

Sentiment Analysis of Movie Reviews Based on CNN-BLSTM

Qianzi Shen, Zijian Wang, and Yaoru Sun^(✉)

Department of Computer Science, Tongji University, Shanghai, China
{sqz, 1410482, yaoru}@tongji.edu.cn

Abstract. Sentiment analysis has been a hot area in the research field of language understanding, but complex deep neural network used in it is still lacked. In this study, we combine convolutional neural networks (CNNs) and BLSTM (bidirectional Long Short-Term Memory) as a complex model to analyze the sentiment orientation of text. First, we design an appropriate structure to combine CNN and BLSTM to find out the most optimal one layer, and then conduct six experiments, including single CNN and single LSTM, for the test and accuracy comparison. Specially, we pre-process the data to transform the words into word vectors to improve the accuracy of the classification result. The classification accuracy of 89.7% resulted from CNN-BLSTM is much better than single CNN or single LSTM. Moreover, CNN with one convolution layer and one pooling layer also performs better than CNN with more layers.

Keywords: Natural language processing · CNN · LSTM · Sentiment analysis

1 Introduction

As the significant carries of emotion, texts are in an irreplaceable position on the Internet. The sentiment analysis is usually used to measure the emotional attitude of texts and related studies grow quickly. In the literature, Hazivassiloglou and McKeown investigated semantic orientation of adjectives based on cluster methods [1]. Turney and Littman chose seven pairs of words that have strong orientation and then set SO-PMI (semantic orientation-pointwise mutual information) to judge a word with a standard word. According to SO-PMI, they chose two groups of words in which one is positive (P word) and the other is negative (N word) as standard words [2]. Kim chose a method based on sentiment knowledge and summed up the weighing of phrases and describing words [3]. Pang studied sentiment analysis of English texts with machine learning [4]. With the development of artificial intelligence and deep learning, more and more related networks and methods have been involved in natural processing. In a variety of networks, CNN [5] is widely used for its generalization. Because of its advantage in learning the higher level features, CNN has achieved a great breakthrough [6] and being applied to many areas, such as image classification, face recognition and other tasks related to image [7]. Among them, LSTM [8] is a kind of recurrent neural networks and is able to keep the information for a long time because it has memory cells. In other words, LSTM is a time-order network [9], while BLSTM (Bidirectional Long Short Term Memory) is a kind of network combining LSTM with BRNN [10],

which has two directions to input data and two hiding layers to save the information in both directions and sharing the same output layer. In this paper, we designed a more complex model containing both CNN and BLSTM for sentiment analysis.

The rest of the paper is organized as follows: Sect. 2 reviews the related work to CNN and LSTM. Section 3 describes the model we used in this work. Section 4 shows the experiments and their results. Section 5 presents the conclusion and future works.

2 Methods and Models

2.1 CNN

Usually, convolutional neural networks [5] are used in image classification. In this study, CNN was used as a feature important part in our sentiment analysis model.

Convolutional neural networks provide a learning model which is end to end and the parameters of the model can be trained with gradient descent algorithm. A typical convolutional neural network consists of an input layer, convolution layers, pooling layers, a full connection layer and an output layer. Its architecture is shown in Fig. 1.

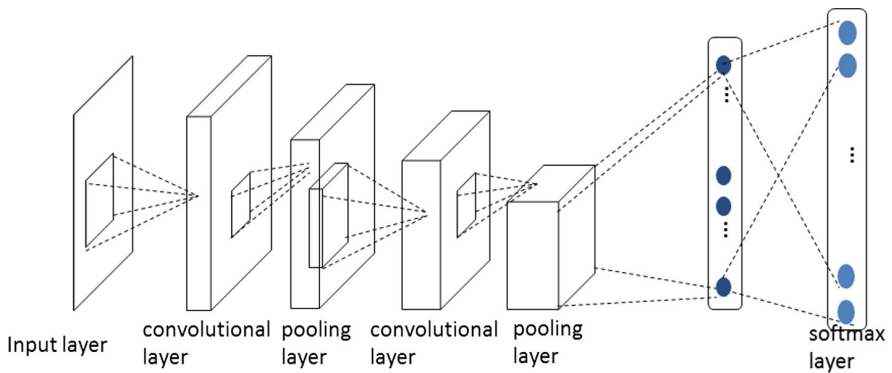


Fig. 1. Structure of typical CNN.

Both images and texts can be presented as a matrix. The matrix V is the input of a CNN. We suppose that H_i is the i layer of the network ($V = H_0$).

If H_i is a convolutional layer:

$$H_i = f(H_{i-1} \odot W_i + b_i) \quad (1)$$

In formula (1) W_i is the weight vector of H_i layer. Operator \odot is the convolution operation of the convolution kernel and the H_{i-1} layer. The output of the convolution will be added with the offset b_i . And finally, H_i could be calculated by a nonlinear excitation $f(x)$.

If H_i is a pooling layer:

$$H_i = \text{subsampling}(H_{i-1}) \tag{2}$$

The purpose of this layer is to reduce dimensionality and keep the features stable to some extent.

What we want is a probability distribution Y . With several convolution layers and pooling layers, we get a model:

$$Y(i) = P(L = l_i | H_0; (W, b)) \tag{3}$$

Gradient descent algorithm is used to train the parameters (W and b) of each layer.

$$E(W, b) = L(W, b) + \frac{\lambda}{2} W^T W \tag{4}$$

$$W_i = W_i - \eta \frac{\partial E(W, b)}{\partial W_i} \tag{5}$$

$$b_i = b_i - \eta \frac{\partial E(W, b)}{\partial b_i} \tag{6}$$

λ is the parameter to control the overfitting and η is the learning rate.

2.2 LSTM and BLSTM

LSTM [8] is a kind of recurrent neural networks (RNN). Because of its ability to make use of long term information, it is suitable for sequential data processing in texts. LSTM unit is shown in Fig. 2. And BLSTM [10] is included in LSTM, which is bidirectional.

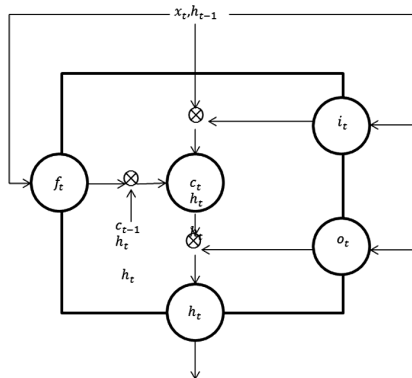


Fig. 2. LSTM memory block.

A LSTM memory block unit contains three gates, input gate, forget gate and output gate. These gates facilitate saving, reading, resetting and updating the long term information. We supposed x as the input, c as the memory cell and the h as the output. We calculate the candidate cell \tilde{c}_t , W_{xc} and W_{hc} are weights of input and the output from previous time.

$$H_i = f(H_{i-1} \odot W_i + b_i) \quad (7)$$

Input gate value i_t is used to control the impact of memory cell. Forget gate value f_t is used to control the impact from history information to current memory. And output gate value is used to control the output of the unit.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (8)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (9)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (10)$$

x_t is the current input value. h_{t-1} is the previous output value. And c_{t-1} is the previous memory cell. σ is an activation function that we usually choose logistic sigmoid algorithm for it.

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \quad (11)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (12)$$

3 CNN-BLSTM Model

3.1 Word Embedding

Word embedding is a significant technology in NLP when referring to deep learning. It turns each word into a feature embedding containing semantic information. In this study, word embedding was calculated with GloVe algorithm (Global Vector for word Representation) which is represented by Stanford in 2014 [11].

It's the loss function of GloVe:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^w f(P_{ij})(u_i^T v_j - \log P_{ij})^2 \quad (13)$$

In this formula, $f(x)$ is a truncation function to reduce the disturbance of frequently used words.

3.2 CNN-BLSTM Model

In this paper, our model combined one-layer CNN and BLSTM. Each word in the texts has been processed to be a 50-dimensional vector in the preprocessing of the model. The embedding processing is shown in Fig. 3.

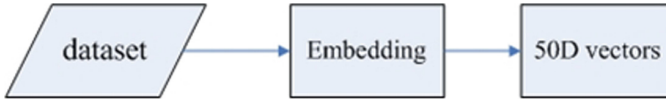


Fig. 3. Word embedding.

The pretrained words in the dataset were projected into word vectors in input part of the our model. A convolution layer, a maxpolling layer, a dropout layer and a BLSTM layer was followed sequentially. The whole structure is shown in Fig. 4.

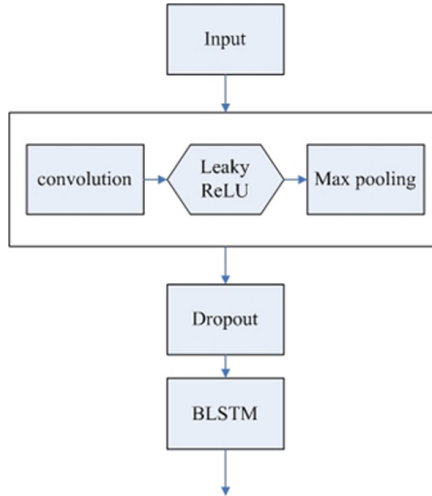


Fig. 4. CNN & BLSTM model structure.

The input is the result from word embedding process. The input layer receive a maximum of 1000 word vectors, each of them has 50 dimensions. We used convolution kernel size of 5 in CNN layer with activation function of LeakyRelU, a max pooling layer followed it with pool size of 4. In order to prevent model overfitting, we used a dropout layer after the last max pooling layer. The last part of the model is a BLSTM layer and two dense in which the last dense used softmax as activation and output 2 dimension vector refer to probabilities of the positive and negative classes.

4 Experiments and Results

4.1 Dataset

We used the dataset about the sentiment classifications of movie reviews in IMDB. The labeled dataset consists of 50000 IMDB movie reviews. They only have two types of reviews, positive or negative. The whole dataset was divided into two parts. 40000 labeled reviews were training set and the other 10000 were test set.

4.2 Experiments

4.2.1 Experiment 1

We compared simple BLSTM and CNN-BLSTM model for sentiment classifications. BLSTM model used a BLSTM layer with 128 unit outputs. CNN-BLSTM model connected a CNN layer with the size of 5-unit kernel, a max pooling layer with pool size 4 and an BLSTM layer with 128-unit output. The maximum feature numbers are 20000, maximum lengths are 1000 and the loss function is categorical cross entropy in the both models. The results are shown in Table 1.

Table 1. Comparison of BLSTM and CNN-LSTM.

	BLSTM	CNN-BLSTM
Accuracy	82.5%	85.3%

The accuracy of each single model is respectively 82.5% and 85.3%. CNN-BLSTM performed better than BLSTM model. The local semantic information in a sentence extracted by the convolution layer could lead to the better performance of CNN-BLSTM.

4.2.2 Experiment 2

We compared six different models. Three of them are models with respectively 1, 2 or 3 CNN layers with a BLSTM layer and a full connected layer with pre-trained word embedding (using GloVe, dimension: 50), and the others are models without pre-trained word embedding. In every model, each CNN layer used 64 filters with kernel size 5, each CNN layer was followed by a max-pooling layer with pool size 4, the last pool layer was followed by a dropout layer and a BLSTM layer with 128 unit outputs, the last layers were two full connected layers with respectively 128 unit outputs/activation function of LeakyReLU and 2-unit outputs/activation function of softmax. Every model was trained with 10 epochs, the training batch size is 128 with loss of categorical cross entropy and RMSProp optimizer. The comparison results are shown in Table 2.

The results showed that the 1CNN-BLSTM model with pretrained word embedding got the best accuracy 89.7%, and the 3CNN-BLSTM model with pretrained word embedding got the lowest accuracy 77.7%.

Table 2. Comparison of 6 models.

	1CNN-BLSTM	2CNN-BLSTM	3CNN-BLSTM	Average
With pretrained word embedding	89.7%	86.2%	77.7%	84.5%
Without pretrained word embedding	85.3%	81.6%	79.1%	82.0%
Accuracy	87.9%	83.9%	78.4%	

At first we could compare the models with pretrained word embedding and models without those, the result showed that average accuracy of the former is 84.5% while average accuracy of the latter is 82.0%. It's obviously that the pretrain word embedding enhanced the performance of CNN-BLSTM, which could be caused by the extra semantic information brought by word embedding.

We could also compared models with different numbers of CNN layer. It was kind of weird that the models with more CNN layer got worse performance. This consequence might be on the contrary of the cognition of us that the model with more complex structure should perform better. But it's known that IMDB datasets only have 50000 samples (40000 for training and 10000 for testing). The consequence could be the results of under-fitting of the complex model.

5 Conclusion and Future Work

In this work, we described a series of experiments with CNN and BLSTM and obtained a satisfactory accuracy that could reach up to 89.7%. The whole process contained two steps. The first step was the data pre-processing and each word was turned into a 50D word vector. The result was obviously better if we did word embedding before the data was put into our model directly.

The second step was to accomplish the sentiment analysis with CNN and BLSTM. In this step, we changed the numbers of layers of convolution and pooling in CNN to find a simple CNN with one layer of convolution and one layer of pooling which performed best by comparison. The accuracy of two-layer CNN which contained two convolution layers and two pooling layers was 86.2% and the accuracy of three-layer CNN which contained three convolution layers and three pooling layers was 77.7%.

In the model, the dropout function was also added to make the result reliable because of its reduction of over fitting. We also investigated whether the additional pre-processing of the data could give the result an advantage like adding POS (part of speech) tagging to the experiment, because not all of the words make sense. In general, adjectives and adverbs describe the feelings of the authors and therefore in the future work, we will devote to exploring more significant methods to do pre-processing of the natural language data.

Acknowledgment. This work was supported by the Grant from the National Natural Science Foundation of China (61173116).

References

1. Hatzivassiloglou, V., McKeown, K.R.: Predicting the semantic orientation of adjectives. In: Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics, pp. 174–181. Association for Computational Linguistics (1997)
2. Turney, P.D.: Thumbs up or down semantic orientation applied to unsupervised of reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computation Linguistics, pp. 417–424. ACL, Somerset (2002)
3. Kim, S.M., Hovy, E.: Automatic detection of opinion bearing words and sentences. In: Proceedings of the Companion Volume to the International Joint Conference on Natural Language Processing (IJCNLP), vol. 8 (2005)
4. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques, pp. 79–86 (2002)
5. Bouvrie, J.: Notes on convolutional neural networks (2006)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
7. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Wang, G.: Recent advances in convolutional neural networks. *arXiv Preprint [arXiv:1512.07108](https://arxiv.org/abs/1512.07108)* (2015)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
9. Sundermeyer, M., Ney, H., Schlüter, R.: From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Trans. Audio Speech Lang. Process. (TASLP)* **23**(3), 517–529 (2015)
10. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45**(11), 2673–2681 (1997)
11. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: *EMNLP 2014*, pp. 1532–1543 (2014)