*Journal of Network Intelligence*
**Taiwan Ubiquitous Information**
 ISSN 2414-8105
Volume **10**, Number **2**, May **2025**

# Research on Chinese Text Sentiment Analysis Algorithm on Deep Learning

Yan-Hong Fu*

School of Humanities and Arts
Nanyang Vocational College of Agriculture, Nanyang 473000, P. R. China
fyh27122@163.com

Boris Chiong

St. Paul University Philippines, Tuguegarao 3500, Philippines
ea9388@163.com

*Corresponding author: Yan-Hong Fu

Received May 6, 2024; revised September 6, 2024; accepted December 23, 2024.

Abstract. *With the advancement of the computer and information technology revolution, the frequent activities of global Internet users in e-commerce and social news platforms produce massive unbalanced text data. People are eager to obtain useful information from these text data and apply it. Text sentiment analysis is one of the solutions. Text sentiment analysis methods are currently relatively mature and applied in multiple fields, such as e-commerce comment classification, news classification, and public opinion control, but there is still an issue of imbalanced distribution of data sample categories. In response to the above issues, this article studies the Chinese text sentiment analysis algorithm based on composite deep neural networks and proposes an adaptive imbalanced text sentiment classification learning method (AILM) based on memory modules. This method uses memory modules to remember difficult samples and improve the frequency of difficult samples in later training by adapting and adjusting the distribution of a few text samples, thereby improving the accuracy of classification. Based on the above research, the universality of AILM was practiced using a Chinese text dataset, and an optimal Chinese news classification model was constructed by fine-tuning parameters.*
**Keywords:** Text classification; Neural networks; Unbalanced learning; Chinese text

1. **Introduction.** Since the beginning of 2000, with the rapid development of the Internet, e-commerce platforms and social networking platforms have sprung up like mushrooms. Internet users have gradually become accustomed to making emotional, political and other statements on the network platform [1]. Today, a large number of imbalanced text comments are generated from social media and e-commerce platforms at every moment, such as Taobao, JD.com, Amazon, Weibo, Twitter, etc. In these massive comment data containing user opinions, most of them exist in unstructured text form, and the distribution of these comment data categories is uneven. That is, there may be situations where there are many samples in certain categories and few samples in others. If this problem can be effectively solved and the information in it can be fully utilized and mined, it will bring huge value information to government departments and related enterprises. However, how to extract and analyze this information has become a challenge. With the rapid development of AI technology and big data technology, new possibilities have been provided to solve the above-mentioned problems. Among them, text sentiment

analysis is one of the most popular research areas in natural language processing (NLP) [2], and how to efficiently solve the problem of imbalance in text classification is also a hot research direction in NLP. Sentiment Analysis (SA), also known as opinion mining, belongs to an important branch of the NLP field [3]. Its main task is to mine and detect text information and emotional tendencies through emotion dictionary-based methods and machine learning, so as to identify Internet users' feelings, emotions and attitudes. At present, text sentiment classification and imbalanced learning have achieved many excellent results in the academic community, but they still have extremely high research and application value in both industry and academia, because they play an irreplaceable role in politics, business, security and other fields.

1.1. **Related work.** Due to the computational performance of hardware, the research community lost interest in neural network-based deep learning in the late 1990s, believing that training deep neural networks was very expensive. But with the rapid development of computing performance in recent years, deep learning has achieved many breakthrough results in NLP. The method of manually creating features based on deep learning greatly saves human resources and performs well in many sentiment classification tasks, outperforming traditional algorithms based on emotion dictionaries and machine learning.

In aspect-level sentiment classification, Onan [4] proposed a deep learning-based method for product review analysis from Twitter, combining GloVe embeddings with a CNN-LSTM architecture. Liu and Guo [5] introduced a unified BiLSTM–Attention–CNN model to address high-dimensional and sparse text data. Huang et al. [6] proposed AEC-LSTM, which leverages emotional semantic information in text topics and context for improved classification accuracy.

Sentence-level sentiment classification treats each input as a single sentence. Early approaches relied on sentiment lexicons, rules or machine learning with handcrafted features. With deep learning advances, neural models now dominate. Xi et al. [7] designed a 3D CNN with spatial pyramid pooling for sentence classification, effectively capturing complex relationships. Zhang et al. [8] proposed 3W-NN with a ternary confidence divider, improving emotion classification accuracy. Tang et al. [9] developed a joint segmentation–classification framework to handle inconsistent sentiment polarity. Khan et al. [10] presented a rule-based method for classifying blog comments into subjective/objective and polarity categories. Rao et al. [11] introduced SR-LSTM, a two-hidden-layer neural model for long-sequence sentiment extraction.

Traditional document-level methods used bag-of-words or fixed-length vectors, leading to semantic loss. Khoo and Johnkhan [12] created WKWSCI, a universal dictionary compared against five others. Wang et al. [13] validated ensemble learning on five classifiers (decision trees, Naive Bayes, KNN, max-entropy, SVM). Heikal et al. [14] combined CNN and LSTM with ensemble methods for Arabic tweet analysis.

Although deep learning models (CNN, RNN, LSTM-2DCNN, Conv-LSTM, SSR-LSTM) have advanced, simultaneous consideration of semantic information, multi-level dependencies and word/sentence importance remains a key challenge.

Imbalanced text data distributions pose additional challenges. Two main approaches address this: data-level resampling and cost-sensitive learning. Resampling includes oversampling, undersampling and hybrid methods. Simple random oversampling duplicates minority samples, causing overfitting; SMOTE [15] interpolates new minority samples; DDPDE [16], DBSMOTE [17] and MWMOTE [18] offer further improvements. Undersampling methods include RUS, Condensed Nearest Neighbor [19], CBUS [20] and EUSBoost [21], but risk information loss. Hybrid sampling (e.g. SMOTE+TomekLinks [22], SMOTE+ENN) combines both to mitigate these issues. Yang et al. [23] incorporated

misclassification and testing costs; Lu et al. [24] proposed optimal cost weight search and fitting methods. Cost-sensitive learning enhances algorithm robustness by weighting losses differently across classes.

1.2. **Motivation and contribution.** This article analyzes the current research status of text sentiment analysis, aiming to construct a composite neural network model for document-level text to explore semantic information and important content features. Based on text data category distributions, we propose an adaptive imbalanced text sentiment classification learning method (AILM) using memory modules. AILM remembers difficult samples and adaptively increases their sampling frequency during training, improving accuracy without manual resampling or loss-sensitivity design. We validate AILM's universality on a Chinese text dataset and build an optimal news classification model by fine-tuning parameters.

## 2. Analysis of relevant principles.

### 2.1. Text representation method.

2.1.1. *Traditional statistical language model.* The traditional language model is specifically manifested as dividing a document into words, and representing the document as a vector according to a certain method. Each dimension of the vector is represented as a word, and the weight corresponding to each dimension represents the importance of the corresponding word in the text. Traditional language models include Boolean models, vector space models, probability models, etc.

(1) Boolean Text Representation Model. The Boolean model is a strict matching model, which is a representation model based on Boolean logic functions and set theory. Its general representation is:

$$D = \big(f(t_1), f(t_2), ..., f(t_n)\big) \tag{1}$$

if document $D$ contains feature $t_i$, then the position $f(t_i)$ corresponding to feature $t_i$ in document $D = 1$, otherwise $f(t_i) = 0$. This model has a simple structure, fast calculation speed, and is easy to understand, but it ignores the weight and order of feature items in the document, and cannot represent the semantic information of the text well.

(2) Vector space model. The Vector Space Model (VSM) converts the frequency of feature words appearing in text into real numbers to construct a high-dimensional vector space, where each dimension of the vector represents the corresponding feature item. Due to the varying importance of different feature items in text, in general representation methods with feature weights, the greater the importance of feature words, the greater their weight. It can generally be expressed as:

$$D = (t_1, w_1; t_2, w_2; ...; t_n, w_n) \tag{2}$$

among them, $t_i$ is the feature item, and $w_i$ is the weight size corresponding to $t_i$, which is generally calculated according to TF-IDF (Term Frequency Inverse Document Frequency).

(3) Probability model. Probability model is a model that uses knowledge such as probability theory and statistics to solve text representation problems. Its basic idea is to use probability to represent information such as word frequency, text frequency, and number of feature items. The similarity between text $D$ and $Q$ is as follows:

$$P(R|D, Q) = \sum_i X_i \times \lg \frac{p_i(1 - q_i)}{q_i(1 - p_i)} \tag{3}$$

this model considers not only the probability correlation between words, but also the probability correlation between documents. Boolean and vector space models, on the

other hand, believe that feature items are linearly independent, while in fact, there is a close relationship between feature items and documents.

2.1.2. *Distributed word embedding model.* Traditional statistical language models are prone to the following problems: (1) dimension explosion, with many feature items in the document and a large spatial dimension of the vector, which often leads to the problem of dimension explosion; (2) Semantic loss, traditional methods assume that words are independent of each other. The above issues will lead to more comprehensive problems, such as low computational efficiency and poor generalization ability caused by dimension explosion. Currently, research has focused on distributed representation methods, which involve embedding words into low dimensional and fixed length vector representations. All vectors form a vector space, and the Euclidean distance between words represents their semantic similarity.

(1) Neural Network Language Model. The distributed word vector embedding model is based on the Neural Network Language Model (NNLM) [25], which uses feedforward neural networks to optimize the objective function. Its original task is to first assign a word vector to each word in the vector space, then use neural networks to build a language model for word prediction, and generate distributed representations of words while optimizing the objective. The objective function is:

$$L(\theta) = \sum_{t} \log P(w_t | w_{t-n+1}, ..., w_{t-1}) \tag{4}$$

Using neural networks, it can model the relationship between the current word and $n-1$ words. This method uses a forward window with a length of $n-1$ to traverse the entire corpus and sum it up, maximizing the probability of the objective function. At the same time, the total probability sum of the predicted words is 1:

$$\sum_{w_t} P(w_t | w_{t-n+1}, ..., w_{t-1}) = 1 \tag{5}$$

The model inputs $N-1$ one hot words into the network through an embedding layer and multiplies them with a $V * M$ matrix (where $V$ is the size of the vocabulary and $M$ represents the mapped dimension) to map $N-1$ distributed word vector representations, with each word vector having a dimension of $1 * M$. At the top of the model, a softmax is added to calculate the probability output of the prediction times, with a dimension of $1 * M$. This method ultimately obtained an embedding matrix of V*M, which has better generalization ability than the n-gram model and largely solves the problem caused by data sparsity.

(2) Skip Gram model. The Skip Gram model removes the nonlinear hidden layer of the feedforward neural network in NNLM and directly connects the embedded layer with the softmax computing layer; Ignoring the sequence information of the contextual environment, summarizing the vector representations of all words into an embedding layer; Incorporate future words after the current word into the context. The definition of hidden layer $h$ is as follows:

$$h = W^T_{(k,.)} := v^T_{w_I} \tag{6}$$

In the output layer, $C$ polynomial distributions are output, and each output is calculated from the same hidden layer and output matrix.

$$p(w_{c,j} = w_{O,c} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{i=1}^{V} \exp(u_i)} \tag{7}$$

The objective function of the Skip-gram model is:

$$L = \log p(w_{O,1}, ..., w_{O,c} | w_I) \tag{8}$$

2.2. **A text classification method based on long short term memory networks.** LSTM includes the following steps:

(1) Step 1: Each cell component will input the output $h_{t-1}$ of the previous cell component and the word vector input $x_t$ of the current cell. By using a forget gate, the value of $[h_{t-1}; x_t]$ will be mapped to a value range of 0-1 to determine which information should be discarded.

$$f_t = \sigma(W_f \cdot [h_{t-1}; x_t] + b_f) \tag{9}$$

(2) Step 2: Update the state of cells and calculate candidate cell state $\tilde{C}_t$. Firstly, calculate $i_t$ through the input gate, and then use the *tanh* activation function to calculate candidate state $\tilde{C}_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}; x_t] + b_i) \tag{10}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}; x_t] + b_c) \tag{11}$$

(3) Step 3: Update cell status $C_t$:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{12}$$

(4) Step 4: Determine the output value through the output gate. Its calculation includes the calculation of the current hidden layer state and the calculation of the output value:

$$O_t = \sigma(W_O \cdot [h_{t-1}; x_t] + b_O) \tag{13}$$

$$h_t = O_t * \tanh(C_t) \tag{14}$$

The LSTM text classification model generally uses the last hidden layer state $h_t$ of the sequence to represent the final output of the text for text classification. There are also some operations that use max pooling or average pooling on $[h_1, ..., h_m]$ to obtain high-level vector representations of the text for text classification.

## 3. Adaptive unbalanced text sentiment classification learning method based on memory module.

3.1. **Adaptive unbalanced learning framework based on memory module.** The architecture of AILM is shown in Figure 1. The AILM workflow can be divided into the following steps:

(1) Initialize the memory module with a capacity of $n$ rows of entries, each with two columns representing the stored sample vector and corresponding labels. Set the difficulty threshold $thr$, $thr = \sigma \log N$, where $\sigma$ is the hyperparameter.

(2) Input the text $T_i$, and the encoder encodes it into a fixed dimensional vector $v_i$. Take the text vector $v_i$ as the query of the memory module, calculate the $k$ entries with the smallest Euclidean distance from the query in the memory module as their $k$ nearest neighbors, and obtain the corresponding distance vector.

(3) Cascade each of the $k$ entries with a query and input it into the decoder module to generate an advanced vector $e_i$ for text sentiment classification.

(4) Use the advanced vector $e_i$ for classification, calculate the error loss between the probability distribution output by the classifier and the ground truth, and determine whether the error has reached the difficulty threshold. If loss is greater than or equal to $thr$, the sample is considered difficult to classify and stored in the memory module for reappearance in the later training process, making it easier to remember the sample.

(5) Repeat steps (2), (3), and (4) until the training is complete.

This section will provide a detailed description of the three modules of AILM [26]: encoder module, memory module, decoder module, and the working mechanism of AILM.

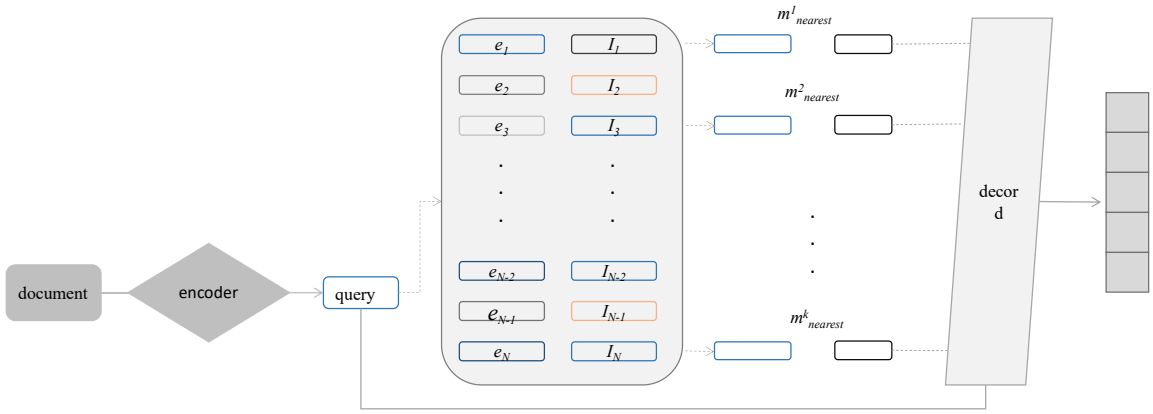


Figure 1. Adaptive unbalanced learning framework based on memory module

3.2. **Encoder module of AILM.** The actual function of the AILM encoder module is a feature extractor, and the AILM encoder module encodes the semantics and position of a given document. This article will use AttDR-2DCNN, as well as popular text classification algorithms RCNN, DPCNN, and TextCNN as semantic and positional encoders to obtain the primary distributed vector representation of documents, and verify the effectiveness of AILM.

3.2.1. *AttDR-2DCNN encoder.* AttDR-2DCNN consists of three components: document representation module, 2D convolution module, and output layer. The document representation module and 2D convolution module in AttDR-2DCNN used in this work are no different. But in the output layer module, a softmax classification layer is used to predict semantic relationship labels $\mathbf{y}$ from a set of discrete classes $Y$:

$$p(y|s) = softmax(W_s o + b_s) \tag{15}$$

where $o$ is the output of the 2D convolution module and the flattened one-dimensional vector is used for the final document level sentiment classification. If AttDR-2DCNN is used as the encoder in this chapter, only the output $o$ of semantic encoding by the document representation module and 2D convolution module needs to be obtained as a query for querying the k-nearest memory bar in the memory module.

3.2.2. *RCNN encoder.* The RCNN model is a highly effective text classification model based on composite neural networks, which combines RNN and CNN, a bidirectional RNN and a max pooling layer, for text classification. It includes the following steps:

(1) Firstly, vector encode word $w_i$ and input the text into the sub embedding layer to obtain a distributed representation $e(w_i)$.

(2) Input word vector $e_i$ into a bidirectional RNN, which can be an implementation of GRU or LSTM, to obtain left context $c_l(w_i)$ and right context $c_r(w_i)$.

(3) Splice the output results of steps one and two to obtain the middle representation $x_i$ of the word:

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \tag{16}$$

(4) Perform nonlinear mapping on $x_i$:

$$y_i^{(2)} = \tanh\left(w^{(2)} x_i + b^{(2)}\right) \tag{17}$$

(5) Perform text representation learning on 1, using a max pooling layer to extract features:

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)} \tag{18}$$

3.2.3. *DPCNN encoder.* In recent years, the feature pyramid structure has been very popular and successfully applied in the field of computer vision. Its basic idea is to greatly improve the ability to detect small objects through simple network connections and transformations, without increasing the computational complexity of the model. The feature pyramid is mainly applied in the field of object detection, and its discovery is mainly aimed at solving (1) how to learn multi-scale feature representations with strong semantic information; (2) How to design a universal feature representation to solve multiple sub problems in object detection; (3) How to efficiently compute multi-scale feature representations without increasing computational complexity. With the tremendous success of FPN in the field of images, people are also trying to use this method to efficiently handle text classification problems in the field of natural language processing. DPCNN mainly includes unsupervised word embedding layer, domain embedding layer, short-circuit connection with pre activation, and downsampling with a fixed number of feature maps.

3.2.4. *Memory module.* The external memory module in AILM serves as a database function to store the features of each difficult to distinguish sample, as shown in the memory module in Figure 1. The memory module contains two columns, each corresponding to a different attribute value. The memory module implemented in this article will store two attributes used for classification, namely sample vector representation $e_i$ and real sample label $y_i$. Each row in the memory module represents a data point information. The memory module can be expressed as:

$$M = \{m_1, m_2, ..., m_N\} \tag{19}$$

where $N$ is the size of the memory module, $m_i = [e_i; y_i]$. Given the output query of the AILM encoder, the $k$ nearest neighbors ($k$ sample vectors closest to the query represent the complete rows of $e_i$) in the memory module are searched for as the output of the module for later training. This allows the model to predict new samples while adding more weight and improving its learning of these difficult samples. This article selects the metric for calculating the distance between query and $e_i$ in the memory module using Euclidean distance calculation:

$$dist(query, e_i) = \sqrt{\sum_{j=1}^{l} (query^j - e_i^j)^2} \tag{20}$$

where $l$ is the dimension of query and $e_i$, and $query^j$ and $e_i^j$ are the $j$-th element of query and $e_i$ vectors, respectively. By searching for the memory module, $k$ nearest neighbors $\{m_{nearest}^1; m_{nearest}^2; ..., m_{nearest}^k\}$ will be returned and cascaded with the query:

$$memory\_out\{p_{nearest}^1; p_{nearest}^2; ...; p_{nearest}^k\} \tag{21}$$

Not all samples are stored in the memory module, only difficult samples are written into the memory module. But how to determine whether the sample representation output by the encoder is difficult? The difficulty of a sample depends on its ability to correctly predict labels. In this section, a hyperparameter $\sigma$ is set, which is the classification difficulty threshold, and $\sigma \propto \ln N$, $N$ is the number of categories classified. This means that if the prediction confidence of the correct category is less than the classification threshold, the sample will be considered a difficult sample and written into the memory module, making the model pay more attention to improving the learning of these difficult samples while predicting new samples. In the case of document level multi classification, this section will use the cross-entropy loss function as the cost function and the classification loss as

the confidence to control whether the sample is written into the memory module:

$$insert(e_i, y_i) = \begin{cases} 1 & \text{if } CEL(y_i^p, y_i) \geq \sigma \\ 0 & \text{if } CEL(y_i^p, y_i) < \sigma \end{cases} \tag{22}$$

where *insert* represents the storage operation. When the value is 1, it represents the storage of the memory module. Otherwise, it will be skipped; $y_i^p$ is the predicted value of the model, and $y_i$ is the true label value.

3.3. **AILM decoder module.** The decoder module will use query and the $k$ nearest neighbors obtained from the memory module as inputs, i.e. *memory_out*. Next, the decoder decodes *memory_out* and obtains the final high-level semantic vector for sentiment classification at the document level. This article will implement two decoder structures: relational self attention feedforward decoder and contextual attention decoder.

3.3.1. *Relationship self attention feedforward decoder.* The input of the relational self attention feedforward decoder is the *memory_out* output from the memory module and the distance vector $dist = \{d_1, d_2, ..., d_k\}$, where $d_i$ represents the distance between the query and its $i$-th neighbor among its nearest $k$ neighbors. The *memory_out* will be inputted into multi-layer perceptrons and multi head attention networks, and RSFD can learn deeper relational representations of *memory_out*. Finally, the RSFD decoder uses the distance vector *dist* to weight the final output of multi head attention, and the high-level vector representation with an output dimension of $N$ (number of categories) is used for the final classification.

3.3.2. *Context attention decoder.* The relational self attention feedforward neural network distinguishes the importance of neighboring vectors in *memory_out* through self attention multi head attention network, and finally uses *dist* to weight the output into high-density semantic vectors for classification. The context attention decoder needs to initialize a context vector $u$ as a query in the attention mechanism to query which neighbors are important for the output. In this decoder, the output *memory_out* of the memory module is first input to a feedforward neural network to obtain annotations:

$$n_i = \tanh(w_n p_{nearest}^i + b_n) \tag{23}$$

Then calculate the weight of each neighbor using the context vector $u$:

$$\alpha_i = \frac{\exp(u^T n_i)}{\sum^k \exp(u^T n_i)} \tag{24}$$

Finally, the neighbors are weighted to output a vector representation with dimension $n$ for classification:

$$output = \sum_{i=1}^{k} \alpha_i \times n_i \tag{25}$$

## 4. Model solving and result analysis.

4.1. **Experimental model.** The dataset used for the experiment in this article is Sina News headlines, which are a single sequence without compositional semantics. Therefore, this chapter will no longer use document level classification methods with compositional semantics, such as Conv-LSTM, AttDR-2DCNN, etc. This article will add an adaptive imbalanced learning module based on memory networks to these excellent text classification models and practice them on a Chinese text dataset [27]. The aim is to construct an excellent Chinese news text classification model and demonstrate the universality of the AILM method. The experimental models used in this article include: single encoder

text classification models, including TextCNN, RCNN, and DPCNN models; A text classification model based on memory network and adaptive imbalanced learning, including TextCNN+AILM, RCNN+AILM, DPCNN+AILM, where the encoder of AILM includes two implementations: relational self attention feedforward decoder RSFD and contextual attention decoder CAD.

4.2. **Experimental content.** This article first explores the impact of memory network-based adaptive imbalanced learning methods on balanced Chinese text datasets based on different text classification models. In addition, this article will also conduct experiments in the following aspects to achieve the best model, thereby achieving the best Sina news title text classification model:

(1) The impact of words on the model.

The semantic structure of Chinese includes the semantics of words and the semantics of characters. Therefore, a news headline sequence can yield different results based on word segmentation or word segmentation methods: character lists and word lists. After dividing the news title text into words or characters to obtain a word list, vector embedding is used to represent the text as input for the post model. The segmentation and segmentation modes can respectively obtain word vector embedding and word vector embedding. This chapter will explore the impact of different embeddings caused by word segmentation on the model.

(2) The impact of the selection of text length *max_len* on the model.

In order to input data into the Chinese news text classification model and conduct small batch sample training, it is necessary to fix the length of the input sequence to *max_len*. For news title samples with a length greater than *max_len*, it is necessary to perform segmentation processing and discard the parts of the sample that are larger than *max_len*; For news title samples with a length less than *max_len*, filling is necessary. There are many ways to fill them, and this article chooses the backward zero filling method, which means filling zeros after the title sequence until the sequence reaches *max_len* length.

(3) The influence of the number of hidden units in RCNN encoder on the model.

The main research content of this article is not only to verify the effectiveness of AILM on Chinese datasets, but also to construct an optimal Chinese news text classification model to meet the needs of real production environments. In the encoder model described above, the performance of RCNN classifier stands out, so the number of RCNN hidden layer units is also discussed as an experimental parameter variable.

4.3. **Experimental result.**

4.3.1. *Overall performance.* This article conducted experiments on the Chinese news title dataset and completed the classification of the dataset. AILM models based on different encoders and decoders were used in the experiment, and the experimental results are presented in Table 1. From Table 1, it can be found that the AILM method of CAD+RCNN is most suitable as a model for Chinese news text classification. Its classification accuracy on the constructed imbalanced Chinese news title dataset reached 92.37%, which is a significant improvement compared to the original RCNN classification model, with an increase of 1.5 percentage points. The AILM model based on RSFD and the AILM model based on CAD have shown significant improvements compared to the corresponding original classification models. Among them, the model based on TextCNN encoder in RSFD has improved by 1.11 percentage points, the model based on RCNN encoder has improved by about 1.1 percentage points, and the model based on DPCNN encoder has improved by about 0.7 percentage points; The overall performance of CAD based models is better than that of RSFD based models, with an average improvement of about 1.1 percentage points.

Among these models, the improvements based on RCNN and TextCNN models achieved the expected results, while the DPCNN based model showed good improvements after using the AILM method, but the performance was still unexpected. The performance was the worst among these three encoding methods, and it is not suitable to use it as a Chinese news text classification model in real life. The experimental conclusion of this section is that the RCNN news text classification model based on AILM is most suitable for constructing the optimal model. In the following sections, we will discuss the optimal parameter combination in constructing the optimal model.

Table 1. Performance of the AILM model on the THUCNews Chinese news dataset

| **Decoder** | **Encoder** | **Accuracy (%)** |
|---|---|---|
| nothing | TextCNN | 90.61 |
| | RCNN | 90.87 |
| | DPCNN | 90.13 |
| RSFD | TextCNN | 91.72 |
| | RCNN | 91.96 |
| | DPCNN | 90.85 |
| CAD | TextCNN | 91.48 |
| | RCNN | 92.37 |
| | DPCNN | 91.04 |

4.3.2. *The impact of word segmentation on the model.* This section of the experiment used both space-based segmentation and jieba based segmentation methods to divide the news title text sequence into characters and word lists for subsequent vector embedding operations.

Table 2. The performance of AILM in word segmentation and phrases segmentation

| **Sequence segmentation method** | **Decoder** | **Encoder** | **Accuracy (%)** |
|---|---|---|---|
| Word segmentation | RSFD | TextCNN | 91.72 |
| | | RCNN | 91.96 |
| | | DPCNN | 90.85 |
| | CAD | TextCNN | 91.48 |
| | | RCNN | 92.37 |
| | | DPCNN | 91.04 |
| Phrases segmentation | RSFD | TextCNN | 88.23 |
| | | RCNN | 89.11 |
| | | DPCNN | 88.03 |
| | CAD | TextCNN | 88.61 |
| | | RCNN | 88.75 |
| | | DPCNN | 88.19 |

From Table 2, it can be seen that among different sequence segmentation methods, the AILM model based on RCNN encoder always performs the best in the corresponding categories. In theory, when the feature items representing the text are sufficient, the semantics in the word vector obtained by embedding after jieba segmentation are more reasonable than those obtained by word segmentation. However, surprisingly, the word

vector embedding model with higher semantics theoretically performs worse, with an average difference of more than 2 to 3 percentage points compared to the word vector-based model, and even 3.6 percentage points in the RCNN+CAD based model. The possible reasons for this are that the dataset is the title dataset, with an average character length of 19 and an average word length of 10. For the model, the input with a sequence of 10 may contain less semantics, and its feature items may not represent the entire sequence well. On the other hand, the input model with a sequence of 19 will have more training parameters to better express the input Chinese text. Therefore, for this news text dataset, the recommended sequence segmentation method is the word segmentation method.

4.3.3. *The influence of the selection of text length on the model.* In order to construct the best Chinese news text classification model, this section searches for the optimal text sequence length setting. In previous experiments, it was found that word segmentation-based methods were not suitable for short text data such as news headlines. In this section, a word segmentation method was used to segment the sequence. In the segmentation results of word segmentation, the average sequence length is 19.2. The *max_len* parameter searched for in this article will be expanded around this value, with a minimum *max_len* of 15 and a maximum of 50. The model used is an AILM model based on RCNN+CAD, TextCNN+CAD, and DPCNN+CAD.

The impact of parameter text length *max_len* on model performance is shown in Figure 2. The horizontal axis in the figure represents the range of text length *max_len* values, and the vertical axis represents accuracy. The trend of the three curves is basically the same, reaching their peak when *max_len* is 25. In the RCNN curve, the classification accuracy of the model can reach 92.37% when *max_len*=25, and the model performs the worst at *max_len*=15, with an effect of only about 90%; When *max_len* starts to increase at 15, the classification accuracy of the model improves accordingly, and the slope of the improvement is large. When it reaches 25, it reaches the peak of classification accuracy. When *max_len* is greater than 25 and gradually increases, the accuracy of the model begins to gradually decrease and stabilizes within a certain level range. The range of this fluctuation indicates that *max_len* has a significant impact on the performance of the model. When the value of *max_len* is smaller than the optimal value, the performance of the model is greatly reduced. However, when the value of *max_len* is larger than the optimal value, it not only does not bring performance gains, but also damages the model; This curve indicates that when the value of *max_len* is less than 25, the input sequence cannot represent the text well. In addition to losing a lot of semantic information, the truncation of the length is due to the short sequence length, which leads to the model not being able to obtain enough feature items, resulting in poor performance. This once again indicates that the segmentation-based method is not suitable for this dataset; When the value of *max_len* is greater than 25, although it can more comprehensively represent the entire sequence information, a large value of *max_len* means that short sequences need to be filled with a large number of zeros to meet the input requirements of the model. During the zero-filling process, too much noise information unrelated to text semantics will be added, thereby reducing the performance of the model.

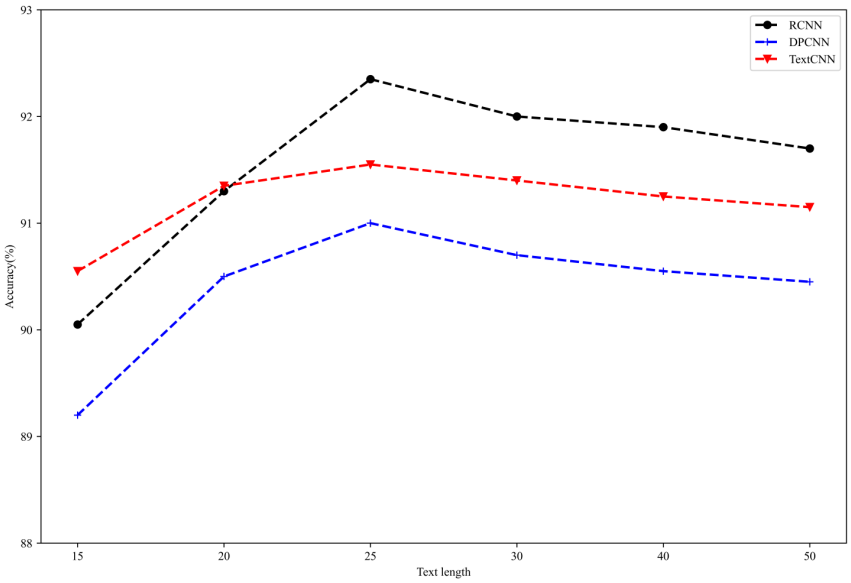


Figure 2. The influence of the selection of text length on the model

4.3.4. *The influence of the number of hidden units in RCNN encoder on the model.* The number of hidden layer units in RCNN encoder is also an important factor affecting model performance. Its value not only affects the accuracy of the model, but also affects the training time of the model. This section will explore the impact of RCNN on model accuracy and training time. The model used is an AILM model based on RCNN+CAD.

Figure 3 shows the impact of the number of hidden units in RCNN on model accuracy, with the vertical axis representing model accuracy and the horizontal axis representing the number of hidden units. From the curve, it can be seen that the accuracy of model performance is positively correlated with the number of units within the range of abscissa values. Figure 4 shows the effect of the number of hidden units in RCNN on model training time. The horizontal axis represents the number of units, and the vertical axis represents the time consumption for every 2000 training steps when the number of small batch training is 64. From the curve, it can be seen that the consumption of model training time is directly proportional to the number of units. When the number of units is 64, it takes about 0.75 minutes per 2000 steps. When the number of units is 512, the consumption time exceeds two minutes. Taking into account the two subgraphs, when the number of units increased from 256 to 512, the accuracy improved by about 0.3 percentage points, while the time consumption increased by about 50%. This indicates that when the number of units reached a certain number, the performance gain became very small, but the time consumption still increased sharply. By significantly increasing the number of units, the performance gain is not only small but also time-consuming, and there may even be a risk of overfitting due to too many parameters. It is not worth increasing the complexity of the model in order to pursue minimal performance gain. Therefore, the conclusion of this experiment is that the optimal number of RCNN hidden layer units is 512.

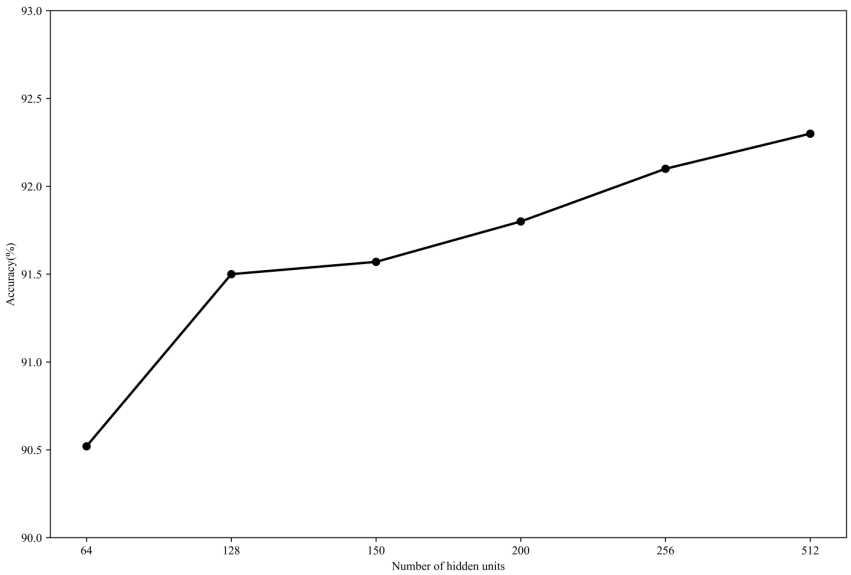


Figure 3. The relationship between the number of hidden units and accuracy.

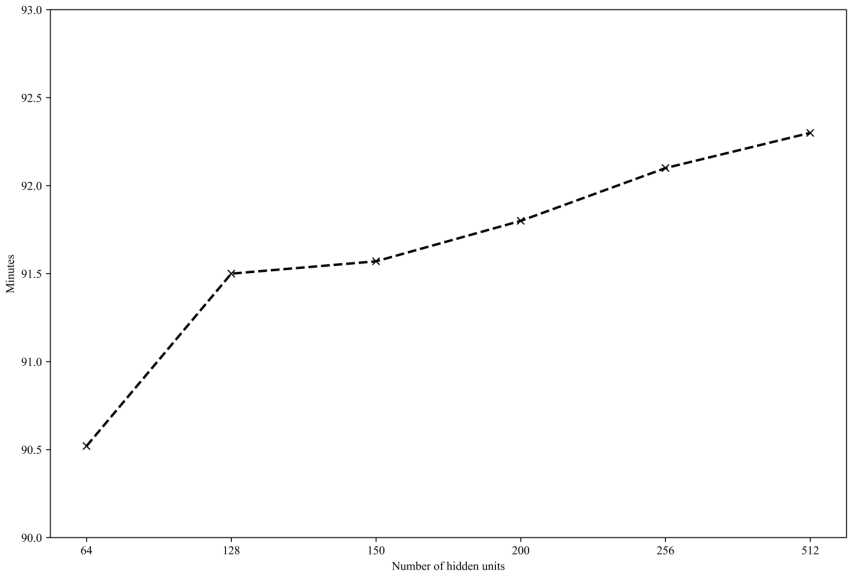


Figure 4. The relationship between the number of hidden units and model training time.

5. **Conclusions.** The content of this article is the text sentiment analysis method, which is an important component of AI technology and has high research and practical value. This article will apply the proposed method to Chinese news text data and explore the related technologies of Chinese text classification. Among them, an imbalanced dataset was constructed using the THUCNews dataset to simulate real production environment data. Then, the influence of Chinese word segmentation methods and sequence length setting model parameters on the model was explored, thus achieving an excellent Chinese news text classification model, which has high practical significance for the implementation of subsequent systems. Although the model proposed in this article has made significant improvements over the existing methods, there are still some directions worth further research: the method in this article does not take into account many other imbalanced learning methods, such as data resampling, cost sensitive learning, ensemble

learning, etc. These methods are not isolated, and in many cases, these methods can be integrated. In future research, if these methods can be integrated, perhaps better performance improvements can be achieved.

**Acknowledgment.** This work is partially supported by the Education and Teaching Reform Project of 2024: A Study on the Joint Construction by Schools and Families in Integrating Chinese Excellent Traditional Culture into Language and Literacy Teaching (No. 2024JGYB011).

## REFERENCES

[1] Y. To, "Friends and foes: rethinking the party and Chinese big tech," *New Political Economy*, vol. 28, no. 2, pp. 299–314, 2023.
[2] P. Nandwani and R. Verma, "A review on sentiment analysis and emotion detection from text," *Social Network Analysis and Mining*, vol. 11, no. 1, p. 81, 2021.
[3] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731–5780, 2022.
[4] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, e5909, 2021.
[5] G. Liu and J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.
[6] F. Huang, X. Li, C. Yuan, S. Zhang, J. Zhang, and S. Qiao, "Attention-emotion-enhanced convolutional LSTM for sentiment analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4332–4345, 2021.
[7] X. Ouyang, K. Gu, and P. Zhou, "Spatial pyramid pooling mechanism in 3D convolutional network for sentence-level classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2167–2179, 2018.
[8] Y. Zhang, Z. Zhang, D. Miao, and J. Wang, "Three-way enhanced convolutional neural networks for sentence-level sentiment classification," *Information Sciences*, vol. 477, pp. 55–64, 2019.
[9] D. Tang, B. Qin, F. Wei, L. Dong, T. Liu, and M. Zhou, "A joint segmentation and classification framework for sentence level sentiment classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1750–1761, 2015.
[10] A. Khan, B. Baharudin, and K. Khan, "Sentiment classification using sentence-level lexical based," *Trends in Applied Sciences Research*, vol. 6, no. 10, pp. 1141–1157, 2011.
[11] G. Rao, W. Huang, Z. Feng, and Q. Cong, "LSTM with sentence representations for document-level sentiment classification," *Neurocomputing*, vol. 308, pp. 49–57, 2018.
[12] C. S. Khoo and S. B. Johnkhan, "Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons," *Journal of Information Science*, vol. 44, no. 4, pp. 491–511, 2018.
[13] G. Wang, J. Sun, J. Ma, K. Xu, and J. Gu, "Sentiment classification: The contribution of ensemble learning," *Decision Support Systems*, vol. 57, pp. 77–93, 2014.
[14] M. Heikal, M. Torki, and N. El-Makky, "Sentiment analysis of Arabic tweets using deep learning," *Procedia Computer Science*, vol. 142, pp. 114–122, 2018.
[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[16] S. Chan, A. Santoro, A. Lampinen, J. Wang, A. Singh, P. Richemond, J. McClelland, and F. Hill, "Data distributional properties drive emergent in-context learning in transformers," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 18878–18891, 2022.
[17] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "DBSMOTE: density-based synthetic minority over-sampling technique," *Applied Intelligence*, vol. 36, pp. 664–684, 2012.
[18] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2012.
[19] F. Angiulli, "Condensed nearest neighbor data domain description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1746–1758, 2007.
[20] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409, pp. 17–26, 2017.

[21] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognition*, vol. 46, no. 12, pp. 3460–3471, 2013.

[22] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[23] Q. Yang, C. Ling, X. Chai, and R. Pan, "Test-cost sensitive classification on data with missing values," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 626–638, 2006.

[24] H. Lu, Y. Xu, M. Ye, K. Yan, Z. Gao, and Q. Jin, "Learning misclassification costs for imbalanced classification on gene expression data," *BMC Bioinformatics*, vol. 20, pp. 1–10, 2019.

[25] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 19–46, 2024.

[26] Y. Ma, Y. Peng, and T.-Y. Wu, "Transfer learning model for false positive reduction in lymph node detection via sparse coding and deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121–2133, 2022.

[27] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, and L. Liu, "Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction," *IEEE Access*, vol. 8, pp. 104555–104564, 2020.