International Journal for Housing Science and Its Applications
Publish August 6, 2025. Volume 46, Issue 3 Pages 5847-5860

https://doi.org/10.70517/ijhsa463503

Efficient Construction Strategy for Natural Language Processing Models in Low-Dimensional Embedding Space Based on Self-Attention Mechanisms

Bo Liu[1,*]
[1] School of Mathematics and Statistics, Hanshan Normal University, Chaozhou, Guangdong, 521041, China
Corresponding authors: (e-mail: sirlb@126.com).

Abstract With the development of natural language processing technology, traditional models face the problems of inefficiency and information loss when processing long sequential texts. The low-dimensional embedding space combined with the self-attention mechanism provides a new direction for optimizing the model, which can capture the text semantic information more efficiently, reduce the computational complexity, and at the same time improve the model's generalization ability and classification accuracy, so as to provide a more efficient solution for natural language processing. In this study, we propose a natural language processing model based on the self-attention mechanism, which combines PCA and VSM to construct a low-dimensional embedding space, aiming to solve the efficiency problem of traditional models for processing long sequential texts. Methodologically, the model adopts principal component analysis for dimensionality reduction of word vectors, uses BiLSTM network to extract text features, and introduces the self-attention mechanism to give different weights to the text. The experiments are carried out on MRD and SST datasets, and the results show that: the training time on the two datasets using the PCA-VSM model is 107s and 104s respectively, which is much better than other models; the model has the highest accuracy when the cumulative contribution rate of the feature values is 90%; under the optimal parameter configurations, the BLEU metrics of this paper's model on the MRD and SST datasets respectively reach 0.369 and 0.381, and the Rouge-L metrics are 0.489 and 0.488 respectively, which are significantly better than the other compared models. It is shown that the self-attention mechanism model based on low-dimensional embedding space can effectively improve the performance of natural language processing tasks.

Index Terms Self-attention mechanism, Low-dimensional embedding space, Principal component analysis, BiLSTM; Vector space mode, Natural language processing

I. Introduction

Natural Language Processing (NLP) is an important research area in the field of Artificial Intelligence, aiming to enable computers to understand and process human language. In NLP, word vector model design is a key task whose goal is to transform words into a set of numerical representations so that computers can analyze and process them [1]. In traditional text representations, words are usually converted into one bit vectors, where each element represents a word, but each word requires an extra dimension, which significantly increases storage and computational costs, word embedding techniques solve this problem by converting words into low-dimensional vectors [2], [3]. While embedding vectors are usually used to represent words, sentences, or other textual data, the semantic and syntactic information between words can be better captured by mapping textual data to real vectors in a low-dimensional space [4]. Embedding vectors can also be understood as a dimensionality reduction technique, which can reduce the complexity of the data and improve the computational efficiency by mapping high-dimensional textual data into a low-dimensional space, as well as help to minimize the impact of dimensionality catastrophe [5]. When constructing low-dimensional embeddings, a variety of methods can be used, such as matrix decomposition, deformable attention, hierarchical embedding, and so on. For example, literature [6] used matrix decomposition methods to construct an NLP word vector model, and formulated semantic rounding networks, semantic fusion alignment methods, and multiview subspace clustering, which together realized the construction of NLP colleges and universities in the low-dimensional embedding space. However, these methods do not solve the three major pain points of model construction in spatially reduced dimensional embedding model construction, i.e., difficulty in maintaining orthogonality of embedding vectors, distraction of attention distribution, and semantic features being hidden [7], [8]. In multilingual task processing, the accuracy of alignment across languages decreases as the

dimensionality decreases. In dynamic semantic space dimensionality reduction, NLP models are prone to ambiguous understanding of contextual semantic associations and word order dependencies [9].

The self-attention mechanism was first proposed in the image domain in the 1990s, but it was not popularized in the deep learning domain until 2014 [10], [11]. Subsequently, the self-attention mechanism has gained a lot of popularity in the field of NLP. The computational layer based on the self-attention mechanism has been applied to many tasks in the field of NLP, such as machine translation, text categorization, and text summarization, because of its efficient computational efficiency and powerful representation learning capability. Literature [12] improved the structure of the encoder of the Transformer model in machine translation using a continuous two-layer self-attention mechanism to enhance the accuracy of semantic understanding during translation. Literature [13] introduced a multi-head self-attention mechanism and a bi-directional long and short-term memory network to optimize the encoder-decoder model for text sentiment analysis, which enabled the model to obtain the highest accuracy rate of 90.34%. Literature [14] designed a capsule network text categorization method based on the self-attention mechanism, which enhances the performance of text feature extraction in the network by enhancing the self-attention mechanism. Literature [15] added the self-attention mechanism to the Transformer model, which effectively solved the problem of common references in abstract text, enhanced the model's ability to understand the text, and effectively realized the abstract text summarization task. Literature [16] utilized selectively connected self-attention to improve the recurrent update problem in deep neural networks for semantic role annotation, accelerating the training speed of the model. Literature [17] constructed a model for semantic relation extraction of medical text supported by multi-hop self-attention mechanism, which utilizes the self-attention mechanism to generate weights for semantic vectors as a way to represent different semantic information.

Natural language processing is an important research direction in the field of artificial intelligence, and with the explosive growth of large-scale text data, how to efficiently process and understand text information has become a key challenge in current research. Traditional natural language processing models often face the problems of information loss and high computational complexity when dealing with long text sequences, especially in the encoder-decoder framework, the fixed-length context vectors are difficult to retain the semantic information of long sequences in full, and they cannot distinguish the importance of words, which leads to limited model performance. In this paper, we propose a natural language processing model based on the self-attention mechanism from two directions, namely, optimizing text representation and feature extraction, and efficiently constructing it in a low-dimensional embedding space. First, for text representation, this study adopts the vector space model (VSM) to describe the text set and constructs word vectors using the statistical information between words, which is trained by the Glove model to effectively capture the syntactic and semantic information of words. Meanwhile, in order to solve the overfitting problem brought by high-dimensional word vectors, Principal Component Analysis (PCA) is introduced for dimensionality reduction, which significantly reduces the computational complexity while retaining the main information of the text. For feature extraction, this model combines the bidirectional LSTM network and the self-attention mechanism to make full use of the contextual information of the text, and gives different weights to different words through the attention mechanism, so as to capture the semantic relationship of the text more accurately. In terms of model structure, the study designs a multi-layered architecture, including input layer, embedding layer, BiLSTM network layer, and self-attention layer, each of which has its own specific function, and together constitute a complete natural language processing system. Through experiments on multiple datasets, the effects of parameters such as cumulative contribution rate of principal component eigenvalues, Dropout value, convolutional kernel size combination, and the size of BiLSTM hidden layer on the model performance are analyzed in depth, and the validity of the layers' structure is verified through ablation experiments.

II. Natural Language Processing Modeling

II. A. Low-dimensional embedding space

II. A. 1) Word vectors

Computers cannot directly deal with problems related to natural language, therefore, it is usually necessary to encode language into numerical form first, and then apply mathematical rules to perform matrix operations on them, this encoding is also known as word embedding. Usually, the vector representations of words with similar meanings in the embedding space should be close. Therefore, when constructing a word embedding space, the primary goal is to capture some kind of relationship in that space, whether it is a relationship between meanings, morphology, context, semantic and syntactic similarity or other words. It follows that the effectiveness of natural language processing tasks also depends heavily on the effectiveness of word embeddings.

In this paper, we use Vector Space Model (VSM) to describe the text set. The basic idea of Vector Space Model is to represent the text in terms of vectors, and each text d_i can be mapped to a feature vector in this space:

$$V(d_i) = \left((t_{i1}, w_{i1}), (t_{i2}, w_{i2}), \cdots, (t_{in}, w_{in})\right) \tag{1}$$

where t_{ik} is the feature term corresponding to d_i and w_{ik} is the weight of the i th feature term. However, the selection of feature terms is generally words or phrases, so text preprocessing is the first prerequisite for text representation.

For two words that appear consecutively, if the number of their consecutive occurrences is relatively large compared to the number of their respective occurrences, then it is very likely that these two words are a new word or phrase, which can be represented as:

$$score\left(w_i, w_j\right) = \frac{Count\left(w_i, w_j\right)}{Count\left(w_i\right) Count\left(w_j\right)} \tag{2}$$

If the value is greater than a threshold, it can be assumed that two consecutive occurrences of the word can be synthesized into a new word. For Chinese word segmentation, this approach simply uses statistical information and does not utilize the syntactic and semantic information of words.

Definition 1: The statistical information matrix of the number of co-occurrences of words in the corpus is X, where X_{ij} denotes the number of times the word w_j occurs in the context of the word w_i.

Definition 2: X_i denotes the number of times all words occur in the context of the word w_i, specifically $X_i = \sum_k X_{ik}$.

Definition 3: The frequency of occurrence of the word w_j in the context of the word w_i is P_{ij}, specifically $P_{ij} = X_{ij} / X_i$.

Definition 4: The word vector matrix is W, where $W \in R^{|V|*d}$, $|V|$ denotes the number of words, and d denotes the word vector dimension.

In addition to the need to utilize the statistical features of word co-occurrence, the training model for word vectors must be efficient and suitable for distributed training. Therefore, the Glove model, which has no hidden layer and has a simpler objective function, is chosen as the training model for word vectors. The objective function for word vector training is:

$$J(W) = \sum_{ij}\left(W_{Ti} W_j - \log X_{ij}\right)^2 \tag{3}$$

In order to remove some low-frequency term noise, a weighting term $f\left(X_{ij}\right)$ is introduced to the objective function, and the weighting term $f\left(X_{ij}\right)$ is chosen as one of the following functions:

$$f\left(X_{ij}\right) = \left(X_{ij} / X_{\max}\right) \tag{4}$$

The final objective function obtained is:

$$J(W) = \sum f\left(X_{ij}\right)\left(W_{Ti} W_j - \log X_{ij}\right)^2 \tag{5}$$

The word vectors obtained by training such a word vector model contain good syntactic and semantic information and can be easily used for new word discovery. In addition, the objective function in Eq. (5) is well suited for distributed training.

II. A. 2) PCA downscaling

The main idea of principal component analysis is to try to recombine the original category features into a new set of mutually unrelated several comprehensive category features to replace the original category features, while according to the actual need to take a few less comprehensive category features as much as possible to respond to the original category features of the information [18].

The dimensionality reduction steps of principal component analysis are as follows:

(1) Compute the covariance matrix $S = \{s_{ik}\}$ of the word frequency-document matrix D.

(2) Compute the eigenvalues λ and eigenvectors e of S.

(3) Arrange the eigenvalues in descending order $(\lambda 1 \geq \lambda 2 \geq \cdots \geq \lambda n)$, and select the corresponding eigenvalues in turn to construct the eigenvector matrix.

(4) Select p important principal components, the selection of p is mainly based on the cumulative contribution rate of the principal components, i.e., the cumulative contribution rate is generally required to reach 85% or more, so as to ensure that the composite variable can include most of the information of the original variable.

(5) Calculate the principal component scores to achieve the purpose of dimensionality reduction.

II. B.Attention mechanisms

II. B. 1) Feasibility of attention mechanisms

In the last few years in the field of deep learning, attention mechanisms have been frequently applied to various models to deal with natural language tasks. First applied in the field of computer vision, the attention mechanism

was then introduced for the first time to the field of natural language processing to perform translation and alignment work on machine translation tasks. The attention mechanism allows the model to identify the information that distinguishes the text, and at the same time it can explain what exactly the model has learned by visualizing the alignment between the input and output sentences in the translation task. The recently proposed self-attention mechanism, which does not use convolutional and recurrent neural networks at all and relies only on itself to act as a feature extractor on its own, has become a hot research work recently and is being explored on various natural language processing tasks.

II. B. 2) Principles of Attention Mechanisms

Attention mechanism generates each element in the sequence as three parts, Query, Key and Value, through neural network, for any Query, the degree of similarity or correlation between it and each Key is calculated by the same way, and then weighted and summed according to the weight coefficients of Value, the global Attention value of this element is calculated [19]. Therefore, in simple terms, the Attention mechanism is essentially a way to deal with overloaded information, by assigning different weights to each part to filter more useful and effective information, the structure of the Attention mechanism is shown in Figure 1. As can be seen from Figure 1, the process of calculating Attention values by the attention mechanism can be divided into three steps. The first step is to calculate the similarity between Query and each Key to get the weight coefficient through the function, and the commonly used similarity functions are dot, general, concat, perceptron, etc., as shown in Equation (6):

$$f\left(Q,K_i\right)=\begin{cases}Q^T K_i, dot\\ Q^T W K_i, general\\ W\left[Q;K_i\right], concat\\ V^T \tanh\left(WQ+UK_i\right), perceptron\end{cases} \tag{6}$$

In the second step, all the weight coefficients of the same Query are subjected to a normalization operation, which unifies and sums the weight scores in different ranges to a probability distribution of 1, highlighting the elements that are more closely related to this element, calculated as shown in Equation (7):

$$a_i = soft\ \max\left(f\left(Q,K_i\right)\right)=\frac{\exp\left(f\left(Q,K_i\right)\right)}{\sum_i \exp\left(f\left(Q,K_i\right)\right)} \tag{7}$$

In the third step, the weights and the corresponding Values are weighted and summed to get the final Attention value as shown in Equation (8):

$$Attention\left(Q,K,V\right)=\sum_{i=1}^{I} a_i V_i \tag{8}$$

Currently, the vast majority of computational processes for the types of attention mechanisms use the above three steps. The Key and Value of an attention mechanism in a natural language processing task are often the same, i.e. $Key = Value$.

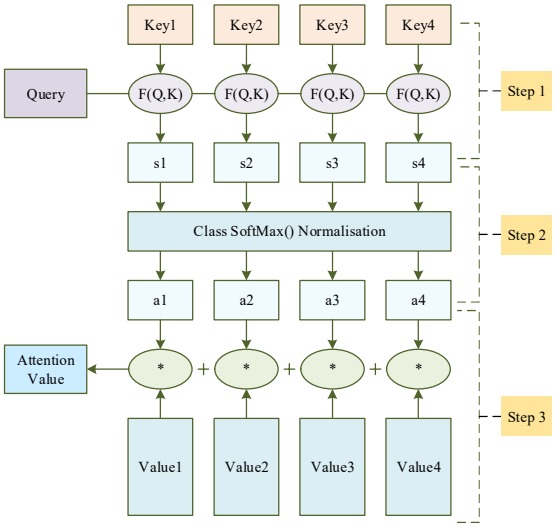


Figure 1: Attention mechanism structure

II. B. 3) Attention mechanism applications

Currently, the encoder-decoder framework is widely used to solve end-to-end (Seq2Seq) problems, and as a result, it has achieved great success in tasks such as machine translation, speech recognition, text summarization, and question-answer systems. The structure of the encoder-decoder framework is shown in Fig. 2, which consists of three parts: an encoder, an intermediate vector, and a decoder, which has a sequence of inputs and a sequence of outputs. The input sequence is encoded into an intermediate vector by the encoder, and the intermediate vector is then decoded into an output sequence by the decoder, which has a fixed length. In practice, the encoder and decoder can be chosen from neural network models such as CNN, RNN, LSTM and GRU, which are not fixed, e.g., RNN is used for encoding and LSTM is used for decoding.

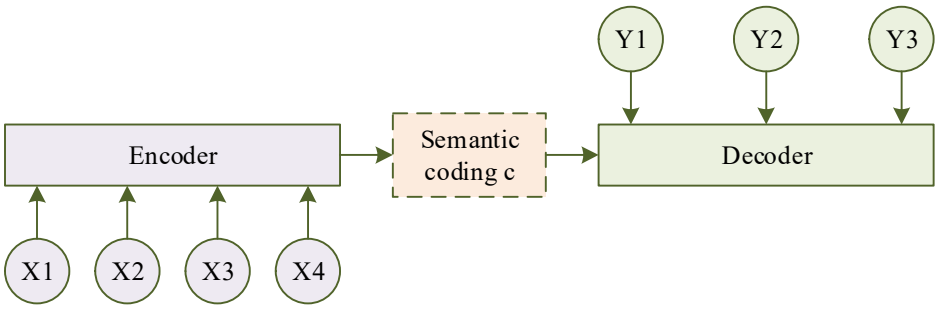


Figure 2: Encoder-decoder structure

In the first step, the encoder encodes the input sequence $x=(x_1,x_2,\cdots,x_T)$ into a context vector c. Assuming that an LSTM is chosen for the encoder, the state output of the current time step is computed by Equation (9), which requires the state output of the previous time step and the input information of the current time step. After all the input information is computed, the state outputs of all time steps are combined to generate the context vector c as shown in Equation (10). The vector c encodes the information of the whole sentence and is a representation of the semantic information of the whole sentence understood. Namely:

$$h_t = f(x_t, h_{t-1}) \tag{9}$$

$$c = q(h_1, h_2, \cdots, h_T) \tag{10}$$

In the second step, assuming that LSTM is also chosen for the decoder, the inputs of the t time step are the vector c and the decoded output s_{t-1} of the previous time step and the predicted word vectors y_{t-1} of the previous time step, which are computed as shown in Eqn. (11), where the function g represents the decoder hidden layer transform. Then in the prediction of y_t based on the vector c and all the predicted words $(y_1,y_2,\cdots,y_{t-1})$, the computation is shown in Eq. (12). Namely:

$$s_t = g(s_{t-1}, y_{t-1}, c) \tag{11}$$

$$p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, y_2, \cdots, y_t\}, c) = q(y_{t-1}, s_t, c) \tag{12}$$

The framework of encoder-decoder can be used to flexibly choose and combine various neural network models, which can solve many difficult problems, such as sequence prediction. However, it is also obvious that the framework has two problems, one is the length of the context vector, a fixed length can only contain a certain amount of information, when the input sequence is relatively short, the vector c extracts all of its information, but when the input sequence is relatively long, the vector c can only discard a part of the information, which results in incomplete decoding of the information, and thus affects the effectiveness of the model. Another problem is the weighting of the input sequence, where each word in the input sequence has the same weight, resulting in the inability to distinguish the importance of the words. For example, one word in the input sequence may correspond to several words in the output sequence, and assigning a higher weight to this word can only improve the model's effect.

Therefore, in order to solve the problem of long sentences and weights, an attention mechanism is introduced to extend the encoder-decoder framework to jointly learn translation and alignment information, as shown in Fig. 3. The input sequence is encoded by the encoder to generate the weighted sum of all words, i.e., the context vector c. Then, when generating the output sequence, the input sequence can participate in the decoding of the output sequence to different degrees. Therefore, by incorporating an attention mechanism into this framework, a connection can be established between the words of the input and output sequences, from which their alignment matrix can be obtained. By visualizing the alignment matrix, it is possible to see the distribution of importance of

each word of the input sequence to the current word to be translated when translating a word, enhancing interpretability.

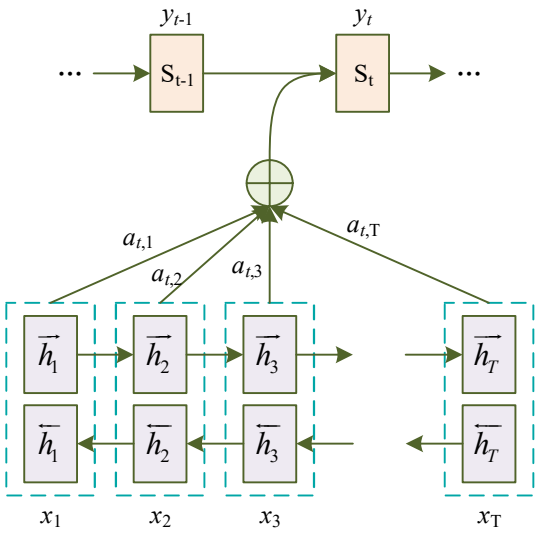


Figure 3: Generate the TTH word

II. B. 4) Types of Attention Mechanisms

Attention mechanisms are categorized into many types to suit different scenarios depending on the computational area employed, information used, structural level and modeling. Global attention mechanism, hard attention mechanism, local attention mechanism and self-attention mechanism are frequently used methods, while self-attention mechanism is used in this paper.

Self-attention mechanism is the most common computation method and its structure is shown in Fig. 4. The self-attention mechanism considers the hidden state outputs of the encoder at all moments while computing the context vector. This approach comprehensively refers to the information of all elements in the input sequence, but the disadvantage is that the weighted computation is larger when the input sequence is longer. Because the global attention mechanism is smooth and microscopic, then it is possible to backpropagate the gradient to the rest of the model, and therefore it can be embedded into the model for direct training.

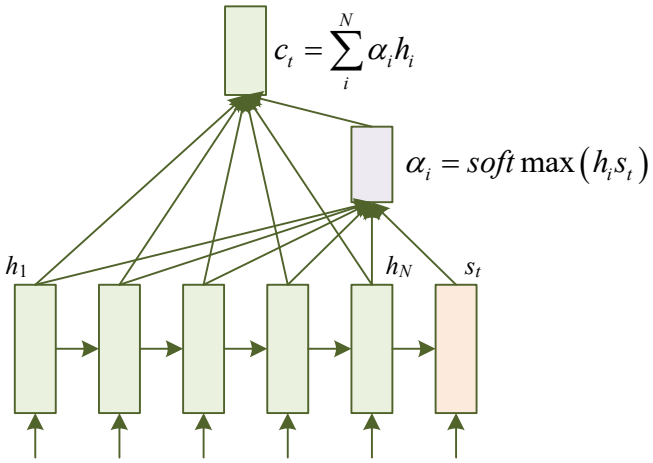


Figure 4: Self-attention mechanism model

II. C.Language processing model based on self-attention mechanism

II. C. 1) Input layer

For the Chinese paper using jieba participle tool for the Chinese text for the word, jieba participle tool is characterized by simple operation, high accuracy, speed, in addition, it can also provide users with lexical annotation

Chinese participle, extract keywords, add custom dictionaries and many other functions. After the completion of the word, the next major operation should be for the user to create a Chinese word mapping of a relationship table, generally the first corpus of all Chinese words that have appeared in the statistics, and then build a dictionary of words, so that a sentence is converted to a string of numbers, different numbers represent different words.

II. C. 2) Embedding layer

The most important role and function of this layer is to map and embed the words in a question and their context into a vector space. This paper uses the above based on PCA-VSM.VSM is a neural network based tool for generating word vectors because the model has been learned from a large number of high quality documents, which makes the model contain a large amount of semantic information. Using such word vectors, the various operations in them are simplified and can be applied to lexical annotation and so on. For word vectors, the dimension can be set as a parameter according to the situation, and the choice of dimension is generally based on the needs and personal experience to complete the evaluation, such as node classification. The range of dimensionality is usually from tens to hundreds of dimensions, and there is an optimal dimensionality. If the dimension is too low, then it does not have high learning ability, and if the dimension is too high, then it is easy to fit, which adopts PCA arithmetic to reduce the dimension, so that it avoids the phenomenon of fitting.

II. C. 3) BiLSTM network layer

The bi-directional LSTM, BiLSTM, is a combination of a forward LSTM and a backward LSTM.The forward state of the LSTM gets part of the input information of the previous semantics with the feature values, while the next state gets part of the input information of the subsequent semantics with the feature values, so that more information about the features is captured directly. The input information of all moments of this structure is processed in two simultaneous bidirectional LSTM units, and the computational process needs to take into account the information content of them.

If the forward output at moment t is $\vec{h}_t$, the formula is $\tilde{\vec{c}} = \tanh\left(\vec{W}_{xc}x_t + \vec{W}_{hc}\vec{h}_{t-1} + \vec{b}_c\right)$ The input gate $\vec{i}_t$, the forgetting gate $\vec{f}_t$, and the output gate $\vec{o}_t$ are shown in Equation (13):

$$\begin{aligned}\vec{i}_t &= \sigma\left(\vec{W}_{xi}x_t + \vec{W}_{hi}\vec{h}_{t-1} + \vec{W}_{ci}\vec{c}_{t-1} + \vec{b}_i\right)\\ \vec{f}_t &= \sigma\left(\vec{W}_{xf}x_t + \vec{W}_{hf}\vec{h}_{t-1} + \vec{W}_{cf}\vec{c}_{t-1} + \vec{b}_f\right)\\ \vec{o}_t &= \vec{W}_{xo}x_t + \vec{W}_{ho}\vec{h}_{t-1} + \vec{W}_{co}\vec{c}_{t-1} + \vec{b}_o\end{aligned} \tag{13}$$

Inside the above equation: x_i corresponds to the input under the t moment. $\vec{c}_{t-1}$ corresponds to the forward input state under the $t-1$ moment cell. $\vec{h}_{t-1}$ corresponds to the forward input state under the hidden layer at moment $t-1$. b corresponds to the bias term. The W corresponds to the parameter matrix.

The input gate mainly controls the forward input generated at moment t. The forgetting gate is mainly to control the forward input generated at the moment $t-1$, thus affecting the input situation at the previous moment. The output gate is based on the input t-moment, based on its eigenvectors, so that the computational equations for the state $\vec{c}_t$ and $\vec{h}_t$ outputs can be known as shown below:

$$\begin{aligned}\vec{c}_t &= \vec{i}_t * \tilde{\vec{c}}_t + \vec{f}_t * \vec{c}_{t-1}\\ \vec{h}_t &= \vec{o}_t \tanh\left(\vec{c}_t\right)\end{aligned} \tag{14}$$

The backward output $\overleftarrow{h}_t$ is computed similarly to the forward output as a whole, and the place to pay special attention to is that the dimension of the forward output vector must be the same as that of the vector of the backward output, because the final t moment of the output result, h_t, is determined by the forward output $\vec{h}_t$ and the backward output $\overleftarrow{h}_t$ are jointly computed as in (15):

$$\vec{h}_t = D*\left(\vec{h}_t \oplus \overleftarrow{h}_t\right) \tag{15}$$

Inside the above equation: $\oplus$ means that two different element vectors corresponding to the same position are summed. D means that the dropout operation is performed based on the final output result so as to avoid overfitting phenomenon.The output result of BiLSTM network layer is $H = \left[h_1, h_2, \cdots, h_t\right]$. Since the output of the upper network at all moments is used as the input of the lower network, the hidden layer state h_t output at each

moment contains contextual information, so H is the feature vector of the BiLSTM network layer after feature extraction and feature combination.

II. C. 4) Self-attention layer

The calculation steps of the self-attention mechanism: the similarity between elements and keywords is calculated and analyzed using the relationship between them. Then, the similarity and the score are normalized, and then the weighted sum is calculated, so that the corresponding self-attention weight of a certain value can be obtained.

First, for a keyword and element to introduce the relevant calculation methods and functions, and then the existence of the two are calculated, the specific formula is as follows:

Cosine similarity method calculation formula:

$$Sim\left(Q,K_i\right)=MLP\left(Q,K_i\right) \tag{16}$$

Dot product method formula:

$$Sim\left(Q,K_i\right)=Q\cdot K_i \tag{17}$$

Multi-Layer Perceptron Machine Method formula (MLP):

$$Sim\left(Q,K_i\right)=\frac{Q\cdot K_i}{\left\|Q\right\|\cdot\left\|K_i\right\|} \tag{18}$$

Inside the above equation: K corresponds to the keyword, Q corresponds to the element value, S corresponds to the data source, and Sim corresponds to the computation of similarity, where the weight coefficients are specifically represented through a and the weight values are presented through V. In the above equation, when $Q=K=V$, it means that when the statement is input, all the words in this statement are computed with the weight values of the self-attention mechanism with respect to other words.

Next, through the Softmax function for the above weights and similarity to calculate, complete the normalization operation, you can get the coefficient a_i, the function's role includes the following two: First, to the original vector of all the values as the object to carry out the normalization process, and then all the scores are processed, into a probability distribution of the weight of the integrated 1. Second, the use of such operations, can be more prominent important parameter elements of the weight and similarity, the specific calculation formula for:

$$a_i=Soft\max\left(Sim_i\right)=\frac{e^{Sim_i}}{\sum_{j=1}^{L}e^{Sim_i}} \tag{19}$$

Finally, the weighted sum of a_i and V_i is obtained as the self-attention value, and the formula is shown in (20):

$$A=\left(Q,S\right)=\sum_{i=1}^{L}a_i\cdot V_i \tag{20}$$

II. C. 5) Loss function

The loss function is one of the keys to enable the network to automatically learn relevant features, and the optimization algorithm continuously updates the direction of gradient descent through the process of minimizing the loss function, which in turn goes through the back-propagation mechanism of the neural network, updating the parameters of each layer of the network until the results converge.

In this paper, we mainly use a variant form of the hinge loss function, and the folding error loss function is the loss function of the maximum error error objective. In the field of machine learning, this loss function is often widely used in the maximum error interval for classification, often used as the maximum objective loss function. After the above calculation and analysis, it can be known what degree of similarity has been achieved between the answer and the question to get a specific score. As shown in equation (21):

$$L(y)=\max\left(0,1-\max\left(y*y'\right)\right) \tag{21}$$

Inside the above equation: y' is the predicted value, when $y'\le -1$ or $y'\ge 1$, it means that the classifier has not classified correctly and the loss function $=0$. If $y'\in(-1,1)$, the classifier is not sure of the result, and the loss function size is not equal to 0. From this, it can be seen that $\tilde{y}=0$, the loss function is the maximum value.

III. Exploring Natural Language Processing Models

III. A. Example analysis of low-dimensional embedding spaces

III. A. 1) Experimental environment

The experimental machine is selected Intel fifth-generation I7-4590 CPU, 32G RAM, the graphics card is GTX750Ti, 4GB video memory with 256G SSD solid state disk, the experimental system is Linux operating system.

III. A. 2) Evaluation indicators

The experiments use the accuracy metric (Accuracy) to measure the performance of the evaluation algorithm, which is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{22}$$

where, TP denotes the number of actual positive and predicted positive samples, FN denotes the number of actual positive and predicted negative samples, TN denotes the number of actual negative and predicted negative samples, and FP denotes the number of actual negative and predicted positive samples.

In order to verify the positive impact of PCA on the time complexity of word vector models, this paper also introduces the training time overhead as an important reference index.

III. A. 3) Experimental data sets

In order to verify the effect of low-dimensional embedding space experiments based on principal component analysis and word vectors, the experiments use two datasets, MRD and dataset SST, which are applied to the study of low-dimensional embedding space for natural language processing models. Among them, MRD consists of data with positive attitude comments accounted for about 2,500, negative attitude comments 2,500, 5,000 sentences each labeled with positive and negative polarity, and 5,000 sentences labeled with subjective and objective labels, in this paper's experiments, 2,000 randomly selected as a training set, 800 as a testing set, and 400 as a validation dataset.The SST dataset consists of 12,000 sentences, 7500 and 3000 sentences are extracted as training and test sets respectively, and 1500 sentences are selected as validation set. The ratio of training set, test set and validation set for the two datasets is 5:2:1 respectively.

III. A. 4) Analysis of experimental results

(1) Analysis of the effect of PCA on training time

In order to verify the positive impact of PCA on word vector training, we analyzed different word vector models completing all the experiments under the same CPU, GPU and framework, where the models are PCA-VSM (word vector model based on Principal Component Analysis - Vector Space), Word2VEC (neural network based word vector model), TF-IDF (based on word frequency - Inverse Document Frequency), BERT (word vector model based on bidirectional encoder representation), Doc2Vec (word vector model based on unsupervised distributed documents), ELMO (word vector model based on deep contextualization), while other hyperparameters, such as the word vector construction method, remain consistent. Figure 5 gives the results of the training time comparison of different word vector models to complete one iteration on the dataset, the horizontal axis is the model, and the vertical axis is the training time.The training time cost of ELMO is very high, which is mainly due to the fact that the deep learning network is trained on the sequential data, and its training time for one iteration of the dataset MRD and the dataset SST is 295s and 480, respectively. Moreover. Our proposed PCA-VSM changes the original redundant word vector model, and the training time of the VSM on the two datasets is 107s and 104s respectively after incorporating the PCA dimensionality reduction.Thus, the positive impact of PCA on the training overhead of the word vector model is verified, and the practical application of the low-dimensional embedding space based on Principal Component Analysis and word vectors is also illustrated.

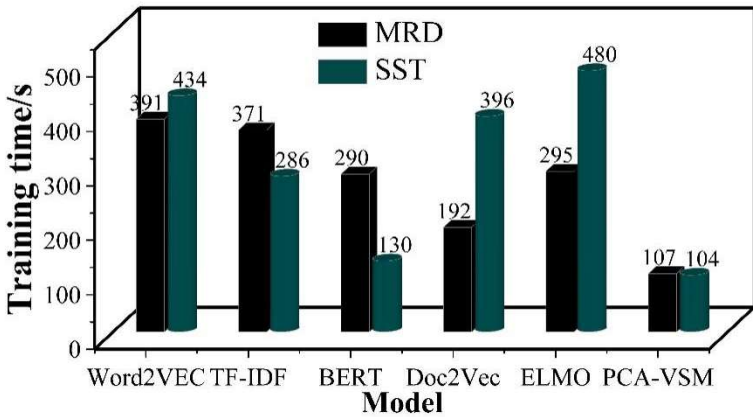


Figure 5: Comparison results of training time

(2) The effect of the cumulative contribution rate of eigenvalues on the accuracy of word vector models

In order to study the influence of the cumulative contribution rate of the eigenvalue values of the principal components on the accuracy of the word vector model, this paper analyzes the relationship between the different cumulative contribution rate on the change of the accuracy of the word vector model under the dataset, and the relationship between the accuracy rate and the cumulative contribution rate of the eigenvalues is shown in Figure 6. As can be seen from Figure 6, let the cumulative contribution rate of feature values be α, when α goes from 75% to 100%, the text features extracted by the VSM model are dimensionality reduced by PCA, and the redundancy in the text features is gradually removed, and the accuracy is increased; when $\alpha = 90\%$, the redundancy in the text features is more sufficiently removed, and the accuracy is also the highest. The accuracy is also the highest. When α decreases from 90%, some of the text used is also eliminated, resulting in a decrease in accuracy. From this, it can be seen that the dimension of the principal components is crucial to the accuracy.

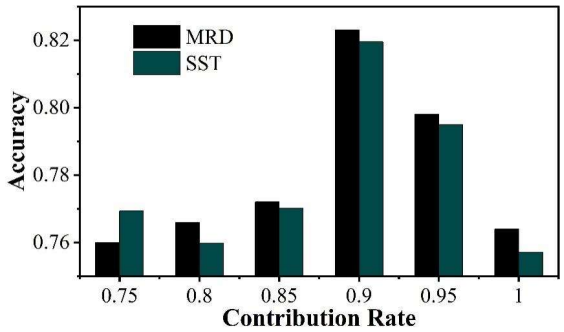


Figure 6: The relationship between accuracy rate and contribution rate

III. B. Model Example Analysis

III. B. 1) Parameter-to-model performance analysis

The determination of hyperparameters causes some degree of variation in the experimental results, and in order to improve the performance of the natural language processing model, the Dropout value, kernel size, and the size of the BiLSTM hidden layer are further investigated in this subsection.

(1) Effect of Dropout value on model performance

In the training process of natural language processing models, in order to prevent overfitting, the Dropout mechanism is introduced. The principle of Dropout is to make some neural network units temporarily discard them from the network according to a certain chance during the training process of the model. Dropout not only improves model generalization, but also prevents model overfitting and speeds up convergence. In order to investigate the effect of Dropout on the model performance, experiments are carried out on the MRD dataset and SST dataset, and the Dropout values are selected as 0.2, 0.3, 0.4, 0.5, 0.6, and 0.7, respectively, and the results of the experiments are shown in Fig. 7. From the experimental comparison results, it can be concluded that the accuracy of the model on the MRD dataset and the SST dataset shows a tendency of first increasing and then decreasing. When Dropout is set to 0.5, its classification accuracy performs the best, which can alleviate overfitting and help optimize the performance of the model, indicating that the model has a better classification effect at this time.

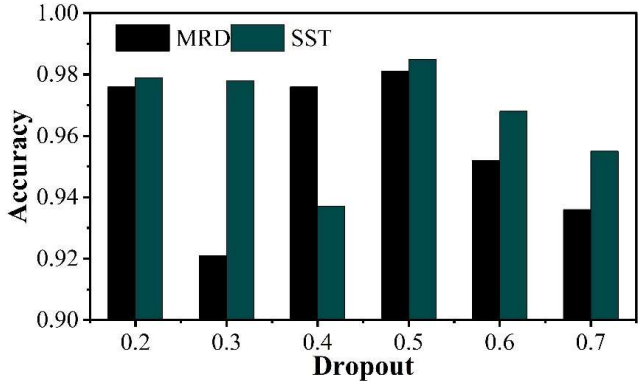


Figure 7: The influence of Dropout values on model performance

(2) Impact of kernel size combinations on model performance

Convolutional kernels of multiple sizes in the natural language processing model extract local feature information of text at different granularities, and different convolutional kernel sizes and number of convolutional kernels will have different degrees of influence on the classification effect. In order to explore the impact of different combinations of convolutional kernel sizes on the model performance, it is set to five different combinations, namely, F (2, 3, 4), G (3, 4, 5), H (4, 5, 6), I (2, 3, 4, 5), J (3, 4, 5, 6), and the number of convolutional kernels of each size is set to 300, and the comparison experiments are carried out on two datasets respectively. The experimental results are shown in Fig. 8. Groups G and I have better classification results on the MRD dataset, with classification accuracies of 96.27% and 96.09%, respectively, and the classification accuracy-rates of both Groups G and I on the SST dataset are also better than the other combinations. Among them, the accuracy-rate of group G is higher than that of group D both on the MRD dataset and on the SST dataset. Therefore, in this paper, we set the kernel size combinations of natural language processing models as (3, 4, 5).

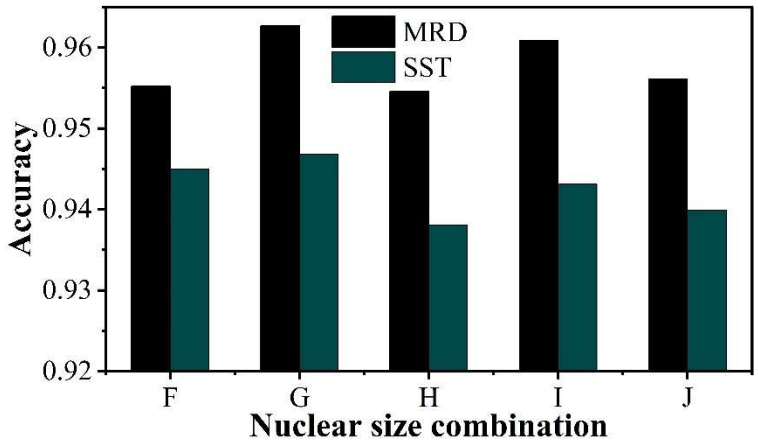


Figure 8: The influence of kernel size combination on model performance

(3) Effect of BILSTM hidden layer size on model performance

In order to investigate the effect of different BiLSTM hidden layer sizes on model performance, a total of seven hidden layer sizes of 100, 200, 300, 400, 500, 600, and 700 were selected for experimental comparisons on two datasets, and the experimental results were analyzed to derive the optimal values of hidden layer sizes. Figure 9 shows the effect of BiLSTM hidden layer size on model accuracy and Figure 10 shows the effect of BiLSTM hidden layer size on model F1. The Acc and F1 values of the model on MRD dataset and SST dataset increase with the increase of hidden layer size, and when the hidden layer size is greater than 600 and less than 700, the hidden Acc and F1 values grow slowly. Although the Acc and F1 values are basically the same when the hidden layer size is 700 and 600, setting the hidden layer size to 700 increases the computational complexity of the model and increases the training time. In this paper, we set the hidden layer size of BiLSTM to 600 from the considerations of computation, training time and effect.

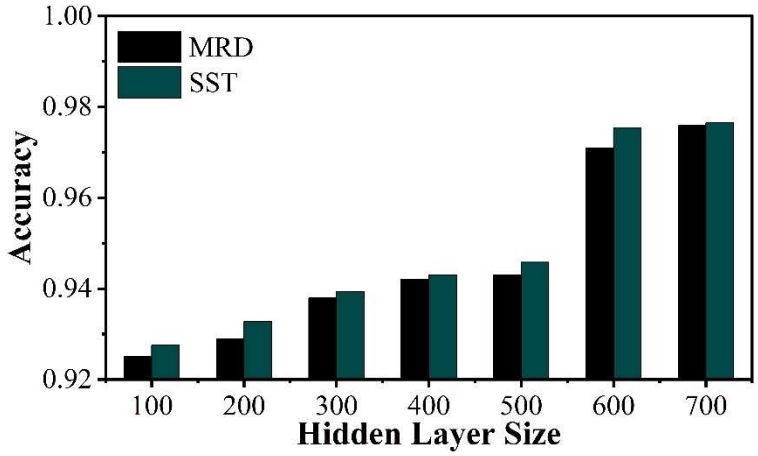


Figure 9: The influence of the hidden layer size of BILSTM on the model accuracy

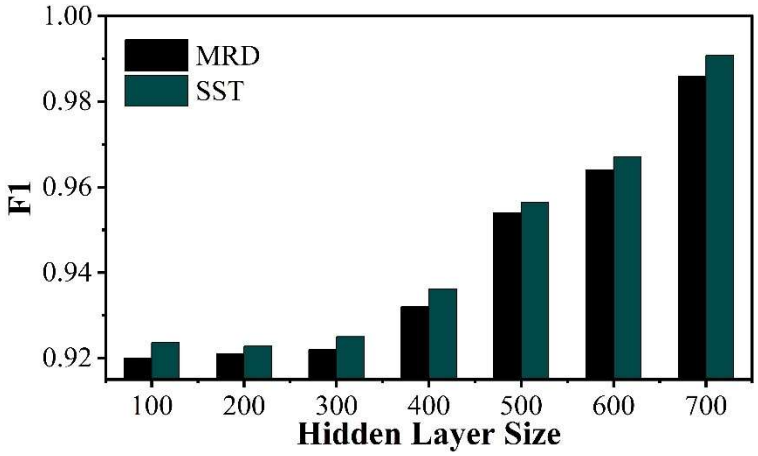


Figure 10: The influence of the hidden layer size of BILSTM on the model F1

III. B. 2) Analysis of loss function changes

The analysis of loss function variation is shown in Fig. 11, where the loss function is the maximum error objective value in the natural language processing model, which can also decrease steadily during the training of the natural language processing model based on the subattention mechanism. The optimizer used in the model is Adam's algorithm which occupies little memory and is simple to implement. The change in the loss function shows that the final loss function is the maximum error objective value and the final loss function decreases smoothly during the training of the model, indicating that the loss functions of the natural language processing models are all decreasing smoothly, further indicating that the model in this paper can be well trained.

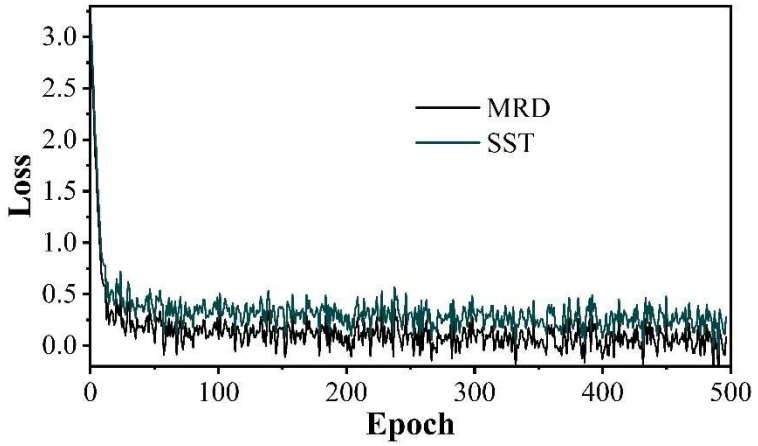


Figure 11: Analysis of Changes in the loss function

III. B. 3) Analysis of ablation experiments

The natural language processing model designed in this paper is a layered structure, and each layer has a different effect on the model, and there are network structures that may also have a negative performance on the model's performance. Therefore, in order to verify the effect of each layer on the model, this paper conducts ablation experiments on the model.

(1) Evaluation metrics

The evaluation metrics are BLEU and Rouge-L (accuracy and recall of the longest common subsequence of the text), both of which are widely used in language processing, and their formulas are as follows:

$$Rouge-L=\frac{\left(1+\beta^2\right)R_{LCS}A_{LCS}}{R_{LCS}+\beta^2 A_{PLS}} \tag{23}$$

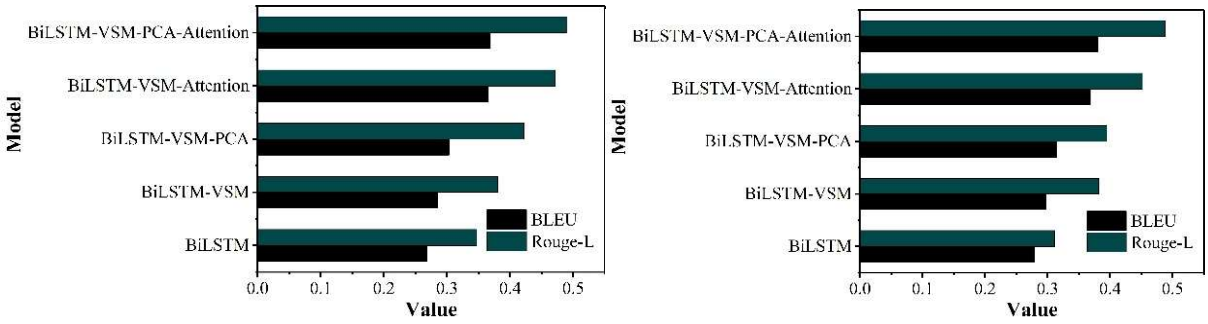
where R and P are the recall and accuracy of the longest common subsequence, respectively, and n is the length of the text sequence, which is taken as $\beta=1.2$ in this paper.

$$BLEU = BP\cdot\exp\left(\sum_{i=1}^{N} w_n \log\left(A_n\right)\right) \tag{24}$$

where $BP=1$, w_n is the weight vector, and A_n is the accuracy.

(2) Analysis of results

Next, ablation experiments were conducted on the natural language processing model of this paper to verify the contribution of each layer in the model. The results of the ablation experiments are shown in Fig. 12, where (a)~(b) denote the datasets MRD and SST, respectively.The results show that the BLEU values of the baseline model BiLSTM on the datasets MRD and SST are 0.268, 0.279, and by adding the principal components, the vector space model, and the self-attention mechanism to the baseline model, the BLEU values reach 0.369, respectively, 0.381, while the corresponding Rouge-L values are 0.489, 0.488, indicating that the principal component, vector space model, and self-attention mechanism have a facilitating effect on the benchmark model BiLSTM, which together form the natural language processing model in this paper and have excellent performance.

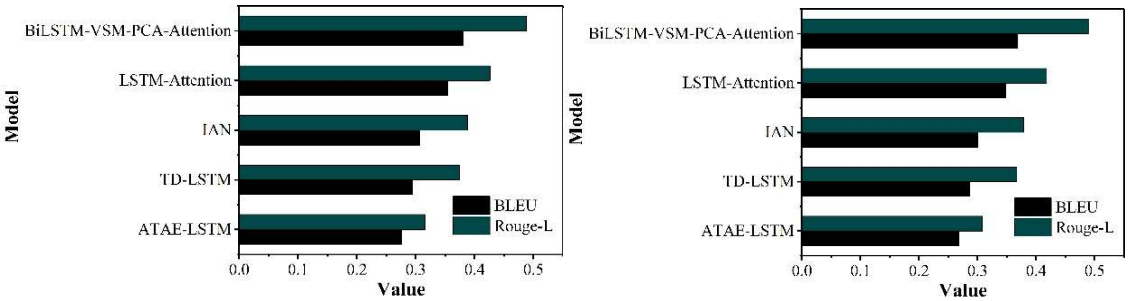


(a)Dataset MRD (b)Dataset SST

Figure 12: Ablation experiment results

III. B. 4) Comparative analysis of models

will be ATAE-LSTM (using two LSTM networks to model the before and after contexts), TD-LSTM (designing aspect embeddings of the target and connecting them to the contextual representation in order to generate the final table using the Long Short Term Memory network and the Attention Mechanism), IAN (aspectual word sequences and contextual sequences are input as word embeddings, and the LSTM network is used to obtain the word in the aspect and context respectively) level hidden states), LSTM-Attention (Sentences are modeled using an LSTM network and the final classification is performed using the self-attention mechanism.) Setting up the comparison models, Figure 13 shows is a comparison between the results of this paper and other mainstream models on the datasets MRD and SST. From the performance of several models on the dataset MRD and SST, it can be seen that BiLSTM-VSM-PCA-Attention performs much better than ATAE-LSTM, TD-LSTM, IAN, and LSTM-Attention, which better illustrates the natural language processing model based on BiLSTM-VSM-PCA-Attention Prioritization.



(a)Dataset MRD (b)Dataset SST

Figure 13: Model Comparative analysis

IV. Conclusion

In this study, an efficient construction strategy for natural language processing models based on self-attention mechanism in low-dimensional embedding space is proposed, and the effectiveness and superiority of the model is verified through experiments. The results show that the training time of the PCA-VSM model on the MRD and

SST datasets is 107s and 104s, respectively, which significantly reduces the training overhead compared with other word vector models such as ELMO (295s and 480s). In terms of parameter optimization, the model performs best on both datasets when the Dropout value is set to 0.5; the classification accuracy reaches 96.27% when the kernel size combination is set to (3,4,5); and the BiLSTM hidden layer size is set to 600, which can reduce the computational complexity while ensuring performance. The ablation experiments further demonstrate that principal component analysis, vector space modeling, and self-attention mechanism all contribute significantly to the benchmark model BiLSTM, improving the BLEU metrics from 0.268 and 0.279 to 0.369 and 0.381 on both MRD and SST datasets.The model comparison study reveals that the proposed BiLSTM-VSM-PCA- Attention model performs much better than mainstream models such as ATAE-LSTM, TD-LSTM, IAN and LSTM-Attention. The comprehensive research results show that the self-attention mechanism based on low-dimensional embedding space can effectively improve the efficiency and accuracy of natural language processing models, providing a feasible technical path for complex text processing tasks.

Funding

This research was supported by the Doctoral Initiation Program, Hanshan Normal University (Project No. QD2023314).

References

[1] He, T., Boudewyn, M. A., Kiat, J. E., Sagae, K., & Luck, S. J. (2022). Neural correlates of word representation vectors in natural language processing models: Evidence from representational similarity analysis of event‐related brain potentials. Psychophysiology, 59(3), e13976.
[2] Bourahouat, G., Abourezq, M., & Daoudi, N. (2024). Word embedding as a semantic feature extraction technique in arabic natural language processing: an overview. Int. Arab J. Inf. Technol., 21(2), 313-325.
[3] Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A survey of text representation and embedding techniques in nlp. IEEE Access, 11, 36120-36146.
[4] Worth, P. J. (2023). Word embeddings and semantic spaces in natural language processing. International journal of intelligence science, 13(1), 1-21.
[5] Singh, K. N., Devi, S. D., Devi, H. M., & Mahanta, A. K. (2022). A novel approach for dimension reduction using word embedding: An enhanced text classification approach. International Journal of Information Management Data Insights, 2(1), 100061.
[6] Liu, B. (2025). An Efficient Construction Method for Matrix Decomposition-Based Natural Language Processing Models in Low-Dimensional Embedding Space. J. COMBIN. MATH. COMBIN. COMPUT, 127, 3319-3337.
[7] Wang, B., Fei, T., Kang, Y., Li, M., Du, Q., Han, M., & Dong, N. (2020). Understanding the spatial dimension of natural language by measuring the spatial semantic similarity of words through a scalable geospatial context window. PloS one, 15(7), e0236347.
[8] Huertas-García, Á., Martín, A., Huertas-Tato, J., & Camacho, D. (2023). Exploring dimensionality reduction techniques in multilingual transformers. Cognitive Computation, 15(2), 590-612.
[9] Jiang, Y., Aragam, B., & Veitch, V. (2023). Uncovering meanings of embeddings via partial orthogonality. Advances in Neural Information Processing Systems, 36, 31988-32005.
[10] Zhao, D., Wang, C., Gao, Y., Shi, Z., & Xie, F. (2021). Semantic segmentation of remote sensing image based on regional self-attention mechanism. IEEE Geoscience and Remote Sensing Letters, 19, 1-5.
[11] Brauwers, G., & Frasincar, F. (2021). A general survey on attention mechanisms in deep learning. IEEE Transactions on Knowledge and Data Engineering, 35(4), 3279-3298.
[12] Liu, H. I., & Chen, W. L. (2022). X-transformer: a machine translation model enhanced by the self-attention mechanism. Applied Sciences, 12(9), 4502.
[13] Xia, H., Ding, C., & Liu, Y. (2020). Sentiment analysis model based on self-attention and character-level embedding. IEEE Access, 8, 184614-184620.
[14] Yu, X., Luo, S. N., Wu, Y., Cai, Z., Kuan, T. W., & Tseng, S. P. (2024). Research on a Capsule Network Text Classification Method with a Self-Attention Mechanism. Symmetry, 16(5), 517.
[15] Kumar, S., & Solanki, A. (2023). An abstractive text summarization technique using transformer model with self-attention mechanism. Neural Computing and Applications, 35(25), 18603-18622.
[16] Park, J. (2019). Selectively connected self-attentions for semantic role labeling. Applied Sciences, 9(8), 1716.
[17] Zhang, T., Lin, H., Tadesse, M. M., Ren, Y., Duan, X., & Xu, B. (2021). Chinese medical relation extraction based on multi-hop self-attention mechanism. International Journal of Machine Learning and12 Cybernetics, 12, 355-363.
[18] Na Wang,Feng Li & Xiaodong Fan. (2025). Quality assessment of Bupleurum chinense by coupling Belousov-Zhabotinsky oscillation with principal component analysis. International Journal of Electrochemical Science,20(7),101033-101033.
[19] Zhang Zhongzi. (2021). Analysis of Volleyball Video Intelligent Description Technology Based on Computer Memory Network and Attention Mechanism. Computational Intelligence and Neuroscience,2021,7976888-7976888.