Coherence based automatic short answer scoring using sentence Embedding

RAMESH DADI (dadiramesh44@gmail.com)
Suresh Kumar Sanampudi

Research Article

Keywords: Essay Scoring, Sentence Embedding, sentence-BERT, Bi-LSTM, Adversarial responses.

Posted Date: March 30th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2730186/v1

License: © This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract

Automatic essay scoring is an essential educational application in natural language processing (NLP). This automated process will alleviate the burden and increase the reliability and consistency of the assessment. With the advance in text embedding libraries and neural network models, AES systems achieved good results in terms of accuracy. However, the actual goals are not attained, like embedding essays into vectors with cohesion and coherence, and providing feedback to students is still challenging. In this paper, we proposed coherence-based embedding of an essay into vectors using sentence-BERT (Bidirectional Encoder Representations from Transformers). We trained these vectors on Long Short-Term Memory (LSTM) and Bi-LSTM (Bidirectional Long Short-Term Memory) to capture sentence connectivity with other sentences’ semantics. We used two different datasets; one is standard ASAP Kaggle, and another is a domain-specific dataset with almost 2500 responses from 650 students. Our model performed well on both datasets, with an average QWK (Quadratic Weighted Kappa) score of 0.76. Furthermore, we achieved good results compared to other prescribed models, and we also tested our model on adversarial responses of both datasets and observed decent outcomes.

1. Introduction

Automated essay scoring (AES) evaluates student responses written for a prompt to reduce human effort and assure consistency in scoring. Most researchers have worked on the ASE systems in the recent past. From Ramesh, D., Sanampudi, S.K (2022), from Table 1, all these approaches are categorized into four classes based on the combination of manually and automatically extracted features and used machine learning and neural networks models to train the essays. The early systems like Page, E.B (1966), Ajay et al. (1973), Burstein, J. (2003), Foltz et al. (1999), Leacock, C., & Chodorow, M. (2003), Adamson et al. (2014), Cummins et al. (2016). Manually extracted features like a bag of Words (BoW), term frequency, inverse document frequency, word count, sentence count, and sentence length. They used machine learning models like regression, support vector machine, etc., to find the relationship between essays and labels with manually extracted statistical features. Nevertheless, these approaches did not capture semantics and content for the evaluation of an essay.

<table>
<thead>
<tr>
<th>BoW/Tf-Idf</th>
<th>Word2vec/Glove (Word embedding)</th>
<th>USE (sentence embedding)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression models/classification models</td>
<td>The system implemented with BoW features and regression or classification algorithms will have low cohesion and coherence.</td>
<td>The system implemented with Word2vec features and regression or classification algorithms will have low to medium cohesion and coherence.</td>
</tr>
<tr>
<td>Neural Networks (LSTM)</td>
<td>The system implemented with BoW features and neural network models will have low cohesion and coherence</td>
<td>The system implemented with Word2vec features and neural network model (LSTM) will have medium to high cohesion and coherence</td>
</tr>
</tbody>
</table>

In the second category of approaches, Sultan et al. (2016), Contreras et al. (2018), Darwish & Mohamed (2020), and SüzenNeslihan et al. (2020) extracted content-based features with Word2vec Mikolov, Tomas, et al. (2013), and Glove Jeffrey Pennington et al. (2014). However, again, they used a machine learning model for assessment. The machine learning model considers the feature vectors independently, and they do not connect the words to capture the semantics of the essay.
The third and fourth categories of approaches would have extracted features manually, automatically, and trained different types of neural network models. Like Taghipour & Ng (2016), Dong & Zhang (2016), Riordan et al. (2017), Mathias & Bhattacharyya (2018), Wang et al. (2018), Dasgupta et al. (2018), Kumar et al. (2019), and Wilson Zhu and Yu Sun (2020) used pre-trained natural language processing models like word2vec, glove to extract features. Then, they used neural networks CNN (Convolution Neural Network), RNN (Recurrent Neural Network), and a combination of CNN and RNN to fine-tune the essays. These approaches performed well in terms of QWK score. However, the word-level feature extracting methods cannot handle polysemous words, and they miss sentence semantics and connectivity from an essay. Moreover, no AES model has proved the model’s robustness by testing adversarial responses for consistency.

From Horbach A and Zesch T (2019), Riordan Brain et al. (2019), Ding, Yuning, et al. (2020), and Kumar, Y. et al. (2020) proved that the black box type of models is prone to adversarial responses and its challenging task to handle irrelevant, repeated sentences type of responses. So, we need an AES system to handle adversarial responses and evaluate essays based on content.

Contribution.

This paper mainly concentrates on feature extraction without missing coherence and cohesion from an essay. We used a recurrent neural network to capture sentence connectivity to give the final score.

We developed an AES system based on sentence-level embedding to capture sequential features, which fine-tunes the relevance and semantics of individual essays to give the final score.

We evaluated our system on two data sets to prove our model’s robustness. One is the standard data set, and another is a new dataset on the operating system domain created by us, which is publicly available. Our approach outperforms existing AES-based approaches.

We demonstrated the effectiveness of our approach through experimental evaluation on various adversarial responses. Our approach significantly outperformed.

Organization.

The rest of the paper is organized as follows. Section 2 discusses the related work about text embeddings and deep learning models used for AES systems and their challenges. Section 3 presents the proposed method of the AES system on different datasets and sentence embeddings. Section 4 discusses the implemented models, their architectures, and the hyperparameters used during training. Section 5 discusses the experimental results compared with other models and presents the model’s performance on adversarial responses. Finally, section 6 discusses the conclusion and future work.

2. Related Work

Automated essay scoring (AES) is the task of assessing student responses (short or essay); it is the most challenging task in Natural Language Processing (NLP). The main job of the AES system is to extract features from essays with cohesion and coherence and train a neural network model to fine-tune the features and assign a final score. The early systems like Ajay et al. (1973), Burstein, J. (2003), Foltz et al. (1999), Rudner and Liang (2002), Adamson et al. (2014), Cummins et al. (2016). Extracted handcrafted features from the essay assigned score.

With advances in NLP and neural networks, the feature extraction for AES also changed from manual to automatic extraction. The first automatic features extraction method is word2ve, a pre-trained model that can extract word-level context into a vector. Like Sakaguchi et al. (2015), Mathias & Bhattacharyya (2018), Taghipour, K., & Ng, H. T. (2016), Kumar et al. (2019), Xia L et al. (2019) and Wilson Zhu and Yu Sun (2020). Wang et al. (2018b) used a reinforcement
model to train word vectors. However, word-level context encoding fails on polysemous words and does not handle adversarial responses. These models assign a score when a student submits an irrelevant response with few related words.

Rather than training word context vectors only on the LSTM model, researchers like Dong & Zhang (2016) and Dong et al. (2017). Riordan et al. (2017) and Mathias & Bhattacharyya (2018) added an extra CNN layer on top of the LSTM or Bi-LSTM to form sentences. The CNN layers will add a fixed number of words to form sentences. However, with this approach, the model’s accuracy has been increased, the actual sentences have diverged, and actual word connectivity is missing. Still, do not handle irrelevant responses like one-word, random-word responses. Kumar et al. (2019) implemented a domain-based AES system based on word embedding.

In recent researchers embedded essays directly sentence by sentence using USE(Universal Sentence Encoder and sentence-BERT like Pedro Uria Rodriguez et al. (2019), Jiaqi Lun et al. (2020), Yang, R et al. (2020), Song, W et al. (2020), Ormerod, C. M et al. (2021), Doewes, A., &Pechenizkiy, M. (2021). Mayfield, E., and Black, A. W. (2020) used BERT for essay embedding, but they used word level embedding instead of sentence, and trained an LSTM model for fine-tuning. USE and sentence-BERT capture coherence and cohesion from an essay, and the LSTM model will tune sentence connectivity and coherence to assign a score. Fernandez N et al. (2022) used BERT for reading comprehension and trained separately on prompt, text, and student response for assessment. Wang Y et al. (2022) used BERT and LSTM and performed well in terms of accuracy. However, no model showed the model’s robustness and how the system handles adversarial responses.

Table 1 illustrates the possible combinations of text embedding and machine and deep learning models used for essay scoring. Table 2 shows the text embedding technique and vector dimension they create; out of all sentence-BERT will give a low dimension vector for sentence. The best combination is sentence embedding and recurrent neural network because sentence embedding will capture coherence from an essay and can easily handle polysemous words, which was a deficiency in word embeddings.

<table>
<thead>
<tr>
<th>Embedding Model</th>
<th>Vector dimension</th>
<th>Essay Dimension</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2Vec</td>
<td>32 Dimension per word</td>
<td>32 * 96* number of words per each sentence</td>
<td>Word level embedding</td>
</tr>
<tr>
<td>ELMo</td>
<td>1024 dimension per word</td>
<td>1024<em>96</em> number of words per each sentence</td>
<td>Word level embedding</td>
</tr>
<tr>
<td>XP2</td>
<td></td>
<td></td>
<td>Word level embedding</td>
</tr>
<tr>
<td>USE</td>
<td>512 Dimension per sentence</td>
<td>96 *512</td>
<td>High dimension vectors for sentence</td>
</tr>
<tr>
<td>Sentence-BERT</td>
<td>128 Dimension per sentence</td>
<td>96*128</td>
<td>Low dimension vector for each sentence</td>
</tr>
</tbody>
</table>

3. Method

We proposed a different approach that implements a sentence-based text embedding to capture coherence and trained an LSTM and Bi LSTM separately. We used two data sets; one is a standard ASAP data set; another is domain-specific collected from 600 students as an assignment, a total of 2300 responses. Moreover, we tested our model on different
types of adversarial responses to check the system's robustness. The detailed architecture of the proposed AES system with sentences BERT (Devlin et al., 2019) and Bi-LSTM is illustrated in Fig. 1.

3.1 data set

We used the ASAP Kaggle data set, widely used in AES systems. ASAP dataset comprises 12,978 essays of 8th to 10th standard students on eight different prompts. Each prompt consists of 1500 and above essays evaluated by two raters. Prompts 3,4,5,6 are source-dependent essays, and the remaining are others. A detailed description of the essay dataset is presented in Table 3.

Table 3
Kaggle ASAP data set for essay scoring

<table>
<thead>
<tr>
<th>Essay set</th>
<th>No. of essays</th>
<th>Average length of essays</th>
<th>Rating range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1783</td>
<td>350</td>
<td>2–12</td>
</tr>
<tr>
<td>2</td>
<td>1800</td>
<td>350</td>
<td>1–6</td>
</tr>
<tr>
<td>3</td>
<td>1726</td>
<td>150</td>
<td>0–3</td>
</tr>
<tr>
<td>4</td>
<td>1772</td>
<td>150</td>
<td>0–3</td>
</tr>
<tr>
<td>5</td>
<td>1805</td>
<td>150</td>
<td>0–4</td>
</tr>
<tr>
<td>6</td>
<td>1800</td>
<td>150</td>
<td>0–4</td>
</tr>
<tr>
<td>7</td>
<td>1569</td>
<td>250</td>
<td>0–30</td>
</tr>
<tr>
<td>8</td>
<td>723</td>
<td>650</td>
<td>0–60</td>
</tr>
</tbody>
</table>

In addition, we created new data set on the domain operating system (OS) to test the performance of AES systems on domain-specific essays. We framed five basic questions as an assignment from operating systems, a computer science subject, and distributed them to students of various engineering colleges. We got 2981 responses from students after eliminating repeated or multiple responses. Finally, we left with 2390 responses from 626 students. Two subject experts evaluated the new dataset on 0–5, the minimum score is 0, and the maximum score is 5. We used the QWK score to measure the agreement between the two raters. The resulting score is 0.842(QWK). A detailed description of the OS dataset is demonstrated in Table 4.

Table 4
operating system data set (https://github.com/RAMESHDADI/OS-data_1-set-for-AES)

<table>
<thead>
<tr>
<th>Essay id</th>
<th>No. of essays</th>
<th>length of essays</th>
<th>Rating range (min to max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>516</td>
<td>1–23</td>
<td>0–5</td>
</tr>
<tr>
<td>2</td>
<td>596</td>
<td>1–23</td>
<td>0–5</td>
</tr>
<tr>
<td>3</td>
<td>312</td>
<td>1–23</td>
<td>0–5</td>
</tr>
<tr>
<td>4</td>
<td>513</td>
<td>1–23</td>
<td>0–5</td>
</tr>
<tr>
<td>5</td>
<td>453</td>
<td>1–23</td>
<td>0–5</td>
</tr>
</tbody>
</table>

3.2 sentence embedding

In natural language processing, converting text into a vector with context and semantics is challenging. The embedding techniques like word2vec and Glove convert text into vector word by word, but they do not consider surrounding words
and their static. Specifically, not able to handle polysemous words. Moreover, sentence embedding techniques are there, converting word vectors into sentence vectors by taking an average of all words.

So, in our model, we used sentence BERT to convert essays into vectors. Sentence BERT will convert text into vectors dynamically with context and semantics, and it can reconstruct the original sentence from the vector.

First, we removed all special symbols like (@, #) from an essay and tokenized in into sentences. In the ASAP and OS data sets, 96 and 23 maximum number of sentences are in each data set, respectively. Then, we used a pre-trained transformer model, i.e., Sentence-BERT, to embed each sentence into a 128-dimension vector. So, we have 96*128, 23*128-dimension vectors for an essay of ASAP, OS data sets. Finally, we padded all the essays into 96*128 and 23*128 vectors, where 96 and 23 are the maximum number of sentences in ASAP and OS datasets. Table 5 will illustrate the sentence vectors of an essay from both datasets.

### Table 5

<table>
<thead>
<tr>
<th>dataset</th>
<th>Sample Essay embedded vector by BERT</th>
<th>dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASAP</td>
<td>[ 0.01377844 – 0.09247074 0.01014971 … -0.01349053 -0.04146808 0.05626552]</td>
<td>96 * 128</td>
</tr>
<tr>
<td></td>
<td>[-0.01370333 -0.0240192 -0.03880018 … -0.05234249 -0.06115109 0.05296136]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[-0.05529318 -0.02587196 -0.00212097 … 0.02701825 0.02506788 0.00300164]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>…</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>[[-0.04040171 0.00535519 –0.02015072 … -0.07707731 -0.07179338 0.05336216]</td>
<td>23*128</td>
</tr>
<tr>
<td></td>
<td>[-0.00677465 0.01085155 0.03253689 … -0.07136418 -0.05266594 -0.01695863]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0.03762686 0.02985795 –0.05078415 … -0.05361476 -0.02895073 0.04693419]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>…</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ 0. 0. 0. ... 0. 0. 0. ]</td>
<td></td>
</tr>
</tbody>
</table>

### 4. Model

An essay is a sequence of sentences jointly justifying the prompt. So, the implemented model should consider extracting coherence, cohesion, and linguistic features from an essay to give the final score. So, we embedded all essays, sentence-wise, without removing stop words to capture coherence from the essay. Then, we implemented LSTM and Bi-LSTM separately to train the sentence vectors. Moreover, we also implemented a CNN + LSTM model like Taghipour & Ng (2016) will capture N-gram features and then transform them into an LSTM unit.

**LSTM/Bi-LSTM**

LSTM and Bi-LSTM are recurrent neural networks that can process a sequence of information with their memory cell. The memory cell consists of an input gate (2), forgot gate (1), an output gate (3), and a context gate (4) to process the
information and store the long-term dependency information required for feature use and that is passed to the next cell input gate.

Bi-LSTM traverses the sentence vectors in both directions (1)(2) to capture coherence. From each sentence, context information will be stored and attached to the following sentence from these combined sentences; again, some info is stored, like all sentences will summarize and predict the final score for the essay.

\[
\begin{align*}
H(\text{forward}) &= \sigma (w (1) x (1) + w (2) x (2) + w (3) x (3) \ldots \ldots w (t) x (t)) + b (1) \\
H(\text{backward}) &= \sigma (w (1) x (1) + w (2) x (2) + w (3) x (3) \ldots \ldots w (t) x (t)) + b (2)
\end{align*}
\]

Where \(w (1), w (2) \ldots w (t)\) weights, \(b\) is bias, \(\sigma\) is activation function, \(y\) is output, and \(t\) is the number of sentences in the essay.

### 4.1 Implementation and Training RNN

First, we embedded all the essays into vectors using sentence BERT, then padded all the essays into the full-size essay, i.e., 96*128 like in Fig. 2. Then we converted all vectors to 3-dimensional vectors to train on a neural network.

In LSTM, we stacked five layers of LSTM; each unit has an input gate, output gate, and context gate. We used an RMSprop optimizer to reduce the mean square error like Dong et al. (2017), drop rate as 0.5, assign initial learning rate as 0.001, and activate function as ReLU. The hyperparameters of our models are shown in Table 6.

<table>
<thead>
<tr>
<th>Layer</th>
<th>parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding</td>
<td>Sentence Embedding (BERT)</td>
<td>128 size vectors for each sentence</td>
</tr>
<tr>
<td>CNN layer</td>
<td>Input size</td>
<td>(1,96,128), (1, 23,128)</td>
</tr>
<tr>
<td>LSTM layers</td>
<td>No of layers</td>
<td>5</td>
</tr>
<tr>
<td>LSTM Units</td>
<td>LSTM Units</td>
<td>300</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden units</td>
<td>200,100</td>
</tr>
<tr>
<td>Drop out</td>
<td>Dropout rate</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>Recurrent Drop out</td>
<td>0.5</td>
</tr>
<tr>
<td>Others</td>
<td>Epochs</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>Optimizer</td>
<td>Adam</td>
</tr>
</tbody>
</table>

In the training phase, we used 5-fold cross-validation, like Taghipour & Ng (2016), to split the essay vectors into training, testing, and validation in a 70:15:15 ratio for both data sets. To fix the hyperparameters, we trained our model for 10, 15, 20, 35 epochs and fixed the hyperparameters. We use QWK as an evaluation metric in our models to find the agreement between the human and system rater, which has been widely used for AES Taghipour & Ng (2016), Wang et al. (2018b), Tay et al.)
In each fold, we calculated the QWK score. Finally, we use the model that achieves the best performance on training data to predict the test data. Figure 3 illustrates the training and validation loss of the proposed model, and it portrays that our proposed model is neither over fitted nor under fitted.

We used the same hyper parameters and 5-fold cross-validation to train sentence-LSTM and sentence-Bi-LSTM on the OS dataset. OS data set input dimension is 23*128 for LSTM and Bi-LSTM, 23 is the maximum number of sentences, and 128 is the sentence vector.

**5. Result Analysis**

The experiment results we obtained based on the ASAP dataset and OS dataset for the AES system are shown in Fig. 3,4; we observed that our proposed model performance on both datasets is best fitted. We trained the model prompt-wise and calculated training and validation loss prompt-wise.

The results show that our proposed models outperformed and equally performed with other models. The comparison of all baseline models on the ASAP dataset and our proposed models on average QWK score is shown in Table 7. We found that sentence embedding-LSTM and sentence embedding-Bi-LSTM models performed well compared to other models and were consistent with the human rater score. Furthermore, it observed that Sentence Embedding-LSTM and Bi-LSTM performed better than models like Muangkammuen, Panitan, and Fumiyo Fukumoto (2020), Agrawal, Aman, and Suyash Agrawal (EASE) (2018), LSTM-MOT model of Taghipour and Ng (2016), and CNN + LSTM integrating deep learning model (2021). Though SKIPFLOW-LSTM (2017), TSLF-ALL Liu, Jiawei, et al. (2019), and CNN-LSTM Attention Dong, F., Zhang, Y., & Yang, J. (2017). models performed equally with sentence embedding-LSTM and Bi-LSTM. However, these integrated and word embedding models did not capture sentence coherence; because of neural networks, the QWK score was high.

<p>| Prompt wise QWK score with LSTM and Bi-LSTM on ASAP dataset | Table 7. Prompt wise QWK score with LSTM and Bi-LSTM on ASAP dataset |</p>
<table>
<thead>
<tr>
<th>Models</th>
<th>Prompt-1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Avg</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rater1 to Rater 2 agreement</td>
<td>0.721</td>
<td>0.812</td>
<td>0.769</td>
<td>0.851</td>
<td>0.753</td>
<td>0.776</td>
<td>0.720</td>
<td>0.627</td>
<td>0.754</td>
<td>-NA-</td>
</tr>
<tr>
<td>PEG, Page, E.B(1966)</td>
<td>0.82</td>
<td>0.72</td>
<td>0.75</td>
<td>0.82</td>
<td>0.83</td>
<td>0.81</td>
<td>0.84</td>
<td>0.73</td>
<td>0.79</td>
<td>Hand crafted features, simple linear regression model</td>
</tr>
<tr>
<td>Multi-task (word&amp;sentence sentiment)</td>
<td>0.803</td>
<td>0.658</td>
<td>0.664</td>
<td>0.772</td>
<td>0.799</td>
<td>0.816</td>
<td>0.787</td>
<td>0.644</td>
<td>0.743</td>
<td>Word level embedding</td>
</tr>
<tr>
<td>Muangkammuen, Panitan and Fumiyo Fukumoto (2020)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EASE</td>
<td>0.76</td>
<td>0.61</td>
<td>0.62</td>
<td>0.74</td>
<td>0.78</td>
<td>0.78</td>
<td>0.73</td>
<td>0.62</td>
<td>0.71</td>
<td>Hand crafted features</td>
</tr>
<tr>
<td>LSTM-MoT model (Taghipour and Ng) (2016)</td>
<td>0.775</td>
<td>0.687</td>
<td>0.683</td>
<td>0.795</td>
<td>0.818</td>
<td>0.813</td>
<td>0.805</td>
<td>0.594</td>
<td>0.746</td>
<td>Ensemble method on word vectors</td>
</tr>
<tr>
<td>SKIPFLOW-LSTM model (Tay et al.) (2018)</td>
<td>0.832</td>
<td>0.684</td>
<td>0.695</td>
<td>0.788</td>
<td>0.815</td>
<td>0.810</td>
<td>0.800</td>
<td>0.697</td>
<td>0.764</td>
<td>Ensemble method on word vectors</td>
</tr>
<tr>
<td>RL1</td>
<td>0.766</td>
<td>0.659</td>
<td>0.688</td>
<td>0.778</td>
<td>0.805</td>
<td>0.791</td>
<td>0.760</td>
<td>0.545</td>
<td>0.724</td>
<td>Word level embedding</td>
</tr>
<tr>
<td>Wang et al. (2018b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN+ LSTM integrating deep learning model (2021)</td>
<td>0.87</td>
<td>0.64</td>
<td>0.63</td>
<td>0.83</td>
<td>0.86</td>
<td>0.85</td>
<td>0.79</td>
<td>0.53</td>
<td>0.73</td>
<td>Ensemble method on word vectors</td>
</tr>
<tr>
<td>Dong et al. (2017)</td>
<td>0.822</td>
<td>0.682</td>
<td>0.672</td>
<td>0.814</td>
<td>0.803</td>
<td>0.811</td>
<td>0.801</td>
<td>0.705</td>
<td>0.764</td>
<td>Ensemble method on word vectors</td>
</tr>
<tr>
<td>(LSTM-CNN attention)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2 BERT</td>
<td>0.817</td>
<td>0.719</td>
<td>0.698</td>
<td>0.845</td>
<td>0.841</td>
<td>0.847</td>
<td>0.839</td>
<td>0.744</td>
<td>0.794</td>
<td>word level embedding and used regression</td>
</tr>
<tr>
<td>ALBERT</td>
<td>0.807</td>
<td>0.671</td>
<td>0.672</td>
<td>0.813</td>
<td>0.802</td>
<td>0.816</td>
<td>0.826</td>
<td>0.700</td>
<td>0.766</td>
<td>Sentence embedding</td>
</tr>
<tr>
<td>Ormerod, C. M et al. (2021)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tran-BERT-MS-ML-R, Wang, Y et al. (2022)</td>
<td>0.834</td>
<td>0.716</td>
<td>0.714</td>
<td>0.812</td>
<td>0.813</td>
<td>0.836</td>
<td>0.839</td>
<td>0.766</td>
<td>0.791</td>
<td>Token(word) level embedding, each word with 510</td>
</tr>
</tbody>
</table>
However, like other models, our model consistently performed on source-dependent essay traits like prompt 3, 4, 5, 6; these essay traits' rating range is between 1 and 6. Furthermore, based on the QWK score, the performance of our models and other baseline models on persuasive, narrative, and expository essay traits is the same and a little bit high. However, the performance was reduced when we used the CNN layer on the Sentence Embedding-LSTM, Bi-LSTM neural networks model; with this, we conclude that when we split essays or sentences into tokens, the coherence and cohesion were missing. Though Ormerod C. M et al. (2021) and Wang Y et al. (2022) used BERT for text embedding, they tokenized the essay into words. Moreover, they were embedded through BERT and performed well but could not capture coherence from an essay.

The models trained on the ASAP dataset were also used on the OS dataset with the same hyper parameters, and it is performed with an average QWK score of 0.746 and 0.751 by Bi-LSTM and LSTM models. Table 8 illustrates the prompt-wise QWK score and average QWK score of the word embedding, Sentence Embedding-LSTM, and Sentence Embedding-Bi-LSTM model on the OS dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt-1</th>
<th>Prompt-2</th>
<th>Prompt-3</th>
<th>Prompt-4</th>
<th>Prompt-5</th>
<th>QWK score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word embedding model</td>
<td>0.745</td>
<td>0.738</td>
<td>0.634</td>
<td>0.722</td>
<td>0.742</td>
<td>0.716</td>
</tr>
<tr>
<td>Sentence Embedding-LSTM*</td>
<td>0.766</td>
<td>0.759</td>
<td>0.696</td>
<td>0.758</td>
<td>0.776</td>
<td>0.751</td>
</tr>
<tr>
<td>Sentence Embedding-Bi-LSTM*</td>
<td>0.761</td>
<td>0.758</td>
<td>0.688</td>
<td>0.751</td>
<td>0.772</td>
<td>0.746</td>
</tr>
</tbody>
</table>

5.1 Testing on Adversarial responses
To know how models perform on adversarial responses and to comprehend whether the essays are evaluated based on the semantics. We prepared eight test cases on adversarial responses like irrelevant, relevant responses, prompt as a response, and repeated sentences. Table – 9 and 10 compares actual and predicted scores of word embedding (word2vec) and sentence embedding models on all eight test cases. From test cases 1,2,3, the difference between the actual and predicted scores of the word embedding model is high. On the other hand, the sentence embedding model performed well.

In tests case-5 and 6, the word embedding model underperformed when we tested irrelevant responses with few words matched with content. On the other hand, in sentence repeated responses, test cases-5, and six capture the semantics of the essay and do not consider duplicate sentences while providing the final score. For example, from Table 10, test cases 1,3 are irrelevant responses, test case 2 is one sentence response, test case 4 is 50% relevant, and the remaining irrelevant response. In test case 5, the sentences are repeated. In all these cases, our model performed well.

We observed that the word embedding model is underperformed and does not capture essay semantics. In contrast, Sentence Embedding with LSTM performed well on irrelevant responses, given a final score based on relevance.

We strongly argue that our model captures coherence and cohesion from essays while evaluating and assigning a score. We used sentence-to-sentence embedding and trained neural networks to capture sequence-to-sequence patterns from the essay. However, our model’s average QWK score is greater than or equal to other baseline models, but our model assesses essays based on coherence and cohesion. Moreover, our model performance is consistent in semantics and relevance while testing adversarial responses.

### Table 9

Testing and comparing Results of proposed model and word embedding model on adversarial responses

<table>
<thead>
<tr>
<th>Test case</th>
<th>Adversarial responses</th>
<th>Actual Score (human rater score)</th>
<th>Predicted Score of proposed models (Sentence embedding with LSTM)</th>
<th>Predicted Score of Word embedding model (Word2vec + LSTM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Essay from Prompt - 1 which is relevant to the response</td>
<td>9</td>
<td>9.5</td>
<td>7.7</td>
</tr>
<tr>
<td>2</td>
<td>One sentence response for prompt-1</td>
<td>2</td>
<td>2.2</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>Essay from Prompt-1 which is relevant to the response</td>
<td>7</td>
<td>7.1</td>
<td>5.2</td>
</tr>
<tr>
<td>4</td>
<td>Prompt as response</td>
<td>0</td>
<td>0.38</td>
<td>2.1</td>
</tr>
<tr>
<td>5</td>
<td>Irrelevance response to the prompt-1 (where few words are matching to the prompt)</td>
<td>0</td>
<td>0.89</td>
<td>2.90</td>
</tr>
<tr>
<td>6</td>
<td>Response with repeated sentences (where 50% of the answer is correct)</td>
<td>5</td>
<td>4.82</td>
<td>7.6</td>
</tr>
<tr>
<td>7</td>
<td>One word response</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>8</td>
<td>Response with 50% relevant answer and remaining 50% is irrelevant.</td>
<td>6</td>
<td>5.88</td>
<td>7.2</td>
</tr>
</tbody>
</table>
## Table 10
testing adversarial responses on OS data set.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Adversarial response</th>
<th>Prompt</th>
<th>Actual Score</th>
<th>Predicted Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It is a good interaction between user and the computer.</td>
<td>Explain the advantages of a multiprocessor system?</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>Intermediate to user and hardware</td>
<td>Explain the advantages of a multiprocessor system?</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>A system is a group of interacting or interrelated elements that act according to a set of rules to form a unified whole. A system, surrounded and influenced by its environment, is described by its boundaries, structure and purpose and expressed in its functioning.</td>
<td>Explain how operating system handle multiple tasks at a time?</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>A <strong>system</strong> is a way of working, organizing, or doing something which follows a fixed plan or set of rules. You can use <strong>system</strong> to refer to an organization or institution that is organized in this way. The Operating system determines which task the cpu will work on at any given time, pausing tasks as needed, so that all the tasks as completed as efficiently as multiprocessor operating system use a single cpu to work on a number of tasks.</td>
<td>Explain how operating system handle multiple tasks at a time?</td>
<td>2</td>
<td>2.6</td>
</tr>
<tr>
<td>5</td>
<td>The os assigns a number of tasks to a cpu and perform all the tasks one by one using various kinds of priorities. The way of deciding the priorities are the scheduling algorithms. There are 2 types of scheduling algorithms and they are preemptive and non-preemptive. And there are 4 ways for deciding the priorities. They are first come first service, shortest job first, priority scheduling, round robin. Based on the requirements the type of scheduling is allotted and the processes get executed. The os assigns a number of tasks to a cpu and perform all the tasks one by one using various kinds of priorities. The way of deciding the priorities are the scheduling algorithms. There are 2 types of scheduling algorithms and they are preemptive and non-preemptive. And there are 4 ways for deciding the priorities. They are first come first service, shortest job first, priority scheduling, round robin. Based on the requirements the type of scheduling is allotted and the processes get executed.</td>
<td>Explain how operating system handle multiple tasks at a time?</td>
<td>4</td>
<td>5.1</td>
</tr>
</tbody>
</table>

## 6. Conclusion

In this paper, we proposed and implemented a novel approach for an AES system with a combination of sentence embedding with an LSTM and Bi-LSTM. All the models are trained and tested on the Kaggle ASAP and OS datasets. In this approach, we embedded the essay sentence by sentence after preprocessing to capture sequence-to-sequence coherence patterns and trained on recurrent neural networks. Specifically, we compared our models: Sentence embedding-LSTM and Sentence Embedding-Bi-LSTM with baseline models of AES; it has been observed that Sentence Embedding-LSTM and Bi-LSTM performed well among all models. Moreover, our proposed models outperformed other baseline models without missing coherence. Some of the model's performance is high in terms of QWK score, but so far, all models have used word-based embedding.

We tested our proposed models on adversarial responses and compared the actual and predicted scores. While testing on adversarial responses, our model assigned scores for essays based on coherence and cohesion; when we were given
irrelevant responses and repeated sentence essays, our model captured content and assigned scores. From the results, it’s proved that our models are performing consistently. Furthermore, we also trained our models on different datasets like OS and observed consistent performance.

In the future, we will continue our study on trait-based AES systems and test the model on more adversarial responses to test the robustness of the model. To handle Out of Vocabulary (OOV) words, we are creating a separate corpus related to the OS dataset domain to handle OOV words.

Declarations

Ethics approval and consent to participate: Not applicable

Consent for publication: Not applicable

Availability of data and materials: https://github.com/RAMESHDADI/OS-data_1-set-for-AES

Competing interests: The authors declare that they have no conflict of interest.

Funding: Not Applicable

Authors’ contributions: All authors equally contributed and approved the final manuscript.

Acknowledgements: We thank SR University and JNTU college of Engineering Jagitial, students, and faculty for collecting the Essay dataset.

References


**Figures**
Figure 1

Architecture of AES system
Figure 2

Working of LSTM model on sentence vectors of each essay

Figure 3
prompt wise training and validation loss of OS data set model

Figure 4

prompt wise training and validation loss of ASAP data set model

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- OSdataset.csv