Article

Sentiment Analysis of Chinese Product Reviews Based on Fusion of DUAL-Channel BiLSTM and Self-Attention

Ye Yuan *, Wang Wang, Guangze Wen, Zikun Zheng and Zhemin Zhuang

College of Engineering, Shantou University, Shantou 515063, China; 21wwang1@stu.edu.cn (W.W.); 21gzwen@stu.edu.cn (G.W.); 21zkzheng@stu.edu.cn (Z.Z.); zmzhuang@stu.edu.cn (Z.Z.)
* Correspondence: yuanye@stu.edu.cn

Abstract: Product reviews provide crucial information for both consumers and businesses, offering insights needed before purchasing a product or service. However, existing sentiment analysis methods, especially for Chinese language, struggle to effectively capture contextual information due to the complex semantics, multiple sentiment polarities, and long-term dependencies between words. In this paper, we propose a sentiment classification method based on the BiLSTM algorithm to address these challenges in natural language processing. Self-Attention-CNN BiLSTM (SAC-BiLSTM) leverages dual channels to extract features from both character-level embeddings and word-level embeddings. It combines BiLSTM and Self-Attention mechanisms for feature extraction and weight allocation, aiming to overcome the limitations in mining contextual information. Experiments were conducted on the onlineshopping10cats dataset, which is a standard corpus of e-commerce shopping reviews available in the ChineseNlpCorpus 2018. The experimental results demonstrate the effectiveness of our proposed algorithm, with Recall, Precision, and F1 scores reaching 0.9409, 0.9369, and 0.9404, respectively.

Keywords: natural language processing; sentiment classification; BiLSTM; self-attention; scalable multi-channel

check for updates

Citation: Yuan, Y.; Wang, W.; Wen, G.; Zheng, Z.; Zhuang, Z. Sentiment Analysis of Chinese Product Reviews Based on Fusion of DUAL-Channel BiLSTM and Self-Attention. *Future Internet* **2023**, *15*, 364. https://doi.org/10.3390/fi15110364

Academic Editors: Edson Talamini, Letícia De Oliveira and Filipe Portela

Received: 22 October 2023
Revised: 7 November 2023
Accepted: 9 November 2023
Published: 10 November 2023

1. Introduction

With the development of the Internet, online transactions have become increasingly common. This has led to the emergence of a new business phenomenon—product reviews, which help consumers make informed choices and provide feedback on product usage. However, the manual review and categorization of product reviews incur high costs. Therefore, efficiently processing and classifying product reviews as positive or negative is a pressing issue for many businesses [1]. With the advancement of computational power, deep neural networks have shown promising results in text sentiment analysis. Methods such as Long Short-Term Memory (LSTM) neural networks [2], Convolutional Neural Networks (CNN) [3], and the use of character-level and word-level embeddings have gained popularity. Deep learning techniques have also garnered attention in the field of business data processing [4].

Text sentiment analysis, also known as opinion mining or sentiment analysis, is the process of analyzing, processing, summarizing, and inferring subjective texts with emotional connotations. Currently, the mainstream research directions in text sentiment analysis can be broadly categorized into three types: sentiment lexicon-based methods [5], machine learning-based methods [6], and deep learning-based methods [7]. These techniques find applications in various domains, including most social media platforms in both English and Chinese languages, online reviews, business investments, and transfer learning. Examples of such domains include Twitter, Weibo, product reviews, and stock markets, among others.

The machine learning-based approach involves training a sentiment classifier using labeled data, which can be categorized into supervised, semi-supervised, and unsupervised learning methods. The main difference lies in the amount of labeled data used during the training process. Three traditional machine learning models commonly used in sentiment analysis are SVM (Support Vector Machine) [8], NB (Naive Bayes) [9,10], and KNN (K-Nearest Neighbors) [11,12]. Wawre [13] conducted a comparative analysis of SVM and NB methods and found that when the dataset is large enough, Naive Bayes achieves higher accuracy than SVM. Huq et al. [14] applied KNN and SVM for sentiment polarity identification in Twitter texts. In KNN, they classified texts based on the Euclidean distance measured between them, while SVM used principal component analysis to classify sentiment labels. Due to the low dimensionality of the experiment, it was difficult to determine the hyperplane in SVM, making KNN more effective in classifying positive and negative emotions. Machine learning, benefiting from a larger training dataset, provides more accurate sentiment analysis with better scalability and reproducibility. However, machine learning-based text sentiment classification methods still have some limitations compared to deep learning. Higher accuracy relies on high-quality training datasets, which require costly manual labeling. Moreover, subjective data labeling can impact the final classification performance. Therefore, model tuning and optimization are necessary to enhance the robustness and generalization ability of the model.

In the domain of natural language processing (NLP), LSTM (Long Short-Term Memory) has been widely used in sentiment analysis. LSTM is a special type of recurrent neural network (RNN) that effectively captures semantic information in text and addresses the issue of long-term dependencies. However, LSTM can only infer future information based on the preceding context. In Chinese language environments, BiLSTM (Bidirectional Long Short-Term Memory) [15] is employed to both consider contextual aspects and capture bidirectional semantic dependencies, thus improving the accuracy of sentiment analysis models. To enhance the accuracy of sequence models, the Attention mechanism was introduced by Bengio et al. in 2015 [16]. The core idea is to establish a fuzzy cognition at a macroscopic level and assign higher weights to more important features within the sequence. This mechanism leads to significant improvements in the performance of sequence models. In 2017, Ashish Vaswani et al. [17] further extended the Attention mechanism to propose Self-Attention. It involves taking a weighted average of all inputs, mapping them to a range between zero and one using softmax, and ensuring that the sum of all weights equals one. By assigning greater weights to important tokens, the model converges faster and exhibits better robustness. In recent years, with the integration of Self-Attention in the field of text sentiment analysis [18–20], related algorithms have achieved promising results in sentiment analysis tasks [21].

The performance of deep learning in sentiment analysis varies with the granularity of the input. When larger granularity sentences are used as input, the effectiveness is not as good as when more fine-grained inputs like word vectors or character vectors are used [22]. To deal with large volumes of text data, Google proposed Word2Vec word embedding tools in 2013. In 2014, Kim et al. [23] trained word embeddings using Word2Vec to transform words in text into fixed dimensional word embeddings, which were then used as inputs for convolutional neural networks (CNNs). Various kernel sizes were employed to extract local features, effectively validating the usefulness of word embeddings. Meanwhile, the Kalchbrenner team [24] designed K-Max Pooling based on Max Pooling principles to extract the k highest-ranking features in each sliding process. This method has gradually been applied in various fields. In 2015, the Zhou team [25] proposed a solution to the lack of context information in CNN by incorporating semantic information from context. By combining the strengths of CNN and LSTM [26], this approach was applied to text sentiment analysis, yielding promising results. In order to enhance the ability to capture textual information, Dilated Convolution Networks [27] were subsequently applied to text sentiment analysis. In 2019, Liu et al. [28] combined BiLSTM with the Attention mechanism, focusing on the hierarchical structure of text, and achieved good results across different datasets.

In 2021, Chenquan Gan [29], using a multi-channel CNN-BiLSTM with attention for sentiment analysis in Chinese text, involved leveraging both the original contextual features and multi-scale higher-level contextual features in a joint architecture of CNN and RNN. This was achieved by incorporating different scales of high-level features in the multi-channel architecture, which allowed for capturing both the original contextual characteristics and the multi-scale higher-level contextual characteristics. Ruifan Li [30] proposed DualGCN, which consists of two modules: SynGCN and SemGCN. SynGCN module was used as a generator, while SemGCN and Attention were combined to form a dual-channel model that simultaneously considers the complementarity between syntactic structures and semantic relationships.

However, since the attention mechanism mainly focuses on the output layer, it fails to leverage the relationships between different parts of a sequence when dealing with inputs of mixed granularity. While these methods have achieved good results, there still remains a problem of underutilizing the correlations between different granularities when using only the BiLSTM model connected with attention at the character level.

In addition, BERT has also achieved great success in text sentiment analysis [31]. BERT aims to pre-train deep bidirectional representations of unlabeled text by jointly conditioning on both left and right context across all layers. Therefore, fine-tuning the pre-trained BERT model with just an additional output layer allows for the creation of state-of-the-art models for various tasks without requiring extensive architectural modifications for specific tasks. K. Pipalia et al. [32] provided a comprehensive comparison of different BERT models in the context of sentiment analysis. However, BERT-based models also face the issue of not being able to directly handle excessively long texts and can only process a maximum of 512 tokens consecutively. Due to the linguistic characteristics, Chinese long texts can generate tokens that exceed the length of the original text by twofold, thus limiting the comprehensive processing of lengthy texts and requiring truncation. Researchers from the Allen Institute for AI [33] proposed the Longformer model based on the foundational Transformer model in 2020, which can handle sequences of up to 4096 tokens. This represents a significant increase compared to the 512-token limit of BERT variants.

To address the problems of insufficient correlation mining, the impossibility to process long text all at once and vague mapping relationships within the same sequence, this paper proposes the Sac-BiLSTM algorithm. It combines the features extracted from both character-level and word-level embeddings, forming a dual-channel architecture. Additionally, self-attention is integrated to improve accuracy.

The Sac-BiLSTM algorithm incorporates the ability of BiLSTM to capture long-range dependencies during the feature extraction process. It also leverages CNN and dilated convolutional networks to extract local features. By assigning corresponding weight matrices to convolutional kernels of different granularities, the issue of refining granularity is resolved. Furthermore, a module combining a self-attention mechanism and pooling is added after the convolutional layer, allowing for the reassignment of weights to the post-convolutional features. This enables the construction of a deep learning algorithm suitable for sentiment classification of Chinese products, based on the characteristics of Chinese product reviews.

In addition to the convolutional neural network, the Sac-BiLSTM algorithm utilizes a concatenated bidirectional long short-term memory network (BiLSTM) to analyze contextual information. It further employs non-static character and word embeddings as two separate channels for training, enabling the algorithm to better capture word semantics.

Chapter 2 of the paper elaborates on the materials used by the proposed method, Chapter 3 presents details of the experimental process and results, Chapter 4 discusses the obtained experimental results, and Chapter 5 presents conclusions based on experimental findings.

The contributions of this paper are as follows: i. Proposing a dual-channel network structure that combines Self-Attention with BiLSTM, considering the syntactic structure and semantic relevance of the given sentence. Specifically, we integrate the parallel networks through an interactive Text eigenvector. ii. Designing two learning networks: a character-level network and a word-level network. The character-level network encourages the learning of deep semantic information and local relationships in sentences, while the word vector matrix network learns global contextual features and information. iii. Conducting extensive experiments on the onlineshopping10cats standard dataset uploaded to ChineseNlpCorpus in 2018 to validate the effectiveness of the proposed method.

2. Materials and Methods

2.1. General Framework

The Sac-BiLSTM algorithm proposed in this paper for text sentiment analysis is illustrated in Figure 1. The algorithm consists of three main parts.

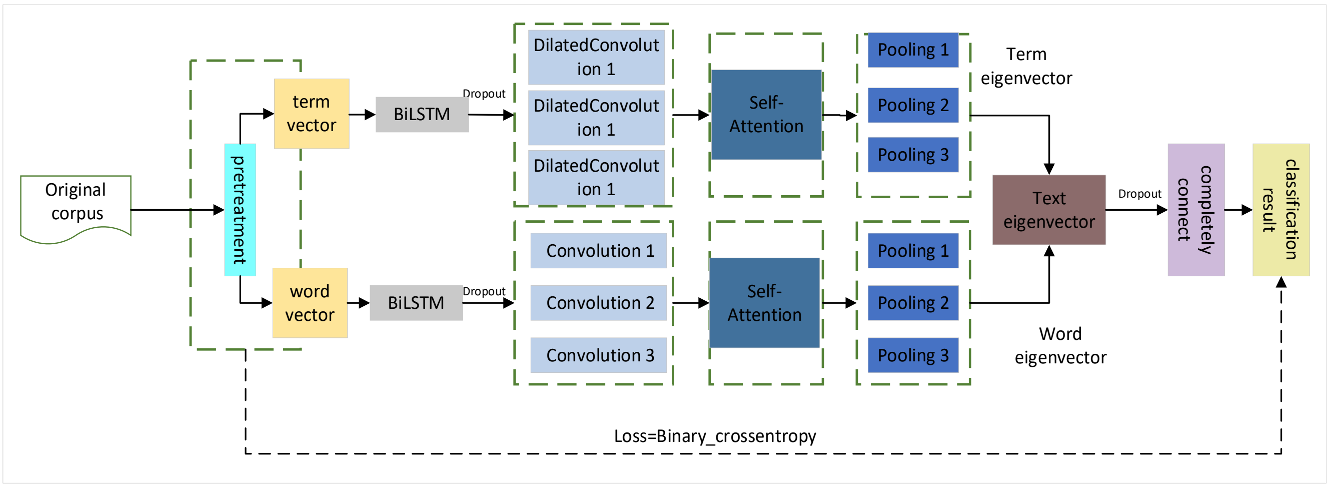

Figure 1. Sac-BiLSTM architecture.

Firstly, the tokenizer utilizes Word2Vec for preprocessing the dataset. The hierarchical classifier feature of Word2Vec is employed to separately train word embeddings and character embeddings.

Secondly, considering the characteristics of Chinese text, the algorithm treats word embeddings and character embeddings as two distinct channels. This allows for the extraction of data features generated in different contexts. As a result, word embeddings capture semantic nuances more precisely, while character embeddings reflect the fundamental characteristics of the text.

Thirdly, the algorithm combines BiLSTM with CNNs of various kernel sizes and DilatedConvolution. This encourages the character-level network to learn deep semantic information and local relationships within sentences, while the word embedding matrix network captures global contextual features and information. Subsequently, the Self-Attention mechanism is integrated into BiLSTM as a mapper for contextual semantic structure, while CNN serves as a feature extractor for semantic information. Self-Attention is used as a weight allocator to discover deep semantic information within sentences.

The sequential order of the BiLSTM and CNN modules in the model is a crucial consideration. Placing the BiLSTM module before the CNN module allows for the exploitation of the BiLSTM's ability to capture global context information. By leveraging the bidirectional nature of the BiLSTM, it can effectively learn long-term dependencies and extract contextual information within the text sequence. This is particularly advantageous for handling lengthy texts that contain rich contextual information and semantic associations. In contrast, the CNN module is primarily designed to extract local features and patterns. Due to the limited receptive field of convolutional kernels, CNN may struggle to capture the entire global context of lengthy texts. It may not fully consider the global semantic information that is crucial for understanding the overall structure and meaning of the text. Considering the specific characteristics of the dataset used in this study, which consists of lengthy texts exceeding 1900 characters, the decision to place the BiLSTM module before the CNN module is justified. This configuration ensures that the global context information is adequately captured by the BiLSTM, while the CNN module can focus on extracting local features and patterns within the context provided by the BiLSTM.

2.2. Specific Structure

The Sac-BiLSTM algorithm proposed in this paper consists of six specific layers.

(1) Embedding Layer: The data were loaded using the pandas library and stopwords were eliminated based on a predefined list. Tokenization was performed on the data, resulting in word-level and character-level representations. In the word-level tokenization, words associated with personal names, place names, organization names, and other proper nouns were excluded from the tokenization output. Lastly, the word2vec.Word2Vec function was utilized to create and train both word embedding and character embedding models.The pretrained word embeddings and character embeddings from Word2Vec were used as the initial weights for the embedding layer. This allowed the embedding layer the loading of the pretrained embedding models and representation of the text in vector form. In the pretrained models, Word2Vec was used to obtain word vector representations, while a character embedding model was used to obtain character vector representations. By using these pretrained word and character embeddings as the initial weights of the embedding layer, the network possesses some level of semantic understanding during the initial training phase. This approach improves the performance and generalization ability of the model while reducing training time and data requirements. The preprocessed dataset after this stage is shown in Figure 2 (where larger characters or words represent more frequent parts in the dataset).

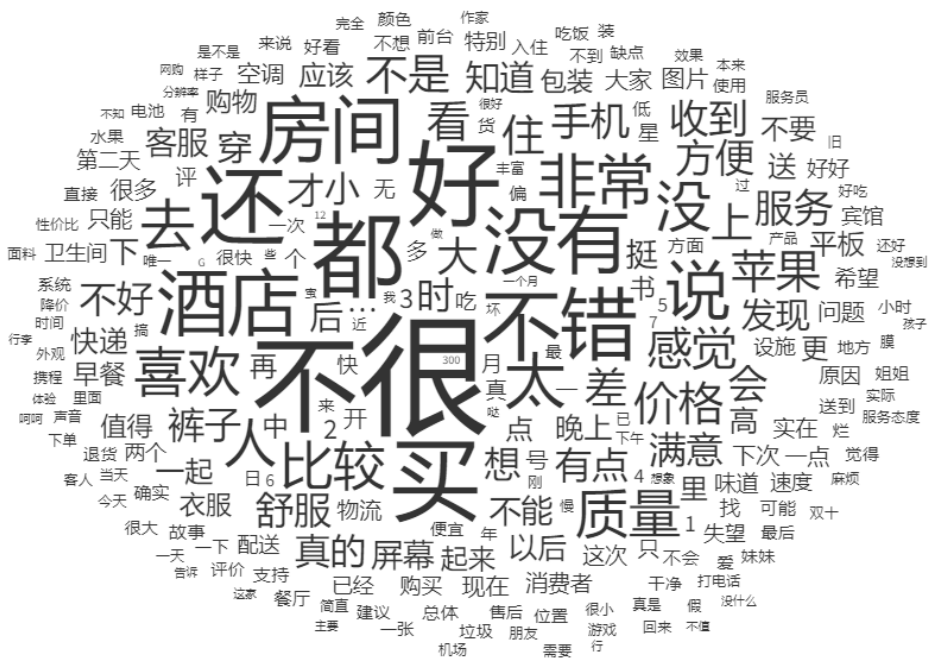

Figure 2. Product Reviews Word Cloud Display of Segmented Text.

(2) BiLSTM layer. For the feature vectors in the training corpus, they are first encoded through the BiLSTM layer. In other words, for each word (or character) in the product reviews, the BiLSTM layer utilizes the current input and its previous hidden state to calculate the next hidden state. This enables each time step's output to incorporate not only the feature vector information of the current vocabulary but also the feature information of all preceding and succeeding vocabularies. Consequently, the feature vectors at each time step can contain more comprehensive information. The structure of the BiLSTM layer is illustrated in Figure 3.

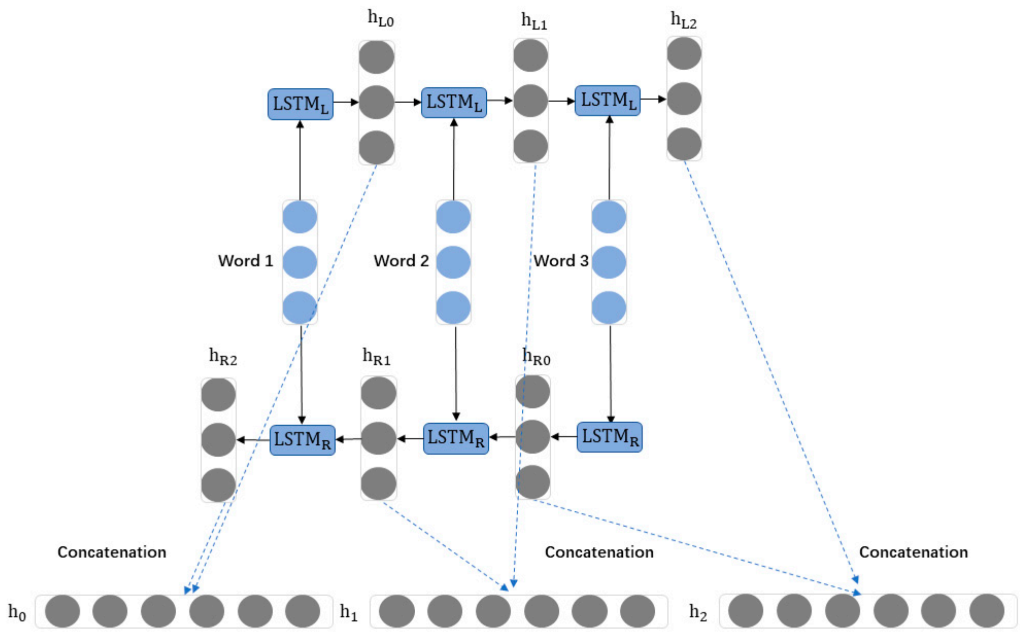

Figure 3. BiLSTM layer diagram.

The vectors obtained from the forward LSTM (h_{L0}) and the backward LSTM (h_{R0}) are merged and input into the subsequent CNN layer. (3) Convolutional Layer. In a CNN, a convolutional kernel function can generate a feature sequence. By using convolutional operations with kernels of different sizes on input word and character vectors, more feature sequences can be generated to capture a more comprehensive context of textual information. Assuming h is the kernel size, b is the bias term, and W_h is the weight matrix for different granularities of convolutional kernels, $W_h \in R^{(h\times k)}$. In this article, one-dimensional convolution operation with stride s = 1 is used, and weight matrix W_h is dot multiplied with the feature matrix composed of word vectors to obtain a sequence of features consisting of $n - h + 1$ outputs. We let the output results obtained by different convolution kernels be $C_{(h,i)}$, that is, the i output of the convolution kernel with kernel size h is calculated by the following Formula (1):

$$C_{h,i} = f(W_h X_{i:i+h-1} + b). \tag{1}$$

In the word vector channel, the formula for using dilated convolution is as follows (2), where $C_{h,i}$ is the receptive field size of this layer, k_n is the kernel size of this layer s_i, and is the stride of the ith layer:

$$C_{h,i} = C_{h-1,i} + (k_n - 1)\prod_{n=1}^{n-1} s_i. \tag{2}$$

As the convolution kernel moves, we obtain a lot of feature sequences $C_{(h,i)}$. The final generated feature graph set is C. The calculation formula is as follows (3):

$$C = C_{(h,1)}, C_{(h,2)}, \cdots, C_{(h,n+h-1)}. \tag{3}$$

To extract higher-level textual features, it is necessary to stack more convolutional layers. By stacking multiple convolutional layers, the network can gradually learn higher-level abstract features. Each convolutional layer can capture features of different scales and complexities.

After performing the convolution or dilated convolution operations, the resulting outputs are inputted into the Self-Attention layer. The outputs from different numbers of convolutional filters are all fed into this layer. As shown in Figure 4, input $C_{(h,i)}$ and its neighboring contexts are stored in key-value pairs Q(K,V), and the calculation formula is shown as Equation (4).

$$Attention(C_i, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{4}$$

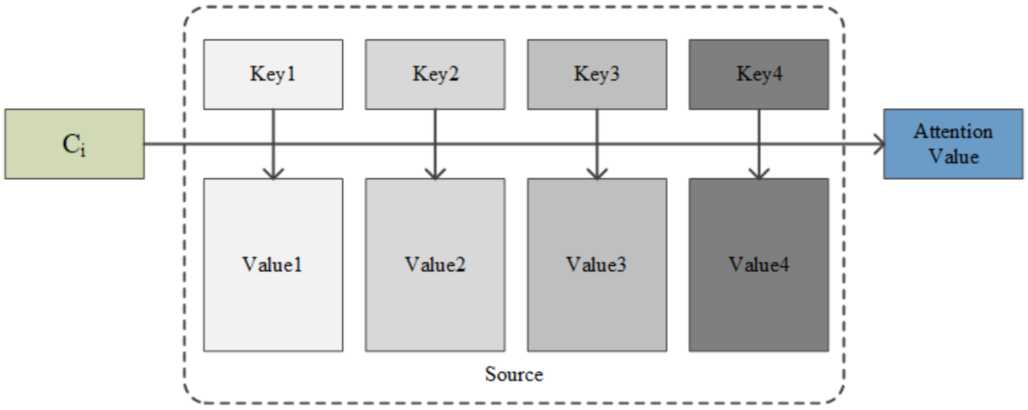

Figure 4. Schematic diagram of the self-attention mechanism.

In order to improve the speed of training convergence, the activation function of Relu is adopted. The Relu activation function can make the output value of some neurons be zero, which makes the network sparse, thus reducing the interdependence between parameters and avoiding overfitting.

(4) Pooling layer. Pooling operation is added after each convolution or dilated convolution in order to generate feature vectors with fixed dimensions and reduce the feature dimension while retaining the original feature information. Max pooling is adopted to calculate convolution kernels of different sizes and extract optimal features. Assuming ${C_{(h,j)}}^{(1)}$ is Maximum pooling of different convolution kernels (as shown in Formula (5)), the result is a feature sequence.

$${C_{(h,i)}}^{(1)} = max(C_{(h,i)}). \tag{5}$$

Assuming that the number of convolution nuclei in each group is m, the final feature set output after pooling is shown in Equation (6).

$$C^{(2)} = flatten({C_{(h,1)}}^{(1)}, {C_{(h,2)}}^{(1)}, \cdots, {C_{(h,m)}}^{(1)}). \tag{6}$$

(5) Merge layer. Assuming D_1 and D_2 represent the word vector channel and character vector channel, respectively, the feature sequences extracted from both channels are combined to obtain global information, forming the final set of text vectors $C^{(3)}$ shown in Equation (7).

$$C^{(3)} = {C_{(D_1)}}^{(2)} \oplus {C_{(D_1)}}^{(2)}. \tag{7}$$

To avoid overfitting, regular optimization is applied to the merged text vectors using the Dropout optimization strategy,with a dropout rate of 0.35, i.e., 35% of the trained parameters are randomly dropped out each iteration.

(6) Fully connected layer. After convolution, pooling, and merging, the text vector collection is passed through a fully connected layer to obtain sample classification. At the same time, binary cross-entropy loss is used as the loss function to optimize the model by minimizing the cross-entropy between the predicted values of sentences in the training samples and the actual values. This completes the task of sentiment analysis. The larger the error, the larger the gradient of the parameters, which allows for faster convergence.

2.3. Experimental Data

The standard dataset used in this paper for e-commerce shopping comments is the online_shopping_10_cats dataset uploaded in 2018 to the ChineseNlpCorpus. This dataset is divided into 10 categories (books, tablets, mobile phones, fruits, shampoo, water heaters, Mengniu dairy products, clothing, computers, and hotels), with a total of 62,772 comment data, including 31,351 positive comments and 31,421 negative comments. The e-commerce comment labels are divided into two categories, 1 and 0, where label 1 represents positive comments and label 0 represents negative comments. The specific data distribution of the dataset is shown in Table 1. In this paper, the training and testing corpora were selected with proportions of 90% and 10%, respectively. We divide 20% of the training set into validation sets.

Table 1. Dataset data distribution.

Classes	Positive Comment Number	Negative Comment Number
Book	2100	1751
Pad	5000	5000
Phone	1163	1158
Fruits	5000	5000
Shampoo	5000	5000
Water Heater	100	475
Mengniu Dairy	992	1041
Clothes	5000	5000
Computer	1996	1996
Hotel	5000	5000

In addition, to validate the effectiveness and robustness of the proposed method, comparative experiments are conducted on two additional datasets: a Chinese food delivery dataset with a total of 12,000 samples (4000 positive samples and 8000 negative samples), and a Weibo comment dataset with a total of 119,988 samples (59,993 positive samples and 59,995 negative samples).

3. Experiments and Results

3.1. Computing Environment Configuration

The experiments in this paper were conducted based on the deep learning framework TensorFlow using the word segmentation tool jieba and the feature vector training tool GOOGLE Gensim Word2Vec 3.8.3. The Sac-BiLSTM algorithm was built using Python language. The experimental environment and settings are shown in Table 2.

Table 2. Experimental environment and configuration.

Experimental Environment	Environment Configuration
Operating System	Win10
CPU	i5-7300HQ CPU @ 2.50 GHz
Memory	8 GB
Deep Learning Framework	TensorFlow 2.1.0-cpu
Programming Language	Python 3.7
Word Segmentation Tool	jieba
Feature Vector Training Tool	Word2Vec (gensim 3.8.3)
Programming Environment	Anaconda 3

3.2. Evaluation Index

In this paper, experimental evaluation metrics including Accuracy, Precision, Recall, and F1 score are employed, which are calculated based on the confusion matrix presented in Table 3.

Table 3. Classification confusion matrix.

Real Class	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

In Table 3, TP and TN, respectively, represent the number of positive and negative samples predicted correctly.

Accuracy *Acc* represents the proportion between the predicted samples conforming to the label and the total samples, as shown in Equation (8).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{8}$$

Recall rate *Recall* represents the proportion of positive samples with correct prediction in all positive samples, as shown in Equation (9).

$$PRecall = \frac{TP}{TP + FN}. \tag{9}$$

Precision *Precision* represents the probability that all predicted positive samples are actually positive samples, as shown in Equation (10).

$$Precision = \frac{TP}{TP + FP}. \tag{10}$$

Value $F1$ score $F1$ represents the final classification effect obtained by weighted harmonic calculation of accuracy rate $Precision(P)$ and recall rate $Recall(R)$, as shown in Equation (11).

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2 \cdot \frac{P \cdot R}{P + R}. \tag{11}$$

Among them, $P = \frac{TP}{TP+FP}$.

3.3. Data Preprocessing

For preprocessing the existing textual data, comment sentences are segmented into words and stop words are removed using the Harbin Institute of Technology stop word list. Based on the given corpus, Word2Vec can represent a word as a vector. In this paper, the skip-gram model is used to establish the word embedding model. We suppose the dimension of the word vector (or characters vector) is k. A sentence consists of n words (or characters). If the word vector (or characters vector) of the ith word is n, the sentence can be described as X_i as shown in Equation (12):

$$X_{1:n} = X_1 \oplus X_2 \oplus \cdots \oplus X_n, \tag{12}$$

where $\oplus$ represents the connection operator.

At the same time, through the string slicing function, a word function for word segmentation is established, and then Word2Vec is used to convert it to a character vector. Similar to the word vector, if the dimension of the characters vector is k, a sentence consists of n characters, and the character vector of the first character is Y_i; then, the sentence can be described as shown in Equation (13):

$$Y_{1:n} = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_n. \tag{13}$$

3.4. Algorithm Parameter Selection

(1) Feature vector training.

The datasets after word segmentation and word segmentation are used as word vector and word vector training corpus, respectively. The Word2Vec tool in the Gensim package of Python language is used for training, and the word vector and word vector models are constructed. The Word2Vec parameter settings are shown in Table 4.

Table 4. Word2Vec parameter Settings.

Parameter	Character	Word
sg	Skip-gram	Skip-gram
size	300	300
min_count	3	3
window	10	10
workers	4	4

(2) Algorithm parameter setting.

The number of LSTM units in the BiLSTM network needs to be determined. The number of neurons determines the feature extraction capability of the layer, increasing the number of neurons that can retain more feature information. However, an excessive number of neurons increases the computational time of the algorithm and has limited effects on improving the results. Therefore, this paper sets the number of LSTM units to 20 so as to balance the time cost while retaining as much feature information as possible.

The main parameters of the CNN include: (1) kernel size, which is the size of the area for convolving the input feature vector; (2) the number of convolutional kernels; (3) dropout ratio, which is the proportion of trained parameters that are randomly dropped out; (4) batch_size of the stochastic gradient descent algorithm; (5) number of iterations; (6) optimization function; (7) Self-Attention output dimension.

The Sac-BiLSTM uses kernel sizes of three, five and seven. The more convolutional kernels used, the more features can be extracted by the layer. To enhance computational efficiency, the number of convolutional kernels is usually set to 2^n, where n is a positive integer, so the number of convolutional kernels is set to 128. Generally, the larger the batch_size, the more accurate the determined descent direction, and the smaller the training oscillations. However, more epochs are required to achieve the same accuracy. Thus, to achieve the optimal balance between time and convergence accuracy, the batch_size is set to 64. In addition, the dimensionality of Self-Attention is set to maintain the same dimensionality in order to redistribute weights for previous outputs without losing useful information.

To explore deeper data information, the proposed Sac-BiLSTM algorithm adopts a stacked form of multiple convolutional layers. The length of the text ranges from a minimum of 10 characters to a maximum of 1900 characters. Therefore, the use of three convolutional layers can more accurately extract features. The sizes of the convolutional filters are set to three, five, and seven. The more convolutional filters, the more features the layer extracts, and the number of convolutional filters is generally set as 2^n, where n is a positive integer. Hence, the number of convolutional filters is set to 128.

In general, a larger batch size leads to a more accurate descent direction, resulting in smaller training oscillations. However, it requires more epochs to achieve the same accuracy. To achieve the best balance between time and convergence accuracy, the batch size is set to 64. Additionally, the dimension of the Self-Attention layer is set to be the same as the previous layer to redistribute weights and avoid losing valuable information.

After 15 training iterations, the accuracy on the validation set reaches its maximum value and stabilizes in subsequent training. Therefore, the number of iterations is set to 15. Compared to traditional gradient descent algorithms, the Adam algorithm has high computational efficiency and requires less memory. It also prevents excessive learning step size through bias correction terms. Thus, the final parameter values are determined as shown in Table 5.

Table 5. Algorithm parameter Settings.

Parameter	Value
BiLSTM units	20
Convolution kernel size	3, 5, 7
Number of convolution kernels	128
Self_Attention Output dimension	128
Dropout	0.35
Batch_size	64
Iterations	15
Optimization Function	Adam

3.5. Hyperparameter Setting

The gradient descent method and the Adam optimizer are used for model training. Employing a model classifier using a sigmoid classifier, final output emotional categories of predicted $\hat{y}$, $\hat{y}$ are defined as shown in Equation (14):

$$\hat{y}^{(i)} = \frac{1}{1 + e^{-({w_i}^T + C + b_i)}}. \tag{14}$$

Here, $\hat{y}^{(i)}$ represents the prediction value of the ith sample, and w_i and b_i represent the weight and bias to be trained. During model training, if the error between the predicted value and the actual value is large, the magnitude of various parameter adjustments in the backpropagation training process will be larger, thus making the training converge more quickly. Conversely, if the error between the predicted and actual values is small, the magnitude of various parameter adjustments must be smaller to reduce oscillations.

The binary cross-entropy loss function is employed as the chosen loss function to optimize the model for sentiment analysis. This is achieved by minimizing the cross-entropy between the predicted sentence values and the true values in the training samples. A larger error leads to a larger gradient of the parameters, allowing for faster convergence. The specific formulation of the cross-entropy loss function can be observed in Equation (15):

$$Loss = -\sum_x [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]. \tag{15}$$

3.6. Experimental Results

This study conducted experiments using the online_shopping_10_cats dataset from the ChineseNlpCorpus, which was uploaded in 2018, among other datasets.The proposed Sac-BiLSTM algorithm achieved a stable accuracy of 0.9463 on the test set, and it quickly converged to an accuracy above 0.94 within the first five epochs on the validation set. The final values for Recall, Precision, and F1 score were 0.9409, 0.9369, and 0.9404, respectively. The accuracy variation during training on the validation set is shown in Figure 5.

In the testing sets of the food delivery dataset and the Weibo comments dataset, the accuracy obtained was 87.7652 and 97.7588, respectively. The accuracy variation curves on the validation set are shown in Figures 6 and 7.

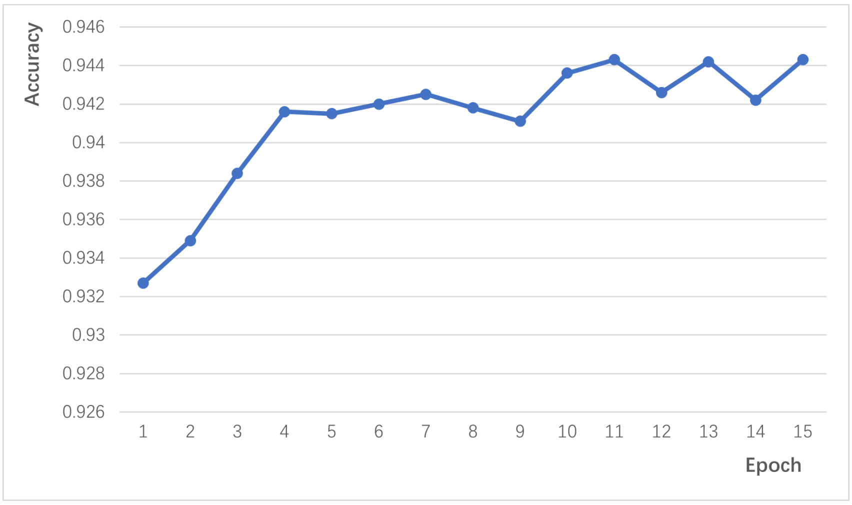

Figure 5. Accuracy Variation Curve of Sac-BiLSTM on the Validation Set of the Online Shopping Review Dataset.

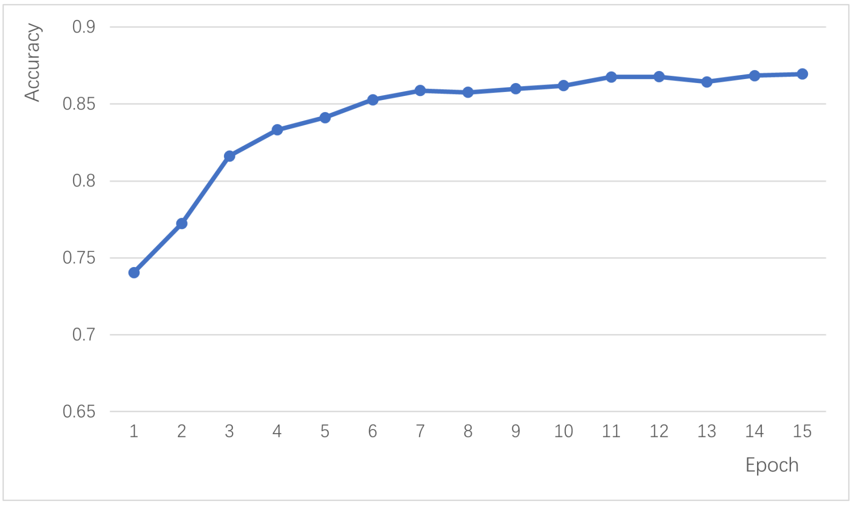

Figure 6. Accuracy Variation Curve of Sac-BiLSTM on the Validation Set of the Food Delivery Review Dataset.

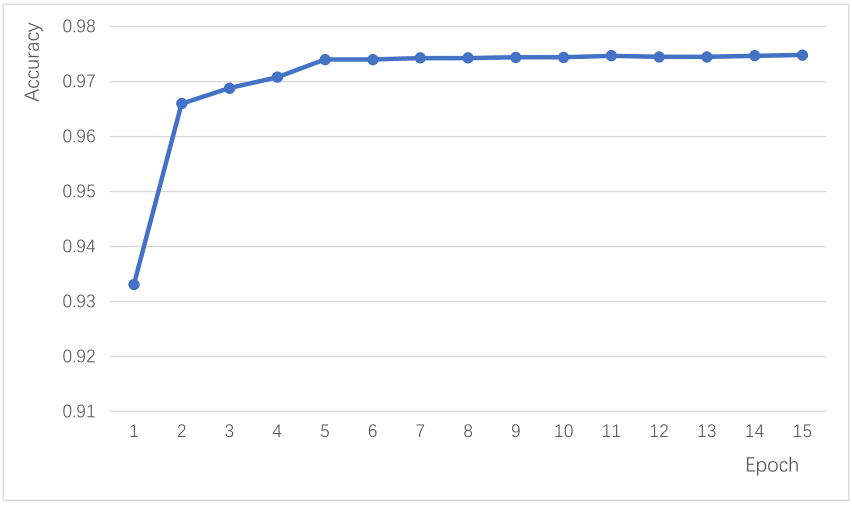

Figure 7. Accuracy Variation Curve of Sac-BiLSTM on the Validation Set of the Weibo Comments Dataset.

4. Discussion

4.1. Comparison and Analysis with Other Methods

The proposed Sac-BiLSTM algorithm in this study achieves efficient text sentiment analysis, with the model rapidly converging within the first five iterations and maintaining a classification accuracy above 0.94. To validate the effectiveness and superiority of the proposed algorithm, it is compared with other mainstream text classification algorithms, including Static-CNN, Non-Static-CNN, S-Non-CNN (Multi-channel Convolutional Neural Network), CNN-character (Character-level Convolutional Neural Network), DCCNN (Dual-channel Convolutional Neural Network), BiLSTM-CNN Concatenation, and BiLSTM-CNN Parallel.

In this section, we provide a brief introduction to different comparative models, all of which use the Word2Vec model as their pre-trained model.

1. WordCNN-static: This model uses pre-trained word embeddings as the embedding layer. Multiple one-dimensional convolutions with different sizes are applied to the output of the embedding layer, followed by batch normalization, activation, and pooling. The outputs of the pooling layers are concatenated, flattened, and passed through a Dropout layer and a fully connected layer for classification.

2. CNN-character: This model uses pre-trained character embeddings as the embedding layer. Different-sized convolutional kernels are applied to the input, followed by Max pooling to reduce the dimensionality of each convolutional kernel's output, resulting in a set of feature vectors. The pooling results of all convolutional kernels are then concatenated and further processed by a fully connected layer for feature extraction and classification.

3. S-Non-CNN: This model defines two different word embedding layers that load pre-trained word embeddings. Next, different-sized convolutional kernels are applied to each channel independently, followed by non-linear transformations. The outputs of each convolutional kernel are then reduced in dimensionality through Max pooling, resulting in two sets of feature vectors. These feature vectors are flattened and concatenated to form a global feature vector. The global feature vector is further processed by a fully connected layer for feature extraction and classification.

4. DCCNN: This model defines two different word embedding layers that load different pre-trained word and character embeddings. Different-sized convolutional kernels are applied to the input, followed by non-linear transformations and dimensionality reduction, resulting in two sets of feature vectors. These feature vectors are flattened and concatenated to form a global feature vector. The global feature vector is further processed by a fully connected layer for feature extraction and classification.

5. BiLSTM-CNN Concatenation: This model loads pre-trained word embeddings and applies a BiLSTM layer for sequence modeling. Then, different-sized convolutional kernels are applied to the output of the BiLSTM layer, followed by non-linear transformations and dimensionality reduction, resulting in a set of feature vectors. These feature vectors are flattened and further processed by a fully connected layer for feature extraction and classification.

6. BiLSTM-CNN Parallel: This model uses two input channels. The model applies a BiLSTM layer for sequence modeling on inputA, resulting in a sequence output (outputA). Simultaneously, different-sized convolutional kernels (three, five, and seven) are applied to inputB, followed by non-linear transformations and dimensionality reduction, resulting in a set of feature vectors (outputB). These feature vectors are flattened and merged with outputA, resulting in a comprehensive feature vector. The comprehensive feature vector is further processed by a fully connected layer for feature extraction and classification.

The text classification results of this algorithm compared to the other six algorithms on various test datasets are shown in Tables 6–8.

Table 6. Text classification results of different algorithms on the Online Shopping Review dataset test set.

Algorithm	Feature	Acc	Precision	Recall	F1
Static-CNN	Static Single-channel	0.9188	0.9007	0.9390	0.9195
CNN-character	Static Single-channel	0.9209	0.9006	0.9442	0.9219
S-Non-CNN	Non-static Dual-channel	0.9336	0.9449	0.9190	0.9318
DCCNN	Non-static Dual-channel	0.9372	0.9376	0.9352	0.9364
BiLSTM-CNN series	Non-static Single-channel	0.9351	0.9408	0.9271	0.9339
BiLSTM-CNN parallel	Non-static Dual-channel	0.9350	0.9413	0.9262	0.9337
Sac-BiLSTM	Non-static Dual-channel	0.9463	0.9469	0.9409	0.9464

Table 7. Text classification results of different algorithms on the food delivery review dataset test set.

Algorithm	Feature	Acc	Precision	Recall	F1
Static-CNN	Static Single-channel	0.8376	0.8602	0.8432	0.8560
CNN-character	Static Single-channel	0.8464	0.8457	0.8731	0.8592
S-Non-CNN	Non-static Dual-channel	0.8494	0.8221	0.8930	0.8339
DCCNN	Non-static Dual-channel	0.8564	0.8021	0.8477	0.8688
BiLSTM-CNN series	Non-static Single-channel	0.8539	0.8057	0.8079	0.8538
BiLSTM-CNN parallel	Non-static Dual-channel	0.8551	0.8341	0.8880	0.8602
Sac-BiLSTM	Non-static Dual-channel	0.8776	0.8636	0.8980	0.8804

Table 8. Text classification results of different algorithms on the Weibo comment dataset test set.

Algorithm	Feature	Acc	Precision	Recall	F1
Static-CNN	Static Single-channel	0.9261	0.9026	0.9492	0.9253
CNN-character	Static Single-channel	0.9443	0.9389	0.9276	0.9130
S-Non-CNN	Non-static Dual-channel	0.9508	0.8627	0.9494	0.9129
DCCNN	Non-static Dual-channel	0.9566	0.9329	0.9587	0.9556
BiLSTM-CNN series	Non-static Single-channel	0.9448	0.9332	0.9538	0.9434
BiLSTM-CNN parallel	Non-static Dual-channel	0.9268	0.9031	0.9501	0.9260
Sac-BiLSTM	Non-static Dual-channel	0.9775	0.9940	0.9592	0.9763

By comparing the results of various methods in Tables 6–8, it can be observed that the proposed Sac-BiLSTM algorithm outperforms the other six methods in all performance metrics.

Analyzing the results of the comparative experiments, we can understand the impact of different feature fusion and algorithm combinations on model performance in sentiment analysis tasks. From the results in Tables 6–8, the following can be concluded.

(1) Compared to Static-CNN, the Character-level Convolutional Neural Network (CNN-character) utilizes pre-trained character embeddings as input, validating that using character embeddings as the original features is superior to using word embeddings.

(2) The multi-channel algorithm can improve the accuracy of the model to some extent. The S-Non-CNN multi-channel algorithm trains with both static and non-static word embeddings in two separate channels, resulting in higher Accuracy, Recall, and F1 score values compared to single-channel algorithms.

(3) BiLSTM can improve the accuracy of the model. Adding a BiLSTM layer in the network architecture allows each time step's feature vector more information. Experimental results show that adding BiLSTM to the convolutional neural network can achieve higher accuracy. Comparing the effects of BiLSTM-CNN concatenation and BiLSTM-CNN parallelization in sentiment classification, it is found that adding BiLSTM in series with CNN performs better than adding them in parallel. Additionally, the dual-channel algorithm using character/word embeddings can better utilize the ability of BiLSTM to extract global feature vectors, resulting in better performance compared to that of algorithms that combine BiLSTM and CNN in series or in parallel.

The proposed Sac-BiLSTM introduces a novel dual-channel architecture, where features are separately extracted from character and word vectors and then fused before classification, allowing the model the learning of semantic features of different sizes. Furthermore, the proposed method incorporates Self-Attention units into the dual-channel network structure, making it easier for the network to capture long-distance dependent features in sentences. The experimental results on the online_shopping_10_cats dataset demonstrate that the Sac-BiLSTM achieves an Accuracy of 0.9463, Recall of 0.9409, Precision of 0.9469, and F1-score of 0.9404, which are the highest among all the algorithms.In the food delivery dataset, the Accuracy is 0.8776, Precision is 0.86363, Recall is 0.8980, and the F1-score is 0.8804. In the Weibo comment dataset, the Accuracy is 0.9775, Precision is 0.9940, Recall is 0.9592, and the F1-score is 0.9763. These metrics indicate that the Sac-BiLSTM model performs the best among all the models. The experimental results show that the performance of the model is influenced by the size of the data, with better results observed with larger datasets. Moreover, the model demonstrates good robustness. In the imbalanced classification task for water heater reviews, the Sac-BiLSTM model achieves an Accuracy of 94.9874, matching the results of the test set. Additionally, in the imbalanced food delivery comment dataset, the Sac-BiLSTM model still outperforms other models.

The accuracy variations of different algorithms on the validation set of the product review dataset at different iteration numbers are shown in Figure 8. From Figure 8, it can be observed that the static single-channel Static-CNN algorithm has the lowest accuracy and significant fluctuations. Other methods that optimize using non-static and multi-channel approaches show improved accuracy and stability. The proposed Sac-BiLSTM algorithm achieves significantly higher accuracy than other algorithms, with smaller fluctuations. It rapidly converges within the first five iterations and reaches a stable accuracy level after the tenth iteration.

Furthermore, the proposed algorithm introduces a novel dual-channel structure that fully utilizes the feature extraction capabilities of CNN combined with dilated convolutions. This allows the algorithm stabilization at a high level within a small number of iterations. Additionally, the Sac-BiLSTM incorporates Self-Attention units in the BiLSTM module, enabling the model to weight the importance of different words and better capture key information for sentiment expression. Compared to a regular BiLSTM, this further improves accuracy. Additionally, the accuracy of Sac-BiLSTM is higher than that of BiLSTM-CNN (series and parallel) in different categories. In the tablet and fruit categories, the accuracy of Sac-BiLSTM reaches 88.5293 and 87.3329, respectively. On the other hand, the accuracy of the BiLSTM-CNN (series) is 84.6752 and 83.9825, and the accuracy of the BiLSTM-CNN (parallel) is 85.6842 and 86.5796.

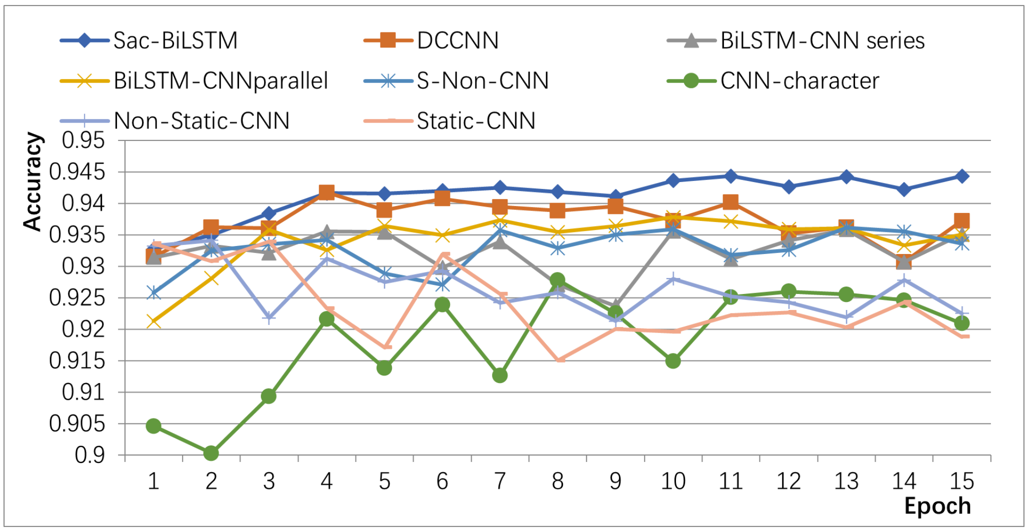

Figure 8. Accuracy Comparison of Different Algorithms on the Test Set of the Online Shopping Review Dataset at Different Iterations.

4.2. Feature Analysis of the Proposed Dual-Channel Structure

In this paper, a novel dual-channel structure, BiLSTM-Dilated Convolution combined with CNN, is designed for text sentiment analysis. It was found that when the BiLSTM module is connected to CNN through a single channel, it fails to understand deep semantic correlations and faces difficulties in simultaneously capturing local and global information. Consequently, the accuracy decreases compared to using only CNN, as shown in Table 6, where both the serial and parallel BiLSTM-CNN methods have lower accuracy than DCCNN.

The DCCNN model, in the aspect of text sentiment analysis, computes faster than current mainstream text sentiment classification methods and can achieve high accuracy in a very short number of iterations. This ability to quickly capture features makes it more effective in extracting and modeling local features. However, for global models, it may overlook longer dependency relationships. On the other hand, algorithms like BiLSTM-CNN serial and BiLSTM-CNN parallel can learn deeper contextual information in text sentiment analysis, enabling simultaneous modeling of local features and global context information, resulting in higher precision values. However, these methods struggle with feature fusion and suffer from reduced accuracy due to repeated features.

The proposed dual-channel approach effectively addresses these issues. In this study, a new dual-channel connection method is designed to connect the BiLSTM module separately with CNN and Dilated Convolution. This involves setting up channels for word vectors and character vectors, combining word embedding channels with dilated convolutions, and character embedding channels with convolutions. The word embedding channel captures word-level global information and semantic associations using dilated convolutions, capturing a broader range of contextual information. Simultaneously, the character embedding channel utilizes convolutions to capture local information and semantic correlations between adjacent characters, learning features from different window sizes. The combination of features from both channels provides a richer representation, effectively addressing the problem of understanding deep semantic correlations.

Experimental verification shows that when the convolutional kernel sizes are set to three, five, and seven with a quantity of 128, the Sac-BiLSTM algorithm achieves the best performance.

In conclusion, the proposed Sac-BiLSTM algorithm accurately extracts features from Chinese product review corpus for sentiment analysis. This algorithm utilizes fine-grained character vectors to assist in extracting semantic information from word vectors, constructs a non-static dual-channel, and combines long short-term memory models with convolutional neural networks and dilated convolutions to avoid semantic extraction biases. It effectively extracts feature information from text, thereby improving the accuracy of sentiment analysis. Additionally, the Sac-BiLSTM algorithm proposed in this paper incorporates the Self-Attention mechanism to better capture key information in the text, enhancing the model's understanding of text relationships and contextual information, further improving the accuracy of sentiment analysis. Experimental results demonstrate the higher precision of the proposed Sac-BiLSTM algorithm.

4.3. Shortcomings and Prospects

Currently, there are still directions for improvement in algorithms to further enhance their performance and application scope.

Firstly, the interpretability of the algorithm can be improved. Although the Sac-BiLSTM algorithm performs well in terms of accuracy, its internal decision-making process is not easily understood. Further research can explore methods such as visualization to make the algorithm's decision-making process more transparent and interpretable to users, thereby enhancing trust in the algorithm.

In addition, efficiency and real-time performance are also important aspects to consider. Despite achieving good accuracy, the Sac-BiLSTM algorithm has a high computational complexity and requires longer processing time. To meet the demands of real-time applications, further optimization of the algorithm's structure and parameters can improve its running efficiency.

Lastly, the impact of data quality and scale on the algorithm should also be emphasized. The performance of the algorithm largely depends on the quality and scale of the training data. Future research can focus on constructing larger-scale, high-quality sentiment-annotated datasets to further enhance the algorithm's performance and generalization ability.

In summary, looking ahead, the Sac-BiLSTM algorithm has broad application prospects in the field of text sentiment analysis. By improving the algorithm's interpretability, generalization ability, efficiency, and real-time performance, as well as focusing on data quality and scale, the algorithm's performance and applicability can be further enhanced, promoting its application in areas such as business analytics and user experience improvement.

5. Conclusions

To improve the accuracy of text sentiment analysis and address issues related to inaccurate extraction of key information and limited contextual associations in text, this paper proposes the Sac-BiLSTM algorithm, which effectively utilizes the characteristics of Chinese product review corpus. The Sac-BiLSTM algorithm leverages fine-grained character vectors to assist in extracting semantic information from word vectors, constructs a non-static dual-channel, and combines long short-term memory models with convolutional neural networks to extract features from text more effectively. Additionally, the proposed Sac-BiLSTM algorithm incorporates the Self-Attention mechanism to enhance the extraction of textual features and address issues related to inaccurate extraction of key information and limited contextual associations in the text. Experimental results demonstrate that the proposed Sac-BiLSTM algorithm achieved accuracy rates of 94.63%, 87.76%, and 97.75% on different datasets, which were significantly higher than those of the other six methods used for comparison.

The Sac-BiLSTM algorithm proposed in this paper contributes to a more accurate understanding of user sentiments towards products, reduces business analysis costs for companies, increases user engagement, provides warnings about product quality, and offers assistance in improving product image and quality.

Author Contributions: Conceptualization, W.W. and Y.Y.; methodology, W.W.; validation, G.W. and Z.Z. (Zhemin Zhuang); formal analysis, G.W.; data curation, Z.Z. (Zikun Zheng); writing—original draft preparation, W.W.; writing—review and editing, Z.Z. (Zikun Zheng); funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Youth Fund the National Natural Science Foundation of China (61002047).

Data Availability Statement: The data set used is: https://github.com/SophonPlus/ChineseNlpCorpus/blob/master/datasets/weibo_senti_100k/intro.ipynb (accessed on 8 November 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Punetha, N.; Jain, G. Bayesian game model based unsupervised sentiment analysis of product reviews. *Expert Syst. Appl.* **2023**, *214*, 119128.
2. Edara, D.C.; Vanukuri, L.P.; Sistla, V.; Kolli, V.K.K. Sentiment analysis and text categorization of cancer medical records with LSTM. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 5309–5325.
3. Bhuvaneshwari, P.; Rao, A.N.; Robinson, Y.H.; Thippeswamy, M. Sentiment analysis for user reviews using Bi-LSTM self-attention based CNN model. *Multimed. Tools Appl.* **2022**, *81*, 12405–12419.
4. Luo, Y.; Yao, C.; Mo, Y.; Xie, B.; Yang, G.; Gui, H. A creative approach to understanding the hidden information within the business data using Deep Learning. *Inf. Process. Manag.* **2021**, *58*, 102615.
5. Liu, X.; Tang, T.; Ding, N. Social network sentiment classification method combined Chinese text syntax with graph convolutional neural network. *Egypt. Inform. J.* **2022**, *23*, 1–12.
6. Majumder, M.G.; Gupta, S.D.; Paul, J. Perceived usefulness of online customer reviews: A review mining approach using machine learning & exploratory data analysis. *J. Bus. Res.* **2022**, *150*, 147–164.
7. Jing, N.; Wu, Z.; Wang, H. A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction. *Expert Syst. Appl.* **2021**, *178*, 115019.
8. Obiedat, R.; Qaddoura, R.; Ala'M, A.Z.; Al-Qaisi, L.; Harfoushi, O.; Alrefai, M.; Faris, H. Sentiment analysis of customers' reviews using a hybrid evolutionary svm-based approach in an imbalanced data distribution. *IEEE Access* **2022**, *10*, 22260–22273.
9. Kewsuwun, N.; Kajornkasirat, S. A sentiment analysis model of agritech startup on Facebook comments using naive Bayes classifier. *Int. J. Electr. Comput. Eng.* **2022**, *12*, 2829–2838.
10. Dake, D.K.; Gyimah, E. Using sentiment analysis to evaluate qualitative students' responses. *Educ. Inf. Technol.* **2023**, *28*, 4629–4647.
11. Benarafa, H.; Benkhalifa, M.; Akhloufi, M. WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 3. [CrossRef]
12. Shamrat, F.M.J.M.; Chakraborty, S.; Imran, M.M.; Muna, J.N.; Billah, M.M.; Das, P.; Rahman, M.O. Sentiment analysis on twitter tweets about COVID-19 vaccines using NLP and supervised KNN classification algorithm. *Indones. J. Electr. Eng. Comput. Sci.* **2021**, *23*, 463–470.
13. Wawre, S.V.; Deshmukh, S.N. Sentiment classification using machine learning techniques. *Int. J. Sci. Res.* **2016**, *5*, 819–821.
14. Huq, M.R.; Ahmad, A.; Rahman, A. Sentiment analysis on Twitter data using KNN and SVM. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 19–25.
15. Jiang, W.; Zhou, K.; Xiong, C.; Du, G.; Ou, C.; Zhang, J. KSCB: A novel unsupervised method for text sentiment analysis. *Appl. Intell.* **2023**, *53*, 301–311.
16. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5999–6009.
18. Abdullah, T.; Ahmet, A. Deep learning in sentiment analysis: Recent architectures. *Acm Comput. Surv.* **2022**, *55*, 1–37.
19. Nguyen, H.D.; Huynh, T.; Hoang, S.N.; Pham, V.T.; Zelinka, I. Language-Oriented Sentiment Analysis Based on the Grammar Structure and Improved Self-attention Network. In Proceedings of the 15th International Conference, ENASE 2020, Prague, Czech Republic, 5–6 May 2020; pp. 339–346.
20. Gan, C.; Wang, L.; Zhang, Z. Multi-entity sentiment analysis using self-attention based hierarchical dilated convolutional neural network. *Future Gener. Comput. Syst.* **2020**, *112*, 116–125. [CrossRef]
21. Yan, S.; Wang, J.; Song, Z. Microblog Sentiment Analysis Based on Dynamic Character-Level and Word-Level Features and Multi-Head Self-Attention Pooling. *Future Internet* **2022**, *14*, 234. [CrossRef]
22. Kudo, T.; Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv* **2018**, arXiv:1808.06226.
23. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
24. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.
25. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A C-LSTM neural network for text classification. *arXiv* **2015**, arXiv:1511.08630.

26. Ghourabi, A.; Mahmood, M.A.; Alzubi, Q.M. A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages. *Future Internet* **2020**, *12*, 156. [CrossRef]
27. Gan, C.; Wang, L.; Zhang, Z.; Wang, Z. Sparse attention based separable dilated convolutional neural network for targeted sentiment analysis. *Knowl.-Based Syst.* **2020**, *188*, 104827. [CrossRef]
28. Liu, G.; Guo, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* **2019**, *337*, 325–338. [CrossRef]
29. Gan, C.; Feng, Q.; Zhang, Z. Scalable multi-channel dilated CNN–BiLSTM model with attention mechanism for Chinese textual sentiment analysis. *Future Gener. Comput. Syst.* **2021**, *118*, 297–309. [CrossRef]
30. Li, R.; Chen, H.; Feng, F.; Ma, Z.; Wang, X.; Hovy, E. Dual graph convolutional networks for aspect-based sentiment analysis. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Virtual, 1–6 August 2021; pp. 6319–6329.
31. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Pipalia, K.; Bhadja, R.; Shukla, M. Comparative Analysis of Different Transformer Based Architectures Used in Sentiment Analysis. In Proceedings of the 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 4–5 December 2020; pp. 411–415. [CrossRef]
33. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The Long-Document Transformer. *arXiv* **2020**, arXiv:cs.CL/2004.05150.