Journal of Applied Testing Technology, Vol 23(Special Issue 1), 21-29, 2022

The Effect of Fine-tuned Word Embedding Techniques on the Accuracy of Automated Essay Scoring Systems Using Neural Networks

Tahereh Firoozi[1*], Okan Bulut[1], Carrie Demmans Epp[2], Ali Naeimabadi[2] and Denilson Barbosa[2]

[1]Centre for Research in Applied Measurement and Evaluation, University of Alberta, Edmonton, Canada; tahereh.firoozi@ualberta.ca
[2]Department of Computing Science, University of Alberta, Edmonton, Canada

Abstract

Automated Essay Scoring (AES) using neural networks has helped increase the accuracy and efficiency of scoring students' written tasks. Generally, the improved accuracy of neural network approaches has been attributed to the use of modern word embedding techniques. However, which word embedding techniques produce higher accuracy in AES systems with neural networks is still unclear. In addition, the importance of fine-tuned word embedding techniques on the accuracy of the AES systems is not justified yet. This study investigates the effect of fine-tuned modern word embedding techniques, including pretrained GloVe and Word2Vec, on the accuracy of a deep learning AES model using a Long-Short Term Memory (LSTM) network. The dataset used in this study consisted of 12,978 essays introduced in the 2012 Automated Scoring Assessment Prize (ASAP) competition. Results show that fine-tuned word embedding techniques could significantly improve the accuracy of the AES (QWK= 0.79) compared with the baseline model without pretrained embeddings (QWK = 0.73). Moreover, when used in AES, the pre-trained GloVe word embedding (QWK= 0.79) outperformed Word2Vec (QWK = 0.77). The results of this study can guide future AES studies in selecting more appropriate word representations and how to fine-tune the word embedding techniques for scoring-related tasks.

Keywords: Automated Essay Scoring, Glove Embedding, Neural Networks, Word Embeddings, Word2Vec

1. Introduction

A major educational assessment challenge is to assess students' written tasks reliably and efficiently. Human raters usually perform this task, but we need methods that scale more easily to large online courses or high-stakes test-taking settings where there may be thousands of submissions to grade. Automated Essay Scoring (AES) tries to solve this problem by developing computational models to assess students' written tasks automatically. Recently, with the advent of deep learning models and advanced tools for extracting linguistic features (Uto, Xie, & Ueno, 2020), the accuracy of AES systems has been significantly improved in a way that AES systems perform comparably to human raters in terms of their reliability and accuracy (Dong, Zhang, & Yang, 2017). Studies have shown that deep learning models, including Convolutional Neural Networks (CNNs) and Long-Short-Term Memory (LSTM), can achieve state-of-the-art accuracy in AES (Taghipour & Ng, 2016). Similar to promising improvements observed with deep learning, modern word embedding techniques, including pretrained GloVe embedding (Pennington, Socher, & Manning, 2014) and Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013), have helped improve the accuracy of neural networks in general. However, it is still not clear how fine-tuned word embedding techniques can improve the accuracy of deep learning AES models. Hence, this study investigated the effect of fine-tuned Word2Vec and GloVe embeddings on the accuracy of AES systems using neural networks. The questions which guided this study are:

1. Do LSTM models using GloVE and word2vec embedding techniques outperform a baseline LSTM model without word embedding techniques?
2. How does fine-tuning the word embeddings impact the performance of the AES systems?
3. How much does parallelization (i.e., number of workers) improve the speed of model training and memory usage?

1.2 Advances in Automated Essay Scoring

The application of AES to assess students' written tasks in educational assessments continues to grow as AES has become sufficiently reliable in various scoring tasks (Lottridge, Burkhardt, & Boyer, 2020; Uto, 2021; Uto, Xie, & Ueno, 2020). This increased reliability relied on advances in Natural Language Processing (NLP) and machine learning algorithms (Yan, Rupp, & Foltz, 2020). Advancements in NLP, such as the advent of linguistic tools to extract theory-based extensive linguistic features, including cohesion and coherence (Yang, Cao, Wen, Wu, & He, 2020), helped traditional machine learning-based AES models produce more accurate prediction results (Shin & Gierl, 2020). Advances in machine learning algorithms and the emergence of deep learning models, such as Long Short-Term Memory (LSTM), also enabled AES systems to achieve higher accuracy through learning the implicit representation of students' writing patterns without external human-designed feature engineering (Kumar & Boulanger, 2020; Rodriguez, Jafari & Ormerod, 2019; Lottridge, Godek, Jafari & Patel, 2021).

The Hewlett Foundation sponsored the Automated Scoring Assessment Prize (ASAP) to create a competition to demonstrate machine scoring capability for AES in 2012. The AES models proposed for the competition resulted in state-of-the-art reliability indices (QWK > 0.70). Although the competition is closed now, researchers can still use the dataset to improve the accuracy of AES systems. Various studies focused on learning better essay representation with neural network models using the same dataset (e.g., Alikaniotis, Yannakoudakis & Rei, 2016; Taghipour & Ng, 2016; Zhao, Zhang, Xiong, Botelho & Heffernan, 2017). Among these studies, Taghipour and Ng (2016) and Alikaniotis *et al.,* (2016) achieved state-of-the-art accuracy, QWK = 0.746 and QWK = 0.73, respectively, for their AES systems. They tried various recurrent layers for their model, including basic recurrent units (Elman, 1990), gated recurrent units (Chung, Gulcehre, Cho & Bengio, 2014), and LSTM, and the LSTM outperformed the others.

LSTM is a specific form of Recurrent Neural Networks (RNNs) that learns order dependence in sequence classification tasks. LSTM has feedback connections that process the entire sequence of data and learns the long- and short-term dependencies by using a cell state (i.e., long-term memory) in which the previous weight information of the input states is stored. While the forget gate (i.e., short-term memory) placed below the cell state is used to adjust the weights in the cell states based on the error rate, it receives in the recursive training process. The forward-backward (i.e., backpropagation) training process and the ability to handle long-term dependencies to consider the context in sequential text analysis caused the LSTM models to achieve state-of-the-art accuracy in the ASAP competition. Another significant feature of the systems that achieved state-of-the-art results using the ASAP dataset is the word embedding techniques used for text representation (Taghipour & Ng, 2016; Alikaniotis 3. 2016).

1.3 Text Representation Techniques

For computational text analysis, each text should be represented by a numerical vector in a vector space. Each text vector contains various dimensions consisting of numerical values assigned to each word as the basic elements in the text representation. So, for text vector representation, the numerical values (i.e., weights) of the words should be calculated. Generally, the techniques for text representation can be divided into four categories: one-hot-coding, frequency-based (e.g., TFIDF, LSA, and LDA) (e.g., Landauer, Foltz & Laham, 1998), word embedding (e.g., Word2Vec and GloVe) (e.g., Mikolov *et al.,* 2013), and contextual embedding (Elmo, GPT-2, and BERT) (e.g., Birunda & Devi, 2021; Devlin, Chang, Lee, & Toutanova, 2018).

One of the elementary techniques for vector representation is one-hot encoding, where each word is given one dimension using a binary value (1 or 0) that indicates the presence or absence of the word. The main limitation of one-hot encoding is that it is computationally expensive when dealing with a large amount of text data, as each text representation requires millions of dimensions for sparse word representations. Frequency-based techniques, such as TFIDF and Bag of Words (BoW), tried to solve the problem with sparse vectors by considering

the frequency of the words in vector representation rather than their presence or absence in a text (Araujo, Golo, Viana, Sanches, Romero & Marcacini, 2020). Although these text representation techniques created the possibility of using machine learning algorithms for automated text analysis, the meaning of words is not considered in calculating word vectors in text representation.

Modern word embedding techniques solved the limitation of the conventional frequency-based models by considering semanticity (i.e., the meaning or sense of the words in a context that the word is used) in calculating vector representations. For example, the word "bank" has different meanings in these contexts: "Bank of England", "memory banks", and "river bank." When the context of the word is not considered, the numerical representation (i.e., feature vectors) might not be calculated accurately, leading to the misrepresentation of the essays (i.e., data points) in feature space. Word embedding techniques are unsupervised learning algorithms that take a corpus of words and transform them into high-dimensional vectors in a meaningful manner that semantically relevant words are clustered together. In word embedding techniques, each word is represented with an information-rich dense vector containing ten or hundred dimensions, decreasing the computational cost in large volumes of text data (Pennington *et al.,* 2014). The popular word embedding techniques are GloVe and Word2Vec. There are two versions of Word2Vec — Continuous Bag of Words (CBOW) and Skip-Gram. The CBOW model learns the embedding by predicting the probability of each word based on its surrounding words or context constrained to the window size specified before model training. The Skip-Gram model learns by predicting the probability of the context of a word--gram neighboring words-based on a set of words. Word2Vec tries to capture the co-occurrence of words one window at a time.

The GloVe model combines the matrix factorization methods (Cai, He, Wang, Bao, & Han, 2009) and the window-based methods to consider both the statistical and contextual information of words in calculating word vectors. Hence, the GloVe learns the embeddings based on a co-occurrence matrix showing the count of the overall statistics of how often words appear together in a text based on their semantic similarity. The vector spaces of the regular word embeddings can be trained on the dataset of the target task. However, given that the pretrained word vectors with a high number of parameters improve model robustness and uncertainty estimates, they are often preferred over the regular word embedding techniques (Hendrycks, Lee & Mazeika, 2019). When vector spaces are pre-trained on massive corpora, it is called a pre-trained word vector representation. The pretrained word vectors can also be downloaded for free and mapped to the word vectors of the target task. GloVe model was trained on five corpora, Such as Gigaword, Wikipedia, and Twitter with about 55 billion tokens and 400,000 most frequent words (Pennington *et al.,* 2014). Hence, the GloVe pretrained model contains rich statistical and contextual information about the most frequent words used as embeddings for various text classification tasks. Research showed that word embedding models outperform conventional embeddings in NLP tasks, such as text detection (e.g., Gao, He, Zhang & Xia, 2017), sentiment analysis (e.g., Jang, Kim, Harerimana, Kang & Kim, 2020), text summarization (e.g., Haider, Hossin, Mahi & Arif, 2020), and feature selection (e.g., Wu, Li, Guo, Wang, Ren, Wang & Yang, 2022; Tian, Li & Li, 2018).

Recently, contextual embedding methods, such as BERT (Devlin, Chang, Lee & Toutanova, 2018), outperformed the current word embedding techniques by showing state-of-the-art accuracy in text classification tasks (e.g., Fernandez, Gosh, Liu, Wang, Choffin, Baraniuk & Lan, 2022; Ormerod, Malhotra & Jafari, 2021). Contextual-based embeddings learn the context of the words using transformers containing an attention mechanism that enables the model to learn the global dependency of the text using the Next Sentence Prediction (NSP) and the Mask Language Model (MLM) training strategies. In addition, the BERT learns the sequence of words in a text bi-directionally (left and right sides). The attention mechanism and bidirectional learning of the text are two unique advantages of transformer-based language methods to capture the contextual information of words in a text at the global level (beyond the neighboring words). Although contextual embedding methods showed promising results in text classification tasks[1], such as AES, this study focuses on word embedding techniques for text representation. In practice, the computational cost of transformer models such as BERT may not be justified unless its unique features provide benefits to the tasks such as transfer learning and multilingual AES, which word embedding models cannot solve. Hence, for supervised

[1]https://github.com/NAEP-AS-Challenge/info/blob/main/results.md

text classification, such word embedding techniques can provide the same benefits with less computational cost (Mayfield & Black, 2020).

Although the advantages of word embedding techniques on text classification tasks are justified, it is still unclear how fine-tuned word embedding techniques can affect the accuracy of the AES systems using neural network models. Thus, this study investigated the effect of fine-tuned word vector representations on improving AES neural networks. To this aim, we used pretrained GloVe and Word2Vec embeddings and examined how fine-tuning their hyperparameters affects the accuracy of a two-layer LSTM model.

2. Methods

2.1 Data

The dataset used in this study was introduced in the ASAP competition organized by Kaggle in 2012[2]. The data contains eight essay prompts that are accompanied by many essay responses. The average length of each essay is between 350 to 650 words, and the possible score ranges across each prompt (Table 1). Two human raters scored each essay, and the maximum of the two scores was announced as the final score for the essays. When there was a significant discrepancy between the scores of the two raters, the essay was scored by a third human rater to increase the scoring reliability. In order to use all the prompts together, in this study, we normalized the scores to have values between 0 and 1. The data was divided into the train (60%), validation (20%), and test (20%) sets. The train set was used to train embeddings and LSTM models, the validation set was used for hyperparameter tunning of the trained models, and the test set was used to see the result of the final model on the unseen dataset.

2.2 Data Analysis

Data analysis was divided into three steps: data processing and embedding, model development, and model evaluation. Preprocessing was performed in Python 3.6 using the NLTK library. All words were converted into lowercase and lemmatized (Bird, Klein & Loper, 2009). Non-alphabetic words and numbers were removed. Punctuation was kept and treated as separate words. This cleaned data was then tokenized at the word level. Each token was assigned a unique numeric index so that the index matched the location of the word in an embedding matrix. In addition, the embedding weight matrices were also constructed for the unique words in the essay sets using Word2Vec and GloVe.

Table 1. Descriptive statistics of the ASAP dataset

Prompt	Number of Essays	Number of Words		Scores		
		Mean	SD	Range	Mean	SD
1	1783	350	2.01	2-12	8.53	1.53
2	1800	350	1.73	1-6	3.41	0.77
3	1726	150	0.81	0-3	1.84	0.81
4	1772	150	0.96	0-3	1.43	0.93
5	1805	150	0.86	0-4	2.41	0.96
6	1800	150	0.86	0-4	2.72	0.96
7	1569	250	2.37	0-30	16.07	4.57
8	723	650	0.93	0-60	36.98	5.66

2.2.1 Model Architecture

A two-layer LSTM neural network was developed using Tensorflow and Keras. This LSTM was used as our primary prediction system. The model's input was the word embedded matrix, and the hidden layer was a two-layer neural LSTM to learn the features end-to-end. The initial embedding layer served as a lookup table to map the input tokens into word vectors of different dimensions. The output was the predicted score for the essays in the range of 0 to 1. Figure 1 shows a conceptual representation of our model using the embedding techniques. The baseline model against which we compared the performance of the word embedding techniques is similar to the model architecture in Figure 1, except for the embedding layer. In the Baseline model, rather than GloVe and Word2Vec, the word frequencies were used for text representations.

[2]https://www.kaggle.com/c/asap-aes/data

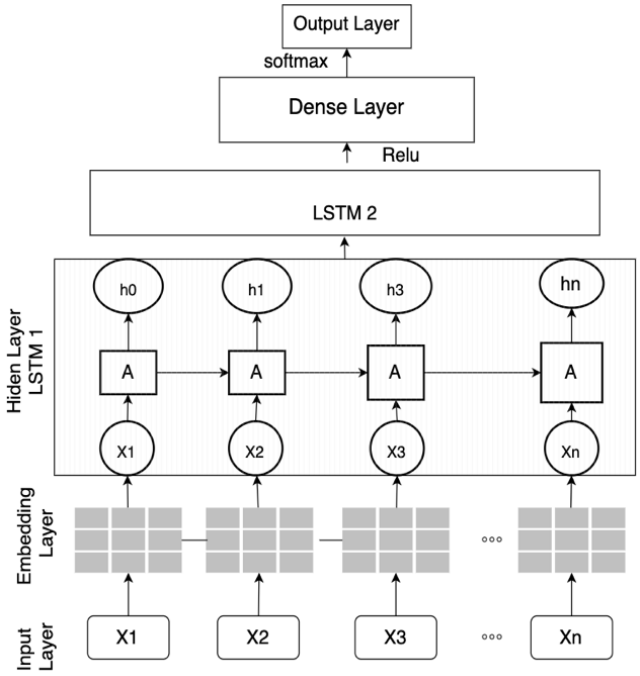

Figure 1. Conceptual representation of the AES model.

In this study, the LSTM model takes the input feature embedded matrix. The model comprises a sequential input layer corresponding to the number of words in each essay, followed by two LSTM layers (the recommended number of layers for text classification) and a dense layer. Feature extraction was conducted at the LSTM layers with a pooling layer to down-sample the feature maps and compress the dimensions. We used max pooling- a function that calculates the maximum value for patches of a feature map- in our model because research showed that max pooling performs the best among other pooling functions in classification tasks using the LSTM model (Kao, Sun, Wang & Wang, 2020). The number of hidden units after the pooling layer was 64. To prevent model overfitting, we followed the optimal dropout rate of 50% (q = 0.5) (e.g., Baldi, Pierre, Peter & Sadowski, 2013; Pauls & Yoder, 2018) in the hidden nodes for each epoch. A non-linear activation function (Relu) is applied to the model to introduce some nonlinearity in the model learning. Relu was used over the softplus and sigmoid functions to reduce the computational time (Pauls & Yoder, 2018). The output dimension of the dense layer before the scoring layer was 32. The softmax activation function was used at the scoring layer because we considered the task as a regression problem. Hence the softmax function converted the output of the dense layer (a vector of real numbers) to the original continuous scores [0,1] after normalization. Our objective was to minimize the square error between predicted and actual scores. The training was conducted at 10 epochs and a batch size of 64.

2.3 Evaluation Framework

We adopted a Quadratic Weighted Kappa (QWK) score for model evaluation to measure performance accuracy (Williamson, Xi & Breyer, 2012). Given that the original kappa coefficient assumed nominal categories, QWK was later introduced to extend the kappa coefficient to non-nominal categories through weighting. The idea behind QWK is that the misclassifications are also given partial credit (weight) based on how close they are to the correct class. QWK was the official agreement measure in the ASAP competition, where the dataset of the current study originated. Also, most studies that developed AES systems using the competition dataset reported kappa as their main evaluation criteria.

2.4 Tuning Process

After training the built model using the training set (n = 7,768), we used the evaluation set (n = 2,595) to tune the hyperparameters of the embeddings to improve the performance and efficiency of the model. The best hyperparameters were selected for the model based on the model accuracy (QWK score) after each epoch. The initial settings were adopted from similar models available on GitHub (Alikaniotis *et al.,* 2016). In order to decide on the efficacy of the embedding models in terms of the used memory space and the processing time, three more settings (i.e., time, number of workers, and used memory) were also included in this study. In addition, throughout the procedure, we tuned one parameter at a time while holding the others constant.

3. Results

3.1 Hyperparameter Tunning: Word2Vec Model

Table 2 shows the settings of the Pretrained Word2Vec approach. The model variables for tuning Word2Vec include the number of features, minimum word count, number of workers, window (context), and the algorithm employed to learn the embedding features. Also,

Word2Vec can use the Continuous Bag of Words (CBOW) or skip-gram methods, represented as algorithms 0 and 1, respectively. The results showed that compared with the initial settings, the optimized settings after the fine-tuning process increased the QWK for the Word2Vec model from 0.73 to 0.76.

Table 2. Settings of the pre-trained Word2Vec model

Word2Vec model variables	Initial setting	Optimized setting after tuning procedure
Number of features	300	300
Minimum word count	40	20
Context	10	20
Algorithm	0	1
Time(s)	608	620
Used memory (Gigabytes)	3.55	1.15
Number of workers	4	6
QWK	0.73	0.76

The number of features is the size of the vector space specified as part of the pretrained GloVe and Word2Vec models. In this study, the number of features was kept to 300, which can be considered a reasonably large value since it resulted in high accuracy in previous studies (Dong *et al.,* 2017). The minimum word count for training is the threshold for ignoring the words with fewer total frequencies in the context. Figure 2 shows the influence of minimum word count on QWK.

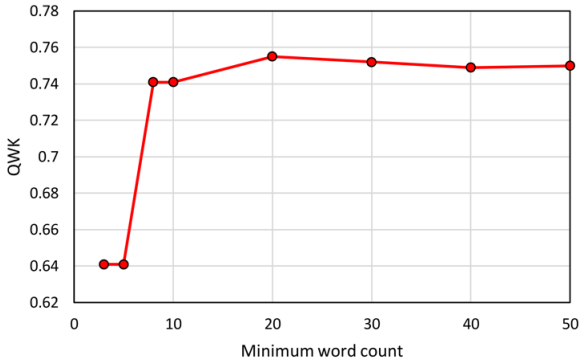

Figure 2. Effect of minimum word count on QWK of Word2Vec model.

Initially, the minimum word count was set to 40. Increasing the minimum word count to 50 (i.e., the highest possible value) resulted in a negligible increase in the model's accuracy. However, changing it to lower values like 5 (as an extreme case) led to high variance and overfitting, leading to a lowered score from 0.741 to 0.641. In addition, changing the minimum word count to 20 raised QWK to 0.755. Therefore, the optimized minimum word count was set to 20. Another parameter in tuning Word2Vec is the window size. When the window size increases, more context can be captured for estimating the weights of words. However, large window sizes can also decrease the quality of model training (Levy & Goldberg, 2014). Thus, we carefully increased the window size of the initial setting from 10 to 15, 20, and then 25. The optimal result was achieved when the window size was set to 20. The algorithm in Table 2 represents CBOW (0) and skip-gram (1). The results showed that QWK for CBOW and skip-gram methods were 0.73 and 0.74, respectively. Therefore, only the skip-gram algorithm was used for the rest of the computations of the Word2Vec model.

Time, used memory, and the number of workers in Table 2 represent the time and computational cost of the model using the set parameters. Generally, an increase in the number of workers results in a faster training process. In this study, we could increase the number of workers from 4 to 6. As Table 2 shows, the set parameters for Word2Vec increased the model training time and decreased the used memory.

3.2 Hyperparameter Tunning: GloVe Embedding

Results of fined tunning parameters, including the number of components (features), number of threads, window, learning rate, and epochs for the pretrained GloVe model, are represented in Table 3. This tuning procedure increased the accuracy of the GloVe embedding AES model from 0.76 to 0.79 (i.e., a 3% increase in prediction accuracy).

As Table 3 indicates, the initial setting for the number of components (i.e., 300 dimensions GloVe) resulted in the best accuracy for the model. The impact of window size on QWK in GloVe was similar to that of Word2Vec. GloVe had the highest QWK value when the window was set to 20. Furthermore, as Figure 3 depicts, raising the learning rate from 0.05 to 0.15 is advantageous in terms of both computational cost and accuracy. Hence, the optimum learning rate was set at 0.15. Based on the computational cost, the time, the number of threads, and memory usage,

the results show that the optimized settings for GloVe are more efficient than the initial settings.

Table 3. Settings for GloVe word embedding

GloVe model variables	Initial setting	Optimized setting after tuning procedure
Number of components	300	300
Window	10	20
Learning rate	0.05	0.15
Epochs	10	10
Time(s)	885	414
Number of threads	6	6
Used memory (Gigabytes)	3.59	1.51
QWK	0.76	0.79

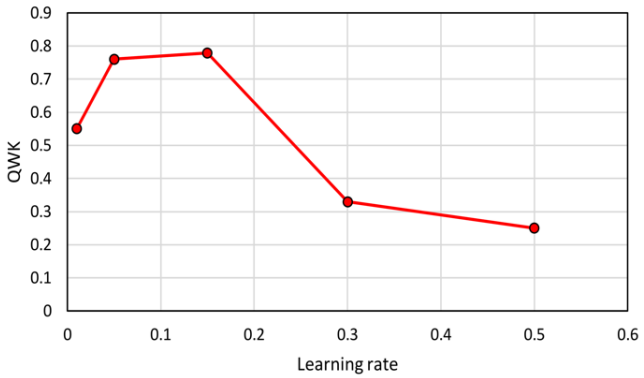

Figure 3. The effect of learning rate on QWK for GloVe word embedding.

3.3 Models Performance

The results of different word embedding techniques incorporated into the LSTM model are summarized in Table 4. As Table 4 shows, the word embedding techniques improved the accuracy of the baseline LSTM model. More specifically, the word2Vec model before fine tuning resulted in the lowest improvement, and fine tunned GloVe resulted in the highest improvement (6%) in the model's accuracy. The GloVe model before fine tuning and the Fine Tunned Word2Vec performed the same (QWK = 0.77) in increasing the accuracy of the baseline model.

Table 4. Accuracy of the LSTM model using word embeddings

Embedding techniques	QWK	Accuracy improvement (%)
LSTM (Baseline model without embedding technique)	0.73	–
Word2Vec (skip-gram) (before Fine Tuning)	0.75	2
GloVe (before Fine Tunning)	0.77	4
Fine Tuned Word2Vec	0.77	4
Fine Tuned Glove	0.79	6

4. Discussion

This study addressed the research question of whether fine-tuned modern word embedding techniques could impact the accuracy of deep learning AES models. The research question was introduced to understand whether the combination of recent advances in NLP techniques and deep learning algorithms can produce more accurate AES systems. In line with the result of various shared task studies (Dong *et al.,* 2017; Zhao *et al.*, 2017), this study indicated that pre-trained word embedding techniques could significantly improve the accuracy of AES neural network models. Results showed that when word embeddings are fine-tuned, they can not only improve the accuracy of the AES systems, but they also help increase the computational efficacy of the models in terms of training time. Inconsistent with Dong *et al.* (2017), our study showed a tradeoff between accuracy and efficacy in making decisions for the number of pre-trained dimensions (50 or 300).

This study adds to the AES literature by showing the importance of word embedding techniques in the accuracy of AES models. Our results showed that GloVe word embedding outperformed Word2Vec in improving the accuracy of AES systems. However, the inconsistency of this result with that of other studies (Pickard, 2020; Salehi, Cook, & Baldwin, 2015) shows that the superiority of these two embedding techniques can depend on the task. For example, in essay tasks where the context of use is an important factor, GloVe can outperform Word2Vec. In contrast, in predicting the semantic compositionality of multiword expressions where the context of use is not a determining factor, Word2Vec showed superiority over GloVe embeddings (Pickard, 2020). Overall, our results

can guide future AES studies in selecting more efficient word representations for their models. Moreover, this study's tuned parameters for word embeddings can be used as initial settings for future similar studies.

Although the results of this study show the performance of fine-tuned word embedding techniques in predicting essay scores, further studies are needed to investigate the accuracy of the AES systems when other advances in NLP, including tools that extract linguistic features that are known to be important to writing quality (Uto, Xie, & Ueno, 2020), are jointly used with word embedding techniques as part of deep learning models. Moreover, given the importance of context on essay tasks, future studies can also investigate the effect of other contextualized word embedding techniques, such as BERT (Devlin *et al.,* 2018), on the accuracy of AES systems.

This study used all eight essay prompts in the ASAP dataset. It is also possible to explore and compare the performance of the word embedding techniques for each prompt separately. This future investigation would allow researchers to see whether word embedding techniques perform differently on texts with different attributes. Given the variance in expectations across writing styles, future studies can compare the effect of different word embedding techniques on model performance for other writing genres, including persuasive, argumentative, and narrative texts.

5. References

Alikaniotis, D., Yannakoudakis, H. & Rei, M. (2016). Automatic text scoring using neural networks. arXiv preprint arXiv:1606.04289. https://doi.org/10.18653/v1/P16-1068

Araujo, A., Golo, M., Viana, B., Sanches, F., Romero, R. & Marcacini, R. (2020, October). From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering. *In Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional* (pp. 378-389). SBC. https://doi.org/10.5753/eniac.2020.12144

Baldi, Pierre, and Peter J. Sadowski (2013). Understanding dropout. *Advances in Neural Information Processing Systems*.

Bird, S., Klein, E. & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.

Birunda, S. & Devi, R. K. (2021). A review on word embedding techniques for text classification. *In Innovative Data Communication Technologies and Application*. Springer, 267-281. https://doi.org/10.1007/978-981-15-9651-3_23

Cai, D., He, X., Wang, X., Bao, H., & Han, J. (2009, June). Locality preserving nonnegative matrix factorization. In 21st International Joint Conference on Artificial Intelligence.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dong, F., Zhang, Y. and Yang, J. (2017, August). Attention-based recurrent convolutional neural network for automatic essay scoring. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017) (pp. 153-162). https://doi.org/10.18653/v1/K17-1017

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179-211. https://doi.org/10.1207/s15516709cog1402_1

Fernandez, N., Ghosh, A., Liu, N., Wang, Z., Choffin, B., Baraniuk, R. & Lan, A. (2022). Automated Scoring for Reading Comprehension via In-context BERT Tuning. arXiv preprint arXiv:2205.09864. https://doi.org/10.1007/978-3-031-11644-5_69

Gao, J., He, Y., Zhang, X. & Xia, Y. (2017, November). Duplicate short text detection based on Word2vec. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 33-37). IEEE. https://doi.org/10.1109/ICSESS.2017.8342858

Haider, M. M., Hossin, M. A., Mahi, H. R. & Arif, H. (2020, June). Automatic text summarization using gensim word2vec and k-means clustering algorithm. In 2020 IEEE Region 10 Symposium (TENSYMP) (pp. 283-286). IEEE https://doi.org/10.1109/TENSYMP50017.2020.9230670

Hendrycks, D., Lee, K. & Mazeika, M. (2019, May). Using pre-training can improve model robustness and uncertainty. In International Conference on Machine Learning (pp. 2712-2721). PMLR.

Jang, B., Kim, M., Harerimana, G., Kang, S. U. & Kim, J. W. (2020). Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. *Applied Sciences*, 10(17), 5841. https://doi.org/10.3390/app10175841

Kao, C. C., Sun, M., Wang, W., & Wang, C. (2020, May). A comparison of pooling methods on LSTM models for rare acoustic event classification. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 316-320). https://doi.org/10.1109/ICASSP40776.2020.9053150

Kumar, V., & Boulanger, D. (2020, October). Explainable automated essay scoring: Deep learning really has pedagogical value. In Frontiers in education (Vol. 5, p. 572367). Frontiers Media SA. https://doi.org/10.3389/feduc.2020.572367

Landauer, T. K., Foltz, P. W. & Laham, D. (1998). An introduction to latent seman ic analysis. *Discourse Processes*, 25(2-3), 259-284. https://doi.org/10.1080/01638539809545028

Lottridge, S., Godek, B., Jafari, A., & Patel, M. (2021). Comparing the robustness of deep learning and classical automated scoring approaches to gaming strategies. Technical report, Cambium Assessment Inc.

Lottridge, S., Burkhardt, A. & Boyer, M. (2020). Digital module 18: Automated scoring https://ncme.elevate.commpartners.com. *Educational Measurement: Issues and Practice*, 39(3), 141-142. https://doi.org/10.1111/emip.12388

Levy, O. & Goldberg, Y. (2014, June). Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 302-308). https://doi.org/10.3115/v1/P14-2050 PMid:25270273

Mayfield, E. & Black, A. W. (2020, July). Should you fine-tune BERT for automated essay scoring? In Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications (pp. 151-162). https://doi.org/10.18653/v1/2020.bea-1.15

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv. https://doi.org/10.48550/arXiv.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

Ormerod, C., Malhotra, A. & Jafari, A. (2021). Automated essay scoring using efficient transformer-based language models. rXiv preprint: https://arxiv.org/abs/2102.13136

Pennington, J., Socher, R. & Manning, C.D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on Empirical Methods In Natural Language Processing (EMNLP) (pp. 1532-1543). https://doi.org/10.3115/v1/D14-1162

Pickard, T. (2020, December). Comparing word2vec and GloVe for automatic measurement of MWE compositionality. In Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons (pp. 95-100).

Pauls, A. & Yoder, J. (2018) Determining optimum drop-out rate for neural networks. Midwest Instructional Computing Symposium (MICS).

Rodriguez, P. U., Jafari, A. & Ormerod, C. M. (2019). Language models and automated essay scoring. arXiv preprint arXiv:1909.09482.

Salehi, B., Cook, P. & Baldwin, T. (2015). A word embedding approach to predicting the compositionality of multiword expressions. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 977-983). https://doi.org/10.3115/v1/N15-1099

Shin, J. & Gierl, M.J. (2020). More efficient processes for creating automated essay scoring frameworks: A demonstration of two algorithms. Language Testing, p. 0265532220937830. https://doi.org/10.1177/0265532220937830

Taghipour, K. & Ng, H.T. (2016, November). A neural approach to automated essay scoring. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 1882-1891). https://doi.org/10.18653/v1/D16-1193

Tian, W., Li, J. & Li, H. (2018, July). A method of feature selection based on Word2Vec in text categorization. In 2018 37th Chinese Control Conference (CCC) (pp. 9452-9455). IEEE. https://doi.org/10.23919/ChiCC.2018.8483345

Uto, M. (2021). A review of deep-neural automated essay scoring models. *Behavior Metrika*, 48(2), 459-484. https://doi.org/10.1007/s41237-021-00142-y

Uto, M., Xie, Y. & Ueno, M. (2020). Neural Automated Essay Scoring Incorporating Handcrafted Features. In Proceedings of the 28th International Conference on Computational Linguistics, pages 6077-6088, Barcelona, Spain (Online). International Committee on Computational Linguistics https://doi.org/10.18653/v1/2020.coling-main.535

Williamson, D. M., Xi, X. & Breyer, F. J. (2012). A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice*, 31(1), 2-13. https://doi.org/10.1111/j.1745-3992.2011.00223.x

Wu, C., Li, X., Guo, Y., Wang, J., Ren, Z., Wang, M. & Yang, Z. (2022). Natural language processing for smart construction: Current status and future directions. *Automation in Construction*, 134, 104059. https://doi.org/10.1016/j.autcon.2021.104059

Yang, D., Rupp, A.A. & Foltz, P.W. eds. (2020). Handbook of automated scoring: Theory into practice. New York, NY: Taylor & Francis Group/CRC Press. https://doi.org/10.1201/9781351264808

Zhao, S., Zhang, Y., Xiong, X., Botelho, A. & Heffernan, N. (2017, April). A memory-augmented neural model for automated grading. In Proceedings of the 4th (2017) ACM Conference on Learning@ scale (pp. 189-192). https://doi.org/10.1145/3051457.3053982