Automated Essay Grader: Classifying "Good" Writing using Logistic Regression

Tammie Oh
Dept. of Computer Science
Occidental College
Los Angeles, CA
oht@oxy.edu

Jason Lee
Dept. of Computer Science
Occidental College
Los Angeles, CA
jlee9@oxy.edu

***Abstract*—Essays are one of most important for assessing academic excellence, along with the ability to connect other ideas with the ability to recall, but manual assessments take a lot of time. Manual grading takes substantial amount of evaluator's time and hence it is an expensive process. Automated grading if proven effective will not only reduce the time for assessment but comparing it with human scores will also make the score realistic. This project aims to develop an automated essay assessment system by use of machine learning techniques by classifying a corpus of textual entities into small number of discrete categories, corresponding to total grades, 0 to 12. We have implemented our model in Python using specific modules and libraries, such as NLTK, scikit-learn, and pandas. Logistic regression technique was utilized for training the model along with making the use of various other classifications. We intended to train the model, splitting the dataset into 70% training and 30% testing. Then, we were able to see the performance of the model with the following scores: 0.920 for precision, 0.876 for recall, and 0.896 for f1-measure scores.**

I. Introduction

Computerized scoring have been implemented in our daily lives for a few decades. From children to the elderlies, computerized scoring is of significant help in managing even the most basic tasks in life. Numerous researches prove the viability of using computers day by day, but humans still over-rely on some areas such as grading short essays. In fact, this is because human ratings are often based on a rubric set by the grader that can potentially affect the constitution of a "good" writer; hence, determining "good" writing is subject to the grader's opinion. The initial works of computerized grading focused on reliability and repeatability for computer against human gradings. Although in most cases the computerized grading showed better performance than the human counterpart, there is an issue that can mislead the grading. It is unclear to draw the boundaries of constituting what a "good writer" is. There is no specific standard or evaluation scheme to determines if an essay is good or bad, and this problem statement led to our project.

The impacts that computer have on our writings have been in use since the 1960s. Even the most basics of computers such as processing of words, is of great help to many in updating their writings. Many studies have proven that computers hold the capacity to function as a more effective tool. Revision and feedback are important part of the writing. Students in general require input from their teachers for mastering the art of writing. However, responding to student papers can be a burden for teachers. Particularly giving feedback to a large group of students on frequent writing assignments can be hectic from the teacher's point of view. In our model the scores would be much more detail oriented than the ratings provided by two human raters.

Further, grammar checking is one of the more complicated tasks for word processing, and the more irregular and exception-filled the language, the more difficult the problem becomes. The simplest method of fixing grammatical errors, which was used for the experiments for this project, is the process of rules matching, that is, constructing a rule that applies to a given grammar and then checking that the given input follows, or does not follow, that rule. One thing that differs in the methods of grammar checking systems is whether or not the

system is checking for negative or positive grammar. Intuitively, it seems like it might be easier to define the properties that are correct in a grammar, as there are a set number of grammatical configurations that are correct and an infinite number of configurations that are incorrect. The problem is that describing all of the correct configurations for a grammar checker requires that for every check, it must look at every single rule to see if a given example is in the grammar. This process is necessarily slower than a system that uses a relatively few rules per check to see if something is not in the grammar. Since speed is not of great concern for the system in this paper, the rules checking could have been implemented either way, but for simplicity, we chose to implement rules that check for specific errors in grammar instead of using a model of correct grammar to find incorrect examples. For a small system, it is easier to describe a few things in English that are grammatically incorrect than every rule that is correct.

II. Related Works

Zac Rider [1] used various techniques that could be used for grammar checking and constructed rules using POS tagging. To generate rules, the author implemented a hand construction and an algorithm that randomly generated large numbers of rules and compared against a large corpus to find valid rules. In working through his model, the author stated that it was significantly difficult to be able to tell how well the system can find the errors, because it was easier to find corpus that were correct than corpus that intentionally made mistakes and then make note of those mistakes. For the case of ”there”, ”they’re”, and ”their”, he noted that some errors were not accurately discovered because cases like these are contextual spelling errors, thus they can still be found after going through the method. Further, according to the author, the major issue that he had dealt for these grammar rules was checking if a specific or general case does not trigger, affecting the result to falsely output the error percentages. The author concluded that it requires substantial amount of resources to create all of the rules necessary to properly define grammar problems. Using handmade rules take exponential amount of time, and using a random method obscures the rule creation process, but is not always accurate. In addition, the author found out using a random rule generating algorithm tended to generate cases that do not help define a grammatical error.

Leah S. Larkey [2] used several text classification methods to build an automated essay grading model. She trained Bayesian independence classifiers and k-nearest-neighbor classifiers to assign scores to essays that were already graded by humans. The author mentions that these scores were combined with several other summary text measures using linear regression and those were applied to a new set of essays. The author used five different data sets and collected essays from different number of points evaluated written by different age groups. The author noted that the quality of the essays were all advanced – some of the essays were from exams for college students. Similarly to our data set, this author’s data set also had an imbalance, which consisted a higher ratio of essays that had higher scores. The number of features that the author selected was relatively lower than the features in our model. Excluding the pre-processing steps, the only feature the author used was the Expected Mutual Information (EMIM). Her model showed excellent performances, the results varied a bit, but the overall percentages were high. One thing the author found striking was that adding a text complexity feature to her model did not appear to produce much better performance. She noted that additional variables did improve performance on the training data, but the same did not hold in her testing data. She concluded that the agreement between the computerized rating and the manual grading were as good as the agreement among the human graders.

III. Methodology

3.1 Set-up and Pre-processing

This section reports on the approach and methodology of our model. We started by importing the necessary modules and libraries including *nltk*, *scikit-learn*, *pandas*, and *numpy*. Then, we downloaded our data files from the Automated Essay Scoring page on Kaggle. In order to read the data from the downloaded file, we used the *pandas.read_table*. The features and columns that we focused on in the *training_set_rel3.tsv* file [3] include the essay, essay identification number, essay

set number, score given by first rater, and score given by second rater. For this project, we focused on essay set 1 because each essay set had a different scoring criteria.

There were certain features that required us to pre-process some of our data. These features included checking spelling errors and the number of sophisticated words. We created an object to remove stop words, using the *stopwords* module from NLTK. Next, we created a tokenizer to remove whitespace and punctuation for all sentences from each essay. Finally, we used the *WordNetLemmatizer()* from NLTK in order to lemmatize the words from our dataset, *nltk.words* corpus and used that to compare the spelling errors within the essays.

After reading through our datasets collected from Kaggle, we added the necessary columns into the panda data-frame, and moved forward with creating additional features. The first five columns of the data-frame include the essay identification number, the essay set number, the essay itself, and the scores given by both the first and second grader. For our Logistic Regression model, we wanted to find a way to split the test scores into two classes, rather than having 12 different classes. We created another column called "total_score", which added the scores of the first two raters. We classified the variation of the scores into two: any score that was lower than a score of 8 was labeled 0, and a score of 8 or above was labeled as 1.

3.2 Feature Extraction and Calculation

We have selected several features that would help us determine the proficiency of an essay for our model. The measure of getting scores by our model are as follows: (0) number of words, or the length of the essay (per word), (1) number of sentences in an essay, (2) average sentence length, or number of words in each sentence, (3) lexical diversity of the words, or number of unique words, (4) number of spelling errors, (5) number of sophisticated words, and (6) number of grammar errors.

For our first feature, we stored the values of each essay into a list and created a column in the data frame, labeled as "num_words" to display the length of each essay. Next, we counted the number of sentences in the essay by using the *import textstat* module. This module helped us count the number of sentences in each essay. Again, we stored these values into a list and created another column in our data frame and labeled it as "num_sentences". For the third feature, we used feature (0), essay length, and divided it by feature (1), number of sentences in the essay. The corresponding column in the data frame was labeled as "avg_sent_length". We used the *collections* module and *import Counter* to create a variable that would store the number of unique words in an essay. Similarly, we stored these values into a list and created a column labeled as "lexical_diversity". We created a method for our spelling error feature that inputs a list of words as the parameter and returns an integer value representing the number of spelling errors in that list of words. For our database of correct words, we used the nltk's library to retrieve all the English words, and read through the corpus, lemmatizing each words. Then, we added each lemma into a new list. In our method, the words in the essay were also lemmatized for consistency. The method will check if the lemmatized word in the essay exists in our database of correctly spelled words. If such word does not exist, the spelling error integer value will increase by 1. In terms of pre-processing the data beforehand, we ignored words that were all upper-case letters and stop words. The Kaggle dataset replaced all specific locations, names, or other personal information with an "@" symbol in addition to the general terminology. For instance, if a student wrote a specific address, the text was replaced to "@LOCATION", for the case of our model recognizing such word as a spelling error if the word did not exist in the database. We labeled this column as "spelling_error".

Similarly to the spelling checking feature, we created a separate method for feature (5), number of sophisticated words. It essentially takes an input of a list of words as the parameter and returns an integer value for the number of sophisticated words that exist in each essay. For this feature, we downloaded a corpus containing 5000 English words from online [4]. If a word in the essay does not exist in the corpus, the method classifies that specific word as "sophisticated". We also omitted words that were in upper-case letters and stop words, in the same context as the spelling error feature.

Finally, for the grammar error feature, there are

several different grammar rules that were taken from multiple websites and consolidated into ten different grammar rules. First, we examined the case of the ”they’re”, ”their”, and ”there”. We implemented the feature in our model so that our method for this feature would recognize and classify the different cases of each words that are used grammatically correctly, based on the POS tags. If the POS tag after ”their” is a verb, then that would count as a grammar error, due to the fact that ”their” is a possessive pronoun. Similarly, if the POS tag after ”there” is an adjective, noun or past-tense verbs or gerunds, then that would count as a grammar error because ”there” is used as an object. Verbs such as ”is” and ”are” can follow after ”there” , and they are classified as verbs, specifically ”VB”. If the POS tag after ”they’re” is a conjunction or a plural noun, then that would count as a grammar error because ”they’re” is equivalent to ”they are”. If there is a plural noun after the word ”are”, then that is a subject verb disagreement. In addition, conjunctions should not follow the verb, ”are”. The same logic applies to the case of ”your” and ”you’re” and ”this” and ”these”. The feature determines if the word is correctly used based on the POS tags of each word. Next, we implemented few grammar rules related to comma errors, including commas before ”because” or ”however”, commas after a capitalized word, commas followed by quotations, and comma followed by a word that is any form of verbs that count as a grammar error. Further, we applied rules that include subject-verb agreement errors and contractions. If there are any contractions in the essay, then that would count as an error. We followed the convention of treating contractions informal in sophisticated or academic papers. For subject-verb agreement error, if a word’s POS tag is a singular noun, and the following word’s POS tag is a verb, then the verb must be plural, and vice versa. Finally, we included possessive and plural forms of nouns rules. If a noun is a plural and the POS tag of the following word is a noun, then that would count as an error. For instance, the method would recognize ”sisters car” as an error, when it should be ”sister’s car”.

3.3 Evaluation

We wanted to see if there were some features that correlated more with our label than others. Thus, we graphed each feature in relation to the label, and calculated their correlation coefficient scores. Once the features and the labels were all stored into the data frame, we proceeded to build our Logistic Regression model. We first stored all the features into a numpy array, called ”X_feat”, and also stored all the labels into a numpy array, called ”Y_label”. We then split our training and testing data into 70% training data and 30% testing data with a random state of 30, so that the training and testing would split similarly every time we run the code. Then, we used the *scikit-learn* library to build our Logistic Regression model and train the data using our X_train and our Y_train. We used *solver='newton-cg'* [5] to optimize our model for a binary classification problem. Finally, we calculated the precision and recall scores of our model using *LogisticRegression,predict()* function, and used our testing data. We were able to find the weights of all the features by using *logistic_regression.coef_[0]*. This is also shown in Figure 5. This returned the weights of each of the features and its significance in predicting classes. By using *scikit-learn.metrics*, we can print out the precision, recall, and f-measure scores using *classification_report*.

IV. Result and Analysis

4.1 Dataset

The dataset used in this project is from Kaggle. This dataset comprises of eight student-written essays prepared by The Hewlett Foundation. According to the website, the selected essays range from an average length of about 150 to 550 words and were written by students ranging from Grade 7 to Grade 10. For this project, we focused on essay set 1 because each essay set had a different scoring criteria. The length of essay set 1 is 1783. All the essays from this website are human graded.

4.2 Experiment and Results

This section reports the outcomes produced when running our model with a few visual charts and graphs.

Figure 1 shows the precision and recall values, along with f1-score and support values for both

	precision	recall	f1-score	support
0	0.883	0.773	0.824	88
1	0.956	0.980	0.968	447
accuracy			0.946	535
macro avg	0.920	0.876	0.896	535
weighted avg	0.944	0.946	0.944	535

Fig. 1. Precision and Recall

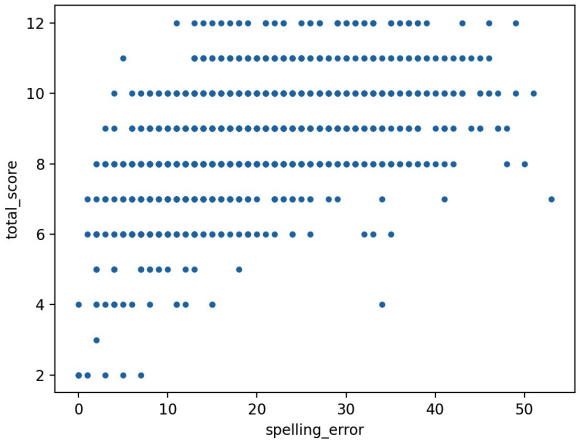

Fig. 2. Correlation Graph: number of spelling errors vs total scores of essays

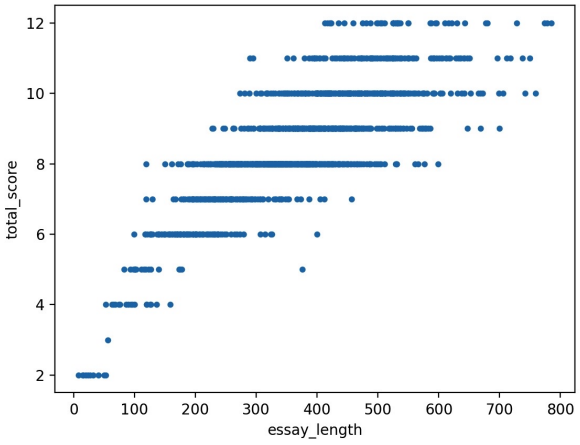

Fig. 3. Correlation Graph: essay length vs total scores of essays

```
essay correlation: 0.7952031738151722
num_sentences correlation: 0.6600608006248951
avg_len_sentences correlation: -0.14949268528676576
lexical_diversity correlation: 0.8345814021046977
spelling_error correlation: 0.5012950219487329
sophisticated_words correlation: 0.6170407524761794
grammar_error correlation: 0.24969265695093448
```

Fig. 4. Results on correlations

```
Feature: 0, Score: 0.01407
Feature: 1, Score: 0.04109
Feature: 2, Score: 0.00041
Feature: 3, Score: 0.03237
Feature: 4, Score: 0.02432
Feature: 5, Score: -0.04427
Feature: 6, Score: -0.01855
```

Fig. 5. Weights of each feature used in the model

essays classified as 0 and 1. The values in the figure were computed using the *scikit-learn* modules. The outcomes are represented in decimal forms except the support column, which is displayed in whole numbers.

Figures 2 and 3 are two correlation graphs of the seven features, which are the number of spelling errors compared with the total essay scores and essay length compared with essay scores, respectively. Both graphs show a positive correlation, meaning there were more spelling errors found in higher scores and essays tended to be longer in highly scored essays. Both correlation graphs were plotted for each feature we used in the model.

Figure 4 displays several percentages for different correlations of essays, including data from Figures 2 and 3. All of the correlations had positive percentages except average sentence length correlation,

	essay_id	essay_set	essay	rater1_dom...	rater2_dom...	total_score	label	essay_length	num_sente...	avg_sent_le...	lexical_dive...	spelling_err...	sophisticat...	grammar_e...
0	1	1	Dear local ...	4	4	8	1	338	15	22.53333	184	16	23	5
1	2	1	Dear @CAP...	5	4	9	1	419	20	20.95000	216	32	35	4
2	3	1	Dear, @CA...	4	3	7	0	279	14	19.92857	167	12	12	6
3	4	1	Dear Local...	5	5	10	1	524	20	26.20000	275	41	51	1
4	5	1	Dear @LO...	4	4	8	1	465	30	15.50000	226	18	27	5
5	6	1	Dear @LO...	4	4	8	1	246	13	18.92308	139	10	21	0
6	7	1	Did you kn...	5	5	10	1	499	29	17.20690	259	15	28	6
7	8	1	@PERCENT...	5	5	10	1	482	35	13.77143	243	14	31	11
8	9	1	Dear reade...	4	5	9	1	443	34	13.02941	244	20	25	8
9	10	1	In the @LO...	5	4	9	1	502	24	20.91667	239	30	41	14
10	11	1	Dear @LO...	4	4	8	1	325	21	15.47619	216	18	33	2
11	12	1	Dear @CAP...	4	4	8	1	392	24	16.33333	171	30	47	11
12	13	1	Dear local ...	4	3	7	0	204	6	34.00000	124	15	34	0
13	14	1	My three c...	3	3	6	0	307	24	12.79167	144	35	46	1
14	15	1	Dear, In th...	3	3	6	0	176	11	16.00000	108	16	17	2
15	16	1	Dear @CAP...	6	6	12	1	528	33	16.00000	283	36	59	4
16	17	1	Dear Local...	4	4	8	1	335	18	18.61111	164	8	26	15
17	18	1	Dear Local...	4	4	8	1	366	15	24.40000	188	9	20	11
18	19	1	I agre waf...	2	2	4	0	66	7	9.42857	53	14	37	0
19	20	1	Well comp...	3	3	6	0	156	11	14.18182	98	7	8	3
20	21	1	Dear @CAP...	4	4	8	1	367	19	19.31579	200	20	30	2
21	22	1	Dear local ...	2	1	3	0	56	2	28.00000	39	2	1	0
22	23	1	Dear local ...	5	5	10	1	521	29	17.96552	249	27	39	10
23	24	1	Dear local ...	6	5	11	1	559	34	16.44118	294	33	38	6
24	25	1	Dear @CAP...	4	4	8	1	293	16	18.31250	139	15	12	4

Fig. 6. Data set, all features are listed following the label column

meaning essays with higher scores had relatively shorter sentences than the essays with lower scores.

Figure 5 shows the weights of each features. The features with greater weights have more importance in determining the specific class.

The dataset of our model was trained beforehand and the output of the training. Each time an essay is inputted, our model classifies the essay into one of the binary classification grades – 0 or 1 for our model. Any grammatical error found will be numbered and displayed. A score is then produced by our Logistic Regression function after taking all the factors into consideration such as going through pre-processing and feature extraction.

V. Discussion

When it comes to determining whether or not a certain word is correct or correctly used, it is crucial to make sure that that word exists in the corpus. However, we faced a limit where some words would not exist in the corpus because some words are more sophisticated than regular words or words that were recently discovered, such as Gmail, Facebook, YouTube, cyberbullying, etc. In addition, in figure 1, the support value for essays classified as 0 were 88 and essays classified as 1 were 447, meaning essays classified as 0 and 1 were not balanced, which might have affected the precision and recall. Consequently, our precision and recall could have been higher for essays classified as 1 because there were a lot more essays labeled as 1 than essays that were labeled as 0. Thus, because more data for label 1 was classified than label 0, the model can classify writers that are ”good” writers better than writers that are ”not good”, since the data in one class is higher than the other and it the higher one can be generalized easier.

Moreover, some of the features showed unexpected outcomes. As shown in figures 2 and 4, we did not expect the correlation between number of spelling errors and the total scores of essays to show a positive correlation, and that is generally not the case. It turned out that spelling error did not have a significantly negative effect on the essay grades. In fact, the model gave a higher score to the essays that contained more spelling errors, and our main hypothesis for this is that there is a limitation in looking for errors solely by looking if the incorrectly spelled word is not in the corpus and therefore determining that the word is misspelled. When writing a formal paper, the convention is that even a single spelling or grammatical error is not acceptable. Such mistake is perceived careless of the writer upon the grader, and such mediocre mistake should not be allowed in a formal paper. With this convention, we also had expected the outcome of our model to show a negative correlation of spelling errors compared to the essay scores. It was striking to notice the very few spelling errors in poorly scored essays and the opposite for the highly scored essays. Besides this limitation, there is also a flaw of our model not being able to overturn the incorrectly recognized word to a correctly spelled word by looking at the context of the sentence.

Although we did not focus on analyzing the content of the essay, we were able to predict the type of the writer and the content of the essay. For instance, the correlation between the essay length and sentence length compared to essay scores also interesting. We were able to discover a pattern such that essays with higher scores tended to have longer essays in length, while their sentences tended to be shorter. This could mean that the writer of the essay might be discussing different and introducing new ideas but doing so without being redundant. In addition, we noticed a high correlation in the lexical diversity correlation graph, meaning the writer is not constantly using the same number of words, but is introducing and utilizing different vocabularies to express new ideas. We expected the correlation to be positive to some extent, but did not expect such high numbers.

Looking at the features and the results comprehensively, the boundary of a ”good” essay is drawn more clearly. Using our model, we were able to notice that not only long essays with high frequency of advanced vocabularies were viewed more positively, but essays that had more concise sentences with minimal grammar errors contributed in boosting the scores.

VI. Threats to validity

For the grammar and spelling checking feature, the accuracy and the corpus comparison are questionable. When experimenting our model, we initially used the POS-tagger from NLTK. We inputted

the entire essay into our model and it returned the tagged data back. However, we found out that a word was often tagged differently than when we would have tagged it. Thus, we tried to change the input from the entire essay to the essay split up into multiple sentences, and store those multiple sentences in an array. When we did that, the word that was tagged differently was tagged the way that we thought should have been correct. Although there could have been human bias or error, it shows that the NLTK POS-tagger can be tagged in several ways depending on the parameter and the way the text is inputted.

Also, for the corpus and features of spelling checking errors and sophisticated words, we lemmatized the words to calculate the grammar errors using the built-in functions. However, we do not know how the built-in function works and lemmatizes the words. Consequently, we do not know the accuracy of it, so this might have affected the spelling errors and sophisticated words as well.

VII. Conclusion

Automated essay grading is a vital natural language processing application. It has been studied quite a lot of times, used with different techniques such as latent semantic analysis. This current approach that we presented in this project tried to model the language features such as grammatical correctness, usage of advanced vocabulary, and measurements of length of the essay. The results we received from our model's performance was satisfactory and encouraging. Across all domains we used, the proposed methodology appears to be working the best. The future scope of this particular topic can branch out in many different ways and fields. One such way is to come up with a better approach than using linear regression to calculate. Another way could be looking into the content and comprehension of the essay and choosing those as a factor in constituting a "good" writing.

References

[1] Z. Rider, "Grammar Checking using POS Tagging and Rules Matching.", Computer Science Department, Swarthmore College. 2005.

[2] L. S. Larkey, "Automated Essay Grading Using Text Categorization Techniques," Center for Intelligent Information Retrieval , Amherst.

[3] "The Hewlett Foundation: Automated Essay Scoring," Kaggle, 2012. [Online]. Available: https://www.kaggle.com/c/asap-aes/data. [Accessed: 07-May-2020].

[4] "Word frequency data," Word frequency: based on 450 million word COCA corpus. [Online]. Available: www.wordfrequency.info/sample.asp.

[5] "sklearn.linear_model.LogisticRegression," scikit learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 07-May-2020].

[6] S. Lebowitz, "Here are the top 10 grammar mistakes people make, according to Microsoft," Business Insider, 14-Mar-2017. [Online]. Available: https://www.businessinsider.com/microsoft-data-the-most-common-grammar-mistakes-2017-3. [Accessed: 07-May-2020].

[7] "15 Common Grammar Mistakes That Kill Your Writing Credibility," Authority Self-Publishing, 06-May-2020. [Online]. Available: https://authority.pub/common-grammar-mistakes/. [Accessed: 07-May-2020].

[8] "Part-of-Speech Tutorial," Google Sites. [Online]. Available: https://sites.google.com/site/partofspeechhelp/\#TOC-NN. [Accessed: 07-May-2020].