Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

An Efficient Construction Method for Matrix Decomposition-Based Natural Language Processing Models in Low-Dimensional Embedding Space

Bo Liu[1,✉]

[1] *School of Mathematics and Statistics, Hanshan Normal University, Chaozhou, Guangdong, 521041, China*

ABSTRACT

Natural language processing (NLP) is developing very rapidly in the field of artificial intelligence, and has become an important direction in the development of computer science field and artificial intelligence industry. In this paper, in order to realize the efficient construction of natural language processing model in low-dimensional embedding space, firstly, a word vector learning model is constructed based on matrix decomposition for word vectors in natural language processing. On this basis, in order to further realize the efficient construction of natural language processing models, this paper designs the Semantic Discarding Network (SDN) and Semantic Fusion Alignment Method (SFA) for the problem of interfering semantics of the model and the problem of a single way of fusion of local inference results. Finally, the SDF-NN natural language processing model is proposed and a multi-view subspace clustering (DLTE) method based on deep low-rank tensor embedding is proposed. The results of the research experiments show that the average performance index of this paper's word vector model for each task in three corpora ranges from 71.55 to 89.11, and the performance is stable and the time overhead in the three corpora is 3.93, 7.29, and 13.42 minutes, respectively, and the speed of the model has been significantly improved and the overall performance is better. In addition, the natural language processing model (SDF-NN) constructed in this paper achieves the best performance in the comparison test with strong competitiveness, which further validates the performance of the matrix decomposition-based natural language processing model in this paper, and provides the method and direction for its efficient construction in low-dimensional embedding space.

Keywords: matrix decomposition, natural language processing model, SDN, semantic fusion alignment method SFA, DLTE

✉ Corresponding author.
E-mail address: sirlb@126.com (B. Liu).
Received 25 October 2024; Revised 10 December 2024; Accepted 05 March 2025; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-185

1. Introduction

The most important feature of the digital era is that the continuous progress of digital technology and new development concepts promote the traditional industrial system to gradually get rid of the traditional mode and move towards digitalization [1]. Natural language processing (NLP) technology is one of the most representative and pioneering technologies. The increasingly mature natural language processing technologies, such as content generation, text detection and emotion recognition, promote the emergence of new quality productivity, thus realizing the optimization and upgrading of all aspects of social production [2-4].

Word embedding, as a core step in solving natural language processing tasks, has received increasing attention from researchers in recent years. Its purpose is to convert text data in discrete character format into continuous real-valued vectors that can be processed by the model and contain rich implicit semantic information, i.e., generating word vectors [5-6]. As the foundation of natural language processing tasks, word vectors are the prerequisite for accomplishing all natural language processing tasks [7]. In recent years, word vectors have been widely used in a wide variety of natural language processing tasks, such as named entity recognition, sentiment analysis, and machine translation [8-11]. Word vectors, as a low-dimensional continuous vector represented by words in real number space, can capture both semantic and syntactic information of words, and larger unit-level vector representations during model processing tasks can be obtained by combining word vectors [12-15]. Also, since the objects processed by NLP tasks are text data in character format, machine learning and deep learning models cannot learn directly from text string inputs [16-17]. Therefore, how to convert text data into real-valued vector format that can be processed by the model and embed the rich implicit knowledge information contained in the corpus into the generated word vectors is the focus and difficulty of the word embedding approach [18-20].

Existing methods for learning word vectors can be broadly categorized as neural network based and matrix decomposition. Neural network based learning of word vectors involves building a language model based on the relationship between the context and the target word, and obtaining the word vectors by training the language model.Wadud, M. A. H. et al. showed that pre-trained word embedding vectors based on deep learning cannot be applied to all types of natural language processing tasks, and thus proposed a local word embedding vector generation process for specific word embedding vectors, which, in comparison to the previous one exhibits higher accuracy [21]. AlSurayyi, W. I. et al. compared the performance of word vector embedding methods based on different neural network architectures in the task of comment sentiment analysis, and significantly improved the accuracy of the model in the task of multi-sentiment categorization by converting the comment text into global vector embeddings that can be pre-trained [22]. Yao, H. et al. used word embedding and convolutional neural network to generate sentence vectors and constructed 3D sentence feature tensor by transforming the word vectors in order to solve the semantic dependency problem between words this in English sentence similarity computation task [23]. Xiong, Z. et al. introduced the word vector learning strategy of continuous document subset optimization and vector combinatorial regression on the basis of continuous space language models CBOW and Skip-Gram model, which makes the cosine distance of the generated word vectors more reasonable, and shows good performance in grammar and context capturing for natural language processing tasks [24]. Liu, B. examined bag-of-words language model and deep learning approach in text sentiment analysis task, which utilizes CBOW language model to construct text word vectors and capture their semantic features by convolutional neural network to achieve accurate sentiment judgment [25]. However, most let the natural processing task of learning word vectors based on neural networks shows high accuracy and robustness, but the acquisition of effective word vectors is based on training a large-scale text corpus, which undoubtedly increases the computational cost.

The matrix decomposition-based word vector learning model, which obtains low-dimensional continuous word vectors by decomposing matrices extracted from a text corpus, is also widely used in natural language processing tasks.Qiu, J. et al. found that all the negative sampling models can be unified into a matrix decomposition framework with a closed form, and that the matrix decomposition-based word vector embedding model significantly improves the traditional web mining task's efficiency [26]. Acharya, A. et al. investigated a word vector compression method for natural language processing tasks, compressing the word embedding layer during the training process through a low-rank matrix decomposition technique, which effectively reduces the impact on the classification accuracy during the sentence classification process [27]. Zhu, Z. et al. evaluated an e-commerce recommender system based on a matrix decomposition method of rating data, where semantic features of textual reviews are input into a probabilistic matrix decomposition model to obtain the hidden semantic vectors of users and items, which in turn improves the accuracy of item rating prediction [28]. Xie, R. et al. explored the word embedding coding method for automatic word semantic prediction, which utilizes matrix decomposition technique to learn the semantic relations contained in word vectors, and performs reliable semantic prediction in noise semantics knowledge base annotation validation and neologism phrase annotation suggestion [29]. Liu, N. et al. established a recommender system based on sentiment analysis and matrix decomposition, constructed a scoring matrix integrating user feature matrix and item feature matrix, and quantitatively analyzed the sentiment information in the reviews to make the model have good recommendation performance [30]. Li, W. et al. added linear constraints into the singular value decomposition model based on word co-occurrences, which can flexibly encode both the a priori knowledge of words based on the retained semantic and syntactic information as well as compute the categorical data through matrix decomposition, which significantly improves the categorization performance of the sentences [31]. Therefore, it is necessary to carry out further research on the matrix decomposition-based word embedding model for natural language processing and to look forward to its future development direction.

Word vectors play an important role in natural language processing, so this study is oriented to the efficient construction of natural language processing models, and firstly proposes a word vector learning method (KbEMF) that fuses semantic information, which adds domain knowledge constraint terms to the model of matrix decomposition for learning word vectors, so that word vectors obtained from pairs of words possessing strong semantic relations are relatively approximated. On this basis, Semantic Discarding Network (SDN) and Semantic Fusion Alignment (SFA) methods are designed to realize the optimal construction of natural language processing models, and the multi-view subspace clustering (DLTE) method based on deep low-rank tensor embeddings is used to further improve its constructive effect in low-dimensional embedding space. Finally, the effect of the method model in this paper is proved through experiments, and the performance of the model has obvious improvement.

2. Natural language processing techniques

2.1. Concepts and Processes of Natural Language Processing

Natural Language Processing (NLP) is an important research direction in the field of Artificial Intelligence and in the field of Computer Science. Its research involves establishing various theories and methods that enable effective communication between humans and computers in natural language. Natural Language Processing is an area of interaction between computer science, human linguistics. Intelligent communication with computers using natural language is a goal that has long been pursued in research. Because it has important theoretical significance and very obvious practical significance. People can use computers with their common and habitual language without spending extra time and energy to learn various complicated computer languages. People can also further explore the mechanism of human language and intelligence through this research.

Natural language processing mainly studies the theoretical methods and applications to realize effective communication between human and computer in natural language, and the natural language processing process is shown in Figure 1. The data of language is called corpus, and feature engineering refers to the processing of converting natural language into feature vectors of words (referred to as word vectors), because words are the smallest unit that represents semantics in natural language. The goodness of word vectors directly determines the high performance of the task model, so feature engineering is a crucial step, for which constructing and training a model for word vectors is the optimal solution.

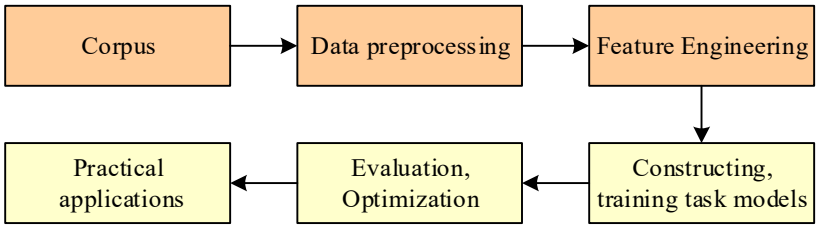

Fig. 1. General process of NLP

2.2. Word Vector

The translation of a natural language understanding problem into a machine learning problem requires the conversion of human-recognizable characters into computer-recognizable mathematical symbols. Thus, the concept of "word vectors" is introduced when used for natural language processing problems. In NLP, the most commonly used word representation is the intuitive One-hot Representation, which represents each word as a vector with certain dimensions, where only one dimension is 1 and the rest of the dimensions are 0. This One-hot Representation uses sparse coding for storage, which is very simple, but there is an important problem. This One-hot Representation uses sparse coding for storage, which is very simple, but there is an important problem, that is, the phenomenon of "lexical gap" will occur. That is to say, when words are based on this kind of representation, it is impossible to find the correlation between words, and the vector representation between any two words is isolated. Therefore, the word vectors used in natural language processing models are not One-hot Representation, but Distributed Representation, which is a kind of low-dimensional real vectors. Simply put, a word vector is a low-dimensional continuous vector of words in real number space, which can capture both semantic and syntactic information of words. The biggest contribution of this representation of word vectors is to make word vectors have distance properties to reflect the correlation or similarity between words. Nowadays, there are many libraries of trained word vectors, which can be directly applied to neural network models dealing with natural language problems in recent years, word vectors have been widely used in a variety of natural language processing tasks, such as named entity recognition, sentiment analysis, and machine translation. Vector representations at a larger unit level (e.g., phrases, sentences, paragraphs, chapters) are often required in processing the above tasks, and these vectors can be obtained by combining word vectors. Therefore it is very important to learn good quality word vectors.

3. Word vector modeling for natural language processing

As the smallest unit to represent semantics in natural language, the goodness of word vectors directly determines the performance of the task model. Therefore, to effectively realize the efficient construction of natural language processing model in low-dimensional embedding space, this paper is first oriented to natural language processing and uses matrix decomposition to realize the optimization of word vector model.

3.1. Matrix decomposition

Before studying word vector models based on matrix decomposition, this section first provides an introduction to matrix decomposition. Matrix decomposition is usually the decomposition of an original high-dimensional matrix into the product of several relatively low-dimensional matrices. This enables high-dimensional data to obtain a low-dimensional data representation, which facilitates high-dimensional data processing. The non-negative matrix decomposition, on the other hand, is to add non-negative restrictions on the basis of matrix decomposition, usually into two non-negative low-dimensional matrices, generally of the form:

$$V = W \times H, W \geq 0, H \geq 0 \tag{1}$$

where $V \in R^{m \times n}$, $W \in R^{m \times k}$, $H \in R^{k \times n}$, and also requires that the elements of W, H are greater than 0. H is called the weight matrix, W is called the basis matrix, and the original matrix V can be viewed as a non-negative weighted sum of the basis matrices. Non-negative matrix factorization is an unsupervised learning algorithm, the idea of the algorithm is to use the product of two non-negative matrices to approximate the original matrix, which can achieve the purpose of matrix dimensionality reduction. Non-negative matrix decomposition is widely used in many fields, such as image processing, semantic recognition, and feature selection.

The conceptual decomposition of a matrix is an extension of the nonnegative matrix decomposition, and the basic form is:

$$X = XWH^T, W \geq 0, H \geq 0 \tag{2}$$

In fact, it is also a form of matrix factorization, which decomposes the original matrix X into the product of three matrices: X, W and H, where W is called the correlation matrix, which forms a certain "concept" by weighting the original matrix, and H is called the encoded matrix, which is approximated by the weighted sum of the "concepts" constructed by W, which is also the origin of the "concept" decomposition. In solving matrices W and H, it is necessary to transform the conceptual decomposition of matrices into an optimization problem, and then solve W and H by solving the optimization problem. Conceptual decomposition of matrices is called a generalization of nonnegative matrix decomposition because it does not require the original matrix to be nonnegative, so it has a wider range of applicability than nonnegative matrix decomposition.

3.2. Matrix Decomposition Word Vector Learning Model with Fused Semantic Information

Existing word vector learning methods utilize the contextual information of a word to predict the meaning of that word, and make words with similar contextual information have similar meanings, so the corresponding word vectors are closer in spatial distance. However, the acquisition of effective word vectors is based on training a large-scale text corpus, which undoubtedly makes the computational cost high. Moreover, in the process of learning word vectors, only the text corpus information is used, but the semantic information between words is ignored, which makes it difficult to guarantee the quality of the obtained word vectors. For this reason, this paper considers extracting semantic information from the domain knowledge base and integrating it into the process of word vector learning, which is corrected by the rich semantic information of the knowledge base, and proposes the KbEMF model, which is based on matrix decomposition to learn word vectors. Its EMF model (Eclipse Modeling Framework) as a framework to add domain knowledge constraints, so that word pairs with strong semantic relations are closer to the learned word vectors in the real number space, that is, more approximate.

3.2.1. Extracting semantic information and constructing a semantic matrix. In this paper, we choose Word Net as an a priori knowledge base.Word Net is a wide-coverage semantic network of English vocabulary, which organizes words with the same meaning into synonym sets, each of which

represents a basic semantic concept, and which are also connected by various relations (e.g., whole-part relations, context relations).

In this paper, we construct a semantic relation matrix $S \in R^{V \times V}$ based on the set of synonyms and the relational words between the sets, and each element $S_{ij} = S\left(w_i, w_j\right)$ of it indicates the semantic correlation between the i th word w_i and the j th word w_j in the vocabulary V. If $S_{ij} = 0$ indicates that words w_i and w_j are not semantically related, and vice versa $S_{ij} \neq 0$ indicates that words w_i and w_j are related. For simplicity, in this paper, the semantic relation matrix S is constructed as a 0-1 matrix, and if words w_i and w_j have the above semantic relation then order $S_{ij} = 1$, otherwise $S_{ij} = 0$.

3.2.2. Modeling semantic constraints. In this paper, we construct the semantic constraint model based on the premise that word pairs with semantic relevance w_i, w_j learned word vectors are more similar and have closer distances in the real number space, and in this paper, we adopt the Euclidean distance of the vectors as a scale to measure the degree of similarity of the word pairs, i.e., $d\left(w_i, w_j\right) = \left\|w_i - w_j\right\|^2$. Therefore, the semantic constraint model can be expressed as follows:

$$\begin{aligned} R &= \sum_{x_i, v_j \in V} S_{ij} \left\|w_i - w_j\right\|^2 = \sum_{i,j=1}^{|V|} S_{ij} \left(w_i^T w_i + w_j^T w_j - 2 w_i^T w_j\right) \\ &= \sum_{i=1}^{|V|} \left(\sum_{j=1}^{|V|} S_{ij}\right) w_i^T w_i + \sum_{j=1}^{|V|} \left(\sum_{i=1}^{|V|} S_{ij}\right) w_j^T w_j - 2 \sum_{i,j=1}^{|V|} S_{ij} w_i^T w_j \\ &= \sum_{i=1}^{|V|} S_i w_i^T w_i + \sum_{j=1}^{|V|} S_j w_j^T w_j - 2 \sum_{i,j=1}^{|V|} S_{ij} w_i^T w_j \\ &= tr\left(W^T S_{row} W\right) + tr\left(W^T S_{col} W\right) - 2tr\left(W^T S W\right) \\ &= tr\left(W^T \left(S_{row} + S_{col} - 2S\right) W\right) \end{aligned} \tag{3}$$

The final resulting semantic constraint model is:

$$R = tr\left(W^T \left(S_{row} + S_{col} - 2S\right) W\right) \tag{4}$$

where $tr(\cdot)$ denotes the trace of the matrix, S_i denotes the sum of all element values of row i of the semantic matrix S, i.e., the sum of row i of S. S_j denotes the sum of all element values of column j of semantic matrix S, i.e., the sum of column j of $\mathbf{S}$. S_{row} denotes the diagonal matrix with S_i as the value of the diagonal element, and S_{col} denotes the diagonal matrix with S_j as the value of the diagonal element.

3.2.3. Model Fusion. The semantic constraint model R is combined with EMF to obtain the matrix decomposition word vector learning model KbEMF that incorporates semantic information:

$$\begin{aligned} O = &-tr\left(X^T C^T W\right) + \sum_{w \in V} \ln\left(\sum_{X'_w \in S_w} e^{X'^T_w c^T w}\right) \\ &+ \gamma tr\left(W^T \left(S_{row} + S_{col} - 2S\right) W\right) \end{aligned} \tag{5}$$

Eq. γ is the semantic combination weight, which indicates the weight size of the semantic constraint model in the joint model. γ plays an important role in the word vector learning process,

and setting this parameter too small will weaken the influence of a priori knowledge on word vector learning, and if it is too large, it will destroy the generality of word vector learning, which is not conducive to the learning of word vectors in either case. The goal of the model is to minimize the objective function O, and the variable alternation iteration strategy is used to find the optimal solution. When $\gamma = 0$ indicates that no semantic information is fused, it is an EMF model.

3.2.4. Model solving. The objective function, i.e., Eq. (5) is not a joint convex function with respect to C and W, but it is a convex function with respect to C or W. Therefore, in this paper, we adopt the variable alternating iterative optimization strategy widely used in matrix decomposition to find the optimal solution of the model. The partial derivatives are obtained for C and W respectively:

$$\frac{\partial O}{\partial C} = \left(E_{X|C^T W} X - X\right) W^T \tag{6}$$

$$\frac{\partial O}{\partial W} = C\left(E_X \mid C^T W X - X\right) + \gamma\left(L + L^T\right) W \tag{7}$$

where: $L = S_{row} + S_{col} - 2S; E_{X|C^T W} X$ located in w rows and c columns has value $E_{X|C^T W} X(w,c) = Q(w,c)\sigma\left(c^T w\right)$, Q located in w rows and c columns has value $Q(w,c) = k\frac{\#(w)\#(c)}{\#(w,c)} + \#(w,c)$, $\sigma(x) = \frac{1}{1+e^{-x}}$. The optimized update strategy in this paper is:

$$W \leftarrow W + \eta\left[C\left(E_{X|C^T W} X - X\right) + \gamma\left(L + L^T\right) W\right] \tag{8}$$

$$C \leftarrow C + \eta\left[\left(E_{X|C^T W} X - X\right) W^T\right] \tag{9}$$

In a loop first iteratively update W until the objective function O converges on W, and then iteratively update C so that the objective function O converges on C again, and then the loop ends, and so on until the final objective function converges on both C and W.

4. Natural language processing models based on matrix decomposition

4.1. Natural Language Processing Optimization Model Construction

On the basis of the matrix decomposition word vector model in this paper, in order to further optimize the construction of the natural language processing model, this paper constructs a natural language processing optimization model based on deep learning. Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM), Convolutional Neural Network (CNN) are currently the most commonly used models for deep learning in natural language processing. To address the problem of interference semantics prevalent in the existing models, Semantic Discard Network (SDN) is designed to autonomously discard some of the interference semantics during training. Aiming at the problem of a single fusion method for local inference results, Semantic Fusion Alignment (SFA) is designed to fuse local inference results in a more reasonable and effective way. Finally, the SDF-NN model, a semantic entailment relation inference model containing the above two novel components, is proposed.

This paper is based on proposing a complete semantic implication relation inference model SDF-NN (Semantic Droppingand Fusion Neural Network) based on the decomposition attention mechanism, which can fully reflect the effectiveness of Semantic Dropping Networks (SDN) and Semantic Fusion Alignment methods (SFA). The overall framework of the model is shown in Figure 2. The model has four steps like the previous decomposition-based attention model: encoding, attention, comparison and aggregation. The model differs in that SDN is applied in the comparison step and SFA in the aggregation step. Assume two sentences $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_n)$, a is a word vector

representation of a premise of length m, and b is a word vector representation of a hypothesis of length n. We assume that a_i, $b_j \in R^d$, and d are the dimensions of the word vectors. The goal is to predict the label y to determine the relationship between a and b, y which can be neutral, contradictory, or entailed.

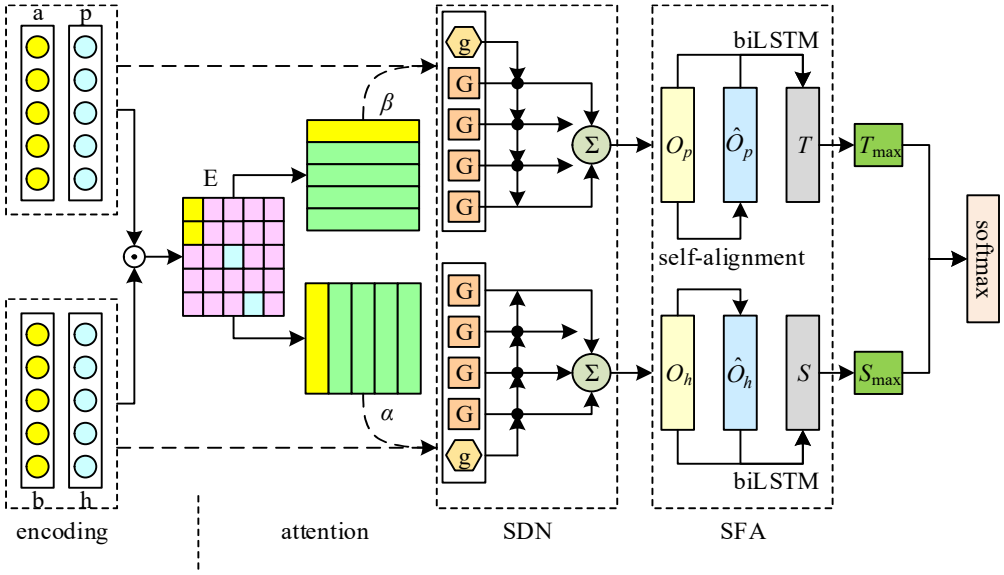

Fig. 2. SDF-NN structure

1) Sentence encoding. First, two sentences are input into each of the two directions in a temporal order $LSTM$. The output of each hidden layer is used as the encoded representation of the word vectors of the corresponding inputs:

$$\begin{aligned} p_i &= biLSTM\left(a_i\right) \\ h_j &= biLSTM\left(b_j\right) \end{aligned} \tag{10}$$

Matrices $P = \left[p_1, \ldots, p_m\right] \in R^{2r \times m}$ and $H = \left[h_1, \ldots, h_n\right] \in R^{2r \times n}$ are the outputs of the hidden layer of biLSTM. r is the number of neurons in the hidden layer.The formula for the LSTM model is given below:

$$\begin{aligned} i_t &= \sigma\left(W_i x_t + U_i h_{t-1}\right) \\ f_t &= \sigma\left(W_f x_t + U_f h_{t-1}\right) \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1}\right) \\ c_t &= f_t \square\ c_{t-1} + i_t \square\ \tanh\left(W_c x_t + U_c h_{t-1}\right) \\ h_t &= o_t \tanh\left(c_t\right) \end{aligned} \tag{11}$$

a_i and b_j, respectively, are sequentially inputted into the model based on the time series t as x_t. LSTM utilizes a memory block consisting of an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t to produce a hidden layer output h_t. Meanwhile, encoding using biLSTM is more efficient than LSTM encoding. That is, the sentence is forward encoded and then reverse encoded, and each word will have two corresponding hidden layer outputs $\vec{h}_t$ and $\overleftarrow{h}_t$, which are concatenated and spliced in series as the final encoded form of the word.

2) Attention mechanism. We use the attention framework in the paper by Parikh et al. to construct aligned text pairs. First, the encoded matrices of the two sentences are multiplied to obtain an un-normalized attention matrix:

$$e_{ij} = p_i^T h_j \tag{12}$$

Here $e_{ij} \in E^{m \times n}$ is the attention weight of each element in the attention representing the corresponding encoded word. The attention matrix is then normalized by rows and columns respectively:

$$\begin{aligned} \alpha_j &= \sum_{i=1}^{m} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{m} \exp\left(e_{kj}\right)} p_i, \forall j \in [1,\ldots,n] \\ \beta_i &= \sum_{j=1}^{n} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{n} \exp\left(e_{ik}\right)} h_j, \forall i \in [1,\ldots,m] \end{aligned} \tag{13}$$

Normalized by rows to α_j, and by columns to β_i. $\alpha_j \in R^{2r}$ is the subfragment of P that is aligned with h_j, a recoded representation of P based on the distribution of attention in h_j, for each word in P. $\beta_i \in R^{2r}$ is the subfragment in H, which is aligned with p_i and is a recoded representation of pair H based on the distribution of attention in p_i for each word in H. $\left(\alpha_j, h_j\right)$ and $\left(\beta_i, p_i\right)$ are referred to as aligned text pairs.

3) Semantic Discard Network (SDN). In the "Comparison" phase, the purpose of discarding interfering information is realized by the SDN network. The local inference results O_p and O_h are obtained by aligning the text pairs accordingly and feeding them into the Semantic Discarding Network (SDN) after obtaining each aligned text pair.

4) Semantic Fusion Alignment Approach (SFA). Finally, we align the relevant local inference results by applying mutual attention to local inference results $O_p \in R^{r \times m}$ and $O_h \in R^{r \times n}$ obtained by the SFA method, and in the aggregation phase, we align the relevant local inference results by applying mutual attention to these local inference results to obtain inference-aligned representations $\hat{O}_p \in R^{r \times m}$ and $\hat{O}_h \in R^{r \times n}$. These self-aligned representations contain not only the information about the local inference results themselves, but also the relevant linkages to the information about the other local inference based on the attentional weights. The final vector for prediction is obtained from these semantically aligned representations and the original local inference result $V \in R^{4r}$. V is fed into the final classification feed-forward neural network, which has only one layer of feed-forward, uses a softmax classification function, and outputs three categories. During training, the category with the largest output weight is used as the prediction. $W_s \in R^{4r \times 3}$ is a weight matrix:

$$y = soft\max\left(W_s^T V\right) \tag{14}$$

4.2. DLTE-based spatial embedding methods

In natural processing models, most methods have difficulty in learning accurate low-dimensional embedding representations when the input high-dimensional data exhibit complex nonlinear relationships. Second, most methods only consider the global structure of the potential embedding space without mining the higher-order correlations of different views in the potential embedding

space. Finally, most methods only consider capturing the structure of manifolds without considering the optimal manifolds. To this end, this paper proposes a novel multi-view subspace clustering (DLTE) method based on deep low-rank tensor embedding.

The framework of DLTE is shown in Fig. 3, and the method integrates deep NMF into the multi-view subspace clustering framework. It can effectively utilize the complex nonlinear relationships of high-dimensional data in the original space and learn better deep embedding representations. In addition, weighted low-rank tensor constraints are utilized to capture the global structure and higher-order correlations of different views in the deep embedding space. In addition, the optimal flow shape is mined by the optimal graph Laplacian operator.

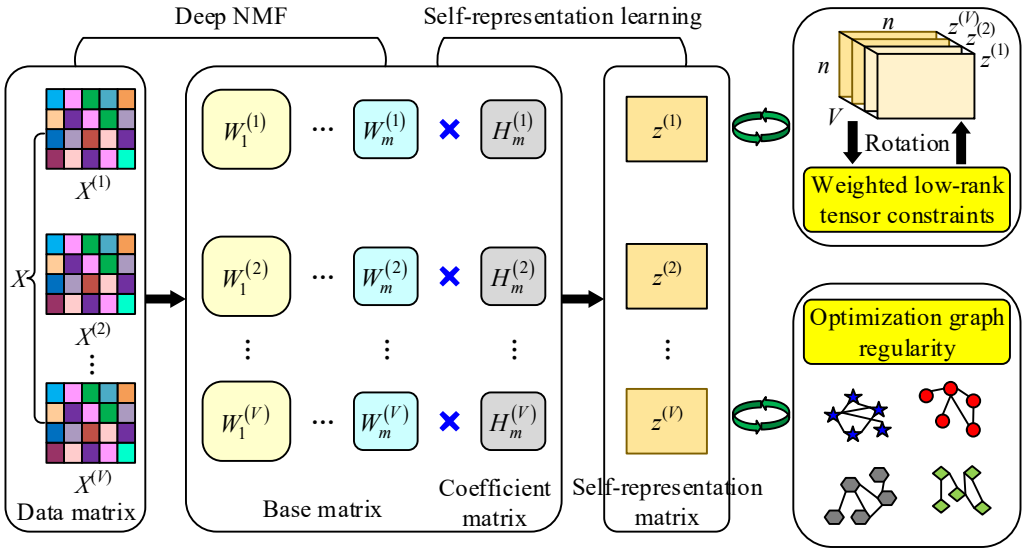

Fig. 3. DLTE framework

The goal of the DLTE method is to mine the deep embedding space based on multi-view subspace clustering of the potential embedding space, and in the deep embedding space, to further mine the global structure and local geometric structure.

Notation and related definitions:

To enhance the readability of this chapter, this subsection gives the notation and related definitions used in this chapter. Specifically, for tensor $\mathcal{Z}$, matrix Z and vector z are denoted using special letters, uppercase letters and lowercase letters, respectively. For tensor $\mathcal{Z}$, it is denoted by the F-paradigm $\|\mathcal{Z}\|_F = \sqrt{\sum_{i,j,k} |z(i,j,k)|^2}$, where $z(i,j,k)$ denotes the (i,j,k) th element of tensor $\mathcal{Z}$. The k th positive slice of tensor $\mathcal{Z}$ is denoted as $Z^{(k)}$. The fast Fourier transform (FFT) of tensor $\mathcal{Z}$ along the third dimension is denoted as $\overline{\mathcal{Z}} = fft(\mathcal{Z},[],3)$. Similarly, $\mathcal{Z}$ can be obtained by the inverse fast Fourier transform, i.e., $\mathcal{Z} = ifft(\overline{\mathcal{Z}},[],3)$. The block vectorization of $\mathcal{Z}$ and its inverse are denoted as $bvec(\mathcal{Z}) = \left[Z^{(1)};\cdots;Z^{(n_3)}\right]$ and $bvfold\left(bcirc(\mathcal{Z})\right) = \mathcal{Z}$, respectively. The block cyclic matrix $bcirc(\mathcal{Z})$ consisting of $Z^{(k)}$ is denoted as follows:

$$bcirc(\mathcal{Z}) = \begin{bmatrix} Z^{(1)} & Z^{(n_3)} & \cdots & Z^{(2)} \\ Z^{(2)} & Z^{(1)} & \cdots & Z^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ Z^{(n_3)} & Z^{(n_3-1)} & \cdots & Z^{(1)} \end{bmatrix} \tag{15}$$

Multi-view subspace clustering based on deep embedding:

Let $X=\left[X^{(1)},X^{(2)},\ldots,X^{(V)}\right]$ denote a set of multiview data with V different views, where the data matrix of the V rd view is denoted as $X^{(v)}\in\square^{d^{(v)}\times n}$. Based on the existing work, they assumed that the input data is extracted from a low-dimensional embedding space, where good experimental performance proves its effectiveness. They utilize matrix decomposition to obtain a cleaner potential embedding space, which effectively avoids the effects of noise and outliers in the high-dimensional data and thus improves the clustering results. However, they only considered the negative effects of noise and outliers while ignoring the complex nonlinear relationships of the data. Inspired by ODD-NMF, deep NMF can capture the complex nonlinear relationships of data well. Therefore, deep NMF is integrated into the multi-view subspace clustering framework with the following objective function formula:

$$\begin{aligned}&\min_{W_i^{(v)},H_i^{(v)},Z^{(v)}}\sum_{v=1}^{v}\left\|X^{(v)}-W_1^{(v)}\ldots W_m^{(v)}H_m^{(v)}\right\|_F^2+\alpha\left\|H_m^{(v)}-H_m^{(v)}Z^{(v)}\right\|_F^2\\&s.t.W_i^{(v)}\geq 0,H_i^{(v)}\geq 0\end{aligned}\tag{16}$$

where $W_i^{(v)}\in\square^{l_{i-1}\times l_i}$ denotes is the base matrix of the i rd layer of the v nd view, here $i=1,\ldots,m$ and if $i=I$ then $W_1^{(v)}\in\square^{d^{(V)}\times l_1}$, l and l_i denote the layer size vector and the dimension of the i th layer, respectively, $H_i^{(v)}\in\square^{l_i\times n}$ denotes the depth embedded representation of the i th layer of the v th view, $H_m^{(v)}$ denotes the coefficient matrix of the last layer of the v th view, $Z^{(v)}\in\square^{n\times n}$ denotes the self-represented coefficient matrix of the v th view, $\|\cdot\|_F$ denotes the $F-$ paradigm, and α is a balance parameter.

Weighted low-rank tensor constraints:

Most of the previous multi-view subspace clustering algorithms utilize the kernel-paradigm constraints to mine the global structural information in the potential embedding space. Moreover, they only study the pairwise correlations of different views in the potential embedding space, while the higher-rank correlations of different views are often ignored, which leads to suboptimal results.

To address the above issues, this subsection introduces weighted low-rank tensor constraints to explore the global structural information while capturing the higher-order correlations of different views. Therefore, based on the existing work and equations, the weighted low-rank tensor constraint is formulated as follows:

$$\left\|\mathrm{Z}\right\|_{\omega,\otimes}=\sum_{k=1}^{n_1}\sum_{i=1}^{\min(n_1,n_2)}\omega_i\sigma_i\left(\overline{Z}^{(k)}\right)\tag{17}$$

where $\mathrm{Z}\in\square^{n\times V\times n}$ denotes a tensor, where its dimension is obtained by rotating the transformation from $n\times n\times V$, $\overline{Z}^{(k)}$ denotes the i th frontal slice of $\overline{\mathrm{Z}}$, and ω_i denotes the weights of $\sigma_i\left(\overline{Z}^{(k)}\right)$. By ω the a priori knowledge of the matrix singular values can be fully taken into account. Due to $V\square n$, the number of elements in the weighting vector ω is related to the number of views V.

Most of the existing multi-view subspace clustering methods based on latent embedding fail to fully exploit the local structural information, which is crucial for the performance of data clustering. As described in existing work, when two data points are close to each other in the original data space, they also remain close in the mapped low-dimensional potential space. Here, this subsection treats $\left\{Z_1,Z_2,\ldots,Z_n\right\}$ as n graph nodes, constructs the graph and mines the local geometric structure using the graph Laplacian operator, which can be formulated as:

$$\sum_{i,j=1}^{n}\left\|Z_i-Z_j\right\|^2M_{ij}=Tr\left(Z^TDZ\right)-Tr\left(Z^TMZ\right)=Tr\left(Z^TLZ\right)\tag{18}$$

where $Tr(\cdot)$ denotes the trace operator function of the matrix, $L = D - M$ denotes the Laplace matrix, and here M and D denote the similarity and diagonal matrices, $D_{ii} = \sum_{j=1}^{n} M_{ij}$ respectively.

With the existing work, it can be observed that the data of different views are extracted from different manifolds. Therefore, the above graph Laplacian operator will lead to sub-optimal clustering performance. Therefore optimal manifolds are to be used, where the common manifold is considered as the optimal manifold. L_{op} can be obtained in the form of graph regularization, which is defined as:

$$L_{op} = D_{op} - M_{op} \tag{19}$$

where $D_{op} = \left\{ d_{ii}^{op} \right\}^{n \times n}$ denotes a diagonal matrix whose each element is computed as $d_{ii}^{op} = \sum_{j=1}^{n} m_{ij}^{op}$, and M_{op} denotes the optimal similarity matrix, which can be defined by taking the minimum similarity value of all views as follows:

$$M_{op} = \left\{ m_{ij}^{op} \right\}^{n \times n}, m_{ij}^{op} = \min \left\{ m_{ij}^{(v)} \right\}_{v=1 \ldots V} \tag{20}$$

where $M^{(v)} \in \square^{n \times n} = \left\{ m_{ij}^{(v)} \right\}^{n \times n}$ is the adjacency matrix of view v, which is generated by using the k-nearest neighbor (KNN) algorithm. Then, the adjacency matrix $M^{(v)}$ is defined as follows:

$$m_{ij}^{(v)} = \begin{cases} sim_{ij} & if X_i^{(v)} \in \mathrm{N}^{k}\left(X_j^{(v)}\right) or X_j^{(v)} \in \mathrm{N}^{k}\left(X_i^{(v)}\right) \\ 0 & otherwise \end{cases} \tag{21}$$

where $X_i^{(v)}$ and $X_j^{(v)}$ denote the i th and j th samples of the v rd data matrix $X^{(v)} \in \square^{d^{(N)} \times n}$ respectively, where $d^{(v)}$ denotes the dimension of the v th view data, $\mathrm{N}^{k}\left(X_i^{(v)}\right)$ denotes the set of k nearest neighbors of $X_i^{(v)}$, and sim_{ij} denotes the similarity between $X_i^{(v)}$ and $X_j^{(v)}$. Similarity can be measured by different algorithms such as heat kernel, binary, cosine and dot product.

L_{op} is the Laplace matrix with the most similar manifold common to all views. Then, to ensure that the self-representation coefficient matrix of each view in the deep embedding space respects the new optimal manifold, the optimal graph regularization formula can be defined as follows:

$$\min_{Z^{(r)}} \mathrm{Tr}\left(Z^{(v)T} L_{op} Z^{(v)} \right) \tag{22}$$

5. Analysis of the effect of natural language processing models based on matrix decomposition

5.1. Matrix Decomposition Word Vector Model Experimental Results

In this paper, the experiments are conducted on three English corpora of different sizes such as enwiki9, WebBase, and Common Crawl (CC). After preprocessing, the vocabularies of the three corpora are 56466, 277704, and 400000, respectively.In this paper, a context window of size 10 is used to compute to get the context co-occurrence matrix of words. In order to evaluate the quality of the word vector model, this paper conducts experiments on several different evaluation tasks, i.e., word similarity task, downstream natural language processing tasks (text categorization task and named entity recognition task). In the experiments, the KbEMF method proposed in this paper is mainly compared with word vectors based on matrix factorization of global co-occurrence information (MF), Word2vec, log-bilinear regression model based on global co-occurrence information (GloVe), Fasttext, and Koan, and the number of dimensions of all word vectors in the experiments is set to 300.

5.1.1. Word Vector Quality and Effectiveness Analysis. Taking the experimental results of each model in the three corpora, the results of the word quality review comparison are shown in Table 1. From the table it can be seen that the average performance index of this paper's model on each task in the three corpora ranges from 71.55 to 89.11. The entity naming task Conll has the best performance, and the performance of the performance on Conll is improved by 0.30 to 1.48 compared to other methods, which indicates that the performance of the word vectors obtained through the word vector method of this paper is stable and the overall performance of the model is better.

Table 1. Comparison results of word quality evaluation

Method model	Word similarity		Classification task		Named entity task	
	MEN	WS353	SST	5AG	Conll	BTC
GloVe	70.52	67.45	80.31	87.61	87.92	72.13
Word2vec	75.23	70.09	80.24	86.72	87.63	71.34
Fasttext	74.16	66.93	81.73	87.95	88.75	71.78
Koan	74.66	67.72	82.45	88.93	88.71	72.50
MF	75.35	71.47	79.73	87.72	88.81	72.17
Ours	75.48	71.55	80.11	88.48	89.11	72.58

In order to evaluate the computational efficiency of each method model in learning word vectors, this paper counts the time overhead and computational resources of all the methods in three different sizes of corpora, and the comparison of the time overhead of word vector processing is shown in Table 2. In terms of computation time this paper's method and MF have a big advantage over other methods such as Word2vec, GloVe, Fasttext and Koan. The time overhead of this paper's method in the three corpora is 3.93, 7.29 and 13.42 minutes, respectively, which still has a certain advantage compared to the good hair MF, and the speed of the model has been significantly improved. The above experiments fully confirm the better performance of this paper's model, while increasing to improve the learning efficiency of word vectors.

Table 2. The time cost comparison of word vector processing

Method	Training corpus		
	enwiki9 (0.1B)	WebBase (3B)	CC (6B)
GloVe	45.72mins×10Cc	182.84mins×10Cc	270.43mins×10Cc
Word2vec	59.53mins×20Cc	1211.43mins×20Cc	2369.80mins×20Cc
Fasttext	67.46mins×20Cc	1843.67mins×20Cc	3419.36mins×20Cc
Koan	32.42mins×10Cc	874.19mins×10Cc	1490.29mins×10Cc
MF	13.31mins×1Cc	67.65mins×1Cc	153.88mins×1Cc
Ours	3.93mins×1Cc	7.29mins×1Cc	13.42mins×1Cc

5.1.2. Analysis of Matrix Sampling Methods. In order to explore the effect of sampling size k on the performance of the word vectors obtained from the model and the computation time, this paper conducts exploratory experiments on sampling size k on three corpora. To evaluate the performance of the extracted word vectors, this paper uses a representative benchmark score (similarity task MEN) to measure the performance gain. The trend of the model evaluation scores versus computation time is shown in Fig. 4, from which it can be found that to achieve at least 98% of the performance of the MF method, the sampling size of this paper's method requires only 40,000, 50,000, and 50,000 on enwik9, WebBase, and on CC, respectively, which is about 71%, 18%, and about 13% of the size of the original information matrix. As the size of the original information matrix increases, the proportion of the model of this paper's method that needs to be sampled gradually decreases. This indicates that the

method in this paper has a greater computational efficiency advantage when word vectors at larger word list sizes.

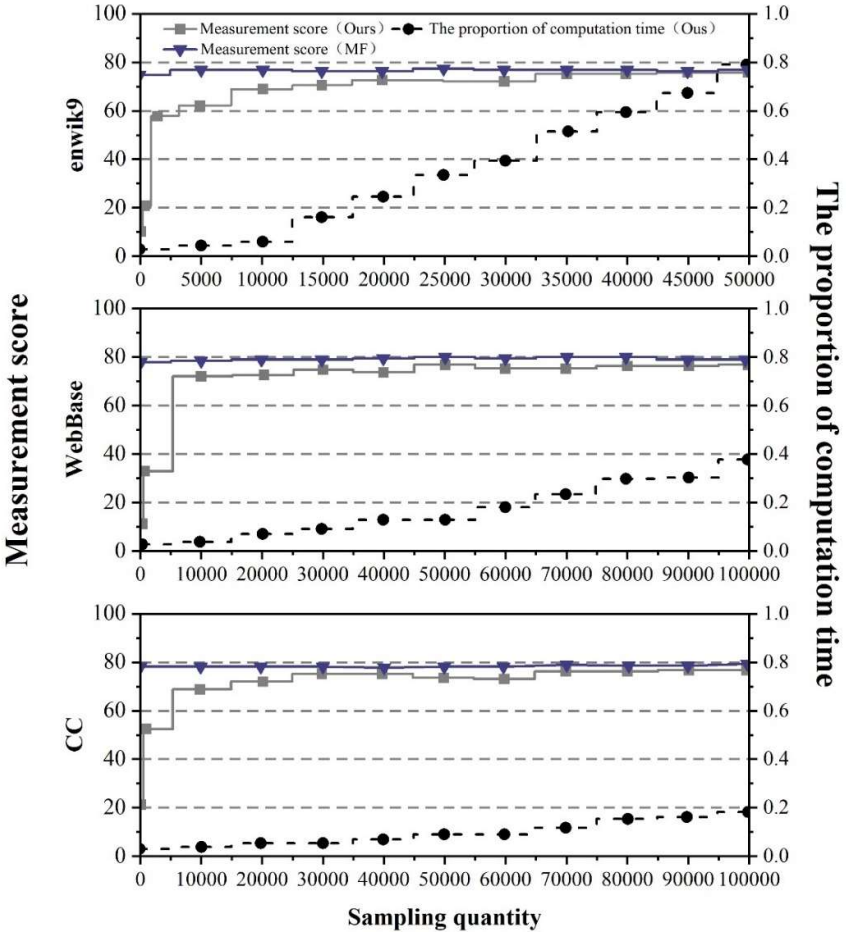

Fig. 4. Model measurement score and calculation time change trend

5.2. Performance Analysis of Natural Language Processing Models

In order to further verify the performance of the natural language processing model based on matrix decomposition in this paper, the Stanford parser is chosen for the experiments in this chapter. The diverse language model officially provided by Stanford can support syntactic parsing in English, Chinese and other languages. The English corpus uses the International Semantic Evaluation Contest Text Semantic Similarity Task dataset. Due to the variability of the sample sources of the datasets included in the STS task in different years, the average Pearson's coefficient of each dataset is used here as the final index.The STS task, as a widely used English text semantic similarity dataset in the international world, with sentence pairs of samples covering a number of different domains, is able to fully validate the ability of the generalization of the FPBR model on samples from different domains. Considering the intuitiveness of the data, all Pearson correlation coefficients in the experimental results of this paper are enlarged 100 times by default. Two types of comparison baselines are chosen for the experiments in this chapter, word vector-based weighting model and neural network model. The two main types of word vector weighting models are TF-IDF and SIF. The selected comparison neural network models include CNN, RNN, LSTM, ST, QT and InferSent. The experimental results of the English text of the model are shown in Fig. 5, and the experimental results show that InferSent has good performance on each dataset, which are all above 55, proving that InferSent is an excellent baseline model. And the method in this paper has the best performance on all datasets. The performance of STS-13 and STS-14 tasks is improved by 3.3 and 2.4 over other optimal performances,

respectively, and the performance of this paper's model is much better, showing quite good competitiveness.

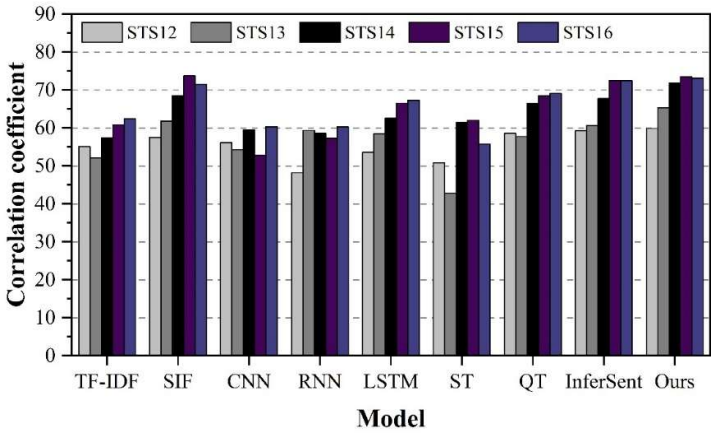

Fig. 5. Model English text experiment results

The natural language processing model in this paper utilizes matrix decomposition to construct a word vector model as a basis, emphasizing that each word is intrinsically linked as a syntactic component in the whole sentence, and when the sentence length is larger, the corresponding syntactic structure becomes more complex, and the overall effect of the model improves more obviously. In order to verify this property, this paper draws 200 samples from the Chinese and English datasets according to different lengths for experiments, under the condition that other parameters are consistent. The effect of sample length on model performance is shown in Figure 6, from which the results show that sentence length is basically positively correlated with various performance indicators.

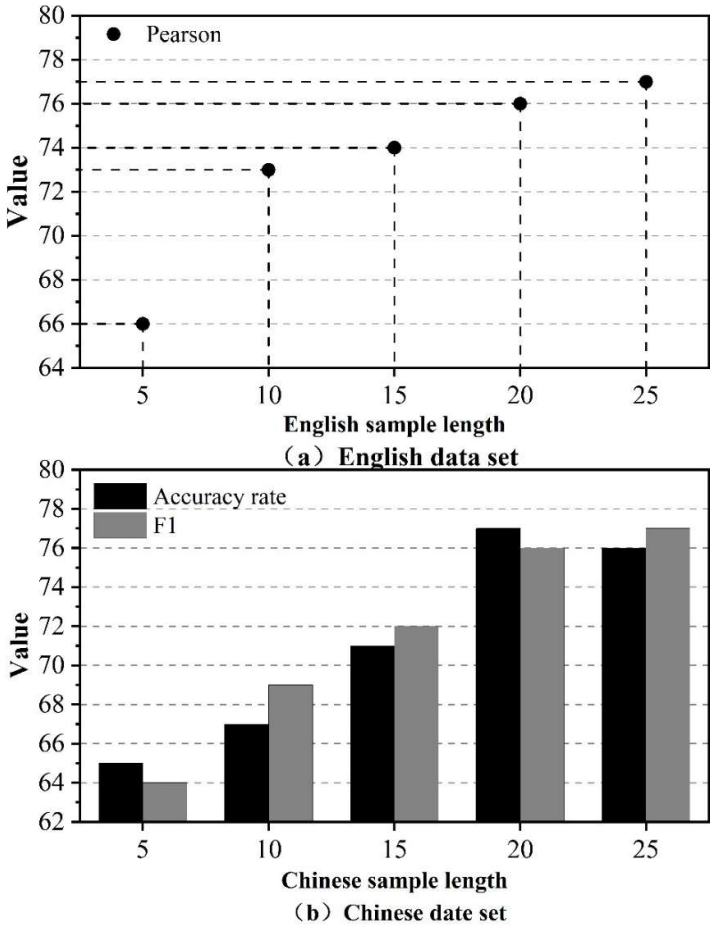

Fig. 6. The effect of sample length on model performance

To further validate the performance of the model in this paper, the model in this paper is compared with some neural network models, and the baseline models include CNN, RNN, LSTM, ST, QT, and InferSent. Considering that the samples of the various datasets of the STS-16 task are on the low side, this subsection uses the 11 datasets provided by STS-14 and STS-15, which involve many domains such as news, forums, images, etc., which can fully test the generalization of the FPBR model in different domains. The results of model comparison are shown in Fig. 7. From the results of the comparison experiments, we can see that QT and InferSent, as the best current sentence embedding techniques, both achieve the best performance on 5 datasets in total, and their performance on other datasets is also quite stable. InferSent learns generalized sentence embedding expressions using the SNLI dataset, which proves that the supervised model of a single task can be free from inductive favoritism and succeed in the field of generic text expressions. Unlike InferSent, QT is an unsupervised model, according to the given target sentence and several contexts, the model needs to select reasonable contexts among the given multiple utterances, and the whole selection process can also be regarded as the approximate generation process of contexts, which improves the quality of the sentence representations learned by the model. And the model in this paper outperforms the comparison baseline on more than half of the datasets, and the effect improves significantly on the 14'OnWN, 14'Tweet-news, and 15'Ans.-forum datasets by 2.85, 3.53, and 2.41, respectively. The performance of the model in this paper is further validated.

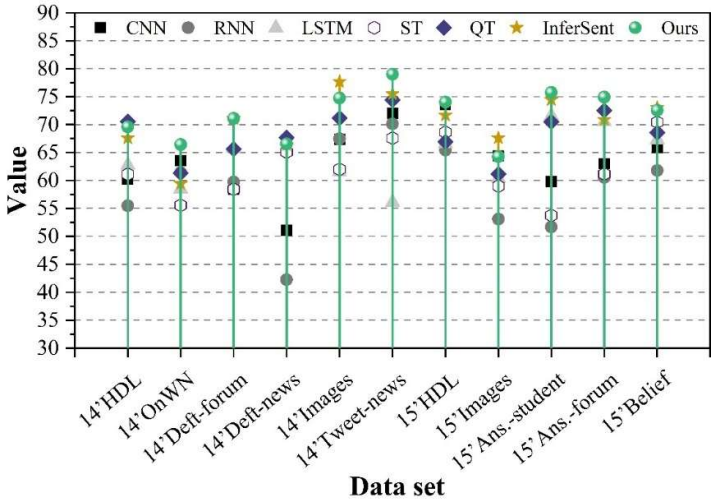

Fig. 7. Model comparison results

6. Conclusion

The development of natural language processing technology provides a more comprehensive understanding of the processing method, this study for the natural language processing model in the low-dimensional embedding space of the efficient construction of the problem, starting from the word vector, constructed based on the matrix decomposition of the word vector learning model. The goodness of word vectors determines the performance effect of natural language processing models, and the word vector learning model constructed in this paper can improve the quality of word vectors and the model time overhead in a more comprehensive way. In the experiments, the average performance index of the word vector model in this paper ranges from 71.55 to 89.11, and the overall performance is good. Among them, the entity naming task Conll has the best performance, which is improved by 0.30~1.48 compared with the performance of other methods, and the speed of the model has been significantly improved, which increases the learning efficiency of word vectors. In addition, the sampling amount of the word vector model in this paper only needs 40,000~50,000, and the proportion of the model that needs to be sampled is gradually reduced, which has a greater advantage when it comes to word vectors in a larger word list size.

Based on this, this paper constructs the best performance of the natural language processing model (SDF-NN) on each dataset. It improves the performance over the other best by 3.3 and 2.4 in STS-13 and STS-14 tasks. And the sentence length of the model is basically positively correlated with each performance index, indicating that the larger the sentence length is, the more obvious the overall effect of the model is improved. In addition, through the comparison with other models, it is found that the performance of this paper's model is better on most of the datasets, which further validates the performance of this paper's model and demonstrates a high degree of competitiveness.

Funding

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

References

[1] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. Multimedia tools and applications, 82(3), 3713-3744.

[2] Pilehvar, M. T., & Camacho-Collados, J. (2020). Embeddings in natural language processing: Theory and advances in vector representations of meaning. Morgan & Claypool Publishers.

[3] Worth, P. J. (2023). Word embeddings and semantic spaces in natural language processing. International journal of intelligence science, 13(1), 1-21.

[4] Jung, S. (2019). Semantic vector learning for natural language understanding. Computer Speech & Language, 56, 130-145.

[5] Li, Y., & Yang, T. (2018). Word embedding for understanding natural language: a survey. Guide to big data applications, 83-104.

[6] Wang, S., Zhou, W., & Jiang, C. (2020). A survey of word embeddings based on deep learning. Computing, 102(3), 717-740.

[7] Asudani, D. S., Nagwani, N. K., & Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. Artificial intelligence review, 56(9), 10345-10425.

[8] Li, S., Hu, J., Cui, Y., & Hu, J. (2018). DeepPatent: patent classification with convolutional neural networks and word embedding. Scientometrics, 117(2), 721-744.

[9] Onan, A. (2021). Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. Concurrency and computation: Practice and experience, 33(23), e5909.

[10] Alami, N., Meknassi, M., & En-Nahnahi, N. (2019). Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. Expert systems with applications, 123, 195-211.

[11] Li, S., & Gong, B. (2021). Word embedding and text classification based on deep learning methods. In MATEC Web of Conferences (Vol. 336, p. 06022). EDP Sciences.

[12] Bokka, K. R., Hora, S., Jain, T., & Wambugu, M. (2019). Deep Learning for Natural Language Processing: Solve your natural language processing problems with smart deep neural networks. Packt Publishing Ltd.

[13] Zhou, Y. (2022). Natural language processing with improved deep learning neural networks. Scientific Programming, 2022(1), 6028693.

[14] Fesseha, A., Xiong, S., Emiru, E. D., Diallo, M., & Dahou, A. (2021). Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya. Information, 12(2), 52.

[15] Jiang, K., Feng, S., Song, Q., Calix, R. A., Gupta, M., & Bernard, G. R. (2018). Identifying tweets of personal health experience through word embedding and LSTM neural network. BMC bioinformatics, 19, 67-74.

[16] Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. C. J. (2019). Evaluating word embedding models: Methods and experimental results. APSIPA transactions on signal and information processing, 8, e19.

[17] Khanal, J., Tayara, H., & Chong, K. T. (2020). Identifying enhancers and their strength by the integration of word embedding and convolution neural network. Ieee Access, 8, 58369-58376.

[18] Xu, K., Zhao, X., Bastani, H., & Bastani, O. (2021, July). Group-sparse matrix factorization for transfer learning of word embeddings. In International Conference on Machine Learning (pp. 11603-11612). PMLR.

[19] Li, K., Zhou, X., Lin, F., Zeng, W., & Alterovitz, G. (2019). Deep probabilistic matrix factorization framework for online collaborative filtering. IEEE Access, 7, 56117-56128.

[20] Liu, H., Zheng, C., Li, D., Shen, X., Lin, K., Wang, J., ... & Xiong, N. N. (2021). EDMF: Efficient deep matrix factorization with review feature learning for industrial recommender system. IEEE Transactions on Industrial Informatics, 18(7), 4361-4371.

[21] Wadud, M. A. H., Mridha, M. F., & Rahman, M. M. (2022). Word embedding methods for word representation in deep learning for natural language processing. Iraqi Journal of Science, 1349-1361.

[22] AlSurayyi, W. I., Alghamdi, N. S., & Abraham, A. (2019). Deep learning with word embedding modeling for a sentiment analysis of online reviews. International Journal of Computer Information Systems and Industrial Management Applications, 11, 15-15.

[23] Yao, H., Liu, H., & Zhang, P. (2018). A novel sentence similarity model with word embedding based on convolutional neural network. Concurrency and Computation: Practice and Experience, 30(23), e4415.

[24] Xiong, Z., Shen, Q., Xiong, Y., Wang, Y., & Li, W. (2019). New Generation Model of Word Vector Representation Based on CBOW or Skip-Gram. Computers, Materials & Continua, 60(1).

[25] Liu, B. (2020). Text sentiment analysis based on CBOW model and deep learning in big data environment. Journal of ambient intelligence and humanized computing, 11(2), 451-458.

[26] Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018, February). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In Proceedings of the eleventh ACM international conference on web search and data mining (pp. 459-467).

[27] Acharya, A., Goel, R., Metallinou, A., & Dhillon, I. (2019, July). Online embedding compression for text classification using low rank matrix factorization. In Proceedings of the aaai conference on artificial intelligence (Vol. 33, No. 01, pp. 6196-6203).

[28] Zhu, Z., Yan, M., Deng, X., & Gao, M. (2022). Rating prediction of recommended item based on review deep learning and rating probability matrix factorization. Electronic Commerce Research and Applications, 54, 101160.

[29] Xie, R., Yuan, X., Liu, Z., & Sun, M. (2017, August). Lexical Sememe Prediction via Word Embeddings and Matrix Factorization. In IJCAI (pp. 4200-4206).

[30] Liu, N., & Zhao, J. (2023). Recommendation system based on deep sentiment analysis and matrix factorization. IEEE Access, 11, 16994-17001.

[31] Li, W., Zhang, J., Zhou, J., & Cui, L. (2018, July). Learning Word Vectors with Linear Constraints: A Matrix Factorization Approach. In IJCAI (pp. 4187-4193).