Article

Automated Essay Scoring: A Siamese Bidirectional LSTM Neural Network Architecture

Guoxi Liang ^{1,2} , Byung-Won On ^{3,*,†}, Dongwon Jeong ³ and Hyun-Chul Kim ⁴
and Gyu Sang Choi ^{5,*}

¹ Department of Global Entrepreneurship, Kunsan National University, Gunsan 54150, Korea; guoxiliang2017@gmail.com

² Department of Information Technology, Wenzhou Vocational & Technical College, Wenzhou 325035, China

³ Department of Software Convergence Engineering, Kunsan National University, Gunsan 54150, Korea; djeong@kunsan.ac.kr

⁴ Department of Technological Business Startup, Kunsan National University, Gunsan 54150, Korea; kimhc@kunsan.ac.kr

⁵ Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea

* Correspondence: on.byung.won@gmail.com (B.-W.O.); castchoi@ynu.ac.kr (G.S.C.); Tel.: +82-63-469-8913 (B.-W.O.)

† Current address: 151-109 Digital Information Building, Department of Software Convergence Engineering, Kunsan National University 558 Daehak-ro, Gunsan, Jeollabuk-do 573-701, Korea.

Received: 14 November 2018; Accepted: 27 November 2018; Published: 01 December 2018

Abstract: Essay scoring is a critical task in education. Implementing automated essay scoring (AES) helps reduce manual workload and speed up learning feedback. Recently, neural network models have been applied to the task of AES and demonstrates tremendous potential. However, the existing work only considered the essay itself without considering the rating criteria behind the essay. One of the reasons is that the various kinds of rating criteria are very hard to represent. In this paper, we represent rating criteria by some sample essays that were provided by domain experts and defined a new input pair consisting of an essay and a sample essay. Corresponding to this new input pair, we proposed a symmetrical neural network AES model that can accept the input pair. The model termed Siamese Bidirectional Long Short-Term Memory Architecture (SBLSTMA) can capture not only the semantic features in the essay but also the rating criteria information behind the essays. We use the SBLSTMA model for the task of AES and take the Automated Student Assessment Prize (ASAP) dataset as evaluation. Experimental results show that our approach is better than the previous neural network methods.

Keywords: automated essay scoring (AES); deep learning; neural network; long short-term memory; essay; rating criteria

1. Introduction

Manual scoring has a large workload and sometimes is subjective according to different experts. The goal of automated essay scoring (AES) is to enable computers to score students' essays automatically, thereby reducing the subjectivity of manual ratings and the workload of teachers and speeding up the feedback in the learning process. Currently, there are some AES systems, such as Project Essay Grade (PEG) [1], Intelligent Essay Assessor (IEA) [2], E-rater [3], and Besty that are applied to educational practice, but these systems are not promising in the future. AES is quite complicated; it depends on how much the machine could understand the language, such as spelling, grammar, semantics and other grading information. Traditional AES approaches were regarded as

a machine learning approach, such as classification [4,5], regression [3,6], or ranking classification problems [7,8]. These approaches make use of various features, such as the length of the essay, Term Frequency-Inverse Document Frequency (TF-IDF), etc., to achieve AES. One drawback of this kind of feature extraction is that it is often time-consuming and the regulation for feature extraction is often sparse, instantiated by discrete pattern-matching and it being hard to generalize.

The neural network and distributed representation [9,10] have provided tremendous potential for natural language processing. A neural network can train an essay represented by distributed representation and producing a single dense vector that represents the whole essay. Furthermore, the single dense vector and the score are trained by the neural network to form a one-to-one correspondence. Without any other handcrafted features, a nonlinear neural network model has been shown its special advantages—that it's much more robust than the traditional statistical models across different domains. Recently, many researchers have studied AES using neural networks [11–16] and made quite good progress. These researchers mainly focus on convolutional neural networks (CNN) [17–19], recurrent neural networks [20] (RNN, the most widely used RNN is long short-term memory (LSTM) [21]), the combination of CNN and RNN (LSTM), attention mechanisms, and some special internal features representation, such as coherence feature among sentences [18]. CNN has a good application in the image [22,23], and it can also be applied to sequence models [24]. RNN is very advantageous for sequence modeling. Google applied the attention module to the language mode directly [25,26]. However, at present, the researchers applied all kinds of models to AES only considering the essay itself while neglecting the rating criteria behind the essay. In this paper, we considered this kind of information and gave an interpretable novel end-to-end neural network AES approach. We represent rating criteria by introducing some sample essays (the following short as the sample) with different ranks which were provided by domain experts (if not, manually get an average one from the dataset instead). Thereby, we get some essay pairs as new inputs to AES. Each pair consists of an essay itself and a sample. We proposed a Siamese Bidirectional Long Short-Term Memory Architecture (SBLSTMA) to receive the new input to achieve AES. Because the rating information was also involved, our SBLSTMA model can capture not only the semantic information in the essays but also the information beyond of the dataset—rating criteria. We explored the SBLSTMA model for the task of AES and used the Automated Student Assessment Prize (ASAP) dataset (ASAP, <https://www.kaggle.com/c/asap-aes/data>) as evaluation. The results show that our model empirically outperforms the previous neural network AES methods.

Figure 1 shows the overall framework of the approach. Different from the previous approaches that train or predict the dataset directly (the above of Figure 1), we added rating criteria as a part of input (the bottom of Figure 1). Experience tells that human raters give scores not only by essays themselves but also by rating criteria (We use samples instead). Our model is to imitate this behavior of human raters. We believe that essays don't have all the rating information and some of that is beyond the essays. Therefore, to take this kind of information as a part of the input is a benefit for scoring. We briefly describe how to make use of this sample first. We simply mark v as the distribute represent function; then, $v(e)$ and $v(s)$ are the word embeddings of essay e and sample s , respectively. The difference between the essay vector $v(e)$ and the sample vector $v(s)$ is defined as the distance information of these two. Mark $dist = v(e) - v(s)$ as the distance information, subsequently, as shown in Figure 2, $dist$ and $v(e)$ are fed into the model together. We mark pair $(v(e), v(s))$ as the new input, and we can also construct a map to represent the label of pair $(v(e), v(s))$. The detail description of input was described in Section 3.1.

The prime contributions of our paper are as follows:

- For the first time, we introduce some samples to represent the rating criteria to increase the rating information and construct a pair consisting of an essay and a sample as the new input. We can understand it as how similar is the essay and sample or how close is the essay and sample. This, to a certain extent, is similar to semantic similarity [27] and question–answer matches [14]. We introduce it to AES.

- We provide a self-feature mechanism at the LSTM output layer. We compute two kinds of similarities: the similarity between sentences in the essay and the similarity between essay and sample. The experiment shows that it is a benefit for the essays, which are long and complicated. This idea is inspired by the SKIPFLOW [14] approach, but we make an extension of it.
- We proposed a Siamese Bidirectional Long Short-Term Memory Architecture (SBLSTMA); this is a Siamese neural network architecture that can receive the essay and sample in each side. We use the ASAP dataset as an evaluation. The results show that our model empirically outperforms the previous neural network AES approaches.

This paper is organized in the following way: Section 2 discusses the related work, Section 3 describes automated essay scoring, and Section 4 is about the experiment, results, and discussion. Finally, conclusions are in Section 5.

Figure 1. Overall framework of the approach.

Figure 2. Siamese bidirectional long short-term memory architecture model architecture.

2. Related Works

Research on AES began decades ago. In the field of application, the first AES system named Project Essay Grade (PEG) [1] for automating the educational assessment was seen in 1967. Intelligent Essay Assessor (IEA) [2] adopts a Latent Semantic Analysis (LSA) [28] algorithm to produce semantic

vectors for essays and computes the semantic similarity between the vectors. The E-rater system [3], which can extract various grammatical structure features of the essay, now plays a facilitating role in the Graduate Record Examination and Test of English as a Foreign Language. In the early research of AES, it was regarded as a semi-automated machine learning approach based on various feature extractions. Larkey [4] and Rudner and Liang [5] treated AES as a kind of classification using bag-of-words features. Attali and Burstein [3] and Phandi [2] used regression approaches to achieve AES. Yannakoudakis et al. [7] took automated essay scoring as a ranking problem by ranking the order of pair essays based on their quality. Features such as words, Part-of-Speech (POS) tagging, n-grams features, complex grammatical features, are extracted. Tandalla [29] used traditional machine learning approaches to extract multi-features to achieve AES, including regular expression from the text and trained on ensemble learning approaches like RF and GBM. Arif Mehmood et al. [30] also proposed a model performing AES using multi-text features and ensemble machine learning. Chen and He [8] described AES as a ranking problem which took the order relation among the whole essays into account. The features contain syntactical features, grammar and fluency features as well as content and prompt specific features. Shristi Drolia et al. [31] proposed a regression-based approach for automatically scoring essays that are written in English; they use standard Natural Language Processing (NLP) techniques for extracting the features from the essays. Phandi et al. [6] made use of correlated Bayesian Linear Ridge Regression approach to tackle domain-adaptation tasks. McNamara et al. [32] evaluated the use of a hierarchical classification approach to the automated assessment of essays. This research computes the essay scores by using a hierarchical approach, analogous to an incremental algorithm for hierarchical classification. Fauzi et al. [33] used an automatic essay scoring system which was based on n-gram and cosine similarity to extract features and also considered the word order. Based on the existing automated essay evaluation systems, Zupanc et al. [34] proposed an approach which incorporates additional semantic coherence and consistency attributes. They extracted the coherence attributes by transforming sequential parts of an essay into the semantic space and calculating the changes between them to estimate coherence of the essay. All of these methods mentioned above are all kinds of machine learning that needs a handcrafted features extraction. The application fields of that have certain limits, and the average accuracy is not always good.

Since deep learning was introduced into natural language processing, more and more researchers have carried out related research. Cícero Nogueira dos Santos [17] proposed a deep convolutional neural network which focuses on different levels of analysis that from character-level to sentence-level information to perform sentiment analysis of short essays. Wenpeng et al. [18] investigated machine comprehension on a question answering (QA) benchmark called MCTest. They proposed a neural network framework, termed hierarchical attention-based convolutional neural network (HABCNN), to address this task without any handcrafted features. HABCNN employs an attention mechanism to weight the key phrases, key sentences and key snippets that are relevant to answering the question. Zhang et al. [19] gave a sensitivity analysis of one-layer CNN to explore the effect of architecture components on model performance to distinguish between important and comparatively inconsequential design decisions for sentence classification. Yang et al. [35] proposed a hierarchical attention network for document classification. The model has a hierarchical structure that mirrors the hierarchical structure of documents, and it also has two levels of attention mechanisms applied at the word and sentence level, enabling it to attend differentially to more and less important content when constructing the document representation. Dong and Zhang [24] employed a convolutional neural network (CNN) for the effect of automatically learning features. Kumar et al. [36] introduced a novel architecture for AES grading by combining three neural building modules: Siamese bidirectional LSTMs applied to a model and a student answer, a new pooling layer based on earth-mover distance across all hidden states from both LSTMs and a flexible final regression layer to output scores.

Especially in 2012, Kaggle launched a competition on AES called ‘Automated Student Assessment Prize’ (ASAP, <https://www.kaggle.com/c/asap-aes/data>) sponsored by the Hewlett Foundation. Hewlett hopes data scientists and machine learning specialists help solve a fast, effective and affordable

solutions for automated grading of student-written essays. At that time, the competitors mostly use machine learning algorithms which need handcrafted features extraction. Recently, many researchers have conducted a series of neural network-based AES studies using ASAP data sets. Alikaniotis et al. [11] employed a neural model to learn features for essay scoring automatically, which leverages a score-specific word embedding (SSWE) for word representations. Alikaniotis's experiment shows that SSWE is better for word embedding compared with other pre-trained word embeddings like word2vec, and LSTM [21] structure can capture the semantic information of the essay better than support vector machine (SVM). Taghipour et al. [12] developed an approach based on recurrent neural networks to learn the relation between an essay and its assigned score, without any feature engineering. They combined convolutional neural networks and recurrent neural networks for AES and demonstrated that LSTM and CNN are capable of outperforming systems that extensively require handcrafted features. In this paper, CNN was taken as an optional layer before inputting into LSTM, especially for those essays with a long length. Dong et al. [13] thought that, when using RNN and CNN to model input essays, the relative advantages of RNN and CNN cannot be compared based on the single vector representations of the essays. In addition, different parts of the essay can give different contribution for scoring. Therefore, they introduced the attention mechanism on the basis of CNN and RNN and found that the attention mechanism helps to find the keywords and sentences that contribute to judging the quality of essays. By building a hierarchical sentence-document model to represent essays, the model uses the attention mechanism to decide the relative weights of words and sentences automatically. The model can learn text representation with LSTMs which could model the coherence and coherence among a sequence of sentences. Furthermore, attention pooling is used to capture more relevant words and sentences that contribute to the final quality of essays. Borrowing the idea from Dong, we also use an attention mechanism at the LSTM layer. Tay et al. [14] described a new neural architecture that enhances vanilla neural network models with auxiliary neural coherence features and proposed a new SKIPFLOW mechanism. The SKIPFLOW model alleviates two problems: one is to alleviate the inability of current neural network architectures to model flow, coherence and semantic relatedness over time; the other one is to ease the burden of the recurrent model. To do so, the SKIPFLOW models the relationships between multiple snapshots of the LSTM's hidden state over time. As the model reads the essay, it models the semantic relationships between two points of an essay using a neural tensor layer. Eventually, multiple features of semantic relatedness are aggregated across the essay and used as auxiliary features for prediction. Then, they use the semantic relationships between multiple snapshots as auxiliary features for prediction. The SKIPFLOW mechanism based on LSTM architecture, which incorporates neural coherence features, implements an end-to-end AES approach. Inspired by this, furthermore, we put forward a self-information mechanism that is an extension from the essay to the essay and sample (rating criteria). Ref. [13,14] was also taken as a baseline in this paper.

3. Automated Essay Scoring

In this section, we define the input data, the evaluation metric, model architecture, and model training.

3.1. Description of Input

In supervised learning, we train the model by examples and their labels. In this paper, our inputs were reconstructed that contain the essay and sample, we need to construct a map to make a label for each new input, and after training, we need to be able to compute the original essay score by the inverse of the mapping. We define it officially as follows.

Let G be the score set, $i \in G$ is a score, $|G| = K$, $i \in [0, K]$; Let E be the essays set, e_i is the i th essay, $|E| = N$, $i \in [1, N]$; let S be the sample set, s_j is a sample, where j is a score, $j \in G$, and $|S| = C$ is the number of samples set S . Usually, C is less than or equal to K . Let v be the word embedding function; we simply mark $v(x)$ as the word embedding of essay X . $dist_{i,j} = v(e_i) - v(s_j)$ was marked as the distance information between e_i and s_j . Let f be the score function, for the essay e_i of which the score

is j , we mark $f(e_i) = j$; similarly, for the sample s_i of which the score is j , we mark $f(s_i) = j$. Mark $p_{i,j} = (e_i, s_j)$ as an input; then, set $P = \{p_{i,j} | e_i \in E, s_j \in S\}$ as the input dataset. Compared with the original essay dataset E , the new dataset P was expanded by C times.

We use score function φ to represent the score of input $p_{i,j}$; that is to say, we mark the score of $p_{i,j}$ as $\varphi(p_{i,j})$. We define $\varphi(p_{i,j})$ as:

$$\varphi(p_{i,j}) = Cf(e_i) + (C - 1)f(s_j) \quad (1)$$

where $C = |S|$ is the number of the sample set. Obviously, Equation (1) is a monotone function that is used to initialize the input's label. In particular, when $C = 1$, Equation (1) will degenerate into Equation (2):

$$\varphi(p_{i,j}) = f(e_i) \quad (2)$$

From Equation (1), we have

$$f(e_i) = \frac{\varphi(p_{i,j}) - (C - 1)f(s_j)}{C} \quad (3)$$

From Equations (1) and (3), we know that $f(e_i)$ is independent from $f(s_j)$, while, if we use $\tilde{\varphi}(p_{i,j})$ to denote the prediction value of $\varphi(p_{i,j})$, then $f(e_i)$ will be changed. We use $\tilde{f}(e_i)$ instead of the prediction value of $f(e_i)$ shown in Equation (3). Then, we have:

$$\tilde{f}(e_i) = \frac{1}{C} \sum_{s_j \in S} \frac{\tilde{\varphi}(p_{i,j}) - (C - 1)f(s_j)}{C} \quad (4)$$

Equations (3) and (4) are used to evaluate the test results of the model. In particular, when $C = 1$, Equation (4) will degenerate into Equation (5):

$$\tilde{f}(e_i) = \tilde{\varphi}(p_{i,j}) \quad (5)$$

Equations (2) and (5) are consistent in form. Here, we get the new input and their scores (labels). In the actual training, we can gradually increase the number of the sample set. Empirical results show that, usually, $C \leq 5$ can we get a good result; in rare cases, we need a further discussion at the circumstance of $C > 5$.

Now, we just use the samples as a part of the input. In two ways, we can get the samples. One is that the experts provide us with some samples with different ranks. The other one, also used in this paper, is to leverage the average value of the vector representation of all the essays which have the same rank to denote the sample vector. For the specific process, we get the samples vector according to Equation (6).

Assume that M is the number of all the essays with the same score j , e_i is one of them; then, the sample s_j 's vector was given by Equation (6)

$$v(s_j) = \frac{1}{M} \sum_{i=1}^M v(e_i) \quad (6)$$

where v is the word embedding function which we defined earlier in this section. For the different score j , we can easily get the sample vector $v(s_j)$. The experiment shows that such a way to get the sample is feasible.

3.2. Evaluation Metric of Output

Essay score predictions are evaluated using objective criteria. Quadratic weight Kappa (QWK) measures the agreement between two raters. Different from Kappa, QWK considers quadratic weights

by a quadratic weight matrix. This metric typically varies from 0 (only random agreement between raters) to 1 (complete agreement between raters). If there is less agreement between the raters than expected by chance, this metric may go below 0. The QWK is calculated between the automated scores for the essays and the resolved score for human raters on each set of essays. The official evaluation metric of ASAP Kaggle competition is QWK. Moreover, many follow-up researchers who use ASAP datasets to study AES take QWK as an evaluation metric. In this paper, our experiment dataset is the ASAP dataset as well. To make a better comparison with the relevant research, we adopt QWK as an evaluation metric too.

The QWK is defined as follows:

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \quad (7)$$

where i and j are the human rating and machine rating, respectively; N is the number of possible ratings. The matrix O is constructed over the essay ratings, such that $O_{i,j}$ corresponds to the number of essays that received a rating i by a human and a rating j by machine. A histogram matrix of expected ratings, E , is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that E and O have the same sum. From these three matrices W , E and O , the quadratic weighted kappa is calculated by Equation (8):

$$k = 1 - \frac{\sum W_{i,j} O_{i,j}}{\sum W_{i,j} E_{i,j}} \quad (8)$$

3.3. Model Architecture

In this section, we introduce the overall architecture of the model. Figure 2 shows the SBLSTMA model.

As shown in Figure 2, the SBLSTMA model consists of three modules: Ma, Mb, and Mc. The different module combinations receive different inputs. The combination of module Ma and Mc receives the essay only; Mb and Mc receive distance information; Ma, Mb, and Mc receive both essay and sample. The results of the three combinations are different. Usually, the third one is the best, the first one is the worst and the second is in the middle of the two. This confirms our previous hypothesis that the more input of scoring information, the better the scoring results. The detail will be discussed in Section 4.3.

3.3.1. Embedding Layer

Our model accepts a pair as a training instance each time. Each pair contains an essay e_i and a sample s_j as shown in Figure 2. The essay was represented as a fixed-length sequence in which we pad all sequences to the maximum length. Subsequently, each sequence is converted into a sequence of low dimensional vectors via the embedding layer. For the convenience of description, we use the function v to represent the process of word embedding. $v(e_i) \in \mathbb{R}^{|V| \times D}$ and $v(s_j) \in \mathbb{R}^{|V| \times D}$ are the word embedding outputs, where $|V|$ is the size of the vocabulary and D is the dimension of the word embedding.

After word embedding, we use $dist_{i,j} = v(e_i) - v(s_j)$ to represent the distance information between essay $v(e_i)$ and $v(s_j)$. We think that the distance information can be trained in the model and the new inputs make the model easier to converge, especially for those data sets with smaller data volumes.

3.3.2. Convolution Layer

This layer is an optional choice that you can skip this layer, especially for essays with a short length. We do the convolutional operation on prompt 8 which has the longest average length and has the fewest examples. The specific description of the dataset is in Section 4.1. After the dense representation of the long input sequence is calculated, it is fed into the LSTM layer of the network. For the long length essays, it might be beneficial for the network to extract local features from the

sequence before applying the recurrent operation. This optional characteristic can be achieved by applying a convolution layer on the output of the embedding layer.

3.3.3. LSTM Layer

The sequence of word embeddings obtained from the embedding layer (or convolution layer) is then passed into a long short-term memory (LSTM) network [21]:

$$h_t = LSTM(h_{t-1}, x_t) \quad (9)$$

where x_t and h_t are the input vectors at time t . The LSTM model is parameterized by output, input and forget gates, controlling the information flow within the recursive operation. The following equations formally describe the LSTM function:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (10)$$

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (11)$$

$$\tilde{c}_t = \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \quad (12)$$

$$\tilde{c}_t = i_t \circ \tilde{c}_t + f_t \circ \tilde{c}_{t-1} \quad (13)$$

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad (14)$$

$$h_t = o_t \circ \tanh(c_t) \quad (15)$$

At every time step t , LSTM outputs a hidden vector h_t that reflects the semantic representation of the essay at position t . The final representation of the essay is again feature-extracted in the self-information layer. In the experiment, we use bidirectional LSTM [37,38] and the attention mechanism [18] in the LSTM layer.

3.3.4. Self-Feature Layer

In this layer, we describe how to extract the self-feature from the vectors obtained from the bidirectional LSTM layer. We think that the essay vector e_i and distance information vector $dist_{i,j}$ should have some external relationships and the adjacent sentences in the essay should have some internal relationships, so we try to describe these relationships. Let he be the essay hidden layer, he_t denotes the vector at position t of he ; let hd be the distance information hidden layer, hd_t denotes the vector at position t of hd . Let δ be the length of sentence (we assume the lengths are the same in different sentences). Then, we compute the similarity of vector he at position t and $t + \delta$, and we call this similarity inner-feature:

$$inner - feature = \frac{he_t \cdot he_{t+\delta}}{|he_t| \cdot |he_{t+\delta}|} \quad (16)$$

Furthermore, we can compute the similarity in the same position t of vector he and hd , and we call this similarity a cross-feature:

$$cross - feature = \frac{he_t \cdot he_t}{|he_t| \cdot |he_t|} \quad (17)$$

‘.’ in Equations (16) and (17) are dot products. Then, both inner-feature and cross-feature are concatenated into vectors (we named these inner-feature and cross-feature directly) respectively and output to the next layer. Besides inner-feature and cross-features, we have two other main outputs: essay hidden layer and distance information hidden layer. We can do two kinds of processing for these two layers. One way is to take vectors at the last position of he and hd directly; the other way is to take the mean vector over time. We name these two vectors he -vector and hd -vector. As Figure 2 shows, four vectors are output to the full connect layer.

3.3.5. Fully-Connected Layer

Subsequently, we get four vectors obtained from the self-information layer: he-vector, hd-vector, inner-feature, and cross-feature. We can concatenate these four vectors into one. Then, we output the concatenate vector to the Softmax layer.

3.3.6. Softmax Layer

This layer is to classify the output of the fully connected layer. Its classification is achieved by Equation (18)

$$s(x) = \text{sigmoid}(w \cdot x + b) \quad (18)$$

where X is the input vector (the output of fully-connected layer), w is the weight vector, and b is the bias.

3.4. Training

The optimization algorithm we adopt is the Adaptive Gradient Algorithm [39] and the loss function we use is cross entropy loss function. It is defined as Equation (19)

$$H(x, \tilde{y}) = - \sum_i (p(y_i) \log q(\tilde{y}_i) + (1 - p(y_i)) \log(1 - q(\tilde{y}_i))) \quad (19)$$

where $Y, \tilde{y}$ are the true label and predicted label of the training essays, respectively; p, q are the probabilities. In addition, we use the dropout mechanism to avoid training overfitting. Our training method is to train a fixed number of epochs, and each epoch was trained, the QWK value is tested with the validation data; then, the parameters of the best QWK value are saved and used for the model predicting on the test dataset. The specific training hyper-parameters are listed in Table 1.

Table 1. Training hyper-parameters.

Layer	Parameter Name	Parameter Value
Embedding Layer	Pretrained embedding	GloVe 50-dimensional [40]
Convolution Layer	Window size	5
	Filters	20
LSTM Layer	Layers	1
	Hidden units	64
	Dropout	0.75
Self-feature Layer	Attention length	50
	Epochs	100–300
	Batch size	100–200
	Learning rate	0.01

4. Experiments

In this section, we describe the procedure of the experiment, including setup, baseline, results, and discussion.

4.1. Setup

The dataset we used is ASAP, a Kaggle competition dataset sponsored by the William and Flora Hewlett Foundation (Hewlett Foundation) in 2012. Many researchers have done the AES study on this dataset; choosing this dataset will help us to compare it with the previous experimental results. It contains eight prompts, each of which is a different genre. It was described in Table 2.

We take Stanford's publicly available GloVe 50-dimensional embedding [40] as pre-trained word embedding instead of training it ourselves. Because we think that using the third party pre-trained word embedding makes the model more generally and more opening, the data is tokenized with an

Natural Language Toolkit (NLTK, <http://www.nltk.org/>) tokenizer. For those words that can't be found in pre-trained word embedding, we replace them with UNKNOW. In addition, we adopt QWK mentioned in Section 3.2 to measure the output results and use 5-fold cross-validation to evaluate our model.

The software environment in the experimental program run is under Windows 10, Python 3.6, TensorFlow-gpu 1.4, and hardware is CPU: Intel(R) Xeon(R) L5640 @2.27GHz 2.26GHz; RAM: 16G; HDD:100G; GPU:GTX1080i.

Table 2. Statistics of ASAP dataset.

Prompt	Number of Essays	Average Length	Scores
1	1788	350	2–12
2	1800	350	1–6
3	1726	150	0–3
4	1772	150	0–3
5	1805	150	0–4
6	1800	150	0–4
7	1569	250	0–30
8	723	650	0–60

4.2. Baseline

To evaluate the performance of our model, we take two models that have the best Kappa value at present as our baselines. One is the SKIPFLOW model [14] that demonstrates state-of-the-art performances on the benchmark ASAP dataset. The other one is also based on ASAP called attention based recurrent convolutional neural network (LSTM-CNN-att) [13] which incorporates the latest neural algorithms such as attention mechanism, CNN, LSTM, etc., The two models both adopt 5-fold cross-validation to evaluate and the measure metric is QWK. The results of the two baseline model are listed in Table 3.

Table 3. The Quadratic weight Kappa (QWK) value compared with the baseline model.

Model	Prompts								Average
	1	2	3	4	5	6	7	8	
LSTM-CNN-att	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
SKIPFLOW	0.832	0.684	0.695	0.788	0.815	0.810	0.800	0.697	0.764
SBLSTMA	0.861	0.731	0.780	0.818	0.842	0.820	0.810	0.746	0.801

4.3. Results and Discussion

The results are listed in Table 3. Our model SBLSTMA outperforms both baseline models (LSTM-CNN-att and SKIPFLOW) by approximately 5% on average QWK (quadratic weighted Kappa). The results are statistically significant with $p < 0.05$ by 1-tailed t -test.

From Table 3, we know that the empirical results have been significantly improved. We think that this is because the knowledge of the rating criteria–distance information plays a very significant role. To explain it, we further decompose the model SBLSTMA to another two submodels. As described in Section 3.3, the model SBLSTMA consists of module Ma, Mb, and Mc, in which we can get three combined models: Ma + Mc, Mb + Mc, and Ma + Mb + Mc. Ma + Mc means the model receives the essay only without receiving the rating criteria information, and, during the training, it also computes the inner-feature information in the essay. Mb + Mc receives the distance information; during the training, it computes the inner-feature information in the distance information. Ma + Mb + Mc receives an essay and sample during the training; it computes inner-feature information and cross-feature information. We give the experimental results in Table 4, where the sample sets used were listed in Table 5.

The information distance is based on the sample set described in Section 3.1. It is directly related to the quality of the experimental results. We need to find the samples that could reflect the rating criteria as accurately as possible. The maximum element of sample set depends on the range of essay's score, but we can't select all the different score essays as the samples, especially for the essays with a large score range; if so, the training will be very time-consuming, and the results are not necessarily good. Empirical results show that, usually, for the dataset that has a narrow score range, we can take all the samples with different scores as a sample set, such as prompts 3, 4, 5, and 6; for the dataset that has a large score range, we can make some of the samples as a sample set, such as prompts 1, 2, 7, and 8. The way we get a sample set for the dataset that has a large score range according to the steps as follows:

- ① According to Equation (6), we compute all the samples s_j of each prompt.
- ② For each s_j in a prompt, make a pre-training under Mb + Mc and gives a sort, of which the order is sorted by the quality of Kappa value of the training results.
- ③ Take the first sample in the sort gives in step ② as the initial sample set. If the training results are less than the threshold (the result expectation was initialized before), then continue to add the second sample in the sort into the samples set, ..., until the results are greater than the threshold or all the samples are added into the sample set.

Take prompt 4, for example: the scores are 0, 1, 2, 3, and the corresponding samples are s_0, s_1, s_2, s_3 . By pre-training, we get a sort of $[s_2, s_1, s_3, s_0]$, which means that the training result of s_2 is the best one, s_1 is the second one, and so on. Then, we first take sample set $\{s_2\}$ as the initial sample set, $\{s_2, s_1\}$ as the second one, and so on. Table 5 shows the samples that we used in the experiment.

The results of each decomposed submodel listed in Table 4 shows that the Kappa value under model Mb + Mc is better than Ma + Mc. It means that the distance information as an input is useful for training. Such an input based on rating criteria contains more rating information, and it does reflect a certain distance between the essay and sample. For a more intuitive explanation, we provide the Kappa value diagrams of the first 100 epochs of all eight prompts under Ma + Mc and Mb + Mc shown in Figure 3.

Table 4. The Kappa value under different module combinations.

Module Combination	Prompts								Average
	1	2	3	4	5	6	7	8	
Ma + Mc	0.521	0.486	0.546	0.685	0.8	0.704	0.469	0.425	0.560
Mb + Mc	0.722	0.670	0.724	0.797	0.817	0.816	0.795	0.658	0.757
Ma + Mb + Mc	0.861	0.731	0.780	0.818	0.842	0.820	0.810	0.746	0.801
SBLSTMA (best of above)	0.861	0.731	0.780	0.818	0.842	0.820	0.810	0.746	0.801

Table 5. The sample set was used in the experiment.

Prompt	Sample Set	Prompts	Sample Set
1	$\{s_3, s_7, s_9\}$	5	$\{s_0, s_1, s_2, s_3, s_4\}$
2	$\{s_1, s_3, s_4, s_5\}$	6	$\{s_0, s_1, s_2, s_3, s_4\}$
3	$\{s_0, s_1, s_2, s_3\}$	7	$\{s_6, s_{10}, s_{16}, s_{22}, s_{24}\}$
4	$\{s_0, s_1, s_2, s_3\}$	8	$\{s_{29}, s_{46}\}$

Figure 3 intuitively shows that the Kappa value under Mb + Mc is better than the value under Ma + Mc. Furthermore, Table 6 shows the mean value and standard deviation value under Ma + Mc, Mb + Mc, and Ma + Mb + Mc. The mean value reflects how good the training results are, while standard deviation indicates the size of training space and the training stability. It is obvious that, based on a greater mean value, the greater standard deviation, the better the results.

Figure 3. Each prompt's Kappa value comparison under Ma + Mc and Mb + Mc at the first 100 epochs (prompt7 and prompt8 are 300epochs), where E denotes the output under Ma + Mc, D denotes the output under Mb + Mc. The X-axis and Y-axis denote epochs and Kappa value, respectively.

From Table 6, we can conclude that the training under Mb + Mc is better than the training under Ma + Mc, and the training under Ma + Mb + Mc is much more stable than the other two. Table 6 also tells the mean value and standard deviation of prompt 8 are relatively worse for the first 100 epochs. We consider this due to the fewest number of essays and the longest essay length and the largest size of the score range of prompt 8. For the other prompts, we can increase the number of samples set to improve the training effect, but, for prompt 8, we are not able to do this. When increasing the number of the sample set of prompt 8, the training process is not stable and is hard to converge. Therefore, in the experiment, the sample set number of prompt 8 is the smallest one.

Furthermore, from Table 4, we know that the results under Ma + Mb + Mc are the best. The average Kappa value of Ma + Mb + Mc is 0.44 greater than that of Mb + Mc. In particular, prompt 2 and prompt 3, which have the worst Kappa value in baseline models, were improved obviously in our model. We think that the input under this model contains more information: essay, distance information and self-feature mechanism, which are good for rating. The value of parameter δ , which denotes the length of sentence defined in Section 3.3.4, was fed as 10. To explain it clearly, we take prompt 2 and prompt 3 for example. We give these two prompts' Kappa value diagrams at the first 100 epochs under Ma + Mc, Mb + Mc, and Ma + Mb + Mc shown in Figure 4. From the figure, we can easily see that model Ma + Mb + Mc made a further improvement than model Ma + Mc and Mb + Mc.

Table 6. The mean value and standard deviation of each prompt's Kappa value at the first 100 epochs under Ma + Mc, Mb + Mc, and Ma + Mb + Mc.

		Prompts								Average
		1	2	3	4	5	6	7	8	
Mean	Ma + Mc	0.366	0.367	0.477	0.606	0.759	0.613	0.240	0.26	0.461
	Mb + Mc	0.614	0.493	0.542	0.711	0.694	0.691	0.26	0.313	0.540
	Ma + Mb + Mc	0.751	0.621	0.681	0.754	0.739	0.727	0.576	0.373	0.653
Std.Deviation	Ma + Mc	0.052	0.069	0.058	0.083	0.048	0.103	0.134	0.066	0.077
	Mb + Mc	0.139	0.148	0.111	0.119	0.192	0.209	0.218	0.090	0.153
	Ma + Mb + Mc	0.037	0.094	0.103	0.033	0.096	0.055	0.137	0.172	0.091

Figure 4. Kappa value comparison under Ma + Mc, Mb + Mc and Ma + Mb + Mc (prompt2 and prompt3), where E denotes the output under Ma + Mc, D denotes the output under Mb + Mc, M denotes the output under Ma + Mb + Mc. The X-axis and Y-axis denote epochs and Kappa value, respectively.

5. Conclusions

In this paper, we represent the rating criteria behind the essay by some samples and take it as a part of the input. Meanwhile, a self-feature mechanism at the LSTM output layer was provided as well. Then, we propose a novel model, a Siamese Bidirectional Long Short-Term Memory Architecture (SBLSTMA), to learn the text semantics and grade essays automatically. Our approach outperforms the baseline by approximately 5%. By decomposing the model, we find that the model with distance information input is much better than the one without. It means that it is feasible to represent rating criteria from samples. We also hypothesize that distance information derived from the difference between the examples and the mean example benefits all the other supervised learning methods. We will try using this approach in other fields in the coming future to check whether the hypothesis is right or not. In addition, we will also consider applying data augmentation technology to enhance the essay dataset of which the example is relatively small.

Author Contributions: All authors of this paper discussed the contents of the manuscript and actively contributed in the processing of implementation. Conceptualization, D.J.; Software, G.L.; Writing—Original Draft Preparation, G.L.; Writing—Review and Editing, B.-W.O.; Supervision, H.-C.K.; Funding Acquisition, B.-W.O. and G.S.C.

Funding: This research was supported by the following fund projects: Wenzhou Public Technology Planning Program (No. S20160018); the National Research Foundation of Korea Grant funded by the Korean Government(NRF-2016R1A2B1014843 and NRF-2017M3C4A7068188); the Ministry of Trade, Industry and Energy (MOTIE, Sejong, Korea) under the Industrial Technology Innovation Program (No.10063130); the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1A2B4007498); the MSIP (Ministry of Science, ICT and Future Planning), Gwacheon, Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2016-0-00313) supervised by the IITP (Institute for Information and communications Technology Promotion).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ellis, B. Grading essays by computer: Progress report. In Proceedings of the Invitational Conference on Testing Problems, New York, NY, USA, 29 October 1966; pp. 87–100.
2. Foltz, P.W.; Laham, D.; Landauer, T.K. Automated essay scoring: Applications to educational technology. *Proc. EdMedia* **1999**, *99*, 40–64.
3. Attali, Y.; Burstein, J. Automated essay scoring with e-rater v.2.0. *ETS Res. Rep. Ser.* **2004**, *2*, 1–21.
4. Larkey, L.S. Automatic essay grading using text categorization techniques. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 24–28 August 1998; pp. 90–95. [\[CrossRef\]](#)
5. Lawrence, M.R.; Liang, T. Automated essay scoring using bayes' theorem. *J. Technol. Learn. Assess.* **2002**, *1*, 3–21.
6. Phandi, P.; Chai, K.M.A.; Ng, H.T. Flexible domain adaptation for automated essay scoring using correlated linear regression. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 431–439.
7. Yannakoudakis, H.; Briscoe, T.; Medlock, B. A new dataset and method for automatically grading esol texts. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies—Volume 1, Association for Computational Linguistics, Portland, Oregon, 19–24 June 2011; pp. 180–189.
8. Chen, H.; He, B. Automated essay scoring by maximizing human-machine agreement. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, DC, USA, 18–21 October 2013; pp. 1741–1752.
9. Hinton, G.E. Learning Distributed Representations of Concepts. In Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, MA, USA, 15–17 August 1986; pp. 1–12.
10. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
11. Alikaniotis, D.; Yannakoudakis, H.; Rei, M. Automatic Text Scoring Using Neural Networks. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 715–725.
12. Taghipour, K.; Ng, H.T. A Neural Approach to Automated Essay Scoring. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1882–1891.
13. Dong, F.; Zhang, Y.; Yang, J. Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, BC, Canada, 3–4 August 2017; pp. 153–162.
14. Tay, Y.; Phan, M.C.; Tuan, L.A.; Hui, S.C. SKIPFLOW: Incorporating Neural Coherence Features for End-to-End Automatic Text Scoring. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence), New Orleans, LV, USA, 2–7 February 2018.
15. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.

16. Lee, K.; Han, S.; Han, S.; Myaeng, S. A discourse-aware neural network-based text model for document-level text classification. *J. Inf. Sci.* **2017**, *44*, 715–735. [CrossRef]
17. Santos, C.N.D.; Gatti, M. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In Proceedings of the 25th International Conference on Computational Linguistics, Dublin, Ireland, 23–29 August 2014; pp. 69–78.
18. Yin, W.; Ebert, S.; Schütze, H. Attention-Based Convolutional Neural Network for Machine Comprehension. In Proceedings of the 2016 NAACL Human-Computer Question Answering Workshop, San Diego, CA, USA, 12–17 June 2016; pp. 15–21.
19. Zhang, Y.; Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv* **2015**, arXiv:1510.03820.
20. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv* **2015**, arXiv:1506.00019.
21. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
22. Zhang, Y.D.; Muhammad, K.; Tang, C. Twelve-layer deep convolutional neural network with stochastic pooling for tea category classification on GPU platform. *Multimed. Tools Appl.* **2018**, *77*, 22821. [CrossRef]
23. Wang, S.H.; Lv, Y.D.; Sui, Y.; Liu, S.; Wang, S.J.; Zhang, Y.D. Alcoholism Detection by Data Augmentation and Convolutional Neural Network with Stochastic Pooling. *J. Med. Syst.* **2018**, *42*, 2. [CrossRef] [PubMed]
24. Dong, F.; Zhang, Y. Automatic Features for Essay Scoring—An Empirical Study. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–4 November 2016; pp. 1072–1077.
25. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
26. Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Kaiser, L. Universal Transformers. *arXiv* **2018**, arXiv:1807.03819.
27. Mueller, J.; Thyagarajan, A. Siamese Recurrent Architectures for Learning Sentence Similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
28. Landauer, T.K.; Foltz, P.W.; Laham, D. Introduction to Latent Semantic Analysis. *Discourse Process.* **1998**, *25*, 259–284. [CrossRef]
29. Tandalla, L. Scoring Short Answer Essays. ASAP Short Answer Scoring Competition—Luis Tandalla's Approach. In *ASAP Short Answer Scoring Competition—Luis Tandalla's Approach*; 2012, Volume 9. Available online: <https://kaggle2.blob.core.windows.net/competitions/kaggle/2959/media/TechnicalMethodsPaper.pdf> (accessed on 14 November 2018).
30. Mehmood, A.; On, B.; Lee, I.; Choi, G.S. Prognosis essay scoring and article relevancy using multi text features and machine learning. *Symmetry* **2017**, *9*, 11. [CrossRef]
31. Drolia, S.; Rupani, S.; Agarwal, P.; Singh, A. Automated Essay Rater using Natural Language Processing. *Int. J. Comput. Appl.* **2017**, *163*, 44–46. [CrossRef]
32. McNamara, D.S.; Crossley, S.A.; Roscoe, R.D.; Allen, L.K.; Dai, J. A hierarchical classification approach to automated essay scoring. *Assess. Writ.* **2015**, *23*, 35–59. [CrossRef]
33. Fauzi, M.A.; Utomo, D.C.; Setiawan, B.D. Automatic Essay Scoring System Using N-Gram and Cosine Similarity for Gamification Based E-Learning. In Proceedings of the International Conference on Advances in Image Processing, Bangkok, Thailand, 25–27 August 2017; pp. 151–155.
34. Zupanc, K.; Bosnić, Z. Automated essay evaluation with semantic analysis. *Knowl.-Based Syst.* **2017**, *120*, 118–132. [CrossRef]
35. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical Attention Networks for Document Classification. In Proceedings of the NAACL-HLT 2016, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.
36. Kumar, S.; Chakrabarti, S.; Roy, S. Earth Mover's Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading. In Proceedings of the Twenty Sixth International Joint Conference on Artificial Intelligence (IJCAI17), Melbourne, Australia, 19–25 August 2017.
37. Graves, A.; Fernández, S.; Schmidhuber, J. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. *ICANN* **2005**, 3697, 799–804.
38. Schuster, M.; Paliwal, K.K. Bidirectional Recurrent Neural Networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]

39. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
40. Jeffrey, P.; Richard, S.; Christopher, D.M. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp.1532–1543.

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).