AutoGrade

PROJECT REPORT

INFORMATION RETRIEVAL

2017

Ayushree Gangal

Deva Surya Vivek Madala

Shreyash Krishna

Acknowledgements

We would like to take this opportunity to express our gratitude towards Professor Ashish Sureka and Teaching Fellow Anjali Goyal for their constant guidance and support. We believe that this project would not have been possible without them and their valuable contributions.
We would also like to thank Professor Jitendra Khanna for his feedback and insights on the process of grading a piece of writing from an editor’s perspective.

Table of Contents

Acknowledgements **2**

1. Introduction **4**

2. Data Set **4**

3. Features **4**

Word Count Ratio 4

Sentence Length 4

Voice of the Essay 5

Tense of the Essay 5

Spell Check 5

Grammatical Errors 6

Vocabulary 6

Semantic Similarity 6

4. Feature Selection **7**

5. Machine Learning Algorithms **9**

k-Nearest Neighbours (kNN) 9

Sequential Minimal Optimizer (SVM) 9

Linear Regression 10

6. Evaluation and Results **10**

7. References **12**

1. Introduction

AutoGrade applies principles of Machine Learning and Information Retrieval to solve the increasingly time consuming problem of grading papers. Grading multiple papers on the same topic involves huge redundancies in terms of evaluating language, format and structure of the papers. In this context, AutoGrade presents itself as a potential solution by evaluating the aforementioned aspects faster and more objectively by automating the grading process. Moreover, its potential applications range from grading essays in standardised tests such as SAT, GRE and TOEFL to being employed by writers as a language, format and structure check before any submissions.

2. Data Set

The training set and test data were collected from *Kaggle*[1]. Both sets of data consist of 150-500 word essays written by students across grades 7 through 10. Out of the 8 available sets, the software trained on essays from set 1, which consisted of approximately 2000 essays. Approximately 600 essays were used for testing the trained models (retrieved from *Kaggle* itself).

3. Features

AutoGrade uses Python to build the following features as part of the preprocessing phase. After building these features, the software uses *Weka*[2] to build classifiers based on scores calculated for each feature.

1. Word Count Ratio

   Description: This feature calculates the ratio of word count of the essay to the specified word limit. It keeps track of how far an essay is from the specified word limit in terms of there being either too many or too few words. This feature assumes same weightage for equal number of words above or below the word limit. This feature uses *textstat* to tokenise and count the number of words in the document.
   Score: The score for this feature is calculated as : 1-WC/WL. Subtracting the ratio from 1 is a way of normalising the score (equivalent to taking absolute value of the ratio) and realising the assumption made above.

2. Sentence Length

Description: Longer sentences tend to be more convoluted, less effective and less coherent owing to their verbosity and may negatively impact the final grade. This feature keeps track of such long sentences. This feature makes use of the word count feature discussed above as well as *nltk*[3] to tokenise the text into sentences and count the number of sentences.

Score: The score for this feature is calculated by dividing the number of sentences with 15 or more words[4] by the total number of sentences in the essay. A large ratio implies that an average sentence of the essay is long.

3. Voice of the Essay

Description: It is recommended that any piece of writing be in active voice rather than passive voice for a better impact[5]. This feature evaluates whether, on an average, a sentence in the essay has been written in active or passive voice. This feature uses S*pacy*[6] to identify the voice of a sentence by analysing the structure of the sentence: in active voice, the subject performs the active verb's action whereas in passive voice, the subject gets acted upon by the verb's action (which is no longer active)[7].

Score: The score for this feature is calculated by dividing the number of sentences written in active voice by the total number of sentences. A large ratio suggests that an average sentence in the essay is written in active voice, which may impact the final grade in a positive manner.

4. Tense of the Essay

Description: A good piece of writing should be written in the same tense[8] (regardless of the choice of tense). Mixing different tenses may result in a negative impact on the final grade. This feature uses *nltk* to identify different parts of speech (verbs, nouns, adjectives, etc.) and focusses mainly on verbs.

Score: The score for this feature is calculated by first finding out what the dominant tense verb in the essay is and then dividing the number of such verbs by the total number of verbs. A large ratio implies that there is one dominant tense in the essay which may have a positive impact on the final grade.

5. Spell Check

Description: A good piece of writing minimises the number of spelling errors[9]. This feature first tokenises the text into words and then uses *enchant's spell checker*[10] to look up the spelling of these words and returns a count of the number of spelling errors occurring in the document.
Score: The score for this feature is calculated by dividing the number of spelling errors by the total number of words in the document. A large ratio suggests a high number of spelling mistakes, which may have a negative impact on the final grade.

6. Grammatical Errors

Description: An impactful piece of writing is usually a function of negligible to no grammatical errors[11]. This feature keeps track of the proportion of grammatical errors in the document by using *language check*[12]. For each sentence, *language check* checks whether the sentence follows certain grammatical rules or not.
Score: The score for this feature is calculated by dividing the number of grammatical errors by the total number of words in the document. A large ratio implies a large number of grammatical errors, which may impact the final grade in a negative manner.

7. Vocabulary

Description: A good piece of writing involves using sophisticated vocabulary[13]. This feature employs a bag of words model: it uses *nltk* to first tokenise the text into words and then removes all the stop words and returns the count of unique words.
Score: The score for this feature is calculated by dividing the number of unique words by the word limit. The rationale behind using word limit (rather than word count) is to take care of cases where the ratio may be high owing to the fact that the essay had very few words to begin with, which shouldn't be the case. Thus, a large ratio implies good use of vocabulary only in cases where the essay is of a sufficient length, which may impact the final grade in a positive manner.

8. Semantic Similarity

A good piece of writing should be coherent as well as relevant to the given topic[14]. The concept of semantic similarity is being used in two features to judge both relevance of the essay to the topic and the coherence of the essay itself. It is being calculated by using *WordNet*[15] and *nltk.* The text is first tokenised into sentences and for each pair of sentences, their semantic similarity is computed as follows:

Step 1: Word pairs are formed: (i,j) such that i belongs to the first sentence and j belongs to the second sentence. The root of both words are compared and a semantic score is assigned to the pair. This process is repeated for all possible pairs for the two sentences.

Step 2: Out of all such scores, the highest score is taken to be the semantic similarity of the two sentences. This process is repeated for all pairs of sentences in the document.

Step 3: The semantic similarity score of the entire piece of text (either paragraphs or the document as a whole) is computed by taking the average of the semantic similarity scores assigned to each pair of sentences.

Step 4: The average score obtained in the previous step is then multiplied by the $\log_2$ of the number of sentences. This normalisation is done to ensure that essays with very few sentences (and thus far away from the specified word count) do not receive high scores.

8.1 **Semantic Similarity of the essay with the topic**: this feature evaluates the relevance of the essay to the given topic by computing the semantic similarity of each sentence from the topic and each sentence from the essay. A high score implies that the essay is fairly relevant to the topic.

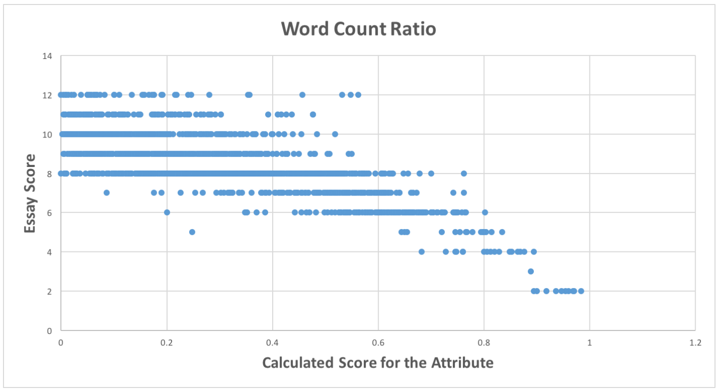
8.2 **Semantic Similarity of the essay**: this feature evaluated the coherence of the essay itself by computing the semantic similarity of each sentence of the essay to every other sentence of the essay. A high score implies that the essay is fairly coherent.

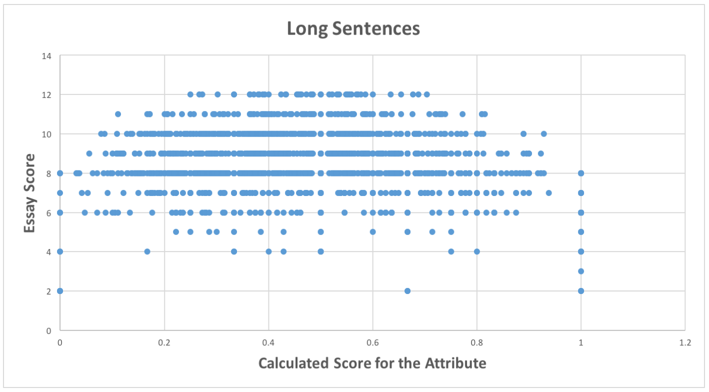
4. Feature Selection

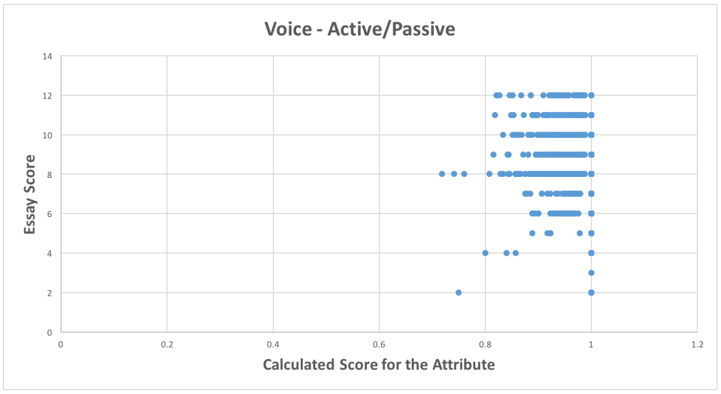
Feature selection aids in improving the accuracy of the classifiers as well as reduces the chances of overfitting by removing irrelevant data. It also results in a decrease in the training time as irrelevant features are removed. AutoGrade uses the Relief F-Attribute Eval (Relief algorithm) to carry out feature selection and has had to remove two attributes : Grammatical Errors and

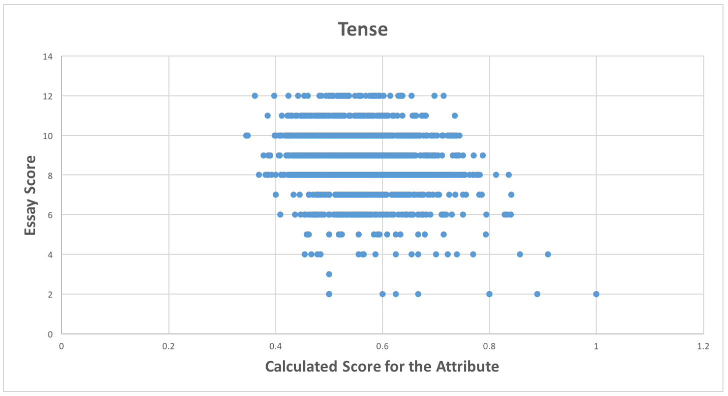
Sentence Length as they were ranked the lowest and removing them resulted in an increase in accuracy of the classifier.

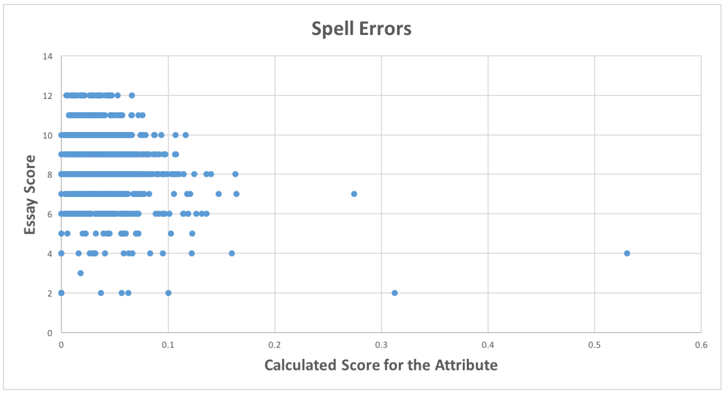
The following charts were generated during the training phase. Each chart displays the correlation of each of the nine attributes (independent variable) with the score (dependent variable).

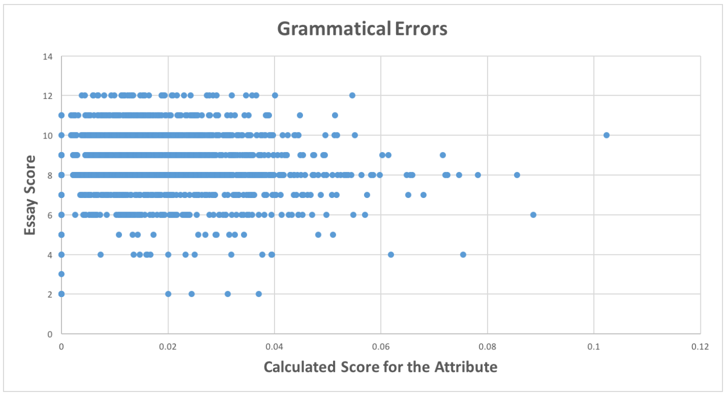


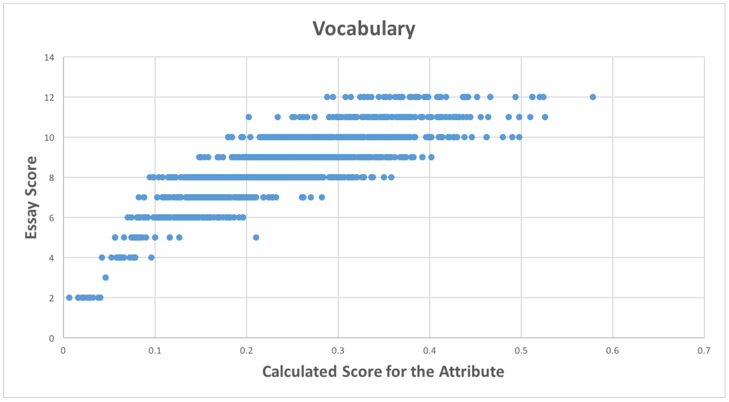


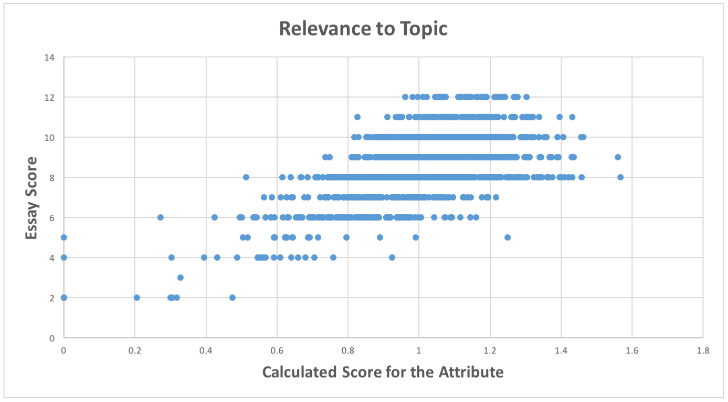


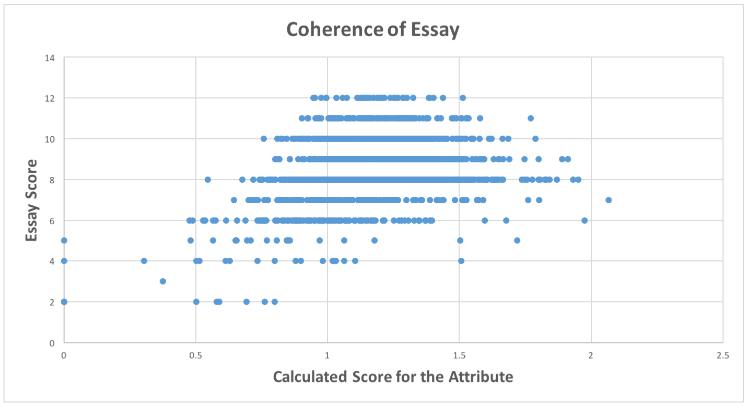












5. Machine Learning Algorithms

AutoGrade built its classifiers in *Weka* by using three machine learning algorithms:

1. k-Nearest Neighbours (kNN)

This classifier predicts class of the training vector using Euclidean distance between vectors. In the equation below R_i represents the region for i th class, and $\vec{x}$ is our training vector. Vector x is said to be nearest to class $'i'$ if-

$$R_i = \{x : d(x, x_i) \; < d(x, x_j), i \neq j \;\}$$

where j is any other class.

This classifier works in an n-dimensional space, where each axis represents a feature and each data point has its ith coordinate as the value of the ith feature. Before classifying a datapoint in a particular class, the classifier analyses its k- nearest neighbours and classifies the data point accordingly.

2. Sequential Minimal Optimizer (SVM)[16]

This is a non-probabilistic classifier that works in an n-dimensional space, where n represents the number of features (in our case, n = 9 before feature selection and 7 after feature selection). It plots each data item in this n-dimensional space with its ith coordinate being the value of the ith feature for the data point. It then uses hyperplanes to distinguish between various classes and consequently, classify data points accordingly. The maximum margin hyperplane is calculated as-

$$w^T x_i + w_0 \geq 0 \quad \forall i : y_i = +1$$
$$w^T x_i + w_0 \leq 0 \quad \forall i : y_i = -1$$

In this case, y=+1 is the first class and -1 is the second. The Sequential Minimal Optimizer normalizes each of the attributes and consequently the final score is calculated.

3. Linear Regression[17]

This is a statistical technique that constructs a linear equation with the appropriate coefficients and values of the independent variables, and the solution to this equation is the predicted value of the dependent variable (in our case, the grade). The coefficients are estimated statistically during the training phase and once these coefficients have been calculated, different values of the independent variables (features) produce corresponding values of the dependent variable (grade). This classifier assumes a linear relationship between the independent and dependent variables. Mathematically,

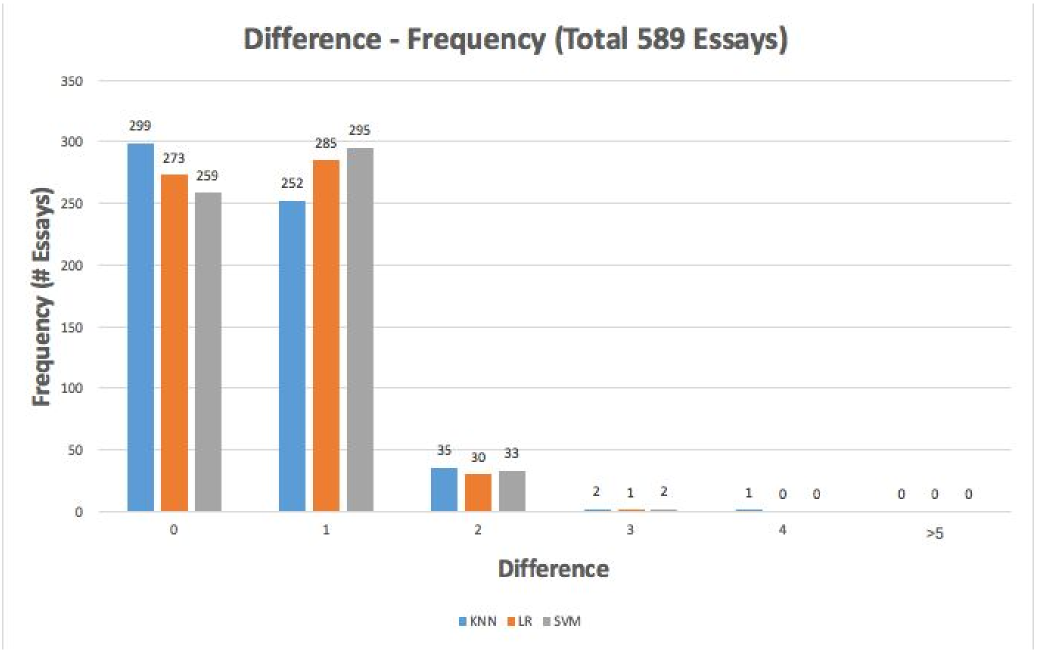
$$h(x) = \sum_{i=0}^{9} \theta_i x_i$$

The parameters θ_i are chosen to minimize the least-squares cost function.

6. Evaluation and Results

The accuracy, precision, recall and F-measure of the three classifiers were computed based on results obtained from a test data set containing 589 essays (written along the training data set format). For each classifier, the difference of the predicted score and the actual score was calculated and the frequency of each difference value was tabulated against the number of essays for that value. The following table demonstrates the number of essays classified in a particular class that varies from the actual class by the given difference. For example, for the kNN classifier, 299 essays had the same predicted and actual scores while 252 essays differed from the actual score by 1.

Frequency	KNN	LR	SVM
0	299	273	259
1	252	285	295
2	35	30	33
3	2	1	2
4	1	0	0
>5	0	0	0



Accuracy:

For computing the accuracy of the classifiers, the correctly classified instances included all those instances whose score differed from the actual score by 1.

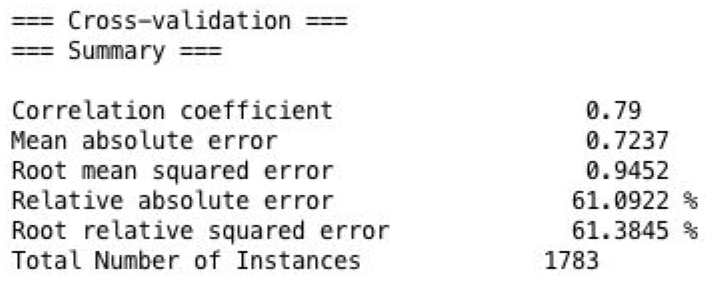
- kNN: $((299 + 252) / 589) \times 100 = 93.5\%$
- Linear Regression: $((273 + 285) / 589) \times 100 = 94.7\%$
- SMO (SVM): $((259 + 295) / 589) \times 100 = 94.1\%$

Weighted Average Metric → **Classifier ↓**	Precision	Recall	F-Measure
kNN	0.51	0.15	0.21
Linear Regression	0.51	0.12	0.18
SMO (SVM)	0.47	0.12	0.18

The precision, recall, f-measure values are low in absolute terms as the data in this case consists of multiple classes with continuous (numeric) data points. We compute the metrics for a score vs all other scores by reducing it to a binary classification: 'score x' vs 'not score x'.
As the # of essays that belong to 'score x' << # of essays that belong to ' not score x', the metrics are low in absolute terms.

Other metrics (done using *Weka*):

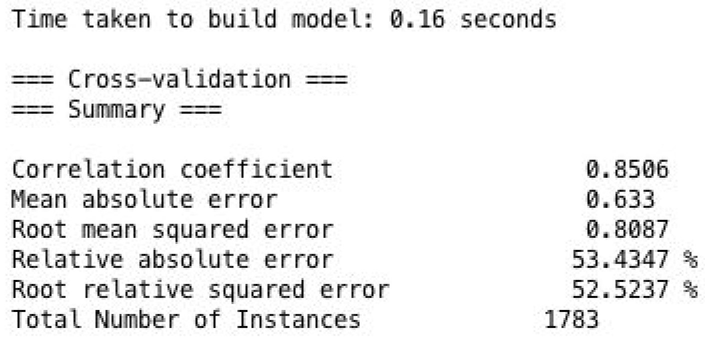
- kNN:



```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.79
Mean absolute error                      0.7237
Root mean squared error                  0.9452
Relative absolute error                 61.0922 %
Root relative squared error             61.3845 %
Total Number of Instances             1783
```

- Linear Regression:

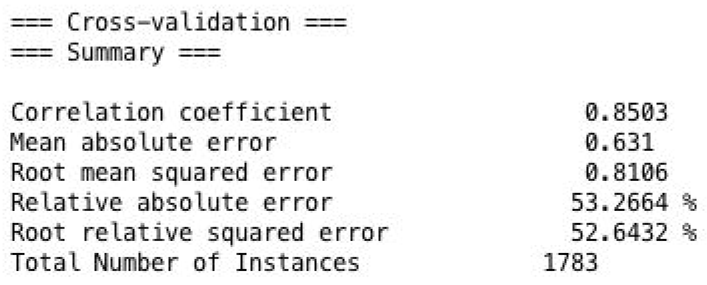


```
Time taken to build model: 0.16 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8506
Mean absolute error                      0.633
Root mean squared error                  0.8087
Relative absolute error                 53.4347 %
Root relative squared error             52.5237 %
Total Number of Instances             1783
```

- SMO (SVM):



```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8503
Mean absolute error                      0.631
Root mean squared error                  0.8106
Relative absolute error                 53.2664 %
Root relative squared error             52.6432 %
Total Number of Instances             1783
```

References

Valenti, Salvatore, et al. “An Overview of Current Research on Automated Essay Grading.” *Journal of Information Technology Education* , vol. 2, 2003, jite.org/documents/Vol2/ v2p319-330-30.pdf.

Song, Shihui, et al. “Automated Essay Scoring Using Machine Learning.” http://cs229.stanford.edu/proj2013/SongZhao-AutomatedEssayScoring.pdf

Sundaram, Arun. “A Comparison of Machine Learning Techniques on Automated Essay Grading and Sentiment Analysis.” *Ohio State University*, 2015, etd.ohiolink.edu/!etd.send_file?accession=osu1428460405&disposition=inline/

Mihalcea, Rada. “Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity.” *The Association for the Advancement of Artificial Intelligence*, www.aaai.org/Papers/AAAI/2006/AAAI06-123.pdf

“Welcome to the Purdue OWL.” *Purdue OWL: Essay Writing*, owl.english.purdue.edu/owl/resource/685/01/.

Kaggle.com. The hewlett foundation: Automated essay scoring - evaluation, February 2012. http://www.kaggle.com/c/asap-aes/details/evaluation.

Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

“Natural Language Toolkit¶.” *Natural Language Toolkit — NLTK 3.2.5 Documentation*, www.nltk.org/.

“SpaCy - Industrial-Strength Natural Language Processing in Python.” *SpaCy - Industrial-Strength Natural Language Processing in Python*, spacy.io/.

Enchant, abiword.github.io/enchant/.

WordNet Interface, www.nltk.org/howto/wordnet.html.

"Compute Sentence Similarity Using Wordnet - NLP FOR HACKERS." *NLP-FOR-HACKERS*, 8 Aug. 2017, nlpforhackers.io/wordnet-sentence-similarity/.

Ray, Sunil, et al. "Understanding Support Vector Machine Algorithm from Examples (along with Code)" *Analytics Vidhya*, 13 Sept. 2017, www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ .

"Linear Regression for Machine Learning." *Machine Learning Mastery*, 21 Sept. 2016, machinelearningmastery.com/linear-regression-for-machine-learning/.

Relief: http://www.win.tue.nl/~mpechen/publications/pubs/ZafraISDA2010.pdf