Gao, J.F., Yao, R.X., Lai, H., Wu, H.T. (2019). Setiment analysis of tourism reviews: An exploratory study based on CNNs built on LSTM model.. In *Proceedings of The 19th International Conference on Electronic Business* (pp. 535-540). ICEB, Newcastle upon Tyne, UK, December 8-12.

Sentiment Analysis of Tourism Reviews: An exploratory study based on CNNs built on LSTM model

(Work in Progress)

Jinfeng Gao, Huanghuai University, Henan, China, gaojinfeng0213@126.com
Ruxian Yao, Huanghuai University, Henan, China, yaoruxian@126.com
Han Lai*, Huanghuai University, Henan, China, han.lai@hotmail.com
Haitao Wu, Huanghuai University, Henan, China, wht@huanghuai.edu.cn

ABSTRACT

This study is to develop a sentiment analysis system for customers' review on a scenic site. It is based on Convolutional Neural Networks (CNNs) built on Long Short-Term Memory (LSTM) models for text feature extraction under a deep learning framework. The CNNs built on LSTM models applies convolutional filters of CNNs repeatedly operate on the output matrix of LSTM to obtain robust text feature vector. In this study, the optimal parameter configurations for each component of CNNs and LSTM are given individually in the first place. Then, the entire optimal parameter configuration for the integration recognition frame of the system is identified around the optimum of each component. The results demonstrate that, by employing such a method, the accuracy for sentiment analysis with CNNs built on LSTM model, compared with a single CNNs or LSTM model, is improved by 3.13% and 1.71% respectively.

Key words: sentiment analysis, CNNs, LSTM, Classification

*Corresponding author

INTRODUCTION

Tourist's review on a scenic site contains marketing information, feedback to services and customer's sentiment, which is critical to improve the management of the scenic spot by means of designing more effective operating pattern. Sentiment analysis of tourist' review is an integration of natural language processing and machine learning. Different to English segmented on spaces, the Chinese text needs a tokenizer such as Jieba and ICTCLAS etc to convert it into a sequence of words.

The word embeddings that maps words or phrases to vectors of real numbers is widely used to capture meaningful syntactic and semantic regularities helps learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. As for representation of the text as fixed-length feature vector, the bag-of-words (BOW) and skip-gram are first employed to obtain acceptable accuracy (Mikolov, et al. 2013). By keeping the context information, deep learning techniques have significantly outperformed traditional methods. Two of the most popular deep learning techniques for sentiment analysis are Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) (Grefenstette, et al. 2013; Zhang, et al. 2017)

The CNNs only extract the local features, while LSTMs are a type of network that has a memory that remembers previous data from the input and makes decisions based on that knowledge. Therefore, LSTMs are more directly suited for text inputs, since each word in a sentence has meaning based on the surrounding words. Since, this research developed a sentiment analysis system for customers' comments based on LSTM-CNN model to extract text feature vector. In other words, the LSTM layer is generating a new encoding for the original input. The output of the LSTM layer is then fed into a convolution neural network layer which we expect will extract local features for sentiment classification.

SYSTEM DIAGRAM

To realize sentiment analysis of comments, the comments text is first preprocessed by Jieba to segment the comment string into corresponding word sequences. Then, each word of the tokenized comment text is mapped into *d*-dimensional word embedding by Word2Vector. Next, the text feature of comment is selected by LSTM, CNN or CNN built on LSTM. Last, the sentiment classifier based on neural network is used to realize the sentiment analysis of comment. It is necessary to address that the last two steps can work in deep learning frame. The left of figure 1 describes the dataflow of the system and the right gives out the corresponding modules.

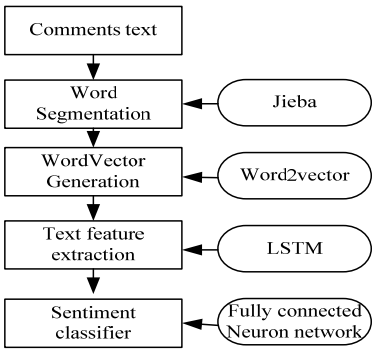

Figure 1: Dataflow of sentiment analysis system.

TEXT FEATURE EXTRACTION

Suppose there is a tokenized text sequence composed of n tokens (words), noted as $t(w_1, w_2, \cdots, w_n)$. Each token $w_i (1 \le i \le n)$ is represented by a *d*-dimensional word embedding trained by word2vec or GloVe models. Thus the text t is converted to a text matrix A with dimensionality $n \times d$. The columns of text matrix are tokens sequence, and the rows are word embedding of each token.

Based On LSTM

A layer with multiple LSTM units works as the feature extraction layer in the network, and the input of each unit corresponds to the tokens of sequence text (Tang, Duyu, et al. 2016). An LSTM unit is a memory cell composed of four main components: an input gate, a self-recurrent connection, a forget gate and an output gate. The forget gate determines how much previous context information h_{t-1} and current input information x_t are discarded via the value of factor f_t , 0 indicates overall discarded, 1 does overall remained, which is computed by formula (1). An input gate controls the effect to current memory cell i_t , according to the previous state h_{t-1} and the current state x_t , computed by formula (2). Then, generate candidate vector $\tilde{c}_t$, computed by formula (3), to update the current memory cell. The new state is computed by formula (4). The output gate factor computed by formula (5) allows how much information h_t to next memory cells, computed by formula (6). The output h_n of last LSTM unit is the obtained text feature vector with the same dimensionality with word embedding.

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

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

$$\tilde{C}_t = \tanh\left(w_c \cdot \left[h_{t-1}, x_n\right] + b_c\right) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_n \quad (4)$$

$$O_t = \sigma\left(w_O \cdot \left[h_{t-1}, x_n\right] + b_O\right) \quad (5)$$

$$h_t = O_t * \tanh(C_t) \quad (6)$$

Based On CNN

The text matrix is denoted by A with dimensionality $n \times d$, and $A[p:q]$ represents the sub-matrix of A from row *p* to *q*. Supposed there are a series of filter matrix $w_i (1 \le i \le m)$ with filter window $h_i \times d$. The output sequence $O_i (1 \le i \le n - h_i + 1)$ of the convolution operation is obtained by repeatedly applying the filter w_i on sub-matrices of text matrix A, computed by formula (7), where $k = 1, \cdots, n - h_i + 1$, and $\cdot$ is the dot product between the sub-matrix and the filter. A bias term b is added to each O_i , then activated by function f to obtain the feature map $C_i \left(1 \le \mathrm{i} \le n - h_i + 1\right)$ for filter w_i , computed by formula (8). The dimensionality of each feature map C_i will vary as a function of the text length and the filter window size. To obtain a fixed-length feature vector, k-max pooling function is applied to each feature map C_i to extract $k (k \ge 1)$ scalars. Then, all scalars extracted from each feature map are concatenated into a fixed-length text feature vector (Lakshmi, et al. 2017; Yoon Kim, 2004)

$$O_i = w_i \cdot \mathrm{A}\left[k : k + h_i - 1\right] \quad (7)$$

$$C_i = f\left(O_i + b\right) \tag{8}$$

Based On CNN Built On LSTM

In above Section A and Section B, the text feature vector is only extracted by LSTM or CNN separately. In this section, we resort to a CNN structure built on the outputs of LSTM to obtain a more robust text feature vector, as shown in figure 2. The structure of CNN and LSTM in the figure are similar to the one depicted in above two section. In the CNN built on LSTM approach to text feature extraction, the text matrix A first passes through the LSTM layer to obtain the corresponding output of LSTM $h\left(h_1, h_2, \cdots, h_n\right)$. Since each output of LSTM unit $h_i (1 \le i \le n)$ is same *d*-dimensional vector with each token, the output of LSTM can also be transferred into $n \times d$ matrix, noted as H. Similar with text feature extraction based on CNNs, the CNNs built on LSTM approach operates on the output matrix of LSTM H to obtain the text feature vector.

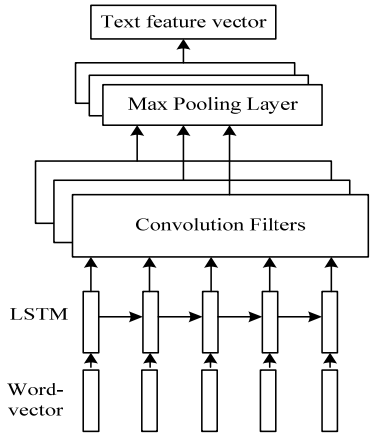

Figure 2: Text feature extraction based on CNN built on LSTM.

CORPUS AND DATABASE

To obtain the large enough corpuses for word embedding training, two steps were taken by the researcher: origin text extraction from the Wiki dump on 20th November, 2018 and conversion of extracted origin from traditional Chinese to simple Chinese to build the corpus database, noted as CCDatabase. This resulted in a larger database, which includes 14,327 Chinese character categories, in total 1.38GB.

SSC database, as tourist comments database that is collected by a research group of tourism big data under Inspur Company, is employed in this research. This database contains 1,000,000 comments labeled as three classes, positive, neuter and negative denoted as 1, 2 and 3 respectively. The length of comments is variable from zero to 536 Chinese characters, common in 35 to 155 Chinese characters. The average length of comments is about 76 Chinese characters.

EXPERIMENTS AND ANALYSIS

In this paper, all experiments are conducted under the base recognition frame, composed of Chinese word tokenizer, word embedding, text feature extraction and classifier four components. Moreover, to reduce over-fitting, the dropout layer is succeeded after text feature extraction layer. The dataflow of base recognition frame is depicted in section 2, and the configuration of each component is shown in table 1. To study the effect of different components with different parameter settings, we hold all other settings constant and vary only the interested component. For every configuration that we consider, we employ 5-fold cross validation and only report the mean of 5-fold cross validation.

Table 1: Overview of base recognition frame.

Description	Values
Word tokenizer	Jieba
Word embedding	Google Word2vec
Text feature extraction	LSTM, CNNs or LSTM+CNNs
Drop rate	0.5
Classifier	Two-layer Neural Network

Text Feature Extraction With LSTMs

The word embedding is semantic representation of input word in lower dimension. Thus, dimensionality (d) of word embedding is key factor for a nice property of text sentiment analysis. Moreover, the performance of LSTM is dependent on the size of context window. Therefore, we first explore the sensitivity of sentiment analysis system based on LSTM with different dimensions of word embedding and different sizes of word context window.

Word embedding in all applications, the dimensionality of word embedding with the range from 160 to 300 has been commonly used. In our system, we consider variations from 160 to 240 with the step 20. For each dimension, we take a different size of context window (k) from 20 to 100 with the step 20 so that we produce 25 different configuration settings. We measure the accuracy of the system for the 25 different parameter settings as shown in Table 2. From the table, the parameter setting d=200 and k=80 contributes the best recognition accuracy which is bold.

Table 2: The accuracies of the system with different d and k.

d\k	20	40	60	80	100
160	79.52	79.92	80.87	81.52	81.43
180	80.18	81.23	81.63	82.13	82.07
200	80.27	81.34	81.92	**82.85**	82.71
220	80.31	81.39	81.87	82.72	82.68
240	80.24	81.27	81.71	82.55	82.53

Text Feature Extraction With CNNs

To evaluate the effect of text feature extraction based on CNNs to text sentiment analysis, we explore the different parameters configurations of CNNs with the other components of base recognition frame constant. The discriminating performance of text feature extraction with CNNs is dependent on the number of the convolutional filters and *k*-max pooling schema.

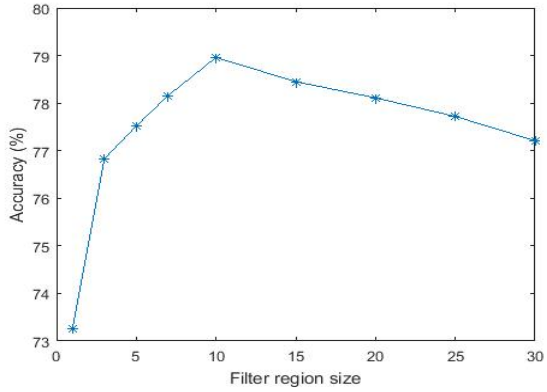

Figure 3: Accuracy trends with single different filter region sizes setting.

Since the dimensionality of word embedding with 200 obtains the best recognition accuracy in table 2, keep the dimensionality of word representation with 200 constant to focus on the comparison of other parameters setting. Once the dimensionality is specified, the size of filter region only relies on its height. We first explore the effect of single filter region size and set the number of feature maps for this region size to 100 (as in the baseline configuration). We consider filter region sizes of 1, 3, 5, 7, 10, 15, 20, 25 and 30, and report results as shown in figure 3.

We also explored the effect of combining different filter region sizes, while keeping the number of feature maps for each region size with 100. From the figure 3, we can observe that the optimal single region size is 13. In light of combining several region sizes close to the optimal single region size can improve performance, but adding region sizes far from the optimal value may hurt performance. Therefore, we explore the combination of several region sizes nearby this single best size 13, including combining both different region sizes (12, 13), (12, 13, 14), (12, 13, 14, 15) and (10, 11, 12, 13) and copies of the optimal sizes (13, 13), (13, 13, 13) and (13, 13, 13, 13). The results are reported in table 3.

Table 3: Accuracy of filter regions with several region sizes.

Multiple region size	Accuracy (%)
(12, 13)	82.23
(12, 13, 14)	**82.73**
(12, 13, 14, 15)	82.51
(11, 12, 13, 14, 15)	82.51
(10, 11, 12, 13)	82.45
(13, 13)	81.26
(13, 13, 13)	81.35
(13, 13, 13, 13)	81.34

As for the pooling strategy, we prefer *k*-max pooling strategy rather than average strategy. The *k*-max pooling strategy extracts top-level k scalars from each feature map to together generate text feature vector. We explored the *k* from 1 to 10 when the number of feature map is set to 100 and the single region size is set to 13. The results are shown in figure 4 and we found 1-max pooling fared best. The $k > 1$ does not show any obvious improvement in our early experiments.

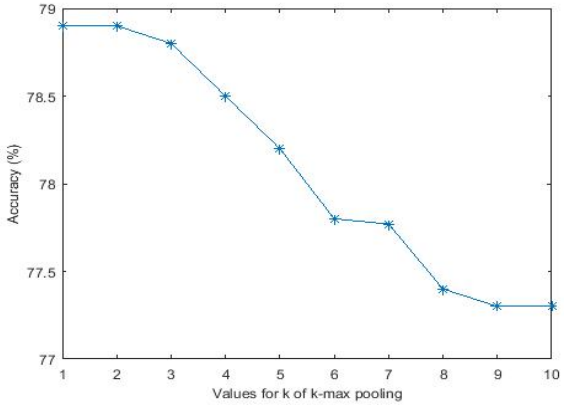

Figure 4: Accuracy trends with different *k* for *k*-max pooling.

The combining filter size (12, 13, 14) and 1-max pooling setting obtained optimal text recognition accuracy as shown in table 3 and figure 4, respectively. We again hold these optimal configurations constant within the base recognition frame, and change only the number of feature maps taking 10, 50, 100, 150, 200, 300, 400, 500, 600 and 700. The result is reported in figure 5.

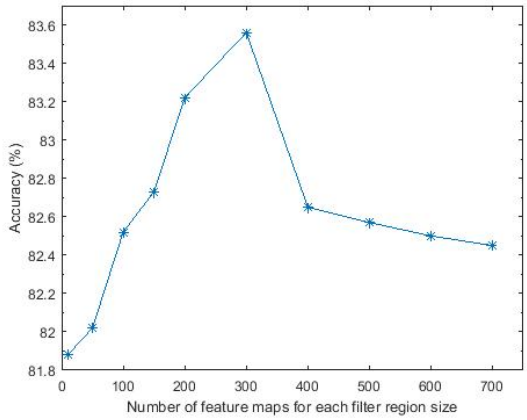

Figure 5: Accuracy trends with different number of feature map for each region size.

Text Feature Extraction With CNNs Built On LSTM

The section 4.B illustrated the parameters configuration with 1-max pooling, multiple filter size (12, 13, 14,) and the number of feature map setting to 300 for each filter region size achieves optimal result for sentiment analysis based on CNNs. So, we hold the optimal parameter configuration of CNNs invariable and search optimal configuration of LSTM parameters around the optimal parameter settings of separate LSTM, dimensionality of word embedding 200 and size of word context 80, as reported in section 4.A. We consider variation of word embedding dimension from 180 to 230 with step 10, and each dimension takes a group of context windows with size from 70 to 100 stepped by 5. The results are reported in Table 4.

Table 4: The accuracies of text sentiment analysis.

d/*m*	70	75	80	85	90	95	100
180	80.23	81.19	81.82	82.56	82.61	82.48	82.27
190	80.97	81.67	82.03	82.92	82.83	82.71	82.37
200	81.33	81.79	82.62	**83.11**	82.96	82.80	82.65
210	81.36	81.63	82.71	82.96	83.05	82.91	82.74
220	81.29	81.42	82.35	82.84	82.93	82.65	82.49
230	81.18	81.31	81.96	82.23	82.67	82.53	82.31

Similarly, we also keep the best parameters setting with dimensionality of word embedding 200 and size of word context window 85 illustrated in table 2 constant to find best parameter setting for number of feature map and size of filter region of CNNs in CNNs built on LSTM feature extraction approach with 1-max pooling. Based on above experimental results of separate CNNs for feature extraction in section 4.B, we first search the optimal filter region size around best multiple sizes (12, 13, 14) and best single size 13 with the number of feature map setting to 300 for each filter size. The results are reported in table 5.

We further set optimal filter region sizes as (11, 12, 13) as shown in table 5 to find optimal number of feature map for each filter size around 300. We consider the variation from 250 to 350 with step 20, then, find the best number of feature map configuration with 290 to obtain the best sentiment analysis accuracy 83.82%, as illustrated in figure 5.

Table 5: Accuracies of CNNs built on LSTM with different filter region size setting.

Single size	Accuracy (%)	Multiple region size	Accuracy (%)
10	80.42	(10, 11, 12)	83.18
11	80.56	(11, 12, 13)	**83.56**
12	**80.83**	(12,13, 14)	83.26
13	80.21	(10, 11, 12, 13)	83.12
14	80.18	(11, 12, 13, 14,)	83.08
15	79.88	(10, 11, 12, 13, 14)	82.89

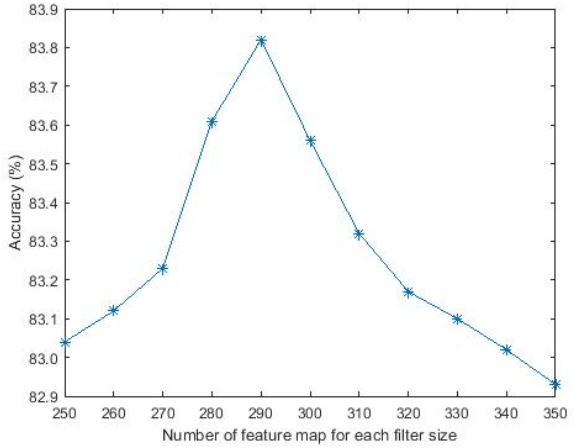

Figure 5: Accuracy trend with different number of feature map for each filter region size.

Experimental results demonstrate that the accuracy for sentiment analysis with CNNs built on LSTM model improved by 3.13% and 1.71% respectively, compared with separate CNNs and LSTM model.

ACKNOWLEDGMENTS

The authors would like to thank the research was supported by Henan Natural Science Foundation (No.1623004100195) and Henan Key Laboratory of Smart Lighting.

REFERENCES

Grefenstette, E., Dinu, G., Zhang, Y. Z., Sadrzadeh, M., & Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. arXiv preprint arXiv:1301.6939.

Lakshmi B S, Raj P S, Vikram R R, et al. (2017) Sentiment Analysis using Deep Learning Technique CNN with K-Means[J]. *International journal of pure and applied mathematics,114*(11 2),47-57

Mikolov, T., Yih, W. T., & Zweig, G. (2013, June). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 746-751).

Tang, D., Qin, B., Feng, X., & Liu, T. (2015). Effective LSTMs for target-dependent sentiment classification. arXiv preprint arXiv:1512.01100.

Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882..

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.