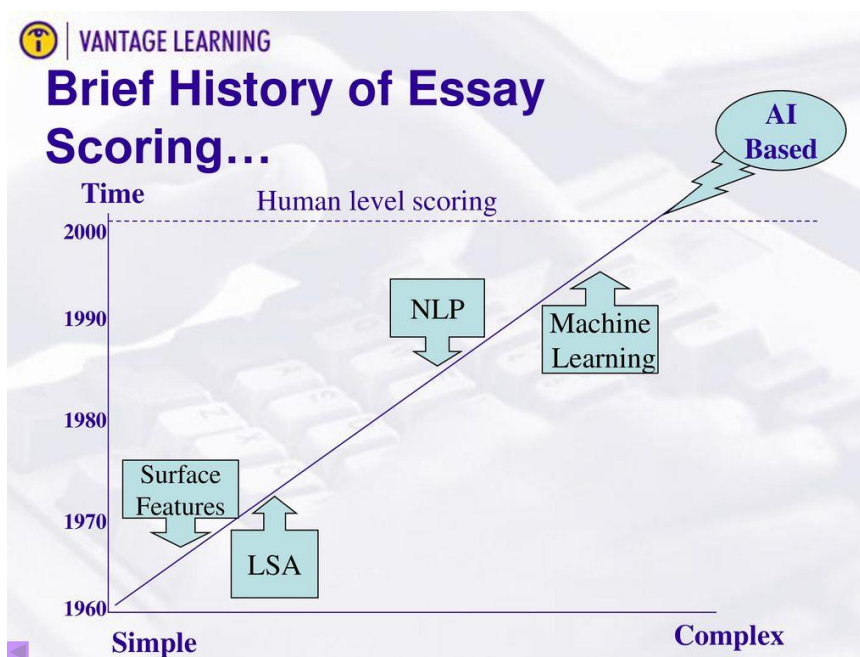


Fig 5. History of Essay Grading

2.2.1 A PAPER ON AUTOMATIC SCORRING OF ESSAYS RELATED TO NEURAL NETWORKS

This is one of the earliest paper that implemented the ideas of deep gaining knowledge of to computerized essay scoring. The paper makes use of an extended quick time period primarily based totally recurrent neural community to seize the semantics of the essay. These vectors had been convoluted with 1D convolution to extract the crucial data and carry out a dimensional reduction. The contextual data extracted from recurrent layer turned into then given to ‘suggest over time’ which calculates the suggest of expected rankings over all the time times and produces the very last score.

2.2.2 A PAPER ON ESSAY SCORING OF SOURCE DEPENDENT ESSAYS WHERE THE APPROACH IS CO-DEPENDENT

This paper gives an research of the usage of co-interest primarily based totally neural community for source-established essay scoring. The significance of every a part of essay extra as it should be also, this paper suggests that the co-interest primarily based totally neural community affords a dependable rating prediction of source-established responses. The paper used Glove embeddings to extract phrase vectors and carried out a convolutional interest over the embeddings. LSTM become used to extract the semantic records and a co-interest layer calculated the similarity among the essay and the article, therefore the rating become predicted.

2.3 TOOLS AND TECHNOLOGIES

The project is realized with the core architecture of LSTM approach to the process of free text assessment and scoring. In doing so, we have utilized certain tools and technologies.

TensorFlow

TensorFlow is a bendy structure permits clean that allows for easy computation deployment over a variety of platforms and from computers to group of networks to cellular and aspect things. It changed into utilized in growing the deep neural community model.

NLTK

NLTK (Natural Language Toolkit) is one of the maximum effective NLP libraries, which includes programs to make machines recognize human language and respond to it with the best response. NLTK changed into used to carry out pre-processing of textual content facts to match the model.

3. SYSTEM ANALYSIS

Our aim is to build an implementation that can take in an essay and grade it automatically. We shall implement feature extraction and to train the dataset and validate them we shall use neural networks such as feed forward, Long Short Term Memory to evaluate these essays with goal of achieving minimum error and higher kappa values.

3.1. DATASET

We use the dataset provided by the Hewlett Foundation for the Kaggle competition Automated Student Assessment Prize. To speed up computation, there are eight alternative sets of essays, each derived from a single prompt. The length of selected essays varies from 150 to 550 words. Two or three teachers graded each essay by hand. In total, there are roughly 13000 samples. Due to a shortage of computing resources, we used essays from SET1 to calculate the system's correctness in this implementation. Ten percent of the essays were utilised for testing, eighty percent for training, and ten percent for validation. Each set has its own grading system.

Essay prompts were separated into two categories: Source Dependent and Persuasive/Narrative/Expository prompts. Persuasive/Narrative/Expository draws on respondent's broad declarative knowledge, whereas Source Dependent examines the respondent's domain-specific knowledge including Physics, Literature, or Social Studies. The level of domain expertise required for each prompt differed. Two graders graded the essays as a whole.

The parameters in the dataset are as follows:

1. essay set: The set of a given essay.
2. essay id: A unique identifier for each individual student essay.
3. rater1 domain1: Rater 1's domain 1 score
4. essay: The ascii text of a student's response
5. domain1 score: Resolved score between the raters
6. rater2 domain1: Rater 2's domain 1 score.

Statistics of ASAP dataset.

Prompt	Number of Essays	Average Length	Scores
1	1788	350	2–12
2	1800	350	1–6
3	1726	150	0–3
4	1772	150	0–3
5	1805	150	0–4
6	1800	150	0–4
7	1569	250	0–30
8	723	650	0–60

Fig 6. Dataset Statistics

essay_id	essay_set	essay	rater1	rater2	rater3	domain1	rater1	rater2	domain2	rater1_tra	rater1_tra	rater1_tra	rater1_tra	rater1_tra	rater1_tra
19237	7	One time I was going to see my frie	11	12		23				3	3	2	3		
19238	7	One @CAPS1, I was very patient wh	9	11		20				2	2	2	3		
19239	7	Being patient! Patience is a good le	8	11		19				2	2	2	2		
19240	7	Being patient is a very hard thing fo	9	7		16				2	2	2	3		
19241	7	At one @CAPS2 in left you would h	8	8		16				2	2	2	2		
19242	7	I remember one time I was impatie	9	8		17				2	3	2	2		
19243	7	Patience in @CAPS1: Understandin	12	12		24				3	3	3	3		
19244	7	It was my first game. I was pumped	10	9		19				2	3	2	3		
19245	7	Hunting when you are turkey hunti	8	8		16				2	2	2	2		
19246	7	One day, two years ago I was going	7	8		15				2	2	2	1		
19247	7	In my opinion being patient is wher	5	8		13				0	1	2	2		
19248	7	Have you ever had to be patient be	9	12		21				2	3	2	2		
19249	7	You will say that you are board very	6	9		15				0	2	2	2		
19250	7	One time I was patient was when I	11	12		23				3	3	2	3		
19251	7	one time I was in patient. We was	7	4		11				2	2	1	2		
19253	7	One time I had to use patience whe	8	8		16				2	2	2	2		
19254	7	Being patient is hard. I hate waiting	4	6		10				0	1	1	2		
19255	7	When I was @NUM1 yrs old my moi	9	6		15				2	2	3	2		
19256	7	One time when I was patient was w	8	8		16				2	2	2	2		
19257	7	When I was patient I was sitting at	8	8		16				2	2	2	2		
19258	7	We were going to @CAPS1. Me and	9	12		21				2	2	2	3		
19259	7	Patience is a hard thing to have, bu	7	8		15				2	2	2	1		
19260	7	Patience is a hard thing to have, bu	8	8		16				2	2	2	2		

Fig 7. Training Set












 test_set.tsv	13-05-2022 10:43	TSV File	5,146 KB
 training_set_rel3.tsv	13-05-2022 10:43	TSV File	15,942 KB
 training_set_rel3	13-05-2022 10:43	Microsoft Excel 97-2...	19,486 KB
 training_set_rel3	13-05-2022 10:43	Microsoft Excel Work...	6,547 KB
 valid_sample_submission_1_column	13-05-2022 10:43	Comma Separated V...	11 KB
 valid_sample_submission_1_column_no_header	13-05-2022 10:43	Comma Separated V...	11 KB
 valid_sample_submission_2_column	13-05-2022 10:43	Comma Separated V...	37 KB
 valid_sample_submission_5_column	13-05-2022 10:43	Comma Separated V...	84 KB
 valid_set.tsv	13-05-2022 10:43	TSV File	5,171 KB
 valid_set	13-05-2022 10:43	Microsoft Excel 97-2...	6,264 KB
 valid_set	13-05-2022 10:43	Microsoft Excel Work...	2,112 KB

Fig 8. Train, Validation and Test Division

3.2. FEATURES

(NLTK)Natural Language Toolkit was used to produce a total of 8 features. This largely entailed deleting proper noun placeholders, formatting the content, tokenizing, and removing punctuation from essays. The retrieved features can be classified into the following categories:

The features related to writing quality include:

- errors in grammar (e.g., subject-verb agreement)
- usage (e.g., preposition selection)
- mechanics (e.g., capitalization)
- style (e.g., repetitious word use)
- discourse structure (e.g., presence of a thesis statement, main points)
- vocabulary usage (e.g., relative sophistication of vocabulary)
- sentence variety

Several heuristic features were produced that are likely to contribute to a successful essay. In our system we extracted 10 features that are number of words, spelling errors, average word length, number of sentences, number of characters, average words in a sentence, noun count, adverb count, verb count, adjective count.

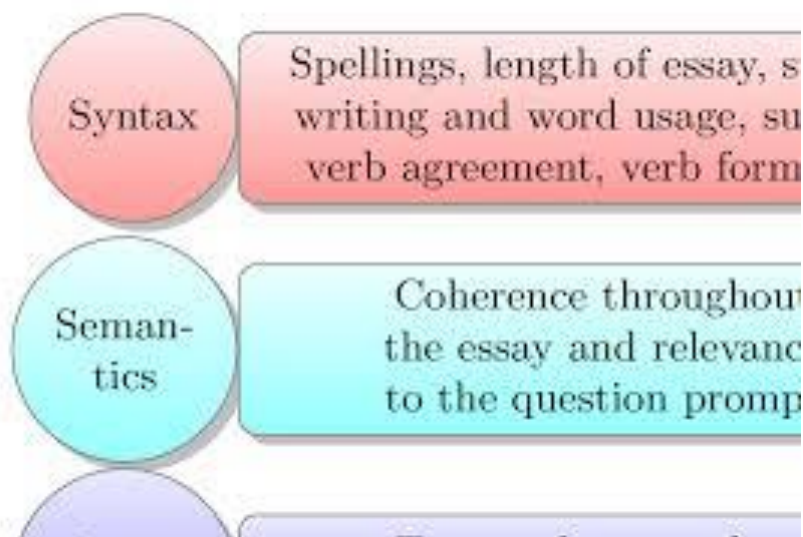


Fig 9. Features

3.3 WORD EMBEDDINGS

Each essay has a vector representation. All punctuation marks are removed from the data before it is processed. Each essay is represented as a vector using GloVe.

We take GloVe word vectors of all the words in an essay, average them to get the essay vector. We experimented with GloVe word vectors with a dimension of 300.

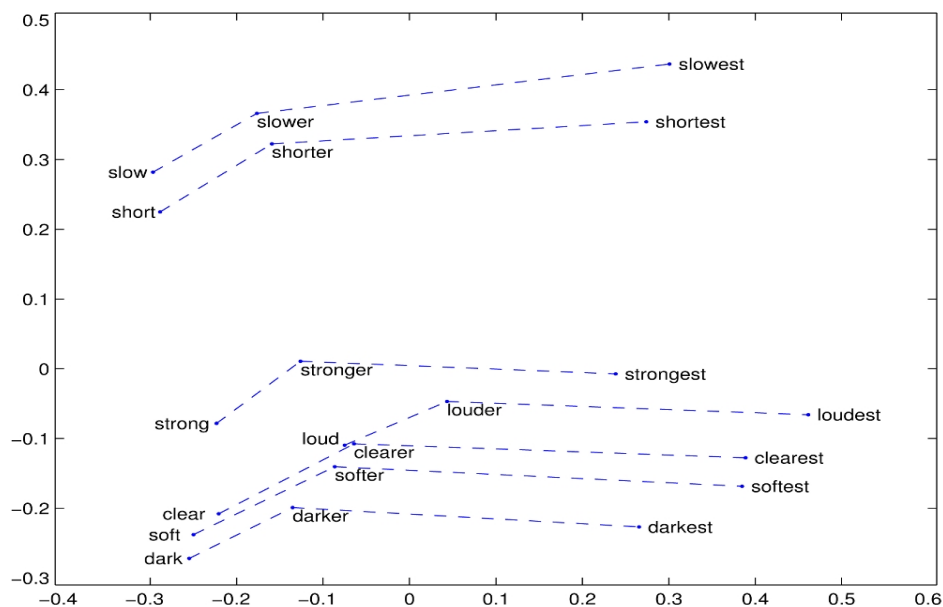


Fig. 10 GloVe Embeddings

4. SYSTEM DESIGN

This project's evaluation metric will be Quadratic Weighted Kappa. When compared to the simple % agreement calculation, this is a more robust indicator. The metric's value ranges from 0 to 1. If the value tends to be close to zero then it means that the agreement between raters is less and if the value is closer to one then the agreement is more. For each set of essays, the quadratic kappa is derived by comparing the system predicted scores to the scores given by human raters.

$$kappa(\kappa) = \frac{P_o - P_e}{1 - P_e}$$

Fig 11. Cohen Kappa Score Formula

4.1. PROPOSED METHODOLOGY

We extracted a total of 10 features from each essay. We selected features that may be used as approximations for what a human evaluator would look for when scoring the essay. For this challenge, we used two different models. We employed a simple Feed Forward network with word embeddings and features extracted in the first method. In the second we used LSTM with feature Extraction. Scores were generated by the models for a different set of test data. Finally these scores were used to compare with human graded scores using kappa scores. The kappa score signifies how close our system generated scores to match with human graded scores.

Cohen's Kappa statistic (κ)	Strength of agreement
< 0.00	Poor
0.00–0.20	Slight
0.20–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Substantial
0.81–1.00	Almost perfect

Fig 12. Cohen Kappa Score statistics

Implementation:

We used two different implementations

- Feed forward neural networks on Glove word embeddings. We implemented four layers on the neural network. The vector dimensions we chose was 300.
- In the next method we trained the feed forward neural networks on extracted features taken from the essays. We achieved higher value of kappa score in this case compared to word embeddings.

LSTM Model Implementation

- We implemented Long Short Term Memory on features extracted from the essays.

4.1.1. FEED FORWARD NEURAL NETWORK

We tested with numerous types of input for our Neural Network. We created an Essay vector for each essay by averaging all of the word vectors in the essay. GloVe vectors were used to represent words in this way. Our bag of words model was built using this method of producing Essay vectors. The first layer is the input layer, which takes in the essays' averaged GloVe vector representation. We used different activation functions like sigmoid and tanh for multiple hidden layers of the network. There is one neuron at the final output layer and produces final outcome by using the Rectified Linear Unit activation function. The following were the hyper-parameters:

- 300-800 number of epochs
- Training 80%, validation 10%, testing 10%
- Dropouts were used for avoiding overfitting
- Mean squared error for checking loss
- Rms for optimization
- Activation functions tanh and sigmoid

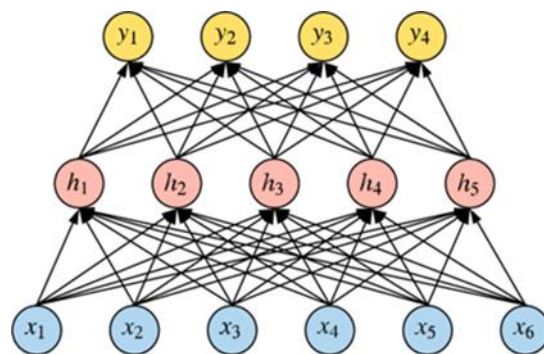
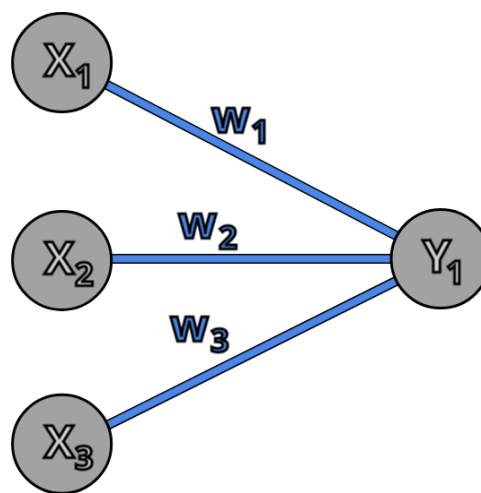


Fig 13 . Feed Forward Neural Network

The feed forward neural network shown above consists of three layers. These layers are called as the input layer, hidden layer and output layer. The input layer is represented by the nodes that are marked with the parameters x_1, \dots, x_6 and the hidden layer is represented by the nodes that marked with the parameters h_1, \dots, h_5 and finally the output layer is represented by the nodes that are marked with the parameters y_1, \dots, y_5 . Generally the number of nodes keep decreasing layer by layer. There can even exist more than one hidden layer and in such a scenario the neural network has multiple hidden layers.

At first, the first layer accepts the inputs and then uses the appropriate functions to calculate the value and then the next layer also does the same actions and finally reaches the last layer. At the output layer we compare the output produced by our network to the actual value and then calculate how different it is. We call this as the cost. Then we propagate this cost back into our network so that all the parameters that are inside our layers are tuned so that the network produces an output that is closer to the actual output and this process repeats until the cost is a very small value and this is when we can say that our neural network has learned something.



$$Y_1 = \text{Activation}(W_1 \times X_1 + W_2 \times X_2 + W_3 \times X_3)$$

Fig 14. Formula for Neural networks

4.1.2. RECURRENT NEURAL NETWORK

Recurrent neural networks are one of the most popular machine learning models, attracting researchers from a wide range of disciplines. Recurrent neural networks are potentially more powerful than feed-forward neural networks and are capable of learning more complicated patterns from data. As a result, in addition to the feed forward neural network, we used recurrent networks in our implementation.

Because the textual data is treated as a single vector, a single feed-forwarding neural network may not perform effectively. In fact, such a model has no recall of any previously ingested data point. There are no links between the data points because they are handled independently. This is not a suitable approach to grade essays automatically. Furthermore, the sequence of words is critical. When reading a text, recurrent neural networks can behave like a human by iterating through each word and taking into account all prior information.

The x in the above diagram represents the input word, while the y represents the predicted term. Each h in the hidden layer holds all information from the previous layer, thus that the latest output incorporates data from all inputs. This starting state is usually set to zero because there is no data before x_1 and h_0 .

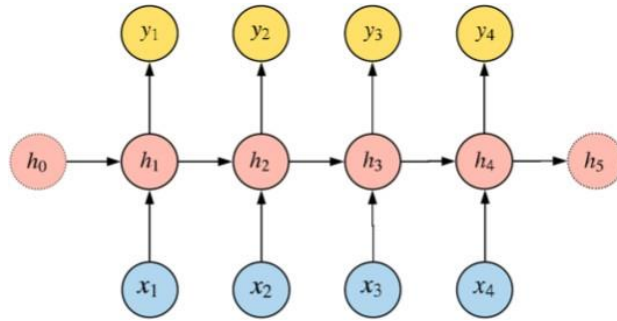


Fig 15. Recurrent Neural Network

In the process of converting the string or the sentence input into the form of vectors the sequence of words in a sentence or the sequence information might get discarded and the output might occur in a jumbled form. Once the sequence information gets discarded the accuracy or the output value tends to get deflected towards the lower side and it will get difficult for the human interpreter to understand the output.

If we take an example of Google Assistant or Amazon Alexa and we ask some questions it will be telling us the answer or the reply in proper sentences. Over here the sequence information is very much important and in order to control this sequence information we have used recurrent neural networks.

Recurrent neural networks do not really consist of layers they just represent one layer that is laid out as a sequence during the time. After supplying the vector of the first word and the activation function we get the output and we supply this output that is second activation function along with the vector of the next word to the same network and then we get our output individually. In the given diagram above the activation functions are represented by parameters $h_0 \dots h_5$ and the inputs and outputs are represented by $x_0 \dots x_4$ and $y_0 \dots y_4$.

$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}$$

Fig 16. Formulae for RNN

4.1.2.1. LSTM

The paper utilizes a long short term based recurrent neural network to capture the semantics of the essay. The words of the essay were represented by one-hot encoded vectors. These vectors were convoluted with 1D convolution to extract the essential information and perform a dimensional reduction. The contextual information extracted from recurrent layer was then given to 'mean over time' which calculates the mean of predicted scores over all the time instances and produces the final score.

We used a LSTM with a score layer at the end. The LSTM takes a sequence of word vectors that match to the essay's words and generates a vector that encapsulates the information in the essay. This vector is converted into a score in the desired range by the scoring layer at the end. The feed forward networks mentioned above use a bag of words paradigm, which has limited power. As a result, the decision to use an LSTM-based architecture was made so we can capture sequence information.

The diagram below depicts the steps taken by the LSTM to construct a new state utilising the old state and current input. The state vector is split into two components by LSTM: state component h and memory component c . Input gate I forget gate f , and output gate o were examples of other gates. The forget gate determines how much of the prior memory will be used in the computation.

The input gate controls how much of the previous state and input x should be kept. The results of three other gates are combined to create the new memory cell c . Finally, a new state has been founded. LSTM can help us get a better result from sequential textual input and avoid the vanishing gradient problem.

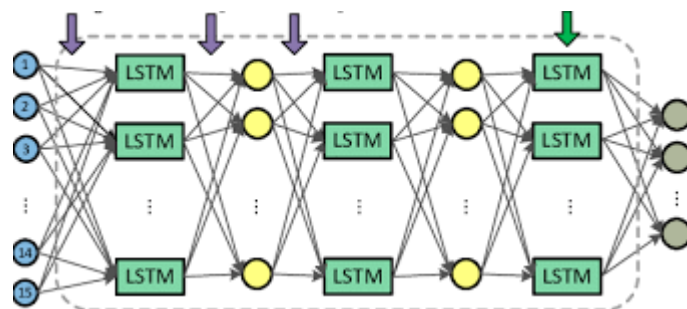


Fig 17. LSTM Network

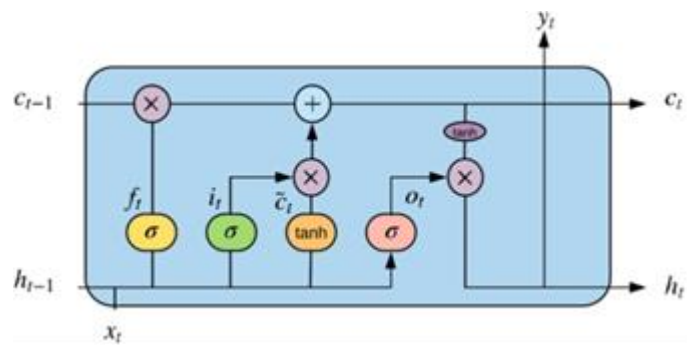


Fig 18. LSTM Cell

As shown above in order to predict a word that is almost at the end of the network we have to know the word that was predicted in the early stages or at the beginning of the network training and because of the vanishing gradient problem the traditional RNN's have short term memory so they don't remember what occurred at the starting of the sentence and they have a short-term memory of just a few words nearby.

The hidden state is actually called as short-term memory and it is called as a memory cell because it is actually containing short term memory. Therefore, if we want to have or remember long term memory then we have to introduce another memory called as the long-term memory. In the above diagram there are two states c and h . The h is the hidden state that is the short-term memory and c is the self-state that is the long-term memory. We store the meaningful words or key words into this long-term memory. Whenever it comes to the forget gate the network understands that it has to discard that word and store the new important word and the input gate adds a memory of the new word and forget gate discards the memory of the old word.

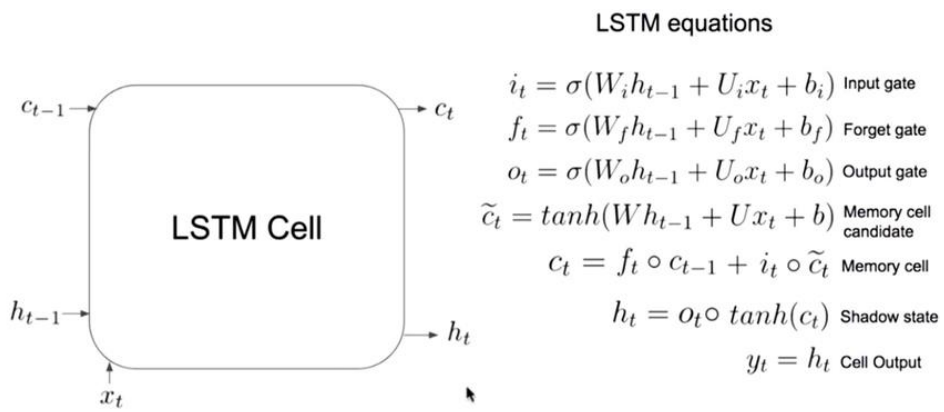


Fig 19. LSTM Equations

4.2. SYSTEM ARCHITECHTURE

- Feed forward neural networks on Glove word embeddings. We implemented four layers on the neural network. The vector dimensions we chose was 300.
- In the next method we trained the feed forward neural networks on extracted features taken from the essays. We achieved higher value of kappa score in this case compared to word embeddings.
- We implemented Long Short Term Memory on features extracted from the essays.

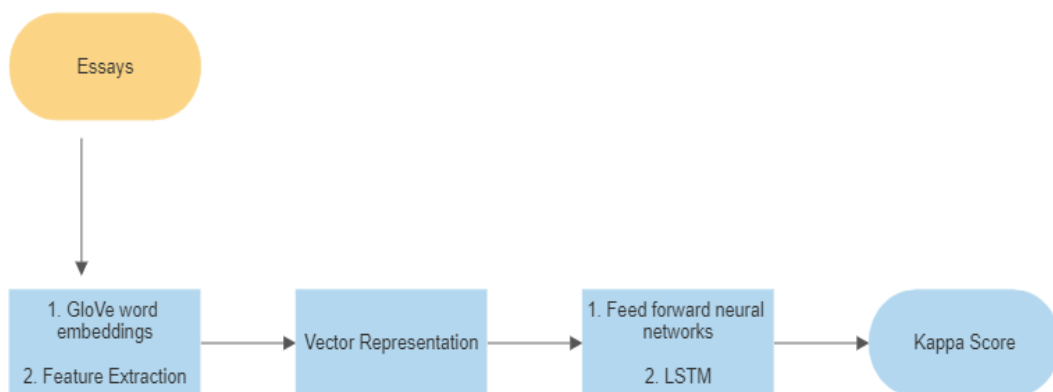


Fig 20. System Architechure

4.2.1. MODEL FOR WORD EMBEDDINGS

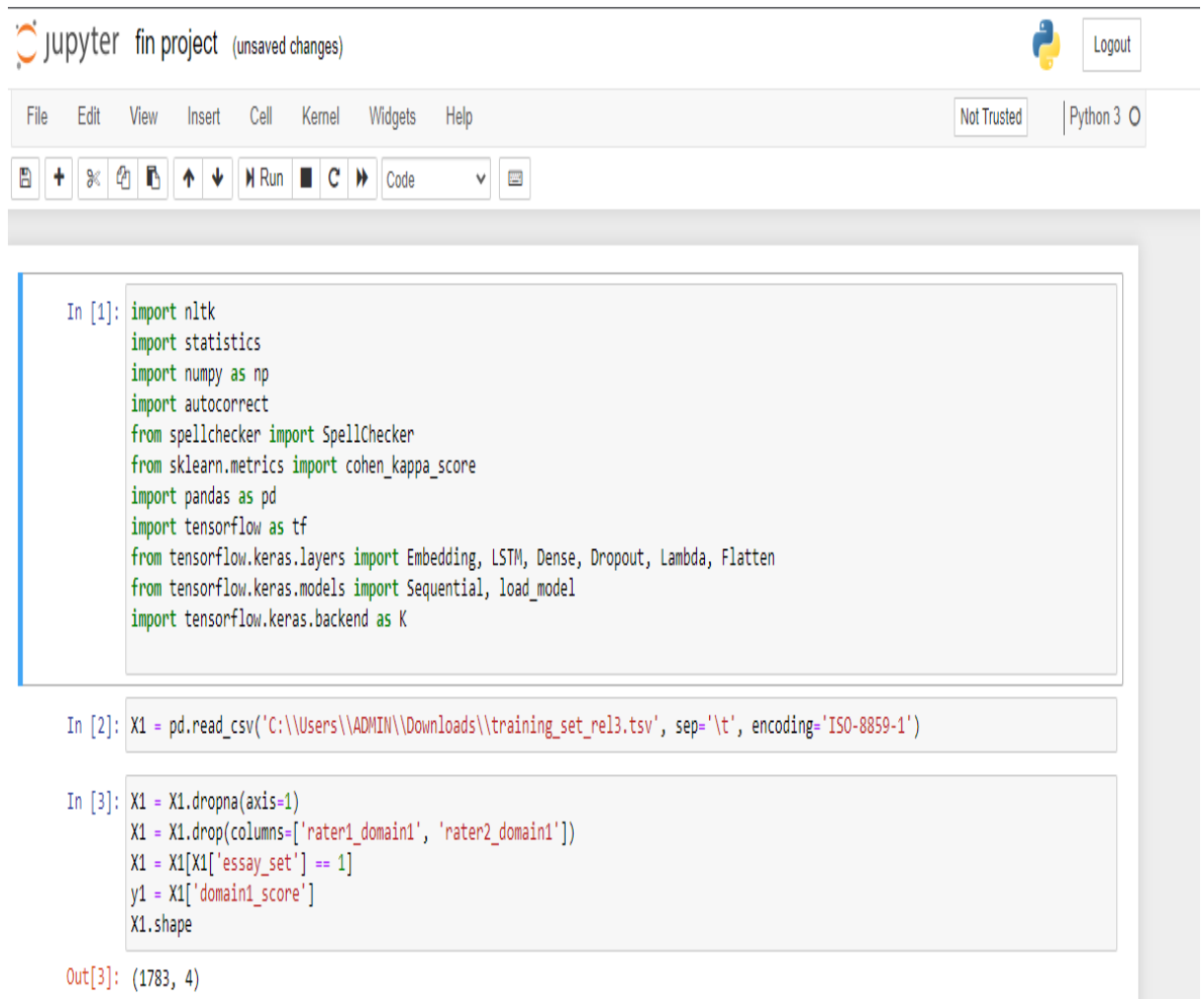
Firstly, we have imported the sequential model in order to build the model and in the model we took 128 nodes totally for the first layer and gave the word embeddings matrix as the input after constructing it from glove word embeddings and then applied the sigmoid activation function which gives the value in the range of 0 to 1.

In the next layer we took 64 as the total number of nodes and similarly applied the sigmoid activation function with a dropout of 0.2. In the third layer we took 24 nodes and did the same process again and finally we got the one output node in the last layer by applying the RELU activation function which gives values in the range of $\max(0, x)$.

The optimizer that we have used is the rmsprop optimizer that is used to send the value again back to network so that each node can get an idea of the loss that occurred and subsequently modify the weights to reduce the loss and we used the mean square to reduce the loss and find the loss

5. PSEUDO ALGORITHM

Dataset loading



The image shows a Jupyter Notebook interface with the title "fin project (unsaved changes)". The top bar includes a "Logout" button and a "Not Trusted" warning. The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar includes icons for saving, adding cells, running, and other standard Jupyter actions. The notebook contains three code cells. The first cell imports various libraries including nltk, statistics, numpy, autocorrect, spellchecker, sklearn, pandas, tensorflow, and keras. The second cell reads a CSV file from a local path. The third cell performs data cleaning and selection on the loaded data. The output of the third cell is displayed below it.

```
In [1]: import nltk
import statistics
import numpy as np
import autocorrect
from spellchecker import SpellChecker
from sklearn.metrics import cohen_kappa_score
import pandas as pd
import tensorflow as tf
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Lambda, Flatten
from tensorflow.keras.models import Sequential, load_model
import tensorflow.keras.backend as K

In [2]: X1 = pd.read_csv('C:\\Users\\ADMIN\\Downloads\\training_set_rel3.tsv', sep='\\t', encoding='ISO-8859-1')

In [3]: X1 = X1.dropna(axis=1)
X1 = X1.drop(columns=['rater1_domain1', 'rater2_domain1'])
X1 = X1[X1['essay_set'] == 1]
y1 = X1['domain1_score']
X1.shape

Out[3]: (1783, 4)
```


Glove file loading

```
def loadGloveModel(gloveFile):
    print("Loading Glove Model")
    f = open(gloveFile, 'r', encoding="utf8")
    model = {}
    for line in f:
        try:
            splitLine = line.split()
            word = splitLine[0]
            embedding = np.array([float(val) for val in splitLine[1:]])
            model[word] = embedding
        except:
            print(word)
    print("Done.", len(model), " words loaded!")
    return model
```

Glove embeddings and feature extraction

```
In [5]: glove_embeddings = loadGloveModel('C:\\Users\\ADMIN\\Downloads\\glove.6B\\glove.6B.300d.txt')
```

```
Loading Glove Model  
Done. 400000 words loaded!
```

```
In [7]: def typos(words):  
        spell = SpellChecker()  
        misspelled = spell.unknown(words)  
        return misspelled, len(misspelled)  
def spelling_errors(data):  
    words = tokenize_words(data)[0]  
    return int(typos(words)[1])
```

```
In [8]: def get_stop_words():  
        stop_words = nltk.corpus.stopwords.words('english')  
        return stop_words
```

```
In [9]: def tokenize_sentences(data):  
        sent_token = nltk.tokenize.sent_tokenize(data)  
        return sent_token, len(sent_token)  
def sent_count(data):  
    return tokenize_sentences(data)[1]
```

Number of words, Spelling errors, number of sentences, average word length

```
In [10]: def tokenize_words(data, punc=True, remove_stop_words=False, lower=True):
        if lower:
            word_tokens = nltk.tokenize.word_tokenize(data.lower())
        else:
            word_tokens = nltk.tokenize.word_tokenize(data)
        if punc:
            word_tokens = [word for word in word_tokens if word.isalnum()]
        if remove_stop_words:
            stop_words = get_stop_words()
            word_tokens = [word for word in word_tokens if word not in stop_words]

        return word_tokens, list(set(word_tokens)), len(word_tokens)
    def word_count(data):
        return tokenize_words(data)[2]
```

```
In [12]: def token_length(data):
        words = tokenize_words(data)[0]
        len_of_tokens = {}
        for word in words:
            len_of_tokens[word] = len(word)
        return len_of_tokens
    def avg_length(words):
        return statistics.mean(token_length(words).values())
```

Average number of words in a sentence, parts of speech tagging

```
In [13]: def sentence_length(sent):  
    len_of_sent = []  
    for s in sent:  
        len_of_sent.append(tokenize_words(s)[2])  
    return len_of_sent, statistics.mean(len_of_sent)  
def avg_word_sentence(data):  
    return sentence_length(tokenize_sentences(data)[0])[1]
```

```
In [14]: def char_count(data):  
    return len(data.lower().replace(' ', ''))
```

```
In [15]: def pos_tags(data):  
    sent = tokenize_sentences(data)[0]  
  
    noun_count = 0  
    adj_count = 0  
    verb_count = 0  
    adv_count = 0  
  
    for s in sent:  
        tags = nltk.pos_tag(tokenize_words(s)[0])  
        for tag in tags:  
            if tag[1][0] == 'N':  
                noun_count += 1  
            elif tag[1][0] == 'J':  
                adj_count += 1  
            elif tag[1][0] == 'V':  
                verb_count += 1  
            elif tag[1][0] == 'R':  
                adv_count += 1  
    return noun_count, adj_count, verb_count, adv_count
```

Feed forward neural network on word embeddings

```
In [23]: def get_model_nn():
          model = tf.keras.Sequential()
          model.add(tf.keras.layers.Dense(128, input_dim=300, activation='sigmoid'))
          model.add(Dropout(0.2))
          model.add(tf.keras.layers.Dense(64, activation='sigmoid'))
          model.add(Dropout(0.2))
          model.add(tf.keras.layers.Dense(24, activation='sigmoid'))
          model.add(Dropout(0.2))
          model.add(tf.keras.layers.Dense(1, activation='relu'))
          print(model.summary())
          model.compile(optimizer='rmsprop', loss='mean_squared_error', metrics=['accuracy'])
          return model
```

```
In [26]: embedding_matrix = np.zeros((len(X1.index), 300))
print(embedding_matrix.shape)
for i in range(0, len(X1.index)):
    tokens = tokenize_words(X1.loc[i]['essay'])
    count = 0
    for token in tokens[1]:
        try:
            embedding_matrix[i] = np.add(embedding_matrix[i], glove_embeddings[token])
            count += 1
        except KeyError:
            pass
    embedding_matrix[i] = np.divide(embedding_matrix[i], count)
```

(1783, 300)

```
In [27]: embedding_matrix_train = embedding_matrix[:len(X1_train.index)]
embedding_matrix_val = embedding_matrix[len(X1_train.index):len(X1_train.index) + len(X1_val.index)]
embedding_matrix_test = embedding_matrix[len(X1_train.index) + len(X1_val.index):]
```

Normalization for data centring and removing outliers

```
n [33]: def normalize(data,mean,std):  
        return ((data-mean)/std)  
  
X1_norm = X1.copy(deep=True)  
X1_norm['num_chars'] = X1_norm['num_chars'].apply(normalize,args=[X1.mean(axis=0)['num_chars'],X1.std(axis=0)['num_chars']])  
X1_norm['num_words'] = X1_norm['num_words'].apply(normalize,args=[X1.mean(axis=0)['num_words'],X1.std(axis=0)['num_words']])  
X1_norm['num_sents'] = X1_norm['num_sents'].apply(normalize,args=[X1.mean(axis=0)['num_sents'],X1.std(axis=0)['num_sents']])  
X1_norm['avg_word_length'] = X1_norm['avg_word_length'].apply(normalize,args=[X1.mean(axis=0)['avg_word_length'],X1.std(axis=0)['avg_word_length']])  
X1_norm['noun_count'] = X1_norm['noun_count'].apply(normalize,args=[X1.mean(axis=0)['noun_count'],X1.std(axis=0)['noun_count']])  
X1_norm['adj_count'] = X1_norm['adj_count'].apply(normalize,args=[X1.mean(axis=0)['adj_count'],X1.std(axis=0)['adj_count']])  
X1_norm['verb_count'] = X1_norm['verb_count'].apply(normalize,args=[X1.mean(axis=0)['verb_count'],X1.std(axis=0)['verb_count']])  
X1_norm['adv_count'] = X1_norm['adv_count'].apply(normalize,args=[X1.mean(axis=0)['adv_count'],X1.std(axis=0)['adv_count']])  
X1_norm['spelling_errors'] = X1_norm['spelling_errors'].apply(normalize,args=[X1.mean(axis=0)['spelling_errors'],X1.std(axis=0)['spelling_errors']])  
X1_norm['avg_word_sent'] = X1_norm['avg_word_sent'].apply(normalize,args=[X1.mean(axis=0)['avg_word_sent'],X1.std(axis=0)['avg_word_sent']])  
print(X1_norm.head())
```

Feed forward neural network on feature extraction

```
In [34]: def get_features_model_nn():  
        model = tf.keras.Sequential()  
        model.add(tf.keras.layers.Dense(10, input_dim=10, activation='tanh'))  
        model.add(tf.keras.layers.Dense(4, activation='tanh'))  
        model.add(tf.keras.layers.Dense(1, activation='relu'))  
        model.compile(optimizer='rmsprop', loss='mean_squared_error', metrics=['accuracy'])  
        return model
```

```
In [36]: features = X1_norm[['num_chars', 'num_words', 'num_sents', 'avg_word_length', 'noun_count', 'adj_count', 'verb_count', 'adv_count', 'spe]  
features_array = np.asarray(features)
```


LSTM on feature extraction

```
In [63]: def get_features_model_lstm():  
  
    model = Sequential()  
    model.add(LSTM(128, input_shape=[1,10], activation = 'tanh',return_sequences=True))  
    model.add(Dropout(0.4))  
    model.add(Dense(32, activation='tanh'))  
    model.add(Dropout(0.4))  
    model.add(Dense(1, activation='relu'))  
  
    model.summary()  
    model.compile(loss='mean_squared_error', optimizer='rmsprop', metrics=['accuracy'])  
  
    return model
```

Outputs:

Features extraction

essay_id	essay_set	essay	domain1_score	num_chars	num_words	num_sents	avg_word_length	avg_word_sent	spelling_erros	noun_count	adj_
0	1	1	Dear local newspaper, I think effects computer...	8	1538	331	16	5.006369	20.687500	4	74
1	2	1	Dear @CAPS1 @CAPS2, I believe that using compu...	9	1870	418	20	5.000000	20.900000	11	104
2	3	1	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...	7	1263	279	14	4.965986	19.928571	4	79
3	4	1	Dear Local Newspaper, @CAPS1 I have found that...	10	2642	520	27	5.712551	19.259259	28	164
4	5	1	Dear @LOCATION1, I know having computers has a...	8	2105	462	30	5.117347	15.400000	7	106

essay	domain1_score	num_chars	num_words	num_sents	avg_word_length	avg_word_sent	spelling_erros	noun_count	adj_count	verb_count	adv_count
Dear local newspaper, I think effects of computer...	8	1538	331	16	5.006369	20.687500	4	74	19	69	24
@CAPS1 @CAPS2, I relieve that g compu...	9	1870	418	20	5.000000	20.900000	11	104	24	85	17
Dear, @CAPS1 @CAPS2 @PS3 More and more peopl...	7	1263	279	14	4.965986	19.928571	4	79	22	53	14
Dear Local newspaper, @CAPS1 I have found that...	10	2642	520	27	5.712551	19.259259	28	164	41	97	32
Dear ICATION1, now having puters has a...	8	2105	462	30	5.117347	15.400000	7	106	31	89	35

Feed forward neural networks on word embedding

```
In [30]: embedding_model.fit(embedding_matrix_train, y1_train, epochs=300, validation_data=(embedding_matrix_val, y1_val))
```

Epoch 295/300
1427/1427 [=====] - 0s 198us/sample - loss: 1.1694 - accuracy: 0.0000e+00 - val_loss: 1.0861 - val_a
ccuracy: 0.0000e+00
Epoch 296/300
1427/1427 [=====] - 0s 299us/sample - loss: 1.1494 - accuracy: 0.0000e+00 - val_loss: 1.0507 - val_a
ccuracy: 0.0000e+00
Epoch 297/300
1427/1427 [=====] - 0s 200us/sample - loss: 1.1442 - accuracy: 0.0000e+00 - val_loss: 1.0983 - val_a
ccuracy: 0.0000e+00
Epoch 298/300
1427/1427 [=====] - 0s 179us/sample - loss: 1.0894 - accuracy: 0.0000e+00 - val_loss: 1.0656 - val_a
ccuracy: 0.0000e+00
Epoch 299/300
1427/1427 [=====] - 0s 181us/sample - loss: 1.1057 - accuracy: 0.0000e+00 - val_loss: 1.0529 - val_a
ccuracy: 0.0000e+00
Epoch 300/300
1427/1427 [=====] - 0s 183us/sample - loss: 1.1315 - accuracy: 0.0000e+00 - val_loss: 1.0575 - val_a
ccuracy: 0.0000e+00

```
Out[30]: <tensorflow.python.keras.callbacks.History at 0x1d64f9756c8>
```

Generated scores by using above model

```
In [31]: results = embedding_model.predict(embedding_matrix_test).flatten()
results = [int(r) for r in results]
results
```

```
Out[31]: [10,
          8,
          8,
          7,
          7,
          9,
          11,
          8,
          7,
          8,
          7,
          8,
          10,
          8,
          8,
          7,
          8,
          9,
          8,
```

Dataset after normalizing

	essay_id	essay_set	essay \
0	1	1	Dear local newspaper, I think effects computer...
1	2	1	Dear @CAPS1 @CAPS2, I believe that using compu...
2	3	1	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...
3	4	1	Dear Local Newspaper, @CAPS1 I have found that...
4	5	1	Dear @LOCATION1, I know having computers has a...

	domain1_score	num_chars	num_words	num_sents	avg_word_length \
0	8	-0.221126	-0.277644	-0.746351	-0.007471
1	9	0.358690	0.453147	-0.305963	-0.027029
2	7	-0.701395	-0.714439	-0.966546	-0.131474
3	10	1.706937	1.309936	0.464717	2.160992
4	8	0.769102	0.822742	0.795009	0.333307

	avg_word_sent	spelling_erros	noun_count	adj_count	verb_count	adv_count
0	0.419999	-0.542608	-0.532164	-0.708112	-0.204692	0.065043
1	0.447851	0.836959	0.394898	-0.263516	0.432873	-0.589108
2	0.320529	-0.542608	-0.377654	-0.441354	-0.842257	-0.869458
3	0.232804	4.187337	2.249023	1.248110	0.911046	0.812644
4	-0.273018	0.048635	0.456703	0.358918	0.592264	1.092995

Features extraction feed forward neural networks

```
features_model.fit(features_array_train, y1_train, epochs=300, validation_data=(features_array_val, y1_val))
```

```
Epoch 295/300  
1427/1427 [=====] - 0s 60us/sample - loss: 0.5931 - accuracy: 0.0000e+00 - val_loss: 0.5744 - val_ac  
curacy: 0.0000e+00  
Epoch 296/300  
1427/1427 [=====] - 0s 59us/sample - loss: 0.5924 - accuracy: 0.0000e+00 - val_loss: 0.5801 - val_ac  
curacy: 0.0000e+00  
Epoch 297/300  
1427/1427 [=====] - 0s 57us/sample - loss: 0.5916 - accuracy: 0.0000e+00 - val_loss: 0.5785 - val_ac  
curacy: 0.0000e+00  
Epoch 298/300  
1427/1427 [=====] - 0s 57us/sample - loss: 0.5937 - accuracy: 0.0000e+00 - val_loss: 0.5876 - val_ac  
curacy: 0.0000e+00  
Epoch 299/300  
1427/1427 [=====] - 0s 54us/sample - loss: 0.5919 - accuracy: 0.0000e+00 - val_loss: 0.5774 - val_ac  
curacy: 0.0000e+00  
Epoch 300/300  
1427/1427 [=====] - 0s 57us/sample - loss: 0.5921 - accuracy: 0.0000e+00 - val_loss: 0.5772 - val_ac  
curacy: 0.0000e+00
```

Out[36]: <tensorflow.python.keras.callbacks.History at 0x1d65139c988>

LSTM model on feature extraction

```
In [64]: lstm_model2 = get_features_model_lstm()
lstm_model2.fit(features_array_train, y1_train, epochs=500, validation_data=(features_array_val, y1_val))

Epoch 495/500
1427/1427 [=====] - 0s 143us/sample - loss: 0.6769 - accuracy: 0.0000e+00 - val_loss: 0.6296 - val_a
ccuracy: 0.0000e+00
Epoch 496/500
1427/1427 [=====] - 0s 147us/sample - loss: 0.6840 - accuracy: 0.0000e+00 - val_loss: 0.6256 - val_a
ccuracy: 0.0000e+00
Epoch 497/500
1427/1427 [=====] - 0s 145us/sample - loss: 0.6756 - accuracy: 0.0000e+00 - val_loss: 0.6193 - val_a
ccuracy: 0.0000e+00
Epoch 498/500
1427/1427 [=====] - 0s 148us/sample - loss: 0.6536 - accuracy: 0.0000e+00 - val_loss: 0.6212 - val_a
ccuracy: 0.0000e+00
Epoch 499/500
1427/1427 [=====] - 0s 148us/sample - loss: 0.6701 - accuracy: 0.0000e+00 - val_loss: 0.6327 - val_a
ccuracy: 0.0000e+00
Epoch 500/500
1427/1427 [=====] - 0s 145us/sample - loss: 0.6727 - accuracy: 0.0000e+00 - val_loss: 0.6279 - val_a
ccuracy: 0.0000e+00

Out[64]: <tensorflow.python.keras.callbacks.History at 0x1d66a1ed988>
```


6. RESULT

We find that the simple feed-forward neural network delivers the best kappa score after reviewing the results of our implementation. Both of our models, on the other hand, were able to detect and score the essay set's score ranges. With a Quadratic Weighted Kappa score of 0.7758, we were able to meet our goal.

Both feed forward neural networks and LSTM showed good accuracy. Feed forward neural networks on features extraction shown the best accuracy. Word embeddings did not show accuracy as expected.

Because essays are responses to specific prompts, some of the attributes that are important to the prompts are extracted by features of essays. We are also confident that, based on our understandings and research studies on various techniques, implementing the system across additional sets of essays in the dataset would boost the system's accuracy.

Model	Word Vector	Kappa
3 – Layer NN	Features	~0.7758
4 – Layer NN	GloVe	~0.6751
LSTM	Features	~0.7634

Table 1. Result Table showing Kappa Score

7. TESTING

Cohen Kappa score achieved by applying the glove embeddings matrix to feed forward neural network-

```
In [32]: percent = cohen_kappa_score(y1_test.values,results,weights='quadratic')  
print(percent)
```

```
0.6751431487536113
```

Cohen Kappa score achieved by applying the extracted features matrix to feed forward neural network-

```
percent = cohen_kappa_score(y1_test,feature_results,weights='quadratic')  
print(percent)
```

```
0.7758512132513198
```

Cohen Kappa score achieved by applying the extracted features matrix to LSTM-

```
percent = cohen_kappa_score(y1_test,lstm_result_2,weights='quadratic')
print(percent)
```

```
0.7634842519685039
```

```
essay_id      essay_set      essay  domain1_predictionid  domain2_predictionid
2383      1      "I believe that computers have a positive effect on people. They help you stay in touch with family in a couple different ways they exercise
y a big role while you're on the computer. They need to move the mouse and press the keys on a keyboard. Your hands learn all the keys from memorization. It'
challenge your mind to be greater and exercise your hands and they make life easier for kids and the average person. This is why, I think computers have goo
2384      1      "Dear @CAPS1, I know some problems have come up where some individuals do not agree with the benefits that computers provide us with. I am wr
ng detail just by viewing articals that the internet and computers offer. I have to admit that you have to sit down and read @CAPS2 which doesn't benefit you
ost of @CAPS2 services are completely free! Instead of wasting minutes on your expositive cellular phone, you can just chat with your friends and family online
2385      1      "Dear to whom it @MONTH1 concern, Computers aren't the reason why people don't want to excersise or veiw the nature. Its the people computers
2386      1      "Dear @CAPS1 @CAPS2, @CAPS3 has come to my attention that some people do not believe in our in technology and their computers. I however beli
ial people. Computers give us access to pictures and information on far away people or places. This is a very useful thing, since @CAPS3 allows you, our loca
ey got there @CAPS3 could be too late. And thus I implore you, our local paper, to support our fight to keep computers. Some people @MONTH1 say that computer
2387      1      "Dear Local newspaper, I think that people have lately spent too much time on their computers. People who have been doing so are increasing t
. It is important to at least exercise for @NUM12 hrs. a day. Do something like take a walk in the park, go hiking, ride a bike, do push ups. There's all kind
ith strangers. In conclusion, you should be concerned about people's use of computers. It increases obecity. It decreases the interaction with others and you
2388      1      "Dear @CAPS1, @CAPS2 is in my firm belief that to many people spend far too much time on their computers these days. Computers can be a very
you do @CAPS2 nepeditivley that @CAPS2 starts to harm you. You won't be able to laugh and joke around with your friends the way you used to because you'll be
our progress how we and just lives in general. So spending to much time on a computer is a serious health issue because of the lack of exercise that you rece
cause @CAPS2 allows no exercise, however when we balance our time between the computer and everything else we got the best of both world message."      2388
2389      1      "Dear local newspaper, @CAPS1 your computer have you so addicted? Computer's have an effect on people because people line to go on myspace, l
everychild that has a computer in there house watches music videos because they have nothing else better to do. Also, if you've noticed on the videos there.
2390      1      "Dear @CAPS1, I agreed with the people that feel that people should spend more time with the family instead of being stuck on the computer or
2391      1      "Dear @CAPS1 @CAPS2, I do not believe that computers are helping today's society. I feel computers are making our situation here in @LOCATION
hey met. (@CAPS10 was a secluded area). They buyer wasn't planning on buying the car he was going to rob the man and kill him. This proves that the internet
2392      1      "Imagine that you walk into a friends home and their home is just actually dirty. That the reason they don't got time to clean is because the
es to stick together and connect if their children are always playing games on the @CAPS3 and the father/mother is on the internet. This can also bring confl
This is one of the most dangerous things to happen. The time is now to help people to get up and out of all these crazy stuff. Would you want to have a lazy
2393      1      "Dear people of the @LOCATION1, I would like to inform you on the dangers of computes. Picture, if you will, an empty world. @CAPS8, there ar
CAPS8, that's hardly probable, of course we'd all got to work."" but I wouldn't be so sure. What would you do if you were ""this close"" to winning your favo
stics go up the life expecting will go down. We will die sooner. Is that the kind of them we want over our beads raising our children? ""This is a computer @
adults. @LOCATION2, teens such as myself. I should know. So please write an article on it, warn the people about the danger they are unknowing subjected to.
2394      1      "We all have computers at home and I think that computer benefit society. Because is that computer's can help people read in some cases. My na
```

Fig 21. Test Set

8. CONCLUSION AND FUTURE WORK

As we predicted, each category's properties contribute to a good prediction. Our final model includes features from all of the categories we tested. On non-context specific writings, our approach performs better. By combining content and advanced NLP features, performance on topic-specific and richer essays can be improved.

The project's purpose was to develop an automated method for grading essays. Simple features are employed in the models, which focus on the essay's sentence patterns and the commonalities of the words used in the essay. The final results show that by including Deep Learning into this project, a model with excellent accuracy may be developed. The huge value of open-source NLP libraries, as well as analysing the sentence structure that acts as a benchmark for a decent essay, proved to be a wonderful learning curve for us in this project.

Because of the limited data set, writings similar to the training essays can be evaluated immediately with acceptable results using this model. The model, on the other hand, can be used as a benchmark for future work in the field of Automated Essay Grading for essays in many domains.

By combining content and advanced NLP features, performance on topic-specific and richer essays can be improved. Complex recurrent neural networks with contextual information can also improve the accuracy of the system and deliver more accurate answers for essays from various domains

9. REFERENCES

- [1] Dong, F., Zhang, Y., & Yang, J. (2017). Attention-based recurrent convolutional neural network for automatic essay scoring. In Proceedings of the 21st conference on computational natural language learning (CoNLL 2017) (pp. 153–162).
- [2] Fonti, V., & Belitser, E. (2017). Feature selection using lasso. VU Amsterdam Research Paper in Business Analytics.
- [3] Abbass, H. A. (2019). Social integration of artificial intelligence: Functions, automation allocation logic and human-autonomy trust. *Cognitive Computation*, 11(2), 159–171.
- [4] Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic text scoring using neural networks. *ArXiv Preprint ArXiv:1606.04289*.
- [5] Boulanger, D., & Kumar, V. (2019). Shedding light on the automated essay scoring process. In Proceedings of the 12th International conference on educational data mining (EDM).
- [6] Liang, G., On, B.-W., Jeong, D., Kim, H.-C., & Choi, G. (2018). Automated essay scoring: A Siamese bidirectional LSTM neural network architecture. *Symmetry*, 10(12), 682.
- [7] Cozma, M., Butnaru, A. M., & Ionescu, R. T. (2018). Automated essay scoring with string kernels and word embeddings. *ArXiv Preprint ArXiv:1804.07954*.
- [8] M. R. Md. Haider Ali Annajiat Alim Rasel Arshad Arafat, "Automated Essay Grading with Recommendation," 2016.
- [9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan,

Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei 2020. Language Models are Few-Shot Learners.

[10] Chollet, F. a. 2015. Keras. Github. Retrieved from <https://github.com/fchollet/keras>

[11] Dimitrios Alikaniotis, Helen Yannakoudakis and Marek Rei. 2016

[12] Fei Dong, Yue Zhang and Jie Yang. 2017. Attention baes recurrent convolutional neural networks for automatic essay scoring. 21st Conference on Computational Natural Language Learning, (pp. 153-162).

[13] Robert Susik. 2020. Recurrent autoencoder with sequence-aware encoding. Computing Research Repository. arXiv:2009.07349

[14] Sammut Claude.2010. Mean Squared Error. In Encyclopedia of Machine Learning (pp. 653--653). US, Springer.

