This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942614, IEEE Access

Sentiment Analysis of Text Based on Bidirectional LSTM with Multi-head Attention

Fei Long[1, 2], Kai Zhou[2], Weihua Ou[3, *]
1 School of Electrical Engineering, Guizhou Institute of Technology, Guiyang, China
feilong@git.edu.cn
2 College of Big Date and Information Engineering, Guizhou University, Guiyang, China
1328245669@qq.com
3 College of Big Date and Computer Science, Guizhou Normal University, Guiyang, China
ouweihuahust@gmail.com

***Abstract*—**Recent years, many scientists address the research on text sentiment analysis of social media due to the exponential growth of social multimedia content. Natural language ambiguities and indirect sentiments within the social media text have made it hard to classify by using traditional machine learning approaches, such as support vector machines, naive Bayes, hybrid models and so on. This article aims to investigate the sentiment analysis of social media Chinese text by combining Bidirectional Long-Short Term Memory (BiLSTM) networks with a Multi-head Attention (MHAT) mechanism in order to overcome the deficiency of Sentiment Analysis that is performed with traditional machine learning. BiLSTM networks, not only solve the long-term dependency problem, but they also capture the actual context of the text. Due to the fact that the MHAT mechanism can learn the relevant information from a different representation subspace by using multiple distributed calculations, the purpose is to add influence weights to the constructed text sequence. The results of the numerical experiments show that the proposed model achieves better performance than the existing well-established methods.

Keywords—Chinese product reviews text; Bidirectional LSTM; Multi-Head Attention mechanism

I. INTRODUCTION

The sentiment Analysis[1], which is called to be the crown of the artificial intelligence field[2], aims to use corresponding technical means to tap the potential emotional attitude (such as criticism, praise, and so on) that is contained in text. With the rapid growth of information volume, it is impossible to accomplish this task only through manual processing. The fact promotes the development of sentiment analysis technology towards a more intelligent direction. Machine Learning (ML)[3], by virtue of its excellent modeling ability, shines brilliantly in natural language processing (NLP) tasks. The most common ones are Support Vector Machines[4], Naive Bayes and hybrid models. Although ML is relatively mature, its inherent flaws have led to some thorny issues. Firstly, the process of building sentiment dictionary is time-consuming and labor-intensive, and some prior knowledge is required. Secondly, their effectiveness relies heavily on the application of feature selection strategies and parameter tuning. Based on the shortcomings mentioned above, the employment of these methods has been greatly reduced.

Presently, more and more Deep Learning (DL) techniques have been borrowed into NLP tasks[5]. Bengio et al.[6] proposed a language constructing model by using Artificial Neural Networks (ANN) that perform mapping of word to a low-dimensional space; *Word2vec* is an open source tool for Google to represent words as real numerical vectors in 2013 [7][8]. Its purpose is to process text content as vector operations in vector space, and similarity in vector space can be used to express semantic similarity of text. Jeffery Pennington et al. [9] proposed the *Glove* in 2014, which combines global statistical information and local context information to train word vectors, by adding statistical word vectors and parameter adjustments. kim et al.[10] achieves the considerable results in sentence classification with simple Convolutional Neural Networks (CNN). Long Short-Term Memory (LSTM) stores historical information by constructing a memory unit, each temporal state saves the previous input information, which can effectively alleviate the long-distance dependence problem of Recurrent Neural Networks (RNN) [11][12]. However, LSTM ignores future information. The BiLSTM contributes to the solution of obtaining both historical information and future information by using the bidirectional propagation mechanism, which helps to achieve better performance in SEA tasks[13][14].

In the human visual attention mechanism, humans do not observe everything from beginning to end, but its main objective is to select the most critical pieces of information from a large volume of data. DL draws on the attention mechanism of human beings in order to put more weight values on the key information that are more valuable to the judgment of sentiment. The first application of the attention mechanism was in the machine translation based on *seq2seq* in NLP. Calculation of the probability distribution of attention in the *seq2seq* model, it can highlight the impact of key inputs on output [15][16]. In 2017, the Google machine translation team proposed a new concept of MHAT [17], which is a refinement of the traditional one. Its advantage is to capture relevant

information in different subspaces through multiple distributed calculations.

The main contribution of this paper is to make the multi-head attention mechanism not only limited to machine translation tasks, but to the field of text sentiment analysis. In addition, considering that text sentiment analysis is fundamentally a sequence problem, this paper combines a multi-head attention mechanism with the BiLSTM into a hybrid model for text sentiment analysis [18].

II. RELATED WORK

A. word segmentation

Words are usually the smallest, meaningful basic element used to represent text. However, compared with English, Chinese language has the following natural disadvantage in writing habits. There is no clear indicative marker between “word” and “word”. The first step of sentiment analysis of a Chinese text is to approach problem of word segmentation. According to language habits,the word segmentation is the process of splitting a long sentence into meaningful words.

The *Jieba* is a statistical-based word segmentation algorithm and is considered to the one of the best Chinese word segmentation methods. It supports three models, in which this paper uses an accurate model suitable for text analysis. For example, a sentence "我来到北京看天安门", after the wording is "我/来/到/北京/看/天安门/". At the same time as the word segmentation, we cleaned the data by loading the stop word table to remove the special symbols. The purpose of the *Jieba* word segmentation is to divide the sentence into words one by one, which lays a good foundation for the further establishment of the dictionary.

B. Word embedding

In NLP tasks, we first need to consider how words are represented in the computer. In general, there are two ways of representing: discrete representations and distributed representations. The typical discrete representation is *one-hot* encoding, which is a traditional rule-based and statistical-based natural semantics processing method. The *one-hot* encoding regards words as a symbol and it expresses each word as a long vector whose dimension is equal to the size of the vocabulary. The word vector obtained by *one-hot* encoding is equal to 1 only at the position where the word is in the dictionary, and the remaining values are equal to 0. For example, one-hot encoding of the word ‘apple’ is expressed as [0, 0, 0, 1, 0, 0, 0, 0, …,0]. The word vector obtained by *one-hot* encoding is binary (only 0 and 1 elements), sparse (most elements are 0), and high-dimensional (the size of the dictionary). In short, the word vector represented by *one-hot* encoding is only a single individual and it fail to establish the semantic relationship between “word” and “word”. In addition, the dimension disasters also increase the burden of time and space for computational processing. In order to avoid the above defects, this paper uses a distributed representation - word embedding. In contrast to *one-hot* encoding, word embedding learns from corpus data to obtain continuous dense vectors. In addition, word embedding can calculate the similarity relationships between “word” and “word”. To put it plainly, the geometric relationship between word vectors corresponds to the semantic relationship between them. The geometric distance between “word” and “word” is related to the semantic distance. Methods for calculating the similarity between “word” and “word” are the Euclidean distance, the cos angle, and so on. In the following paragraphs, we will introduce the three word embedding algorithms used in this paper.

1) Embedding layer learning

It involves word embedding learning. The Keras framework, contains the Embedding Layer (EML), which makes it easier to map words into vectors. The EML takes an integer as input and it returns the associated dense vector by looking up the internal dictionary. Its initial weight is randomly generated. The back propagation (BP) algorithm is used to gradually adjust these word vectors just like in the training process of ANN. The spatial structure is changed so that the next algorithmic model can be utilized. Once the training is completed, an optimal word vector matrix is generated.

2) Word2vec

The advantage of *word2vec* is that it can train large-scale corpora to produce low-dimensional word vectors. *Word2vec* contains SG (Skip-gram) and CBOW (Continuous Bag of Word). The basic idea of CBOW is to predict the target words by a given context word. The model structure of the CBOW is shown in Fig.1. The CBOW is a simple three-layer neural network. Assuming the window size is equal to 2, the context words can be represented as (w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}). The implementation steps of the CBOW are as follows. First, a weighted average of (w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}) is determined. Then, the hidden layer state is derived through the projection relationship of the input layer, and the final output w_t is the predicted target word. The word vector is a by-product of CBOW model, but it is exactly what we need.

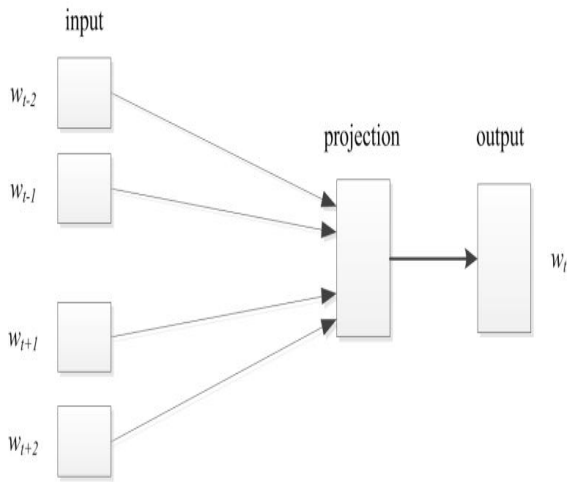

Fig. 1. The structure of the CBOW model

Considering that the coverage of words largely affects the quality of the word vector, the *Chinese Wikipedia corpus* is selected as the training data set, the process of training can capture the similarity between "word" and "word". For example, if we input the word "足球" then obtained output comprises of five words closest to "足球"heir similarities are the following:{ 足 球 员 : 0.5393211841583252, 足 球 队:0.5392793416976929, 橄榄球: 0.5193971395492554, 篮球: 0.5180296897888184, 俱乐部: 0.4991834759712219}

3) Glove

According to the developers of *Glove*, *word2vec* is only a model related to a local context window, and it seldom uses some statistical information in corpus. So, Jeffrey Pennington et al. introduced the *Glove*[8], which combines both global statistical information and local context information. Its main characteristic is that it introduces the co-occurrence probability matrix. Lest consider that the co-occurrence matrix is denoted by X, and the dictionary dimension by N. X_{ij} is the number of times that the words of order i and j appear together in one window in the entire corpus. At the same time, we use X_i to represent the total number of occurrences of all words (of the window) together with the word of order i (the center word). Also, P_{ij} is used to indicate the probability that the word of order j appears in the window of word i.

$$X_i = \sum_k X_{ik} \tag{1}$$

$$P_{ij} = p(j|i) = \frac{X_{ij}}{X_i} \tag{2}$$

Through some examples, we can find that the ratio of P_{ik} / P_{jk} depends on the words of order i, j, k. By fitting the relationship with the unknown function F, the form of the general model is as follows:

$$F(w_i, w_j, \widetilde{w}_k) = \frac{P_{ik}}{P_{ij}} \tag{3}$$

Where $w \in \mathrm{R}^d$ represents the word vector, $\widetilde{w} \in \mathrm{R}^d$ represents the context word vector. After a series of derivations, see [8] for the specific process, we get the cost function J,

$$J = \sum_{ik} f(X_{ij})(w_i^T \widetilde{w}_j + b_i + b_j - \log X_{ij})^2 \tag{4}$$

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha}, & if \ x < x_{max} \\ 1, & otherwise \end{cases} \tag{5}$$

Where $f(x)$ is a weight function and the empirical value of α is 3/4.

The *Glove* model uses *Chinese Wikipedia corpus* training to obtain word vectors. We did the following experiments to evaluate the effect of the word vector. The word "法国" and the word "巴黎" are related, the word "意大利" and the word "罗马" are related, the cosine similarity of the calculation ("法国" - "巴黎") and ("意大利" - 罗马") is 0.6751479308174202.

the word vector trained by the EML, the *word2vec*, and the *Glove,* with a dimension equal to 100. The advantage of doing this is to avoid the effects of different dimensions and only compare the pros and cons of the algorithm.

C. Strategies for Reducing Over-fitting

The well-designed algorithm model, which performs well on the training set, is not satisfactory in the test set. We call this phenomenon on over-fitting. How to reduce over-fitting is critical to optimizing the model. In the following paragraphs, we will introduce the three methods used in this paper.

1) dropout

The dropout operation randomly selects some neurons in the network layer with the dropout parameter and discards their input and output features (set to 0)[19]. The ρ refers to the proportion of features set to zero. If ρ=0.5 is set, the return vector of a layer of output samples is [1.2, 0.8, 0.2, 0.9, 1.9, 2.0]. After dropout, the vector becomes [0, 0.8, 0, 0.9, 0, 2.0.]. The reason why the dropout can reduce over-fitting is that it can capture more randomness. The core idea is to introduce noise into the output value of the network layer, breaking the inconspicuous accidental mode. If there is no noise, the learning mode of the network becomes single, will remember these casual modes.

2) L_2 regularization

The cost function of adding the L_2 regularization term can be expressed as:

$$Costfuncti\ on = Loss + \frac{\lambda}{2m}\|w\|^2 \tag{6}$$

After deriving the above formula, it is found that the trend of w is reduced, so the regularization of L_2 is also called weight attenuation. Since the weight is reduced, the complexity of the model is limited, which makes the distribution of weight values more regular. We know that in neural networks, simple models are less prone to over-fitting than complex models. Therefore, forcing the model weights to take only smaller values can effectively reduce the over-fitting.

3) Early stopping

The early stopping method is a cross-validation strategy. The principle is that when the performance of the model on the verification set is no longer good, we stop training. In the Keras architecture, we call the *callback* function to terminate the training. The callbacks record the output values for each iteration, including the verification loss value, the verification accuracy rate, the training loss value, and the training accuracy rate. There are two important parameters in callbacks: "monitor" and "patience", where "monitor" is used to supervise changes in the output value.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942614, IEEE Access

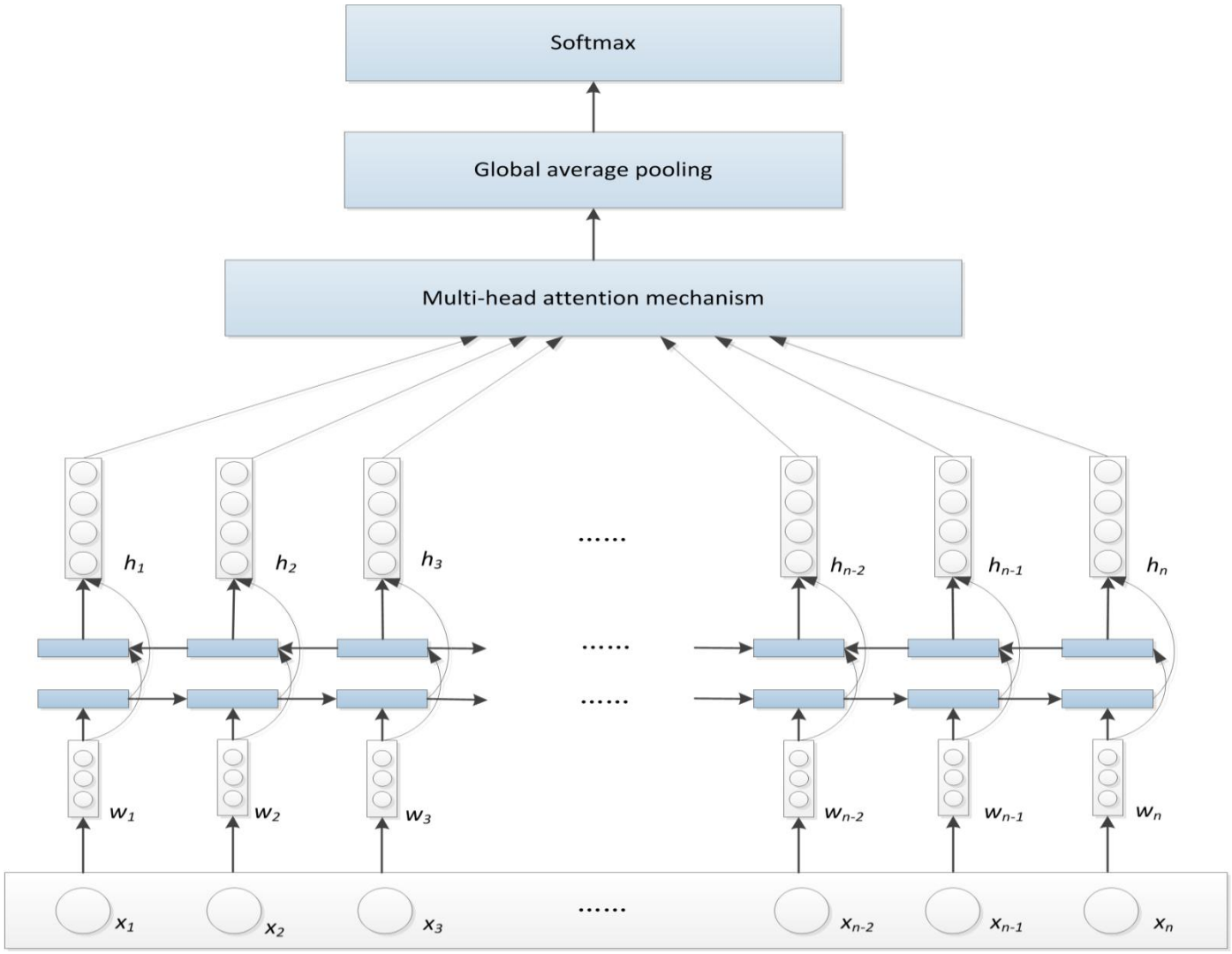

Fig. 2. The structure of Bidirectional LSTM with Multi-head Attention mechanism

In this experiment, setting "monitor" equals the verification accuracy. If "patience" is equal to 10, indicating that the value of the accuracy rate does not increase during 10 epochs, the model will automatically stop training. Because after that, the model is over-fitting.

III. BIDIRECTIONAL-LSTM BASED ON MULTI-HEAD ATTECTION MECHANISM

A hybrid model based on bidirectional LSTM combined with multi-head attention mechanism is built for text sentiment analysis tasks. The diagram of the structure is shown in the following Fig.3, the proposed model structure is mainly divided in four parts.

- Get the word embedding of the text;
- Enter the word vector into the BiLSTM to obtain the text representation feature;
- Introduce Multi-head Attention mechanism to capture relevant information on different subspaces and highlight the importance of different features;
- Use *softmax* layer to acquire emotional classification;

A. Word vector representation

Words are located at the lowest end of the model structure, where a subtext X consisting of n words is represented as (x_1, x_2, x_3, …, x_{n-2}, x_{n-1}, x_n), the input layer is represented as:

$$S = [w_1, w_2, w_3, \cdots, w_{n-2}, w_{n-1}, w_n] \in R^{n \times d} \quad (7)$$

B. Bidirectional LSTM

LSTM is consistent with the typical RNN framework, but it uses different ways to calculate the hidden state, which can solve the problem that Recurrent ANN cannot deal with long-distance dependence. However, the excellent capabilities of LSTM are not learned through algorithms, but the inherent advantages of its model structure. The LSTM model consists of a series of repeated memory units, each of which contains three gates with different function. The structure of such a unit is shown in the following Fig.2. Using the text feature vector S as input, and the t^{th} word as an example, the values of the

respective states of the LSTM unit of the t^{th} word are as follows:The specific calculation formula is as follows, where σ represents sigmoid function, ⊙ denotes dot multiplication.

The f_t is the forget gate:

$$f_t = \sigma(W_f w_t + U_f h_{t-1} + b_f) \tag{8}$$

The it is the input gate:

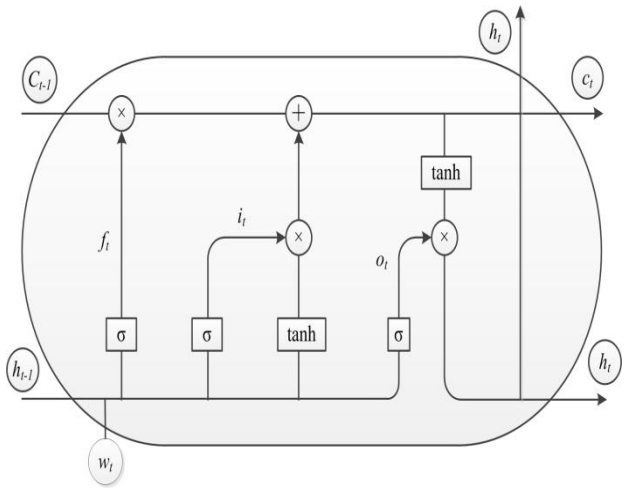

Fig. 3. The structure of the LSTM unit

$$i_t = \sigma(W_i w_t + U_i h_{t-1} + b_i) \tag{9}$$

The $\widetilde{c}_t$ stands for the candidate memory cell status at the current time step, where tanh is the tangent hyperbolic function;

$$\tilde{c}_t = tanh\,(W_c w_t + U_i h_{t-1} + b_c) \tag{10}$$

The c_t represents the state value of the current time in memory cell: the value of f_t and i_t range from 0 to 1. The calculation of $i_t \odot \tilde{C}_t$ indicating which new information is stored in c_t from the candidate unit $\widetilde{c}_t$. The calculation of $f_t \odot c_{t-1}$ indicating which information is retained and which is discarded in the previous memories c_{t-1}.

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{11}$$

The o_t is output gate:

$$o_t = \sigma(W_o w_t + U_o h_{t-1} + b_o) \tag{12}$$

h_t is the hidden layer state at time t:

$$h_t = o_t \odot tanh(c_t) \tag{13}$$

The Long-Short Term Memory only considers the historical information of the sequence, but it is often not enough. If you could access future information as you would access previous one, this is would be very beneficial for the sequence of tasks. The bidirectional LSTM consists of a forward Long-Short Term Memory layer and a backward Long-Short Term Memory layer. The principle of operation is as follows: the forward layer captures the historical information of the sequence; the backward layer captures the future information of the sequence. Both layers are connected to the same output layer. The biggest highlight of this structure is that the sequence context information is fully considered. Let's consider that the input of time t is the word embedding w_t, at time t -1, the output of the forward hidden unit is $\vec{h}_{t-1}$, and the output of the backward hidden unit is $\overleftarrow{h}_{t+1}$, Then the output of the backward and of the hidden unit at time t is equal as follow [20]:

$$\vec{h}_t = L(w_t, \vec{h}_{t-1}, c_{t-1}) \tag{14}$$

$$\overleftarrow{h}_t = L(w_t, \overleftarrow{h}_{t+1}, c_{t+1}) \tag{15}$$

where $L(\cdot)$ denotes the hidden layer operation of the LSTM hidden layer. The forward output vector is $\vec{h}_t \in R^{1\times H}$ and the backward output vector is $\overleftarrow{h}_t \in R^{1\times H}$, and they should be combined to obtain the text feature. It should be clarified that H denotes the number of hidden layer cells:

$$H_t = \vec{h}_t \left\| \overleftarrow{h}_t \right. \tag{16}$$

C. Multi-head Attention Mechanism

The basic component of the MHAT mechanism is the attention, but the difference is that the MHAT model performs multiple distributed calculations, which has the ability to process complex information.

1) Scaled dot-product attention

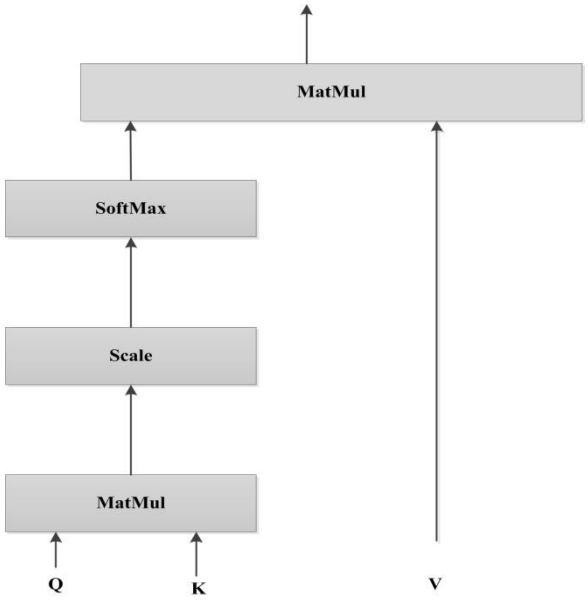

Fig. 4. The flowchart of scaled dot-product attention

A scaled dot-product attention can be described as mapping a query and a set of key-value pairs to an output. The process of calculating the attention, is mainly divided into the following four steps [20]:

- The weights of the query and each key are calculated by considering similarity. The similarity function used in this paper is the dot product;
- The next step is the scale operation, where the factor $\sqrt{d_k}$ plays a moderating role, so that the dot-product will not be too large;
- The obtained weights are Normalized by using a *softmax* function;
- The similarity and the corresponding key value V are equal to the weighted sum.

Based on the above steps, we can get the following formula:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{17}$$

2) Multi-head Attention

MHAT is the optimization of the traditional attention mechanism and it is also superior in performance. The structure of the MHAT mechanism is shown in Fig.4. Firstly, Q, K, V are input to scaled dot-product attention, through a linear transformation. This process calculates one head at a time. Note that there are h times to do here, which is actually called multi-head, but the parameters W for each linear transformation of Q, K, V are different. Then, all the scaled dot-product attention results of m times, are concatenated and the value obtained by a linear transformation is used as the result of the MHAT[21].

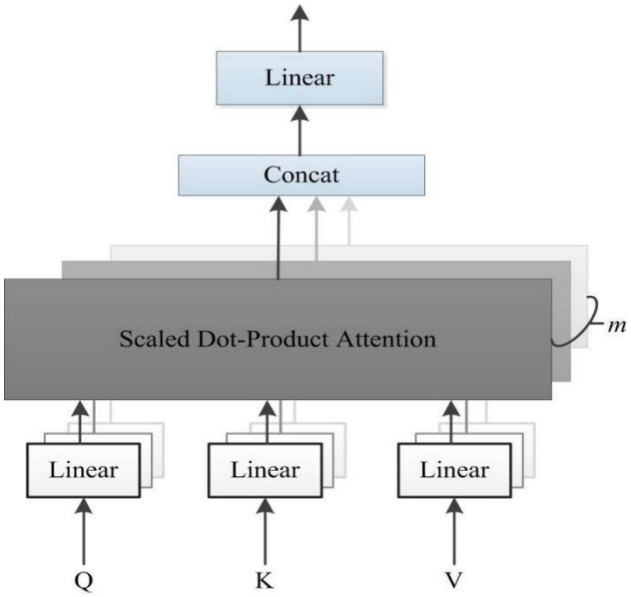

Fig. 5. The structure of the Multi-head Attention

The formula is expressed as follows:

$$head_i = attention\left(QW_i^Q, KW_i^K, VW_i^Q\right) \tag{18}$$

$$Multihead(Q,K,V) = Concat(head_1,\cdots,head_m)W^O \tag{19}$$

3) Self-Attention

This approach employs a self-attention to mine the internal relations of sentences, where [22] $K = V = Q$. For example, each word contained in an entered sentence, will have to do the attention calculation with all other words of the sentence. This MHAT mechanism generates a weight matrix α and a feature representation v.

$$u_t = tanh(W_s H_t + b_s) \tag{20}$$

$$v = Multihead(U,U,U) \tag{21}$$

D. Global average pooling layer

The fully connected network has always been the standard structure of the classification network. Generally, there is an activation function *softmax* to perform classification in a fully connected network. The function of the fully connected network is to stretch the feature map into a vector, to multiply the vector, and to ultimately reduce its dimension. This vector is input into a *softmax* layer to get the corresponding score of each category. However, there are two problems in the fully connected network. The first is that the number of parameters is too large, which reduces the speed of training, and the second is that it is easy to perform over-fitting. Global average pooling skillfully avoids the above shortcomings on the premise of achieving the same effect, which adds the input features sequences to the average [24]. For example, in the form of a three-dimensional tensor (of samples, steps, features), the output is a two-dimensional tensor (of samples, features). After introducing the MHAT mechanism to the whole sentence, the corresponding output feature matrix is v, and the feature vector of each word in the sentence is $(v_1, v_2, \cdots, v_n)$. The global average pooling of the entire sentence is expressed as follows:

$$v_{gap} = Global(v_1, v_2, \cdots, v_n) \tag{22}$$

E. Softmax layer

We feed the resulting vector v_{gap} directly to the *softmax* layer for sentiment analysis prediction. The prediction result is the following:

$$\hat{y} = softmax(wv_{gap} + b) \tag{23}$$

The purpose of cross-entropy was introduced to evaluate the model, which reflects the gap between the real emotional categories y and the predicted emotional categories $\hat{y}$.

$$loss = -\sum_i y_i \log \hat{y}_i \tag{24}$$

Where i is the index number of the sentence.

IV. EXPERIMENTAL ANALYSIS

A. Data set

The data in this paper are from the product reviews of Taobao e-commerce platform. In order to express their own emotions, customers scored the reviews in the range of 1-5, of which 1-2 is divided into negative and 3 is divided into neutral, 4-5 is divided into positive. The dataset contains 19,465 comments, of which the 7,854 are positive, 4,124 are neutral, and 7,487 are negative. Moreover, we have introduced a negative text label of -1, a neutral equal to 0, and a positive equal to 1. we divided the data set into a training set and a test one, following a ratio of 8:2. The data set sample is shown in table 1.

TABLE 1. DATA SET SAMPLES

positive	neutral	negative
很好用，快递很给力，帮我送到目的地。	作为个人经验在网上谈谈可以，但拿来出书就有点过了，书中还有些明显的谬误。不过文笔还不错，建议写写散文好了。	再也不想来这里买东西了，太慢了，10 天还没收到，很失望。
酒店很超值，干净卫生，，主动提出酒店有大床带有单床的房间，而且价格不变，下次去肯定还选它。	中规中矩的五星级酒店，但酒店双人房的床较小，其他还可以，毕竟也是连锁酒店。另外，酒店的大堂确实有些气势！	使用电源适配器容易发热，有点担心会烧坏，触摸板的按键有点硬。

B. Metrics

we tend to formulate corresponding metrics according to the actual application scenarios. For text classification, the most usual performance metrics are the following: precision *(p)*, recall *(r)*, F-1 scores (*f1*), accuracy (*acc*).

TABLE 2. CLASSIFICATION RESULT MATRIX

Predictive value	Real value	
	Class 1	Class 2
Class 1	True class1 (T_1)	False class2 (F_1)
Class 2	False class2 (F_2)	True class2 (P_2)

From the table above, we can get the following formula:

p: The ratio of the correct data for a certain type of prediction to the data for that type of prediction.

$$p = \frac{T_1}{T_1 + F_1} \tag{25}$$

r: The ratio of the correct data for a certain type of prediction to the actual data for that type.

$$r = \frac{T_1}{T_1 + F_2} \tag{26}$$

f1: Weighted harmonic mean of *p* and *r*.

$$f1 = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2(T_1 + F_1)(T_1 + F_2)}{T_1} \tag{27}$$

acc: The ratio of the number of correctly classified samples to the total number of samples.

$$acc = \frac{T_1 + T_2}{T_1 + F_1 + F_2 + T_2} \tag{28}$$

C. Experimental environment and parameters setting

1) Experimental environment

TABLE 3. EXPERIMENTAL ENVIRONMENT

projects	configuration
CPU	Core Tetranuclear i7-7700k
GPU	GTX1080 Ti GAMING X 11GB
system	Ubuntu16.04
frame	Tensorflow1.13.1 with Keras2.24
language	Python 3.7.1

2) parameters setting

TABLE 4. PARAMETER SETTING

Parameters	Value
Dimension(*d*)	100
Number of hidden units Bi-LSTM(*H*)	128
Maximum sentence length	100
Window_size	7
Heads(*m*)	8
Dropout (ρ)	0.5
Regularization coefficient (η)	0.001
Batch_size	64
Activation function	Relu
Epochs	30
Patience	10
Conv1d	256
Kernel_size	3
Pool_size	2
Optimizer	Adam[25]

TABLE 5. COMPARISON OF DIFFERENT ALGORITHMS

Model /word2vec	**Class 1: positive /%**			**Class 2: neural /%**			**Class 3: negative /%**			***acc* / %**
	p	***r***	***f1***	***p***	***r***	***f1***	***p***	***r***	***f1***	
CNN	89.90	88.99	89.44	92.52	93.88	93.20	88.73	88.97	88.85	90.01
BiLSTM	88.78	91.14	89.94	98.15	91.14	89.94	89.30	90.37	89.83	90.80
Attention -BiLSTM	91.65	89.56	90.59	99.73	90.45	94.87	86.75	93.25	89.88	91.16
BiLSTM-MHAT	92.71	90.20	91.44	97.28	93.46	95.33	88.90	93.34	91.06	92.11

D. Algorithm

Step 1. If loading Wikipedia is true:

(a) then convert wikipedia.xml to wikipedia.text;

(b) then segmentation (remove stop words);

(a) Let d=100, use word2vec or glove to train the word vector, and save the training result;

Step 2. Read data set file:

(a) Let {positive=1, negative=-1, neural=0};

(b) *Jieba* segmentation → statistical word frequency → build a dictionary;

(c) Set maximum sentence length=100, uniform text length;

Step 3. Load the pre-trained word vector:

(a) If the word is in the model, return the corresponding word vector;

(b) If the word is not in the model, return 0;

(c) The shape of the data set is (sample, 100, 100);

Step 4. Input train data set:

(a) Build a bidirectional LSTM with an output shape of (sample ,100, 256);

(b) Build a multi-head attention with an output shape of (sample ,100, 128);

(c) Build a global average pooling with an output shape of (sample ,128);

(d) Build the classification layer, in which the emotion classification is obtained by the *softmax* classifier (17), and the weight w is continuously updated by the loss function (24), and the final output shape is (sample, 3);

(e) Save model and weight;

Step 5. Input test data set:

(a) Load the pre-training model and weight;

(b) Return *p*, *r*, *f1*, *acc*, *loss*;

E. Experimental results and analysis

1) Comparison of word embedding

TABLE 6. THE WORD EMBEDDING PERFORMANCE

Word embedding	**Weighted average / %**			**acc / %**
	p	***r***	***f1***	
EML	91.33	91.14	91.18	91.14
Glove	91.41	91.32	91.35	91.31
Word2vec	92.23	92.11	92.13	92.11

The BiLSTM-MHAT model is used in this section. Additionally, the weighted average of *p*, *r*, and *f1* is measured for the overall performance of the word embedding. The overall classification effeciency of both the *word2vec* and the *glove* methods on the used dataset, is slightly better than that of the *Embedding layer*, and the *word2vec* achieves the optimal performance. Compared with the *Embedding layer*, the *word2vec* increased the efficiency by 0.9%, 0.97%, 0.95% and 0.91% on the *p*, *r*, *f1* and *acc* metrics respectively. Compared with *Glove*, the *word2vec* model achieved an improved performance by 0.82%, 0.79%, 0.78% and 0.80% on *p*, *r*, *f1* and *acc* respectively.

Although *word2vec* achieved the best results in this dataset, it did not prove its general superiority or inferiority. It has only shown that the *word2vec* is more suitable for the data set selected in this experiment. In addition, if we want to pursue faster training speed and to have more resources, the *embedding layer* is undoubtedly the best choice.

2) Comparisons of Algorithms

This section selected the *word2vec* algorithm to pre-train the word vector. The Bidirectional LSTM - Multi-head Attention model was compared to the CNN, to the Bidirectional LSTM, and to the Bidirectional LSTM - Attention. The effects of each class are measured by the *p*, *r*, and *f1* metrics and the overall classification effect of the model was measured by the *acc*. The numerical experimental results are shown in table 4.

Overall, the accuracy of the proposed model was equal to 92.11%, which is higher by 2.10% ,1.31% and 0.95% than that of the CNN, the Bidirectional LSTM and the bidirectional LSTM - attention respectively. From the perspective of categories, the model proposed in this paper achieves the best results in all three categories. Although the proposed model is not producing all indices *p*, *r*, and *f1*, we should pay more attention to *f1* compared to *p* and *r*, because it better reflects the overall performance of the model in this category. This is why we also ask for their weighted average *f1* after finding *p*, *r*. From a model's perspective, each model achieved better results in the neutral category than the other two categories, with the positive category followed by the negative category.

CNN is mainly used for image processing[26][27], and its performance on the sequence problem is less conspicuous. Bidirectional LSTM can learn the context information of the text, which is why it has achieved excellent results by virtue of its uncomplicated model. If we add a single attention to the bidirectional LSTM, the classification effect of the whole model will be improved slightly. If a Multi-head Attention mechanism is added, a large increase is achieved. The difference between the two is that the Multi-head Attention is calculated *m* times rather than only once, which has the advantage of allowing models to learn relevant information in different representation subspaces.

3) Attention heat-map visualization

Regarding the experimental results, the heat map was used to visualize the effect of the attention. We used the shade of the color to indicate the weight of the attention. The darker the color, the larger the weight, and the lighter the color, the smaller the weight. An example is shown below.

Fig. 6. Attention heat-map visualization examples

This paper provides a simple visualization sample for the attention-based NLP tasks which can visualize the text with attention weights as background. It considers the words' list and the corresponding weights as input and generate the figure to visualize the attention-based text. This figure can be used to demonstrate the attention ability of the model.

V. CONCLUSION

This research introduces a text sentiment analysis method, based on the BiLSTM with MHAT mechanism. The original text is first pre-processed as a word vector, and then the traditional LSTM model is replaced with a bidirectional LSTM to obtain the context of the text. At the same time, considering that the different components of the sentence have different degrees of influence on the final text sentiment classification, we should force on the specific components, such as a special set of related words or phrases. Therefore, this paper introduces the attention mechanism. However, a text may have multiple sentences, or a sentence may have multiple components that together form the overall semantics of the entire sentence. In order to understand them, we need to perform multiple self-attention mechanisms to extract the information we need from different parts of the text. This paper introduces a MHAT mechanism. We send the text features captured by the BiLSTM to the MHAT mechanism to obtain the influence weights of different features. Experiments show that the introduced model further improves the stability and the effectiveness, compared with the traditional DL method. In addition, we know that the quality of text into a word vector has a crucial impact on the classification effect. In the performed experiment, we focused on the performance of the *embedding layer, glove, and word2vec* by applying them on a specific data set, trying to determine the optimal *word embedding* algorithm.

In the course of the study, we found some limitations in this study. Data is not just a single word, but they have a more complex form. For example, the text is mixed with expressions, pictures, and even short videos. Identification and consideration of non-textual emotional information is a very interesting research topic. In addition, the data set is more fine-grained, and the data division does not apply only to three categories, but to many more. Of course, the above-mentioned questions surely lead to a more difficult and complex data pre-processing task, and to a higher overall complexity of the potential applied algorithm.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61863006 and 61762021, 61962010), the Science and Technology Planning Project of Guizhou Province (Grant Nos. 20191416 and [2017]1130, [2017]5726-32), Excellent Young Scientific and Technological Talents of Guizhou [2019]-5670.

REFERENCES

[1] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1-167, 2012

[2] H. Lu , Y. Li , M. Chen , H. Kim and S. Serikawa , "Brain Intelligence: go beyond artificial intelligence", *Mobile Networks and Applications*, vol.23, pp.368-375, 2018.

[3] H. Tang, S. Tan and X. Cheng, "Research on Sentiment Classification of Chinese Reviews Based on Supervised Machine Learning Techniques," *Journal of Chinese Information Processing*, vol. 21, no. 6, pp. 88-126, 2007.

[4] T. Mullen and N. Collier, "Sentiment analysis using support vector machines with diverse information sources," in *Proceedings of the 9th conference on empirical methods in natural language processing,* 2004.

[5] Y. Lecun, Y. Bengio, G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 28 May 2015.

[6] Y. Bengio, R. Ducharme, P. Vincent, et al., "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 6, pp. 1137-1155, 2006.

[7] T. Mikolov, I. Sutskever, K. Chen, et al., "Distributed Representations of Words and Phrases and their Compositionality," 2013, *arXiv:1310.4546.* [Online]. Available: https://arxiv.org/abs/1310.4546

[8] T. Mikolov, K. Chen, G. Corrado and D. Jeffrey, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st International Conference on Learning Representations*, 2 May 2013.

[9] J. Pennington, R. Socher, C. Manning, "Global vectors for word representation," *Proceedings of the 2014 conference on empirical methods in natural language processing,* 2014, pp. 1532-1543.

[10] Y. Kim, "Convolutional neural networks for sentence classification," 2014,*arXiv:1408.5882.*[Online].Available:https://arxiv.org/abs/1408.5882

[11] R. Socher, C. Manning, A. Ng, "Learning continuous phrase representations and syntactic parsing with recursive neural networks," in *Proceedings of the NIPS-2010 deep learning and unsupervised feature learning workshop,* 2010, pp. 1-9.

[12] D. Li, J. Qian, "Text sentiment analysis based on long short-term memory," in *Proceedings of IEEE International Conference on Computer Communication and the Internet*, 2016, pp. 471-475.

[13] R. Brueckner, B. Schulter, "Social signal classification using deep LSTM recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 4823-4827.

[14] Y. Liu, C. Sun, L. Lin, et al., "Learning natural language inference using bidirectional LSTM model and inner-attention," 2016, *arXiv:1605.09090.* [Online].Available: https://arxiv.org/abs/1605.09090

[15] Kim Yoon, Denton Carl, Hoang Luong and M. Alexander, "Structured attention networks," 2017, *arXiv:1702.00887.* [Online]. Available:https://arxiv.org/abs/1702.00887

[16] Z. Yang, D. Yang, C. Dyer, et al., "Hierarchical attention networks for document classification," in *Proceedings of the conference on Human Language Technologies*, 2016, pp. 1480-1489.

[17] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is all you need," in *Proceedings of the conference on Advances in Neural Information Processing Systems*, 2017, pp. 5998-6008.

[18] Y. Wang, M. Huang, L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *Proceedings of the conference on empirical methods in natural language processing,* 2016, pp. 606-615.

[19] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol.15, no. 1, pp. 1929–1958, 2014.

[20] A. Graves, N. Jaitly and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proceedings of IEEE workshop on automatic speech recognition and understanding*, 2013, pp. 273-278.

[21] X. Wan, "Co-training for cross-lingual sentiment classification," in *Proceedings of the 4th International Joint Conference on Natural Language Processing*, 2009, vol. 1, pp. 235-243.

[22] I. Sutskever, O. Vinyals, Q. Le, "Sequence to sequence learning with neural networks," in *Proceedings of Advances in neural information processing systems*, 2014, pp. 3104-3112.

[23] M. Lin, Q. Chen, S. Yan, "Network in network," 2013, *arXiv:1312.4400.* [Online]. Available: https://arxiv.org/abs/1312.4400

[24] Diederik P. Kingma, Ba Jimmy, "A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations*, 2015, 5.

[25] X. Zhang, J. Zhao, Y. LeCun, "Character-level convolutional networks for text classification," 2014, *arXiv:1412.6980.* [online]. Available: https://arxiv.org/abs/1412.6980

[26] Q. Zhou, W. Yang, et al., "Multi-scale deep context convolutional neural networks for semantic segmentation," *World Wide Web*, vol. 22, no. 2, pp. 555-570, 2019.