A Text Classification Method Based on the Merge-LSTM-CNN Model

Kunfu Wang, Pengyi Zhang and Jian Su
System Engineering Research Institute of China State Shipbuilding Corporation, Beijing, China

Abstract. An MLCNN (Merge-LSTM-CNN) based text classification model combining a Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) is proposed due to CNN’s deficiency in obtaining context dependency in text and feature loss of the deep neural network in terms of text character extraction. First, the vector representation of input text is realized through word embedding, then full-text semantics are integrated by extracting the local features of the text through a three-layer CNN. Meanwhile, LSTM is used to store the features of historical information in the text to obtain its context-related semantics. Second, input vectors are integrated with the outputs of CNN in each layer respectively to allow the reuse of original features.

1. Introduction
Text classification is an important research subject in the field of natural language processing (NLP).[1] Able to effectively organize and manage complex text information, it has been widely used in such different fields as homepage retrieval, spam filtering and emotion analysis,[2,3] attracting the extensive attention of researchers. At present, common text classification methods falls into two types: the first is based on traditional machine learning methods (e.g. Support Vector Machine (SVM), Naive Bayes, decision tree and K-Nearest Neighbor (KNN)). The text representation of these methods is usually a high-dimensional sparse vector, but this has poor feature expression ability, so it requires artificial feature engineering,[4] bringing high costs in processing massive datasets and corpora. The second type is the text classification method based on the deep neural network. This method uses such models as CNN, Recurrent Neural Network (RNN) and LSTM to improve its structure for text classification. Deep learning was initially applied in the image processing and speech recognition fields where it achieved significant effects because the original data of images and speech is continuous and dense with local correlation. The key problem in text classification is text representation, which usually appears in the form of feature representation in traditional machine learning. The most common feature representation method is the bag-of-words model,[5] but this method cannot cover the relationships between words and also ignores word order. Differing from traditional machine learning, deep learning uses distributed representation[6] to train the text into a low-dimensional dense vector, then uses CNN and LSTM structures to automatically acquire feature expression capabilities, remove complex artificial feature engineering and solve problems efficiently.

In recent years, many scholars at home and abroad have studied how to improve the accuracy of text classification. Qiu Ningjia et al.[7-8] proposed an SVD-CNN bullet screen text classification algorithm model combining improved active learning to solve the problem that text feature dimensionality reduction based on traditional CNN models using a pooling layer will lose much textual semantic information. Liang et al.[9] proposed an emotion classification model based on polarity transfer and LSTM. First, in order to capture deeper semantic information, LSTM is extended
to a tree structure-based RNN, and second, the polarity transfer model is introduced based on the contextual correlations of words. Lu et al.[10] proposed a P-LSTM model for emotion classification according to the significant classification performance of LSTM in emotion classification, which uses three-word phrases as the input for vectorization. In addition, P-LSTM introduces a phrase factor mechanism to make information extracted from the text more accurate according to the feature vectors of the embedding layer and LSTM hidden layer.

In this paper, an MLCNN text classification model based on LSTM and CNN is proposed in which text after word embedding is integrated into the outputs of each convolutional layer to enhance the transmission of the original features. The experimental results show that this model has a better classification effect.

2. Related Research
With the prevalence of the mixed use of many neural networks in the fields of speech recognition and computer vision, the combination of CNN and LSTM models in the field of NLP has also increased. Zhang et al.[11] proposed a novel CNN-LSTM text classification model according to the advantages of two different deep learning models, i.e. CNN and LSTM. CNN-LSTM uses CNN to extract higher levels of word representation sequences and input them into LSTM to obtain sentence representations. This can capture the local features of the text and acquire the temporal semantics of the sentence, and performs well in emotional classification. For Chinese text classification, Li et al.[12] proposed a model combining bidirectional LSTM and CNN to automatically acquire word- and character-level features. First, bidirectional LSTM is used to capture the historical and future information of each moment, then carry out classification according to the extracted CNN features.

However, the increase in neural network depth also brings many problems. Model training becomes increasingly difficult, and the time complexity is much higher than that of the traditional machine learning method. Meanwhile, the learning period increases, the convergence speed becomes slower, the gradient disappears and the classification results are affected. The increase in the number of network layers will gradually lose the original features of the input, declining the effects of the classification model. In order to alleviate the above problems, inspired by the DenSeNet model theory proposed by Huang et al. and based on the advantages of LSTM and CNN, the MLCNN text classification model based on LSTM and CNN established in this paper allows the use of word embedding to express the input text as a low-dimensional vector, uses LSTM and CNN to process the word vector separately, extracted local features while taking context semantics into account, integrates the original input into the output of each convolutional layer to make the reuse of original features possible, and finally uses the Softmax function as the classification function to obtain classification results.

The remaining chapters of this paper are organized as follows: Section 3 gives a detailed description of the MLCNN model; Section 4 gives the experimental results and related parameters; and Section 5 summarizes this paper and lays out prospects for follow-up studies.
3. MLCNN Model
In this paper, both LSTM and CNN show their superiority in text classification tasks. To improve the classification effect, after using their respective advantages for reference, and considering feature loss caused by increasing the number of network layers, an MLCNN model is proposed which has the following main five parts:

1. Using word embedding for text representation;
2. obtaining context-related semantics through LSTM;
3. using CNN for text feature extraction;
4. realizing the reuse of original features by integrating input vectors;
5. using the Softmax function to obtain classification results.

The structure of the MLCNN model is shown in Figure 1, and the arrow in the figure indicates the transfer of the feature vector.

3.1. Text Representation
In NLP tasks, words or phrases are usually used as the basic units and represented by fixed-length real vectors. This method is called “word embedding” or “word vectors”. One-hot encoding is another method which is often used for showing words. In addition, the use of one-hot encoding in deep learning can lead to many problems such as the curse of dimensionality caused by the high dimensionality of word vectors. In order to avoid the above problems, word embedding is used in this paper to represent words as similar continuous dense real vectors in speech and images. These word vectors contain richer semantic information, enabling the MLCNN model to model more complex contexts. When the word in the text is represented by word vectors, each word is randomly initialized to a fixed-length vector. In this case, the \( t \) th word in the text is represented as \( x_t \in \mathbb{R}^n \), in which \( n \) is the dimension of the word vector. When the text length is \( T \), the input text is represented as:

\[
X = [x_1, x_2, \ldots, x_T] \in \mathbb{R}^{T \times n}
\]

As word vectors are randomly initialized, the word vector must be updated as long as the neural network model is trained in the experiment.

3.2. Acquiring Context-related Semantics
LSTM is an improved RNN-based network structure which effectively preserves long-sequence history information by adding a memory cell, input gate, forget gate and output gate, thereby mitigating information loss due to the large number of RNN training layers.

Figure 2. Structure of LSTM cell

The structure of LSTM is shown in Figure 2. The memory cell \( C_t \) is configured to store the current time history information. The input gate \( i_t \) determines how much the input vector changes the information in the memory cell at the current time. The forget gate \( f_t \) determines the extent to which previous historical information affects the information in the current memory cell. The output gate \( o_t \) is used to control the output of information in the current memory cell. When the input word vector matrix is \( X = [x_1, x_2, \ldots, x_T] \) and \( x_t \) is the dimension vector, the LSTM updating equations are:
Where $h_f$ is the final output of the LSTM unit, $\sigma(\cdot)$ is the Sigmoid activation function, $\tanh(\cdot)$ is a hyperbolic tangent function, $W_i$, $W_o$ and $W_f$ are the weight matrices of the input gate, output gate and forget gate respectively, and $b_i$, $b_o$ and $b_f$ are the bias terms of the three control gates respectively. In summary, LSTM will filter information based on the three gating units and accumulate information through linear self-joining memory cells as an intermediary to obtain the outputs of hidden layers at the current time.

3.3. Text Feature Extraction

CNN is the main method for extracting data features in deep learning. It is generally composed of six layers: among them, the input layer arranges the word vectors corresponding to words in the text from top to bottom into a matrix. The convolutional layer uses the convolution kernel to perform feature extraction and feature mapping on the text data. The excitation layer adds nonlinear mapping for linear convolution operations. The pooling layer is divided into maximum pooling and average pooling, which perform the downsampling and sparse treatment of feature vectors to reduce the amount of data operations. The fully connected layer usually refits the pooled features to reduce the loss of feature information. The output layer is used to output the results.

The main idea of CNN is local connection and parameter sharing. The convolutional layer is the core of CNN. It performs convolution operations on words to obtain more advanced feature representation. Each convolution kernel is subject to convolution operation with different local windows of input features. The eigenvector obtained by the operation is processed by the nonlinear activation function $f$ to generate the features to be output by the layer, and the equation is as follows:

$$O = f(Wx + b)$$  \hspace{1cm} (7)

Where $W$ is the convolution kernel, $x \in \mathbb{R}^{p \times m}$ is the input word vector matrix and parameter $b$ is the bias term. The most commonly used nonlinear activation functions are Sigmoid and ReLU.

In order to speed up the convergence of the model and reduce its learning period, the use of ReLU will significantly reduce the amount of computation in the entire learning process compared with such activation functions as Sigmoid. The equation for ReLU is as follows:

$$f(x) = \max(0, x)$$  \hspace{1cm} (8)

Where $x$ is the output vector of the neural network in the previous layer. The ReLU function changes the input negative value to 0 while the positive value remains unchanged. This operation makes the neural network sparse, reduces the interdependence of parameters and alleviates the over-fitting problem.

3.4. Reuse of Original Features

In this paper, the input vector processed by word embedding is merged with the output vector of each layer of CNN through the CONCAT operation, realizing the reuse of original features and reducing the feature loss caused by feature transmission between the middle layer and interlayer of the neural network. In Figure 1, when the input vector is subject to the CONCAT operation with the output vector of each layer of CNN, the dimensions of the vectors to be combined must be the same. In this
case, padding is added to the convolutional layer of CNN to make the dimensions of the input and output vectors of each layer of CNN consistent. The padding parameter has two modes, namely SAME and VALID. When padding is SAME, the size relationship between the input and output vectors is as shown in equation (9):

$$\lambda_{\text{output}} = \left\lceil \frac{\lambda_{\text{input}}}{S} \right\rceil$$  \hspace{1cm} (9)

When padding is VALID, the size relationship between the input and output vectors is as shown in equation (10):

$$\lambda_{\text{output}} = \left\lceil \frac{\lambda_{\text{input}} - F + 1}{S} \right\rceil$$  \hspace{1cm} (10)

In (9) and (10), $\lambda_{\text{input}}$ is the size of the input vector, $\lambda_{\text{output}}$ is the size of the output vector, $F$ is the size of the convolution kernel and $S$ is the step size. The mode of padding is set as SAME, which keeps the dimensions of the input vector and output vector the same.

When the input word vector matrix is $X^i = [x_1, x_2, \ldots, x_T]$, $x_i \in \mathbb{R}^n$ is the $n$-dimensional word vector corresponding to the $i$th word in the text. In the convolution kernel $W = [v_0, \ldots, v_{L - 1}]$, $\lambda$ is the number of words in the convolution kernel, and the following operations are performed:

$$O_i = \text{ReLU} \left( \sum_{i=0}^{T} W(x_{i+i}) + b \right)$$  \hspace{1cm} (11)

The obtained output matrix of the first layer of CNN can be represented as $O^1 = [o_1, o_2, \ldots, o_T]$. The CONCAT operation is performed on the original feature matrices $X^i \in \mathbb{R}^{T \times n}$ and $O^1 \in \mathbb{R}^{T \times n}$ to obtain the merged feature matrix $X^2 = [o_1, \ldots, o_T, x_1, \ldots, x_T]$; that is, CONCAT is a horizontal concatenation process on the one-dimensional array which does not involve such mathematical operations as adding or multiplying among the feature matrices. Thus, the problem that the training time is greatly increased due to the complexity is alleviated, and the influence of the fusion process on the model training is reduced.

$X^2 = [o_1, \ldots, o_T, x_1, \ldots, x_T]$ is used as the input matrix of the second layer of CNN. The convolution operation is performed according to equation (11) to obtain the output matrix $O^2 = [o_1, o_2, \ldots, o_T]$ of the second layer of CNN, in which $k$ is the number of word vectors. The $O^2 \in \mathbb{R}^{T \times n}$ subject to the CONCAT operation with the original feature matrix $X^4 \in \mathbb{R}^{T \times n}$ to obtain $X^3 = [o_1, \ldots, o_T, x_1, \ldots, x_T]$, which is input to the third layer of CNN. $O^3 \in \mathbb{R}^{T \times n}$ is output by the convolution operation, in which $j$ is the number of word vectors. The output matrices $O^4 \in \mathbb{R}^{T \times n}$ and $X^4 \in \mathbb{R}^{T \times n}$ of the third layer of CNN are subject to the CONCAT operation to obtain the input matrix $X^5 \in \mathbb{R}^{T \times n}$ of the next layer of the network structure, in which $m$ is the number of word vectors.

3.5. Acquisition of Classification Results

After the convolution operation, the extracted features are transferred to the pooling layer (Max Pooling) which further simplifies feature representation and reduces the dimensions of the feature vector. In Figure 1, the K-Max maximum pooling operation is performed by using $X^4 \in \mathbb{R}^{T \times n}$ as the input matrix of the pooling layer, and the K-Max pooling takes out K large eigenvalues in each convolution kernel and retains the original order of these eigenvalues. Therefore, the number of features is reduced and the most valuable feature information is obtained.

After the convolution and pooling operations, the output feature matrix and the output matrix of LSTM obtained by equation (6) show different dimensions, and the merge layer is used to calculate the word vector matrix output by different network structures. The two types of branches for independent feature learning are merged and transmitted to the fully connected layer. The fully
connected layer re-fits its input features, reducing the dimensions of the feature vectors. Dropout is used to prevent overfitting and improve the generalization of the model. Finally, the Softmax function is used to output the probability distribution of the category, and the probability of classifying \( x \) as category \( j \) is as follows:

\[
P(y = j|x; \theta) = \frac{e^{\theta_j^T x}}{\sum_{k=1}^{K} e^{\theta_k^T x}}
\]

The text classification algorithm of the MLCNN model based on MLTM and CNN is described as Algorithm 1:

Algorithm 1: Text classification algorithm based on the MLCNN model

**Input:** text collection  
**Output:** probability value of the category to which the text belongs

1. Begin /*Read the text, perform word segmentation on the text, and set the padding parameter mode to SAME*/
2. \( X^1 = \text{embedding}(T, n) \) /*Process text into word vector matrices \( X^1 \) containing \( T \times n \)-dimensional vectors by word embedding*/
3. FOR \( \theta = 1 \) to \( 3 \) DO //traverse the \( \theta \) th layer of CNN
4. \( O^\theta = \text{CNN}(X^\theta) \) /*The input matrix \( X^\theta \) of the \( \theta \) th layer of CNN is subjected to the convolution operation of equation (11) to obtain the output matrix \( O^\theta \) of the \( \theta \) th layer of CNN*/
5. \( X^{\theta+1} = \text{concat}(O^\theta, X^1) \) /*concatenate \( O^\theta \) with the original feature matrix \( X^1 \) to obtain the input matrix \( X^{\theta+1} \) of the next layer of the network structure*/
6. END FOR
7. \( \mu = \text{MaxPooling}(X^4) \) /*The input matrix \( X^4 \) is subject to K-Max maximum pooling to obtain the output matrix \( \mu \) of the pooling layer*/
8. \( H = \text{LSTM}(X^1) \) /*Through the calculation of \( X^1 \) by equations (2), (3), (4), (5) and (6), the output matrix \( H \) of LSTM is obtained. This step is performed in parallel with steps 3 to 7*/
9. \( M = \text{Merge}(\mu, H) \) /*Merge \( \mu \) and the output \( H \) of LSTM through the merge layer to obtain the input matrix \( M \) of the fully connected layer*/
10. \( Q = \text{FullyConnected}(M, \text{Dropout}) \) /*Re-fit \( M \) through the fully connected layer obtain the feature matrix \( Q \), and use \( \text{Dropout} \) to prevent overfitting*/
11. \( P = \text{softmax}(Q) \) /*Call the \( \text{softmax} \) function to obtain the probability value \( P \) of the category to which the text belongs*/
12. End

4. Summary

This paper proposes an MLCNN model for text classification which converts text into low-dimensional word vectors through word embedding, adopts CNN to extract the local features of text and combines LSTM to preserve the feature of historical information in text sequences so as to make up for the insufficiency of CNN in extracting contextual association semantics. The merging of the original input with the output of each layer of CNN reduces the original feature loss and improves the accuracy of text classification. In the next phase of research, we will try to introduce the attention mechanism mentioned in order to explain the importance of each sentence and word to its classification category intuitively. Second, by properly increasing the number of network layers to cope with more complex long text classification problems, a better deep learning model will be explored and the accuracy of text classification will be improved.
5. References

[1] Lin Yi-ou, Lei Hang, Li Xiao-yu, et al. Deep learning in natural language processing: methods and applications [J]. Journal of University of Electronic Science and Technology of China, 2017, 46(06): 913–919.

[2] LEE G, JEONG J U, SEO S, et al. Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network [J]. Knowledge-Based Systems, 2018, 152: 70–82.

[3] ZHANG D, HUA X, SU Z, et al. Chinese comments sentiment classification based on word2vec and SVM perf[J]. Expert Systems with Applications, 2015, 42(4): 1,857–1,863.

[4] SHANG C, LI M, FENG S, et al. Feature selection via maximizing global information gain for text classification[J]. Knowledge-Based Systems, 2013, 54 (Complete): 298–309.

[5] ZHANG Y, JIN R, ZHOU Z H. Understanding bag-of-words model: a statistical framework[J]. International Journal of Machine Learning and Cybernetics, 2010, 1(1–4): 43–52.

[6] KIM H K, KIM H, CHO S. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation [J]. Neurocomputing, 2017, 266: 336–352.

[7] Qiu Ning-jia, Cong Lin, Zhou Si-cheng, et al. SVD-CNN screen text classification algorithm with improved active learning [J]. Journal of Computer Applications, 2019, 39(03): 644–650.

[8] WU H, LI D, CHENG M, et al. Chinese Text Classification Based on Character-Level CNN and SVM [C]. International Symposium on Intelligence Computation and Applications (ISICA), 2018: 227–238.

[9] Liang Jun, Chai Yu-mei, Yuan Hui-bin, et al. Emotional analysis based on polarity transfer and LSTM recursive network [J]. Journal of Chinese Information Processing, 2015, 29(05): 152–159.

[10] LU C, HUANG H, JIAN P, et al. A P-LSTM Neural Network for Sentiment Classification [J]. Computer Science, 2017, 10234: 524–533.

[11] Zhang Y, Yuan H, Wang J, et al. YNU-HPCC at EmoInt-2017: Using a CNN-LSTM Model for Sentiment Intensity Prediction [C]. Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2017: 200–204.

[12] Li Y, Wang X, Xu P, et al. Chinese Text Classification Model Based on Deep Learning [J]. Future Internet, 2018, 10(11): 113.