INTEGRATING ATTENTION FEEDBACK INTO THE RECURRENT NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, an improved long short-term memory (LSTM) structure, called hidden attention long short-term memory (HA-LSTM), is proposed to reduce the long-term memory loss when updating the LSTM neural network at each time step. The HA-LSTM structure is different from the standard LSTM structure because a scaled dot-product attention-based sliding controller is introduced to the LSTM structure. The design of the sliding attention controller provides the traditional attention mechanism with a time-varying property, which makes the attention mechanism more suitable for time-series analysis tasks. Traditionally, the inputs to the attention mechanism are the hidden state vectors at all time steps. In contrast, the length of the inputs to the sliding attention controller is range-limited, which means it uses less memory than the traditional attention mechanism. In addition, the HA-LSTM structure integrates the attention mechanism into the standard LSTM structure, which provides the structure with advantages over both the standard LSTM structure and the attention mechanism. Different from most works, which perform unilateral computations on the attention mechanism after collecting the hidden state vectors at all time steps, the information of the gate vectors and the cell state vector of the HA-LSTM structure are based on feedback from the attention mechanism. In other words, the HA-LSTM structure's feedback property helps the cell state vector retain valuable information. To evaluate the performance of the HA-LSTM structure, four text benchmark datasets are used in experiments for text classification tasks. The model presented here is compared with two classic models and with a state-of-the-art model presented in recent years. Most outcomes indicate that the HA-LSTM structure is superior to the other structures.

1 INTRODUCTION

Because the recurrent neural network (RNN) (Rumelhart et al., 1986) features a memory-updated design, it specializes in addressing time-varying issues. Therefore, the RNN is used frequently in natural language processing. For example, phrase representation learning (Cho et al., 2014), text sentiment analysis (Li & Qian, 2016), and text classification (Zhou et al., 2016) have been popular research fields in recent years. However, the RNN also has led to problems, such as the exploding gradients and the vanishing gradients (Hochreiter, 1998; Pascanu et al., 2013). To solve those problems, the proposed LSTM (Hochreiter & Schmidhuber, 1997) structure introduces gate vectors.

Although the LSTM structure has replaced the traditional structure of the RNN in sequence-to-sequence learning (Sutskever et al., 2014), the issue of retaining valuable information from the long-term memory remains. The unidirectional LSTM structure proves that the RNN might forget the initial information if the sequences are long. Therefore, a bidirectional LSTM (BiLSTM) (Schuster & Paliwal, 1997) structure is developed to take advantage of the forward and backward properties of the time series where two reversed sequences are used as the input vectors for the two parallel LSTM units, respectively. Although the bidirectional design can increase the potential for retaining the initial memory, the computation is twice as complex as the computation for a single layer. Also, because the BiLSTM structure makes no changes to the cell-updating rules, the memory loss problem might remain in the unidirectional LSTM structure.

In recent years, the attention mechanism combined RNN models (Bahdanau et al., 2015; Raffel & Ellis, 2015; Yang et al., 2016; Ismail et al., 2019) have performed well on text addressing tasks and image detection tasks. On text addressing tasks, each word vector computed by the recurrent layer multiplies a corresponding ratio calculated by the attention mechanism (Bahdanau et al., 2015; Raffel & Ellis, 2015; Yang et al., 2016) to represent the importance of the word vector in the sentence. On image detection tasks, in contrast, the model (Ismail et al., 2019) applies the attention mechanism to the newly generated inputs before passing them to the recurrent layer. The input at each time step is a concatenation of all input vectors from past to present. Although the model outperforms the traditional models that apply the attention mechanism on the output of the recurrent layer, the computational complexity increases substantially due to the input-cumulative property. Also, if the unprocessed input vectors are independent of each other, the performance is worse than the performance of the related input vectors.

Except for the attention mechanism related RNN model described above, other variations of RNN models have been presented in recent years. For example, the ordered neurons LSTM (ON-LSTM) (Shen et al., 2019) features a tree-based structure that uses the master forget gate and the master input gate to address the long-term memory and short-term memory issues by sequential neurons.

Despite the fact that the works listed above achieved good performance by using the hidden state vectors as the input for the next attention layer, the unilateral computations might limit the performance to two layers. That is, the outputs of the attention layer cannot feed back directly into the recurrent layer. Hence, to achieve reciprocity between the recurrent layer and the attention layer, the HA-LSTM[1] structure is proposed. The HA-LSTM structure uses the range-limited hidden state vectors as inputs to the attention mechanism and passes the corresponding outputs of the attention method back to the previous recurrent layer to update the cell state. Inspired by the scaled dot-product attention mechanism (Vaswani et al., 2017), which has shown good performance on text-addressing tasks, the HA-LSTM structure adopts the self-attention mechanism (Vaswani et al., 2017) as the attention computation.

The primary contributions and features of the proposed HA-LSTM structure are as follows. (1) Different from most works (Bahdanau et al., 2015; Raffel & Ellis, 2015; Yang et al., 2016; Ismail et al., 2019), the HA-LSTM structure integrates the attention mechanism into the standard LSTM structure, which means that the attention mechanism is part of the update mechanism of the LSTM structure at each time step. (2) A sliding attention controller is designed to make the attention mechanism a time-varying updated method, which in turn makes it easier to integrate the traditional attention methods into the cell of the RNN. (3) Because the HA-LSTM structure is a standard LSTM structure-based design, it retains the benefits of the traditional LSTM structure and takes advantage of the attention mechanism. (4) The number of hidden vectors applied to the attention mechanism of the HA-LSTM cell is range-fixed, i.e. the memory cost does not increase as the time step increases. (5) The attention mechanism usually computes the hidden state vectors from the previous recurrent layer, which is a type of unidirectional computation concept. In contrast, the HA-LSTM structure is a bidirectional computation that uses the feedback from the attention mechanism to integrate the past hidden states into the cell computations. (6) The attention mechanism is applied to the hidden state vectors in the HA-LSTM structure. Therefore, the correlation between the input vectors does not have a significant impact on the performance of the HA-LSTM structure.

In this work, the HA-LSTM structure, with three different lengths of sliding attention controller, is compared with the following three structures: LSTM, BiLSTM, and ON-LSTM. Given their robust designs, the LSTM structure and the BiLSTM structure are commonly used on many different types of tasks, which makes it possible to use multiple criteria to make comparisons. The ON-LSTM structure, a state-of-the-art structure proposed in recent years, has been shown to provide extraordinary performance on natural language tasks. Because the design of the experiments in this work is based on text classification, it is appropriate to compare the HA-LSTM structure with the ON-LSTM structure.

[1]The code implement is available at https://github.com/zz2232/ha-lstm.

2 HA-LSTM

Computation of the HA-LSTM structure consists of two parts: (1) updates of the recurrent cell and (2) self-attention computations of the specific range of hidden state vectors. To clarify, the content of the HA-LSTM structure is separated into two subsections: (1) the cell structure and (2) the sliding attention controller.

2.1 CELL STRUCTURE

The HA-LSTM structure for each cell is depicted in Figure 1. Similar to the LSTM cell, the HA-LSTM structure is implemented as:

$$\mathbf{f}_t = \sigma_s(\mathbf{W_f}\mathbf{x}_t + \mathbf{U_f}\mathbf{a}_{t-1} + \mathbf{b_f}) \tag{1}$$

$$\mathbf{i}_t = \sigma_s(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{a}_{t-1} + \mathbf{b_i}) \tag{2}$$

$$\mathbf{o}_t = \sigma_s(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{a}_{t-1} + \mathbf{b_o}) \tag{3}$$

$$\hat{\mathbf{c}}_t = \tanh(\mathbf{W_{\hat{c}}}\mathbf{x}_t + \mathbf{U_{\hat{c}}}\mathbf{a}_{t-1} + \mathbf{b_{\hat{c}}}) \tag{4}$$

where $\mathbf{f}_t \in \mathbb{R}^{H\times 1}$, $\mathbf{i}_t \in \mathbb{R}^{H\times 1}$, $\mathbf{o}_t \in \mathbb{R}^{H\times 1}$, and $\hat{\mathbf{c}}_t \in \mathbb{R}^{H\times 1}$ are the forget gate, the input gate , the output gate, and the cell input activation vector, respectively, and H denotes the number of hidden units and t denotes the t-th time step. The input vector is $\mathbf{x}_t \in \mathbb{R}^{D\times 1}$, where D is the length of the word embedding vector; $\mathbf{a}_t \in \mathbb{R}^{NH\times 1}$ is the hidden attention vector calculated by the dot-product attention (Vaswani et al., 2017)-based sliding attention controller mechanism where parameter N denotes the total number of past time steps applied in the attention mechanism; and $\mathbf{h}_t \in \mathbb{R}^{H\times 1}$ is the hidden state vector. Operators $\circ$ and $\sigma_s()$ denote the element-wise product and the sigmoid function, respectively. The corresponding weight matrices in equations 1-4 are $\mathbf{W_f} \in \mathbb{R}^{H\times D}$, $\mathbf{W_i} \in \mathbb{R}^{H\times D}$, $\mathbf{W_o} \in \mathbb{R}^{H\times D}$, $\mathbf{W_{\hat{c}}} \in \mathbb{R}^{H\times D}$, $\mathbf{U_f} \in \mathbb{R}^{H\times NH}$, $\mathbf{U_i} \in \mathbb{R}^{H\times NH}$, $\mathbf{U_o} \in \mathbb{R}^{H\times NH}$, and $\mathbf{U_{\hat{c}}} \in \mathbb{R}^{H\times NH}$, respectively. The biases in equations 1-4 are $\mathbf{b_f} \in \mathbb{R}^{H\times 1}$, $\mathbf{b_i} \in \mathbb{R}^{H\times 1}$, $\mathbf{b_o} \in \mathbb{R}^{H\times 1}$, and $\mathbf{b_{\hat{c}}} \in \mathbb{R}^{H\times 1}$, respectively.

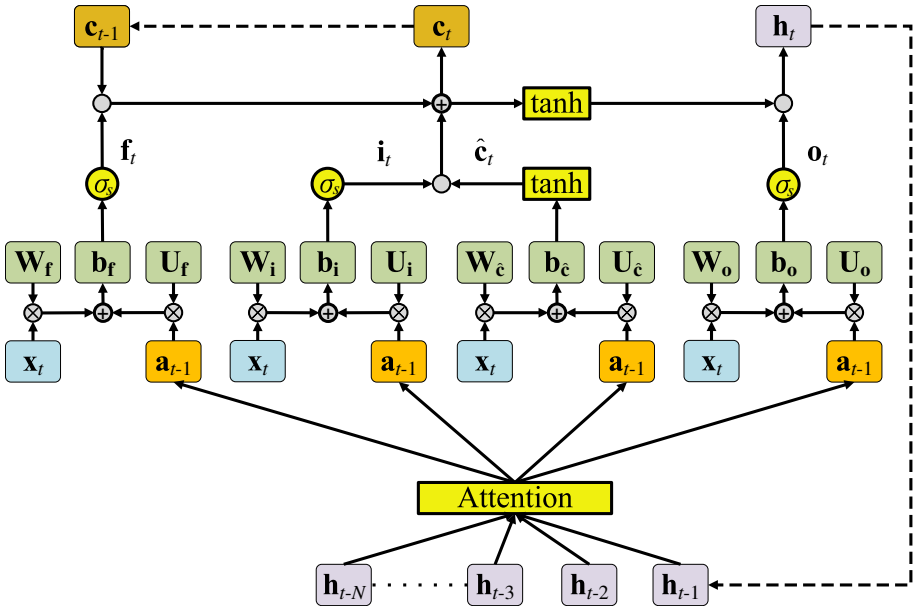


Figure 1: The cell of the HA-LSTM structure; the operator $\otimes$ denotes the matrix multiplication and the operator $\oplus$ denotes the matrix addition.

The forget gate determines the erased ratio of the cell state $\mathbf{c}_{t-1} \in \mathbb{R}^{H\times 1}$. The input gate computes the ratio of the cell input activation vector, which is added to the cell state vector. The process is implemented as:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \hat{\mathbf{c}}_t. \tag{5}$$

The hidden state vector is extracted from the cell state vector by the output gate, which is also the cell output at each time step. Specifically, the hidden state vector can be expressed as:

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t). \tag{6}$$

2.2 SLIDING ATTENTION CONTROLLER

The scaled dot-product attention (Vaswani et al., 2017) is used in the sliding attention controller of the HA-LSTM structure to obtain the hidden attention vector $\mathbf{a}_t$, as shown in Figure 2. The attention mechanism is expressed as $A()$; other procedures are defined as:

$$\mathbf{H}_{t-1} = [\mathbf{h}_{t-1}\ \mathbf{h}_{t-2}\ \mathbf{h}_{t-3} \dots \mathbf{h}_{t-N}]^T \tag{7}$$

$$A(\mathbf{H}_{t-1}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{8}$$

where $\mathbf{H}_{t-1} \in \mathbb{R}^{N \times H}$ is the hidden state matrix at the previous time step, which consists of the past hidden state vectors $\mathbf{h}_{t-w} \in \mathbb{R}^{H \times 1}$ for $w = 1, 2 \dots N$. Matrices $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{N \times d_v}$ in equation 8 denote the query matrix, the key matrix, and the value matrix, respectively, which are generated individually by linearly projecting $\mathbf{H}_{t-1}$ by $\mathbf{W}_Q \in \mathbb{R}^{H \times d_k}$, $\mathbf{W}_K \in \mathbb{R}^{H \times d_k}$, and $\mathbf{W}_V \in \mathbb{R}^{H \times d_v}$, respectively. Parameters $d_k \in \mathbb{R}$ and $d_v \in \mathbb{R}$ are the projected dimensions.

$$\mathbf{H}_{t-1}\mathbf{W}_Q = \mathbf{Q}, \quad \mathbf{H}_{t-1}\mathbf{W}_K = \mathbf{K}, \quad \mathbf{H}_{t-1}\mathbf{W}_V = \mathbf{V}. \tag{9}$$

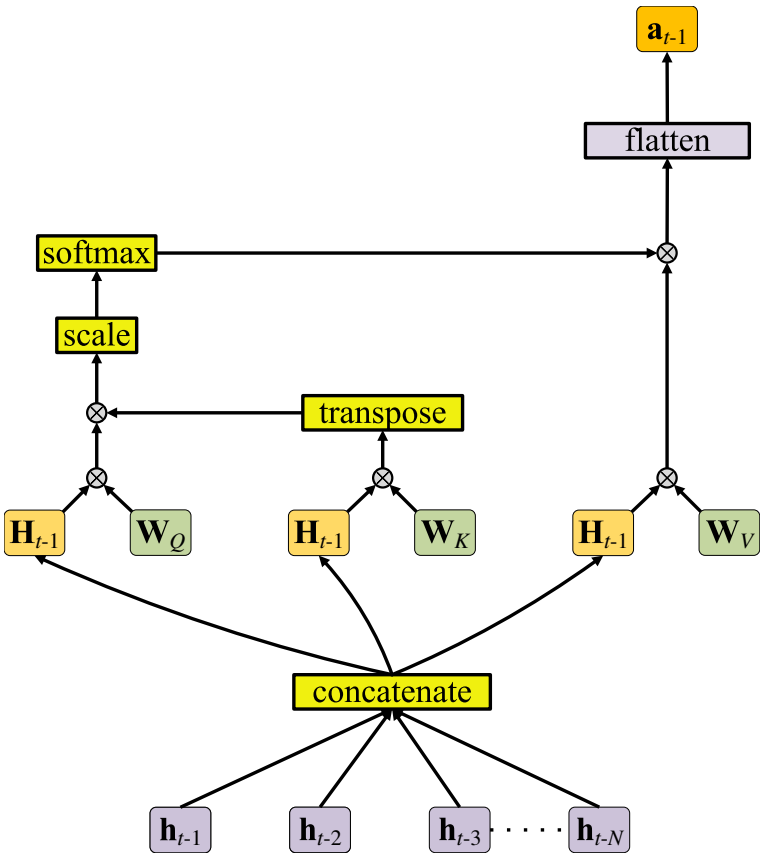


Figure 2: The computations of the sliding attention controller of the HA-LSTM structure.

The function $\mathrm{softmax}()$ in equation 8 is defined as:

$$\mathbf{Z} = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,N} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N,1} & z_{N,2} & \cdots & z_{N,N} \end{bmatrix} \tag{10}$$

$$\mathrm{softmax}(\mathbf{Z}) = \frac{e^{z_{i,j}}}{\sum_{w=1}^{N} e^{z_{i,w}}}, \quad i, j = 1, \ldots, N \tag{11}$$

where $\mathbf{Z} \in \mathbb{R}^{N \times N}$ represents the matrix $\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}$, and $z_{i,j}$ denotes the corresponding element of matrix $\mathbf{Z}$. Considering operating dot product might cause small gradients to approach regions of $\mathrm{softmax}()$. Hence, the value of $\mathbf{Q}\mathbf{K}^T$ is scaled by $\frac{1}{\sqrt{d_k}}$, and the output of $\mathrm{softmax}()$ is multiplied by the value matrix $\mathbf{V}$. Using the attention mechanism, the hidden attention vector $\mathbf{a}_{t-1}$ can be expressed as:

$$\mathbf{a}_{t-1} = \mathrm{flatten}(A(\mathbf{H}_{t-1})) \tag{12}$$

where $\mathrm{flatten}()$ denotes the process of flattening, which reduces the dimension of the matrix.

3 EXPERIMENTS

3.1 DATASETS

Four datasets–AG's news (Zhang et al., 2015), DBPedia (Zhang et al., 2015; Lehmann et al., 2015), text retrieval conference (TREC) (Li & Roth, 2002; Hovy et al., 2001), and 20 Newsgroups (Lang, 1995; Pedregosa et al., 2011)–are used to evaluate performance. Table 1 presents the number of classes, training samples, and test samples of each dataset. The corresponding hyperparameters of the datasets are also included in Table 1.

Table 1: Settings of the datasets in the experiments.

Dataset	Classes	Training Samples	Test Samples	Batch Size	Epochs
AG's News	4	120,000	7,600	500	40
DBPedia	14	560,000	70,000	500	20
TREC	6	5,452	500	120	50
20 Newsgroups	6	11,314	7,532	200	60

AG's News. The dataset of AG's News is directly from the work (Zhang et al., 2015), which involves four topics. Each sample contains the title, the description, and the class fields, but only the class and description fields are considered in the experiments. Also, the dataset is evenly distributed, with 30,000 samples for training and 1,900 samples for testing in each class.

DBPedia. In the experiments, the DBPedia dataset used is from the work (Zhang et al., 2015), which is collected from Wikipedia (Lehmann et al., 2015). Three fields are included in each sample: (1) the corresponding class, (2) the title, and (3) the abstract. In the experiments, only the class and abstract fields are used, and each class contains 40,000 samples for training and 5,000 samples for testing.

TREC[2]. TREC is a text dataset for question classification, which contains six question categories.

20 Newsgroups[3]. The 20 different newsgroups in the 20 Newsgroups dataset are grouped into six topics.

[2]https://cogcomp.seas.upenn.edu/Data/QA/QC/

[3]The dataset is utilized from the website http://people.csail.mit.edu/jrennie/20Newsgroups/ through the scikit-learn Python module.

3.2 NEURAL NETWORK STRUCTURES

The pre-trained 400,000 vocabularies with 100-dimension word vectors generated by the GloVe[4] method (Pennington et al., 2014) are used in the experiments, and the embedding vector length is set to 100. For fairness, three layers are used in each model for all schemes, including LSTM, BiLSTM, ON-LSTM, and HA-LSTM with three different values of N structures, as depicted in Figure 3, where $\mathbf{P}_m \in \mathbb{R}$ for $m = 1, \ldots, S$ denotes the corresponding probability of the category, and $S \in \mathbb{R}$ is the number of classes in the text classification task. Note that the HA-LSTM model with $N = 4$, $N = 8$, and $N = 12$ is denoted as HA-LSTM-4, HA-LSTM-8, and HA-LSTM-12, respectively. The only difference between the models is in the first layer, which is based on the corresponding scheme.

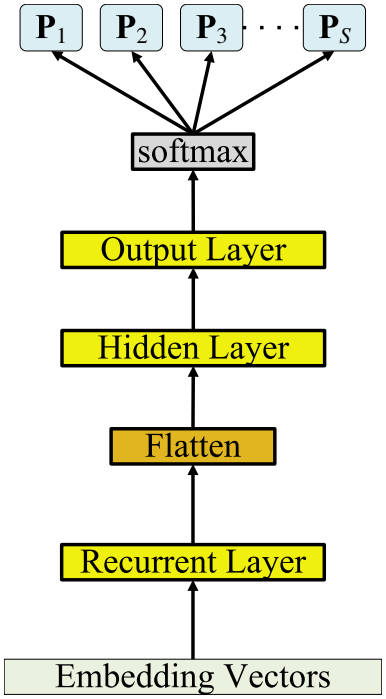


Figure 3: The model structure for testing.

In the recurrent layer, the hidden state vector size is 128, and the output vector size of the recurrent cell at each time step is 128. In the BiLSTM structure, the hidden state vector size in the forward direction and in the backward direction is 64. The final output vector of the recurrent cell is the concatenation of the output vectors computed from the forward direction and the backward direction; hence, the size at each time step becomes 128. In the hidden layer, the output vector size is 32. In the output layer, the output vector size is the number of the classes of each dataset.

The dropout (Srivastava et al., 2014) method, with dropout rate set to 0.1 is used between the hidden layer and the output layer. The Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2015) method is applied as the optimization in the models and the learning rate is set to 0.0006 in all cases. Dimension parameters are set $d_k = d_v = H$ in the HA-LSTM scheme, and the chunk size factor proposed in the ON-LSTM structure is set to 1 in the experiments.

3.3 RESULTS

The test accuracy of the models on four datasets is summarized in Table 2 through Table 5, where the length of the input sequences is limited to six different cases to evaluate the structures for addressing the amount of information, and $q \in \mathbb{R}$ denotes the length of the input sequences. The test accuracy

[4]https://nlp.stanford.edu/projects/glove/

is rounded down to four decimal places and expressed as a percentage. To observe the trend in test accuracy, the numerical results are plotted in Figure 4 through Figure 6. Each figure consists of two parts: (a) graph of the HA-LSTM-4 model and the other three models, and (b) graph of the HA-LSTM model with $N = 4, 8, 12$, respectively.

The experimental results of the AG's News dataset are summarized in Table 2 and plotted in Figure 4(a) and Figure 4(b). The HA-LSTM model for different values of N achieves better results than the results of the other three models in most cases. As the length of input sequences increases, the HA-LSTM model with $N = 4, 8, 12$ still outperforms the other three models. These results indicate that the HA-LSTM structure has the ability to reduce the loss of long-term memory. A comparison of the results of the HA-LSTM model with $N = 4, 8, 12$ shows that there are no obvious performance gaps with different values of N.

Table 2: Test accuracy on AG's News.

AG's News	q					
Models	20	40	60	80	100	120
LSTM (Hochreiter & Schmidhuber, 1997)	87.94	89.64	89.76	88.98	89.46	89.42
BiLSTM (Schuster & Paliwal, 1997)	87.75	88.89	88.90	89.26	89.19	88.80
ON-LSTM (Shen et al., 2019)	88.05	89.64	89.43	89.51	89.56	89.75
HA-LSTM-4	88.57	**90.18**	**90.17**	**90.19**	89.81	90.01
HA-LSTM-8	**88.78**	89.90	89.80	89.92	90.03	**90.09**
HA-LSTM-12	88.22	89.55	90.03	89.77	**90.05**	89.88

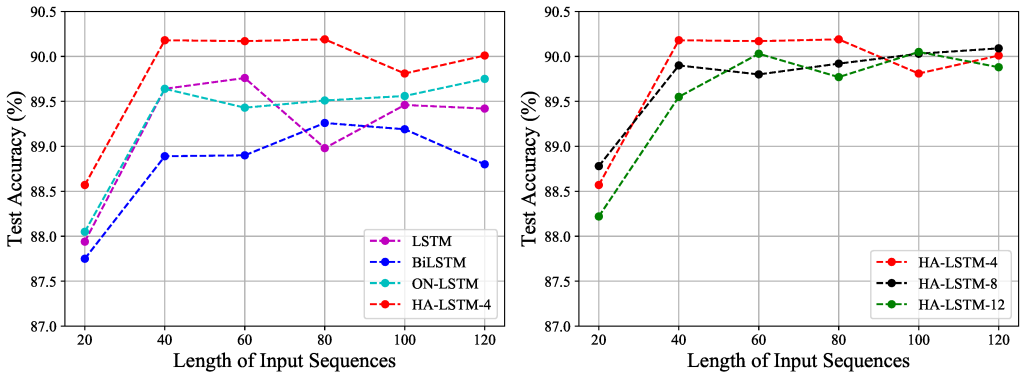


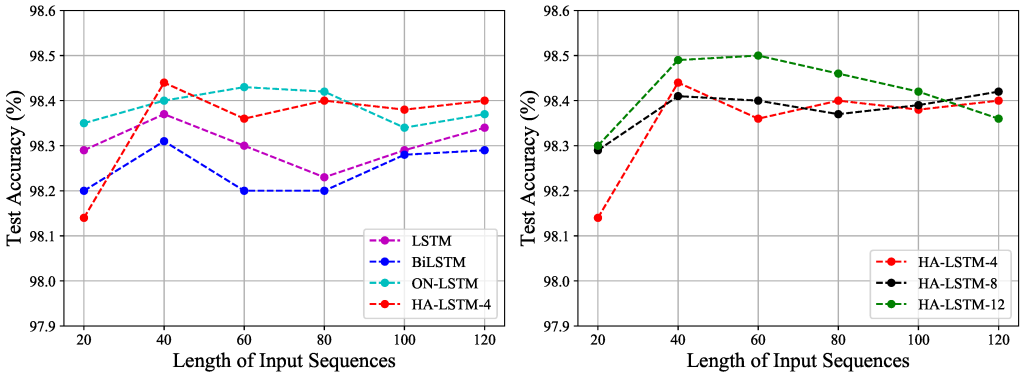
Figure 4: Graphs of test accuracy on AG's News.

For the DBpedia experiments, the test accuracy of each model is summarized in Table 3; the graphs are plotted in Figure 5. Based on the information presented in Table 3 and Figure 5(a), the results show that when the length of the input sequences is 60 and 80, respectively, the performance of the HA-LSTM-4 model is stable, but inferior to the performance of the ON-LSTM model. As the length of input sequences increases to 100 and 120, the test accuracy of the HA-LSTM-4 model is better than the test accuracy of the other models, which indicates that the long-term memory can be maintained by the HA-LSTM structure when the length of the input sequences increases.

Compared with the HA-LSTM model with $N = 4, 8, 12$ in Table 3 and Figure 5(b), the performance of the HA-LSTM-4 model and the performance of the HA-LSTM-8 model is similar. When the length of the input sequences is less than 120, the performance of the HA-LSTM-12 model is better than the performance of the other models. However, the test accuracy decreases gradually as the length of the input sequences increases. As the length of the input sequences is 120, the test accuracy of the HA-LSTM-12 model approaches the test accuracy of the ON-LSTM model. When the length of the input sequences is 20, the test accuracy of the HA-LSTM model with $N = 4, 8, 12$ is low. The poor performance might be a result of the initial value of the trainable parameters. As the length of the input sequences increases, the effect of initialization decreases, as shown in Table 3 and Figure 5.

Table 3: Test accuracy on DBPedia.

DBPedia	q					
Models	20	40	60	80	100	120
LSTM (Hochreiter & Schmidhuber, 1997)	98.29	98.37	98.30	98.23	98.29	98.34
BiLSTM (Schuster & Paliwal, 1997)	98.20	98.31	98.20	98.20	98.28	98.29
ON-LSTM (Shen et al., 2019)	**98.35**	98.40	98.43	98.42	98.34	98.37
HA-LSTM-4	98.14	98.44	98.36	98.40	98.38	98.40
HA-LSTM-8	98.29	98.41	98.40	98.37	98.39	**98.42**
HA-LSTM-12	98.30	**98.49**	**98.50**	**98.46**	**98.42**	98.36



(a) (b)

Figure 5: Graphs of test accuracy on DBPedia.

The test accuracy of the models on TREC is summarized in Table 4. It is obvious that the HA-LSTM model with $N = 4, 8, 12$ outperforms the other three models. The results indicate that the HA-LSTM model achieves the best test accuracy when $N = 12$.

Table 4: Test accuracy on TREC.

Models	TREC
LSTM (Hochreiter & Schmidhuber, 1997)	94.40
BiLSTM (Schuster & Paliwal, 1997)	94.80
ON-LSTM (Shen et al., 2019)	95.20
HA-LSTM-4	95.80
HA-LSTM-8	95.40
HA-LSTM-12	**96.20**

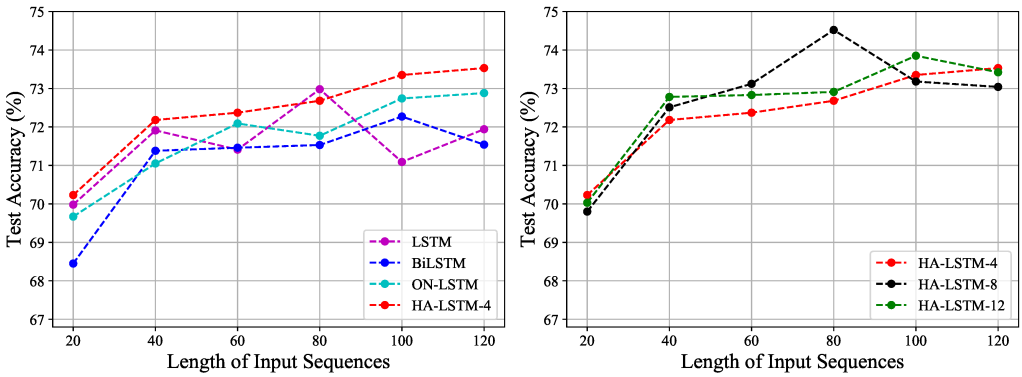
The test accuracy of the HA-LSTM-4 model relative to the accuracy of the other three models on 20Newsgroups is summarized in Table 5 and shown visually in Figure 6(a). The test accuracy of the HA-LSTM-4 model is better than the test accuracy of the other three models, except when the length of the input sequences is 80. As shown in Figure 6(a), the test accuracy of the HA-LSTM-4 increases as the length of the input sequences increases, which indicates that the HA-LSTM-4 model has the ability to retain the valuable part of long-term memory.

A comparison of the results of the HA-LSTM model with $N = 4, 8, 12$, as shown in Table 5 and Figure 6(b), indicates that the test accuracy of the HA-LSTM model with $N = 8, 12$ decreases when the length of the input sequences is 100 and 120, respectively. Despite the decrease in performance as the length of the input sequence increases, the results show that the two models remain competitive.

In summary, as shown in Figure 4, Figure 5, and Figure 6, the increase in test accuracy in the HA-LSTM model with $N = 4$ is the most stable among the three different values of N. Although the

Table 5: Test accuracy on 20Newsgroups.

20Newsgroups	q					
Models	20	40	60	80	100	120
LSTM (Hochreiter & Schmidhuber, 1997)	69.98	71.91	71.41	72.98	71.09	71.94
BiLSTM (Schuster & Paliwal, 1997)	68.45	71.38	71.46	71.53	72.27	71.54
ON-LSTM (Shen et al., 2019)	69.67	71.05	72.09	71.77	72.74	72.88
HA-LSTM-4	**70.23**	72.18	72.37	72.68	73.35	**73.53**
HA-LSTM-8	69.80	72.51	**73.12**	**74.52**	73.18	73.04
HA-LSTM-12	70.03	**72.78**	72.83	72.91	**73.85**	73.42



(a) (b)

Figure 6: Graphs of test accuracy on 20Newsgroups.

HA-LSTM model with $N = 8, 12$ does outperform the HA-LSTM model with $N = 4$ in many cases, the drop speed of test accuracy is fast. For example, the test accuracy of the HA-LSTM-12 model on DBPedia begins to decrease when the length of the input sequences is greater than 60, and there is a peak in the test accuracy of the HA-LSTM-8 model in 20Newsgroups when the length of the input sequences is 80.

4 CONCLUSION

According to experimental results using the different datasets, the HA-LSTM structure, in most cases, is superior to the other methods. The experimental results also indicate that the HA-LSTM structure is capable of retaining the valuable part of long-term memory as the length of the input sequences increases. In terms of memory cost of the HA-LSTM unit, it is shown that the performance of the HA-LSTM model with $N = 4$ is competitive with the performance of other models in many cases. That is, the low cost of additional memory indicates that the HA-LSTM structure has enormous potential to be combined into the deep neural network models.

REFERENCES

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998. ISSN 0218-4885.

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

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001.

Aya Abdelsalam Ismail, Mohamed Gunady, Luiz Pessoa, Héctor Corrada Bravo, and Soheil Feizi. *Input-Cell Attention Reduces Vanishing Saliency of Recurrent Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2): 167–195, 2015.

Dan Li and Jiang Qian. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pp. 471–475, 2016.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Colin Raffel and Daniel P. W. Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv:1512.08756*, 2015.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, pp. 318–362. MIT Press, Cambridge, MA, USA, 1986.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11): 2673–2681, November 1997.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 649–657, Cambridge, MA, USA, 2015. MIT Press.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv:1611.06639*, 2016.