# Machine Learning Engineer Nanodegree
# Capstone Proposal

Vasu Mistry

May 24, 2017

## 1 Domain Background

Automated essay scoring (AES) has been in the research area of computer science since the early 1966 [1]. Predicting the score of an essay so that the score might seem like it has come from a human reader is a bit daunting task because there are numerous quantified features that have to be extracted from the essay as well as many unquantifiable properties like the perceptions of the writer while writing the essay and his thoughts that he is trying to inscribe on the paper. Therefore, the behavior of the essay inherently noisy, non-stationary and deterministically chaotic. The quantifiable data that can be extracted from an essay is relatively easy for the computer to process rather than processing the ideas or thoughts of the writer in the essay, which may or may not affect the scoring of an essay by a computer.

### 1.1 Motivation

Personally, I came across this problem during my internship at a start-up. For recruiting content-writers for their website, during the interview phases, a candidate must write a 250 word long essay to demonstrate his skills. At present, this grading is being done manually by the start-up. My goal is to make grading of these essays automatically and I plan to begin my approach through this Capstone Project.

## 2 Problem Statement

The Hewlett Foundation sponsored the Automated Student Assessment Prize on Kaggle - AES Challenge, challenging teams to produce essay evaluation models that best approximate human graders. Many competitive exams try to include maximum number of multiple choice questions since written essays are being graded manually and take up a lot of time to evaluate. While it is a known fact that written essays provide opportunities to challenge the students with more sophisticated measures of ability, the ease of bubbled-answers checking promises a faster evaluation of the student.

The project aims to build a regression model that can take in an essay and automatically output the grade of that essay. Using feature-extraction and Machine Learning algorithms, this task can be automated. The output value will be a continuous value between 2-12.

# 3 Datasets and Inputs

## 3.1 Datasets

The dataset was provided as a part of the Kaggle - AES Challenge [1]. For this competition, there are eight essay sets with an average length of 150-550 words. The students writing these essays are from Grade 7 to Grade 10. Each essay was graded by 2 graders and each essay has its own unique characteristics.

For the purpose of this project, I will use Essay set 1 from the data-set to train and test my model.

| Essay Set | Essay Type | Domain | Score Range | Average Length of words | Total |
|---|---|---|---|---|---|
| 1 | Persuassive/Narrative/Expository | Letter writing | 2-12 | 350 | 1783 |

Table 1: Data-set characteristic

Essays had been anonymized before being released to the public using the Named Entity Recognizer (NER) developed by the Stanford Natural Language Processing group [2]. Replacement IDs of the @ sign followed by words in all capitals were used instead. Name Entities of People, Organizations, Locations, Times/Dates, Numbers, Percents, E-mail Addresses, and Money were replaced.

## 3.2 Inputs

Essay set 1 contains:

- essay_id: A unique identifier for each individual student essay
- essay: The ascii text of a student's response
- rater1_domain1: Rater 1's domain 1 score (Range: 1-6)
- rater2_domain1: Rater 2's domain 1 score (Range: 1-6)
- domain1_score: Sum of rater1 and rater2's scores. (Range: 2-12)

# 4 Solution Statement

The input does not contain any features, so appropriate features will have to be extracted. An overview of related prior work [3][4][5] indicates that linear regression works well for essay grading applications, hence I will use Linear Regression for the learning model. Since, no validation test essay set was provided in the contest, a 5-fold cross validation will be done to train and test the learning model. This will help to guard against overfitting[3]

[1] https://www.kaggle.com/c/asap-aesm

# 5 Benchmark Model

The automated reader developed by the Educational Testing Service, e-Rater, used hundreds of manually defined features. It was trained on 64 different prompts and more than 25,000 essays. Evaluated on the quadratic weighted kappa calculated between the automated scores for the essays and the resolved score for human raters on each set of essays, e-rater could only achieve the kappa score below 0.5 [6]. Hence, I would desire to develop a learning model that has at least a kappa score of 0.5.

# 6 Evaluation Metrics

The quadratic weighted kappa score is a measure of agreement of our scores and the human annotator's gold-standard. 0 represents only random agreement between the raters and 1 is full agreement. For N possible essay ratings, an N X N matrix O is constructed where $O_{i,j}$ represents the number of essays receiving grade i from the first grader and j from the second rater. Additionally, a matrix E is constructed the same way, but assuming there is no correlation. The matrices are normalized so that they have the same sum. An N X N matrix w is also calculated where:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{1}$$

The quadratic weighted kappa is calculated by:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \tag{2}$$

The Quadratic Weighted Kappa metric typically varies from 0 - only random agreement between raters - to 1 (complete agreement between raters). In the event that there is less agreement between the raters than expected by chance, this metric may go below 0.

# 7 Project Design

## 7.1 Programming language and Libraries

- Python 2.7
- Scikit-Learn Open source machine learning library for Python
- NLTK (Natural Language Processing Toolkit) Open source Natural Language Processing Library [2]
- Grammar-Check Open source library for checking grammatical and spelling mistakes
- NumPy and Pandas - A fundamental package for scientific computing in Python. They provide advanced data structures which are highly useful for data analysis.
- Textmining A Python library for performing statistical text mining.

[2] Prerequisite: Java

## 7.2 Methodology

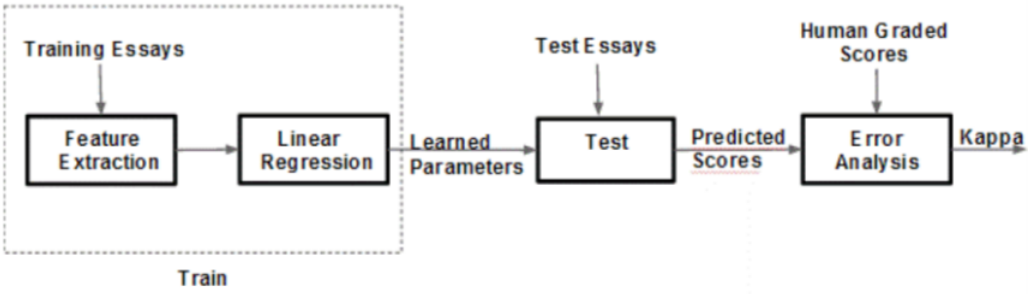


Figure 1: Implementation Methodology.

We hypothesize that a good prediction for an essay score would involve a range of feature types such as language fluency and dexterity, diction and vocabulary, structure and organization, orthography and content. A good model would incorporate features from each of these areas to arrive at a good prediction [3].

## 7.3 Step Wise approach

1. **Preprocessing:**
   The data was already preprocessed by the organisation using the Stanford NLP tools [3]. The named entities were marked with *""@Text"*. Necessary steps will be taken to ignore these tagged words during the feature extraction (especially spelling mistake checking) stage.

2. **Feature extraction:**

   - Heuristic features
     Several heuristic features that are likely to contribute to a good essay will be generated. Some of the heuristic features are: word count, long word count, average word length per essay, quotation mark count etc.
   - Spelling and Grammatical features
     It is likely for a student to make grammatical and spelling errors, using the open source libraries mentioned in section 7.1 these features will be generated.
   - Part of Speech (POS) tags
     A **count** for most regular POS tags will be used, for example, count of Proper Noun count, Adjective count, Adverb count etc.
   - Other features
     Several other features like domain words (number of words that relevant to the domain of the essay), punctuation count, word to sentence ratio etc will also be generated.

3. **Training and Cross Validation:**
   The features extracted will be fed to a Linear Regression training model and a 5 fold cross validation strategy will be used to guard against overfitting.

[3] https://nlp.stanford.edu/software/

4. **Forward feature selection:**
   It is highly unlikely that all of the features generated during the feature extraction will contribute towards grading the essay. To eliminate the poor performing features, a greedy forward feature [4] [5] algorithm will be used.

5. **Final training and testing:**
   After obtaining a list of selected features from the above step, the learning model will be trained and tested. The predicted values will be checked with the evaluation metrics to verify if the learning model has performed better than the benchmark set or not.

# References


[1] Md. Haider Ali Annajiat Alim Rasel Arshad Arafat, Mohammed Raihanuzzaman. *Automated Essay Grading with Recommendation*. 04 2016.

[2] Trond Grenager Jenny Rose Finkel and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. *Proc. of ACL*, pages 363–370, 2005.

[3] Ashwin Apte Manvi Mahana, Mishel Johns. *Automated Essay Grading Using Machine Learning*. 12 2012.

[4] Neri F. Cucchiarelli A. Valenti, S. *An Overview of Current Research on Automated Essay Grading,*. 2003.

[5] Burstein J. Attali, Y. Automated essay scoring with e-rater® v.2. *The Journal of Technology, Learning, and Assessment*, 2006.

[6] et al. Foltz, PeterW. *Implementation and applications of the Intelligent Essay Assessor*. Routledge Handbooks, 2013.


[4] https://en.wikipedia.org/wiki/Feature$_s$*election*
[5] https://www.cs.cmu.edu/ kdeng/thesis/feature.pdf