A Two-Fold Exploratory Study on AES

Abhinandan Dubey
Dept. of Computer Science
Stony Brook University
New York, USA

Abstract

Automated Essay Scoring systems are being widely used in the industry - The ETS uses AES to grade the AWA section of the new GRE, and same applies to the GMAT. The present research has been limited to rely heavily on extracting carefully designed features to evaluate and score essays through training on huge datasets. This makes it impossible for primary school teachers to use such systems for grading. Moreover, until recently, even the systems which involve training on huge datasets yielded average results. We present a supervised system which improves the state-of-the-art system, and another, unsupervised system which perform as good as many supervised systems.

Introduction

In terms of current research, Automated Essay Scoring is perhaps one of the areas which have been over-looked. There hasn't been much of active research going on concerning essay scoring systems. While the critics argue that AES cannot possibly grade essays in a meaningful manner, however, they often fail to realize that while we, humans, can identify good pieces of writing, we often find it difficult to quantify or even articulate why a piece of writing holds value. Automated Essay Scoring systems fill this gap by establishing a common, quantifiable basis for grading essays. The development of automated essay scoring systems is a common ground for test developers like ETS, cognitive scientists, and computer scientists. Previously, giants like ETS have worked extensively in AES and related areas, having the benefit of the huge datasets and models which get trained over every single applicant that registers for the test.

(Shermis and Burstein 2002) provides an excellent description of how can automated essay scoring be implemented as evaluation supplement in classrooms, the latest advances in the field and how reliable they are.

The main contributions of this work are:

1. An unsupervised essay scoring system that runs on extensive set of highly engineered and hand-crafted features to find out specific cues to grade essays in the cases where sample size could be as low as 5 graded essays.
2. A state-of-the-art system using Neural Networks for grading essays - this involves training on conventional datasets.

Subproblems Our work involved solving several subproblems and going through a lot of literature about understanding what features really impact essay scores.

Approach Description

Our study involves two folds. For the first fold we have designed and implemented **UNTraSE**, an Unsupervised, Trait-Specific Essay Scoring System, which works on a trained calibrator and a set of natural-language analysis modules which capture various salient features of essays. This system only requires as low as 5 *"best-quality"* essays. We were initially looking into methods proposed in (Forman and Cohen 2004), which allow to built efficient systems learning from little data. But as it turns out, even these approaches require a considerable amount of data which defeats its very purpose of having an unsupervised system. The system works out by scoring the essay on different trait-specific features which include syntactic, lexical and semantic cohesive devices. For second fold, we have designed and implemented **DeepScore**, a neural-network based essay scoring system. The model has been trained and tested (on a separate set) over *"The Hewlett Foundation: Automated Essay Scoring"* dataset on Kaggle by The Hewlett Foundation which consists of around 1785 essays on 8 different topics, human-graded in a score range of 0-6.

We have also built a simple, linear regression based model which works on simple statistical features such as noun count, verb count, sentence count, word count etc, as suggested in (Manvi Mahana 2012). We present the results of both the systems in the results section and also provide a thorough analysis of the optimizations we did in the course, and how this study largely differs from the previous work in AES.

Implementation & Evaluation

Both of the models have been implemented in Python 2.7.13, and require machine-learning based libraries. The implementation will be done in python. We will use NLTK, SciPy, NumPy and sklearn libraries. Other than that, we will be using Keras on top of TensorFlow.

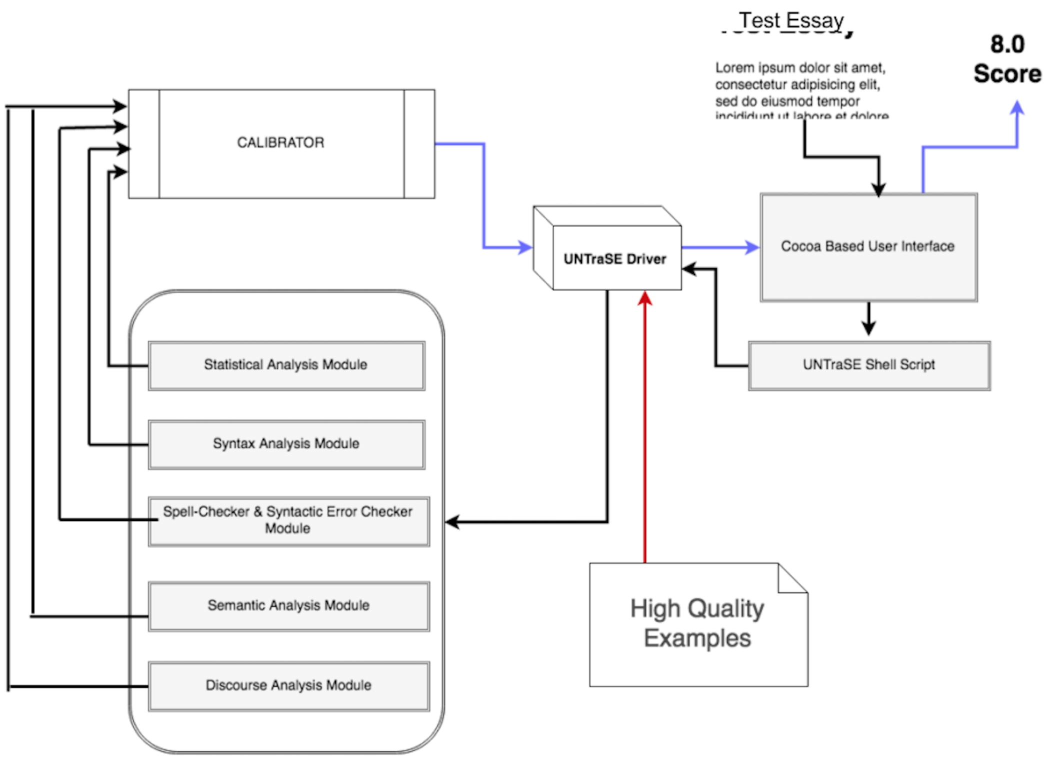
Test Essay
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore
8.0 Score
CALIBRATOR
UNTraSE Driver
Cocoa Based User Interface
UNTraSE Shell Script
Statistical Analysis Module
Syntax Analysis Module
Spell-Checker & Syntactic Error Checker Module
Semantic Analysis Module
Discourse Analysis Module
High Quality Examples

Figure 1: UNTraSE - Architecture

We will evaluate our model using Quadratic Weighted Kappa (Cohen 1960) which measures inter-rater agreement for qualitative (categorical) items on test set. The equation for weighted κ is:

$$\kappa = 1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}x_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}m_{ij}}$$

where k=number of codes and w_{ij}, x_{ij}, and m_{ij} are elements in the weight, observed, and expected matrices, respectively.

UNTraSE - Unsupervised Trait-Specific Essay Scoring

UNTraSE runs on a modular architecture. There are 5 modules which capture diverse cohesive devices of the text and quantify it based on hardcoded metrics. The driver program aggregates all the output scores from the five modules and sends them to a **calibrator**, which has been trained over a regression model. The calibrator takes in all the scores from the modules and outputs a final score weighted appropriately. The calibrator was also trained over the dataset of over 1783 essays to ensure that it generalizes well. A User Interface has also been developed, especially for purposes of interactivity. The User Interface takes in the essay and outputs the final score. The UI is a standalone application written in Objective-C over *Cocoa Framework* in the XCode Environment, and thus supports all OS X operating systems. We plan to dockerize the UNTraSE Engine to skip the setup issues. Now, we will discuss different modules which form the building blocks of the system,

Statistical Analysis Module The module extracts statistical features from the essay including word count, long-word count (words with 8 or more letters), average number of words in a sentence, and the sentence count. The same features are extracted from all the "perfect essays" provided to the system. The metrics are then normalized, and passed on for calculating variance with the test essay. The variance is calculated and a score is assigned based on the variance of different metrics.

Syntax Analysis Module Ten different Part of Speech counts are computed for the test essay and each "perfect-essay" individually. The Part-of-Speech include: Nouns, Verbs, Adjectives, Pronouns, Modal Auxiliary (*"can", "cannot", "could", "couldn't"*), Prepositions, or Conjunction, sub-ordinating conjunctions, Adverbs, Determiners, Coordinating Conjunctions (*"neither", "therefore", "versus", "whether"*). These counts are then fed into a scoring function which normalizes them according to the word count for each essay. Again, the test essay is compared against the perfect essays and a score is computed and sent to the driver program.

Spell Checker & Syntactic Error Checker This module is in a passing build mode, and the ability to identify

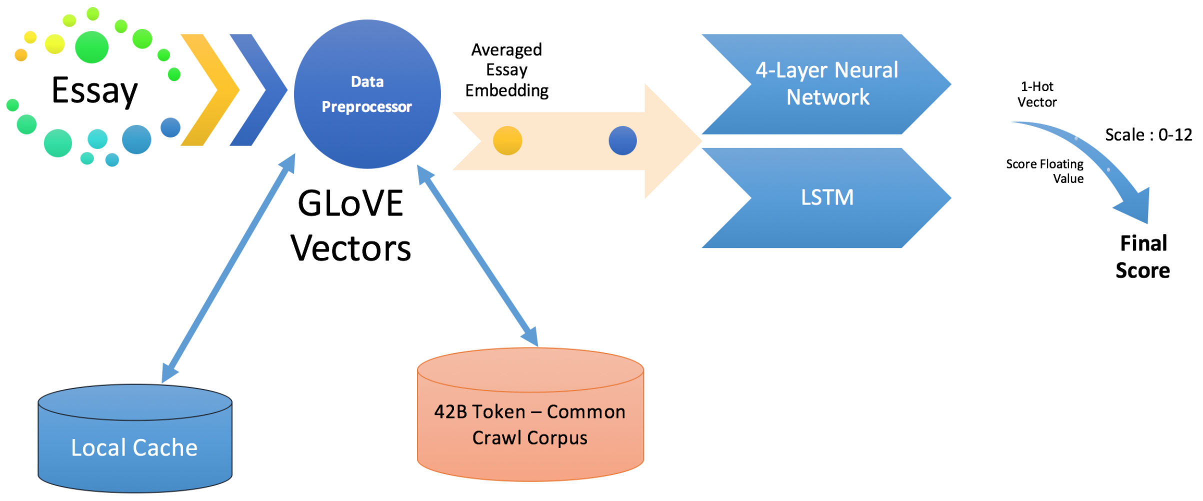

Figure 2: DeepScore - WorkFlow

new types of errors are being continually added, in addition to improving the ones already implemented. Currently, the module can identify any miss-spelled words and a good variety of syntactic errors. The syntactic error checker works using a **Context-Free Grammar** using a recursive approach over the syntax tree. The syntactic errors have been categorized into different "types". *type-1 errors* include the errors where the sentence formation is incorrect or the sentence is incomplete. The same is detected using a recursive approach using Penn TreeBank and a handcrafted function that traverses over the tress. *type-2 errors* include miss-spelled words that come within some edit-distance such as *"can't'* vs *"cant"*.

Semantic Analysis Module The module performs Latent Semantic Analysis over test essay and the set of "perfect-essays". Using a cosine spatial distance, it then assigns how the given test essay is apart from each of the "perfect-essays". The maximum value of cosine similarity is used to compute the score for this module. Thus, it would be in the best interests of the user to specify as diverse essays as possible in the set of "perfect-essays".

Discourse Analysis Module This module extracts key-ideas from the essays. This is done using a context-free grammar looking for entities, key elements in Noun Phrases, Verb Phrases. It uses a regular-expression based parser and finds the similarity between key-ideas presented in the essay. The same is done using a unigram model, and a bigram model, interpolated according to a calibrated interpolation model, and a score is assigned.

Calibrator A calibrator aggregates all the scores from different modules and fits them into a calibrated linear regression model. The basic function and utility of the calibrator is to provide a score weighed appropriately according to different textual cues as a human grader would do.

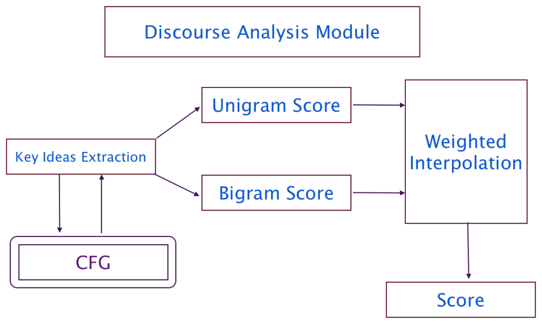

Figure 3: Discourse Analysis Module

Text from Image Using Tesseract-OCR Engine

We have added the capability to just click the images of typed or printed essays as photos. The images are parsed using the Tesseract OCR Engine to extract raw text and then are sent for processing.

DeepScore

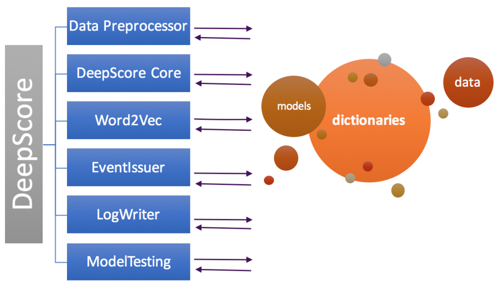

Figure 4: DeepScore - Block Diagram

We built a 4-layer Neural Network and an LSTM network for the second fold of our study. DeepScore runs primarily on the 4-layer neural network model which works over 300-dimensional GLoVE Vectors.

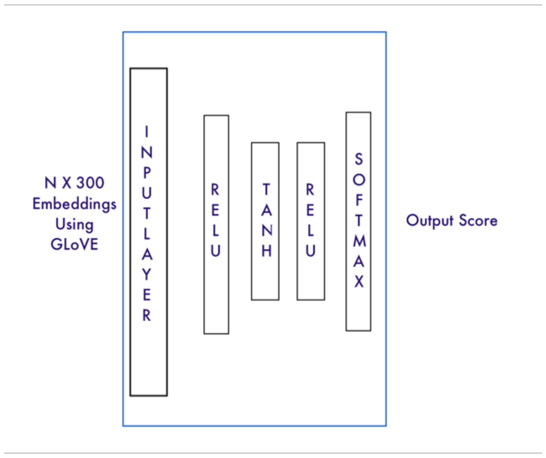

Figure 5: DeepScore - 4-Layer NN

GLoVE Vector Embeddings An essay is preprocessed using UTF8 decoding. If the tokenizer fails to decode a token, it is then decoded using **latin-1** decoding scheme. We are using the GLoVE (Jeffrey Pennington 2014) vector embeddings to represent all words in an essay and then an averaging technique to get a 300-dimensional word embedding for an essay. We achieved quite good results using pre-trained GLoVe word vectors with a simple averaging technique. This part required several optimizations. Considering the size of the corpus, (42 Billion tokens), we need to find a way to do an efficient lookup and at the same time ensure that we do not look-up for the same word again in the corpus. To worsen the situation, we also had to deal with clumped words, such as *"gothere"*, for which our preprocessor decomposes the clump, and finds the word vectors of the constituents.

For fast lookup, we just match the first letter of the word in the corpus and skip if there isn't a match. This largely helps rather than doing a linear search for a token in the corpus. If the first letter is matched, we further match the split token. This has lead to several performance gains and the same have been presented in the Analysis section.

Training

4-Layer Feed-forward NN We used a 4-layer neural network for training purposes. With a **Cross-Entropy Loss** function, and an objective to maximize the accuracy.

For a set of predictions $y\prime$, and their corresponding actual scores y, the objective is to minimize the cross entropy loss. More specifically, we are trying to classify a given set of essay-vectors into 12 possible classes, labeled using 1-hot vector encoding.

The objective is thus a classification problem which predicts an output $y \in \{0, 1\}$, for an essay vector $\mathbf{x}$. The probability for predicting each outcome is modeled using the logistic function

$$\phi(z) = 1/(1 + e^{-z})$$

If y and $y\prime$ represent the actual and predicted classes respectively, we can use cross entropy to get a measure for similarity between y and $y\prime$:

$$H(y, y\prime) = -\sum_i y_i \log y\prime_i = -y \log y\prime - (1-y)\log(1-y\prime)$$

We are using **ADAM Optimization Algorithm** (Diederik P. Kingma 2014) for better results than normal gradient descent. We chose Adam Optimizer for two main reasons:

1. The Adam Optimizer uses moving averages of the parameters. This enables the model to use a step size that is effectively larger, and
2. The algorithm converges to a step size without fine tuning.

Finally, we discuss the specifics of our Neural Network model. We have implemented a 4-layer feed-forward neural network which has a ReLU activation layer of 12 neurons with 300-dimension input.

$$h^{(1)} = \mathtt{max}(0, \mathbf{W^{(1)}}\dot{\mathbf{x}} + b^{(1)})$$

Following the ReLU Layer we have a Hyperbolic Tangent layer:

$$h^{(2)} = \mathtt{tanh}(\mathbf{W^{(2)}}\dot{\mathbf{h^{(1)}}} + b^{(2)})$$

Followed by a 8-neuron ReLU layer:

$$h^{(3)} = \mathtt{max}(0, \mathbf{W^{(3)}}\dot{\mathbf{h^{(2)}}} + b^{(3)})$$

Finally, to ensure a sooth analytic distribution, we have a 13-neuron output layer with `softmax` activation:

$$y\prime = \sigma(\mathbf{h^{(3)}})_j = \frac{e^{h_j^{(3)}}}{\sum_{k=1}^{K} e^{h_k^{(3)}}}$$

LSTM-based Model We designed and implemented a simple LSTM model with a scoring layer at the end. The LSTM takes words of an essay as time-steps, as it receives a stream of word vectors for the essay and computes a vector that captures the information present in the essay. The output layer converts this vector into a continuous score value.

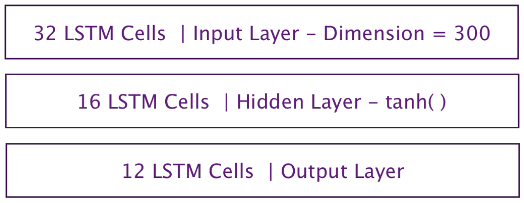
32 LSTM Cells | Input Layer – Dimension = 300
16 LSTM Cells | Hidden Layer – tanh( )
12 LSTM Cells | Output Layer

Figure 6: Single Timestep LSTM Model

Table 1: Results

Model	Train	Test
2-layer NN with `sigmoid` activation	87.5%	41.50%
Single Timestep LSTM	94.07%	89.60%
DeepScore: 4-layer NN	**94.37%**	**92.04%**
Taghipour et al (EMNLP'16): CNN+LSTM(10)	-	82.1%
EASE (SVR)	-	76.1%
UNTraSE	74.76%	74.76%

Results

We were able to improve the current state-of-the-art system (Taghipour and Ng 2016) by **9.34%**, using DeepScore. But more surprisingly, we were able to achieve a near accuracy to EASE (SVR) with UNTraSE, which is an unsupervised system ! (Note that since UNTraSE does not require training, we report the same for training and testing).

Findings and Analysis

Hypothesis

Our initial hypothesis is that with proper feature engineering, it is possible to achieve around the same value of QWK as with state-of-the-art techniques. While our current unsupervised model isn't as good as the Neural Network based model, it has nearly the same accuracy as other state-of-the-art systems like EASE (edX 2013), which are supervised. It is an undeniable fact that neural networks possess a very high expressive power compared to hand engineered features.

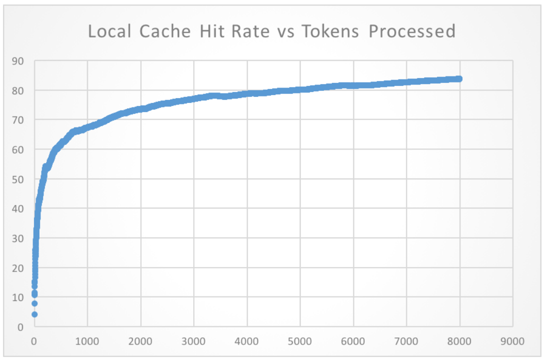

Figure 7: Cache Hit Rate vs Tokens Processed

Findings on Modularization

Our goal was to create an unsupervised essay grading system which performs as good as the state-of-the-art systems which run supervised learning algorithms and churn out on huge datasets. We introduced a modular design similar to (BURSTEIN and MARCU 2000) for building upon the current feature set by adding new textual cohesive devices which correspond to diverse characteristics of writing as perceived by a human grader. Our model clearly shows that modularity in such scoring systems help improve a lot. Our modular design currently includes syntactic variety , the similarity of ideas , correct usage and grammar, and organization of ideas at a very high level.

Optimizations on Lookup Timings

Our system is highly efficient and adapts very quickly to the new words learnt. All new words are stored in a local cache which prevents the system to search again for the same token in the whole corpus, thus saving time. Figure above shows the variation of cache hit rate as the number of files, and thus tokens are processed. As we can see, the hit rate rises very quickly and nearly stabilizes at 90%.

Conclusions & Future Work

We plan to continue working on the project and end up with a publishable work. These are some additional milestones we have planned to achieve. Some of the optional features of the system which we will implement are -

1. Research on how feature engineering coupled with RNNs can allow even better results.
2. Performing a user-study involving volunteers to write essays and then grading it using our system, comparing it with human graded scores.
3. Add support for extracting text from scanned or even clicked images of essays.

References

BURSTEIN, J., and MARCU, D. 2000. Benefits of modularity in an automated essay scoring system. *ISI, University of Southern California (Association for Computational Linguistics)*.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement. 20 (1): 3746.*

Diederik P. Kingma, J. B. 2014. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*.

edX. 2013. Enhanced ai scoring engine. *edX.org*.

Forman, G., and Cohen, I. 2004. Learning from little: Comparison of classifiers given little training. *Hewlett-Packard Research Laboratories 1501 Page Mill Rd., Palo Alto, CA 94304*.

Jeffrey Pennington, Richard Socher, C. D. M. 2014. Glove: Global vectors for word representation. *Computer Science Department, Stanford University, Stanford, CA*.

Manvi Mahana, Mishel Johns, A. A. 2012. Automated essay grading using machine learning. *Stanford University*.

Shermis, M. D., and Burstein, J. C. 2002. Automated essay scoring - a cross disciplinary perspective. *Routledge, 2002*.

Taghipour, K., and Ng, H. T. 2016. A neural approach to automated essay scoring. *Conference on Empirical Methods in Natural Language Processing (Association for Computational Linguistics)*.