Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring

Fei Dong and **Yue Zhang**[*] and **Jie Yang**
Singapore University of Technology and Design
{fei_dong, jie_yang}@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Abstract

Neural network models have recently been applied to the task of automatic essay scoring, giving promising results. Existing work used recurrent neural networks and convolutional neural networks to model input essays, giving grades based on a single vector representation of the essay. On the other hand, the relative advantages of RNNs and CNNs have not been compared. In addition, different parts of the essay can contribute differently for scoring, which is not captured by existing models. We address these issues by building a hierarchical sentence-document model to represent essays, using the attention mechanism to automatically decide the relative weights of words and sentences. Results show that our model outperforms the previous state-of-the-art methods, demonstrating the effectiveness of the attention mechanism.

1 Introduction

Automatic essay scoring (AES) is the task of automatically assigning grades to student essays. It can be highly challenging, requiring not only knowledge on spelling and grammars, but also on semantics, discourse and pragmatics. Traditional models use sparse features such as bag-of-words, part-of-speech tags, grammar complexity measures, word error rates and essay lengths, which can suffer from the drawbacks of time-consuming feature engineering and data sparsity.

Recently, neural network models have been used for AES (Alikaniotis et al., 2016; Dong and Zhang, 2016; Taghipour and Ng, 2016), giving better results compared to statistical models with handcrafted features. In particular, distributed word representations are used for the input, and a neural network model is employed to combine word information, resulting in a single dense vector form of the whole essay. A score is given based on a non-linear neural layer on the representation. Without handcrafted features, neural network models have been shown to be more robust than statistical models across different domains (Dong and Zhang, 2016).

Both recurrent neural networks (Williams and Zipser, 1989; Mikolov et al., 2010) and convolutional neural networks (LeCun et al., 1998; Kim, 2014) have been used for modelling input essays. In particular, Alikaniotis et al. (2016) and Taghipour and Ng (2016) use a single-layer LSTM (Hochreiter and Schmidhuber, 1997) over the word sequence to model the essay, and Dong and Zhang (2016) use a two-level hierarchical CNN structure to model sentences and documents separately. It has been commonly understood that CNNs can capture local ngram information effectively, while LSTMs are strong in modelling long history. No previous work has compared the effectiveness of LSTMs and CNNs under the same settings for AES. To better understand the contrast, we adopt the two-layer structure of Dong and Zhang (2016), comparing CNNs and LSTMs for modelling sentences and documents.

Not all sentences contribute equally to the scoring of a given essay, and not all words contribute equally within a sentence. We adopt the neural attention model (Xu et al., 2015; Luong et al., 2015) to automatically calculate weights for convolution features of CNNs and hidden state values of LSTMs, which has been used for obtaining the most pertinent information for machine translation (Luong et al., 2015), sentiment analysis (Shin et al., 2016; Wang et al., 2016; Liu and Zhang, 2017) and other tasks. In our case, the attention mechanism can intuitively select sentences and grams that are more aligned with the props or obviously incorrect. To our knowledge, no prior

[*] Corresponding author.

work has investigated the effectiveness of attention models for AES.

Results show that CNN is relatively more effective for modelling sentences, and LSTMs are relatively more effective for modelling documents. This is likely because local ngram information are more relevant to the scoring of sentence structures, and global information is more relevant for scoring document level coherence. In addition, attention gives significantly more accurate results. Our final model achieves the best result reported on the ASAP[1] test set. We release our code at `https://github.com/feidong1991/aes`.

2 Automatic Essay Scoring

2.1 Task

The task of AES is usually treated as a supervised learning problem, typical models of which can be divided into three categories: classification, regression and preference ranking. In the classification scenario, scores are divided into several categories, each score or score range is regarded as one class and the ordinary classification models are employed such as Naive Bayes (NB) and SVMs (Larkey, 1998; Rudner and Liang, 2002). In the regression scenario, each score is treated as continous values for the essay and regression models are considered, like linear regression, Bayesian linear ridge regression (Attali and Burstein, 2004; Phandi et al., 2015). In the preference ranking scenario, AES task is considered as a ranking problem in which pair-wise ranking and list-wise ranking are employed (Yannakoudakis et al., 2011; Chen and He, 2013; Cummins et al., 2016). The former considers the ranking between each pair of essays, while the latter considers the absolute ranking of each essay in the whole set.

Formally, an AES model is trained to minimize the difference between its automatically output scores and human given scores on a set of training data:

$$\begin{aligned} &\min \sum_{i=1}^{N} f(y_i^*, y_i), \\ &\text{s.t. } y_i = g(t_i), i = 1, 2, ..., N \end{aligned} \tag{1}$$

where N is the total number of essays in the training set, y_i^* and y_i are the golden score assigned by human raters and prediction score made by the AES system of i-th essay in the set respectively, t_i is feature representation of i-th essay, f is the metric function between golden score and prediction score, such as mean square error and mean absolute error, and g is the mapping function from feature t_i to score y_i.

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

2.2 Evaluation Metric

Many measurement metrics have be adopted to assess the quality of AES systems, including Pearson's correlation, Spearman's ranking correlation, Kendall's Tau and kappa, especially quadratic weighted kappa (QWK). We follow the Automated Student Assessment Prize (ASAP) competition official criteria which takes QWK as evaluation metric, which is also adopted as evaluation metric in (Dong and Zhang, 2016; Taghipour and Ng, 2016; Phandi et al., 2015).

Kappa measures inter-raters agreement on the qualitive items, here inter-raters refer to AES system and human rater. QWK is modified from kappa which takes quadratic weights. The quadratic weight matrix in QWK is defined as:

$$W_{i,j} = \frac{(i-j)^2}{(R-1)^2}, \tag{2}$$

where i and j are the reference rating (assigned by a human rater) and the system rating (assigned by an AES system), respectively, and R is the number of possible ratings.

An observed score matrix O is calculated such that $O_{i,j}$ refers to the number of essays that receive a rating i by the human rater and a rating j by the AES system. An expected score matrix E is calculated as the outer product of histogram vectors of the two (reference and system) ratings. The matrix E needs to be normalized such that the sum of elements in E and the sum of elements in O keep the same. Finally, given the three matrices W, O and E, the QWK value is calculated according to Equation 3:

$$\kappa = 1 - \frac{\sum W_{i,j} O_{i,j}}{\sum W_{i,j} E_{i,j}} \tag{3}$$

We evaluate our model using QWK as the metric, and perform one-tailed t-test to determine the significance of improvements.

3 Model

We employ a hierarchical neural model similar to the sentence-document model of Dong and Zhang

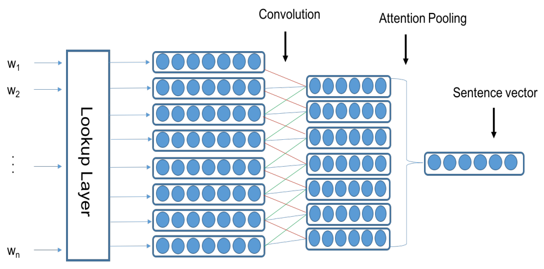

Figure 1: Sentence representation using ConvNet and attention pooling

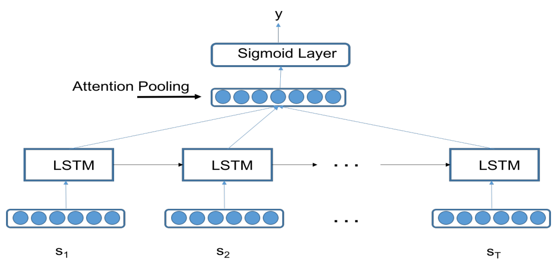

Figure 2: Document (Text) representation using LSTM and attention pooling

(2016) who consider essay script as being composed of sentence sequences rather than word sequences. Different from their model, our neural model learns text representation with LSTMs, which could model the coherence and coreference among sequences of sentences (i.e. capturing more global information compared to CNNs). Besides, attention pooling is both used on words and sentences, which aims to capture more relevant words and sentences that contribute to the final quality of essays.

We investiage two types of word representations, one being character-based embedding, which utilizes a convolutinal layer to learn word representations from raw characters, and the other being word embedding.

Characters For character-based word representation, we employ a convolutional layer over characters in each word, followed by max-pooling and average-pooling layers. The concatenation of max-pooling and average-pooling forms the final word representation for each word.

Let $\mathbf{c}_i^1, \mathbf{c}_i^2, ..., \mathbf{c}_i^m$ be one-hot representation of characters that make up the word w_i, we have the following word representation for w_i using make-up characters:

$$\mathbf{x}_{c_i} = \mathbf{E}_c \mathbf{c}_i \quad (4)$$

$$\mathbf{z}_{c_i}^j = f(\mathbf{W}_c \cdot [\mathbf{x}_{c_i}^j : \mathbf{x}_{c_i}^{j+h-1}] + \mathbf{b}_c) \quad (5)$$

$$\tilde{\mathbf{x}}_i = \max_j \mathbf{z}_{c_i}^j \quad (6)$$

$$\hat{\mathbf{x}}_i = \underset{j}{avg}\ \mathbf{z}_{c_i}^j \quad (7)$$

$$\mathbf{x}_i = \tilde{\mathbf{x}}_i \oplus \hat{\mathbf{x}}_i, \quad (8)$$

where $\mathbf{E}_c$ is the embedding matrix, $\mathbf{x}_{c_i}$ is the embedding vector for $\mathbf{c}_i$, $\mathbf{z}_{c_i}^j$ is the feature map for j-th character in i-th word w_i after convolutional layer, $\mathbf{W}_c$, $\mathbf{b}_c$ are the weights matrix and bias vector respectively, h specifies the window size in the convolutional layer and f is the activation function, here hyperbolic tangent function $tanh$ is used. $\tilde{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_i$ are max-pooling and average-pooling vectors over $\mathbf{z}_{c_i}^j$, and the final word w_i's representation $\mathbf{x}_i$ is the concatenation of $\tilde{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_i$.

Words Given a sentence of words sequence $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n$, an lookup layer map each $\mathbf{w}_i$ into a dense vector $\mathbf{x}_i, i = 1, 2, ..., n$.

$$\mathbf{x}_i = \mathbf{E}\mathbf{w}_i, i = 1, 2, ..., n \quad (9)$$

where $\mathbf{w}_i$ is one-hot representation of the i-th word in the sentence, $\mathbf{E}$ is the embedding matrix, $\mathbf{x}_i$ is the embedding vector of i-th word.

3.1 Sentence Representation

After obtaining the word representations $\mathbf{x}_i, i = 1, 2, ..., n$, we employ a convolutional layer on each sentence:

$$\mathbf{z}_i = f(\mathbf{W}_z \cdot [\mathbf{x}_i^j : \mathbf{x}_i^{j+h_w-1}] + \mathbf{b}_z), \quad (10)$$

where $\mathbf{W}_z$, $\mathbf{b}_z$ are weight matrix and bias vector, respectively, h_w is the window size in the convolutional layer and $\mathbf{z}_i$ is the result feature representation.

Above the convolutional layer, attention pooling is employed to acquire a sentence representation. The structure of a sentence representation is depicted in Figure 1. The details of convolutional and attention pooling layers are defined in the following equations.

$$\mathbf{m}_i = tanh(\mathbf{W}_m \cdot \mathbf{z}_i + \mathbf{b}_m) \quad (11)$$

$$u_i = \frac{e^{\mathbf{w}_u \cdot \mathbf{m}_i}}{\sum e^{\mathbf{w}_u \cdot \mathbf{m}_j}} \quad (12)$$

$$\mathbf{s} = \sum u_i \mathbf{z}_i, \quad (13)$$

where $\mathbf{W}_m$, $\mathbf{w}_u$ are weight matrix and vector, respectively, $\mathbf{b}_m$ is the bias vector, $\mathbf{m}_i$ and u_i are attention vector and attention weight respectively for i-th word. $\mathbf{s}$ is the final sentence representation, which is the weighted sum of all the word vectors.

3.2 Text Representation

A recurrent layer is used to compose a document (text) representation similar to the models of Alikaniotis et al. (2016) and Taghipour and Ng (2016). The main difference is that both earlier work treat the essay script as a sequence of words rathter than a sequence of sentences. Alikaniotis et al. (2016) use score-specific word embeddings as word features and take the last hidden state of LSTM as text representation. Taghipour and Ng (2016) take the average value over all the hidden states of LSTM as text representation. In contrast to the previous LSTM models, we use LSTM to learn from sentence sequences and attention pooling on the hidden states of LSTM to obtain the contribution of each sentence to the final quality of essays. The structure of a text representation using LSTM is depicted in Figure 2.

Long short-term memory units are the modified recurrent units which are proposed to handle the problem of vanishing gradients effectively (Hochreiter and Schmidhuber, 1997; Pascanu et al., 2013). LSTMs use gates to control information flow, preserving or forgetting information for each cell units. In order to control information flow when processing a vector sequence, an input gate, a forget gate and an output gate are employed to decide the passing of information at each time step. Assuming that an essay script consists of T sentences, s_1, s_2, ..., s_T with $\mathbf{s}_t$ being the feature representation of t-th word s_t, we have LSTM cell units addressed in the following equations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot \mathbf{s}_t + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot \mathbf{s}_t + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\tilde{\mathbf{c}}_t &= tanh(\mathbf{W}_c \cdot \mathbf{s}_t + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{i}_t \circ \tilde{\mathbf{c}}_t + \mathbf{f}_t \circ \mathbf{c}_{t-1} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot \mathbf{s}_t + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \circ tanh(\mathbf{c}_t),
\end{aligned} \tag{14}$$

where $\mathbf{s}_t$ and $\mathbf{h}_t$ are the input sentence and output sentence vectors at time t, respectively. $\mathbf{W}_i$, $\mathbf{W}_f$,$\mathbf{W}_c$, $\mathbf{W}_o$, $\mathbf{U}_i$, $\mathbf{U}_f$, $\mathbf{U}_c$, and $\mathbf{U}_o$ are weight matrices and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_c$, and $\mathbf{b}_o$ are bias vectors. The symbol $\circ$ denotes element-wise multiplication and σ represents the sigmoid function.

After obtaining the intermediate hidden states of LSTM $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T$, we use another attention pooling layer over the sentences to learn the final text representation. The attention pooling helps to acuquire the weights of sentences' contribution to final quality of the text. The attention pooling over sentences is addressed as:

$$\mathbf{a}_i = tanh(\mathbf{W}_a \cdot \mathbf{h}_i + \mathbf{b}_a) \tag{15}$$

$$\alpha_i = \frac{e^{\mathbf{w}_\alpha \cdot \mathbf{a}_i}}{\sum e^{\mathbf{w}_\alpha \cdot \mathbf{a}_j}} \tag{16}$$

$$\mathbf{o} = \sum \alpha_i \mathbf{h}_i, \tag{17}$$

where $\mathbf{W}_a$, $\mathbf{w}_\alpha$ are weight matrix and vector respectively, $\mathbf{b}_a$ is the bias vector, $\mathbf{a}_i$ is attention vector for i-th sentence, and α_i is the attention weight of i-th sentence. $\mathbf{o}$ is the final text representation, which is the weighted sum of all the sentence vectors.

Finally, one linear layer with sigmoid function applied on the text representation to get the final score as described in Equation 18.

$$y = sigmoid(\mathbf{w}_y \mathbf{o} + \mathbf{b}_y) \tag{18}$$

where $\mathbf{w}_y$, $\mathbf{b}_y$ are weight vector and bias vector, y is the final score of the essay.

4 Training

Objective We use mean square error (MSE) loss, which is also used in previous models. MSE is widely used in regression tasks, which measures the average value of square error between gold standard scores y_i^* and prediction scores y_i assigned by the AES system among all the essays. Given N essays, we calculate MSE according to Equation 19.

$$mse(y, y^*) = \frac{1}{N} \sum_{i=1}^{N} (y_i - y_i^*)^2 \tag{19}$$

The model is trained on a fixed number of epochs and evaluated on the development set at every epoch. We set the batch size to 10 and the best model is selected on the performance of quadratic weighted kappa on the development set. The details of model hyper-parameters are listed in Table 1.

Character Embeddings The character embeddings are initialized with uniform distribution from [-0.05, 0.05]. The dimension of character embeddings is set to 30. During the training process, character embeddings are fine-tuned.

Layer	Parameter Name	Parameter Value
Lookup	char embedding dim	30
	word embedding dim	50
CNN	window size	5
	number of filters	100
LSTM	hidden units	100
Dropout	dropout rate	0.5
	epochs	50
	batch size	10
	initial learning rate η	0.001
	momentum	0.9

Table 1: Hyper-parameters

Set	#Essays	Genre	Avg Len.	Range	Med.
1	1783	ARG	350	2-12	8
2	1800	ARG	350	1-6	3
3	1726	RES	150	0-3	1
4	1772	RES	150	0-3	1
5	1805	RES	150	0-4	2
6	1800	RES	150	0-4	2
7	1569	NAR	250	0-30	16
8	723	NAR	650	0-60	36

Table 2: Statistics of the ASAP dataset; Range refers to score range and Med. refers to median scores. For genre, ARG specifies *argumentative* essays, RES means *response* essays and NAR denotes *narrative* essays.

Word Embeddings We take the Stanford's publicly available GloVe 50-dimensional embeddings[2] as word pretrained embeddings, which are trained on 6 billion words from Wikipedia and web text (Pennington et al., 2014). During the training process, word embeddings are fine-tuned.

Optimization We use RMSprop (Dauphin et al., 2015) as our optimizer to train the whole model. The initial learning rate η is set to 0.001 and momentum is set to 0.9. Dropout regularization is used to avoid overfitting and drop rate is 0.5.

5 Experiments

5.1 Setup

Data The ASAP dataset is used as evaluation data of our AES system. The ASAP dataset consists of 8 different prompts of genres as listed in Table 2.

There are no released labeled test data from the ASAP competition, thus we separate test set and development set from the training set. The partition exactly follows the setting used by Taghipour and Ng (2016), which adopts 5-fold cross-validation, in each fold, 60% of the data is used as our training set, 20% as the development set, and 20% as the test set. The data is tokenized with NLTK[3] tokenizer. All the words are converted to lowercase and the scores are scaled to the range [0, 1]. During evaluation phase, the scaled scores are rescaled to original integer scores, which are used to calculate evaluation metric QWK values. The vocabulary size of the data is set to 4000, by following Taghipour and Ng (2016), selecting the most 4000 frequent words in the training data and treating all other words as unknown words.

Baseline models We take LSTM with Mean-over-Time Pooling (LSTM-MoT) (Taghipour and Ng, 2016) and hierarchical CNN (CNN-CNN-MoT) (Dong and Zhang, 2016) as our baselines. The former takes the essay script as a sequence of words, which is text-level model and the latter regards the script as a sequence of sentences, which is sentence-level model.

LSTM-MoT uses one layer of LSTM over the word sequences, and takes the average pooling over all time-step states as the final text representation, which is called Mean-over-Time (MoT) pooling (Taghipour and Ng, 2016). A linear layer with sigmoid function follows the MoT layer to predict the score of an essay script.

CNN-CNN-MoT uses two layers of CNN, in which one layer operates over each sentence to obtain representation for each sentence and the other CNN is stacked above, followed by mean-over-time pooling to get the final text representation.

LSTM-MoT is the current state-of-the-art neural model on the text-level and CNN-CNN-MoT is a state-of-the-art model on the sentence-level. Besides, LSTM-LSTM-MoT and LSTM-CNN-MoT are adopted as another two baseline models. The former model takes LSTMs to represent both sentences and texts, and the latter uses CNN representing sentences and LSTM representing texts. Both models use MoT pooling and are sentence-level models. We compare our model (LSTM-CNN-attent) with the baseline models to study CNN representing sentences and LSTM representing texts.

5.2 Results

The results are listed in Table 3. Our model LSTM-CNN-attent outperforms the baseline model CNN-CNN-MoT by 3.0%, LSTM-MoT by 2.2% on average quadratic weighted

[2]http://nlp.stanford.edu/projects/glove/

[3]http://www.nltk.org

Prompts	LSTM-MoT	CNN-CNN-MoT	LSTM-CNN-att
1	0.818	0.805	0.822
2	0.688	0.613	0.682
3	0.679	0.662	0.672
4	0.805	0.778	0.814
5	0.808	0.800	0.803
6	0.817	0.809	0.811
7	0.797	0.758	0.801
8	0.527	0.644	0.705
Avg.	0.742	0.734	**0.764**

Table 3: Comparison of quadratic weighted kappa between different models on the test data.

LSTM-CNN-attent	Average QWK
char	0.738
word	**0.764**
word + char	0.761

Table 4: Comparison of quadratic weight kappa using different features on the test data.

kappa. The results are statistically significant with $p < 0.05$ by one-tailed t-test. Even compared with the ensemble model used by Taghipour and Ng (2016), which ensembles 10 instances of CNN and LSTM of different initializations, our model still achieves 0.3% improvement on QWK.

5.3 Analysis

We perform several development experiments to verify the effectiveness of sentence-document model and text representation with LSTM and attention pooling.

Characters and Words We explore a convolutional layer to learn word representation from char-based CNN to replace word embeddings. In Table 4, we compare the performance of using character embeddings, word embeddings and concatenation of two embeddings. Empirical results show that with only character embedding features, the performance of our model outperforms CNN-CNN-MoT, and is close to LSTM-MoT. However, there is still a big gap between character embedding and word embedding models, which could come from the fact that we use pretrained word embeddings, which helps improve the performance. When both the word and character embeddings are used, the performance does not improve. One possible explanation is that the ASAP dataset is rather small given the model parameters, which has a potential for overfitting if both words and characters are used.

Model	Model Type	Pooling	Avg QWK
LSTM-MoT	document-level	MoT	0.742
LSTM-attent	document-level	attention	0.731
CNN-CNN-MoT	sentence-level	MoT	0.734
LSTM-LSTM-MoT	sentence-level	MoT	0.758
LSTM-CNN-MoT	sentence-level	MoT	0.759
LSTM-LSTM-attent	sentence-level	attention	0.762
LSTM-CNN-attent	sentence-level	attention	**0.764**

Table 5: Comparison between different model types and pooling methods on the test data (only word embeddings used).

Granularity The previous model LSTM-MoT tackles the AES task by treating each essay script as a sequence of words, which makes an essay an extra long sequence. The word number of one essay usually exceeds several hundreds, which makes it difficult to directly use LSTM to learn text representation if only last hidden state is used. It has been verified by Taghipour and Ng (2016) that LSTM with Mean-over-Time pooling outperforms LSTM with only last state. Though MoT pooling could alleviate this problem by considering all the states information, the model is still built on text-level rather than sentence-level. Both LSTM-CNN-MoT and LSTM-LSTM-MoT are sentence-document models. The former explores CNN for sentence representation and LSTM for text representation, and the latter use both LSTMs for sentence and text representation with MoT pooling. In Table 5, LSTM-CNN-MoT and LSTM-LSTM-MoT obtain large improvements compared to LSTM-MoT, especially for prompt 8 essays, of which the average script length is the biggest. This shows that sentence-document model tends to be more effective for long essays.

Local vs Global In Table 5, we compare LSTM-CNN-MoT with CNN-CNN-MoT to analyze the effectiveness of LSTM for text representation over CNN. Both CNN-CNN-MoT and LSTM-CNN-MoT learn hierarchical sentence-document representations. The former employs two-level CNNs for sentence representation and text representation respectively, and mean-over-time pooling is both used after two-level CNNs. The latter employs a CNN to learn sentence representation at the bottom, stacks one layer of LSTM above to learn text representation, and mean-over-time pooling is also used after CNN and LSTM. Compared with CNN-CNN-MoT in Table 5, LSTM-CNN-MoT gives a big improvement. We believe that on text representation layer, LSTMs can learn more global information, such as sentence coherence, while CNNs learn more local features, such as n-grams and bag-of-words. LSTM-LSTM-MoT outperforms CNN-CNN-MoT and gets slightly worse than LSTM-CNN-MoT, which also shows that LSTM is relatively more effective for modeling the documents.

Mean-over-Time vs Attention pooling We compare the two pooling methods adopted in our model, namely mean-over-time pooling and attention pooling in Table 5. The pooling layers are used after both CNN and LSTM layer to get sentence representation and text representation respectively. We find that by attending over words and sentences, we achieve the best performance, which demonstrates that attention pooling helps find the key words and sentences that contribute to judging quality of essays. In contrast to MoT, each word and sentence will be treated equally, which violates human raters' assessing process. Since our model is based on the sentence-level rather than the text-level, we can exert attention pooling to focus on pertinent words and sentences. Note that attention can be weakened when used for an extra long sequence, such as the scenario in the text-level model. Taghipour and Ng (2016) tried to attend over words on their one-layer LSTM model, but failed to beat the baseline model that employs mean-over-time pooling, because of that text-level model contains a quite long sequence of words, which may weaken the effect of attention. On the contrary, sentence-level model contains relatively short sequences of words, which makes attention more effective.

In Table 6, we briefly show two prompts from the AES data, namely Prompt 4 and Prompt 8. Prompt 4 asks for a response based on the last paragraph of a given story and Prompt 8 requires a true story about laughter. Prompt 4 has few number of sentences compared with Prompt 8. For convenience, we take Prompt 4 essays as our examples to analyze the attention mechanism on sentences, and Prompt 8 essays to analyze the attention mechanism on words n-grams. In Table 7, we list all five sentences in order that make up of one response essay from test set in Prompt 4. Each

	Prompt Contents
Prompt 4	Read the last paragraph of the story. "When they come back, Saeng vowed silently to herself, in the spring, when the snows melt and the geese return and this hibiscus is budding, then I will take that test again." Write a response that explains why the author concludes the story with this paragraph. In your response, include details and examples from the story that support your ideas. [5]
Prompt 8	We all understand the benefits of laughter. For example, someone once said, "Laughter is the shortest distance between two people." Many other people believe that laughter is an important part of any relationship. Tell a true story in which laughter was one element or part.

Table 6: Contents of Prompt 4 and Prompt 8

sentence is associated with its attention weight as shown in the table. The 4-th sentence has the biggest attention weight among the five sentence, then followed by the 5-th sentence. Intuitively, we know the 4-th and 5-th sentence can give strong supporting ideas to illustrate why the author concludes the story with the last paragraph. Therefore, it proves that our attention mechanism on sentences captures the key sentences to represent essays indeed.

In Table 8, we list three example sentences in one essay from the prompt 8 test data. The essay is written by students given the prompt described in the Table 6. The highlighted words are the 5-grams[4] that have the highest attention score. It can be easily seen that the highlighted 5-grams are the most relevant to the prompt, which demonstrates our attention-pooling takes an effect on learning sentence representation.

6 Related Work

The first AES system dates back to 1960s (Page, 1968, 1994) when Project Essay Grade (PEG) was developed. Following that, IntelliMetric 2, Intelligent Essay Assessor (IEA) (Landauer et al., 1998; Foltz et al., 1999) have come out. IEA uses Latent Semantic Analysis (LSA) to calculate the semantic similarity between texts and assigns a score to test text based on the score of the training text which is most similar to the given test text. Other commercial system, like e-rater system (Attali and

[4]Since we use a window size of 5 in CNN layer, the attention pooling after CNN layer is attending over 5-grams features.

[5]As Prompt 4 contains a long story in the prompt descriptions, we only pick up the most relevant contents here.

No.	Sentences	Attention weights
1	there was a specific reason as to why the author concluded the story with that quote .	0.17568
2	the author wanted to show how the plant gave saeng a new sense of determination .	0.20358
3	saeng previously was upset and tearing the plant apart .	0.19651
4	but it seemed that she realized how the plant was able to bud to the <unk> and survive .	0.21264
5	so she now was determined to <unk> the <unk> as well and retake the test she failed .	0.21159

Table 7: Attention weights of sentences coming from one student essay in Prompt 4 (The darkness of blue indicates the relative magnitude of attention weights.

Prompt 8	
Example 1	when i was a young boy i used to laugh at anything i could , but as a kid who did n't ?
Example 2	as i got older and grew more , i developed a great sense of humor that to my advantage made me a young people <unk> .
Example 3	i grew more and more <unk> a stronger , more confident sense of humor .

Table 8: Examples of attention pooling over n-grams features in Prompt 8 (The first row specifies the prompt given by the essay designer).

Burstein, 2004), has been deployed in the English language test, such as Test of English as a Foreign Language (TOEFL) and Graduate Record Examination (GRE). Step-wise linear regression is employed in the e-rater systems along with grammatical errors, lexical complexity as handcrafted features.

In the research literature, Larkey (1998) uses Naive Bayes model and takes AES as a classification model. Rudner and Liang (2002) explore multinomial Bernoulli Naive Bayes models to classify texts into several categories of text quality based on content and style features. Chen et al. (2010) formulates the AES task into a weakly supervised framework and employ a voting algorithm.

Other recent work formulate the task as a preference ranking problem (Yannakoudakis et al., 2011; Phandi et al., 2015). Yannakoudakis et al. (2011) formulate AES as a pairwise ranking problem by ranking the order of pair essays based on their quality. Features consist of word n-grams, deep linguistic features, including grammatical complexity, POS n-grams and parsing trees features. Chen and He (2013) formulate AES into a list-wise ranking problem by considering the order relation among the whole essays. Features contain syntactical features, grammar and fluency features as well as content and prompt-specific features. Phandi et al. (2015) use correlated Bayesian Linear Ridge Regression focusing on domain-adaptation tasks. All these previous methods are traditional discrete models using handcrafted discrete features.

Recently, Alikaniotis et al. (2016) employ a long short-term memory model to learn features for essay scoring task automatically without any predefined feature templates. It leverages score-specific word embeddings (SSWEs) for word representations, and takes the last hidden states of a two-layer bidirectional LSTM for essay representations. Taghipour and Ng (2016) also adopt a LSTM model for AES, but use ordinary word embedding and take the average pooling value of all the hidden states of LSTM layer as the essay representations. Dong and Zhang (2016) develop a hierarchical CNN model for regression on AES task by processing texts into sentences and using two layers CNN on both sentence-level and text-level to get the final text representation. Our work contributes to the research literature by systematically investigating CNN and LSTM on sentence-level and text-level modeling, and the effectiveness of attention network on automatically selecting more relevant ngrams and sentences for the task.

Our work is also inline with recent work on building hierarchical sentence-document level representations of documents. Li et al. (2015) build a hierarchical LSTM auto-encoder for documents. Yang et al. (2016) build hierarchical LSTM models with attention for document and Tang et al. (2015) use a hierarchical Gated RNN for sentiment classification. Ren and Zhang (2016) use hierarchical CNN-LSTM model for spam detection. We use a hierarchical CNN-LSTM model for essay scoring, which is a regression task.

7 Conclusion

We investigated a recurrent convolutional neural network to learn text representation and grade essays automatically. Our model treated input essays as sentence-document hierarchies, and employed attention pooling to find the pertinent words and sentences. Empirical results on ASAP essay data show that our model outperforms state-of-art neural models for automatic essay scoring task, giving the best performance. Future work explores the advantage of neural models on cross-domain AES task.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. Yue Zhang is the corresponding author.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. *arXiv preprint arXiv:1606.04289* .

Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series* 2004(2):i–21.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *EMNLP*. pages 1741–1752.

Yen-Yu Chen, Chien-Liang Liu, Chia-Hoang Lee, Tao-Hsing Chang, et al. 2010. An unsupervised automated essay scoring system. *IEEE Intelligent systems* 25(5):61–67.

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. Association for Computational Linguistics.

Yann Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems*. pages 1504–1512.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 968974,*. Association for Computational Linguistics, Austin, Texas, pages 1072–1077. https://www.aclweb.org/anthology/D/D16/D16-1115.pdf.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. Automated essay scoring: Applications to educational technology. In *proceedings of EdMedia*. volume 99, pages 40–64.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25(2-3):259–284.

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 90–95.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* .

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. *EACL 2017* page 572.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Ellis B Page. 1968. The use of the computer in analyzing student essays. *International review of education* 14(2):210–225.

Ellis Batten Page. 1994. Computer grading of student prose, using modern concepts and software. *The Journal of experimental education* 62(2):127–142.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression .

Yafeng Ren and Yue Zhang. 2016. Deceptive opinion spam detection using neural network. In *Proceedings of COLING 2016*. Association for Computational Linguistics, Osaka, Japan, pages 140–150. http://www.aclweb.org/anthology/C/C16/C16-1014.pdf.

Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment* 1(2).

Bonggun Shin, Timothy Lee, and Jinho D Choi. 2016. Lexicon integrated cnn models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272* .

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1882–1891. http://aclweb.org/anthology/D/D16/D16-1193.pdf.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*. pages 1422–1432.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*. pages 606–615.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*. pages 1480–1489.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. 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, pages 180–189.