Channel Attention TextCNN with Feature Word Extraction for Weibo Sentiment Analysis

Journal:	*Transactions on Asian and Low-Resource Language Information Processing*
Manuscript ID	TALLIP-21-0561
Manuscript Type:	Full-length Paper
Date Submitted by the Author:	06-Dec-2021
Complete List of Authors:	Liu, Jiangwei; Anhui University, School of Computer Science and Technology Yan, Zian; Anhui University, School of Computer Science and Technology Chen, Sibao; Anhui University, School of Computer Science and Technology Sun, Xiao; Hefei University of Technology, School of Computer and Information Luo, Bin; Anhui University, School of Computer Science and Technology
Keywords:	Social platforms, Feature word extraction, Sentiment analysis, Attention

Channel Attention TextCNN with Feature Word Extraction for Weibo Sentiment Analysis

JIANGWEI LIU, Anhui University, China
ZIAN YAN, Anhui University, China
SIBAO CHEN*, Anhui University, China
XIAO SUN, Hefei University of Technology, China
BIN LUO, Anhui University, China

Sentiment analysis of Chinese Weibo can help understand society's views on various hot news. Many existing microblog sentiment analysis methods are based on sentiment dictionaries. Still, sentiment dictionaries are easily affected by subjective factors, require a lot of time to build, and need maintenance to prevent obsolescence. In order to extract the rich information in the text more effectively, the Channel Attention TextCNN with Feature Word Extraction model is proposed whose abbreviation is CAT-FWE. The Feature Word Extraction can choose vocabularies that affect the sentiment of reviews and integrate them with multi-level semantic information to make full use of the semantic information of the sentence. At the same time, the Channel Attention TextCNN pays more attention to those meaningful features than traditional TextCNN, and it can eliminate the impacts of features that don't make sense to a certain degree. The CAT-FWE model is applied on the task of fine-grained classification of text sentiment, and experiments show that this model can improve the effectiveness of the job.

CCS Concepts: • **Computing methodologies → Natural language processing**.

Additional Key Words and Phrases: Social platforms, Feature word extraction, Sentiment analysis, Attention

ACM Reference Format:
Jiangwei Liu, Zian Yan, Sibao Chen, Xiao Sun, and Bin Luo. 2021. Channel Attention TextCNN with Feature Word Extraction for Weibo Sentiment Analysis. 1, 1 (December 2021), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As one of the largest Chinese social platforms, people can use Weibo to record their daily mood, express their opinions, communicate with netizens worldwide, and participate in various themes. Netizens on Weibo can see posts and comments on multiple topics, including life, education, technology, military, politics, and entertainment, on this social platform. In addition, the hot search list allows netizens to know the latest major news happening all over the world in real-time. According to a report in September 2020, Weibo has 511 million monthly active users and 224 million daily active users. On such a large platform, research like sentiment analysis has become a hot topic in recent years [31],

*Corresponding author.

Authors' addresses: Jiangwei Liu, Anhui University, Hefei, Anhui Province, China; Zian Yan, Anhui University, Hefei, Anhui Province, China; Sibao Chen, Anhui University, Hefei, Anhui Province, China, sbchen@ahu.edu.cn; Xiao Sun, Hefei University of Technology, Hefei, Anhui Province, China; Bin Luo, Anhui University, Hefei, Anhui Province, China.

we can understand the mental health of contemporary people, their attitudes towards social hot spots, and even help management departments [12] to observe the trend of public social opinions, so as to make correct decisions in time. However, it is impossible to extract this information manually, and we can solve this problem by deep learning.

In existing sentiment classification research, a lot of work is based on sentiment dictionary and traditional feature engineering [29]. Although the sentiment dictionary can quickly process large amounts of data containing emotion in the sentence [6] and helps a lot in judging emotions, we should be aware that it is very time-consuming and labor-intensive to build an emotion dictionary [11]. Besides, the Internet is developing rapidly. Many emerging vocabularies that contain emotions change quickly. A practical, emotional dictionary needs to be maintained continuously. Also, people can observe that many Weibo texts do not explicitly incorporate emotional words, and even some sentences do not contain any emotional words. However, its implicit expression contains powerful emotional factors, so the emotional dictionary still has many limitations.

These years, a model named TextCNN [9] that has not been around for long works pretty well in text sentiment classification tasks. It can recognize words of different sizes in the text and take into account the semantic information of multi-level. Thus the opinions in the text are fully mined. However, it has a shortcoming that it treats all the characteristics the same. For example, compared to the simple classification of text emotions into positive and negative, fine-grained emotion classification has higher requirements for the model. If all features are treated equally, it will significantly reduce the effect of distinguishing similar emotional polarities like excitement and happiness with purely positive and negative thinking.

To solve these problems, we propose a model called Channel Attention TextCNN with Feature Word Extraction. This model firstly converts texts into word vectors with characters as a unit and keeps each sentence the same length, padding zeros to the short sentences. Secondly, use BiLSTM [5], one-dimensional CNN [15] and attention mechanism [18] to obtain the contextual, word-level, and sentence-level information of the text, then use extracted feature words to enhance the multi-layer semantics of the text and decode the semantic information with channel attention CNN. Lastly, classify the decoded information using several linear layers. In this paper, our model is used for categorizing sentiments of happiness, sadness, like, anger, disgust, fear, and surprise, and it achieves state-of-the-art results.

The innovations and contributions of this article are as follows:

- We propose a new method of feature word extraction for sentiment classification. It replaces sentiment dictionaries, and the model can extract sentiment words with sentiment polarity by itself. It does not require manual design and is not affected by the influence of emerging vocabularies. The selected words can effectively improve the accuracy of the model when added for enhancing the information expression.
- For the fine-grained sentiment classification, we proposed a model based on channel attention. To improve the model's sensitivity to emotions, we propose a weight-based mechanism that can assign a higher weight to the features of words that significantly impact sentiment judgment. As for features with less emotional polarity, their ratio is close to zero. However, some features have different effects on sentences in different situations, so we design the weights to be learnable.
- Our model provides a new idea for the design of sentiment analysis models. Residual network [2] is used in many networks, and it directly uses the output of the previous model as the input of the latter model. Based on this model, different extraction methods can be designed to obtain keywords or features to replace the original output of the previous module and input them into the next layer in the form of the residual network.

2 RELATED WORK

Text sentiment classification is a branch of sentiment computing. Its main task is to point out the sentiment tendency of the given text. In this era of the rapid rise of the Internet, major social platforms and e-commerce platforms continue to appear, and the Internet is full of textual information covering almost all aspects. Therefore, more and more researchers set their sights in this direction, and various methods have emerged one after another.

We know that the sentiment judgment can be derived from some keywords in the sentence, so many people consider using sentiment dictionaries to classify sentences by calculating the sentiment value of sentences. Sentiment dictionary refers to a collection of words in which each word containing emotion is labeled with a number indicating sentiment degree. The calculation methods are usually specific rules defined by the researchers themselves, and most of these rules are carried out based on the emotion dictionary. Combined with the advantages of CNN, GRU, the attention mechanism, and the emotional dictionary, Yang et al. [25] categorized the sentiment of comments on the e-commerce library website. Wu et al. [22] constructed several Chinese sentiment dictionaries, including original sentiment dictionary, adverb dictionary, conjunction dictionary, expression dictionary, negative word dictionary, and neologism dictionary. And together with Chinese semantic rules, they divided Weibo texts into three categories: positive, negative, and neutral. Some people like Zhang et al. [28] extracted key sentences through emotional attributes, location attributes, and keyword attributes and then comprehensively considered the dependencies and multiple rules in the sentence to design a set of emotional calculation methods. However, some field-specific words and polysemous words make conventional dictionaries not highly applicable. Therefore, Xu et al. [24] built a specific field dictionary as well as a polysemous dictionary, and Wu et al. [21] a dictionary aimed at college students. They are all used for sentiment computing in specific fields. Nevertheless, these words in the sentiment dictionary are collected manually, and people themselves generally set the sentiment label and calculation rules of each word. Thus the way of sentiment calculation is easily affected by subjective factors. Moreover, a set of rules in many cases only applicable to a specific field.

To eliminate human subjective factors, many usually choose machine learning methods, as the models can learn the hidden features in the samples by themselves. For example, Xia et al. [23] used conditional random field and support vector machine to perform sentiment analysis on online text. Al-Zamzami et al. [1] used the light gradient boosting machine to improve the ability to process high-dimensional and unbalanced samples to help text classification. Vijayaragavan et al. [19] used a support vector machine (SVM) to classify user comments and then used a confusion matrix with a clustering algorithm to analyze the possibility of users buying goods. Hew et al. [4] scored emotionally on the course evaluation on MOOCs using the gradient boosting tree model and the characteristics of various aspects of the course, which improves the ability to identify people's judgment on course satisfaction. There are also scholars like Yang et al. [26] who first classified the personality contained in the text into five categories and then used LSTM to train five personality-based emotion classifiers together with an ordinary emotion classifier. Finally, combined all the classifiers to classify the text.

Meanwhile, deep neural networks are also favored by many researchers with their powerful feature extraction capabilities. When designing natural language processing models, models that deal with time series like RNN, LSTM, and GRU have always been considered. Zhou et al. [32] used superimposed bidirectional LSTM to classify text. Rathnayaka et al. [14] proposed a pyramidal attention network structure based on RNN, which uses both low-level and high-level semantic information for sentiment detection of Weibo text. And since TextCNN was first proposed by Kim [9], CNN is often seen in text classification as well. Of course, with the continuous development of natural language processing technology, more and more novel models are also proposed and used in sentiment analysis, such as attention, transformer, graph attention network, and so on. Without adding too many parameters, Johnson and Zhang [8] used deep word-level CNN to achieve a higher classification accuracy. Lai et al. [10] used BiLSTM to extract context information, graph attention network to extract sentence grammatical information, and successfully classified the sentiment of short texts. Su et al. [16] designed a model that combines the multi-head attention mechanism of BiLSTM and CNN. In addition, the most popular today is, of course, the BERT model proposed in 2018. Its appearance has greatly improved many natural language processing tasks. Just like Palani et al. [13], a T-BERT model was proposed by combining the BERT model with the potential topics extracted from the text, which effectively improves the effect of sentiment classification. Tang and Nongpong [17] proposed a lightweight pre-trained BERT model for character-level transfer learning for sentiment classification tasks.

In summary, the research on sentiment analysis of texts is a direction that has attracted many people's attention. By absorbing the experience of previous work, we propose a CAT-FWE model, which has achieved an excellent effect on the fine-grained sentiment classification of microblogs.

3 PROPOSED METHOD

Weibo comments belong to the category of short texts, and they have a limitation in length. In this paper, each comment is sent into our CAT-FWE model after processing for sentiment classification. The overall framework of the CAT-FWE model could be seen in Figure 1. Our model consists of the embedding part, the multi-level semantic information encoding part, the multi-level semantic information decoding part, and the classifier part. In the following, we will introduce these four parts in detail.

3.1 Embedding Part

At the word embedding stage, many people regard a word as a basic unit, and they use word segmentation tools like JieBa to segment a sentence into several words. However, we couldn't ignore errors caused by such tools, and they may get bigger through backpropagation. For example, the sentence "南京市长江大桥" could be read as "南京市|长江大桥" (Nanjing Yangtze River Bridge) or "南京市长|江大桥" (Mayor of Nanjing Jiang Daqiao). Tools are not like people with daily life experience, the sentence above may be divided into the second one, which deviated from the sentence's meaning. This kind of error can mislead the sentiment classification of sentences to a vast extent.

So considering the situation above, we regard a character as a basic unit, assign a unique number to each character in the dataset. Thus each sentence could be represented by a series of numbers. For those whose length is less than the maximum sentence length in the dataset, pad them with zero. Then, the numbers corresponding to each character are converted into vectors of a specified dimension, which means the embedding representation of a sentence is a fixed-dimensional matrix $I \in \mathbb{R}^{d*l}$, where l denotes the length of each sentence and d denotes the dimension of word embedding. Lastly, the matrix is fed into the next part.

3.2 Multi-level Sementic Information Encoding Part

The primary purpose of this part is to get semantic information hidden in words. As we regard a character as a basic unit, they are just scattered characters without any semantic relation. So firstly, use a BiLSTM model to get low-level contextual information. For the next step, what is needed is to obtain word-level information within characters and the influence of other characters in sentences. Generally, a one-dimension CNN model is used for word-level features, and for sentence-level, we apply an attention model. In addition, it is also noticed that a part of words often determines the sentiment of a whole sentence. Hence we come up with a feature word extraction model capable of getting sentiment

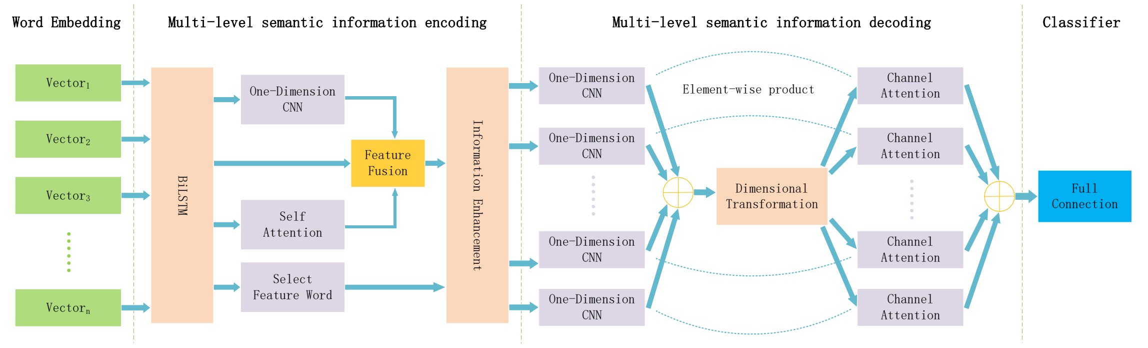

Fig. 1. Our proposed CAT-FWE model. The model consists of four parts in total. The first part is word embedding, which can convert characters into word vectors for model training. The second and third parts are the encoding and decoding of information in turn. The role of the last part is sentiment classification.

words for information enhancement. In the end, combine the contextual information, the word-level information, the sentence-level information together, and make use of the extracted feature words to enhance the multi-level information. The following are the implementation details of each module.

3.2.1 BiLSTM. Long Short-Term Memory [5] is a variant of Recurrent Neural Network. Its unique gate mechanism allows it to handle longer sequences and effectively prevent gradient explosion and gradient disappearance.

Outputs from the embedding part are just vectors of meaningless characters. To obtain mutual relations between them, we choose the LSTM module to get low-level semantic information. However, a single LSTM is only applied in one direction. For some sentences, we can only infer the emotional polarity with both forward and backward information. Therefore we consider the bidirectional LSTM instead. The forward LSTM is used to extract semantic information flowing from front to back, while backward LSTMs are just the opposite. With the input of $I \in \mathbb{R}^{d*l}$, we obtain the forward and backward information as follows:

$$\overrightarrow{I}_f^t = LSTM(I^t, \overrightarrow{I}_f^{t-1}), \tag{1}$$

$$\overleftarrow{I}_b^t = LSTM(I^t, \overleftarrow{I}_b^{t+1}). \tag{2}$$

The superscript t represents the time step, and the arrows indicate the forward and backward directions, respectively. The output dimensions in Equations (1) and (2) are half of the input vector, so by concatenating them on the characteristic dimension, we get the low-level contextual information:

$$I_{bls} = \overrightarrow{I}_f \oplus \overleftarrow{I}_b. \tag{3}$$

Symbol $\oplus$ denotes matrix connection. $I_{bls} \in \mathbb{R}^{d*l}$ is the same size as the input $I \in \mathbb{R}^{d*l}$.

3.2.2 One-dimension CNN. Consider some Chinese words like “发展中国家”, the character “中” can not only form the word “中国”(China) with the character “国”, but also “发展中”(developing) with the word “发展”. It reveals that the combination form of characters in Chinese has a deep impression on sentence comprehension. With a one-dimension CNN, one may get the word-level information.

Support $I_{bls} = [I_{bls}^1, I_{bls}^2, \ldots, I_{bls}^l]$, the dimension of each vector in I_{bls} is d. Firstly, we pad k d-dimension zero vectors at both sides of I_{bls}, and get a new matrix $I_{bls\_t} = [0, 0, \ldots, 0, I_{bls}^1, I_{bls}^2, \ldots, I_{bls}^l, 0, 0, \ldots, 0]$. This new matrix contains

$2k + l$ elements in total. Then, perform convolution operation with a window of size $2k + 1$. The specific calculations are as the following:

$$hc_i^t = W_c^i \otimes I_{bls}[:, t : t + 2k],$$
$$I_{cnn}^t = hc_1^t \oplus hc_2^t \oplus \cdots \oplus hc_r^t.$$

The symbol $\otimes$ denotes convolution operation, $W_c^i \in \mathbb{R}^{d*(2k+1)}$, i ranges from 1 to r, r is a hyperparameter that we can set, and t ranges from 1 to l, the $(t + 2k)_{th}$ element is also included in the calculation. The mathematical form of hc_i^t is actually a number that represents the convolution result of the i_{th} weight coefficient with vectors in t_{th} window. Repeat the operation above for r times, and integrate these numbers into a vector, then we obtain the convolution result at time step t. In fact, we set r to d directly in this paper. Next, we spliced all the vectors together:

$$I_{cnn} = I_{cnn}^1 \oplus I_{cnn}^2 \oplus \cdots \oplus I_{cnn}^l. \tag{4}$$

In this way, this module outputs a matrix $I_{cnn} \in \mathbb{R}^{d*l}$ of the same size as the input I_{bls} that contains the word-level information.

3.2.3 *Self Attention.*

For a sentence like “我想买苹果”, without other sub-sentences, whether the speaker wants to buy a mobile phone or fruit is not sure. Therefore it shows that sentence-level information is important for global understanding. Getting sentence-level information means finding the correlation between characters and other words , and self-attention is a good choice.

Refer to [18], each vector $I_{bls}^i \in \mathbb{R}^d$ in S is converted into vector Q_i, K_i, V_i by three learnable parameters W_q, W_k, W_v. Q_i is used to calculate the similarity with others' key K_j, and V_i represents the value of i_{th} character. The vectors are obtained by the following formulas:

$$Q_i = W_q * I_{bls}^i, \tag{5}$$
$$K_i = W_k * I_{bls}^i, \tag{6}$$
$$V_i = W_v * I_{bls}^i, \tag{7}$$

where $Q_i \in \mathbb{R}^h$, $K_i \in \mathbb{R}^h$, $V_i \in \mathbb{R}^h$, $W_q \in \mathbb{R}^{h*d}$, $W_k \in \mathbb{R}^{h*d}$, $W_v \in \mathbb{R}^{h*d}$, symbol * denotes matrix multiplication and h is a hyperparameter. Following Equations (5) through (7), we get three vectors towards each character, then the sentence-level information matrix is:

$$I_h = Softmax(\frac{(Q * K^T)}{\sqrt{d}}) * V.$$

$Q \in \mathbb{R}^{l*d}$, $K \in \mathbb{R}^{l*d}$, $V \in \mathbb{R}^{l*d}$ are the matrix concatenated by all the vectors in the sentence. Superscript T represents the transpose of the matrix. The result of $Q * K^T$ is a similarity matrix of characters. And for the reason of stabilizing the gradient, we divide the similarity matrix by $\sqrt{d}$. Then utilize a softmax function to normalize the similarity matrix, multiply it with the value matrix. Repeat the above process for m times, we receive a combination in m representation spaces:

$$I_{rep} = I_h^1 \oplus I_h^2 \oplus \ldots I_h^m.$$

Finally, project the combination to a fully connected layer and yields the final output:

$$I_{att} = W_a * I_{rep} + b_a. \tag{8}$$

$W_a \in \mathbb{R}^{d*mh}$, $b_a \in \mathbb{R}^{mh}$ are learnable parameters, and $I_{rep} \in \mathbb{R}^{l*mh}$, $I_{att} \in \mathbb{R}^{d*l}$.

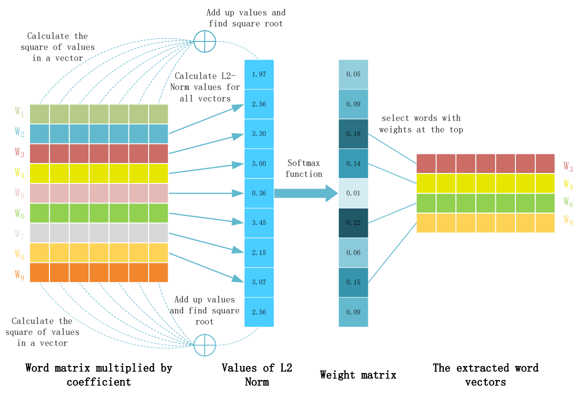
Add up values and find square root
Calculate the square of values in a vector
Calculate L2-Norm values for all vectors
W_1
W_2
W_3
W_4
W_5
W_6
W_7
W_8
W_9
1.97
2.56
3.30
3.00
0.36
3.45
2.15
3.07
2.56
Softmax function
0.05
0.09
0.19
0.14
0.01
0.22
0.06
0.15
0.09
select words with weights at the top
W_3
W_4
W_6
W_8
Add up values and find square root
Calculate the square of values in a vector
Word matrix multiplied by coefficient
Values of L2 Norm
Weight matrix
The extracted word vectors

Fig. 2. The FWE model. Firstly, make an element-wise multiplication on the data matrix and the weight matrix, and then calculate the L2 norm of the resulting matrix. According to the value of the L2 norm, some representative vectors are selected. As for the L2 norm vector in the figure, the color being darker means the value of the L2 norm is larger, which means that the characters represented by the vectors are more influential in the sentence.

3.2.4 Feature Word Extraction. A relatively long sentence always contains many meaningless words. Take "一句简单的回复我的话语，会让我开心一整天"(A simple reply to me will make me happy all day) as an example; it's hard to speculate on the speaker's inner thoughts with the previous sentence, his/her mood is directly shown in the last sentence, especially the word "开心"(happy). So we need to selectly choose characters that have a deep impact on sentence sentiment judgment.

For convenience, people always lengthen relatively short sentences to the same length as long sentences for training. Normally, a conventional way of it is padding short sentences with zeros. It clears that zero is equivalent to meaninglessness because it's thought that zero will not affect the judgment of the sentence at all. Following this thought, the aim to extract some characters that control sentence emotion can be converted to selecting some vectors far from the origin in the coordinate axis. Meanwhile, we use the L2 norm as the quantitative indicator. The specific process is shown in Figure 2.

Given the matrix I_{bls}, we make an element-wise multiplication on it with a coefficient $W_{fea} \in \mathbb{R}^{l*d}$, and get a feature matrix $I_{fea} = I_{bls} \odot W_{fea}$. Based on I_{fea}=$[I^1_{fea}, I^2_{fea}, \ldots, I^l_{fea}]$, we follow the steps below to calculate the weight:

$$V^t = function(I^t_{fea}),$$

$$V = V^1 \oplus V^2 \oplus \cdots \oplus V^l,$$

$$WM = softmax(V).$$

The function is used for compute the value of L2 norm. Then we take out e vectors corresponding to the top values in WM from I_{fea}, and make up the feature word matrix $I_{fwe} \in \mathbb{R}^{e*l}$.

3.2.5 Information Fusion. The diversity of sentences makes it impossible to obtain the expression of the entire sentence with only one kind of information. To improve the model's generalization ability, the next thing is to fuse as much information as possible. Then we can get one that contains information on all levels.

In previous sections, we get the low-level contextual information I_{bls} in Equation (3), the word-level information I_{cnn} in Equation (4), and the sentence-level information I_{att} in Equation (8). It can be concluded from the foregoing that the shape of the three matrices is as large as the input, so we add them up:

$$I_{fusion} = I_{bls} + I_{cnn} + I_{att}.$$

In this way, we get a matrix containing multi-level semantic information. Next, use the extracted feature words I_{fwe} to enhance the fused semantic information with a weighted method:

$$I_{similar} = I_{fusion} * I_{fwe}^{T},$$

$$I_{weight} = Softmax(I_{similar}),$$

$$I_{encode} = I_{fusion} + I_{weight} * I_{fwe}^{T}.$$

At this point, we receive the final output of the encoding part I_{encode}.

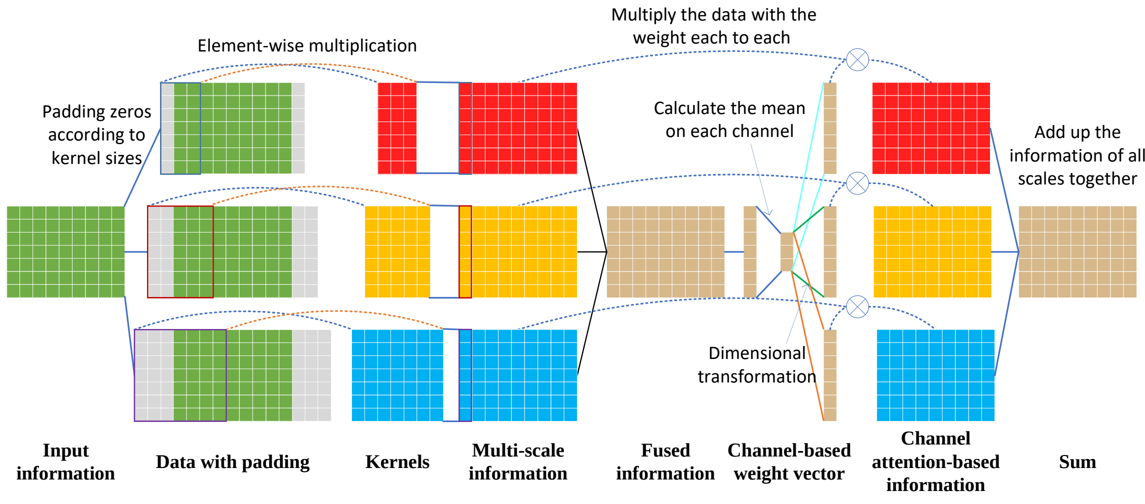
Element-wise multiplication
Multiply the data with the weight each to each
Padding zeros according to kernel sizes
Calculate the mean on each channel
Add up the information of all scales together
Dimensional transformation
Input information
Data with padding
Kernels
Multi-scale information
Fused information
Channel-based weight vector
Channel attention-based information
Sum

Fig. 3. The channel attention TextCNN model. Firstly, the semantic information at different scales is obtained through convolution operation, then calculated the average values of the channels of their sum to obtain the weights of each channel, and applied the weights to each corresponding semantic information. Finally, add up all the weighted information to get the output of the model. We use full connection layers instead of max-pooling layers for dimensional transformation as data may be lost with the pooling function.

3.3 Multi-level Sementic Information Decoding Part

After obtaining the semantically encoded representation of the sentence, the next thing is to decode the encoded output. For sentence information, the classic model one can think of is always RNN, a model that processes sequence information or its variants LSTM and GRU. But when TextCNN was first proposed in [9], the experimental results in [30] showed the effectiveness of CNN in processing text.

The idea of TextCNN is to pass the text through one-dimensional CNNs with different kernel sizes. Then, to unify the size, a pooling network is used to select a maximum value for the output of each CNN. Thus the output of each CNN is a leading feature of the same size. Next, all the outcomes are spliced together and sent to the fully connected layer for classification. For a word vector with multiple features, not every channel dimension feature has the same effect on the word. Some features have a greater degree of influence. So we proposed a model named Channel Attention-based TextCNN. In Figure 3, you can see the specific implementation of the model.

The main difference between TextCNN and our model is the attention mechanism. As well, for the output I_{encode} of the last part, in the beginning, we feed it into several CNNs with different sizes, fill zeros to keep the size unchanged like in Section 3.2.2 and get $[O_1, O_2, \ldots, O_z]$, z is the number of CNNs. To some extent, this can take most of the words with various lengths into account. Then in the case of all feature weights being the same, and to prevent information from being lost, instead of picking out the most representative feature by max pooling, we add outputs of CNNs together:

$$A = O_1 + O_2 + \cdots + O_z.$$

For $A \in \mathbb{R}^{d*l}$, in the channel dimension, calculate the mean $M \in \mathbb{R}^d$ for all the characters on every channel. However, some channels influence each other, so following by $M \in \mathbb{R}^d$, a dimension reduction linear layer is used to acquire the relationship between and a vector $H \in \mathbb{R}^c$ is obtained, here c is smaller than d. Next, each CNN output uses an ascending linear layer to change their dimensions back to d and uses the softmax function to convert them into weights. Then all the weights form the vector $U \in \mathbb{R}^{zd}$ and multiply these weights with the vector channels in a one-to-one correspondence. Then add z weighted matrices up, we get the final output $I_{decode} \in \mathbb{R}^{d*l}$.

3.4 Classifier Part

The last module of the whole model is to classify the previous information. Because what is used for classification is a matrix, and the desired result is a seven-dimensional vector corresponding to the seven-class task. Therefore, the job of the classification layer is to convert a matrix of $d * l$ dimensions into a low-dimensional vector using linear layers, then use the softmax output to label the sentiment polarity of the entire sentence. If you want to use semantic information fully, you can consider adding a few more linear layers and nonlinear activation functions. Still, in the end, our goal is a one-dimensional vector $I_{classify} \in \mathbb{R}^t$, t is equal to 7.

In this paper, we use the cross-entropy loss function L_{cross_entroy} for classification. Also, to prevent overfitting and improve the performance of the model, we use the L2 norm of the model parameters as the regularization loss $L_{regular}$. The regularization loss takes into account the parameters that convert word numbers into word vectors in the word embedding stage. Because integrating scattered characters into meaningful words is also a continuous learning process. Finally, add up the two losses together and train the model:

$$L = L_{cross\_entroy} + L_{regular}.$$

4 EXPERIMENT

In this section, we first introduce the dataset used in the experiments and several scientific indicators. Then we divide the experiments into several parts: “Comparision with Other Models”, “Impact of Extracted Feature Words” and “Impact of Channel Attention TextCNN” respectively. Through experimental comparisons and data analysis, we discussed the role of each module in the model described in the chapter above, which fully proved the significance and validity of the theoretical design of our model.

4.1 Dataset

The dataset used in experiments is proposed in [10], which contains 28781 training samples and 2160 test samples. Labels of this dataset include seven categories: happiness, sadness, like, anger, disgust, fear, and surprise. Table 1 shows the detailed information of each category. According to the statistics in the table, the categories of the dataset used in the experiment are unbalanced. For example, for the category like alone, our training set has more than 7,000 samples,

Table 1. Data distribution of our experiments.

Categories of sentiment	Count	
	Train	Test
Happiness	4371	370
Sadness	3234	380
Like	7160	587
Anger	3395	238
Disgust	4621	426
Fear	3000	47
Surprise	3000	112
Total	28781	2160

accounting for almost a quarter of all training samples. However, there are only 3000 samples for categories surprise and fear with the minimum quantity, and either of them have reached half the number of category like. For such a situation, if we input this dataset into our model, and the evaluation measures of the model's output are high, then the results can undoubtedly prove the superiority of the model.

4.2 Evaluation Measures

In this paper, the problem we solve is to determine the emotional polarity of a given text. In other words, what we are solving is a classification task, and it is a multi-classification task. Generally, evaluation measures for classification tasks include accuracy, precision, recall, and F1-score [3].

In binary classification tasks, precision, recall, and F1-score are usually calculated based on positive samples. The accuracy is the ratio of correctly identified samples to all samples. In some cases, F1-score is a more scientific indicator than accuracy, as it considers more comprehensively. For fine-grained classification tasks, nevertheless, all categories are regarded as positive samples. Therefore, the $micro_{precision}$, $micro_{recall}$, and $micro_{F1-score}$ are all the same as the accuracy. They focus more on categories with more data. While for the micro-averaged performance scores, treat each category as a binary classification task, and take their average values as the final results. These measures are sensitive to small categories, as small categories vary greatly, which can verify the performance of the model.

In consideration of our unbalanced dataset, we should not only pay attention to the model's prediction accuracy but also the effect on categories with fewer data. Thus, when comparing with other models, both the $micro_{F1-score}$ and the $macro_{F1-score}$ of the model are considered. Besides, in module analysis parts, the accuracy, recall, and F1-score are all macro-averaged performance scores, because the accuracy is equal to their micro-averaged performance scores.

4.3 Comparison with Other Models

We list the experimental results of several classification algorithms for comparison to prove the superiority of our model's results compared to other models. The experimental results are shown in Table 2. From the definition, $micro_{F1-score}$ focuses more on samples with a larger sample size, while $macro_{F1-score}$ is more sensitive to samples with small quantities. It can be seen that our model has achieved the best testing results on both $macro_{F1-score}$ and $micro_{F1-score}$. This shows that our model has a stronger ability to deal with sample imbalances and can extract useful semantic information when the amount of data is relatively small.

Table 2. Results compared with other models. The result with superscript 1 is cited from [20]. Those with 2,3 are models proposed in [7, 27] respectively. And the results with superscript 4 are obtained from [10]. The results of our model are shown in the last row.

Model	$Macro_{F1-score}(\%)$	$Micro_{F1-score}(\%)$
Best of NLP&CC2013[1]	31.29	35.21
Lexion[1]	30.77	33.27
SVM vote[1]	23.62	31.11
CSRs[1]	42.07	44.19
E-EMS[2]	35.00	43.90
MCNN[2]	34.62	43.65
EMCNN[3]	35.17	44.22
CNN[4]	67.42	74.23
LSTM[4]	68.32	73.42
LSTM+CNN[4]	69.89	76.42
LSTM+GCN(without syntax information)[4]	74.81	77.99
LSTM+GCN(with syntax information)[4]	79.93	82.32
CAT-FWE(Ours)	80.78	82.82

Table 3. Results under different TextCNNs and dimensions. "-" denotes we don't use TextCNN. (1,2,3,4) means that we use four CNNs with the padding size of 3, 5, 7, 9, as 1 is the padding size, and the rest are the same. And woA/wiA means we use TextCNN without/with attention.

Dimensions	woA/wiA	CNN type	Precision(%)	Recall(%)	F1-score(%)	Accuracy(%)
960	-	-	78.09	76.16	77.05	81.06
	woA	(1,1,1,1)	78.46	76.49	77.38	81.30
		(1,2,3,4)	78.93	75.83	77.19	81.06
	wiA	(1,1,1,1)	76.93	74.07	75.29	78.98
		(1,2,3,4)	78.73	74.81	76.52	79.83
1280	-	-	80.09	76.44	78.02	81.30
	woA	(1,1,1,1)	79.27	76.74	77.89	81.16
		(1,2,3,4)	77.00	76.79	76.74	79.78
	wiA	(1,1,1,1)	75.93	74.15	74.80	78.98
		(1,2,3,4)	79.60	75.41	77.20	80.87
1600	-	-	81.45	76.66	78.72	82.01
	woA	(1,1,1,1)	79.80	78.55	79.11	80.92
		(1,2,3,4)	79.22	77.58	78.34	80.97
	wiA	(1,1,1,1)	78.86	77.61	78.18	81.72
		(1,2,3,4)	80.20	76.71	78.16	80.87
1920	-	-	79.12	76.36	77.59	81.44
	woA	(1,1,1,1)	80.41	78.83	79.41	81.34
		(1,2,3,4)	80.02	77.58	78.68	81.25
	wiA	(1,1,1,1)	80.12	75.25	77.29	81.53
		(1,2,3,4)	81.64	80.08	80.78	82.81

4.4 Impact of Extracted Feature Words

The number of keywords extracted from BiLSTM is a hyperparameter we can set. To find out how many feature words are more suitable, by setting the number of different extracted feature words, we list the results in Table 4.

Table 4. The effect of the extracted feature word vectors on the performance of the model. Numbers in the first column indicate how many feature word vectors we extract. It means that we do not extract word vectors to enhance the fused information and directly output it to the subsequent modules when the number is 0.

Number of Words	Precision(%)	Recall(%)	F1-score(%)	Accuracy(%)
0	78.13	77.12	77.58	79.83
5	80.25	78.60	79.27	81.63
10	81.64	80.08	80.78	82.81
15	79.23	74.56	76.42	81.11
20	79.04	74.20	76.25	80.02
25	79.82	76.95	78.22	81.87
30	82.11	75.95	78.49	82.01

Considering the F1-score in the table, when the fusion features of the model are directly sent to the following modules, its performance is relatively poor. But if we extract several characters from the sentence, the effect of the model has an upward trend, followed by a decline first and then arise. The reason is that when the feature words are added, the model can obtain more semantic information of the sentence and get a richer expression. Since the training data sets are short texts, the extracted characters will contain irrelevant words with the extracted words increasing slightly. They greatly influence non-noise words, thereby affecting the model's performance; therefore, the F and accuracy rate decrease instead of increase. While the number of extracted feature words continues to rise, they can already approximate the expression of the short text, so there will be a situation where the performance continues to increase.

4.5 Impact of Channel Attention TextCNN

The information cannot be fully utilized if the semantic information is directly sent to the classification layer after being fused. So we proposed the channel attention TextCNN. But TextCNN also has many factors that can affect the model's accuracy, so we did a lot of experiments to explore. Table 3 lists our data.

4.5.1 Vector Dimensions. A character is fully represented after input into a model for training when converting it into a vector. If the dimensionality of the vector is low, then the characteristics of the corresponding character may be represented by limited numbers, and the information is likely to be represented insufficiently. With the increment of dimensionality, some characteristic numbers may be useless, and the computational cost will also increase. In other words, the dimension of the vector has a great influence on the model.

For the dimension of the vector, we think of 320 as the basic unit. When the dimensionality of the vector rises, the overall experimental results get better. When TextCNN is not used, the F index and accuracy representing the model's performance positively correlate with dimensionality, but it peaks at 1600 dimensions. That is because the characters' features are expressed fully, but when the dimensionality is too high, the limited learning ability is not enough to make full use of these characteristics, which causes the loss of information and leads to poorer performance.

While using TextCNN, you can see from the table that the effect is inferior to that of not using TextCNN when the dimensionality is low. It is just the opposite of what we discussed above, which is caused by the phenomenon of overfitting. As the model's learning ability is too strong, and each dimension of the vector represents more feature information, the model learns unique features of the vector itself, which leads to the decline of the generalization ability of the model, and thus the performance of the model descends as well. When the dimensionality rises to a certain level, we can see that the effect of the model with TextCNN is better than without it.

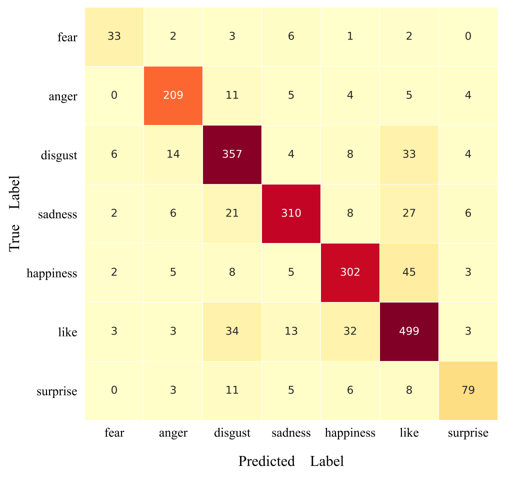

Fig. 4. The confusion matrix of classification results. The ordinate represents the true label of the sample, and the abscissa represents the predicted label. The value in the square box indicates the number of samples.

In addition, through comparison, that whether TextCNN has attention and whether the model has TextCNN has a similar situation. In the case of low dimensions, each dimension contains more information. All information is essential to the characters. Therefore, if attention is taken into account, it will increase the complexity of the model, lead to overfitting, and may reduce the weight of some information, resulting in the loss of characteristics. So in a low dimensionality, we can see that the accuracy is higher when there is no attention. However, it can be found that adding attention can help the model and improve the effect of sentiment classification if dimensionality is appropriate.

4.5.2 CNN Types. As we all know, CNN is used to extract features. The core idea of TextCNN is to use one-dimensional CNN to extract the semantics of text at different scales. That is to say, the type of one-dimensional CNN has a critical influence on the kind and amount of semantic information in a text that models can learn.

In this experiment, we set kernel sizes of four one-dimensional CNNs to two kinds, the first is all 3, and the other is 3, 5, 7, and 9 respectively. The former can only extract the basic semantics, and the range it can extract is limited, but in the case of multiple same sizes, they may fully utilize the semantics of the same character. However, when the kernel sizes become different, the range of information considered for the same character is more extensive, yet the model cannot fully learn the information in the same range.

In Table 3, through comparison, it can be found that regardless of the dimensionality of vectors, when the TextCNN has no attention, the effect of multiple CNNs with the same core size is better than that with different ones. However, after adding attention, the results of CNNs with different cores are much better. Since if the kernel sizes are the same, each CNN has learned semantics in the same range. In the feature dimension, the information gained from CNNs is complementary. When they are fused, we can not ignore the characteristics of all the channels. Some information may be lost due to being given a lower weight due to the attention mechanism. And when acquiring semantics in different sizes, attention can help us extract the important information output by CNNs and reduce the weight of useless information because each output represents different levels of semantic information. There is no complementary relationship between them.

4.6 Confusion Matrix

Finally, let us observe the performance of the model in a visual way. The confusion matrix in Figure 4 contains the classification results output by the trained model on the test set. It can be seen from the figure that the model correctly predicts most of the samples in all categories, which confirms that our model has a great processing capability for sample imbalance.

Next, let us analyze each category in detail. For the samples labeled as fear , samples labeled as sad account for the majority of misclassifications, and many disgusting emotions are mistakenly classified as fear. This is easy to understand because these emotions express people's repulsive psychology, and they all represent negative emotional states. It also indicates that although the number of samples is small, the model has acquired the main characteristics of fear emotion. Pay attention to emotion surprise, the number of samples is also small, but the model does not distinguish this emotion very well. Unique features belong to it are not fully extracted, as the effects of misclassification on other emotions are similar. While the classification effect of emotion anger is excellent because the recall and accuracy are high. The main emotion that anger is mistakenly classified into is disgust, and those samples mistakenly classified into anger are also labeled with disgust. Both of anger and disgust indicate a state of dissatisfaction. Therefore, this happens for granted.

For emotion disgust, sad, happiness, and like, they account for the largest number of samples. The other three emotions are mainly misclassified into like, which is caused by the imbalance of the samples because the training samples of emotion like has the largest proportion. From the figure, we can also find that emotion like is affected by these three categories. However, the quantity of disgust and sad which are classified as like is smaller than that of emotion happiness. Combine the common sense of happiness and like both indicating a positive attitude, we can come to the conclusion that the leading features are extracted. However, the sample imbalance affects so larger that the classification accuracy rate is not very high. The second main misclassified emotion of disgust is anger, which consists of the previous statement. Emotion sadness is divided into disgust, I think it may be due to the cause that the model has extracted the emotional characteristics of depression.

It can be seen from the above analysis that despite the relatively large impact of sample imbalance, the recognition rate of the model is still high. This shows the model's multi-level semantic information coding module has fully learned the emotion contained in sentences. And after analyzing the misclassified data distribution, we can find many key emotional factors are extracted by the model, which is mainly attributed to the channel attention TextCNN. It largely suppresses the expression of useless features and greatly helps the classification effect of the model.

5 CONCLUSION

This paper proposes a CAT-FWE model based on channel attention TextCNN and feature word extraction, and we use it in fine-grained sentiment classification of Chinese microblog short text. Through learning, the channel-based attention module assigns different weights to the feature values on different channels. Meaningful features can get greater weights, while features with less meaning are close to zero. So there are more meaningful features flowing back to help classification tasks. Keyword extraction can help us extract words with emotional polarity, reduce the trouble of artificially constructing emotional dictionaries, and at the same time improve applicability. In general, the entire model effectively helps us remove a lot of invalid information. Our experiments show that the above two modules we proposed are practical. In future work, we can design more effective keyword extraction methods by considering word-level or sentence-level information to improve the ability to acquire semantic information.

ACKNOWLEDGMENTS

The authors would like to thank the editor and anonymous reviewers for their valuable comments and suggestions, which are very helpful in improving this paper. This work was supported in part by NSFC Key Project of International (Regional) Cooperation and Exchanges under Grant 61860206004, and in part by the National Natural Science Foundation of China under Grant 61976004.

REFERENCES

[1] Fatimah Al-Zamzami, Mohamad Hoda, and Abdulmotaleb El-Saddik. 2020. Light Gradient Boosting Machine for General Sentiment Classification on Short Texts: A Comparative Evaluation. *IEEE Access* 8 (2020), 101840–101858.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* (2015). arXiv:1512.03385

[3] Fatemeh Hemmatian and Mohammad Karim Sohrabi. 2019. A survey on classification techniques for opinion mining and sentiment analysis. *Artif. Intell. Rev.* 52, 3 (2019), 1495–1545.

[4] Khe Foon Hew, Xiang Hu, Chen Qiao, and Ying Tang. 2020. What predicts student satisfaction with MOOCs: A gradient boosting trees supervised machine learning and sentiment analysis approach. *Comput. Educ.* 145 (2020).

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[6] Vandana Jha, Savitha R, P. Deepa Shenoy, K. R. Venugopal, and Arun Kumar Sangaiah. 2018. A novel sentiment aware dictionary for multi-domain sentiment classification. *Comput. Electr. Eng.* 69 (2018), 585–597.

[7] Fei Jiang, Yiqun Liu, Huan-Bo Luan, Jiashen Sun, Xuan Zhu, Min Zhang, and Shaoping Ma. 2015. Microblog Sentiment Analysis with Emoticon Space Model. *J. Comput. Sci. Technol.* 30, 5 (2015), 1120–1129.

[8] Rie Johnson and Tong Zhang. 2017. Deep Pyramid Convolutional Neural Networks for Text Categorization, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 562–570.

[9] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1746–1751.

[10] Yuni Lai, Linfeng Zhang, Donghong Han, Rui Zhou, and Guoren Wang. 2020. Fine-grained emotion classification of Chinese microblogs based on graph convolution networks. *World Wide Web* 23, 5 (2020), 2771–2787.

[11] Saif M. Mohammad and Tony Yang. 2013. Tracking Sentiment in Mail: How Genders Differ on Emotional Axes. *CoRR* abs/1309.6347 (2013).

[12] Vadim S. Moshkin, Andrey V. Konstantinov, and Nadezhda G. Yarushkina. 2020. Application of the BERT Language Model for Sentiment Analysis of Social Network Posts *(Lecture Notes in Computer Science, Vol. 12412)*, Sergei O. Kuznetsov, Aleksandr I. Panov, and Konstantin S. Yakovlev (Eds.). Springer, 274–283.

[13] Sarojadevi Palani, Prabhu Rajagopal, and Sidharth Pancholi. 2021. T-BERT - Model for Sentiment Analysis of Micro-blogs Integrating Topic Model and BERT. *CoRR* abs/2106.01097 (2021).

[14] Prabod Rathnayaka, Supun Abeysinghe, Chamod Samarajeewa, Isura Manchanayake, Malaka J. Walpola, Rashmika Nawaratne, Tharindu R. Bandaragoda, and Damminda Alahakoon. 2019. Gated Recurrent Neural Network Approach for Multilabel Emotion Detection in Microblogs. *CoRR* abs/1907.07653 (2019).

[15] Dario Stojanovski, Gjorgji Strezoski, Gjorgji Madjarov, Ivica Dimitrovski, and Ivan Chorbev. 2018. Deep neural network architecture for sentiment analysis and emotion identification of Twitter messages. *Multim. Tools Appl.* 77, 24 (2018), 32213–32242.

[16] Yi-Jen Su, Wu-Chih Hu, Ji-Han Jiang, and Ruei-Ye Su. 2020. A novel LMAEB-CNN model for Chinese microblog sentiment analysis. *J. Supercomput.* 76, 11 (2020), 9127–9141.

[17] Fuhong Tang and Kwankamol Nongpong. 2021. Chinese Sentiment Analysis Based on Lightweight Character-Level BERT. IEEE, 27–32.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.

[19] P. Vijayaragavan, R. Ponnusamy, and M. Aramudhan. 2020. An optimal support vector machine based classification model for sentimental analysis of online product reviews. *Future Gener. Comput. Syst.* 111 (2020), 234–240.

[20] Shiyang Wen and Xiaojun Wan. 2014. Emotion Classification in Microblog Texts Using Class Sequential Rules, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 187–193.

[21] Di Wu, Jianpei Zhang, and Jing Yang. 2020. Sentiment Lexicon for Chinese College Students to Build and Apply. In *CSAE 2020: The 4th International Conference on Computer Science and Application Engineering, Sanya, China, October 20-22, 2020*, Ali Emrouznejad and Jui-Sheng Rayson Chou (Eds.). ACM, 106:1–106:7.

[22] Jiesheng Wu, Kui Lu, Shuzhi Su, and Shi-Bing Wang. 2019. Chinese Micro-Blog Sentiment Analysis Based on Multiple Sentiment Dictionaries and Semantic Rule Sets. *IEEE Access* 7 (2019), 183924–183939.

[23] Huosong Xia, Yitai Yang, Xiaoting Pan, Zuopeng Justin Zhang, and Wuyue An. 2020. Sentiment analysis for online reviews using conditional random fields and support vector machines. *Electron. Commer. Res.* 20, 2 (2020), 343–360.

[24] Guixian Xu, Ziheng Yu, Haishen Yao, Fan Li, Yueting Meng, and Xu Wu. 2019. Chinese Text Sentiment Analysis Based on Extended Sentiment Dictionary. *IEEE Access* 7 (2019), 43749–43762.

[25] Li Yang, Ying Li, Jin Wang, and R. Simon Sherratt. 2020. Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning. *IEEE Access* 8 (2020), 23522–23530.

[26] Wenzhong Yang, Tingting Yuan, and Liejun Wang. 2020. Micro-Blog Sentiment Classification Method Based on the Personality and Bagging Algorithm. *Future Internet* 12, 4 (2020), 75.

[27] Peng Ye, Jayant Kumar, Le Kang, and David S. Doermann. 2012. Unsupervised feature learning framework for no-reference image quality assessment. IEEE Computer Society, 1098–1105.

[28] Shunxiang Zhang, Zhaoya Hu, Guangli Zhu, Ming Jin, and Kuan-Ching Li. 2021. Sentiment classification model for Chinese micro-blog comments based on key sentences extraction. *Soft Comput.* 25, 1 (2021), 463–476.

[29] Shunxiang Zhang, Zhongliang Wei, Yin Wang, and Tao Liao. 2018. Sentiment analysis of Chinese micro-blog text based on extended sentiment dictionary. *Future Gener. Comput. Syst.* 81 (2018), 395–403.

[30] Ye Zhang and Byron C. Wallace. 2017. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification, Greg Kondrak and Taro Watanabe (Eds.). Asian Federation of Natural Language Processing, 253–263.

[31] Jun Zhao, Kang Liu, and Liheng Xu. 2016. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. *Comput. Linguistics* 42, 3 (2016), 595–598.

[32] Junhao Zhou, Yue Lu, Hong-Ning Dai, Hao Wang, and Hong Xiao. 2019. Sentiment Analysis of Chinese Microblog Based on Stacked Bidirectional LSTM. *IEEE Access* 7 (2019), 38856–38866.