Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 16(2): 106-113, Jun. 2018 Regular paper

Text Categorization with Improved Deep Learning Methods

Xingfeng Wang[1,2] and Hee-Cheol Kim[3*], ***Member*, *KIICE***

[1]Information Engineering College, Eastern Liaoning University, Dandong 118000, China
[2]Department of Computer Engineering, Inje University, Gimhae 50834, Korea
[3]Department of Computer Engineering & Institute of Digital Anti-Aging Healthcare (IDA), Inje University, Gimhae 50834, Korea

Abstract
Although deep learning methods of convolutional neural networks (CNNs) and long-/short-term memory (LSTM) are widely used for text categorization, they still have certain shortcomings. CNNs require that the text retain some order, that the pooling lengths be identical, and that collateral analysis is impossible; In case of LSTM, it requires the unidirectional operation and the inputs/outputs are very complex. Against these problems, we thus improved these traditional deep learning methods in the following ways: We created collateral CNNs accepting disorder and variable-length pooling, and we removed the input/output gates when creating bidirectional LSTMs. We have used four benchmark datasets for topic and sentiment classification using the new methods that we propose. The best results were obtained by combining LTSM regional embeddings with data convolution. Our method is better than all previous methods (including deep learning methods) in terms of topic and sentiment classification.

Index Terms: CNN, Disorder, LSTM, Text categorization

I. INTRODUCTION

Text categorization assigns predefined classifications to documents written in natural languages; several forms of categorization can handle a variety of documents detecting topics and spam [1], the sentiments [2, 3] of product/movie reviews, and prediction of trends [4]. Most use traditional methods for text categorization; the models include naïve Bayes, support vector machines (SVM), k-nearest network (KNN), and newer methods [5].

In 1998, McCallum and Nigam [6] compared two naïve Bayes text classifications. The multinomial model was almost always better than the multi-variate Bernoulli model, reducing average error by 27%. In 2001, Soucy and Mineau [7] showed that a simple KNN algorithm was 5% points better than a naïve Bayes model. The SVM, the traditional method, has attracted much attention. In 1999, Joachims [8] introduced a transductive SVM method, which averaged 10 percentage points higher than a KNN method.

The traditional methods are seldom >80% accurate. Today, deep learning—via convolutional neural networks (CNNs) and long-/short-term memory (LSTM)—is often used for text categorization.

CNN usually deals with two-dimensional (2D) structures, such as images, computing a unit for each region. However, text categorization is a 1D problem; it is necessary to solve the word sequence and use the convolution layer to deal with small text regions. Text categorization using CNNs is popular [9]; the first layer searches a table and converts sentences to word vectors. We previously used a CNN to train word vectors and used other methods to learn from additional large corpora [10]. In other words, CNNs have many limitations. Therefore, we prefer other methods. Here, we apply a CNN directly to high-dimensional vectors; we do not employ

Received 14 March 2018, **Revised** 12 June 2018, **Accepted** 14 June 2018
***Corresponding Author** Hee-Cheol Kim (heeki@inje.ac.kr, Tel: +82-55-320-3720)
Department of Computer Engineering & Institute of Digital Anti-Aging Healthcare (IDA), Inje University, 197, Inje-ro, Gimhae 50834, Korea.

Open Access https://doi.org/10.6109/jicce.2018.16.2.106 print ISSN: 2234-8255 online ISSN: 2234-8883

word embedding. We use a graphics processing unit (GPU) to perform computations and handle sparse high-dimensional data, improving accuracy, accelerating training and prediction, and simplifying the system. Although a CNN could theoretically deal with natural language (text), CNNs are not designed for such inputs. CNNs usually accept dense inputs; each value is important, and most inputs are not zero. It is challenging to work with text data; we address this with the aid of LSTM, which is a recurrent neural network (RNN) [9]. LSTM can deal with both images and text (word sequences), recognizing embedded sequences. Compared to traditional recurrent networks, LSTM affords two advantages: a shorter learning time and better feasibility. LSTM can handle text of variable size.

Text categorization using LSTM [11] is complex, reducing training efficiency; we thus simplify the model. Our new method affords higher accuracy and more rapid training. Both our LSTM and CNN models are better than traditional methods. Our results optimize text categorization; we use text regions to establish higher standards than would be possible using isolated words. Thus, as revealed by tests using four benchmark datasets, our regional embeddings were better than the traditional embeddings.

The remainder of this paper is structured as follows. Section II describes the models and methods in preliminary terms (subsection A); subsections B and C describe text categorization using a CNN and LSTM, respectively. Section III contains the results; subsection A describes the experimental environment and processes; subsection B describes the use of a CNN for topic and sentiment classification and subsection C contains the LSTM data; we used four datasets to facilitate direct comparisons among the CNN, the LSTM, and traditional methods. Section IV summarizes our conclusions and describes areas for future research.

II. MODEL AND METHODS

A. Preliminary

CNNs were designed for image classification; we thus first explain how a CNN recognizes images. LSTM is more appropriate for natural language processing; we next introduce the foundations of LSTM.

1) CNN and Image Analysis

For Fig. 1, the computer sees an array of pixel values rather than the image, and the resolution varies [12]. In the present case, the computer sees a 32×32×3 array of numbers. One pixel corresponds to a number from 0 to 255. The computer then calculates image probabilities to define the class including the picture. Here, dog is associated with an 80% probability, cat with 15% probability, and bird with 5% probability.

The computer distinguishes different pictures. When we see a picture of a cat, we search for features such as four legs or whiskers. The computer looks for low-level features (edges and curves) and then more abstract concepts established by the convolutional layers.

2) LSTM

LSTM is a special form of RNN that solves long-term dependency problems not addressed by conventional normal RNNs (Hochreiter and Schmidhuber [11], and later developments). LSTM affords excellent performance when used to address many problems; LSTM remembers information on a long-term basis. All RNNs require chains to interrogate NN modules. In a standard RNN, the repeating module is very simple, such as a single tanh layer. LSTM has a more complex structure. A normal RNN has a single neural network layer; LSTM features four interacting layers [13] (Fig. 2).

In the illustration, the entire vector is carried by each line from a single node output to the inputs of other nodes. The pink circles reflect operations, such as vector rises, as the NN layers learn the yellow boxes. Line combinations reflect a series of related transactions; line branching reflects the copying of content to different locations.

3) Using Pre-trained Word Embedding

"Word embedding" is a group of natural language- processing techniques mapping semantic meaning to geometric space [14]. We search a dictionary to assign a numeric value to every vocabulary, such that the L2 distance between any two vectors reflects the semantic relationship between two correlated vocabularies. These vectors constitute a geometric

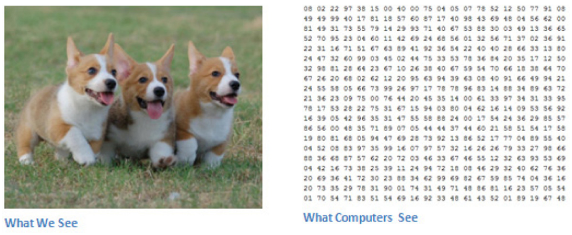


Fig. 1. The human and computer views. From Adit Deshpande's blog [12].

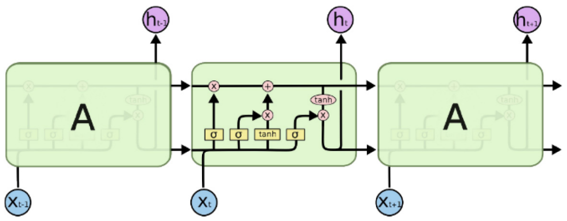


Fig. 2. An LSTM containing four interacting layers.

space termed the “embedding space”. For example, “mango” and “penguin” are vocabularies that are semantically very different; a logical embedding space will express them as widely separated vectors. But “love” and “marriage” are correlative vocabularies; they should be closely embedded. In theory, in a good embedding space, the vector path from “love” to “marriage” will precisely describe the semantic relationship between these two notions. In this circumstance, the relationship is “why x occurs”; one would hope that the vector love-to-marriage would explain this. Fundamentally, we require the vectorial characteristic: marriage + (why x occurs) = love. Next, we can use such vectors to answer questions. For example, we define a new vector “hurt”, and explore its relationships; we should capture: hurt + (why x occurs) = hit, thus a reply to “why hurt occurs?”.

We can compute word-embedding. When we explore a text corpus, we can count and analyze databases using dimensionality reduction techniques, employing a form of neural network exploiting the “word2vec” technique or matrix factorization.

B. The Use of CNN for Text Categorization

1) A Deeper CNN

We apply a CNN to text data. Kim [15] employed multiple filters implemented using a Keras Merge Layer (Fig. 3).

We used 128 filters (size 5) and maximum pooling of 5 and 35 (Fig. 4).

2) Text Categorization using a Sequential CNN

We have a document $D = (w_1, w_2, \ldots)$ and a vocabulary V. The CNN needs to preserve the word order when data are input. However, during text categorization using CNN, we can regard each word as a pixel, and D then equals $|D| \times 1$ pixel; we can also view $|V|$ as a channel. Thus, we can use a CNN to deal with text as if it were an image. As a simple example, consider the vocabulary $V = \{$“don’t”, “hate”, “her”, “like”, “you”$\}$ and a document $D =$ “you like her”. We obtain a document vector $x = [00001|00010|00100]^T$.

When a CNN is used to deal with images, we need to select the region size in advance; when that size is p, we can use $p|V|$ to express region vectors. Using the example above, suppose $p = 2$ and stride=1, we obtain two regions “you like” and “like her” (Table 1).

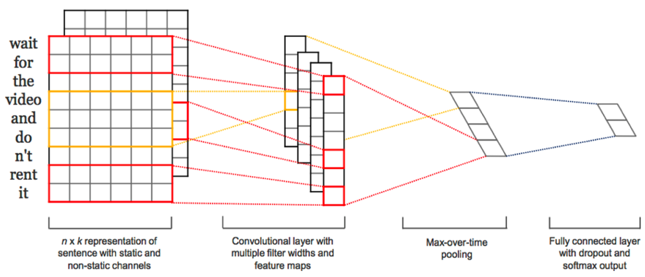

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

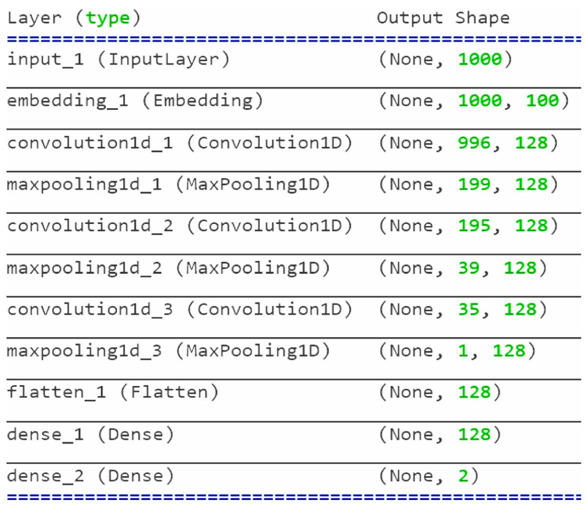
Fig. 3. A convolutional network with multiple filters.

As in image analysis, the text vector is translated to a feature vector. The convolution layer embeds text regions into low-dimensional vector spaces. However, a sequential CNN features a neural network, a convolution layer, and the region.

3) Use of a Disordered CNN for Text Evaluation

Sequential CNNs perform well, but they have one drawback. During imaging, only three RGB channels are required, but, for text, the channel number is $|V|$ (size of vocabulary), which is large. Thus, when p is also large, the number $p|V|$ (the weight vectors) will be very large, rendering training impractical. Thus, we use a different method; we ignore the sequence of the vocabulary, thus changing the region vectors from dimension $p|V|$ to dimension $|V|$ (Table 2).

Thus, fewer parameters are studied. Disordered convolution lies between bow vectors and sequential convolution. Word order is lost in only small regions.



```
Layer (type)                     Output Shape
=================================================
input_1 (InputLayer)             (None, 1000)

embedding_1 (Embedding)          (None, 1000, 100)

convolution1d_1 (Convolution1D)  (None, 996, 128)

maxpooling1d_1 (MaxPooling1D)    (None, 199, 128)

convolution1d_2 (Convolution1D)  (None, 195, 128)

maxpooling1d_2 (MaxPooling1D)    (None, 39, 128)

convolution1d_3 (Convolution1D)  (None, 35, 128)

maxpooling1d_3 (MaxPooling1D)    (None, 1, 128)

flatten_1 (Flatten)              (None, 128)

dense_1 (Dense)                  (None, 128)

dense_2 (Dense)                  (None, 2)
=================================================
```

Fig. 4. A convolutional structure for text categorization.

Table 1. The document vectors of a sequential CNN

		don’t	hate	her	like	you
$r_0(x)$	you	0	0	0	0	1
	like	0	0	0	1	0
$r_1(x)$	like	0	0	0	1	0
	her	0	0	1	0	0

Table 2. Document vectors of the disordered CNN

		don’t	hate	her	like	you
$r_0(x)$	you like	0	0	0	1	1
$r_1(x)$	like her	0	0	1	1	0

4) Pooling Layer for Text Categorization

The pooling layer must sometimes change. Unlike images of a fixed size, documents are variable in length. As the stride is fixed, the convolution layer output must vary in length (Fig. 5).

The convolution layer output may vary in length after standard pooling. Next, the variable will be output to another convolution layer. Finally, the fully connected layer will receive inputs of variable length; this is inappropriate, as these inputs must be of fixed length. Thus, we dynamically control the pooling region size to ensure that the inputs to the fully connected layer are of a fixed length.

5) A Collateral CNN: An Extension

Often, text categorization is performed using CNNs, which commonly have only two convolution and corresponding pooling layers. To improve model accuracy, we explored a collateral CNN featuring two or more convolution layers in parallel. Using this architecture, we found that the convolution-pooled pairs effectively dealt with differently sized regions, developing different vectors for each region but ensuring that the vectors to the final fully connected layer were of the same length.

C. LSTM for Text Categorization

1) Text Classification using LSTM

Next, we explore the text categorization framework "region embedding + pooling". We used an available region-embedding method [16] and found that LSTM was efficient. As no additional data or algorithms were required, we implemented end-to-end supervision. We then simplified the model, accelerating training and improving accuracy.

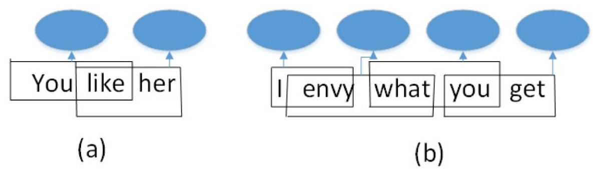


Fig. 5. The convolution layer for text of variable length.

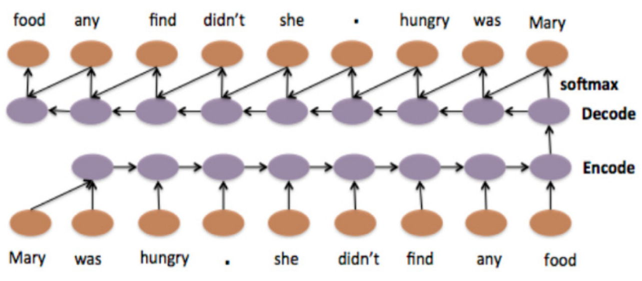


Fig. 6. The standard sequence-to-sequence model.

We encoded all text information in the last output of the RNN before running a feed-forward classification network very similar to a neural translation machine employing sequence-to-sequence learning (Fig. 6).

Also, we used bidirectional LSTM and concatenated both of the final outputs (Fig. 7).

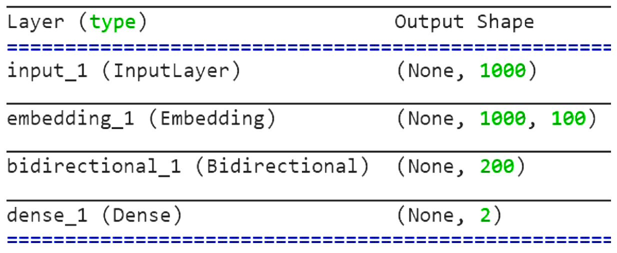
2) Simplifying Sub-problems of the Pooling Simplifying

First, we simplified the "region embedding + pooling" model. With LSTM, it is difficult to use a single vector to represent the entire document. We changed certain text regions to allow representation as simple vectors (Fig. 8).

Thus, we expressed each time step as h_t, and created a document vector by aggregating by pooling layer. No matter how many words are included, if the document is not finished, the LSTM must remember relevant information. Thus, we took a different route to solve the problem simply. We used segments smaller than phrases or sentences, with a focus on representing concepts, and allowed the forget gate to discard old data.

3) Increasing Training Speed

To simplify the sub-problem, we accelerated training in the pooling layer. We used chopping, and we allowed LSTM to embed text regions instead of documents. After this transformation, the sequence (from the beginning to the end of the document) is not important. We separated each document into fragments of a fixed length, and we assumed that these segments were individual documents, which could be han-



```
Layer (type)                    Output Shape
==================================================
input_1 (InputLayer)            (None, 1000)

embedding_1 (Embedding)         (None, 1000, 100)

bidirectional_1 (Bidirectional) (None, 200)

dense_1 (Dense)                 (None, 2)
==================================================
```

Fig. 7. The LSTM structure for text categorization.

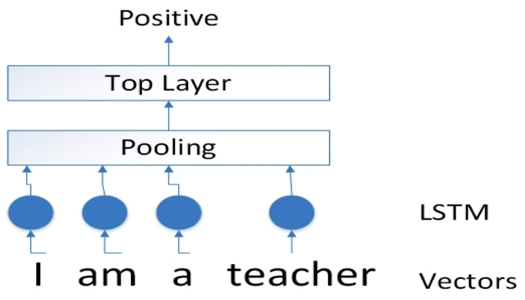


Fig. 8. LSTM featuring pooling.

dled simultaneously. This accelerated training and improved accuracy.

4) Elimination of Input and Output Gates

When we eliminated the input and output gates, accuracy was not affected. In other words, the pooling layer was redundant and the time and memory requirements halved. This is because the input and output gates eliminate irrelevant information, but our max-pooling method filters noise, allowing removal of the gates. The LSTM formulation simplifies to:

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{1}$$

$$u_t = \tanh(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(u)}) \tag{2}$$

$$c_t = u_t + f_t \odot c_{t-1}\,,\ h_t = \tanh(c_t) \tag{3}$$

This is equivalent to rendering all i_t and o_t constant. The model is mentally analogous to the gated recurrent units method [17] but is more straightforward, thus with fewer gates.

5) The Use of Two-Way LSTM to Improve Accuracy

The changes described above greatly reduced the time and memory required for training. If an opposite-direction pooling layer were to be added, what might happen? We coupled forward and backward LSTMs, which further improved accuracy.

III. RESULTS

A. Experimental Environment and Processes

1) Hardware Requirements

Our software runs only on GPUs, such as the Tesla K20. Testing was performed on a Tesla M2070 and a Quadro M6000.

2) Software Requirements

The makefile uses gcc; CUDA must be installed and CUDA v7.5 or higher is recommended. We do not know whether versions earlier than 5.0 work. Testing employed CUDA v7.5 and Linux. In principle, the code should compile and run in other systems as well (e.g., Windows), provided that a GPU and an appropriate version of CUDA are installed. However, we do not guarantee this. Optimally, Perl is required.

3) Experiments

We commence with a simple case lacking unsupervised embeddings. Using the context, the training and testing of neural networks feature the following steps:

a. Generate a vocabulary file from training data. Input a tokenized text file.
b. Generate input files for NN training, including a region file (containing features in the form of sparse region vectors); a target file (containing classification label information); and a word-mapping file (showing the mapping between words and the dimensions of the sparse region vectors).
c. Train the NN while (optionally) measuring classification error rates using validation and/or test data. Trained NNs can be saved.
d. Apply the trained NN to test data.

B. Experiments with the CNN

We explore two kinds of tasks, topic and sentiment classification.

1) Datasets and Data Preprocessing

(a) The Internet Movie Database (IMDB): movie reviews

IMDB dataset [18] is used for sentiment categorization. Movie reviews are evaluated as positive or negative. The dataset contains 25,000 reviews (training and test sets). First, we prepared the raw data; we stamped the text so that affective symbols were treated in advance and changed upper case to lower case.

(b) Elec: electronics product reviews

Elec contains reviews of electronic products and is a part of the Amazon review dataset [19]. Electronics are very different from movies; this is why we chose this dataset. As for IMDB, half the reviews are positive and half negative. We used 25,000 reviews for testing and more for training. Data pretreatment was essentially the same as that for IMDB.

(c) Reuters Corpus Volume 1 (RCV1): topic classification

RCV1 is a Reuters news article corpus described in [20]. There are 103 topic categories; a document might link to several topics. This affords the opportunity to optimize algorithms. There are two kinds of category: single-label (about 55) and multiple-label. Dataset pretreatment is rather complex, and stopwords must be dealt with in advance using RCV1.

(d) 20-Newsgroups (20NG)

29NG [21] includes about 20,000 documents divided into 20 different newsgroups. The contents include subject lines, signature files, quotes, and other texts.

2) Performance Results

Table 3 lists various classification methods, including our new methods, which outperformed all earlier methods.

In terms of the convolution layer, as IMDB and Elec are used to classify sentiments; we set the region size to 3, the stride to 1, the number of weight vectors to 1,000, and max-pooling on. The sequential CNN outperformed the disordered CNN, which, in turn, outperformed the traditional methods; the collateral disorder CNN was optimal. Thus, when the region size is small and max-pooling is chosen, a short phrase conveys a strong sentiment. The high score means that we can ignore the remainder of the review; sentiment classification is already effective.

As RCV1 and 20NG classify topics, we set the region size to 20, the stride to variable (≥2), the weight vector number to 1,000; and chose average-pooling. The disordered CNN outperformed the sequential CNN, which, in turn, outperformed traditional methods. Thus, for topic classification, a large context is more predictive than small segments; retention of word sequence in each region is important, but the use of fewer parameters is even more important, affording better results.

Next, we explored the collateral CNN. When applied to IMDB, this model (with two sequential convolution layers) outperformed sequential CNN. The extra neurons enhanced performance compared to that of the best-performing benchmark and the best former supervised result. Collateral sequential CNN yields good results although the predictive text regions vary in length. When the collateral disorder CNN was applied to IMDB, the error rate was only 8.21%; the model contained three collateral layers, including two sequential convolution layers, each with 1,000 neurons, and the entire document in a single layer with 20 neurons; a normal vector served as the input to the computation unit. However, use of the collateral disorder CNN to evaluate Elec yielded a different result; the best performance (the error rate is 7.68%) was afforded by changing the region sizes of the sequential convolution layers to 3 and 4. Thus, shorter word orders, a larger window in the global context, a disordered convolution layer evaluating n-gram words, and a larger localized region size, were all important.

Finally, we compared the various baseline methods of sentiment and topic categorization. If vocabulary were reduced, sentiment classification performance would fall, but this would not be the case for topic categorization, whether bi- or tri-grams are used.

3) Why is the CNN Effective?

Fig. 9 shows several text regions found using a sequential CNN to evaluate Elec. The structure includes one convolution layer of region size 3 and 1,000 neurons. After transmission by each layer, the top layer contains 1,000 vectors. The negative class in the top layer can be expressed using N_i (the ith highest weight) and the positive class employing P_i (the ith highest weight).

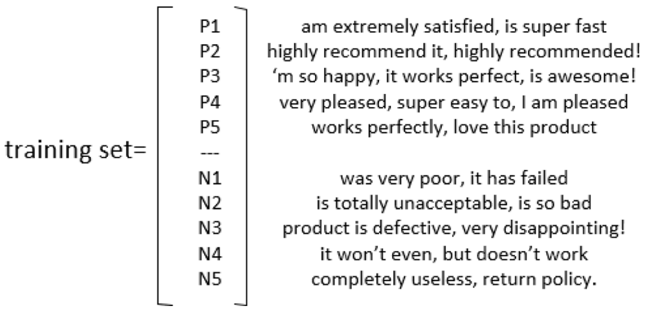
In traditional classification methods [22], only training data appear in test data, facilitating accurate prediction. However, the CNN is different; Fig. 10 show the test set of text regions; the test data were not in the training data (even in part). For example, the training data include the phrase "am extremely satisfied", but only the phrase "am entirely satisfied" appears in the test data; these phrases are very different. It is necessary to resolve this difference, but the sequential CNN can operate successfully even if the phrases are not identical. In another words, the CNN uses words effectively when traditional methods fail.

C. LSTM Experiments

We used four datasets (IMDB, Elec, RCV1, and 20-newsgroup) to directly compare the CNN, LSTM, and traditional

Table 3. Error rates (%) of our methods and earlier methods

	IMDB	Elec	RCV1	20NG
KNN	12.43	11.45	12.97	17.86
SVM	11.17	10.48	12.67	17.79
Neural network	10.08	10.06	12.52	17.29
Earlier CNN	9.66	9.43	10.75	15.42
Disordered CNN	9.17	8.90	**9.84**	**13.73**
Sequential CNN	8.91	8.16	10.48	15.72
Collateral sequential CNN	8.57	8.01	-	-
Collateral disordered CNN	**8.21**	**7.68**		

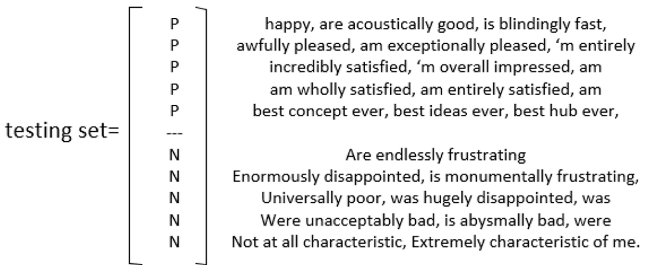


training set=

P1	am extremely satisfied, is super fast
P2	highly recommend it, highly recommended!
P3	'm so happy, it works perfect, is awesome!
P4	very pleased, super easy to, I am pleased
P5	works perfectly, love this product

N1	was very poor, it has failed
N2	is totally unacceptable, is so bad
N3	product is defective, very disappointing!
N4	it won't even, but doesn't work
N5	completely useless, return policy.

Fig. 9. Training set used for text region prediction.



testing set=

P	happy, are acoustically good, is blindingly fast,
P	awfully pleased, am exceptionally pleased, 'm entirely
P	incredibly satisfied, 'm overall impressed, am
P	am wholly satisfied, am entirely satisfied, am
P	best concept ever, best ideas ever, best hub ever,

N	Are endlessly frustrating
N	Enormously disappointed, is monumentally frustrating,
N	Universally poor, was hugely disappointed, was
N	Were unacceptably bad, is abysmally bad, were
N	Not at all characteristic, Extremely characteristic of me.

Fig. 10. Examples of predictive text regions in the testing set. These were not present (either entirely or partially) in the training set.

Table 4. Average and maximum lengths of documents and the class numbers

	Training data	Test data	Average length	Maximum length	Class number
IMDB	30,000	20,000	247	4K	2
Elec	30,000	20,000	138	5K	2
RCV1	17,219	53,747	199	11K	55
20NG	12,354	7,617	237	10K	20

Table 5. Error rates of various methods (%)

	IMDB	Elec	RCV1	20NG
KNN	12.43	11.45	12.97	17.47
SVM	11.17	10.48	12.67	16.85
Previous CNN	9.34	8.31	10.53	15.92
Previous LSTM	8.85	8.06	11.14	14.01
Our CNN	8.89	7.84	**9.96**	14.64
Our LSTM	**8.32**	**7.55**	10.67	**13.53**

methods. The datasets are summarized in Table 4.

We evaluated both sentiment classification (IMDB and Elec datasets) and topic classification (RCV1 and 20NG datasets). We first converted the data to lower-case letters and reduced the training data to the most frequent 30K words, reducing the computational burden. We set a Gaussian distribution with a zero mean, a standard deviation of 0.01, SGD optimization of momentum, and RMSProp to optimize acceleration [23].

The parameters can be varied based on performance; for example, various learning rates can be compared and an optimal rate selected. During pooling, we selected the parameters that best reflected the exploitation capacities of the data. For the IMDB and Elec datasets, we selected max-pooling with k=1. For the RCV1 dataset, we selected average-pooling with k=10. For the 20NG dataset, we selected max-pooling with k=10.

Table 5 shows the error rates; we did not use unlabeled data or engage in pre-training. The CNN featured a single convolution layer with 1,000 feature maps (thus, each location generated 1,000 dimensional vectors). We used a bidirectional LSTM in which each direction delivered a 500-dimensional vector at each time step. LSTM afforded certain advantages. First, LSTM used variably sized embedding regions [24]; the region size was fixed in the CNN. Second, although a CNN can feature multiple convolution layers with distinct region sizes, the LSTM outcomes were nonetheless better, perhaps because of the longer word sequences used by LSTM [25]. In terms of training speed, the CNN was faster than the LSTM, because region embedding is simple in the CNN. Thus, in the future, we will consider combining the two types of region embedding.

IV. DISCUSSION AND CONCLUSIONS

We found that our CNN and LSTM models were better than the traditional methods. Also, we improved existing deep learning methods as follows:

- Use of a disordered CNN to evaluate text. Fewer parameters must be learned; word order is lost only within small regions.
- Variable pool sizes for CNN text. The outputs to the fully connected top layer must be of a fixed length. As the number of pooling units is fixed, we dynamically determined pooling region size. We ensured that all data were covered, without overlaps.
- Use of a collateral CNN. We used several types of embedding for small text regions. As the various CNNs engage in mutual assistance, model accuracy was improved.
- Removal of input/output gates. On LSTM pooling LSTM, if the input and output gates were removed, model accuracy would not be compromised, but the times and memories required for training and testing would be halved.
- Bidirectional LSTM affords improved accuracy. Bidirectional LSTM reduces the time and memory required for training and improves predictive accuracy.

Using the benchmark datasets, our results were better than previous results. However, our methods have certain disadvantages, principally suboptimal efficiency. Because we simplified the training methods, the training times became longer than those of traditional methods, including deep learning methods. Therefore, we plan to further explore sophisticated deep learning methods such as Deep Pyramid. We hope that we can use these methods to improve classification accuracy and efficiency.

ACKNOWLEDGMENTS

This research was supported by the Ministry of Trade, Industry and Energy (MOTIE) of Korea through the education program for creative and industrial convergence (Grant No. N0000717).

REFERENCES

[1] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A Bayesian approach to filtering junk e-mail,” in *Proceedings of AAAI’98 Workshop on Learning for Text Categorization*, Madison, WI, 1998.

[2] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, pp. 79-86, 2002. DOI: 10.3115/1118693.1118704.

[3] B. Pang and L. Lee, “Opinion mining and sentiment analysis,”

Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008. DOI: 10.1561/1500000011.

[4] B. Li, N. Chen, J. Wen, X. Jin, and Y. Shi, "Text categorization system for stock prediction," *International Journal of u- and e-Service Science and Technology*, vol. 8, no. 2, pp. 35-44, 2015. DOI: 10.14257/ijunnesst.2015.8.2.04.

[5] X. Wang and H. C. Kim, "New feature selection method for text categorization," *Journal of Information and Communication Convergence Engineering,* vol. 15, no. 1, pp. 53-61, 2017. DOI: 10.6109/jicce.2017.15.1.53.

[6] A. McCallum and K. Nigam, "A comparison of event models for naïve Bayes text classification," in *Proceedings of AAAI'98 Workshop on Learning for Text Categorization*, Madison, WI, 1998.

[7] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," in *Proceedings IEEE International Conference on Data Mining*, San Jose, CA, pp. 647-648, 2001. DOI: 10.1109/ICDM.2001.989592

[8] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the 16th International Conference on Machine Learning*, Bled, Slovenia, pp. 200-209, 1999.

[9] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, Austin, TX, pp. 2267-2273, 2015.

[10] J. Weston, S. Chopra, and K. Adams, "#tagspace: semantic embeddings from hashtags," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1822-1827, 2014.

[11] S. Hochreiter and J. Schmidhuder, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[12] A. Deshpande, "A beginner's guide to understanding convolutional neural networks," 2016 [Internet], Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/.

[13] C. Olah, "Understanding LSTM networks," 2015 [Internet], Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[14] L. Xu, K. Liu, S. Lai, and J. Zhao, "Product feature mining: Semantic clues versus syntactic constituents," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD, pp. 336-346, 2014.

[15] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1746–1751, 2014.

[16] K. Tai, S. Richard, and M. Christopher, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing, China, pp. 1556-1566, 2015.

[17] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Y. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1724-1734, 2014.

[18] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, pp. 142-150, 2011.

[19] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in in *Proceedings of the 7th ACM Conference on Recommender Systems,* Hong Kong, China, pp. 165-172, 2013.

[20] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: a new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5. pp. 361-397, 2004.

[21] J. Gao, P. Pantel, M. Gamon, X. He, and D. Li, "Modeling interestingness with deep neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 2-13, 2014.

[22] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modeling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD, pp. 655-665, 2014.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, Beijing, China, pp. 1188-1196, 2014.

[24] P. Le and W. Zuidema, "Compositional distributional semantics with long short-term memory," in *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, Denver, CO, pp. 10-19, 2015.

[25] X. Zhu, P. Sobhani, and H. Guo, "Long short-term memory over recursive structures," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, pp. 1604-1612, 2015.

Xingfeng Wang

received the B.E. degree in application of electronic technology from Liaoning Normal University, China, in 1998 and the M.E. degree in computer technology from Dalian University of Technology, China, in 2009. He is a university teacher at Information Engineering College, Eastern Liaoning University, China and currently studies for Ph.D. at Department of Computer Engineering, Inje University, Korea. He has interests in the areas of data mining, neural network, and deep learning. He has published more than 10 papers in these areas.

Hee-Cheol Kim

received the M.S. degree in computer science from Sogang University, Korea, in 1991, and the Ph.D. degree in computer science from Stockholm University, Sweden in 2001. He is a professor at Department of Computer Engineering, Inje University, Korea. He has interests in the areas of human computer interaction, software engineering, and u-healthcare. He has published more than 100 papers in these areas.