A Comparison of Training Models on the Accuracy of Automated Essay Scoring Systems

AP Research

May 2020

Word Count: 5482

Abstract

The field of automated essay scoring (AES) attempts to create a machine learning model which understands the various qualities of essay writing. The performance of such models significantly varies between datasets and architectural approaches. In this paper, several approaches are applied to a diverse set of essay samples to explore state-of-the-art methods and assess the impact of various choices on system performance. Code for this project is publicly available at https://github.com/davidheineman/comparisonaes.

Keywords: automated essay scoring, applied neural networks, natural language processing, feature engineering, machine learning

Table of Contents

A Comparison of Training Models on the Accuracy of Artificial Essay Scoring Systems

Introduction

With greater demand for essay scoring over recent years, reliable grading has become a focus for scoring agencies (Taghipour & Ng, 2016). An automated alternative would provide additional resources to a labor intensive and tedious task and alleviate time constraints resulting in faster, more accurate grading. Automated Essay Scoring (AES) applies natural language processing and computational modeling to accept an essay as input and assign it a numerical score. The system reflects on the quality, content, organization and grammar of an essay to best mimic a model grader. Such a scoring ability would expedite essay grading and minimize human involvement in the entire process. AES may also allow for detailed and targeted feedback to identify strong and weak points within students' writing (Zesch et al., 2015). AES systems consist of two key components: a preprocessor, which prepares input data and a training model which creates a system to assign scores. This paper will compare novel training models and their ability to score essays across standardized conditions.

Review of Literature

This portion evaluates current research on feature selection methods, current training methods and concerns within the field. It will conclude with a discussion of the dataset used in this project and the role of this paper.

Feature Selection Methods

Feature selection, which is the grooming of data to use as input into the training model, exploits quantifiable measures of an essay that correlate with a high score to synthesize input data. Feature-based AES systems convert each essay into a set of numbers which represent the syntactic and stylistic features of an essay, known as an essay vector. Systems which accept an

essay vector as input are known as supervised learning models and make up traditional AES models. Supervised learning requires meaningful features to assign scores accurately, thus significant research has been published on various feature-selection methods.

The simplest features are statistical metrics, such as word count or sentence length. These counts may be synthesized into high-level features like readability and word-to-sentence ratios (Zesch et al., 2015). Beyond statistical features, low-level features[1] have been developed to measure more abstract concepts such as lexical complexity or writing level. Scores for these metrics are created by analyzing the pairs of two, three or four words in each text, known as n-grams[2] (Chen & He, 2013). Various high-level syntactical features, which score the organization and flow of ideas, have also been proposed (Farra, 2015). Diction-based (or linguistic) features, in particular, have been a staple for supervised systems, most using Latent Semantic Analysis to find a lexical similarity between essays (Klebanov & Flor, 2013). Using n-grams to calculate semantic similarity has been used beyond comparing words within essays to comparing the similarity of essays to their prompt (Farag et al., 2018) and to hand-picked high, medium and low scoring examples (Klebanov & Flor, 2013). Text coherence has also shown to be a key feature of AES development and can be calculated by avoiding semantic similarity and analyzing changes in tone and organizational elements instead (Somasundaran et al., 2014).

More complex features see less implementation across academic writing. Numerous features aimed at specific prompt types, such as persuasive or research papers, provide a more precise analysis of complex essays (Persing & Ng, 2015). Some even use a student's test scores and performance on other test sections as a feature (Crossley et al., 2015). These task-dependent

[1] "Low-level" features analyze individual phrases and sentences independently, as opposed to "high-level" features which reflect on the entire essay at once.
[2] *n* represents the number of neighboring terms taken. Tri-grams (*n* = 3) are exclusively used in this paper.

features are more effective when aimed towards specific types of prompts (Zesch et al.. 2015) and are beyond the scope of this paper.

The above feature-preprocessing step is utilized in order to assign a set of values, known as an essay vector, which correlate the quantifiable elements of an essay with its score. A minor change in the pre-processing step can drastically change the output performance of any training model, which makes isolating how training models compare among papers a difficult task, given that these papers draw from varying datasets and feature-selection models. Additionally, hand-crafted features are complicated to create, and thus parsers and pre-built semantic analyzers are often a necessary part of the preprocessing step, often using systems not built for AES, rendering them unreliable in practice (Alikaniotis et al., 2016). Hand-crafted features, despite being accurate correlations of essay score, do not capture the full nuances of grading essays, and the process of understanding an essay is too involved to be completely considered by such features. Similarly, linguistic features require time consuming, manual identification of high-scoring sample essays (Klebanov & Flor, 2013). Feature engineering is also the most involved and difficult part of building a feature-based AES, meaning that a system that works for one dataset may not be easily transferable because of the nature of that dataset's hand-crafted features.

Despite these obstacles, feature-based AES systems are the dominant commercial option for testing agencies. The most popular commercial AES product, e-rater, relies on shallow feature-selection and has been used in various high-stakes assessments (Attali & Burnstein, 2006).

Unsupervised Learning Models

With the drawbacks of feature-based models, recent papers have increasingly turn towards unsupervised neural networks, since this approach removes the need for feature-

selection altogether. Unsupervised training models are defined in an end-to-end space representation, instead of accepting an essay vector which reflects the essay's features, they accept the essay itself, so are theoretically able to capture complex concepts and abstract ideas that would be overlooked by feature-based models. The dense and nuanced nature of unsupervised learning allows a model to be capable of extracting insights that are significantly more difficult to quantify through a hand-crafted feature, seeing as the model identifies these features itself (Chelba et al., 2013). Additionally, neural models appeal to AES applications because of their "end-to-end" nature, meaning they can be easily trained on a new corpus without the need to change the model architecture or create customized hand-crafted features.

In AES, unsupervised models use a word embedding to convert the words of each essay to a dense numerical representation which encapsulates the meaning of each word and its relationship to others, known as a word vector (Pennington et al., 2014). The series of word vectors which create each essay serves at the input to the unsupervised model. Various papers create word vectors using publicly available word embeddings trained on generic corpuses (such as word2vec, etc.) (Wang et al., 2018) but others have proposed training a custom word embedding on the essay inputs themselves, creating a Score-Specific Word Embedding (SSWE) (Alikaniotis, 2016). SSWEs, because of specialization towards each corpus, have shown to improve scores on baseline neural networks (Nguyen et al., 2016). The first major unsupervised AES utilized a Long Short-Term Memory network, (LSTM) (Hochreiter & Schmidhuber, 1997) proposed specifically for its ability to recall information, which allowed the model to develop an understanding of the topic. The LSTM model did not outperform state-of-the-art feature-based models but did when using a SSWE to create the input (Nguyen et al., 2016). A two-layer Convolutional Neural Network (CNN) has been proposed with an ability to encode each essay in

two independent ways (Dong & Zhang, 2016). The first layer used each word as input, allowing the network to analyze low-level features like diction and grammar. The second utilized string kernels to encode each essay, allowing the network to analyze syntax, organization and development of ideas. Variations of a two-layer CNN have shown to slightly improve scoring accuracy, one adding another Recursive Neural Network (RNN) to incorporate the benefits of low-level and high-level scoring (Taghipour & Ng, 2016). Instead of embedding the essay, another approach is to apply hierarchical classification, which utilizes both an essay vector and a feature set as additional input, allowing for features to contribute to a neural model (McNamara et al., 2015 & Yannakoudakis et al., 2011).

The central theme of AES research has shifted from applying supervised to applying unsupervised training models to various datasets. However, neural networks, due to their end-to-end nature, are nebulous and fail to provide insight into grading decisions. The benefit to feature-based models is their ability to provide detailed feedback as to why a score is awarded (Alikaniotis et al., 2016).

The Importance of Dataset Standardization

Comparing training models between AES research is difficult because various models might have different feature-selection methods, embedding methods or datasets altogether. For example, longer papers whose grades focus on abstract ideas rather than writing quality are significantly more difficult for an AES to score (Yang et al., 2018). Not only does a dataset effect AES performance, but the wide variety of feature selection methods and embedding techniques used between AES studies make comparative analysis of training models difficult (Zesch, 2015). In 2012, the “Automated Student Assessment Prize,” (ASAP) funded by the Hewlett foundation, provided detailed training data which has been used by multiple AES studies. The dataset

consists of twenty-thousand pre-scored essays under eight different prompts which vary in topic, complexity and style. This dataset will allow for comparative analysis of the effects of prompt type on model performance. The purpose of this paper is to compare a diverse set of feature-based and neural-based training models on a standard feature-set, embedding technique and dataset.

Methodology

In this section, the reasoning and assembly for the preprocessor, models and training metric will be introduced. It will conclude with an explanation of the evaluation metric and a discussion of the dataset.

Feature Selection

For supervised AES models, feature inputs can be categorized into four types: Grammatical, semantic, content and synthetic features. This section will briefly introduce each type and explain how each were determined. An enumerated list of features and dependencies utilized to determine each feature are included in Appendix A.

Grammatical Features

The preprocessor begins by correcting the grammar of individual words against an open-source English corpus. The number of incorrectly spelled words was stored, and the corrected essay, due to its ability to better demonstrate content, was used as the input for all other features.

Semantic Features

Statistical metrics, which broadly refer to various counts of lexical features, were also calculated. Essays were tokenized into words and sentences, which allowed simple character, word and sentence counts, as well as for counting the lengths for each word, sentence and paragraph. The frequency of each type of punctuation and Part of Speech (POS) tags are also

counted. A Name Entity Recognizer (NER) was used to identify unique nouns and their type. Before continuing, stop words, given by a generic set of 50 commonly used words, were identified and removed before other features were implemented to make the following substantive features more representative of meaningful verbiage. Additionally, all tokens (words not included in the stop word corpus) were counted. Each token is scored against a generic corpus and combined to quantify the essay's writing level. The average difficulty of words is stored as a feature. Tokens are also used to find the Kincaid readability score, a metric commonly used in AES, given by:

$$r = 206.835 - 1.015\frac{|t|}{|s|} - 84.6\frac{|syl|}{|t|}$$

Where $|t|$, $|s|$, $|syl|$ represent the number of tokens, sentences and syllables respectively. Additionally, various other reading scores were included for comparison.

Content Features

The frequency of unique terms used in one essay (Term Frequency) multiplied by the inverse of the frequency of that term used among the set of essays (Document Frequency), gives a Term Frequency-Inverse Document Frequency (TF-IDF) matrix which quantifies the importance of every word within one essay, given how each word is used among the set of essays. A TF-IDF matrix is generated for each set of essays and for each individual essay. The process of Latent Semantic Analysis (LSA) then utilizes both TF-IDF matrices to calculate a vector representation for each essay. Using these vector representations, a cosine distance may be found which can quantify the relationship between any two essays. For each set of essays, a high, medium and low scoring essay, which accurately represented the topic, was hand-picked. Using LSA, a similarity is calculated given cosine distance of their vector representations provided by

the TF-IDF matrix. Additionally, the LSA similarity is calculated between the essays and the verbiage used in their respective prompt.

Synthetic Features

Synthetic features, a function of two or more features, were calculated to add insight into the relationship between basic features. The average word length (the ratio of total characters to total words), type-token ratio (the ratio of unique tokens to total tokens) and lexical diversity (the ratio of name entities to tokens) are notable examples, but for the sake of brevity the full list of synthetic features can be found in Appendix A.

Supervised Model Implementation

Before feeding the preprocessed features into each model, the features were converted to a standard scale dictated by their z-score. Each supervised model was implemented using the Scikit-learn framework, various out-of-the-box functions were used to fit each supervised model. To find the optimal conditions, hyperparameter tuning was performed for each model using Grid Search. As a baseline, a simple Linear Regression is applied which finds the most representative linear relationship between the feature data and the score.[3]

Support Vector Machine

The Support Vector Machine (SVM) is an alternative to the regressor Support Vector Regression which is adapted for classification-based learning (Vapnik et al., 1996). The SVM attempts to maximize the margin between each of an essay's features by generating a function. Unlike a linear regression, the SVM is adaptable to a non-linear feature space, which is useful for AES features that typically have a non-linear distribution. Additionally, the SVM is more suited than the baseline for high-dimensional features because of its simplicity in generating higher-

[3] As an additional baseline, an ElasticNet is trained, which is similar to a linear regression, but includes L1 and L2 regularization methods to prevent overfitting.

dimensional regressions. To take advantage of the ability for a SVM to efficiently separate high-dimensional data, a Radial-Basis Function (RBF) kernel is applied which generates trivial synthetic features to scale inputs into higher dimensions.

Random Forest Ensemble

The Random Forest Ensemble consists of numerous, independently trained Decision Trees which act as several nodes to collectively contribute to a final score. Each decision tree is randomly initialized and trained on a subset of the given training data to generate a rudimentary set of rules (known as decision boundaries) which separate points within the feature space. During evaluation, the distance between a tree's decision boundaries and prediction is used to output confidence in its decision. The total of each node's prediction, weighted by its confidence, outputs the singular score for the Random Forest Ensemble. The Classification and Regression Tree (CART) method is used to train the decision trees (Lewis, 2000).

k-Nearest Neighbor

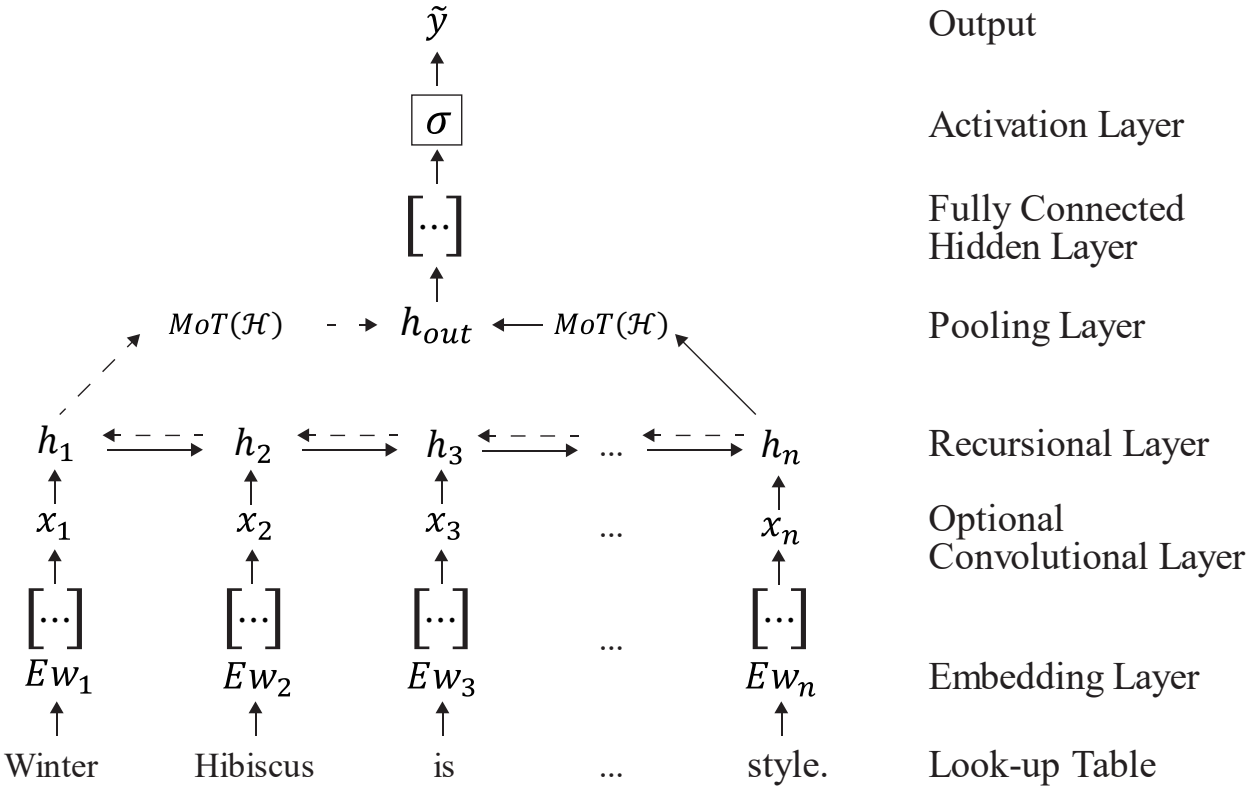
The k-Nearest Neighbor (kNN) attempts to separate essays indiscriminately into categories and assign class labels once evaluation on the dataset has finished. The most common score of the k closest neighbors of the training data is used to assign scores to new input data, weighted by the distance of each neighbor.

Unsupervised Model Implementation

This section will propose a framework from which all unsupervised models are assembled. Figure 1 depicts the overall architecture for a generic unsupervised model. In testing, the embedding and activation layer are applied to all unsupervised models, but the layers which generate the essay representations vary between the models tested.

Figure 1

Illustration of unsupervised learning network architecture.



Embedding Layer

The high-dimensional dense word vector representation is generated from the publicly available Global Vector (GloVe) dataset, retrained on each prompt to create a SSWE. Each essay is then converted to a dense embedding representation defined by $E \in \mathbb{R}^{N \times L}$ where N is the embedding dimensionality and L is the average word length of the essay samples. Higher-dimensional embeddings theoretically store more information about each word (Pennington et al., 2014), so different dimensionalities are tested. The final embedding table L is given by:

$$L = (Ew_1, Ew_2, \ldots, Ew_n)$$

Where w_n is a one-hot identifier for each word and E is the trained SSWE embedding matrix.

Convolutional Neural Network (CNN)

The convolutional layer has traditionally been used in image recognition and has shown effectivity in condensing representations of long groups of text. Each group of word representations are combined into n-grams and are pooled to create higher-level features. Theoretically, the layer synthesizes local features between pairs of words to generate a broader representation of the text. This makes it effective at capturing features stored in the essay's n-grams. Each n-gram embedding representation is concatenated to form $\tilde{x}$, where n is the window size. The $\tilde{x}$ representation has dimensionality $\tilde{x} \in \mathbb{R}^{N \times L}$, where L is the total dimensionality of all combined embeddings. The output from the convolutional layer is given by:

$$Conv(x) = W_{\tilde{x}}\tilde{x} + b_{\tilde{x}}$$

Where $W_{\tilde{x}}$ and $b_{\tilde{x}}$ are a trainable weight and bias which apply to the overall layer. The convolutional layer can be applied both independently and alongside a recursional layer, both tested in this paper.

Recursive Layers

A recursive neural network (RNN) applies a series of directional transformations to generate a collapsed representation of the essay. They apply a transformation to each word within the essay independently.[4] Due to their directional nature, they are capable of scaling to large lengths of text, which has made RNNs a staple of Natural Language Processing and AES, whose input typically involves long sequences and variable lengths of text. The recursive layer can accept either a series of embeddings or a convolutional representation of an essay as input. Two basic recursional layers are tested, a traditional RNN and a Gated Recursional Unit (GRU) network, with the GRU including a basic trainable output transformation.

[4] Each independent transformation is known as a timestep. The number of timesteps equals L.

Long Short-Term Memory Network (LSTM): The LSTM is a modified recursional layer which stores information during each step of evaluation to expand the capability of understanding long sequences of text. In training, the LSTM learns to strengthen or forget certain insights from the text which are incorporated in later representations (Hochreiter & Schmidhuber, 1997). The LSTM provides an output after each word is individually incorporated within the model. The following equations formally structure the LSTM:

$$i_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$f_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = i_t \circ \tilde{c}_t + f_t \circ c_{t-1}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \circ \tanh(c_t)$$

Where x_t is each input vector and h_t is the processed representation at timestep t. With W_i, $W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and b_i, b_f, b_c, b_o being trainable weight and bias variables respectively. $\circ$, σ, $\tanh()$ represent the Hamdard product, sigmoid function and hyperbolic tangent function respectively.

Bi-Directional LSTM: The LSTM processes essays in a left-to-right method, accepting the first input vector and processing its representation before being exposed to the next word. Thus, words that change the meaning of a sentence may be completely ignored by the LSTM because of its left-to-right nature. Thus, another essay representation can be generated with an equivalent LSTM processing the word vectors in a right-to-left method. The BLSTM generates two processed outputs, $\vec{h}_t$ and $\overleftarrow{h}_t$.

Mean over Time: The output from any recursional layer is a final, learned essay vector calculated from the entire input. However, because the vector generated at each timestep by the respective transformations of each recursive gate can encode relevant information about the final score, the vector at each timestep is stored (Taghipour & Ng, 2016). Thus, an average pool of all timesteps is found by applying a Mean over Time (MoT) function given by:

$$MoT(\mathcal{H}) = \frac{1}{N}\sum_{t=1}^{N} h_t$$

Where $\mathcal{H} = (h_1, h_2, \dots, h_M)$ are the intermediate timesteps given by the recursional layer. For vectors processed through the Bi-Directional LSTM, both outputs are separately passed through the MoT layer and concatenated into a singular representation given by:

$$\widetilde{\mathcal{H}} = W_{\widetilde{\mathcal{H}}}\begin{pmatrix} MoT(\overrightarrow{\mathcal{H}}) \\ MoT(\overleftarrow{\mathcal{H}}) \end{pmatrix} + b_{\widetilde{\mathcal{H}}}$$

Where $W_{\widetilde{\mathcal{H}}}$ and $b_{\widetilde{\mathcal{H}}}$ are trainable parameters.

Feed-Forward Activation Layer

Although it is possible to bypass the MoT layer and accept only the final state h_t, for any model, the network is more accurate by using the average state (Taghipour & Ng, 2016). Finally, the MoT state is passed through a fully connected hidden layer, which collapses the high-dimensional input into a singular representation. Each hidden layer is defined as:

$$h_{out} = ReLU\big(W_h(m_1, m_2, \dots, m_n)\big) + b_{out}$$

With $ReLU$ representing the non-linear Rectified Linear Unit activation function and m_t being the final MoT representation of each word. The feed-forward output is not constrained to a range, so a simple activation layer must be used to bound the output to a range $[0,1]$. The final score is given by:

$$\tilde{y} = \sigma(W_{\tilde{y}} h_{out} + b_{\tilde{y}})$$

Where $W_{\tilde{y}}$ and $b_{\tilde{y}}$ are trainable parameters and σ represents the sigmoid activation function. In this case the bias $b_{\tilde{y}}$ is set to the mean gold-standard (or human-graded) score of all samples, which has proven helpful for aligning scores within the prompt range (Tay et al., 2017). During training no other operations are performed, but in evaluation the output is normalized to the range of scores.

Optimization

The optimization function Adam is used to minimize loss due to its simplicity, effectiveness and reliability on complicated networks (Kingma & Ba, 2014). Backpropagation adjusts all trainable variables in layers up to the embedding layer. All models are evaluated using a 5-fold cross evaluation which averages the performance to normalize the random varying ability of each model. The final hyperparameters for all unsupervised models are included in Appendix B.

Evaluation Metrics

Training: Mean Squared Error

Within training, each model attempts to minimize the Mean Squared Error (MSE) to generate the least square model. After the MSE ceases to increase after 5 consecutive training periods, the instance with the lowest MSE on a separate essay set (the validation set) is selected as the final model. The MSE is given by:

$$MSE(y, \tilde{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \tilde{y}_i)^2$$

With N number of essays, $\tilde{y}$ prediction score generated by the AES system and y gold-standard score. The difference is squared and averaged to incorporate all training data and weigh inaccurate scores parabolically.

Evaluation: Quadratic Weighted Kappa

Quadratic Weighted Kappa (QWK) is a metric exclusively used for AES studies and was popularized through the ASAP competition (Shermis, 2012). Due to its synonymy among research, it is used as the evaluation metric to compare models. QWK scores range from $[-1,1]$, with 1 being perfectly and 0 being randomly agreeable. The QWK evaluation begins with a N-by-N matrix W_{ij} which represents a weight matrix where each value corresponds to the percentage of scores that fall under each combination of scores:

$$W_{ij} = \frac{(i-j)^2}{(N-1)^2}$$

Where i, j, and N are the gold-standard scores, AES prediction and total number of essays respectively. Additionally, a N-by-N matrix O_{ij} is constructed corresponding to the number of gold-standard and prediction scores respectively. Another N-by-N matrix, E_{ij} is calculated as the product of the histogram vectors of both scores. E and O then are normalized to output the same sum. All matrices are used to calculate the final QWK score:

$$\kappa = 1 - \frac{\sum_{i,j} W_{ij} O_{ij}}{\sum_{i,j} W_{ij} E_{ij}}$$

Dataset

The data for this paper is provided by the Hewlett Foundation Automated Student Assessment Prize (ASAP) contest, a public competition aimed at designing the best performing AES system. Essays within the dataset have been altered to censor personal information using the Stanford NER, so the NER implemented in the preprocessing step was altered to identify the

Table 1

Description of ASAP Dataset

Essay Set	Training Size	Score Range	Grade Level	Essay Type	Avg. Length
1	1783	2-12	8	Narrative	350
2	1800	1-6	10	Narrative	350
3	1726	0-3	10	Source-Based	150
4	1772	0-3	10	Source-Based	150
5	1805	0-4	8	Source-Based	150
6	1800	0-4	10	Source-Based	150
7	1569	0-30	7	Expository	250
8	723	0-60	10	Expository	650

censored information. From the ASAP data, only the input essay and final score was used to train and evaluate each model. Additionally, the data includes human-based scores for each essay, which allows for measuring machine-human and inter-human agreement. Table 1 summarizes the ASAP dataset.

During evaluation, the dataset was split randomly into the ratio: 70% training set, 15% validation set and 15% test set. The models were trained on the training set chosen by the minimum loss on the validation set and the final QWK score was calculated using the test set. The same split of data was used as input to every model evaluated in the project. All models were developed in Jupyter Notebooks using Python 3.7, with supervised models evaluated using out-of-the-box Skikit-Learn functions and unsupervised models evaluated using Keras and Tensorflow 2. Training was performed using a Nvidia GTX 1060 and Intel Core i7-7700.

Findings & Discussion

This section evaluates individual findings as directed by the methodology and explores independent variables' effect on model performance. Table 2 illustrates the QWK scores for both supervised and unsupervised models among each of the eight prompts, and the average score among all prompts. The 5-fold cross evaluation scores are shown, with each value being the

Table 2

QWK scores for supervised and unsupervised models organized by prompt. "Avg." denotes a simple average of QWK scores among all prompts

	1	2	3	4	5	6	7	8	Avg.
Forest Ensemble	**0.826**	**0.683**	0.618	0.696	**0.784**	**0.683**	**0.740**	*0.678*	*0.713*
Elastic Network	*0.819*	*0.673*	*0.643*	*0.699*	0.779	*0.663*	*0.739*	**0.717**	**0.717**
Linear Regression	0.796	0.653	0.610	0.735	0.741	0.650	0.676	0.617	0.685
kNN Classifier	0.800	0.628	**0.669**	**0.701**	*0.780*	0.637	0.651	0.473	0.667
SVM	0.646	0.605	0.649	0.676	0.747	0.602	0.623	0.479	0.628
BLSTM	*0.820*	0.635	**0.663**	**0.801**	**0.770**	**0.791**	**0.747**	**0.651**	**0.735**
BLSTM-CNN	**0.830**	**0.652**	*0.645*	0.762	0.762	*0.789*	0.743	0.625	*0.726*
CNN	0.757	*0.636*	0.644	*0.787*	*0.778*	0.779	*0.765*	*0.638*	0.723
LSTM	0.789	0.542	0.627	0.773	0.775	0.781	0.753	0.567	0.701
LSTM-CNN	0.724	0.510	0.620	0.755	0.762	0.780	0.716	0.532	0.675
RNN-CNN	0.506	0.445	0.574	0.629	0.557	0.650	0.631	0.499	0.561
RNN	0.332	0.506	0.462	0.633	0.158	0.406	0.521	0.273	0.411
GRU	0.125	0.228	0.125	0.301	0.080	0.334	0.403	0.265	0.233
GRU-CNN	0.000	0.001	0.000	0.081	0.041	0.005	0.033	0.015	0.022

average QWK score across 5 evaluations, enumerated in Appendix C. Unsupervised models with a "-CNN" are RNN models with an additional, optional convolutional layer which collapses the embedding representations before prior to processing the essay. The "CNN" model features the same structure as the RNN models without a recurrent layer. The highest QWK score for each essay is underlined, the highest for each section (supervised, unsupervised) in bold and the second highest italicized. The highest average QWK score was performed by the BLSTM, with the BLSTM-CNN and CNN models also performing higher on average than the best performing supervised network. The BLSTM consistently outperformed the unsupervised networks in Source-Based and Expository prompts (3-8), while no supervised network performed clearly strong among all prompts. However, the Elastic Network performed best among supervised networks despite only having the highest average QWK score on one prompt. The RNN, GRU and GRU-CNN models severely underperformed and, through further investigation, this is

attributed to divergence from the loss function during training, typically the drawback of an oversimplistic network.

For recurrent models with the addition of a convolutional layer, change in model performance was inconsistent. For the BLSTM and LSTM networks, an addition of the CNN decreased average performance by .009 and .026 respectively, with the only improvement being shown by the BSLTM-CNN on narrative prompts (1-2). This is likely attributed to the additional architecture created by the convolutional layer, which collapses embedding information by combining neighboring words into a more efficient representations. For already complex recurrent networks, the collapsed representation likely created unnecessary noise to parse out, resulting in consistently lower QWK scores. However, both the BLSTM and LSTM consistently outperformed simpler recurrent networks (RNN & GRU) by between 0.114 and 0.713 on average. Long-term recurrent networks likely performed better due to their ability to retain information from earlier in a sequence, making them ideal for the typical length of AES input essays. The Bidirectional nature of the BLSTM improved performance over the LSTM by 0.051 and 0.034 (with and without a CNN layer, respectively), which is attributed to being able to apply information from both the beginning and ending of sequences.

Among prompts, model performance was most dependent on the prompts' training size and grade level. Being 58% smaller than the average training set size and having the largest average word length, model scores on prompt 8 are consistently lower among unsupervised models. Additionally, prompts 1, 5 and 7, given to students on the middle school grade level, had higher QWK scores between models than those given to high school students (an average increase of 0.073). No concrete correlation was found between the type of essay given and model

performance. However, because each dataset varied in multiple ways, no conclusive assertions can be made about the impact of prompt grade level or essay length on model performance.

Human-Machine Agreement

Table 3 contains the QWK scores between four high-performing AES models, two singular human graders and the final essay score. The two models scored with the highest average QWK were taken from each category[5]. AES 1, 2, 3 and 4 denote the BLSTM, CNN, Forest Ensemble and Elastic Network networks, respectively. Human-machine (AES-H) agreement which outperforms baseline inter-human (H1-H2) agreement is denoted in bold. The BLSTM and Elastic Network perform significantly higher in human-machine agreement than the next highest accuracy model by type (by 0.088 and 0.111, respectively). In no case does the machine-domain (AES-D) agreement outperform human-domain (H-D) agreement; denoting machine performance does not match human performance on any prompt or model. However, high machine-human agreement, shown by the BLSTM (AES1) and Elastic Network (AES4) across all prompts, exemplifies an approaching similarity between AES models and human graders.

[5] Exempting unsupervised networks whose architecture are too similar to make substantive comparisons. In this case, the third highest unsupervised network, the CNN was chosen as opposed to the BLSTM-CNN.

		H1	H2	D	AES1	AES2	AES3	AES4
	H1		0.721	0.926	0.702	0.572	0.629	**0.766**
1	**H2**			0.924	**0.724**	0.596	0.583	**0.789**
	D				0.832	0.720	0.663	0.855
	H1		0.691	0.814	0.658	0.459	0.594	**0.691**
2	**H2**			0.802	**0.700**	0.472	0.553	**0.696**
	D				0.658	0.459	0.594	0.691
	H1		0.769	0.914	0.659	0.555	0.559	0.600
3	**H2**			0.871	0.597	0.574	0.586	0.583
	D				0.697	0.600	0.588	0.617
	H1		0.851	0.933	0.752	0.733	0.684	0.706
4	**H2**			0.930	0.749	0.733	0.680	0.717
	D				0.788	0.746	0.720	0.738
	H1		0.753	0.883	**0.755**	**0.777**	0.695	**0.766**
5	**H2**			0.879	**0.761**	**0.781**	0.736	**0.792**
	D				0.782	0.783	0.740	0.808
	H1		0.776	0.893	0.760	0.714	0.640	0.718
6	**H2**			0.887	0.760	0.689	0.634	0.671
	D				0.804	0.736	0.665	0.717
	H1		0.721	0.924	0.612	0.597	0.558	0.700
7	**H2**			0.927	0.612	0.601	0.579	0.712
	D				0.750	0.730	0.622	0.780
	H1		0.624	0.883	**0.634**	0.494	0.379	0.617
8	**H2**			0.875	0.562	0.381	0.360	0.592
	D				0.651	0.480	0.389	0.664
	H1		0.738	0.896	0.692	0.613	0.592	0.696
Avg.	**H2**			0.887	0.683	0.603	0.589	0.694
	D				0.745	0.657	0.623	0.734

Table 3

Comparison of agreement (QWK) between high-performance models and human scorers. "H1", "H2" and "D" denote human grader 1 and human grader 2 and the domain score, respectively

Supervised Feature Analysis

To compare features, a trivial random forest regressor was trained on features from all datasets within the project to determine the Gini-importance of each feature. The Gini-importance denotes the likelihood of a feature to serve as the decision boundary on one decision tree, or the influence a feature has on distinguishing between well and poorly written essays. Table 4 contains the Gini-importance and corresponding standard deviation (SD) for significant[6]

[6] "Significant" is the most influential features of each type, not of overall highest Gini-importance, chosen to illustrate multiple types of features because Gini-importance doesn't necessarily indicate a "good" feature.

features organized by type. An enumerated list of features, Gini-importances, and SDs is provided in Appendix D. A higher Gini-importance implies a feature with greater influence over the scoring boundary while the standard deviation indicates how that importance may vary between decision trees. Semantic features, which contain superficial information about an essay, are shown to be the most influential feature on supervised models with Gini-importances consistently higher than other types. However, these features also have a high standard deviation, implying an interchangeability and unreliability of semantic features as predictors. The common AES readability metric enumerated in the method had a lower Gini-importance (1.43%) than the Coleman Liau and Smog readability metrics (2.34%, 3.75%) and 7 other synthetic features. Spelling errors, the sole grammatical feature, had the lowest Gini-importance of all sections.

Feature	Gini	STD
Content		
High-Scoring Similarity	2.16%	0.36%
Medium-Scoring Similarity	1.73%	0.30%
Low-Scoring Similarity	1.67%	0.12%
Polarity	1.53%	0.16%
Subjectivity	1.43%	0.12%
Grammatical		
No. Corrections	1.66%	0.11%
Semantic		
Non-Stop Word Count	3.06%	2.02%
Unique Word Count	3.03%	1.83%
Character Count	2.86%	1.53%
Word Count	2.73%	1.51%
Difficult Word Count	2.55%	1.33%
Synthetic		
Unique POS-POS Ratio	2.82%	1.87%
Coleman Liau Score	2.34%	0.38%
Unique Character-Character Ratio	1.80%	0.58%
Smog Score	1.75%	0.62%
Word-Sentence Ratio	1.74%	0.26%

Table 4

Gini-importance of selected features[6] utilized by the unsupervised models and organized into categories as dictated by the methodology.

Figure 2 displays the absolute value Pearson correlation coefficient ($|r|$) of features pairings ordered by Gini-importance. A high correlation between two features signifies less distinguishing information for a supervised model to observe when separating essays into scores,

a product of two features being "too similar." Since the direction of correlation has little impact on supervised model accuracy (Crossley et al., 2015), the absolute value was taken. Most strikingly, although semantic features had the highest Gini-importance of any type, they also highly correlate with each other, shown by the top 6 features semantic features which all have correlations $|r| > 0.82$. For content features, high and medium scoring similarity had a correlation of $|r| = 0.76$ while their correlation with the low scoring sample similarity was $|r| = 0.52$ and $|r| = 0.42$, respectively. High-Gini synthetic features like the Coleman Liau score, Smog score and Word-Sentence ratio had low correlations with other feature types but high correlations with each other ($|r| > 0.73$).

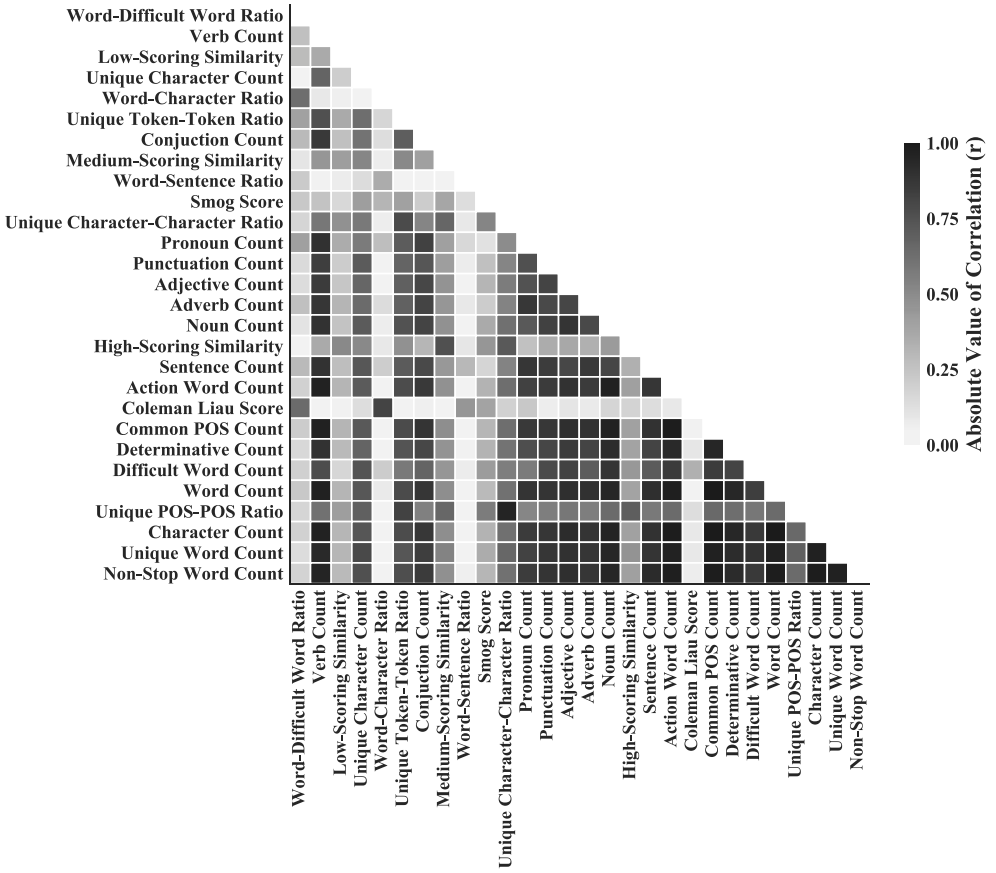


Figure 2
Heatmap of the absolute value of correlations between selected features of statistically significant Gini-importance[7] (> 0.16)

[7] Statistical significance was dictated by Gini-importance, ignoring category to highlight the correlation of commonly used decision tree boundaries.

Limitations

Due to the end-to-end nature of unsupervised models and the relatively complicated structure of high-dimensional feature-based models, it is difficult to confidently ensure the qualities of good essay writing are being learned. However, through 5-fold cross evaluation and splitting data into separate train, test and evaluation sets, the observed performance of each model can be used to predict the objective performance of that model in a generic AES setting. Additionally, through Table 2, the similarity between trained models and human scorers can be observed. Although the ability to confirm whether an AES model is fully understanding writing quality is impossible, the paper offered approaches to reveal human-machine agreement. Computational complexity was not observed, due to its relative unimportance in AES and the simplistic nature of NLP-based models.

Conclusion & Areas of Further Inquiry

The researcher has evaluated novel unsupervised neural networks and traditional supervised feature-based models on a standard, diverse dataset and compared the impact of architectural decisions on model performance. The decisions of feature development and model structure analysis can be extrapolated to generic AES applications in expository, narrative, and source-based prompts. The conditions of the study included a diverse set of training data to show the discrepancy in model accuracy between various types of prompts, lengths, and levels of writing. Although the QWK results are promising across models, AES remains an inadequate substitute for human scoring, as shown by Table 2, and have a vast potential for improvement. Supervised systems have been shown to consistently rely on semantic features, with content-based insights remaining weak in their ability to disambiguate essay quality, although these features have been proven consequential in model accuracy. Unsupervised systems have shown

to effectively process dense representations of essays to generate models with higher accuracy than the traditional methods despite lacking hand-crafted features.

Although this study serves a role in collating existing AES methods, future exploration into state-of-the-art representative features and denser unsupervised models is needed, particularly in developing more effective methods in understanding essay content. For example, ensemble methods, which combine both supervised and unsupervised approaches to create one output weighted by confidence[8] have yet to be introduced and would combine the insights of both approaches. Ensemble methods can also be used to train multiple models of the same architecture and pool results, leading to greater accuracy without the need for a novel approach. Additionally, AES research has primarily focused on models individually trained on separate prompts, with a major prerequisite to papers being a training size of >500 essays needed for an accurate model. In similar applications, researchers have tackled this problem through transfer learning, where a generalized unsupervised model is built which can be applied to AES situations through training (Bengio, 2012). Transfer learning, due to its relative novelty, has yet to be applied in AES, but has the potential of lowering the amount of pre-scored essays needed to generate an accurate model. Automated essay scoring has the ability to expedite the labor-intensive grading process and while this study produced no novel approach, it collated existing methods and laid the groundwork and direction for future exploration into these systems.

[8] Ensemble methods are synonymous in numerous machine learning applications, including natural language processing (Zhang & Ma, 2012).

References

Alikaniotis D., Yannakoudakis H., Rei M. (2016). Automatic text scoring using neural networks. *CoRP*. Retrieved from http://arxiv.org/abs/1606.04289.

Bengio, Y. (2012, June). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning* (pp. 17-36). Retrieved from http://proceedings.mlr.press/v27/bengio12a/bengio12a.pdf.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*. Retrieved from https://arxiv.org/pdf/1312.3005.

Chen, H., & He, B. (2013). Automated essay scoring by maximizing human-machine agreement. *Conference on Empirical Methods in Natural Language Processing*, 1741-1752. Retrieved from https://www.aclweb.org/anthology/D13-1180.

Crossley, S., Allen, L. K., Snow, E. L., & McNamara, D. S. (2015, March). Pssst... textual features... there is more to automatic essay scoring than just you!. *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, 203-207. doi: 10.1145/2723576.2723595

Dong, F., & Zhang, Y. (2016, November). Automatic features for essay scoring - An empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1072-1077. doi: 10.18653/v1/D16-1115

Farag, Y., Yannakoudakis, H., & Briscoe, T. (2018). Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv preprint arXiv*:1804.06898. doi: 10.1145/3318265.3318296

Farra, N., Somasundaran, S., & Burstein, J. (2015, June). Scoring persuasive essays using opinions and their targets. *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, 64-74. Retrieved from https://www.aclweb.org/anthology/W15-0608.

Hamner, B., & Shermis, M.D. (2012). Contrasting state-of-the-art automated scoring of essays: Analysis. *Handbook of Automated Essay Evaluation: Current Applications and New Directions*, 254-263. doi:10.4324/9780203122761

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Kingma, D.P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv 2014. *arXiv* preprint. arXiv:1412.6980.

Klebanov, B. B., & Flor, M. (2013). Word Association Profiles and their Use for Automated Scoring of Essays. *ACL*, 1148-1158. Retrieved from https://www.aclweb.org/anthology/P13-1113.pdf.

Lewis, R. J. (2000, May). An introduction to classification and regression tree (CART) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California* (Vol. 14).

McNamara, D. S., Crossley, S. A., Roscoe, R. D., Allen, L. K., & Dai, J. (2015). A hierarchical classification approach to automated essay scoring. *Assessing Writing, 23*, 35-59. doi: 10.1016/j.asw.2014.09.002

Nguyen, H., & Dery, L. (2018). Neural networks for automated essay grading. Retrieved from https://cs224d.stanford.edu/reports/huyenn.pdf.

Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing*, 1532-1543. doi: 10.3115/v1/D14-1162

Persing, I., & Ng, V. (2015, July). Modeling argument strength in student essays. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 543-552. doi: 10.3115/v1/P15-1053

Somasundaran, S., Burstein, J., & Chodorow, M. (2014, August). Lexical chaining for measuring discourse coherence quality in test-taker essays. *Proceedings of COLING 2014, the 25th International conference on computational linguistics: Technical papers*, 950-961. Retrieved from https://www.aclweb.org/anthology/C14-1090.

Taghipour, K., & Ng, H.T. (2016). A neural approach to automated essay scoring. *EMNLP*. Retrieved from https://aclweb.org/anthology/D16-1193.

Tay, Y., Phan, M. C., Tuan, L. A., & Hui, S. C. (2018, April). SkipFlow: Incorporating neural coherence features for end-to-end automatic text scoring. In *Thirty-Second AAAI Conference on Artificial Intelligence*. arXiv:1711.04981.

Vapnik, V. (2013). *The nature of statistical learning theory.* Springer science & business media.

Wang, Y., Wei, Z., Zhou, Y., & Huang, X. J. (2018). Automatic essay scoring incorporating rating schema via reinforcement learning. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 791-797. doi: 10.18653/v1/d18-1090

Yang, P., Sun, X., Li, W., & Ma, S. (2018). Automatic academic paper rating based on modularized hierarchical convolutional neural network. *CoRP, abs*/1805.03977. Retrieved from http://arxiv.org/abs/1805.03977.

Yannakoudakis, H., Briscoe, T., & Medlock, B. (2011, June). A new dataset and method for automatically grading ESOL texts. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 180-189. Retrieved from https://www.aclweb.org/anthology/P11-1019.

Zesch, T., Wojatzki, M., & Scholten-Akoun, D. (2015, June). Task-independent features for automated essay grading. *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, 224-232. doi: 10.3115/v1/W15-0626

Zhang, C., & Ma, Y. (Eds.). (2012). *Ensemble machine learning: methods and applications*. Springer Science & Business Media.

Appendix A

Predefined features for supervised learning models categorized as dictated by the methodology.

"Dependencies" are Python functions used to determine each feature.

No.	Feature	Description	Dependencies
Content			
1-3	High-, Medium-, Low- Scoring Similarity	TF-IDF similarity to hand-picked samples	SpaCy NLP, Similarity
4	Polarity	Degree of positive/negative sentiment	TextBlob
5	Subjectivity	Degree of emotional/opinionated rhetoric	TextBlob
Grammatical			
6	No. Corrections	Number of corrections made by spell & grammar check	LanguageTool
Semantic			
7	Non-Stop Word Count	Word count excluding commonly used words	SpaCy Stop Words
8	Difficult Word Count	Count of "difficult" words as determined by the TextStat corpus	SpaCy, TextStat
9-12	Unique Word, Unique Character, Unique NER, Unique POS Tag Count	Number of unique instances of various metrics	SpaCy
13	Common POS Count	Total of 10 most common POS tags	SpaCy
14	Action Word Count	Number of adjectives, nouns and verbs	SpaCy
15-27	Character, Word, Sentence, Noun, Adverb, Adjective, Determinative, Pronoun, Conjunction, Verb, Preposition, Proper Noun, Interjection Count	Count of various tokenized variables & POS tags	SpaCy
28-29	Period, Comma Placement	Rate of punctuation	SpaCy
30-32	Unredacted NER, Redacted NER, NER Count	NER counts determined by detected Name Entities and Name Entities redacted by the dataset	SpaCy
33-36	No. Quotations, Questions, Exclamations, Punctuation	Total punctuation	SpaCy
37-45	References to Locations, Persons, Organizations, Dates, Numbers, Percentages, Time, Money, Unrecognized Pronouns	Total number of various redacted Name Entities	SpaCy
Synthetic			
46-53	Coleman Lieu, Smog, Readability, Flesch, Dale Chall, Kincaid, Linsear, Gunning Fog Score	Various readability scores, calculated as a function of semantic features	TextStat

No.	Feature	Description	Dependencies
54-63	Unique POS-POS, Unique Character-Character, Word-Sentence, Unique Token-Token, Word-Character, Word-Difficult Word, Action Word-Word, Unique NER-NER, Redacted-Unredacted NER, Sentence-NER Ratio	Ratio between two semantic features, calculated through simple transformation	SpaCy

Appendix B

Hyperparameters used for all learning models. All hyperparameters not listed are were set to defaults.

Hyperparameter	Description	Value
Convolutional Layer		
Dimensionality	Matches GloVe dimensionality	300
Border Mode	Ensures CNN output retains its initial length	same
Window Size	Number of neighboring terms to collapse into each representation	3
Recursional Layers		
Dimensionality	Matches GloVe dimensionality	300
Dropout_W	Dropout rate for input neurons	0.5
Dropout_U	Dropout rate for recurrent neurons	0.1
Optimizer (Adam)		
Learning Rate	Step size in changing variables through backpropagation	0.001
Beta_1	Rate of decay for initial estimates	0.9
Beta_2	Rate of decay for second-moment estimates	1
Clipnorm	Clip gradient when the L2 norm exceeds this value	10
Clipvalue	Clip gradient when its value exceeds this value	0
Training		
Max Length	Max length of an input sequence	500
Seed	Condition for network initialization	1024
Dropout Probability	Probability of each neuron being ignored during training	0.2
Batch Size	Number of samples in one iteration	20
Epochs	Maximum number of training batches	100
Patience	Number of subsequent epochs of no improvement needed to cease training	5

Appendix C

Individual QWK scores for each fold. The values in Table 2 correspond to the averages of all folds for one prompt as denoted by the "Avg" row.

	1	2	3	4	5	6	7	8
Forest Ensemble								
Fold 1	0.842	0.691	0.637	0.696	0.780	0.718	0.773	0.649
Fold 2	0.838	0.676	0.616	0.670	0.757	0.703	0.739	0.691
Fold 3	0.835	0.680	0.630	0.730	0.789	0.701	0.765	0.706
Fold 4	0.821	0.650	0.645	0.704	0.764	0.673	0.789	0.734
Fold 5	0.835	0.680	0.641	0.739	0.786	0.671	0.757	0.729
Avg	0.826	0.683	0.618	0.696	0.784	0.683	0.740	0.678
Elastic Network								
Fold 1	0.833	0.624	0.629	0.723	0.782	0.670	0.784	0.710
Fold 2	0.824	0.661	0.649	0.669	0.776	0.688	0.752	0.710
Fold 3	0.838	0.668	0.661	0.685	0.801	0.680	0.758	0.676
Fold 4	0.829	0.592	0.693	0.711	0.753	0.666	0.781	0.778
Fold 5	0.816	0.618	0.665	0.722	0.780	0.659	0.744	0.740
Avg	0.819	0.673	0.643	0.699	0.779	0.663	0.739	0.717
Linear Regression								
Fold 1	0.775	0.631	0.604	0.673	0.750	0.680	0.691	0.653
Fold 2	0.749	0.533	0.620	0.692	0.753	0.625	0.664	0.562
Fold 3	0.723	0.626	0.600	0.584	0.722	0.653	0.718	0.668
Fold 4	0.774	0.610	0.627	0.539	0.741	0.652	0.696	0.630
Fold 5	0.760	0.620	0.601	0.682	0.738	0.641	0.610	0.572
Avg	0.796	0.653	0.610	0.735	0.741	0.650	0.676	0.617
k-Nearest Neighbor Classifier								
Fold 1	0.800	0.622	0.709	0.715	0.775	0.596	0.671	0.529
Fold 2	0.799	0.641	0.650	0.684	0.790	0.630	0.660	0.316
Fold 3	0.790	0.671	0.677	0.726	0.766	0.648	0.703	0.507
Fold 4	0.812	0.630	0.670	0.698	0.789	0.612	0.736	0.457
Fold 5	0.814	0.630	0.648	0.713	0.771	0.572	0.655	0.507
Avg	0.800	0.628	0.669	0.701	0.780	0.637	0.651	0.473
SVM								
Fold 1	0.724	0.625	0.667	0.660	0.739	0.649	0.645	0.551
Fold 2	0.713	0.634	0.632	0.728	0.758	0.641	0.668	0.415
Fold 3	0.692	0.571	0.642	0.680	0.733	0.657	0.656	0.478
Fold 4	0.697	0.608	0.651	0.734	0.775	0.644	0.672	0.475
Fold 5	0.740	0.591	0.613	0.690	0.735	0.653	0.627	0.448
Avg	0.646	0.605	0.649	0.676	0.747	0.602	0.623	0.479
RNN-CNN								
Fold 1	0.468	0.408	0.647	0.640	0.494	0.653	0.687	0.498
Fold 2	0.526	0.407	0.595	0.716	0.519	0.645	0.563	0.483
Fold 3	0.531	0.545	0.537	0.560	0.695	0.676	0.682	0.513

	1	2	3	4	5	6	7	8
Fold 4	0.477	0.510	0.500	0.637	0.563	0.614	0.650	0.522
Fold 5	0.528	0.357	0.592	0.591	0.515	0.664	0.571	0.478
Avg	0.506	0.445	0.574	0.629	0.557	0.650	0.631	0.499
				RNN				
Fold 1	0.016	0.647	-0.010	0.702	0.771	0.022	0.234	0.102
Fold 2	0.182	0.679	0.700	0.530	-0.012	0.130	0.669	0.010
Fold 3	0.020	0.631	0.701	0.710	0.009	0.768	0.681	0.318
Fold 4	0.779	0.004	0.401	0.490	0.000	0.331	0.689	0.573
Fold 5	0.663	0.566	0.518	0.732	0.021	0.778	0.332	0.360
Avg	0.332	0.506	0.462	0.633	0.158	0.406	0.521	0.273
				GRU				
Fold 1	0.030	0.331	0.271	0.170	0.000	0.213	0.556	0.216
Fold 2	0.000	0.255	0.010	0.340	0.130	0.000	0.533	-0.002
Fold 3	0.000	0.378	0.270	0.231	0.222	0.329	0.169	0.492
Fold 4	0.593	0.176	0.073	0.494	0.000	0.498	0.285	0.360
Fold 5	0.001	0.000	0.000	0.270	0.048	0.630	0.473	0.260
Avg	0.125	0.228	0.125	0.301	0.080	0.334	0.403	0.265
				GRU-CNN				
Fold 1	0.000	0.000	0.000	0.179	0.000	0.000	0.002	0.025
Fold 2	0.000	-0.002	0.000	0.000	0.207	0.000	0.131	0.018
Fold 3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.018
Fold 4	0.000	0.007	0.000	0.000	0.000	0.000	0.014	0.002
Fold 5	0.000	0.000	0.000	0.226	0.000	0.027	0.019	0.013
Avg	0.000	0.001	0.000	0.081	0.041	0.005	0.033	0.015

Appendix D

Features used by unsupervised models ordered by Gini-importance and categorized as dictated by the methodology.

Feature	Gini	STD	Feature	Gini	STD
Content			**Semantic**		
High-Scoring Similarity	2.16%	0.36%	Proper Noun Count	1.39%	0.15%
Medium-Scoring Similarity	1.73%	0.30%	References to Numbers	1.32%	0.11%
Low-Scoring Similarity	1.67%	0.12%	Redacted NER Count	1.24%	0.26%
Polarity	1.53%	0.16%	Ref. to Unrecognized Pronouns	1.08%	0.14%
Subjectivity	1.43%	0.12%	Interjection Count	0.97%	0.10%
Grammatical			No. Questions	0.79%	0.08%
No. Corrections	1.66%	0.11%	No. Exclamations	0.77%	0.09%
Semantic			References to Locations	0.69%	0.10%
Non-Stop Word Count	3.06%	2.02%	References to Persons	0.63%	0.11%
Unique Word Count	3.03%	1.83%	References to Organizations	0.51%	0.06%
Character Count	2.86%	1.53%	References to Dates	0.47%	0.10%
Word Count	2.73%	1.51%	References to Percentages	0.46%	0.23%
Difficult Word Count	2.55%	1.33%	References to Time	0.18%	0.06%
Determinative Count	2.36%	1.24%	References to Money	0.12%	0.03%
Common POS Count	2.36%	1.17%	**Synthetic**		
Action Word Count	2.31%	1.29%	Unique POS-POS Ratio	2.82%	1.87%
Sentence Count	2.21%	0.77%	Coleman Lieu Score	2.34%	0.38%
Noun Count	2.14%	0.84%	Unique Character-Character Ratio	1.80%	0.58%
Adverb Count	1.96%	0.77%	Smog Score	1.75%	0.62%
Adjective Count	1.95%	0.98%	Word-Sentence Ratio	1.74%	0.26%
Punctuation Count	1.90%	1.49%	Unique Token-Token Ratio	1.70%	0.80%
Pronoun Count	1.85%	0.64%	Word-Character Ratio	1.69%	0.24%
Conjunction Count	1.70%	0.74%	Word-Difficult Word Ratio	1.64%	0.24%
Unique Character Count	1.68%	0.64%	Action Word-Word Ratio	1.58%	0.21%
Verb Count	1.64%	0.78%	Readability Score	1.43%	0.15%
Preposition Count	1.63%	0.35%	Flesch Score	1.42%	0.12%
Unique NER Count	1.63%	0.31%	Dale Chall Score	1.38%	0.13%
Period Placement	1.63%	0.57%	Kincaid Score	1.38%	0.13%
No. Quotations	1.51%	0.41%	Linsear Score	1.29%	0.11%
Comma Placement	1.47%	0.29%	Unique NER-NER Ratio	1.29%	0.35%
Unique POS Tag Count	1.44%	0.18%	Redacted-Unredacted NER Ratio	1.28%	0.23%
NER Count	1.43%	0.30%	Gunning Fog Score	1.27%	0.16%
Unredacted NER Count	1.43%	0.46%	Sentence-NER Ratio	1.21%	0.12%