Text Sentiment Classification Method Based on Bilstm

Wenliang Wang

School of Data Science and Technology, South China Normal University, Guangzhou, China

* Corresponding Author Email: 20218131014@m.scnu.edu.com

Abstract. In the booming age of digital communication and natural language processing(NLP), understanding user sentiments expressed in texts has become increasingly vital. This research addresses the challenge of classifying text sentiments into three distinct categories: positive, neutral, and negative. Recognizing the nuances of human emotion in textual content is paramount for enhancing user experiences and tailoring personalized digital interactions. This study trains a BiLSTM model, supplemented with optimization strategies to adaptively learn from a diverse dataset. Additionally, a large number of baseline experiments were conducted to compare its efficacy with traditional algorithms such as Logistic Regression. The BiLSTM model demonstrated a good accuracy of 78.21% on the test set. Conclusively, while the proposed model shows significant potential, further refinements could be made, emphasizing data augmentation and model regularization to achieve optimal performance.

Keywords: Text sentiment analysis; Bi-directional Long Short-Term Memory; Word2Vec.

1. Introduction

With the advancement of internet technology and the proliferation of social media and digital communication, social networks like Twitter have emerged as pivotal platforms for user sentiment feedback and emotional expression. More and more users prefer to tweet their views and emotions, resulting in a deluge of comments and other textual content on the internet. This plethora of data not only offers a rich source for Natural Language Processing (NLP) but also presents substantial challenges. In this backdrop, sentiment analysis or opinion mining has progressively become a focal area of research in NLP, striving to automatically extract, understand, and interpret the sentiments or views of authors from textual data. In recent years, text sentiment analysis has aroused a wide range of research interests, showing its wide applicability in different fields such as enterprise, government, financial sector, etc.

Text sentiment analysis typically uses methods such as tokenization and vectorization to pull out emotional features from the text. The goal is to figure out the emotion conveyed by the text, which helps in monitoring and predicting the author's sentiment. Looking at the development of sentiment analysis, we can see three main approaches: those based on lexicons, traditional machine learning, and deep learning. In this paper, we present a model that combines CNN with BiLSTM, and among many vectorization methods, it can be combined with Word2Vec to get good results. The comparison of the final results with the baseline experiments can prove that this model is superior to traditional machine learning models.

Recently, there's been more research into a deeper area called emotion analysis, which is not the same as sentiment analysis. While emotions and sentiments are closely related, emotion analysis looks at more detailed, multi-category emotions. It's an interesting field with a lot to do, but it's separate from what we're discussing here. This paper won't go into that area.

2. Literature review

In the realm of text sentiment analysis, lexicon-based approaches were first introduced. As technology progressed, machine learning approaches garnered more attention due to their superior classification outcomes. In recent years, deep learning techniques have been demonstrated to possess even better classification capabilities.

2.1. Lexicon-based approaches

Lexicon-based approaches can reflect the unstructured characteristics of short texts. These approaches utilize existing sentiment dictionaries to filter emotional words in texts and categorize sentiments accordingly. Ma et al. [1] developed a chat system based on sentiment dictionaries. This system leverages the dictionary to extract emotion-related vocabulary from conversations, then calculates sentiment weights of sentences by weighing individual words, adjusted further by syntactic features. Aman and others [2] combined sentiment dictionaries with emotional intensity databases to classify emotions in blog texts, achieving an accuracy of over 66%. Hogenboom and colleagues proposed a cross-lingual sentiment analysis technique based on lexicons [3]. Their findings suggest that using a highly effective emotion dictionary from one language might not be sufficient for analyzing sentiments in a different target language. This is especially the case when the text is merely machine-translated to a reference language and then subjected to emotion classification using existing methods.

However, it is evident that lexicon-based approaches have some shortcomings. While they can handle straightforward sentences effectively, their efficiency significantly diminishes with more complex sentences.

2.2. Traditional machine learning approaches

To address the limitations of the lexicon-based approach in sentiment analysis, researchers began applying machine learning techniques. Pang et al. [4] were pioneers in integrating machine learning into sentiment analysis. Their methodology involved generating syntactic trees from detailed sentence structure analyses. They employed the bag-of-words model in tandem with Naive Bayes, Maximum Entropy Classifier, and Support Vector Machine for an in-depth study and training on IMDB movie review data.

Following that, Tang et al. [5] developed a deep learning system for message-level microblog sentiment analysis. This system combined sentiment-specific word embedding with manually selected emoticons, semantic dictionaries, and other features, using a Support Vector Machine for sentiment classification. Ouyang Chunping's team [6] integrated SVM and KNN algorithms with Naive Bayes for a comprehensive classification study on Sina Weibo text data. In the NLPCC2013 Weibo dataset evaluation, they achieved an F-value of 34%, considered a leading result in the field at the time.

2.3. Deep learning approaches

With the rise of deep learning methodologies, an increasing number of researchers have incorporated them into text sentiment analysis. Common models include Recurrent Neural Networks (RNN), which have advantages in handling sequential data. However, standard RNNs encounter gradient vanishing and exploding problems. Consequently, many scholars shifted their focus to LSTM, effectively addressing long-term dependencies. Teng et al. [7] were among the first to apply Bidirectional Long Short-Term Memory (BiLSTM) for text processing. BiLSTM processes both forward and backward sequence information, capturing contextual data to enhance accuracy. Wang Liya's team [8] introduced a novel text sentiment analysis technique based on character-level word vectors, employing a dual-channel CNN-BiGRU hybrid network strategy.

3. Methodology

This section will introduce the methods used in this article from two perspectives: Text preprocessing and vectorization, model construction.

3.1. Word2Vec

Proposed by Mikolov et al. in 2013[9], Word2Vec is an innovative technique that has dramatically affected the field of NLP. It captures semantic meanings based on the principle that words appearing in similar contexts often have similar meanings.

Traditional NLP tasks used to treat words as atomic units. For instance, the word "school" would be represented as one-hot encoded vectors where its index would be 1, and everything else would be 0. The concept of distributed representations can be traced back to the works of Hinton, McClelland, and Rumelhart in the 1980s [10]. With Word2Vec, words are represented by dense vectors where each word is characterized by many attributes (in the form of dimensions). Such representations are called distributed representations because the semantic information of a word is distributed across all dimensions.

Word2Vec encompasses two architectures: the Continuous Bag-of-Words (CBOW) and the Continuous Skip-gram (Skip-gram).

The CBOW model consists of three layers: the Input, Projection, and Output. CBOW aims to learn word representations so as to predict the target word using its context.

Conversely, the Skip-gram model, which operates in the opposite manner, predicts the context using a given word. Therefore, the model seeks to discern which words are likely to co-occur. For instance, it might predict surrounding words like "the", "cat", and "on" when given the word "sits". This model is particularly adept at handling large datasets and is more robust in dealing with infrequent terms. The objective function of the Skip-gram endeavors to maximize the probability of a word co-occurring with its contextual words. Typically, this is accomplished through the dot product of the embedding vectors and normalizing via the softmax function. The flow chart of skip-gram is shown in Fig.1

Another noteworthy feature of Word2Vec is its ability to capture analogies. For example, the mathematical relationship between the vectors for the words "king" and "man" resembles that between "queen" and "woman".

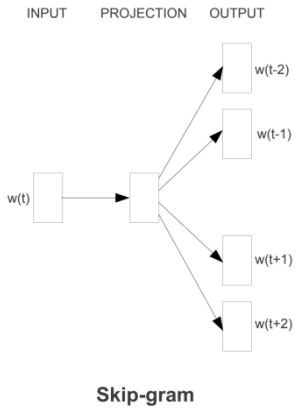


Fig. 1 Skip-gram

3.2. BiLSTM

Deep learning, in the context of sentiment analysis of text, operates by harnessing multi-layered neural networks to learn the intrinsic representations of textual data, and subsequently uses these representations to predict the sentiment of the text. In contrast to traditional machine learning techniques, deep learning excels at detecting more intricate patterns and nonlinear relationships.

3.2.1 BiLSTM overview

To comprehend the workings of BiLSTM, we first need to grasp the foundation upon which it is built: LSTM. The Long Short-Term Memory (LSTM) is a variant of the Recurrent Neural Network (RNN), specifically designed to learn and remember over long sequences and is resistant to the vanishing gradient problem that affects conventional RNNs.

Central to the LSTM's design are its "gate" mechanisms: namely, the input, forget, and output gates. These gates determine how new information is stored, old information is discarded, and the manner in which values are output from the LSTM unit. A visual representation of these gates and their associated mathematical formulas is provided below.

$$I_t = \sigma(X_i W_{xi} + H_{t-1} W_{hi} + b_i) \quad (1)$$

$$O_t = \sigma(X_i W_{xo} + H_{t-1} W_{ho} + b_o) \quad (2)$$

$$O_t = \sigma(X_i W_{xo} + H_{t-1} W_{ho} + b_o) \quad (3)$$

Through the integration of the forget gate, LSTMs can judiciously modulate the flow of information, thereby addressing the challenges of gradient vanishing and explosion prevalent during training over long sequences. This inherent capability makes LSTMs more apt at processing extended sequences compared to traditional RNNs.

3.2.2 BiLSTM - Extending LSTM's Capabilities

Building on the foundational principles of LSTM, the Bi-directional Long Short-Term Memory (BiLSTM) augments its capabilities by simultaneously processing information from the past (backward LSTM) and the future (forward LSTM). While LSTMs significantly mitigate the vanishing gradient problem and efficiently capture long-term dependencies, their unidirectional nature can be restrictive, especially in tasks like sentiment analysis where the context from subsequent words can be as pivotal as preceding ones.

BiLSTM leverages the strengths of two LSTMs, processing sequences from both ends, thereby encapsulating a richer contextual representation of data. This dual directionality has been instrumental in enhancing the performance of various NLP tasks, sentiment analysis being a case in point.

In our research, we introduce a deep learning model, which integrates the spatial feature capturing prowess of Convolutional Neural Networks (CNNs) with the temporal sequencing mastery of BiLSTMs. This synergy is optimized to interpret both the hierarchical structure and the time-sensitive nuances in textual data, ensuring a holistic understanding that is important for precise sentiment classification.

4. Experiment

4.1. Description of dataset

The primary foundation of this study lies in text-based data. Accordingly, we turned to a major social media platform, Twitter, recognized as the largest microblogging platform with over 260 million active users monthly, boasting half a billion tweets daily. This platform is regarded as a reliable and credible channel for information dissemination. The dataset employed for this research pertains to sentiment analysis work regarding issues of six major US airlines. It encompasses over 14,000 data entries, which were scraped in February of 2015. Contributors were solicited to classify tweets as positive, negative, or neutral initially.

The dataset, "US Airline sentiment tweets," was sourced from Kaggle for this study. The specific link is https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment.

Key metrics utilized in this research include text content, sentiment (positive, negative, neutral), airline, and reasons for negativity.

4.2. Data preprocessing

Data preprocessing is a core step in any machine learning or deep learning project, which directly affects the performance and accuracy of the model. The purpose of data preprocessing is to transform raw data into a format that is easier to process and analyze.

In text sentiment classification tasks, text preprocessing and vectorization are particularly critical, because they directly determine how the model interprets and understands the input text data.

4.2.1 Text preprocessing

Text Cleaning: Before processing, it's imperative to ensure that the text is devoid of any extraneous or irrelevant content. Text cleaning primarily involves the removal of content like HTML tags, URLs, non-textual symbols, and any other non-contextual elements that may not add significant value to the analysis. Additionally, standardizing the text, such as converting it to lowercase, ensures consistency, making the further stages of preprocessing more effective. Pythons re library, which leverages the power of regular expressions, offering precise and flexible text search and replacement capabilities.

Tokenization: Textual data, in its raw form, often comprises extensive information sequences that can be difficult for machines to directly analyze or comprehend. Tokenization comes to the rescue by breaking down extensive texts into smaller, more digestible chunks, typically individual words or tokens. This decomposition aids in the subsequent stages of analysis by treating each word or phrase as a distinct entity. So, for the purpose of tokenization, the word_tokenize function from the nltk library was utilized.

Stop Words Removal: Stop words frequently appear in texts but often hold minimal significance, especially in tasks like sentiment analysis. Examples include words like "is", "the", and so on. While they serve essential grammatical purposes, they can introduce noise when analyzing text patterns, especially in tasks such as sentiment analysis. By removing such words, the volume of text data gets significantly reduced, thereby streamlining the dataset and potentially boosting the performance of machine learning models. Furthermore, the removal of these words accentuates the words that truly matter, making patterns more discernible.

4.2.2 Vectorization

Regarding vectorization, we have introduced Word2Vec earlier. During the experiment, Word2Vec and the following two vectorization methods were used. Subsequently, these methods will be combined with traditional machine learning models to serve as a baseline experiment.

CountVectorizer:CountVectorizer is a technique that transforms text data into vectors. It counts the occurrences of each word in a document. Each unique word is assigned a unique ID. When a document is transformed into a vector, each position in the vector corresponds to a unique word's ID, and the value at that position indicates the frequency of that word in the document. This method treats each word's frequency as its feature value, making it a simple yet effective approach for many NLP tasks.

TfidfVectorizer:TfidfVectorizer goes beyond simple counting. TF-IDF stands for Term Frequency-Inverse Document Frequency. This technique not only takes into account the frequency of a word in a specific document (Term Frequency) but also how often it appears across all documents (Document Frequency). If a word appears frequently in a single document but not in many other documents, it's given more weight, implying its importance in that particular document. This method helps in prioritizing words that might be more contextually relevant over those that frequently appear across all documents, like common stopwords.

4.3. Model training

In this section, we first explore traditional machine learning approaches. This sets the baseline for our subsequent deep learning methods. Afterwards, we introduce a deep learning approach using bidirectional long short-term memory networks (BiLSTM) and explore its architecture and parameter selection in detail.

4.3.1 Baseline Model Training

When referring to "traditional" machine learning algorithms, we're speaking about conventional machine learning algorithms. There are many such algorithms, including well-known ones like Naive Bayes, Random Forest, Logistic Regression, Support Vector Machines, and so forth. In this experiment, I chose LR and SVM.

First up, Logistic Regression. Logistic Regression is a predictive model tailored for binary classification problems. For the text sentiment analysis task in this study, we adapted it as a three-

classification model (positive, neutral, negative). In the experiment, we need to constantly adjust the parameters of the model to optimize its performance.

LR can be optimized using regularization techniques, which can be either L1 (Lasso) or L2 (Ridge) regularization. This optimization helps in averting potential overfitting by penalizing high-valued coefficients, thereby ensuring that the model remains general enough for unseen data. Among them, when choosing regularization techniques in text sentiment analysis tasks, L2 regularization is usually more common than L1 regularization, and L2 regularization has a relatively uniform effect on features and will not completely set the coefficients of some features to zero, thus preserving more characteristic information. In text sentiment analysis, different words may carry emotional information in different contexts, so retaining more features helps to capture this information better. Therefore, set penalty=l2.

'C' represents the inverse of the regularization strength. A lower value indicates stronger regularization, which can help prevent overfitting by constraining coefficients.We set C=1.0 here.

'Class weight' used to handle unbalanced classes. For example, in my dataset, the 'positive' sentiment category appears less often than other categories, so fewer categories should be assigned higher weights. So, it can be set to 'balanced' to automatically calculate the weight.

'Solver' is an algorithm for an optimization problem.The 'saga' solver is suitable for large datasets. It uses an optimized version of stochastic average gradient descent, which can converge faster.

Multi_class = 'multinomial' option is chosen because I am dealing with a multiclass classification task(pos,neu,neg). It uses a multinomial logistic regression formulation to handle multiple classes.

max_iter=500: The maximum number of iterations for the solver to converge. Increasing it ensures the optimization process has a chance to converge to a solution.Especially when using 'saga', it is necessary to set a large number of iterations.

Next is the SVM (Support Vector Machine). SVM is a machine learning algorithm introduced in 1995 by Cortes and colleagues. They integrated ideas from statistical learning theory, optimization theory, and kernel theory. By following the principle of structural risk minimization, they proposed this classic algorithm, initially applying it to binary classification tasks [11]. Subsequent researchers have extended its application to multi-class tasks.

In this model, the sigmoid kernel was chosen, which is suitable for complex datasets, such as our high-dimensional text data. Non-linear kernels often deliver better performance for such datasets. The regularization parameter, C, is set to the default value of 1.0, striking a balance where the model is neither too regularized nor prone to overfitting. By using "balanced" as the class weight with the class_weight parameter, the model gives more importance to under-represented classes. This prevents the model from being biased towards the dominant classes, ensuring better predictive performance across all categories. The gamma parameter defines the influence of a single training sample on the decision boundary. With the value set to "scale", it calculates gamma based on the data features, typically offering an appropriate balance for the decision boundary to be neither too soft nor too rigid. Lastly, with max_iter set to -1, the model iterates indefinitely until a convergent solution is found.

4.3.2 BiLSTM model

In our research, we've designed a deep learning architecture, merging the capabilities of convolutional neural networks (CNNs) and bidirectional long short-term memory (BiLSTM) networks. This is exactly the model proposed in this article. The following is the specific model construction and training.

Word Embedding Initialization:Prior to model construction, we initialized an embedding matrix based on pre-trained word vectors from a word2vec model (w2v). This matrix is constructed such that each word indexed by the tokenizer is represented by its corresponding vector from the w2v model. Only the words present in our tokenizer.word_index and within the vocabulary of the word2vec model are considered. The choice to harness external embeddings is motivated by the opportunity to leverage pre-existing semantic relationships between words and to accelerate the model's convergence during training.

Embedding Layer:The foundational layer of the model is an embedding layer. With a vocabulary size (vocab_size) set to 5,000, each word is transformed into a 100-dimensional vector (embedding_size). The embedding weights are initialized with our previously constructed embedding_matrix. Moreover, these weights are set as trainable, allowing the model to fine-tune them based on the task-specific dataset. This ensures that while the model commences training with generalized semantic relationships from the word2vec model, it can further adapt these embeddings to the nuances of our dataset.

Convolutional Layer:Post-embedding, a 1-dimensional convolutional layer is integrated. This layer is designed with 32 filters, and 3-sized kernels. By opting for 'same' padding, the spatial dimensions of the resultant feature maps mirror those of the input, preserving spatial context. The Rectified Linear Unit (ReLU) activation function introduces the required non-linearity, assisting the model in deciphering intricate patterns from the data.

Pooling Layer:A max-pooling operation ensues, with a designated pool size of 2. Such downsampling techniques are renowned for their efficacy in extracting dominant features and curtailing computational demands.

Bidirectional LSTM Layer:Central to our model's architecture are two consecutive bidirectional LSTM layers, each comprising 32 units. This design choice emerged from iterative experimentation: early model architectures were trialed with a single LSTM layer, but through repeated validation and performance evaluation, it was observed that incorporating a second LSTM layer significantly bolstered the model's capacity to capture sequential dependencies in the text.LSTMs inherently are adept at remembering long-term dependencies in sequences, and when used bidirectionally, they can capture both past (backward) and future (forward) contexts. By employing two of these layers in succession, our model can harness a deeper and more layered understanding of the textual data. The first LSTM layer returns sequences, ensuring the subsequent LSTM layer receives a sequence input, thereby perpetuating the sequential context through the depth of the model.The decision to deploy two layers instead of one was influenced by the complexity of sentiment analysis in textual data, especially when subtle contextual cues can significantly influence the sentiment interpretation. By doubling up on the LSTM layers, our model stands better equipped to discern intricate textual patterns and nuances that might be missed with a shallower architecture.

Dropout Layer:To enhance model robustness, a dropout layer with a rate of 50% is added. This layer randomly deactivates a subset of neurons during training, mitigating the risk of overfitting and enhancing the model's generalization capabilities.

Dense Output Layer:Concluding the architecture is a dense output layer featuring 3 neurons, corresponding to the three sentiment classes (positive, negative, neutral). A 'softmax' activation function is employed, normalizing the output to interpret as class probabilities.

Our envisioned architecture combines the feature extraction capabilities of convolutional layers with the sequence modeling strengths of BiLSTMs, all underpinned by rich word embeddings from a pre-trained word2vec model. This combination aims for a comprehensive sentiment analysis approach.

5. Result

5.1. Baseline Model training

To begin with, we analyzed the accuracy achieved through combinations of traditional machine learning models, namely Logistic Regression and Support Vector Machine (SVM), with feature extraction techniques: CountVectorizer and TfidfVectorizer. The accuracy of each baseline model is shown in Table 1.

Table 1. Baseline model accuracy

	LR	SVM
CountVectorizer	75.93%	75.30%
TfidfVectorizer	76.04%	76.17%
Word2Vec	71.60%	69.82%

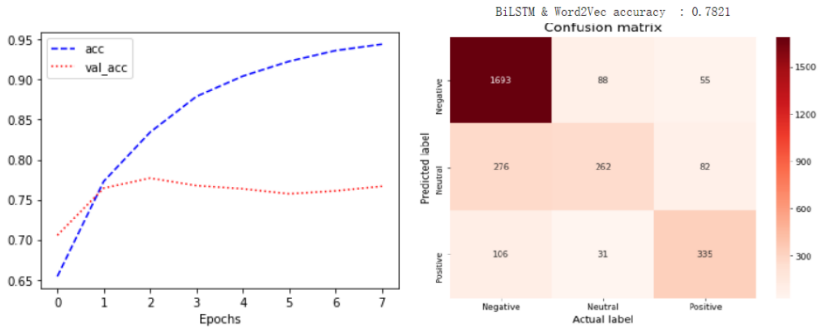
The results indicate that the accuracy achieved is commendable, setting a competitive baseline for our advanced models.

Subsequently, we experimented with integrating the Word2Vec embeddings with these traditional machine learning models.Contrary to our expectations, when using the more advanced and complex Word2Vec word vector model combined with traditional machine learning algorithms, the effect is even worse. But after reflection, it is understandable, because Word2Vec is more suitable for sequence models or more complex models, which is why we will try the BiLSTM model next.

5.2. Proposed Optimal Model-BiLSTM

In our quest for excellence, we introduced an optimal model, capitalizing on the synergy between Word2Vec embeddings and a BiLSTM architecture. The convergence of these two technologies aimed to capture the sequential intricacies of text data, with Word2Vec providing dense vector representations and BiLSTM offering memory cells that can remember long-term textual dependencies.

Let's first examine the training process. As the number of iterations increases, the accuracy consistently improves.



(a) **(b)**

Fig. 2 Performance of our model: (a) Accuracy (b) Confusion Matrix

The BiLSTM model demonstrates excellent performance on the training set. However, a noticeable discrepancy arises when observing the model's accuracy on the validation set. Despite this gap, an accuracy of 78% still highlights the model's decent capability in handling unseen data. In future iterations, we might consider incorporating more strategies to counteract overfitting and further optimize the model.

The accuracy and confusion matrix for our proposed BiLSTM model are shown as Fig.2.

Our experiments show that the combination of Word2Vec and BiLSTM outperforms our baseline method. It is worth noting that by changing the depth of the BiLSTM model and increasing the number of BiLSTM layers, the system is better able to identify subtle contextual clues, thereby enhancing its emotion prediction capabilities. After many trials, we concluded that the model works best when it contains two consecutive BiLSTM layers.

6. Conclusion

In terms of results, our BiLSTM model achieves an accuracy of 78.21%. While this is a good outcome, there remains significant room for improvement. In the future, to enhance the model, we could start by addressing potential issues with our dataset. The volume of data might be insufficient.

Deep learning models, particularly those leveraging word embeddings, often require a substantial amount of data to fully capitalize on their potential. Moreover, as mentioned in Section 5.2, the experiments exhibited signs of overfitting. While I've integrated Dropout layers to mitigate this, it might still be necessary to make further adjustments or employ other regularization techniques. Additionally, the current model's architecture might not be optimal. Upcoming endeavors should involve modifying the size of the convolutional kernels, appending more convolutional layers, and fine-tuning the number of LSTM units. Lastly, experimenting with alternative word vectors, such as GloVe, might be a promising avenue for optimizing the experiments.

References

[1] Ma C.L., Osherenko A., Prendinger H., et al. A chat system based on emotion estimation from text and embodied conversational messengers. In Proc of the 4th Int Conf on Active Media Technology, 2005: 546-548. Piscataway, NJ: IEEE.

[2] Aman S., Szpakowicz S. Identifying expressions of emotion in text. Proc of the 10th Int Conf on Text, Speech and Dialogue. Berlin: Springer, 2007: 196-205.

[3] Hogenboom A., Heerschop B., Frasincar F., et al. Multi-lingual support for lexicon-based sentiment analysis guided by semantics. Decision Support Systems, 2014, 62(1): 43-53.

[4] Pang B., Lee L., & Vaithyanathan S. Proceedings of Empirical Methods in Natural Language Processing (EMNLP2010), 2002: 79-86.

[5] Tang D., Wei F., Qin B., et al. Cooooll: A deep learning system for Twitter sentiment classification. In Proc of the 8th int Workshop on Semantic Evaluation, 2014: 208-212. Stroudsburg, PA: ACL.

[6] Ouyang Chunping, Yang Xiaohua, Lei Longyan, et al. Multi-strategy Approach for Fine-Grained Sentiment Analysis of Chinese Microblog. Acta Scientiarum Naturalium Universitatis Pekinensis, 2014, 50(01): 67-72.

[7] Teng Z., Vo D.T., & Zhang Y. Context-Sensitive Lexicon Features for Neural Sentiment Analysis. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016: 1629-1638.

[8] Wang Liya, Liu Changhui, Cai Chunbo, et al. Chinese text sentiment analysis based on character-level two-channel composite network. Application Research of Computers, 2020, 37(09): 1-6.

[9] Mikolov T., Sutskever I., Chen K., et al. Distributed Representations of Words and Phrases and their Compositionality. In NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013: 3111-3119.

[10] Hinton G. E., McClelland J. L., & Rumelhart D. E. Distributed representations. In Parallel distributed processing: explorations in the microstructure of cognition, 1986, Vol. 1: 77-109. MIT Press.

[11] Cortes C., & Vapnik V. Support-vector networks. Machine Learning, 1995, 20(3): 273-297.