2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

Deep Neural Networks for multi-class sentiment classification

Bohang Chen[1] Qiongxia Huang[1] Yi-Ping Phoebe Chen[2] Li Cheng[1] Riqing Chen[1]*
[1]Digital Fujian Big Data Institute of Agriculture and Forestry, College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou, China
[2]Department of Computer Science and Information Technology, La Trobe University, VIC3186, Australia
fafu_bohang@163.com, jonesky_hqx@163.com, Phoebe.Chen@latrobe.edu.au, li.cheng@fafu.edu.cn
riqing.chen@fafu.edu.cn

Abstract — Traditional text sentiment analysis methods often ignore context information when used in the expression of features. The position of the words in the text makes it difficult to achieve satisfactory results in semantic realization. In recent years, deep learning has obtained good results in text sentiment analysis tasks. Convolutional neural network (CNN) and Recurrent Neural Network(RNN) are two mainstream deep learning algorithms. In this paper, a deep sentiment representation model based on CNNs and long short-term memory recurrent neural network (LSTM) is proposed. The model uses two layers of CNNs to capture the partial features of the text. The model can capture more accurate partial features, after which the features are fed to the LSTM, which can capture the contextual information. Finally, we combine the improved deep learning model with a one-versus-rest training mechanism and apply it to multi-class sentiment classification. We evaluate the proposed model by conducting experiments on datasets. Experimental results demonstrate that the model we designed for multi-class sentiment analysis achieves a accuracy of 78.42% on data set D1 is better than the existing SVMs (support vector machines), CNN, LSTM and CNN-LSTM.

Keywords-Sentiment representation, CNN, LSTM, one-versus-rest training mechanism, sentiment classification.

I. INTRODUCTION

The rapid development of social media networks has subtly changed user behaviour on the Internet. People are increasingly willing to comment on various online platforms rather than passively accessing network knowledge. Therefore, a large number of subjective comments expressing the user's emotions and opinions have emerged on the Internet media.These comments contain information on the emotional polarity of people who are dealing with certain events. Finding an effective way to mine and analyze these data is very important research in the field of natural language processing (NLP). This research is called sentiment text analysis [1]. Through the sentiment analysis of these comments, we can have a better understanding of users' behaviors, find users' preferences for a product, and focus on the hot issues [2].

At present, traditional sentiment analysis tasks are usually based on lexicon or statistical machine learning, which has achieved good results in various applications [3]. The former mainly relies on the lexical datasets and pointwise mutual information (PMI) to judge the emotional tendency of new words, and then sentiment analysis of the whole text is undertaken [4]. The latter usually uses the bag-of-words to represent the text as a fixed-length vector and applies supervised learning to classify the sentiment of the text. However, the bag-of-words model suffers from the curse of dimensionality, and the words are isolated without any connection, so this model is only capable of superficial textual learning, and the lexicon cannot tap into rich relationship information [5]. Distributional representation is the entry point of the combination of deep learning and NLP. These distributed features are learned through a neural network language model. The development of word embedding shows the significance of deep learning in NLP, and its emergence is a breakthrough in the field of sentiment expression. In recent years, this development has escalated the research into neural networks in NLP, along with the release of the word vector tool Word2Vec [6-7], which also became the basis of introducing deep learning in the field of NLP. From 2014, researchers have used different deep neural network (DNN) models, such as convolutional neural networks (CNNs), and recurrent neural networks (RNNs) to make significant progress in traditional NLP applications, including machine translation(Lample et al., 2016)[8] and question answering(Golub and He, 2016)[9].

CNNs (Kim, 2014; Kalchbrenner et al. 2014) [10-11], RNNs (Irsoy and Cardie, 2014) [12] and long short-term memory (LSTM) (Tai et al., 2015; Zhu et al., 2015) [13-14] are generally combined with sequence-based or tree-structured models, which have been successfully employed for sentiment analysis. The sequence-based model obtains deep semantic information by learning the relationships between words which are in different positions of sequence. The tree-structured model learns syntactic information by converting text into a syntactic parsing tree, where each node in the tree corresponds to each word in the text. Thus, deep learning can automatically learn deep sentiment representation on the basis of word embedding, which captures the meaningful features. CNNs can significantly reduce computational complexity as well as the number of training parameters due to its unique weight-sharing structure. In the current deep neural networks model, words

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00142

are usually expressed by word vectors, and texts are based on the word vectors corresponding to words, which construct the matrix expression. The quality of word vectors is directly related to the final model's classification ability. The network fitting classification will be worse when the original vector of the input word contains semantic noise. To avoid the gradient exploding or vanishing problem in standard RNNs, Hochreiter and Schmidhuber [15] proposed that long short-term memory (LSTM) achieved better memory and memory accessing, Cho et al. [16] improved LSTM and applied it to machine translation.

In order to solve the limitations of the aforementioned model, we present a new convolutional and long short-term memory neural network model based on one-versus-rest training method and apply it to the multi-classification sentiment analysis.

II. Related Work

Sentiment analysis techniques can be divided into rule-based methods and statistical-based methods. At present, machine learning is the main method applied to sentiment analysis. Relying on text classification technology in traditional natural language processing, Pan et al. [17] used naive Bayes, support vector machines and the maximum entropy model, and achieved good results on film reviews. Ding et al. [18] proposed the emotional word pairing method to determine emotional polarity based on customer reviews of products.

With the successful application of deep learning methods in computer vision and speech recognition, an increasing number of deep learning technologies have also been applied to natural language processing. Bengio et al. [19] put forward a method of constructing language model by using neural network which mapped word vectors to low-dimensional space and measured the similarity between word and word by distance. Yoon Kim et al. [20] used different lengths filters to complete the convolution of the text matrix, where the width of the filter is equal to the length of the word vector. Then, max-pooling is used to manipulate the vectors extracted by each filter which corresponds to a number, splicing these filters gets a embedding vector to represent the sentence. Recurrent neural networks are a type of neural network that connects the hidden layer to itself. Compared with convolution neural networks, recurrent neural networks can be used to calculate the hidden layer for the next hidden layer, so it can be used to deal with time series problems, such as text generation [21], machine translation [22] and speech recognition [23]. In addition, Tai et al. reformed the recursive neural network using the cell block of LSTM model, which represented every non-leaf node of the tree structure of the network and designed a new model to improve semantic representation [24].Then, Zhou et al. [25] proposed a novel and unified model called C-LSTM for sentence representation and text classification. As this model is able to capture both the local features of phrases as well as global and temporal sentence semantics, the classification effect is better than the traditional LSTM. In the same year, Lai et al. [26] proposed a new recurrent convolutional neural network model and used it in text classification.

In this paper, two layers of CNN with different convolution kernel sizes are used to deal with text data, so our architecture enables LSTM to learn long-range dependencies from higher-order features. Finally, we use a one-versus-rest mechanism to train the model.The experiments demonstrate clearly that our model is superior to a single CNN, LSTM model, and the CNN-LSTM model.

III. The Proposed Approach

A new approach based on CNN and LSTM is proposed in order to capture deep sentiment features. The network structure of the proposed approach is shown in Fig.1.

The 3-CNN layer in the figure indicates that the

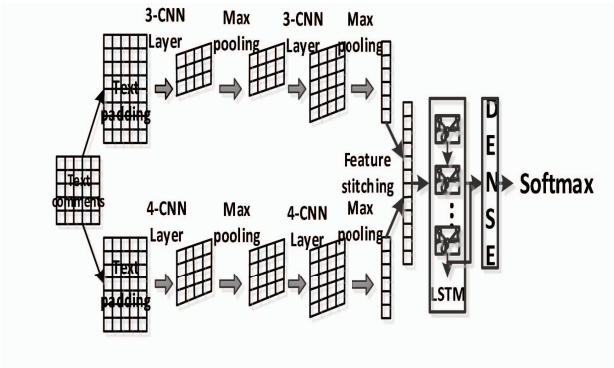

Fig. 1. The architecture of our model for sentence modeling.

convolution layer uses a convolution kernel with a height of 3 units. The 4-CNN layer indicates that the convolution layer uses a convolution kernel with a height of 4 units. The width of all convolution kernels is equal to the word vector and all window movements are 1 unit. This model has a two-layer CNNs and a one-layer LSTM network that are stacked in order. Each word in the comment text is represented by a pre-trained word vector. The model fills the initial text with 0 vectors. The filled text is sent to different convolution layers, and each convolution layer is followed by the largest pool. In the operation, the window movement step corresponding to the pooling layer in the middle of two convolutional layers is set to 15 units. The window movement step corresponding to the pooling layer in front of the LSTM model is set to 1 unit, and the features learned in the pooling layer are finally spliced as the input of the LSTM model. The output of the last moment in the LSTM model is the deep representation of the text that the model ultimately learns. With the deep representation, the polarity of the text can be discriminated by using the fully connected layer and the softmax function.

The following provides some special symbols to help explain the convolution calculation process of this model. Suppose l is the length of the comment text, and the dimension corresponding to the word vector is denoted as

d . The corresponding lengths of the top and bottom filled-in texts are respectively denoted as l_1 and l_2 . As shown in formula (1) and formula (2), the upper and lower filled texts are respectively denoted as $X_P \in R^{l_1 \times d}$ and $X_d \in R^{l_2 \times d}$. x_i^p is the word vector of the i-th position of X^P , x_i^d is the word vector of the i-th position of X^d .

$$X_P = \left\{x_1^p, x_2^p, \cdots x_{l_1}^p\right\} \tag{1}$$

$$X_d \in \left\{x_1^d, x_2^d, \cdots x_{l_2}^d\right\} \tag{2}$$

As shown in formulas (1) and (2), $X_{i:j}^p$ is the stitching of the i-th to j-th word vectors in X_p , and $X_{i:j}^d$ is the stitching of the i-th to j-th word vectors in X_d .

$$X_{i:j}^p = x_i^p \oplus x_{i+1}^p \oplus \cdots \oplus x_j^p \tag{3}$$

$$X_{i:j}^d = x_i^d \oplus x_{i+1}^d \oplus \cdots \oplus x_j^d \tag{4}$$

The model respectively uses X_p and X_d as inputs for 3-CNN and 4-CNN. Equations (5) and (6) perform the convolution calculation to obtain the feature maps C_p and C_d corresponding to 3-CNN and 4-CNN,

$$C_p = f(W_p \circ X_{i:j+h_p-1}^p + b_p) \tag{5}$$

$$C_d = f(W_d \circ X_{i:j+h_p-1}^d + b_d) \tag{6}$$

where $\circ$ is an element-wise multiplication, W and b respectively denotes the weight matrix and bias, and f is the ReLU function.

After obtaining the convolutional calculated feature map, the maximum pooling method is used to further reduce the dimension of the feature map. As shown in Fig.1, the model performs two convolution-pooling operations on the upper and lower rows, the extracted features are combined as the input to the LSTM and the output of the last time of the LSTM is used as the input to the fully connected layer. Let $Y = \left\{y_1, y_2, \cdots, y_k\right\}$ be the output of the fully connected layer, where k refers to the total number of categories and Y as the input to the softmax layer, available:

$$P_i = softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}} \tag{7}$$

where P_i is a probability value corresponding to the i-th category in Y . The category corresponding to the maximum probability value is the final judgment type of the text in the model, the formula can be expressed as:

$$category = \arg\max_{i=\{1,2,\cdots,k\}}(p_i) \tag{8}$$

At the same time, we also propose a one-versus-rest training method combine with our model, as shown in Fig.2. The one-vs-rest method involves training a single classifier per category, where the samples of that category are positive and all the other samples are negative.

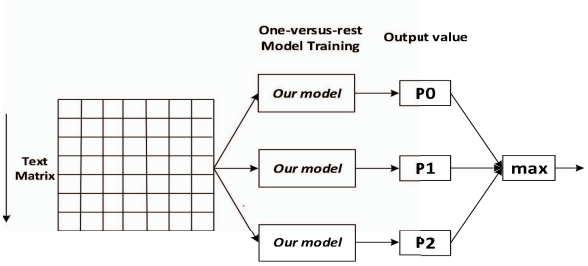

Fig. 2. The model training method.

In this paper, our samples has three categories namely A, B, C. We divide the samples into three training sets, extract the category corresponding to A as positive set, B and C are the negative sets. The category corresponding to B is a positive set, A and C are negative sets. The category corresponding to C is a positive set, A and B are negative sets. These three training sets are trained separately, and three training result files are obtained.

During testing, the corresponding test vectors are tested using these three training results files. A decision is made by applying all the classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score:

$$\hat{y} = \underset{k\in\{1,2,3\}}{argmax}(P_k) \tag{9}$$

The final result is the largest values of the P_k .

A. Convolutional Layer

The word vectors are generated by the Google open source Word2Vec toolkit. We treat each sentence as a single text matrix. Let l be the length of a sentence, and d is the dimensionality of word vectors. The word vectors in the sentence are stacked into a sentence matrix $S \in \mathbf{R}^{l \times d}$. In the convolutional layer, the filter usually selects the size of $n \in R^{h \times d}$, and h represents how many adjacent words slide in a window to get the word vectors, as follows:

$$w_{i:i+h-1} = [w_i, w_{i+1}, \ldots, w_{i+h-1}] \tag{10}$$

Here, the commas represent row vector concatenation. The resulting word vectors in a window are convoluted with the filter, generating a feature map $p_j \in R^{l-h+1}$, and each element p_j of the feature map for window vector is calculated as follows:

$$p_j = f(n \circ w_{i:i+h-1} + b) \tag{11}$$

where $\circ$ is an element-wise multiplication, w and

b respectively denotes the weight matrix and bias, and f is the ReLU function. For m filters of the same length, the generated feature maps can be rearranged as feature representations for each window c_j,

$$C = [p_1; p_2; \cdots; p_m] \tag{12}$$

Here, the semicolons represent column vector concatenation and $C \in R^{m\times(l-h+1)}$. In order to get more contextual information and to eliminate the effects of noise, we use two filters of different sizes for the convolution operation, then they are re-entered into the CNN repeat convolution operation, and the new higher-order window representations are fed into LSTM to preserve the sequentiality of the original sentences. We use C_{new} as the input of the one-layer LSTM where C_{new} is produced as follows:

$$C_{new} = C^T \tag{13}$$

where T denotes matrix transpose.

B. Max-pooling Layer

The main function of max pooling is to remove unnecessary redundant information in the convolutional feature map without damaging the recognition result. There are two reasons to use a max-pooling layer. First, by eliminating non-maximal values, it reduces computation for upper layers. Second, it can extract the local dependency within different regions to keep the most salient information. The obtained region vectors are then fed into LSTM layer.

C. LSTM Layer

Long short-term memory network is capable of learning long-term dependencies. It was first proposed by Hochreiter and Schmidhuber in 1997 [15] and were refined and popularized by many researchers in subsequent work. They work extremely well for a large variety of problems and are now widely used. In our model, we send the feature matrices from the convolutional operations into the LSTM network, which processes the input vectors by introducing a memory cell that is able to preserve the state over long periods of time. LSTM cells contain three additional neural gates: an input gate i_t, a forget gate f_t and an output gate o_t. The cell is dependent on the old hidden state h_{t-1} as well as the current input x_t. The LSTM transition functions are defined as follows:

$$\begin{aligned}
i_t &= \sigma(W^{(ix)}x_t + W^{(ih)}h_{t-1} + b^{(i)}) \\
f_t &= \sigma(W^{(fx)}x_t + W^{(fh)}h_{t-1} + b^{(f)}) \\
o_t &= \sigma(W^{(ox)}x_t + W^{(oh)}h_{t-1} + b^{(o)}) \\
u_t &= \tanh(W^{(ux)}x_t + W^{(uh)}h_{t-1} + b^{(u)}) \\
c_t &= i_t \circ u_t + f_t \circ c_{t-1} \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned} \tag{14}$$

where $\circ$ stands for element-wise multiplication, σ is a sigmoid function that has an output in the range of [0,1], $W^{(ix)}, W^{(fx)}, \cdots, W^{(oh)}, W^{(uh)}$ denote the weights to learn during the process of the model training, and $b^{(i)}, b^{(f)}, b^{(o)}, b^{(u)}$ denote the weight matrix. In addition, the value of i_t, f_t and o_t are in the range of [0, 1].

IV. Performance Evaluation

We apply the designed model for deep representation based on CNN and LSTM to the three categories of sentiment classification to evaluate its effectiveness. In this section, we describe the experiment setting, experiment results, and model analysis in detail.

A. Experiment Setting

We use one dataset to conduct a series of experiments. The dataset has three categories of sina micro-blog comments, captured through a Web crawler and is denoted as D_1. The sentiment polarity of each instance of D_1 is marked as either positive, negative or neutral. All comments are in the Chinese and the percentages of positive comments, negative comments and neutral comments are all 1/3. D_1 contains 10776 comments, 7/8 of which are used for training sets and 1/8 of which are used as test sets.

The proposed model based on one-versus-rest training method is compared with five methods, namely Kim-CNN[20], LSTM, CNN-LSTM, SVM and the proposed model based on general training method. We also use the Word2Vec toolkit to produce word vectors and concatenate them as the input of CNN, LSTM, and CNN-LSTM. The length of the word embedding is set to 20 and is learned from D_1. In contrast, the input of SVM is processed by a bag-of-word model. The accuracy and $F1$ values are used to evaluate the performance of the model. The function is defined as follows:

$$F1 = \frac{2\times P\times R}{P+R} \tag{15}$$

where P is the accuracy, which predicts the proportion of correct positive case data as the positive example data, and R represents the recall rate, which is the number of correct results divided by the number of results that should have been returned.

Our model is implemented on Tensorflow, which has

emerged as a popular Python library. For the hyper parameters settings, the number of filters of width 3 is set to 150, the number of filters of width 4 is set to 150, the memory dimension of LSTM is set to 150, the probability of the dropout layer is set to 0.5, and the Gaussian noise factor is set to 0.01.

B. Experiment Results and Analysis

TABLE I
ACCURACY AND F1 VALUE OF FIVE MODELS ON D_1

Model	F1	Accuracy
SVM	74.70%	74.18%
CNN	76.26%	75.97%
LSTM	75.71%	74.70%
CNN-LSTM	76.24%	76.04%
Our model	77.60%	77.53%
Our model based on one-versus-rest	78.42%	78.42%

Comparison of Methods

SVM We adopt the SVM model of Jonachims T[27]. The core idea of the SVM algorithm is to find a hyperplane with maximum separation from each category of data in the feature space corresponding to a given training set .

CNN We select a convolutional neural network(Kim, 2014)[10]. The input layer is a matrix of sentences. Each row is a word2vec word vector. This is followed by a convolution layer consisting of several filters, the largest pooled layer, and finally the softmax classifier.

LSTM We adopt the standard architecture (Hochreiter and Schmidhuber, 1997) in this work. The model was proposed to avoid the gradient exploding or vanishing problem in standard RNNs.

CNN-LSTM We adopt the model of Zhou[25] is able to capture both local features of phrases as well as global and temporal sentence semantics.

The experiment results of our model and the other five models on data set D_1 are shown in Table 1. It can be seen that the accuracy of the proposed model on D_1 is 78.42%, which is 4 percentage points higher than that of SVM, and the F1 value of the proposed model on D_1 is 0.7842, which is 4 percentage points higher than that of SVM. It is clear that our model achieve the best performance on D_1. SVM obtains an accuracy of 74.18% and an F1 value of 0.747, which is the worst performance of all the models. On the contrary, our model not only extracts the n-gram features, it also captures the long-term dependencies, hence our model achieves the best performance on D1.

It is clear to see that individual CNNs cannot learn long-term dependencies, and individual LSTMs cannot capture local characteristics. SVM feature representation methods often ignore contextual information and cannot achieve satisfactory results in acquiring semantics. By combining CNN and LSTM, the classification accuracy of CNN-LSTM is improved, but the value of F1 is lower than CNN[20]. This may be because the output of CNN is sent directly to the feature information contained in LSTM, and the orderliness of the original data cannot be maintained. The model we proposed has two filters of different sizes. The extracted eigenvectors have a strong generalization ability for the model, while the second convolutional layer can learn the deep expression of the features and send it to the LSTM which can capture the context information, ensuring our model has a very good capability to extract deep emotions. Finally, we combine this model with a one-versus-rest training mechanism to extract rich features from the current category. This feature has a strong characterization effect for this category. The experiment results show that our model achieved good results in multi-class sentiment classification due to its special network structure and new training mechanism.

V. CONCLUSIONS AND FUTURE WORK

Our work mainly consists of the following two contributions: firstly, when learning the expression of words, we use different convolution layers to obtain textual feature information, and then send it into the new convolution layer. Compared with traditional convolutional neural networks, this structure can solve the problem of data sparseness and reduce the noise of text feature information. Finally the text information obtained from the convolutional layer is fed into LSTM for output, so as to preserve the word order information. Secondly, we train our model combined with one-versus-rest mechanism. Compared to the original model, one-versus-rest mechanism can learn more detailed features for the separate categories, which has a strong representation effect for this category. Therefore, our model achieves good performance on sentiment classification. In future work, we propose to study the combination of word embedding technology to produce better pre-training features as the network input while trying to combine it with the new neural network model.

ACKNOWLEDGMENT

This work was supported by Fujian Agriculture and Forestry University Award KFA 17032A.

REFERENCES

[1] Pang B, Lee L. Opinion mining and sentiment analysis[J]. Foundations and Trends® in Information Retrieval, 2008, 2(1–2): 1-135.

[2] Ding X, Liu T, Duan J, et al. Mining User Consumption Intention from Social Media Using Domain Adaptive Convolutional Neural Network[C]//AAAI. 2015, 15: 2389-2395.

[3] Giachanou A, Crestani F. Like it or not: A survey of twitter sentiment analysis methods[J]. ACM Computing Surveys (CSUR), 2016, 49(2): 28.

[4] Turney P D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews[C]//Proceedings of

the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002: 417-424.
[5] Le Q, Mikolov T. Distributed representations of sentences and documents[C]//International Conference on Machine Learning. 2014: 1188-1196.
[6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.
[7] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
[8] Lample G, Ballesteros M, Subramanian S, et al. Neural architectures for named entity recognition[J]. arXiv preprint arXiv:1603.01360, 2016.
[9] Golub D, He X. Character-level question answering with attention[J]. arXiv preprint arXiv:1604.00727, 2016.
[10] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.
[11] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences[J]. arXiv preprint arXiv:1404.2188, 2014.
[12] Irsoy O, Cardie C. Opinion mining with deep recurrent neural networks[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 720-728.
[13] Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks[J]. arXiv preprint arXiv:1503.00075, 2015.
[14] Zhu X, Sobihani P, Guo H. Long short-term memory over recursive structures[C]//International Conference on Machine Learning. 2015: 1604-1612.
[15] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
[16] Cho K, Van Merri ë nboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014..
[17] Pang B, Lee L. Opinion mining and sentiment analysis[J]. Foundations and Trends® in Information Retrieval, 2008, 2(1–2): 1-135.
[18] Ding X, Liu B, Yu P S. A holistic lexicon-based approach to opinion mining[C]//Proceedings of the 2008 international conference on web search and data mining. ACM, 2008: 231-240.
[19] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of machine learning research, 2003, 3(Feb): 1137-1155.
[20] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.
[21] Sutskever I, Martens J, Hinton G E. Generating Text with Recurrent Neural Networks[C]// International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July. DBLP, 2011:1017-1024.
[22] Cho K, Van Merri ë nboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.
[23] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks[C]//International Conference on Machine Learning. 2014: 1764-1772.
[24] Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks[J]. arXiv preprint arXiv:1503.00075, 2015.
[25] Zhou C, Sun C, Liu Z, Lau F. A C-LSTM neural network for text classification[J]. arXiv preprint arXiv:1511.08630, 2015.
[26] Lai S, Xu L, Liu K, et al. Recurrent Convolutional Neural Networks for Text Classification[C]//AAAI. 2015, 333: 2267-2273.
[27] Joachims T. Text categorization with support vector machines: Learning with many relevant features[C]//European conference on machine learning. Springer, Berlin, Heidelberg, 1998: 137-142.