An Integrated Neural Model for Sentence Classification

Yanbu Guo, Weihua Li*, Chen Jin, Yunhao Duan, Shuang Wu

School of Information Science and Engineering, Yunnan University, Kunming 650091
*E-mail: lywey@163.com

Abstract: Text classification is one of the fundamental problems in natural language processing. The difficulty of text classification still lies in the ambiguity and rhetorical of natural language. In order to classify sentences more effectively, we propose a hybrid L-MCNN model to represent the semantics of sentences, and this model consists of two parts, bidirectional long short-term memory (LSTM) and multichannel convolutional neural networks (CNNs). Different from the existing methods, our model utilizes the bidirectional LSTM to acquire the max contextual information of every position in sentences, and then the abstract semantics would be fed into the multichannel convolution layer which uses several filters to extract active local n-gram features. By combining the advantages of both architectures, the L-MCNN model can capture both long-distance dependencies and local information within sentences to improve the classification performance. Several experiments are conducted, and the results indicate that our system outperforms the baseline algorithms on sentiment analysis and question classification tasks.

Key Words: Convolutional Neural Networks, Long Short-Term Memory, Sentence Classification, Text Representation

1. INTRODUCTION

Text classification is one of the fundamental problems in natural language processing. Sentence classification is widely used in E-commerce websites, online social networks and political orientation analyses [1]. Improving the classification performance of sentence could improve the performance of information retrieval system and machine translation system. Hence, sentence classification has witnessed a great deal of attention over ten years [2]. As the key step of sentence classification, sentence modeling aims at representing sentences as meaningful features for text classification and sentiment classification. Most of traditional sentence modeling methods are based on the bag-of-words model which often suffers from the curse of dimensionality, and others use composition methods instead, an algebraic operation over semantic word vectors to produce the semantic sentence vector. However, such methods may perform poorly because of the loss of word order information [3]. Recently, deep learning approaches have gained amazing results on sentence classification tasks due to the ability of the automatic learning features, and what's more, those methods can alleviate the handcrafted feature engineering [4-6]. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have emerged as two mainstream architectures and are often combined with sequence-based or tree-structured models [7-10].

Owing to outstanding capability of extracting local spatial or temporal correlations, CNNs have achieved competitive performance in computer vision [7] and speech recognition [8]. For sentence modeling, CNNs performs excellently in capturing n-gram features at different positions of sentences through convolutional filters, and can extract main relations through pooling operations. CNNs have been successfully combined with both sequence-based model and tree-structured model [9-11] for sentence classification.

Another popular neural models, RNNs, can handle sequences of arbitrary length and extract long-term dependency information. To avoid the problem of gradient exploding or vanishing in RNNs, Long Short-term Memory networks (LSTMs) [12] were designed to better remember and access memories. Moreover, incorporating information on both directions of every position in sequences, BLSTMs are introduced and have been reported to achieve preferable performance in machine translation tasks [13].

In short, CNNs can learn local information from texts but lack the ability of learning sequential correlations, especially long-term dependencies. On the other hand, RNNs is specifically designed to capture long term dependencies, but could not extract n-gram features in the parallel way. However, one of the remaining challenges is to model texts for extracting semantic information when considering the variability of the natural language. Thus, in this work, the LSTMs are used directly to learn long-term dependencies from sequences, and then the CNNs model is constructed on top of the LSTMs to learn the higher-level representation of n-grams for representing texts.

In this paper, we propose a hybrid model by combining long short-term memory networks with convolutional neural networks to model sentences. To take advantage of both CNNs and LSTMs, we design a simple end-to-end architecture by feeding the output of the BLSTMs into the multichannel CNNs framework. The BLSTMs is constructed on the top of pre-trained word vectors from massive unlabeled texts to capture long-term dependencies. Then to learn higher-level representations of texts, the

This work is supported by Integration of Cloud Computing and Big Data, Innovation of Science and Education (2017B00016), Program for Excellent Young Talents, Yunnan University (WX173602), National Science Foundation of China (Grant NO.11661081), the Project of Innovative Research Team of Yunnan Province.

feature of the BLSTMs are organized as window features to serve as the input of multichannel CNNs.

Our key contributions in this paper are as follows:

(1) We propose the hybrid L-MCNN architecture, which utilizes the BLSTMs to capture long-term dependency information from the word embeddings of sentences, and utilizes multichannel convolutions to extract n-gram features for sentence classification.

(2) The hybrid model's performance is comparable to the state-of-the-art methods on two benchmark datasets without any handcrafted features.

(3) The hybrid model firstly captures short and long-term dependencies by concatenating the maximum value of between each word embedding and the contextual information of its two directions, which extract the abstract representation of sentences.

2. Related Work

Deep learning based on neural models has achieved great success in word embeddings [14], sentiment classification and machine translation [8]. Within natural language processing, much work has been involved in learning word representation through neural models [15]. In these methods [14-15], instead of using one-hot vectors by indexing words into a vocabulary, each word is modeled as a low dimensional and dense vector which encodes both semantic and syntactic information of words.

In text representation, neural models are constructed upon either word sequences or the transformed syntactic parse tree. In these models, CNNs and RNNs are two types of popular frameworks. CNNs can capture local correlations well. To extract global features by max-pooling, Collobert [16] applied convolutional filters to successive windows for a given sequence. As a slight variant, Kim [8] proposed a CNNs architecture with multiple filters (with a various window size) and two channels of word vectors. To capture word relations of varying sizes, Kalchbrenner [9] proposed a dynamic k-max pooling mechanism. Recently, Tao [6] applied tensor based operations between words to replace linear operations on concatenated word vectors in the standard convolutional layer and explore the non-linear interactions between n-grams. And Mou [10] also explored convolutional models on tree-structured sentences.

As a type of sequence model, RNNs are able to deal with the variable-length input sequences and discover long-term dependencies. Various variants of RNNs have been proposed to better store and access memories by Hochreiter [12] and Cho [13]. With the ability of explicitly modeling time-series data, RNNs are being increasingly applied to sentence modeling. For example, Tai [5] adjusted the standard LSTMs to tree-structured topologies and obtained superior results over a sequential LSTMs on related tasks.

In this paper, we stack bidirectional LSTMs and CNNs in a unified architecture for semantic sentence modeling. The combination of CNNs and LSTMs can be seen in some computer vision tasks, such as image caption and speech recognition [17]. Most of these models use multilayer CNNs and train CNNs and RNNs separately or throw the output of a fully connected layer of the CNNs into the RNNs as inputs. Different from the previous methods, we apply bidirectional LSTMs to texts and then feed long short-term dependencies features directly to multiple channel CNNs, so our architecture enables the LSTMs to learn long-range dependency contextual information and different regional feature from texts. Experiments on tasks clearly demonstrate the superiority of our model over single CNNs or LSTMs model and other related sequence-based models.

3. L-MCNN Model

As shown in Figure 1, the L-MCNN neural model consists of three parts: BLSTMs Layer, Multichannel Convolution Layer, Flatten layer and Output Layer. The details are described in the following.

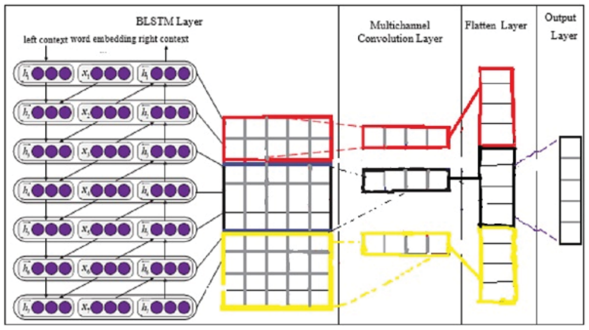
BLSTM Layer
left context word embedding right context
Multichannel Convolution Layer
Flatten Layer
Output Layer

Fig 1. The structure of the L-MCNN model

3.1 Long Short-Term Memory Networks

Given a sequence $s=[x_1,x_2,x_3,\cdots,x_{l-1},x_l]$, where l is the length of texts. x_i is the k-dimensional word vector corresponding to the i_{th} word, and the sequence s are represented by the matrix, here $s\in R^{l\times d}$; In Figure 1, the BLSTMs layer first transform the sentence into a matrix of the word representation, and then takes the word vector matrix as the input in order to produce higher-level presentation of word features. LSTMs was first introduced by Hochreiter [12] and obtained remarkable performance in statistical machine translation [18]. LSTMs is a RNNs architecture specifically designed to bridge long-time delays between relevant input and target events, and this make it suitable for problems where long range context is required.

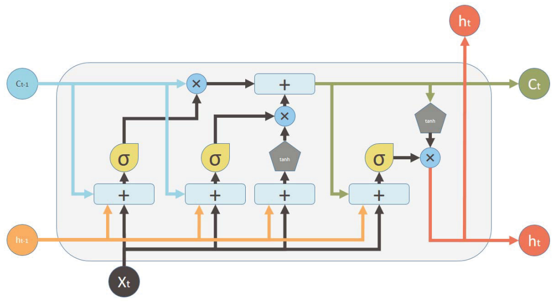

Fig 2. Schematic of the LSTMs unit

LSTMs is explicitly designed to cope with these gradients vanishing problems for time-series data, and therefore we choose the LSTMs upon word embeddings to learn long-term dependencies, which generates middle-level features. Basically, a LSTMs unit is composed of three gates which control the proportions of information to forget and to pass on to the next time step. Figure 2 shows the basic structure of a LSTMs unit. At each time step, the output of the model is controlled by a set of gates; the old hidden state h_{t-1}, the input at the current time step x_t, the forget gate f_t, the input gate i_t and the output gate o_t. These gates collectively decide how to update the current memory cell c_t and the current hidden state h_t.

Formally, the LSTMs transition functions are defined as follows.

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$q_t = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes q_t \quad (4)$$

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (6)$$

Here, σ is the logistic sigmoid function that has an output in [0, 1], tanh denotes the hyperbolic tangent function that has an output in [−1, 1], and ⊗ denotes the element-wise multiplication. Moreover, we can view f_t as the function to control what information from the old memory cell is going to be thrown away, i_t to control how much new information is going to be stored in the current memory cell, and o_t to control what to output based on the memory cell c_t.

For the sequence modeling, it is beneficial to have access to the forward context as well as backward context. Schuster [13] proposed BLSTMs to extend the unidirectional LSTMs by adding a LSTMs layer, where the connections flow in opposite temporal order. On recent years, BLSTMs has been shown as a successful model [23]. Therefore, in this paper, BLSTMs is utilized to capture the forward and backward context. As shown in Figure 1, the BLSTMs layer contains two LSTMs networks for the forward and backward sequence context respectively. Then the two hidden states are concatenated to form the final output. Formally, the concatenation of the i_{th} word is updated in the following equation:

$$h_i = [(\max\{\overrightarrow{h_i}, x_i\}) \oplus (\max\{\overleftarrow{h_i}, x_i\})] \quad (7)$$

Here, element-wise sum ⊕ is used to combine the max-value of between the forward/backward context and the word embedding.

3.2 Multichannel Convolution Operation

The higher-layer semantic information is fed into the multichannel convolution layer from the BLSTMs layer. Then we introduce the multichannel convolutional operations, which is the parallel convolutional operation having three types of filter lengths.

The one-dimensional convolution involves a filter sliding over the matrix vector of each sentence by the BLSTMs layer and detecting the feature at different positions. Let $h_i \in R^d$ be the d-dimensional word vectors for the i_{th} word from the higher semantic of the BLSTM layer. Let $H \in R^{l \times d}$ denote the input sentence, $H = [\mathrm{h}_1, \mathrm{h}_2, \mathrm{h}_3, \cdots, \mathrm{h}_l]$, where l is the length of the sentence. Let L be the length of the filter, $L \in \{2,3,4\}$. The vector $m \in R^{k \times d}$ is a filter for the convolution operation. For each position j in the sentence, we have a window w_j（$\mathrm{j} \in [1, 3]$）with k consecutive word vectors, denoted as:

$$w_j = [h_j; h_{j+1}; h_{j+2}; \cdots; h_{j+k-2}; h_{j+k-1}] \quad (8)$$

Here, the semicolon represent row vector concatenation operator. The filter m convolves with the window vectors (k-grams) at each position to generate the feature map $c \in R^{L-k+1}$; each c_i of the feature map for window vector w_j is produced as follows:

$$c_i = f(w_i * m + b_i) \quad (9)$$

Here, $*$ is element-wise multiplication, $b_i \in R$ is a bias term and f is a nonlinear transformation function, such as sigmoid, hyperbolic tangent. We choose ReLU [19] as the nonlinear function. The L-MCNN model uses multiple filters to generate multiple feature maps. For n filters with the same length, the generated n feature maps can be rearranged as feature representations W for each window w_j:

$$W = [c_1; c_2; c_3; \cdots; c_{n-1}; c_n] \quad (10)$$

Here, semicolons represent column vector concatenation and c_i is the feature map generated with the i_{th} filter. W is the new feature representation generated from n filters for the window w_j.

In brief, this work only utilizes the multichannel convolutional operation and the lengths of filters are 2, 3 and 4 respectively.

3.3 Flatten Layer

Through multichannel convolutional operations on the sentence representation of the BLSTM layer, the hybrid model could capture n-gram feature; and then the new higher features are fed into the flatten layer, which results in capturing both long-term dependencies and local correlations of spatial or temporal information in sentences. Finally the feature will be passed into the output layer, which consists of two layers, dense layer and softmax layer.

3.4 Output layer

The more rich semantic information is fed into the dense layer, which accepts the vector h^* of appropriate dimension about sequences. The dense layer multiplies results from the previous layer with a weight matrix and adds a bias vector, and the ReLU activation function is also applied. The result vectors are finally inputted to the sotfmax layer that outputs the final classification result.

The output h^* of the dense layer is the whole representation of the input s. And then it is passed to a classifier to predict the label y^* which is a discrete set of classes Y. The classifier takes the h^* as the input:

$$p^*(y \mid s) = soft\max(w^s h^* + b^s) \qquad (11)$$

$$y^* = \arg\max_{y^* \in Y}(p^*(y \mid s)) \qquad (12)$$

The training objective is to minimize the categorical or binary cross-entropy loss. The loss is calculated as a regularized sum:

$$L(y_j^*, y_j, \theta) = \sum_{j=1}^{n} y_j(\log(y_j^*)) + \lambda/2 \|\theta\|_2^2 \qquad (13)$$

Where $y_j \in R^m$ is the one-hot represented from the ground truth, $y_j^* \in R^m$ is the estimated probability for each class, m is the size of target classes, and λ is an L2 regularization hyper-parameter.

4. Network Training and Implementation Details

In this section, we provide details about training the neural network. We use Keras and tensorflow to build the neural network for training and testing the L-MCNN model, and the training procedure is speeded up by GPU (Nvidia GTX 1060 6G) on Ubuntu 16.04 LTS.

4.1 Parameter Initialization

Word embeddings: Based on continuous bag-of-words architecture, the word2vec vectors is trained on 100 billion words from Google News[1] [13]. And, the dimension of the vectors is 300. Words not in the set of pre-trained words are initialized by the uniform distribution [-0.25, 0.25].

Weight Matrices and Bias Vectors: Matrix parameters are randomly initialized with the Truncated Normal method, which is similar to Random Normal distribution, but Truncated Normal method would discard the data located outside the mean of the two standard deviations and regenerate the data to form a truncated distribution. Bias vectors are initialized to zero.

4.2 Optimization Algorithm

Parameter optimization is performed with mini-batch stochastic gradient descent. We employ stochastic gradient descent over shuffled batch size 20 to train the model and update parameters with the optimizer AdaDelta [20].

Moreover, we also explored other more sophisticated optimization algorithms such as Adam or RMSProp. But only AdaDelta obtains better performance in experiments.

Fine Tuning: During gradient updates of the neural model, we fine-tune each of initial embeddings, and modify them by back-propagating gradients.

Dropout Training: To avoid over-fitting, we employ dropout [21] and L2 weight regularization to regularize our model. In our model, we either apply dropout to word vectors before feeding the sequence of words into the LSTM layer or to the output of the MCNN layer before the softmax layer. The L2 regularization is applied to the weight of the softmax layer.

Table1 describes the hyper-parameters for all our experiments.

Table1. Hyper-parameters for all experiments

Layer	Hyper-parameter	MR	TREC
CNN	window size	2,3,4	2,3,4
	number of filters	100	100
	dropout	0.5	0.5
BLSTM	initial size	300	300
	dropout	0.2	0.2
other	initial learning rate	0.1	0.1
	batch size	20	32

5. Datasets and Baselines

In this section, we describe the datasets and baselines which are used to evaluate our proposed approach.

5.1 Datasets

In order to illustrate the validity and generalization of the L-MCNN model in this paper, we evaluate the model on two datasets, MR and TREC.

(1) MR[2]: This data set involves movie reviews, and each review is described as one sentence. There are 5,331 positive and 5,331 negative reviews in this data set.

(2) TREC[3]: TREC question dataset consists of 6 different question types: abbreviation, human, entity, description, location, and numeric information. The entire dataset contains 5452 training examples and 500 testing examples.

All experiments were based on the same pre-training 300-dimensional word vector and the same parameters. For MR data set, we use 10-fold cross validation to report the result. For TREC dataset, without a standard development set, we randomly select 10% of the training data as the development set.

Table2 shows the statistics for the datasets，and C，m，L，N，V and Vpre represents number of the target classes, maximum sentence length，average sentence length, dataset size, vocabulary size, number of words in the set of pre-trained word vectors respectively. TestN is the size of test set, where CV means that 10-fold cross validation is used in the data set.

Table2.The statistics for the datasets

Model	C	m	L	N	V	Vpre	TestN
MR	2	37	20	10662	18765	16448	CV
TREC	6	56	10	5992	9592	9125	500

1. https://code.google.com/p/word2vec/
2. https://www.cs.cornell.edu/people/pabo/movie-review-data/
3 http://cogcomp.org/Data/QA/QC/

5.2 Baselines

We compare our model with the state-of-the-art approaches on the datasets, MR and TREC. These methods include NBSVM, CNN-non-static, CNN-multichannel, SVM, DCNN, BiRNN, Bi-LSTM, and ConvBiLSTM.

(1) NBSVM [22] utilizes bigram as the feature of Polynomial Bayesian Model.

(2) SVM [25] uses unigrams, bigrams, POS tags, parser and WordNet as engineered features.

(3) CNN-non-static [8] uses pre-trained vectors from word2vec and word embeddings are fine-tuned during training the model.

(4) CNN-multichannel [8] includes two sets of word vectors initialized with word2vec, and each set of vectors is treated as a 'channel'. A set of vectors are fine-tuned during training the model.

(5) DCNN [9] is dynamic convolutional neural network with k-max pooling.

(6) BiRNN [23] uses traditionally bidirectional recurrent neural network to sentence classification.

(7) BiLSTM [23] uses bidirectional long-term memory network to sentence classification.

(8) ConvBiLSTM [23] uses the convolution operation to merge the local expression of LSTMs into a new expression at each position.

6. Results and Model Analysis

6.1 Experiment Results

In this section, Table3 and Table4 shows the evaluation results on MR sentiment classification and TREC question type classification tasks.

Table3. Results of our hybrid model against other methods on MR dataset

Model	Acc/%	Reported in
NBSVM	79.4	Wang[22]
CNN-non-static	81.5	Kim[8]
CNN-multichannel	81.1	Kim[8]
BiRNN	79.9	Wan[23]
BiLSTM	81.5	Wan[23]
ConvBiLSTM	82.3	Wan[23]
L-MCNN	**82.4**	**Our model**

Table4. Results of our hybrid model against other methods on TREC dataset

Model	Acc/%	Reported in
SVM	95.0	Silva[24]
CNN-non-static	93.6	Kim[8]
CNN-multichannel	92.2	Kim[8]
DCNN	93.0	Kalchbrenner[9]
Bi-LSTM	93.4	Wan[23]
ConvBiLSTM	95.2	Wan[23]
L-MCNN	**97.1**	**Our model**

6.2 Comparison with Previous Work

Sentiment Classification: Table 3 and Figure 3 illustrates the performance of our method and other state-of-art models on MR sentiment dataset. It is obvious that the L-MCNN model achieves preferable performance on sentence polarity classification than others. Compared with the SVM、CNN、BiLSTM and BiRNN model, the accuracy of our hybrid model is improved greatly, this suggests our multi-channel operations can extract more n-grams feature and the LSTM framework is good at deal with time-serious data. Compared to the ConvBiLSTM model, the performance is increased slightly, but the L-MCNN model can extract more local features.

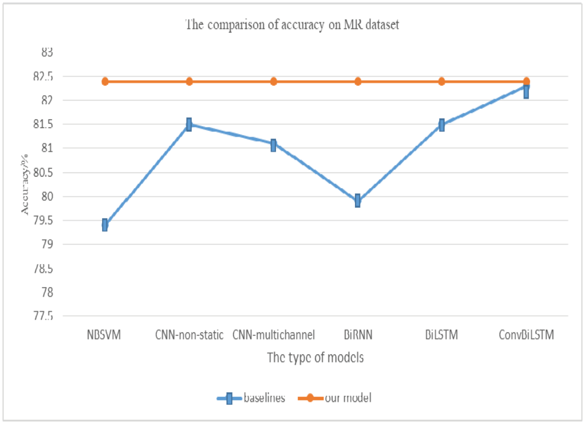

Fig 3. The accuracy on MR dataset

Question Type Classification: The prediction accuracy on TREC question classification is reported in Table 4. From Table4 and Figure 4, the following conclusions are obvious. Our model consistently outperforms all baseline models above mentioned, which means that the L-MCNN model can capture more semantic information of TREC questions efficiently.

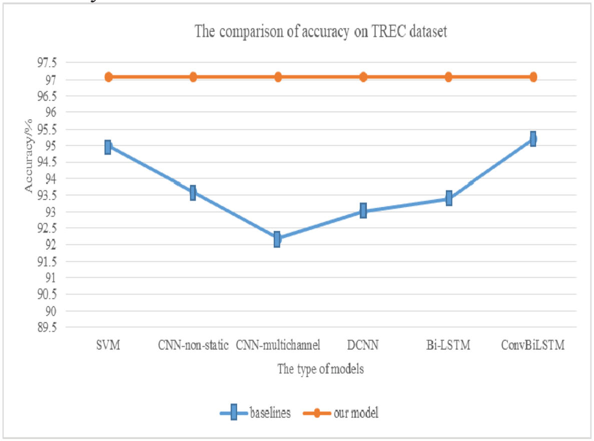

Fig 4. The accuracy on TREC dataset

7. Conclusion and Future Work

In this paper we proposed the L-MCNN model, a novel framework that combines bidirectional long short-term memory networks with convolutional neural networks. The feature extraction method improved by the BLSTM layer can capture more long-term dependencies. Then output sequences of such higher level representations are fed into the multichannel convolutional neural network to learn n-gram feature. The experimental results show the L-MCNN model achieves superior performance on standard sentiment classification and question classification tasks. In the future, we would explore the framework added by attention mechanisms in text classification.

REFERENCES

[1] W. Theresa., W. Janyce, H. Paul, Recognizing contextual polarity: an exploration of features for phrase-level sentiment analysis, Computational Linguistics, Vol.35, No.3, 399-433, 2009.

[2] S. Mihai, T. Julie., N. Ramesh, C. D. Manning, Multi-instance multi-label learning for relation extraction, Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 455-465, 2012.

[3] J. Rie, T. Zhang, Effective use of word order for text categorization with convolutional neural networks, Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, 103–112, 2015.

[4] R. Socher, A. Perelygin, J. Y.Wu, J. Chuang, C. D. Manning, A. Y Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment Treebank, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 1631-1642, 2013.

[5] K. S. Tai, R. Socher, C. D. Manning, Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, 1556-1566, 2015.

[6] L. Tao, B. Regina, J. Tommi, Molding CNNs for text: non-linear, non-consecutive convolutions, Indiana University Mathematics Journal, Vol.58, No.3, 1151-1186, 2015.

[7] D.Y. Tang, B. Qin, T. Liu, Document Modeling with Gated Recurrent Neural Network for Sentiment Classification, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 1422-1432, 2015.

[8] Y. Kim, Convolutional neural networks for sentence classification, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1746-1751, 2014.

[9] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 655-665, 2014.

[10] L.L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, Z. Jin, Discriminative neural sentence modeling by tree-based convolution, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2315–2325, 2015.

[11] D. Misha, D. Alban, K. Nal, B. Phil, F.N. D, Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network,. Computer Science, 2014.

[12] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation, Vol.9, No.8, 1735-1780, 1997.

[13] S. Mike, P. Kuldip k, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing, Vol.45, No.11, 2673-2681, 1997.

[14] M. Tomas, S. Ilya, C. Kai, C. Greg, D. Jeffery, Distributed Representations of Words and Phrases and their Compositionality, Advances in Neural Information Processing Systems, Vol.26, 3111-3119, 2013.

[15] Q. Le, T. Mikolov, Distributed Representations of Sentences and Documents, Proceedings of the 31st International Conference on Machine Learning, 2014

[16] C. Ronan, W. Jason, K. Michael, K. Koray, K. Pavel, Natural Language Processing (Almost) from Scratch, Journal of Machine Learning Research, Vol.12, No.1, 2493-2537, 2011.

[17] T.N Sainath, O. Vinyals, A. Senior, H. Aak, Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks, IEEE International Conference on Acoustics, Speech and Signal Processing, 4580-4584, 2015.

[18] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, In Advances in neural information processing systems, 3104–3112, 2014.

[19] N. Vinod, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, Proceedings of 27th International Conference on International Conference on Machine Learning, 807-814, 2010.

[20] Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. arXiv preprint, arXiv:1212.5701, 2012.

[21] H. Geoffrey E, S. Nitish, K. Alex, S. llya, S. Ruslan R, Improving neural networks by preventing co-adaptation of feature detectors, Computer Science, Vol.3, No.4, 212-223, 2012.

[22] S.D. Wang, C. D. Christopher, Baselines and bigrams: Simple, good sentiment and topic classification, Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 90-94, 2012.

[23] S.X. Wan, Y.Y. Lan, J.F. Guo, J. Xu, L. Pang, X.Q. Cheng, Local Bidirectional Long Short Term Memory for Text Classification, Journal of Chinese Information Processing,Vol.31, No.3, 62-68, 2017.

[24] J. Silva, L. Coheyr, A. C. Mendes, A. Wichert, From symbolic to sub-symbolic information in question classification, Artificial Intelligence Review, Vol.35, No.2, 137-154, 2011.