ACES: AUTOMATED ACADEMIC ESSAY SCORING USING A NATURAL LANGUAGE PROCESSING-BASED REGRESSION MECHANISM

Reinald Adrian Pugoy

Faculty of Information and Communication Studies
University of the Philippines Open University (PHILIPPINES)

Abstract

Academic essays are essential testing instruments that evaluate the students' ability to organize thoughts and synthesize information. However, grading them is an exhausting and cumbersome process that requires considerable manpower. It may be prone to errors, and there are also serious concerns about fairness, such that an essay graded B+ today may be graded B- tomorrow by the same checker. Therefore, the author proposes ACES, an essay scoring mechanism that employs natural language processing (NLP) to address the issue at hand. NLP is a sub-field of artificial intelligence (AI) concerned with granting computers the ability to understand texts in much the same way humans can. With essay scoring reformulated as a regression problem, ACES takes the essay answer as the input, converts it to a vector representation of numbers in the embedding space, and feeds it to the neural network model (which serves as the approximate regression function) to predict its score as the output. In this paper, the author successfully implements four versions of ACES that employ different embedding sources and neural network models, with the ACES variant that considers context and word frequency information performing the best (i.e., ACES-BERT).

Keywords: essay scoring, natural language processing, neural networks, regression

1 INTRODUCTION

An academic essay is said to be one of the most common forms of assessment [1]. Essays are designed to test the students' ability to organize thoughts and synthesize information, which pertains to the combination of different ideas into a single system. They are essential testing instruments for evaluating academic achievement and one's ability to integrate and recall ideas [2]. Performance in academic essay writing can be a significant source of data when making critical decisions about students [1, 3]. However, manually grading such essays is an exhausting and cumbersome process that requires considerable manpower [4]. In other words, it is time-consuming, energy-consuming, and mind-consuming. There are also serious concerns about fairness and equity wherein an essay graded B+ today might be graded B- tomorrow by the same checker [5].

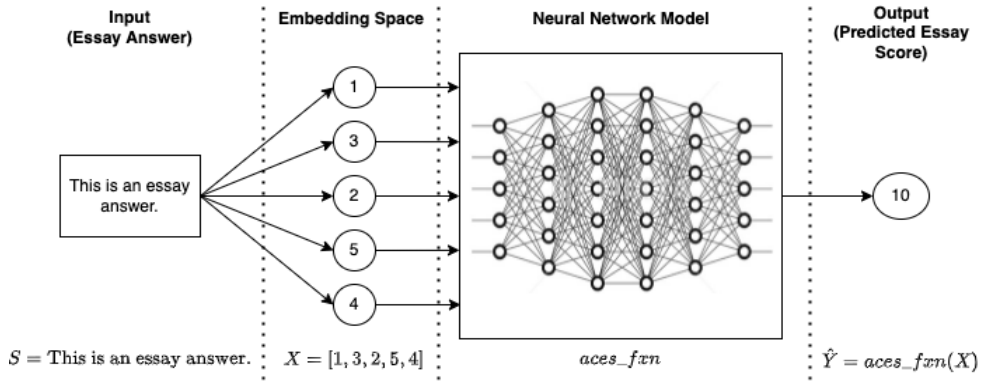

Figure 1. The proposed ACES architecture.

This has then attracted the attention of the research community specializing in natural language processing (NLP) due to its educational value and associated challenges. NLP is a sub-field of artificial intelligence (AI) concerned with granting computers the ability to understand texts in much the same way

humans can. Considered as the intersection of AI and linguistics, NLP enables computers to process human language in textual form to comprehend its whole meaning, complete with the writer's intent [6, 7]. Hence, employing NLP is a natural fit to tackle the aforementioned challenges of essay scoring. This has given rise to automated essay scoring (AES), defined as the task of employing computers to score written text without any human interference [4, 8]. AES, now an important educational application of NLP, is primarily motivated by the potential of reducing labor, providing cost-effective and efficient large-scale grading, and applying fair scoring criteria [9].

Therefore, in this paper, the author's primary objective is to develop an automated **ac**ademic **e**ssay **s**coring mechanism (called ACES) based on the principle of regression using a concise architecture. As illustrated in Figure 1, the proposed architecture is comprised of the following components:

1. The input (S), which takes the essay answer of the student.
2. The embedding space, which converts the essay text into a vector representation of real numbers (X) that the computer can understand and consequently process.
3. The neural network model, which learns or approximates the regression function ($aces\_fxn$) that maps the essay answer to its predicted score. In this study, four neural regression models have been implemented.
4. The output ($\hat{Y} = aces\_fxn(X)$), the predicted score of the essay answer.

It should be noted that neural networks (NN), computing systems inspired by the human brain, are capable of modeling complex patterns [10]. NN, considered state-of-the-art for a wide range of applications, learns by repeatedly examining individual data, generating its corresponding prediction, and making adjustments for incorrect predictions [11]. This also leads us to the author's second objective, which is to compare and contrast the performance of four variants of ACES (i.e, neural regression models) by employing various embedding sources and NN model types.

Nevertheless, the rest of this paper is organized as follows: Section 2 gives the review of related work. Section 3 contains the preliminaries and theoretical framework, which are used as foundational bases of our proposed methodology in Section 4. Section 5 shows our experimental details, such as the datasets used, evaluation metrics employed, and the discussion of this study's results. Finally, Section 6 highlights the conclusion and possible future work.

2 REVIEW OF RELATED WORK

Notable pre-neural network (or pre-deep learning) applications include e-rater [12] and Intelligent Essay Assessor [13]. In 2012, a competition for automated essay scoring called Automated Student Assessment Prize (ASAP) was organized by Kaggle and sponsored by the Hewlett Foundation [14]. The systems and models built during the competition heavily relied on handcrafted features. Researchers have also devoted significant time and effort to designing effective features for AES [15]. These include essay length [16], lexical complexity [12], and syntactic features [16]. In 2012 also, Mahana et al. [2] proposed a machine learning-based model that leveraged numerical features such as word count, word length, sentence count, and part-of-speech count. In 2015, McNamara et al. [17] employed a hierarchical classification approach with linguistic, semantic, and rhetorical features.

3 PRELIMINARIES AND THEORETICAL FRAMEWORK

3.1 Machine Learning and Neural Networks

Neural networks are under the AI domain of machine learning. According to Gartzman [18], to understand both NN and machine learning, we also have to understand the concept of classical or traditional programming. In classical programming, the programmer needs to know exactly and lay down the rules to determine the solution. For example, given a problem of differentiating squares and circles, we can write a program to detect corners. The rules may be set as follows: if the shape has no corners, it is a circle; on the other hand, if it has four corners, it is a square. In machine learning, the machine (computer) learns from samples (data). Using our previous example, we simply design a learning system that takes as many inputs of shapes (and their labels) as possible. Rules are not set; the machine should be able to determine the properties of squares and circles by itself based on the data we feed it.

In the context of NN, we take a look on its foundational blocks, i.e., neurons. The term *neuron* is a synonym for learning unit or function. In mathematics and computer science, a function, typically in the form of $y = f(x)$, has the following components:

- It takes something as an **input** (x).
- It applies some **logic** on the input $(f(x))$.
- It produces the **output** (y).

Hence, an NN is a network of numerous functions whose inputs and outputs are intertwined, with neurons feeding each other [18]. The considerations for designing an NN include modeling inputs and outputs, the type of functions for each neuron, and architecture (which output serves as an input to which neuron). The complex structure of NN allows it to learn complex hidden patterns [19]. This is made possible by showing it numerous samples (data) of correct inputs and outputs, expecting it to also give the correct output for a new sample the NN has never seen before. Typical NNs include multilayer perceptron (MLP) and convolutional neural network (CNN).

3.2 Natural Language Processing Embeddings

The prior section mentions that the input and the manner of its modeling is a design consideration for any NN. For any subsequent studies or problems involving textual inputs, modeling them properly into representations the computer can understand is particularly critical. Hence, embedding techniques (such as TF-IDF, fastText, and BERT) are the means by which words in a given text are expressed numerically and made meaningful for subsequent machine learning.

3.2.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a weighing scheme categorized as a statistical procedure [20]. TF-IDF computes the values for each word in a document, paragraph, or passage through an inverse proportion of the word's frequency in a particular document (TF) to the percentage of documents that the word appears in (IDF). Words with high TF-IDF scores imply a strong relationship with the document they appear in [20]. Given a set of m documents in $D = \{d_1, d_2, ..., d_m\}$, the following formulas are used to compute for TF, IDF, and TF-IDF:

$$TF_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{1}$$

$$IDF_i = log\frac{|D|}{|k : i \in d_k|} \tag{2}$$

$$TFIDF_{ij} = TF_{ij} \times IDF_i \tag{3}$$

where n_{ij} indicates the number of times that term i appears in document j. The denominator of TF_{ij} refers to the total number of terms that appear in j, while the denominator of IDF_i pertains to the number of documents in which term i appears.

3.2.2 Word Embeddings

Pre-trained continuous word representations, also known as word embeddings, are usually seen as an improvement over TF-IDF or any other bag-of-words-based models. Word embeddings have become the basic building blocks of numerous NLP and machine learning tasks [21]. Such embeddings provide distributional information about words that typically improve the models' generalization learned on a limited amount of data [22]. This information is commonly derived from statistics gathered from a large unlabeled corpus of text data, such as Wikipedia and Google News [23]. A pivotal aspect of the pre-training process is to efficiently capture as much statistical information as possible from vast data sources [21]. A standard approach for learning word representations, as implemented in fastText [24], is to train log-bilinear models based on the skip-gram architecture where nearby words are predicted given a source word. Individual words are then represented as real-valued vectors in a shared vector space. In summary, word embeddings provide dense representations of words that capture their meanings. At the same time, bag-of-words-based models result in sparse vectors that only keep track of word frequency information.

3.2.3 BERT

According to Pilehvar and Collados [25], word embeddings are hindered by their static nature because each word is associated with the same embedding regardless of the context it appears in. In other words, such embeddings cannot identify the dynamic nature of each word's semantics. For example, in the two previous embedding techniques, the sentence *"The kid eats the hamburger"* may be treated the same as *"The hamburger eats the kid"*. This is not the case in BERT as this drawback has been addressed by Devlin et al. [26]. BERT is a method for pre-training contextual language representations for transfer learning to multiple NLP tasks [27]. It has resulted in state-of-the-art results for a wide variety of NLP tasks. BERT operates over an input sequence of tokens $T = \{T_0, T_1, ..., T_j\}$, which is composed of vocabulary words and a set of special tokens `[SEP]`, `[CLS]`, and `[MASK]`. These tokens pass through a stack of Transformer encoders to produce their corresponding final representations $\{h_0, h_1, ..., h_j\}$. As the output of each encoder layer can be utilized as contextualized features, let $H^{(l)}$ be a matrix with rows $\{h_0^{(l)}, h_1^{(l)}, ..., h_j^{(l)}\}$ where l corresponds to a particular encoder layer.

During pre-training, BERT is trained end-to-end on a large, unlabeled plain text corpus for two unsupervised tasks: masked language modeling (MLM) and next sentence prediction (NSP). MLM enables BERT to be a truly deep bidirectional model such that each word is represented by both its left and right contexts, starting from the bottom of a deep neural network. This step is accomplished by randomly dividing input tokens into two disjoint sets: masked tokens set X_M and observed tokens set X_O, with approximately 15% of the tokens being masked [27]. Masked tokens are substituted with `[MASK]` 80% of the time, 10% are replaced with a random token, and 10% are left unchanged [28]. BERT then aims to reconstruct and predict these masked tokens using X_O by utilizing a linear layer that maps the final representation of each token (i.e., h_i) to a vocabulary distribution that is trained under cross-entropy loss. Furthermore, the other unsupervised task, NSP, allows BERT to understand sentence relationships that are not necessarily captured by MLM. A binarized NSP task, generated from any monolingual corpus, is pre-trained. Specifically, two text segments, A and B, are processed following this format: $\{\texttt{[CLS]}, T_{A1}, T_{A2}, ..., T_{Aj}, \texttt{[SEP]}, T_{B1}, T_{B2}, ..., T_{Bj}, \texttt{[SEP]}\}$. BERT then identifies whether B is the direct continuation of A in the source by employing a linear layer that operates on the final representation of `[CLS]` (i.e., $h_{\texttt{[CLS]}}$) trained under binary cross-entropy loss. The hidden state of the special `[CLS]` token can serve as the essay text's aggregate sequence representation, informal sentence vector, or pooled contextualized embedding [26, 29]. In theory, any encoder layer may be selected to provide the hidden state of `[CLS]` as the review's representation.

4 METHODOLOGY

In this section, we discuss the implementation of four variants of ACES, utilizing different embedding sources and NN models (i.e., regression functions). Before proceeding any further, it should be noted first that all models operate on the same input sequence, which pertains to the given essay text. This input sequence is represented by $S = \{S_1, S_2, ..., S_f\}$ composed of vocabulary words and tokens and is truncated or padded according to a given fixed length of f.

4.1 MLP using TF-IDF Scores (ACES-MLP)

Figure 2 illustrates the proposed model. Initially, TF-IDF scores are obtained for each token in S. It uses a given essay set to serve as the set of documents D that has vocabulary size v. These scores are represented by the sparse matrix $X = \{X_1, X_2, ..., X_f\}$, where $X \in \mathbb{R}^{f \times v}$. Adding a specified number of hidden layers on top of these TF-IDF scores can provide a certain degree of non-linearity and flexibility. Hence, the MLP leading to the predicted grade $\hat{Y}$ is given by the following equations:

$$\begin{aligned} h_1 &= ReLU(W_1 X + b_1) \\ h_2 &= ReLU(W_2 h_1 + b_2) \\ \hat{Y} &= W_3 h_2 + b_3 \end{aligned} \tag{4}$$

where W_1, W_2, and W_3 are the weight matrices and b_1, b_2, and b_3 are the bias vectors.

4.2 CNN using fastText Embeddings (ACES-CNN)

Figure 3 shows the ACES-CNN model with the matrix of word embeddings as its input. Corresponding vector representations for each token in S is obtained by utilizing fastText embeddings. These are given

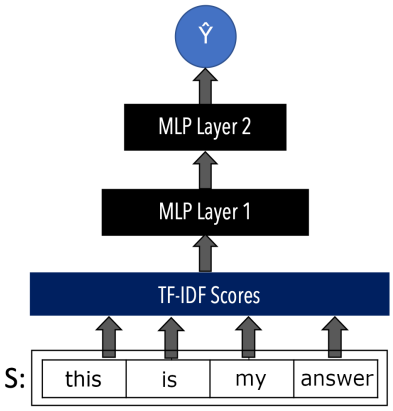

Figure 2. The ACES-MLP model.

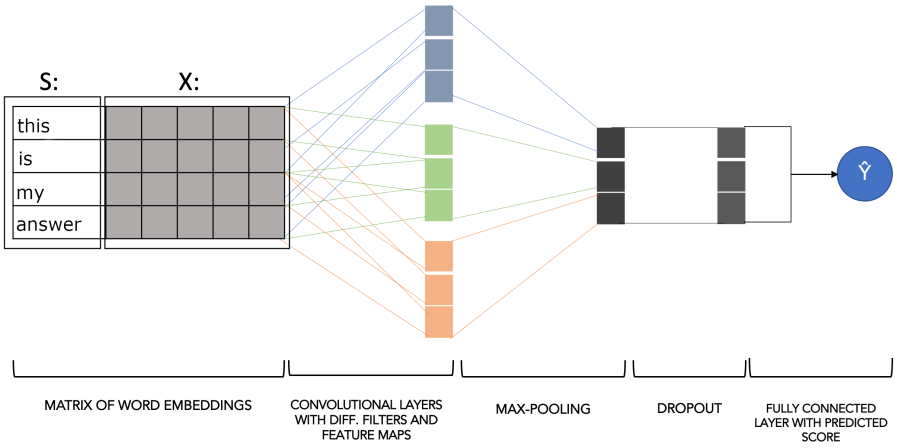

Figure 3. The ACES-CNN model.

by $X_{i:f} = X_1 \bigoplus X_2 \bigoplus ... \bigoplus X_f$, where $\bigoplus$ is the concatenation operator, word vector $X_i \in \mathbb{R}^k$, and k refers to the embedding dimension. Afterward, employing a CNN for a text classification or regression task involves computing the convolved features for all possible windows. A window can be defined as a segment in an essay text of a specified length. These possible windows of length h is represented as $\{X_{1:h}, X_{2:h+1}, ..., X_{f-h+1:f}\}$. Calculating a convolved feature for a particular window is given by the formula below:

$$C_i = ReLU(w^T X_{i:i+h-1} + b) \tag{5}$$

where the convolutional filter $w \in \mathbb{R}^{h \times k}$. Applying the same filter to all possible windows in $X_{i:f}$ then results in a feature map:

$$C = [C_1, C_2, ..., C_{f-h+1}] \in \mathbb{R}^{f-h+1} \tag{6}$$

To capture the most important feature and further reduce the dimensions, max-pooling is applied.

$$\hat{C} = max(C) \tag{7}$$

Assuming that there are m filters, the final feature vector is acquired by concatenating the max-pooled feature maps of the filters. Dropout is also applied to mitigate overfitting.

$$\begin{aligned} z &= [\hat{C}_1, \hat{C}_2, ..., \hat{C}_m] \\ z' &= dropout(z) \end{aligned} \tag{8}$$

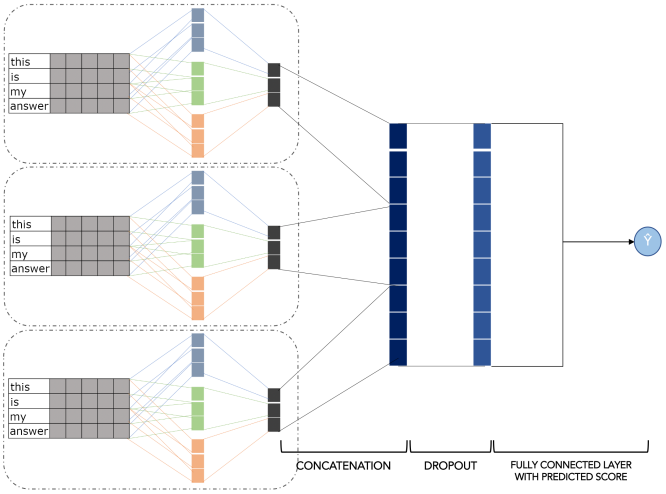

Figure 4. The ACES-3CNN model.

Finally, the feature vector z' is fed to a fully connected layer to obtain the predicted grade $\hat{Y}$:

$$\hat{Y} = Wz' + b \tag{9}$$

where W and b respectively refers to the weight matrix and bias vector.

4.3 Three Parallel CNNs using fastText Embeddings (ACES-3CNN)

Figure 4 presents ACES-3CNN, which largely draws inspiration from the ensemble and random forest methods in machine learning. In this regard, it employs three parallel CNN blocks primarily based on ACES-CNN. The three parallel CNNs produce their respective final feature vectors z_1, z_2, and z_3:

$$\begin{aligned} z_1 &= [\hat{C}_1, \hat{C}_2, ..., \hat{C}_{m_1}] \\ z_2 &= [\hat{C}_1, \hat{C}_2, ..., \hat{C}_{m_2}] \\ z_3 &= [\hat{C}_1, \hat{C}_2, ..., \hat{C}_{m_3}] \end{aligned} \tag{10}$$

where m_1, m_2, and m_3 are the number of filters for the parallel CNN blocks. After which, the three final feature vectors are concatenated together before dropout is applied. Concatenation is a necessary step to simulate the ensemble methods's majority voting scheme.

$$\begin{aligned} z &= [z_1, z_2, z_3] \\ z' &= dropout(z) \end{aligned} \tag{11}$$

The feature vector z' is likewise fed to a fully connected layer, and the predicted grade $\hat{Y}$ is obtained in the same manner described in Eq. 9.

4.4 Fine-tuned BERT Model (ACES-BERT)

The uncased, pre-trained $\text{BERT}_{\text{BASE}}$ model is used, which consists of 12 encoder layers, 12 self-attention heads, and a hidden layer size of 768. BERT processes the input sequence S differently when compared to the first three models. BERT requires every essay text to follow a particular format. For this purpose, it applies WordPiece tokenization to the said input sequence [30]. The format is comprised of token embeddings, segment embeddings, position embeddings, and padding masks. It also appends the `[CLS]` token at the beginning. The newly-formatted input sequence then passes through a stack of Transformer encoders to obtain the essay's contextualized representation. This representation comes from the hidden state of the special `[CLS]` token ($X = h_{\texttt{[CLS]}} \in \mathbb{R}^{768}$), also described as the essay's informal sentence vector or pooled embedding [26]. Finalizing the fine-tuning process, $h_{\texttt{[CLS]}}$ is fed to a final linear layer to compute the predicted grade:

$$\hat{Y} = W(h_{\texttt{[CLS]}}) + b \tag{12}$$

where W and b respectively refers to the weight matrix and bias vector.

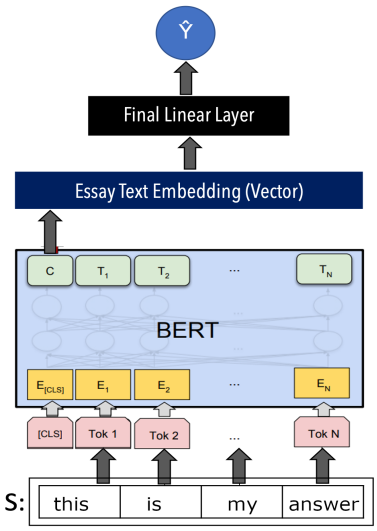

Figure 5. The ACES-BERT model.

4.5 Training

During training, the ACES variants adopt the same loss function, the mean squared error (MSE):

$$MSE = \frac{1}{|Tr|} \sum_{i \in Tr} (Y_i - \hat{Y}_i)^2 \tag{13}$$

where Tr refers to the training samples of a given essay set and Y_i is the ground-truth score given by the teacher to essay i. Moreover, the Adaptive Moment Estimation with weight decay (AdamW) [31] is employed to optimize the loss function. AdamW's advantage is its self-adapting nature of the learning rate during training. This makes the selection of a proper learning rate less cumbersome that consequently leads to faster convergence [32].

5 EXPERIMENTS

5.1 Dataset and Experimental Settings

The dataset used in this study's experiments is the same Kaggle dataset in the Automated Student Assessment Prize competition (ASAP) sponsored by the Hewlett Foundation [14]. The dataset has eight distinct essay sets (i.e., essay questions or topics), composed of nearly 12,976 English essays written by Grade 7 to Grade 10 students. Two checkers graded the essays, and it has been resolved that the average of the two graders' scores shall serve as the ground-truth. The dataset is divided as follows: 80% are reserved for training, while the remaining 20% are for testing. Multiple training and testing instances are performed separately for the essay sets. Without any special mention, for ACES-MLP, ACES-CNN, and ACES-3CNN, the number of epochs and the learning rate are fixed at 10 and 0.001, respectively. The dropout rate is 0.5. For ACES-BERT, the learning rate is in [1e-5, 2e-5], and the number of epochs is 3.

5.2 Evaluation Metrics

To evaluate and compare the respective performances of the ACES regression models, the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) are adopted:

$$RMSE = \sqrt{\frac{1}{|Ts|} \sum_{i \in Ts} (Y_i - \hat{Y}_i)^2} \tag{14}$$

$$MAE = \frac{1}{|Ts|} \sum_{i \in Ts} |Y_i - \hat{Y}_i| \tag{15}$$

where Ts refers to the test instances of a given essay set. While both RMSE and MAE express average model prediction errors, the former penalizes larger errors or deviations from the ground-truth.

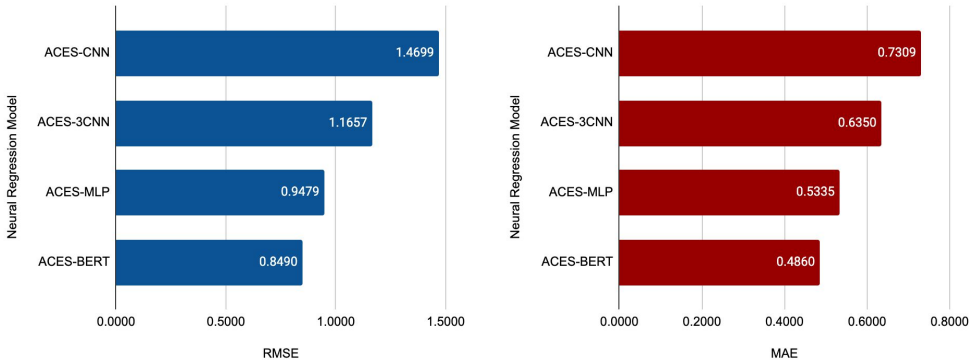

Figure 6. RMSE (left) and MAE (right) comparison of ACES variants.

Table 1: Predicted scores and ground-truth for a sample essay answer (as discussed in Section 5.3.2).

ACES Variant	Predicted Grade
ACES-MLP	3.68
ACES-CNN	3.86
ACES-3CNN	3.86
ACES-BERT	4.04
Ground-Truth	4.00

5.3 Results and Discussion

5.3.1 Performance Comparison

Figure 6 shows that ACES-BERT produces the best (lowest) MAE score of 0.4860. This is followed by ACES-MLP, which has an MAE value of 0.5335. The highest error values are obtained by ACES-3CNN and ACES-CNN, with MAEs of 0.6350 and 0.7309, respectively. The same trend is likewise observed for the models' RMSE values. The lowest RMSE score comes from ACES-BERT (at 0.8490), while the second lowest RMSE is ACES-MLP's (at 0.9479). Still, the CNN-based models have the highest error values, ACES-3CNN's RMSE is 1.1657, and ACES-CNN's is 1.4699. Both evaluation metrics show that ACES-BERT consistently outperforms the other variants of ACES.

These results demonstrate that context is indeed important and should always be considered in automated essay checking. After all, it is only BERT that can identify the dynamic nature of each word's semantic sense. Furthermore, ACES-MLP, which uses the supposedly simpler TF-IDF features, comes in second, while the CNN-based models are the worst-performing ones. This may be attributed to the loss of the word frequency information in CNN once max-pooling is applied. On the other hand, models based on BERT and TF-IDF still keep the word frequency information, which we argue to be crucial features. Finally, ACES-3CNN being better than ACES-CNN proves that three heads are better than one in CNN-based models. Simulating the ensemble or random forest behavior in CNN also have its fair share of benefits.

5.3.2 Per-Example Comparison

For further comparison, a specific example is chosen from the essay set. The instructions are given as follows:

> *Write a letter to your local newspaper in which you state your opinion on the effects computers have on people. Persuade the readers to agree with you.*

A student's essay answer is indicated below:

> *"Dear Local Newspaper, I have been using my computer for a long time now. I didn't have one growing up, but I got one last year. There has been people saying that the computer is good and it helps you*

stay in contact with friends and family it helps you learn about everything and you can look up the news. I totally agree with this. But some people don't like the computer because they spend more time on it then going outside. Here are my thoughts on the computer. First of all, teh computer is a great way to look up the news. You can find more news and weather faster then you can on the television. if you want to find the news, just got to a website that you think has news.

Table 1 shows the predicted grades produced by the ACES regression models. The ground-truth score is 4.00. The closest prediction is obtained by BERT (at 4.04), while the rest of the predicted grades are way lower than the ground-truth.

6 CONCLUSION AND FUTURE WORK

In this paper, the proposed essay scoring mechanism, ACES, has been successfully designed and developed using relevant principles from NLP and machine learning. With a concise architecture of *input-embedding space-NN model-output*, four variants are specifically implemented with different embedding sources and neural network regression models. The results also clearly illustrate that considering context and word frequency information and identifying the words' dynamic semantic sense matter in accurately grading essay items. In the future, it will be of great interest to further enhance regression models by incorporating concepts from unsupervised learning, which would not require labeled samples for training.

REFERENCES

[1] George A Brown, Joanna Bull, and Malcolm Pendlebury. *Assessing student learning in higher education*. Routledge, 2013.

[2] Manvi Mahana, Mishel Johns, and Ashwin Apte. "Automated essay grading using machine learning." In: *Mach. Learn. Session, Stanford University* (2012).

[3] R Freeman and R Lewis. "Planning and Implementing." In: *ASSESSMENT,, Kogan Page Ltd., London, UK* (1998).

[4] Yen-Yu Chen et al. "An unsupervised automated essay-scoring system." In: *IEEE Intelligent systems* 25.5 (2010), pp. 61–67.

[5] Brent Bridgeman, Catherine Trapani, and Yigal Attali. "Considering fairness and validity in evaluating automated scoring." In: *annual meeting of the National Council on Measurement in Education, San Diego, CA*. 2009.

[6] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction." In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.

[7] Julia Hirschberg and Christopher D Manning. "Advances in natural language processing." In: *Science* 349.6245 (2015), pp. 261–266.

[8] Zixuan Ke and Vincent Ng. "Automated Essay Scoring: A Survey of the State of the Art." In: *IJCAI*. 2019, pp. 6300–6308.

[9] Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. "Automatic text scoring using neural networks." In: *arXiv preprint arXiv:1606.04289* (2016).

[10] Sun-Chong Wang. "Artificial neural network." In: *Interdisciplinary computing in java programming*. Springer, 2003, pp. 81–100.

[11] Blaine Rister and Daniel L Rubin. "Piecewise convexity of artificial neural networks." In: *Neural Networks* 94 (2017), pp. 34–45.

[12] Yigal Attali and Jill Burstein. "Automated essay scoring with e-rater® v. 2.0." In: *ETS Research Report Series* 2004.2 (2004), pp. i–21.

[13] Peter W Foltz, Darrell Laham, and Thomas K Landauer. "The intelligent essay assessor: Applications to educational technology." In: *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning* 1.2 (1999), pp. 939–944.

[14] Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. "Flexible domain adaptation for automated essay scoring using correlated linear regression." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 431–439.

[15] Kaveh Taghipour and Hwee Tou Ng. “A neural approach to automated essay scoring.” In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pp. 1882–1891.

[16] Hongbo Chen and Ben He. “Automated essay scoring by maximizing human-machine agreement.” In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1741–1752.

[17] Danielle S McNamara et al. “A hierarchical classification approach to automated essay scoring.” In: *Assessing Writing* 23 (2015), pp. 35–59.

[18] Dalya Gartzman. *Neural networks for dummies: A quick intro to this fascinating field*. June 2019. URL: https://www.freecodecamp.org/news/neural-networks-for-dummies-a-quick-intro-to-this-fascinating-field-795b1705104a/.

[19] Jayanta Kumar Basu, Debnath Bhattacharyya, and Tai-hoon Kim. “Use of artificial neural network in pattern recognition.” In: *International journal of software engineering and its applications* 4.2 (2010).

[20] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries.” In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. New Jersey, USA. 2003, pp. 133–142.

[21] Tomas Mikolov et al. “Advances in pre-training distributed word representations.” In: *arXiv preprint arXiv:1712.09405* (2017).

[22] Ronan Collobert et al. “Natural language processing (almost) from scratch.” In: *Journal of machine learning research* 12.ARTICLE (2011), pp. 2493–2537.

[23] Scott Deerwester et al. “Indexing by latent semantic analysis.” In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[24] Piotr Bojanowski et al. “Enriching word vectors with subword information.” In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[25] Mohammad Taher Pilehvar and Jose Camacho-Collados. “WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 1267–1273.

[26] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding.” In: *arXiv preprint arXiv:1810.04805* (2018).

[27] Jiasen Lu et al. “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks.” In: *arXiv preprint arXiv:1908.02265* (2019).

[28] Mandar Joshi et al. “SpanBERT: Improving pre-training by representing and predicting spans.” In: *arXiv preprint arXiv:1907.10529* (2019).

[29] Chi Sun et al. “How to Fine-Tune BERT for Text Classification?” In: *arXiv preprint arXiv:1905.05583* (2019).

[30] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation.” In: *arXiv preprint arXiv:1609.08144* (2016).

[31] Ilya Loshchilov and Frank Hutter. “Fixing weight decay regularization in adam.” In: (2018).

[32] Chong Chen et al. “Neural attentional rating regression with review-level explanations.” In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 1583–1592.