A Machine Learning Model for Essay Grading via Random Forest Ensembles and Lexical Feature Extraction through Natural Language Processing

Varun N. Shenoy

Cupertino High School

varun.inquiry@gmail.com

Abstract

Written assignments provide a greater degree of assessment on the student's depth of the subject matter and promote deep cognitive analysis. Often, the same essay prompts are recycled year after year. The cultivation of manual data from essays on the same prompt year after year can help teachers create a rapid, autonomous method to grade essays through the fields of statistics and machine learning. This research proposes a novel automated method to grade essays for syntactical correctness through training a machine learning model. The model is trained on a set of 12,975 essays from 8 unique prompts. All of the essay prompts are different, varying from narrative, persuasive, and literary response. Features from the essays were selected and extracted through natural language processing [8]. The final model was able to achieve a quadratic weighted kappa [1] score of 0.96 across all essay sets, a near perfect agreement with the human rater. Therefore, this research represents a near human-level accuracy in essay scoring. To the best of our knowledge, the highest achieved score prior to this work is 0.81 by Tigg *et al.* in [3].

Keywords

Automated Essay Grading, Random Forests, Natural Language Processing

Background

Essay grading algorithms are difficult for teacher's to trust due to the high probability of inaccurate scoring. These are a huge hassle for teachers to deal with and thus, these algorithms do not have the rigor to be used in practical, real world applications.

Introduction

The training data was served as a tab separated value sheet that first needed to be parsed by our algorithm. Each essay consists of 150 to 550 words along with two scores, each in different domains (i.e. language structure, grammar, understanding of the source text). In order to begin training our machine learning classifier, we have to extract features, or defining values, of a sample. Since the essays are nonnumerical in nature, we have to generate numerical features from the text. To optimize our model, we had to extract multiple features from the text, including vocabulary richness, spelling error count, long word count, parts of speech counts, average number of words in a sentence, and sentence count. We hypothesize selecting these features are benchmarks for a writer's language fluency and depth, spelling convention, and vocabulary proficiency. Also, these features were chosen because we believed that they could accurately teach a model that would be able to distinguish between low scoring and high scoring essays with ease.

Materials and Methods

The entire project was built using the Python 3.5 programming language [12] with the following libraries: Sci-kit learn 0.17.1 [10] for machine learning and statistical analysis, Natural Language Tool Kit 3.2.1 [6] for high accuracy natural language processing, and PyEnchant 1.6.7 [11] for spell checking.

The two key methodologies used in this research are machine learning and natural language processing. Machine learning is a field at the crossroads of computer science and statistics where we train mathematical models on data through computer programs. The goal is to get the computer to recognize patterns from features we select from the original data and then be able to classify future feature vectors through the trained model. In short, we want to teach a computer how to solve a problem through statistical probabilities learned from pattern recognition. For the purposes of this research, we trained a random forest classifier [5] to grade essays through a variety of features and data analysis algorithms. Natural language processing is the machine's ability to accurately process and understand text. Part of speech tagging and word stemming (finding the root of a word) was used to aid in the feature extraction process. The first step to creating a successful machine learning model is to define a algorithmic pipeline. In our case, we divided the stages of generating our model into preprocessing, feature extraction, model training, 8-fold cross validation, and final model evaluation as seen in Fig. 1.

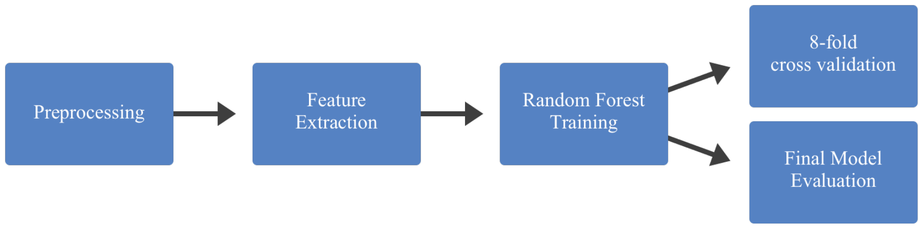

Figure 1: Our machine learning pipeline.

During the preprocessing phase, the training data is loaded from a tab separated value (.tsv) file into a dataframe. Next, features and labels are extracted from each essay. Essays are split into individual words and sentences through tokenization. Features, such as the word count, sentence count, long word count, average number of words in a sentence, and long word count, are computed from the tokens. The aspell dictionary [9] from the PyEnchant framework was used during feature extraction to obtain the count of spelling errors in each essay. If a word's length was greater than seven characters, it was added to the long word count feature. The usage of longer words often indicates a greater depth of language in the student. A method to assess the vocabulary difficulty in an essay is to measure lexical diversity and vocabulary richness. Yule's I characteristic [4] scans through words in an essay and stems each one in order to find the total number of unique words. This number is then normalized and adjusted to prefer non-repeating words to create a quantitative measurement for lexical diversity and vocabulary richness. Part of speech tagging is then used to obtain the number of nouns, proper nouns, and adjectives, in each essay. All of these features for a single essay are then placed in a feature vector, which is part of a larger 2-dimensional vector that contains all of the feature vectors. Each of the individual feature vectors are then mapped to a label in a label vector so the machine learning algorithm knows which features correspond to each score. The scores for the essays in the label vector are directly extracted from the dataset without any manipulation.

The features and labels are then used as input data in our random forest classifier. A random forest classifier creates multiple decision trees, or a forest. Each decision tree fits the input data at random, branching and creating new tree nodes. At each node, a small subset of the feature space is chosen at random and those variables and values are cycled through to optimize a split in the tree. Whenever a new input feature needs to be classified, it is passed down each decision tree in the forest. Whenever a tree reaches a final label on the feature, it "votes" for that label as seen in Fig. 2. The forest chooses the classification having the most votes over all the trees in the forest.

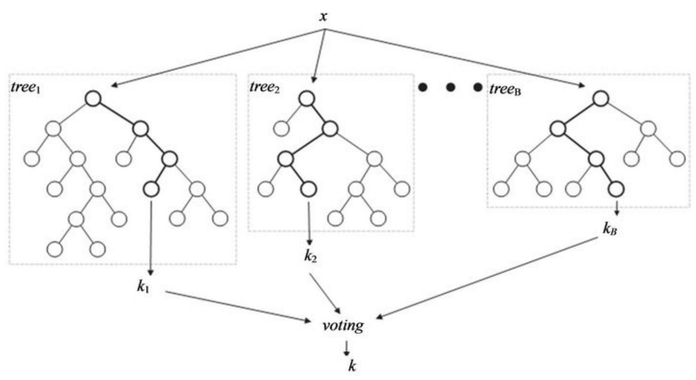

Figure 2: Using random forests in practice for classification tasks.

Decision trees are weak classifiers on their own, but are very powerful and efficient when in forests. Scores tend to increase as the number of trees in the forest increase until a point is reached where the output score is maximized. The error rate of the forest is minimized when the correlation between any two trees is low. Hence, the output score of our forest peaks when the randomness in the forest is maximized. The random forest used in this research was composed of 100 decision trees (see Appendix 1 for a graph showing the optimization of the number of trees in the forest).

$$w_{ij} = \frac{(i-j)^2}{(N-1)^2} \longrightarrow \varkappa = 1 - \frac{\Sigma_{ij}\, w_{ij}\, O_{ij}}{\Sigma_{ij}\, w_{ij}\, E_{ij}}$$

Figure 3: Calculating the quadratic weighted kappa.

The metric which we used to evaluate the random forest was the quadratic weighted kappa (QWK). The calculation process is shown in Fig. 3. A kappa score measures inter-rater agreement between the predicted and actual score, from 0 (pure randomness) to 1 (complete agreement). An N-by-N matrix of weights, w, is calculated based on the difference between raters' scores and then an N-by-N histogram matrix of expected ratings, E, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that E and O have the same sum. The QWK is calculated from these three matrices as seen in Fig. 3. 8-fold cross validation was used with our random forest to increase the accuracy of our kappa score. k-fold cross validation divides the entire dataset into k partitions. It runs an input classifier k times where one partition is used as the training set and k-1 partitions are used as the testing set. In our case, the input classifier was our random forest and k = 8. We took the average of all 8 kappa scores to achieve our final result of a 0.96 quadratic weighted kappa score.

Results and Conclusion

Our final quadratic weighted kappa score is 0.96. Selecting robust features and creating exhaustive decision trees in a random forest have been able to create an machine learning model that rivals human graders nearly all of the time.

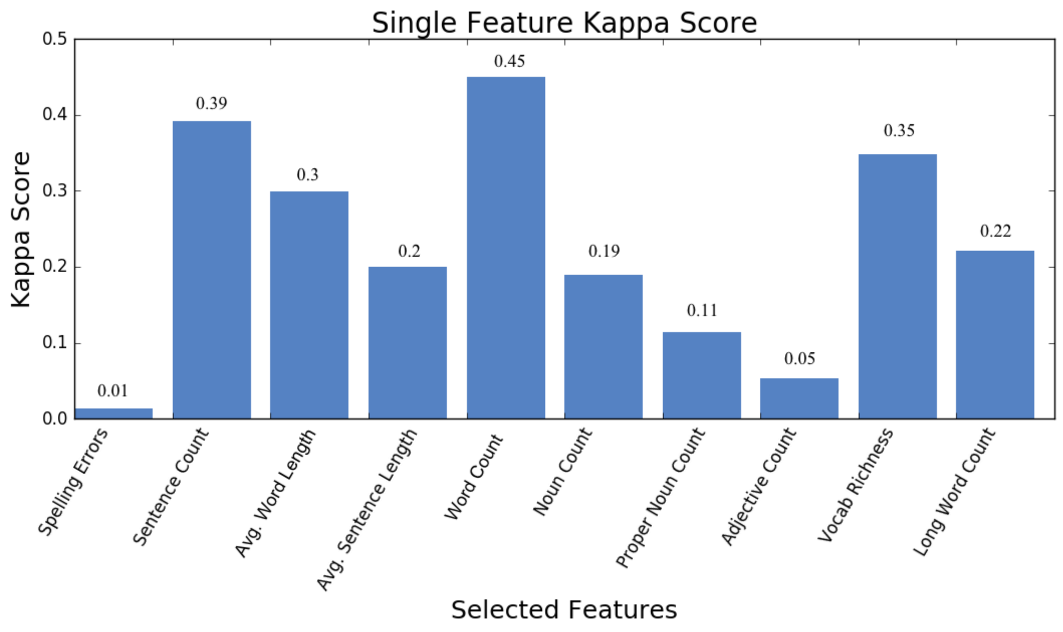

Figure 4: The contribution of each selected feature towards the overall QWK score.

As visualized in Fig. 4, the strongest features used to train our model were word count, sentence count, and vocabulary richness, while parts of speech counts and spelling error counts, contributed the least to optimize our kappa score. Surprisingly, the number of spelling errors in an essay did little to no good in increasing our QWK.

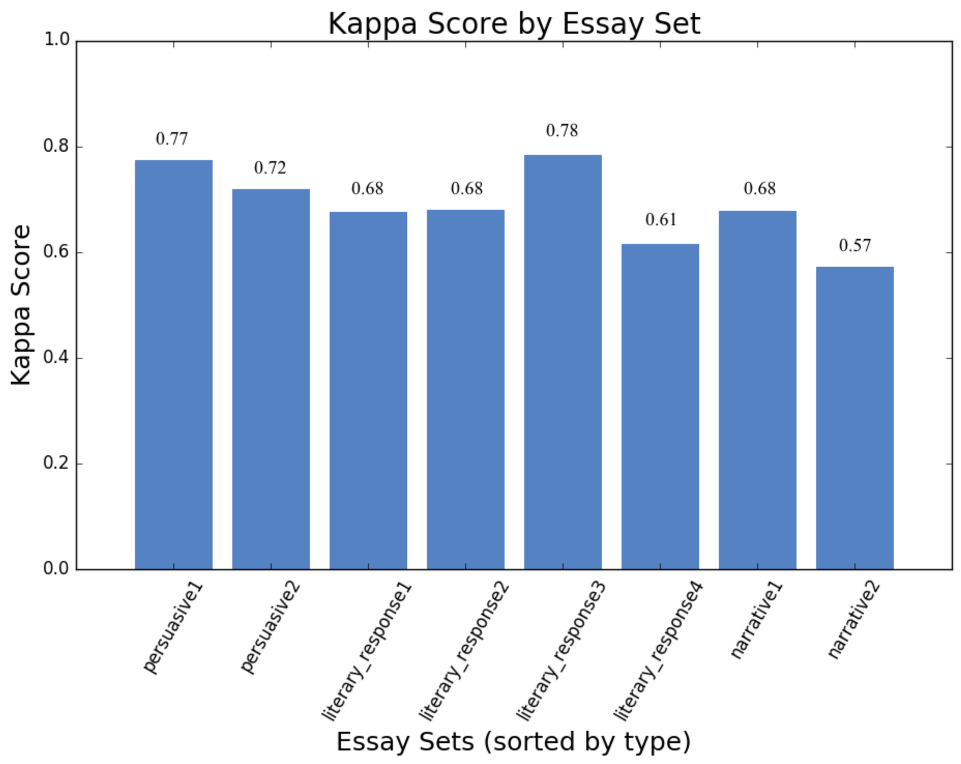

Figure 5: The quadratic weighted kappa score by essay set

Analysis on the data found in Fig. 5 reveals that the average QWK score of the algorithm, by essay set, was about 0.69. This score is evidently lower than our overall score, 0.96. With the whole dataset at hand, our random forest can learn from more varieties of patterns in the essay, rather than essay set by essay set. This caused our score to plummet when individual essay sets were both trained and tested on. Since our features are syntactical rather than based on semantics, the more data there is available to the classification algorithm, the more precise and accurate it will be.

It is clear that our random forest scored well over all 8 essay sets, especially in the persuasive essays. However, it is clear that our algorithm struggled with narrative essay scoring.

This is because narrative essays are more open for the reader, with personal and imaginary stories involved. Excelling in narrative essay scoring would require adding more contextual features, such as sentence structure and proper use of punctuation.

Conclusions

Our machine learning model is near human accuracy in essay grading in eight different prompts, and thus we can help teachers accelerate the grading process through data analysis and machine learning. MOOC (Massive Open Online Courses) websites, like Coursera [7], often lack the resources to grade essays from thousands of students. Through intelligent machine learning algorithms, such as the one proposed in this paper, teachers of MOOCs only have to grade a couple hundred essays as a training set. Although this seems like a lot, it is significantly less than the thousands turned in by students. As the teacher hand-grades more essays, the machine learning model will progressively get smarter.

Acknowledgements

We wish to thank The Hewlett Foundation for providing the dataset used in this research on the data science competition website, kaggle.com [2].

References

1. Cohen J. A coefficient of agreement for nominal scales. Educational and Psychosocial Measurement, 20, 37-46.
2. The Hewlett Foundation: Automated Essay Scoring | Kaggle [Internet]. Kaggle.com. 2016 [cited 31 July 2016]. Available from: https://www.kaggle.com/c/asap-aes

3. Private Leaderboard - The Hewlett Foundation: Automated Essay Scoring | Kaggle [Internet]. Kaggle.com. 2016 [cited 31 July 2016]. Available from: https://www.kaggle.com/c/asap-aes/leaderboard
4. Williams C. Yule's 'Characteristic' and the 'Index of Diversity'. Nature. 1946;157(3989):482-482.
5. Breiman L. Random forests. Machine learning. 2001 Oct 1;45(1):5-32.
6. Bird S, Klein E, Loper E. Natural language processing with Python. " O'Reilly Media, Inc."; 2009 Jun 12.
7. Coursera - Free Online Courses From Top Universities [Internet]. Coursera. 2016 [cited 11 August 2016]. Available from: http://coursera.org
8. Chowdhury GG. Natural language processing. Annual review of information science and technology. 2003 Jan 1;37(1):51-89.
9. GNU Aspell [Internet]. Aspell.net. 2016 [cited 10 August 2016]. Available from: http://aspell.net/
10. Pedregosa F. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2016;(12).
11. Kelly R. PyEnchant [Internet]. Pythonhosted.org. 2016 [cited 2 August 2016]. Available from: https://pythonhosted.org/pyenchant/
12. Python Software Foundation. Python Language Reference, version 3.5. Available from: http://www.python.org

Appendix

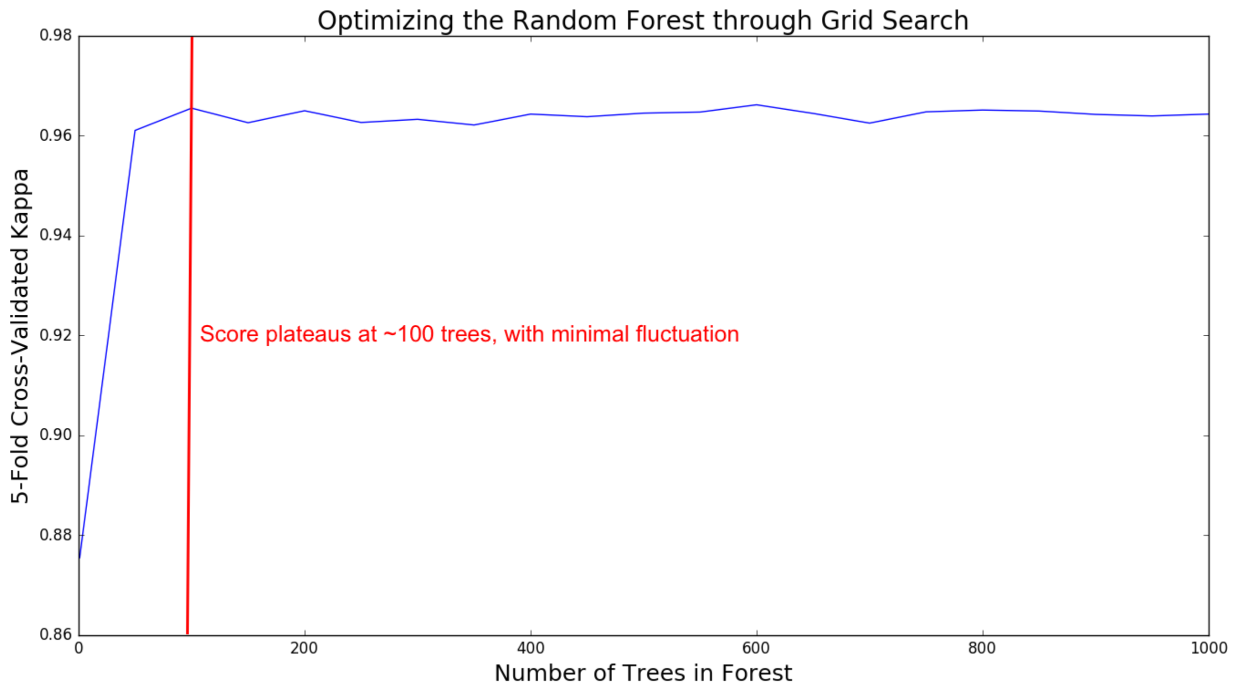

Figure A1: Quadratic weighted kappa score is maximized at 100 decision trees.