International Journal for Housing Science and Its Applications
Publish August 6, 2025. Volume 46, Issue 3 Pages 5511-5523

https://doi.org/10.70517/ijhsa463476

Research on Word Vector Calculation Method of English Corpus Based on Transformer Modeling

Chunmei Qiao[1,*]
[1] The Public Course Teaching Department, Henan Vocational University of Science and Technology, Zhoukou, Henan, 466000, China
Corresponding authors: (e-mail: qcmcassie821826@163.com).

Abstract With the development of deep learning, distributed word vector technology based on neural networks shows strong potential. However, the English corpus is diverse in type and large in volume, and traditional word vector computation is difficult to adapt to the semantic evolution and semantic association of words in long textual contexts in dynamic environments, and the effect varies significantly on different sizes of corpora. In this paper, a word vector computation method Transformer-DSSM based on the Transformer model for English corpus is proposed to solve the problem that traditional word vector computation cannot effectively deal with polysemous words and contextual relevance. The study is based on a deep semantic model, utilizing the Transformer coding layer for feature extraction, and calculating lexical semantic relations through the self-attention mechanism and cosine similarity. Experiments on the SICK and MSRP datasets show that the Transformer-DSSM model obtains excellent performance with Pearson's coefficient of 0.887, Spearman's coefficient of 0.845, and mean squared error of 0.2286 on the SICK test set, and reaches an accuracy of 77.8% and an F1 value of 83.8% on the MSRP test set. In addition, simulation experiments on the English news recommendation datasets MIND and Adressa verify the effectiveness and practicality of the model in word vector representation, providing a new solution for English corpus processing.

Index Terms Word vector computation, Transformer model, Self-attention mechanism, Deep semantic modeling, Polysemantic word processing, Semantic similarity

I. Introduction

Words are the basic building blocks of language, and determining word meanings is the basis for semantic understanding of sentences and texts. When using deep learning for tasks such as computing semantic features, sentiment features, topic features, and named entity recognition, the basic work is the vectorized representation of words [1]-[3]. Currently, the word vectors used are generally single prototype word vectors, i.e., a word is represented using a fixed vector, and the one-to-one modeling of single prototype word vectors words and semantics has achieved good results in most of the tasks, but there is a wide range of many-to-many relationships between words and semantics in actual linguistic expressions [4], [5]. Polysemous words express different semantics in different scenes and contexts. Studies have shown that the proportion of polysemous words occurring in natural language is about 42%, and people themselves can understand the semantics of polysemous words through their own knowledge base and reasoning ability, but such ambiguities are difficult to be distinguished for machine learning [6], [7].

In the field of NLP, the recognition and processing of polysemous words has always been a difficult problem, and the tasks related to this are polysemous word recognition, polysemous word representation, word sense induction, semantic disambiguation, etc. The NLP discipline has been adopting linguistic-based and knowledge-based approaches for decades, and these approaches are seldom applied in reality [8]-[11]. As one of the basic units for expressing semantics, words are the main objects of NLP [12]. The basic concept of word vector is the numerical or vectorized representation of human symbolized words [13]. In recent years, with the continuous development of deep learning, neural network-based distributed word vector technology, based on the algorithmic training of massive corpus, embeds symbolic sentences into a low-dimensional dense vector space, which shows strong potential and application effects in parsing syntax and analyzing semantics [14], [15]. As a global language, the English corpus is diverse in type and large in size, which makes it more demanding for word vector computation. The static word vector representation in traditional word vector computation is difficult to adapt to the semantic evolution of language and semantic association of words in long text contexts in dynamic environments, and the word vector computation of corpora of different sizes is ineffective [16]. Therefore, it is necessary to design new word vector computation methods for English corpus.

With the development of deep learning technology, word vector representations have gradually evolved from the early solo thermal encoding to models such as Word2Vec, ELMo and BERT, showing an evolutionary trend from static to dynamic word vectors. However, the existing studies still face problems such as insufficient processing of polysemous words and weak contextual relevance. Especially for the English corpus, due to its diverse types and large volume, the requirements for word vector computation are higher. Studies have shown that polysemous words account for about 42% of natural language, and traditional single prototype word vectors are difficult to accurately express the semantic differences of these words in different contexts. In this context, this study combines the advantages of the Transformer model and constructs a novel word vector computation model, Transformer-DSSM, which adopts the deep semantic model as the underlying architecture, replaces the traditional DNN network with the Transformer coding layer, captures long-distance dependencies between words through the mechanism of self-attention and combines with the cosine similarity calculation method to evaluate the semantic correlation between word vectors. In order to solve the problem of long text processing and category imbalance, fixed-length complementation and category balancing processing methods are adopted. In this study, the performance of the model is verified on the SICK and MSRP standard datasets, and the practicality is verified on the English news recommendation datasets MIND and Adressa. The optimal parameter configurations of the model in practical applications are explored through region number selection and region length experiments. The study not only proposes a new method for word vector computation, but also verifies the effectiveness of the method through a large amount of experimental data, which provides new ideas and methods for improving the quality of word vector representation in natural language processing and improving the performance of downstream tasks.

II. Word Vector Method and Transformer Principle

This chapter will focus on the methods involved in the computation of word vectors for the English corpus: the word vector method and the Transformer model.

II. A.Word Vector Methods

In order to realize computer recognition and cognition of natural language and to solve natural language processing related problems, it is required to transform the text into vector form before inputting into the model. The accuracy of the word vector representation will directly affect the effectiveness of downstream related natural language processing tasks, and is therefore central within the field of natural language processing. Commonly used word vector techniques are unique hot coding, static word vectors and dynamic word vectors. The order of development of word vector techniques is as follows: Unique Heat Encoding Method, Word2Vec Method, ELMo Method, BERT Model.

Initially, the vocabulary is characterized with the help of one-hot (i.e., One-hot) coding, which generates vectors whose dimension size is always the number of words in the word list. For the N th item of vocabulary, the N th dimension of the vector needs to be set to 1, and the remaining positions are set to 0. Since the dimension of the one-hot encoding is determined by the number of words in the word list, an overly large word list will result in too high a vector dimension, which is not conducive to the downstream neural network computation.

Word2Vec obtains static word vector representations with the help of CBOW and Skip-gram, the basic idea of which is to use massive corpus resources to train vocabulary feature vectors with stronger generalization and less dimensionality. Among them, CBOW uses a three-layer neural network to input the word vectors of the words in the contextual context of a specific center word, and then averages the word vectors of the words in the contextual context into the hidden layer, and finally obtains the word vector computation results of the center word.

The ELMo method uses a multilayer bidirectional recurrent neural network to model the text so that it can capture contextual information and historical linguistic features to generate historical high quality word vectors.The ELMo vector is a contextually relevant vector, which takes into account the contextual information of the word in the whole text sequence, and therefore the vector representation of the same word trained in different contexts will be different, which improves the accuracy of natural language tasks that need to consider contextual information. However, since ELMo uses a recurrent neural network structure, it still suffers from the long range dependency problem faced when dealing with long sequence data. With the emergence of pre-trained models and their excellent performance, more and more researchers are turning their attention to pre-trained models.

BERT is a language representation model based on pre-training, using Masked ML method and Next Sentence Prediction method to learn the semantic representation relationship between words and sentences, to complete the vectorized representation of text sequences The input form of the BERT model is the result of summing up the word vectors, segment vectors and position vectors corresponding to each character. The BERT model utilizes the encoder of the multilayer Transformer model to encode the input text sequences, and the final model outputs a bi-directional linguistic representation that incorporates contextual information.

II. B.Transformer model

The Transformer model encodes text sequences entirely by means of the attention mechanism, and the parallel performance of the model is greatly improved by the ability to process the entire input sequence simultaneously and to realize the computation process by matrix multiplication and parallel computation [17]. All characters of the sequence can be computed in parallel by the Transformer model, which is able to fully capture textual dependencies within the longer text, and also learn a bidirectional feature representation with contextual information.

Transformer consists of two components, encoder and decoder, each of which is composed of several network modules of the same structure. The model structure is shown in Figure 1.

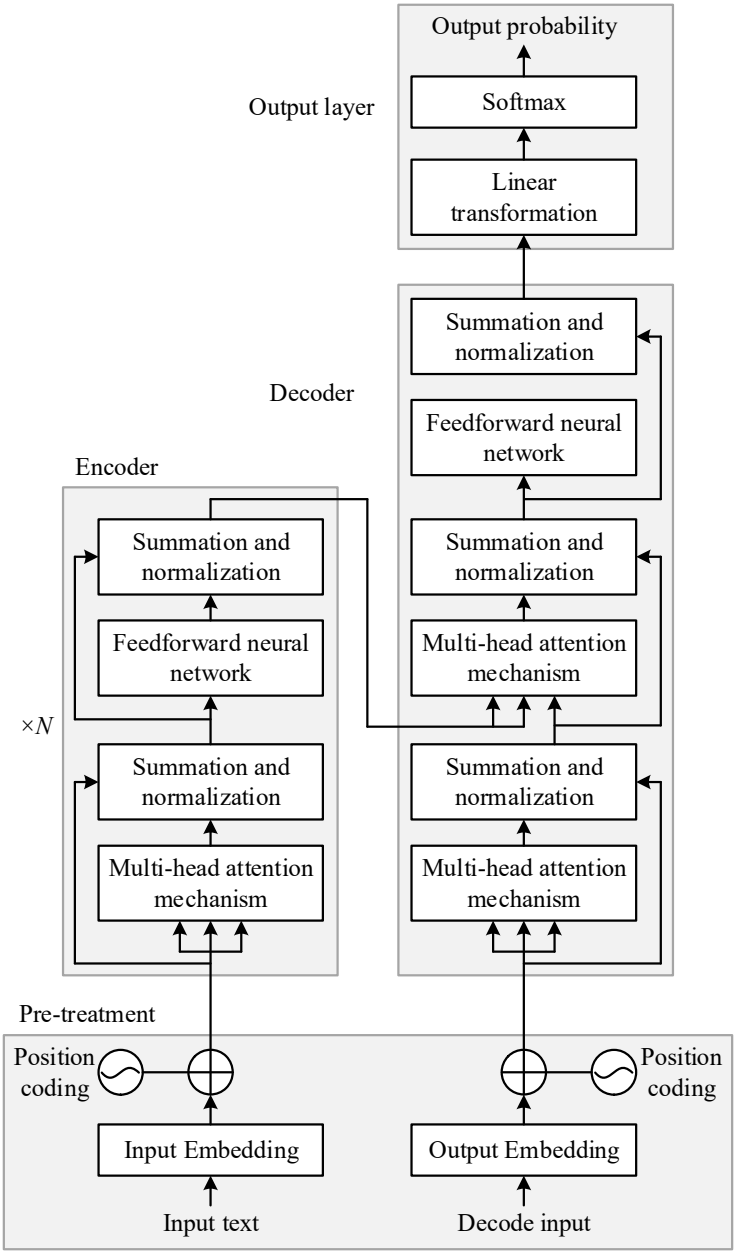


Figure 1: Flowchart of the Transformer model flow

1) Position embedding

The Transformer model introduces a positional embedding mechanism that embeds the position of each word in the input sequence into the word vector so that the model can learn the positional and lexical information. Positional embedding is a technique that maps discrete positional information into a continuous vector space, which is obtained with the help of a linear transformation. The dimension of the position embedding is usually the same as

the dimension of the word vectors of the input sequence, which facilitates the learning of the relationship between the word vectors and the positional information, and the linear transformation process is shown in Eqs. (1) and (2):

$$PE(pos,2i)=\sin(pos/1000^{2i/d_{emb}}) \tag{1}$$

$$PE(pos,2i+1)=\cos(pos/1000^{2i/d_{emb}}) \tag{2}$$

In the above formula, pos is the position in the input sequence used to generate the position vector for each position, i is the dimension number of the position vector, and d_{smb} is the dimension of the word vector in the input sequence.

2) Scaling dot product attention mechanism

The scaled dot product attention mechanism is one of the core components in Transformer, which utilizes the three input vectors of query, key and value to compute the attention score and the final output vector.

The specific calculation formula is shown in (3):

$$Attention(Q,K,V)=softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

3) Multi-head Attention Mechanism

Multi-head attention mechanism is an improvement of the traditional attention mechanism, which divides the query, key and value vectors of the input sequence into multiple heads, each of which can independently compute the attention scores and output vectors, thus enabling the model to focus on different parts of the input sequence at the same time and obtain different attention distributions.

When the model receives the text sequence data, the multi-head attention mechanism will generate the corresponding Q,K,V vectors from the input vectors of each position by linear transformation operation, and these three vectors are shared by all positions. Next, Q,K,V are fed to multiple heads separately, and each head will perform the computation of the self-attention mechanism. After the computation is completed, the outputs from multiple heads are stitched together and processed through residual concatenation and normalization layers to obtain the final output vector.

The formulation of the multi-head attention mechanism is as follows:

$$MultiHead(Q,K,V)=Concat(head_1,\cdots,head_h)W^0 \tag{4}$$

$$head_i=Attention(QW_i^Q,KW_i^K,VW_i^V) \tag{5}$$

In Eqs. (4) and (5), Q,K,V have the same meanings as in Eq. (3), and W is the transformation parameter. The attention weight vector of each head is multiplied by the corresponding transformation parameters W_i^Q,W_i^K and W_i^V, where i denotes the head number. The attention weights obtained for each head are then weighted and summed with the value vector to obtain the output $head_i$ for each head. The output of the multi-head attention mechanism is obtained by splicing $head_1,head_2,\cdots,head_h$. Finally, the spliced results are linearly transformed to obtain the final output vector of the multi-head attention mechanism, where the variation parameter is W^O.

4) Feedforward Neural Network

The use of feedforward neural network layer mainly lies in the ability to enhance the nonlinear representation of the model, and the feedforward neural network calculation process is shown in Equation (6).

$$FFN(x)=\max(0,xW_1+b_1)W_2+b_2 \tag{6}$$

where W_1,W_2,b_1,b_2 are the parameters of the feed-forward neural network layer, and $W_1\in R^{d\times d_f},W_2\in R^{d_f\times d}$, $b_1\in R^{d_f},b_2\in R^d$.

5) Residual connection and normalization layer

Residual connection and normalization layer are important mechanisms in the Transformer model, and their role is mainly to improve the performance and training efficiency of the model. Residual connectivity makes the model deeper and thus improves representativeness, while the normalization layer mitigates the problem of vanishing gradients and explosions by normalizing the data, thus improving the stability and performance of the model. Layer

normalization is to normalize the inputs $(x_1, x_2, \cdots, x_d)$ of the current layer to obtain $(x_1', x_2', \cdots, x_d')$, and the overall computation process is shown in Eqs. (7) ~ (9) are shown.

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i \tag{7}$$

$$\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2 + \varepsilon} \tag{8}$$

$$x_i' = f\left(g\frac{x_i - \mu}{\sigma} + b\right) \tag{9}$$

where g and b are parameters and f is a nonlinear function.

III. Transformer-based word vector computation model for English corpus

In this paper, we will use word vector computation, based on the principle of Transformer model, to construct a word vector computation model Transformer-DSSM for English corpus.After the semantic similarity and distance calculation between the vectors, the relationship between words in English corpus is objectively and realistically reflected.

III. A. Model Architecture

In this paper, we propose the semantic similarity computation model is to replace the initial representation layer of the DNN network with the deep semantic model as the base layer, and utilize the coding layer of the Transformer for feature extraction, and the matching layer will compute the correlation coefficient of the two short texts obtained from the upper layer, which is mostly used as a cosine function.

III. A. 1) Improvements to the DSSM model

The original DSSM model is mostly used in retrieval scenarios to train semantic level matching using the user's click data, when more clicks indicate higher relevance, which allows linking Query and Doc together [18].

A word2vec approach is used to do the representation of word vectors. Use an open-source third-party Python library called gensim to perform word vector mapping. Gensim can learn regular vector representations from messy raw corpora, making the word embeddings obtained in this way direct, efficient, and easy to operate.

It is very convenient to use gensim to train word2vec, the training steps are as follows:

1) the English corpus preprocessing: each line is only a short text, for the Chinese and English processing is different, the Chinese should be first split, words and words by the space between the words, this paper chooses the splitter tool is the jieba splitter; for the English language, there is a natural space separation, so there is no need to split the words, direct use.

2) After processing the corpus will be written as an iterator, each iteration of the returned sentence is a list of words, you can use gensim in word2vec.py in the LineSentence() method to achieve.

3) Just input the result of the above processing into the word2vec object built into gensim for training.

III. A. 2) Fixed-length patching

In the work of mapping word vectors, the problem of unregistered words often arises. Unregistered words, also called raw words, are interpreted in two ways: firstly, they are words that are not recorded in the existing vocabulary; secondly, they are words that are not in the English corpus that we have now. Based on the second interpretation, unoccurring words are also out-of-set words (OOV), i.e., words that do not appear in the training set. The more common way to deal with the OOV problem is to add a new UNK token as an unregistered word in the vocabulary list, and then randomly initialize that word vector.

In this paper, in order to ensure the completeness of the input as well as the subsequent effect of the model, we use the fixed-length complementary approach, first manually set the length of each input sentenceLength, if a certain document mapped as a word vector is smaller than the sentenceLength, it will be complemented with UNK; if it is larger than that, it will be rounded off.

III. A. 3) Category imbalances

The current methods for dealing with the category imbalance problem are: undersampling, oversampling, and individual processing samples. The two methods of undersampling and oversampling are briefly described below:

1) Undersampling

Undersampling refers to the uneven categories in the data set, remove most of the categories of some of the samples to make the data set of samples of roughly equal categories. Random undersampling is in the original sample set, randomly select the majority of the categories of samples in some of the samples, these samples from the data set to delete.

2) Oversampling

The idea of oversampling is the opposite of undersampling, which is to add some of the samples of a few categories in the data set, so that the proportion of categories in the data set is relatively balanced. Generally in a few categories of data set randomly select the existing data to add to the data set, this way to become random oversampling.

III. B. Model implementation

III. B. 1) Input layer

The input layer of the model, the word vectors are first defined and then each word is mapped to the corresponding word vector, which can be obtained as two vector matrices.

Gated Recurrent Unit (GRU), GRU is an effective variant of LSTM network, but its structure is relatively less complex [19]. One of the major features of LSTM is that it can solve the problem of long and short term dependency, as its variant, GRU network can naturally be solved. The difference is that three gate functions input gate, forget gate and output gate are introduced in LSTM, and one less gate function is replaced by update gate and reset gate in GRU.

The size of the reset gate determines the degree of inflow of information from the previous time step, the larger it is the more information from the previous time step is retained by the current state. The forward propagation formula for the network is as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{10}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{11}$$

$$\overline{h}_t = \tan h\left(W \cdot [r_t * h_{t-1}, x_t]\right) \tag{12}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{13}$$

III. B. 2) Representation layer

The representation layer of the model is proposed based on the Transformer coding layer, and each coding layer can be decomposed into two parts, one is the network layer under the self-attention mechanism, and the other is the fully-connected network layer, and at the same time normalization adjustment is to be performed among each other.

In this paper, the Transformer coding layer with multiple coding blocks is used as the representation layer of the model for text feature extraction, and its input is the feature vector and position vector stated in the previous layer, which passes through three coding blocks in turn, and the output of the previous block is used as the input of the next block, and the single coding block mainly consists of two parts: the multi-head attention mechanism and the feed-forward neural network.

After the feature extraction of the input data, the position coding is added and then it is brought to the coding layer part of the Transformer for the computation of self-attention, normalization and full connectivity.

III. B. 3) Matching layer

After encoding the vocabulary separately using the Transformer encoder, the semantic information representation is obtained separately.

The semantic information of the question is denoted as Q, and the semantics of the vocabulary is denoted as D. Dotwise multiplication is performed to obtain an interactive information matrix about the question and word set, and the values of the elements in the matrix are the correlation between the question and the vocabulary, and the mutual information matrix is then probabilized row-by-row and column-by-column by using a softmax layer to get the attentional weights of the vocabulary to the question α and of the question to the We can get the attention weight from the word to the question α and the attention weight from the question to the vocabulary β. Then multiply α and β by Q and D to get the representation based on the attention mechanism, and then the cosine value can be used for similarity calculation.

$$y_Q = \alpha \cdot Q \tag{14}$$

$$y_D = \beta \cdot D \tag{15}$$

$$R(Q,D)=\cos(y_Q,y_D)=\frac{y_Q^T y_D}{\| y_Q \| \| y_D \|} \tag{16}$$

III. B. 4) Gradient update section

The model in this paper is a supervised learning and thus requires training the network by minimizing the loss function. The input to the model is given a probability defined as:

$$P(D\mid Q)=\frac{exp(\gamma R(Q,D))}{\sum_{D'\in D} exp(\gamma R(Q,D'))} \tag{17}$$

where γ is the empirical coefficients of the variables, in order to make the softmax function smoother, D and Q are the set of all docs in the set:

1) The set of all docs associated with Q in the training sample (positive sample), which is denoted by D^+ in the following.

2) In the training sample, with respect to each Q, also randomly selected negative samples (docs not related to Q) from the database, denoted below by D^-, the set D contains all D^+ and D^- corresponding to Q, i.e.:

$$D^+ \in D \tag{18}$$

$$D^- \in D \tag{19}$$

The loss function used by the model is the cross-entropy loss function, which is the inverse of the likelihood function for all positive samples after a given Query:

$$\begin{aligned} L(A) &= -\log \prod_{(Q,D^+)} p(D^+\mid Q) \\ &= -\sum_{(Q,D^+)} \log p(D^+\mid Q) \end{aligned} \tag{20}$$

The function model uses the cross-entropy loss function as the loss function of the model, the two-dimensional vector will get the conditional probability value of each sample after passing softmax, select the value of the first column and use the loss function to calculate the loss value. At the same time the model uses Adam optimizer to accelerate the training of the model.

IV. Transformer-DSSM model performance experiments

In this chapter, the Transformer-based word vector computation model for the English corpus constructed in this paper (hereafter referred to as the “Transformer-DSSM model”) will be quantitatively analyzed on the SICK and MSRP datasets to qualitatively assess the Transformer-DSSM model's Performance.

IV. A. Experimental environment

The detailed configuration of the experimental environment for this section is shown in Table 1. The programming language used in this experiment is Python 3.5. Python has a collection of many AI toolkits that can quickly build models. Currently commonly used deep learning frameworks include Tensorflow, Theano and Pytorch, etc. Transformer-DSSM is built using Google's open-source Tensorflow, which is based on data flow graphs, and is able to be trained on CPU or GPU.

Table 1: Experimental environment setting

Experimental environment setting	Content
Operating system	Win 10
CPU	Intel(R) Core(TM) i5-8250U CPU@1.60GHz(8)
Graphics card	NVIDIA GeForce MX150
Memory	8G
Programming language	Python 3.5
Deep learning framework	Tensorflow 1.13

IV. B. Experimental data sets

The datasets used in this experiment are the SICK dataset and the MSRP dataset, respectively.The basic conditions of the SICK and MSRP datasets, including the data volume, sentence length, and the distribution of positive and negative samples, are shown in Table 2. It can be seen that the average length of the MSRP dataset is much larger than the average length of the SICK dataset, which means that the sentences in the MSRP dataset contain more semantic information.

Table 2: Statistics of SICK and MSRP data sets

Dataset	SICK				MSRP		
	Training	Verification	Testing	Overall	Training	Testing	Overall
Amount of data	4500	500	4927	9927	4076	1725	5801
Average length	9.68	9.89	9.66	9.68	20.95	20.78	20.9
Maximum length	36	30	30	36	41	37	41
Minimum length	3	4	3	3	6	7	6
Proportion of positive and negative samples	2.74	3.03	2.87	2.82	2.08	1.98	2.05

IV. C. Analysis of experimental results

IV. C. 1) SICK Test Set Experimental Results and Analysis

The performance of the Transformer-DSSM model in this paper with other comparative models on the SICK dataset is specifically shown in Table 3. The γ, ρ Pearson's coefficient, Spearman's coefficient, and MSE are the mean square error in the table. Ou From the experimental results, it can be seen that the Sentence-BERT model, although using a powerful BERT model to model sentence pairs, its use of twin BERT to model the input two sentences separately, lack of sentence interaction information, and has the lowest performance among all the compared models. The Transformer-DSSM model in this paper, on the other hand, has the highest performance among all Xu models, with the highest γ value and ρ value of 0.887 and 0.845, respectively, and the lowest MSE value of 0.2286.

Table 3: Experimental results on the SICK test set

Model	γ	ρ	MSE
Meaning Factory	0.827	0.7728	0.3231
ECNU	0.841	0.7691	0.3257
WSV-SCNN-SV	0.8125	0.758	0.3612
BiLSTM	0.8516	0.7957	0.2851
Attentive BiLSTM	0.8548	0.7981	0.2742
Sentence-BERT	-	0.7447	-
Tree-LSTM	0.8674	0.8075	0.2533
MPCNN	0.8684	0.8057	0.2603
PWIM	0.8794	0.8206	0.2337
Transformer-DSSM	0.887	0.845	0.2286

IV. C. 2) MSRP Test Set Experimental Results and Analysis

The performance of this paper's Transformer-DSSM model on the MSRP dataset with other comparative models is specifically shown in Table 4. The Transformer-DSSM model in this paper is very competitive, differing only by 0.011 (Acc) and 0.010 (F1) from the best results, but with shorter training time and high interpretability. The multi-granularity learning model relies on pre-training on the language modeling task and does not perform as well as the Transformer-DSSM model in this paper when there is no pre-training.The MPCNN model uses some additional resources and the Transformer-DSSM model in this paper performs comparably to this model when it does not use external resources.The ABCNN model also requires a range of external sparse features and a layer-by-layer training strategy. In contrast, the Transformer-DSSM model in this paper does not require any additional resources, external features, or pre-training strategies, and obtains competitive results.

V. Simulation experiment on word vector computation for the English corpus

In this chapter, we will carry out word vector computation simulation experiments on the English corpus real news recommendation datasets MIND and Adressa to explore the effectiveness of the Transformer-DSSM model

proposed in this paper in the representation of word vectors, and to test the practicality of the Transformer-DSSM model in this paper.

V. A. Experimental data set

MIND dataset is a news dataset released by Microsoft in 2020, and the data mainly comes from MSN News, which provides well-divided training and testing sets.Adressa dataset is a news recommendation dataset released by the Norwegian University of Science and Technology (NTNU) in cooperation with Adressavisen, and it contains three-months' log data. The statistical information of the two datasets is specifically shown in Table 5, and the average word count of the body of MIND and Adressa dataset are 584.37 and 518.5 respectively.

Table 4: Experimental results on the MSRP test set

Model	Acc	F1
Multi-granularity learning (without pretraining)	72.50%	81.40%
Multi-granularity learning (with pretraining)	78.10%	84.40%
Multi-granularity learning (with extra MT metrics)	78.40%	84.60%
MPCNN (without POS embeddings)	77.80%	-
MPCNN (without Para. embeddings)	77.30%	-
MPCNN (POS and Para. embeddings)	78.60%	84.70%
ABCNN (with sparse features)	78.90%	84.80%
Transformer-DSSM	77.80%	83.80%

Table 5: Statistics of MIND and Adressa datasets

Dataset	Number of users	Number of news	Number of clicks	Average number of words in the title	Average number of words in text
MIND	94066	65348	7817780	11.55	584.37
Adressa	3083348	48575	2722460	6.82	518.5

V. B. Evaluation indicators

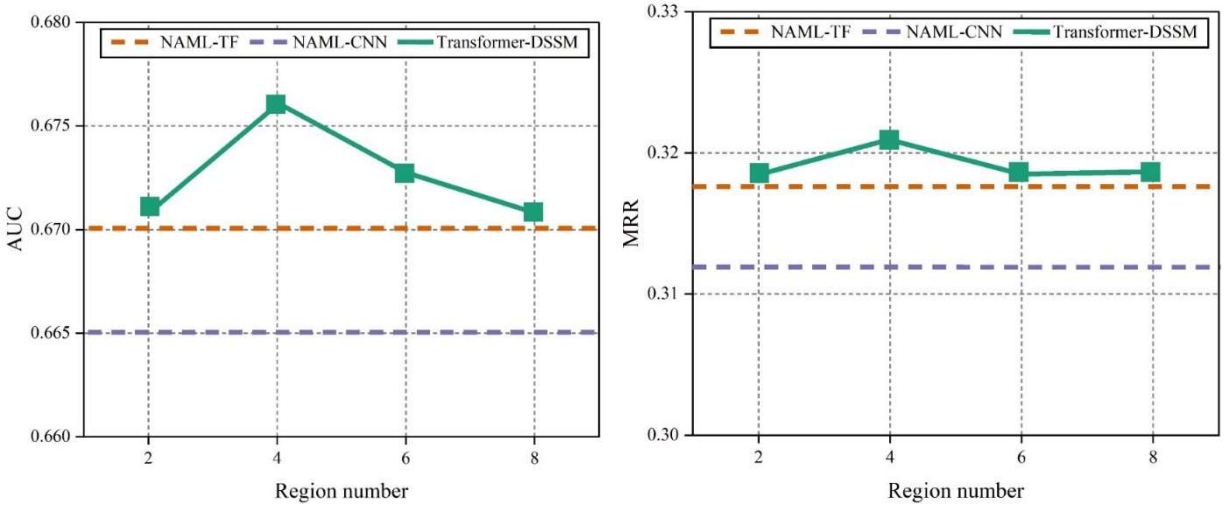
In this paper, four metrics, AUC, MRR, nDCG@5 and nDCG@10, are used on the MIND dataset; two metrics, AUC and F1, are used on the Adressa dataset.

V. C. Analysis of experimental results

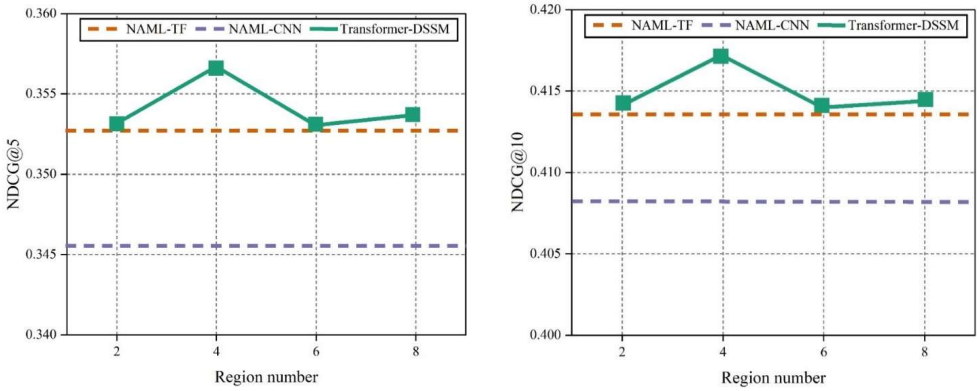
V. C. 1) Results and analysis of experiments on the selection of the number of regions

In order to explore the problems related to the Transformer-DSSM model in the selection of body regions, this paper has done experiments related to the selection of the number of regions. The parameter configuration of the optimal performance of the NRTA model on each dataset is used in the experiments, and the length of the fixed region is L=50. The comparison model selected for this experiment is the NAML model, and in this paper, we will use the NAML model based on two text encoders, Transformer and CNN, respectively, to compare the Transformer-DSSM model. The variation of each metric of the Transformer-DSSM model with the number of regions on the MIND dataset is specifically shown in Fig. 2, and Figs. (a) to (d) correspond to the AUC, MRR, nDCG@5, and nDCG@10 metrics, respectively. The orange dashed line in the figure marks the optimal value of NAML model with CNN text encoder, and the purple dashed line marks the optimal value of NAML model with Transformer text encoder. It can be seen that, on the MIND dataset, the Transformer-DSSM model in this paper always achieves the optimal results in each index, and the values of each index are higher than the optimal values of the NAML model.

The variation of the Transformer-DSSM model metrics with the number of regions on the Adressa dataset dataset is specifically shown in Figure 3. Figures (a) and (b) correspond to AUC and F1 indicators, respectively. On the Adressa dataset, the Transformer-DSSM model in this paper still achieves the optimal results in each index, which is higher than the optimal values of the NAML model based on Transformer and CNN text encoders respectively.

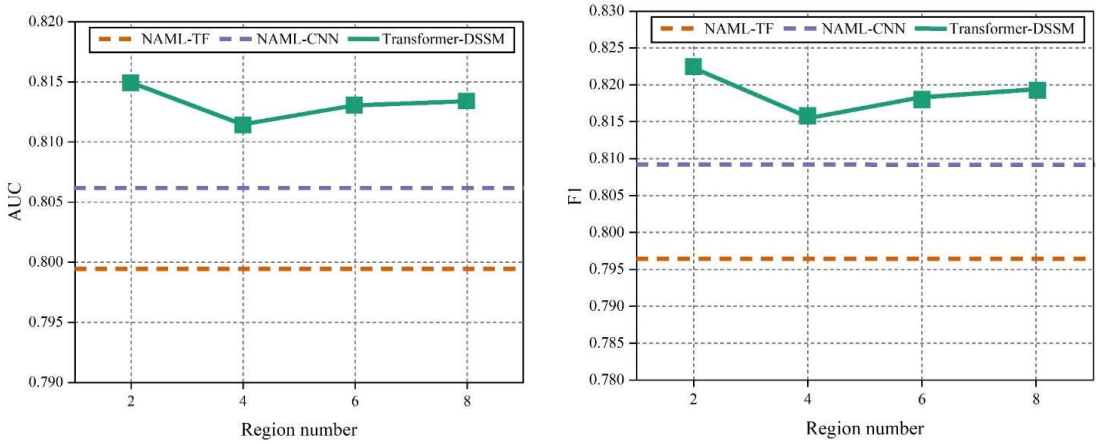


(a)AUC (b)MRR



(c)nDCG@5 (d)nDCG@10

Figure 2: Change of various indicators on MIND dataset



(a)AUC (b)F1

Figure 3: Change of various indicators on Adressa dataset

Overall, selecting the beginning and the end of the body can make the Transformer-DSSM model achieve good results on both datasets, and it is recommended to select the area covering the first 50-100 words at the beginning of the body and 100-200 words at the end of the body when applying the NRTA model.

V. C. 2) Regional Length Experiment Results and Analysis

In this paper, we compare the performance of Transformer-DSSM model under different region lengths, and the optimal parameter configuration and optimal region selection method of Transformer-DSSM model are still used in the comparison. The changes of each indicator of Transformer-DSSM model with the length of the region on the MIND dataset are specifically shown in Fig. 4. Figures (a) to (d) represent AUC, MRR, nDCG@5, and nDCG@10 metrics, respectively. When the Transformer-DSSM model adopts the optimal region selection method, the region length can significantly outperform the NAML model in each metric of the two datasets whether it is 20 words, 50 words or 100 words.

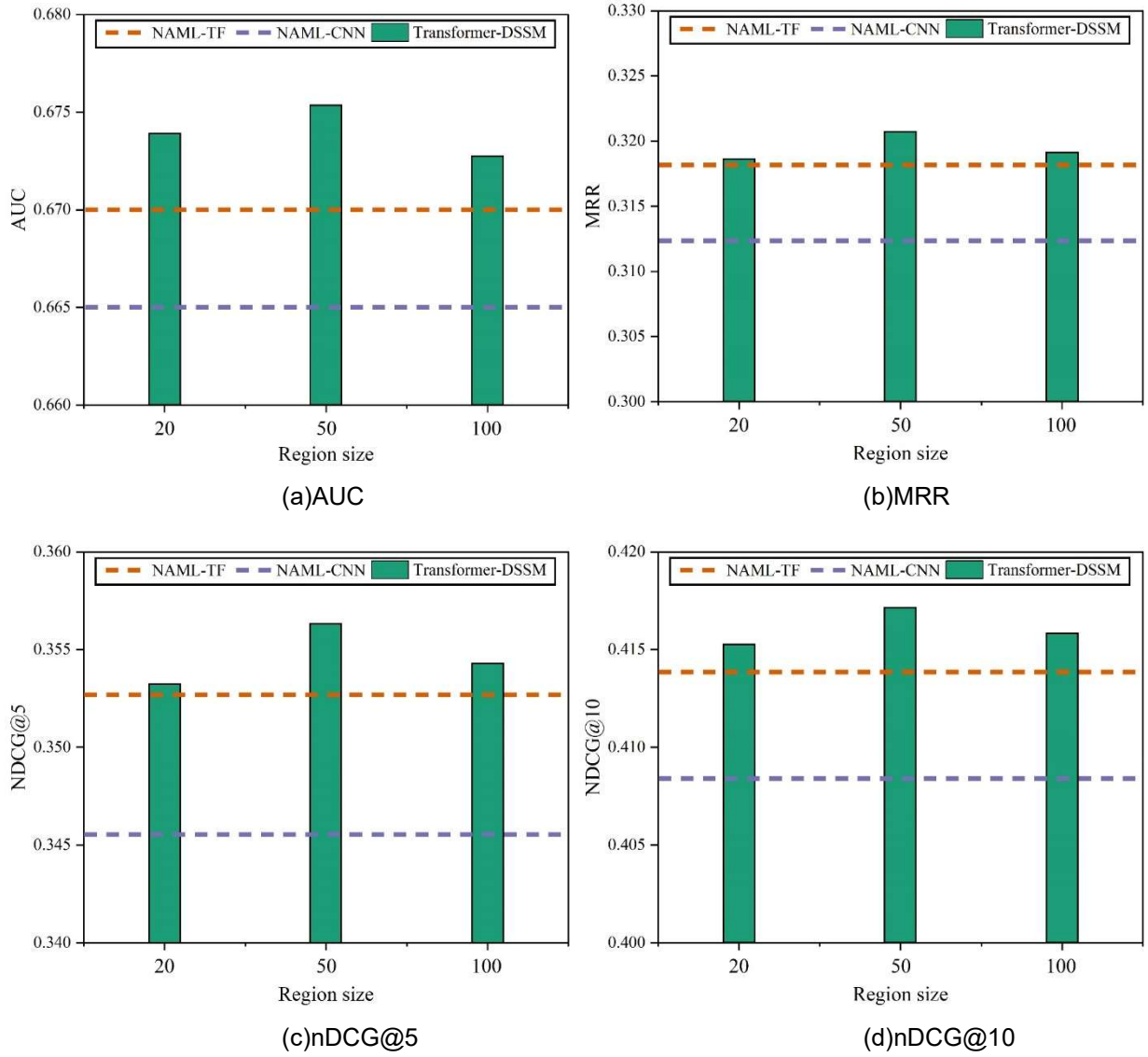
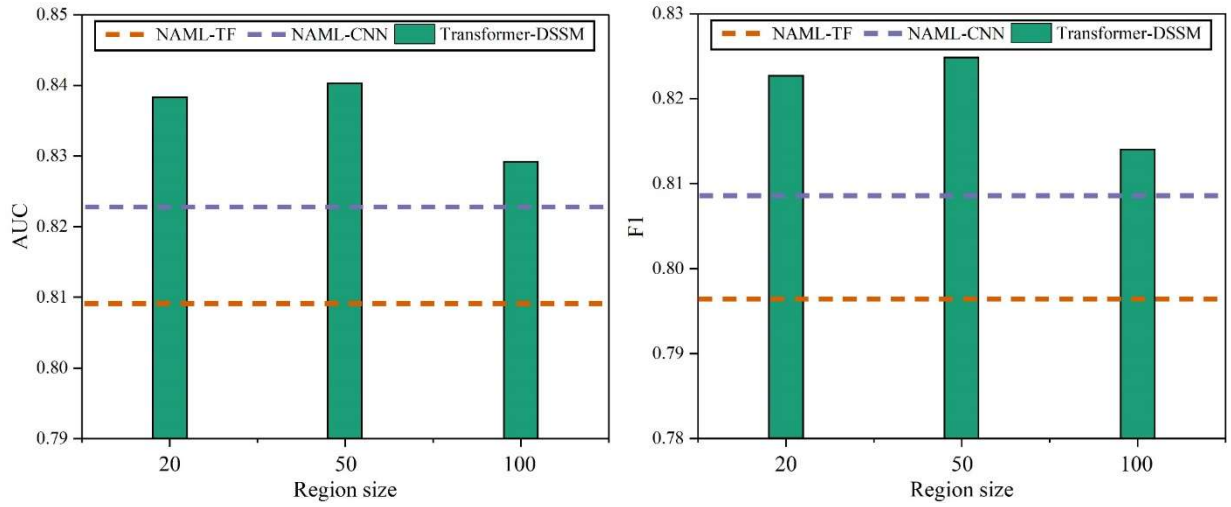


Figure 4: Change of various indicators with different region sizes on MIND dataset

The variation of Transformer-DSSM model metrics with region length on the Adressa dataset is specifically shown in Figure 5. Figures (a) and (b) correspond to AUC and F1 metrics, respectively. It can be seen that on the Adressa dataset, the Transformer-DSSM model still significantly outperforms the comparative NAML model in the case of any region length on the AUC and F1 metrics.



(a)AUC (b)F1

Figure 5: Change of various indicators with different region sizes on Adressa dataset

Overall, a region length of 50 words is optimal on both datasets, so a body region length of around 50 words is recommended when applying the Transformer-DSSM model.

VI. Conclusion

In this paper, Transformer-DSSM, an English corpus word vector computation method based on the Transformer model, is proposed, and the validity and practicality of the method is verified through a series of experiments.

On the SICK test set, the Transformer-DSSM model achieves the optimal performance of Pearson's coefficient of 0.887 and Spearman's coefficient of 0.845, and the mean square error is reduced to 0.2286, which is better than many existing comparison models. On the MSRP test set, the model obtains 77.8% accuracy and 83.8% F1 value, which differ from the optimal model performance by only 0.011 and 0.010, and without the use of additional resources, external features or pre-training strategies. Simulation experiments on real English news recommendation datasets MIND and Adressa show that the Transformer-DSSM model outperforms the baseline model NAML for different number of regions and region lengths. The experimental results find that, when selecting region configurations of the first 50 to 100 words at the beginning of the body text and 100 to 200 words at the end, as well as setting the region length to around 50 words, the model achieves the best results on both datasets. Through the mechanism of multi-head attention, the Transformer-DSSM model is able to effectively capture the long-distance semantic dependencies between words, which solves the problem of insufficient treatment of polysemous words in traditional word vector computation.

The method proposed in the study provides a new solution for word vector computation in English corpus, which is of positive significance for improving the performance of natural language processing systems.

Acknowledgement

This work was supported by Innovation and Practice of an Educational Model for Vocational Undergraduate Public Basic Courses Emphasizing Both Moral and Technical Training, Industry-Education Integration, and Integrated Assessment, Henan Provincial Vocational Education Teaching Reform Project (2024.05858).

References

[1] Deng, L., & Zhao, Y. (2023). Deep learning-based semantic feature extraction: A literature review and future directions. ZTE communications, 21(2), 11.

[2] Kabra, B., & Nagar, C. (2023). Convolutional neural network based sentiment analysis with tf-idf based vectorization. Journal of Integrated Science and Technology, 11(3), 503-503.

[3] Sabri, T., El Beggar, O., & Kissi, M. (2022). Comparative study of Arabic text classification using feature vectorization methods. Procedia Computer Science, 198, 269-275.

[4] Liu, N., Tan, Q., Li, Y., Yang, H., Zhou, J., & Hu, X. (2019, July). Is a single vector enough? exploring node polysemy for network embedding. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 932-940).

[5] Yang, X., & Mao, K. (2016). Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge. Expert Systems with Applications, 56, 291-299.

[6] Vicente, A. (2018). Polysemy and word meaning: An account of lexical meaning for different kinds of content words. Philosophical Studies, 175(4), 947-968.
[7] Martelli, F., Perrella, S., Campolungo, N., Munda, T., Koeva, S., Tiberius, C., & Navigli, R. (2025). DiBiMT: A Gold Evaluation Benchmark for Studying Lexical Ambiguity in Machine Translation. Computational Linguistics, 1-71.
[8] Kaddoura, S., & D. Ahmed, R. (2022). A comprehensive review on Arabic word sense disambiguation for natural language processing applications. Wiley interdisciplinary reviews: data mining and knowledge discovery, 12(4), e1447.
[9] Lopukhina, A., Laurinavichyute, A., Lopukhin, K., & Dragoy, O. (2018). The mental representation of polysemy across word classes. Frontiers in psychology, 9, 192.
[10] Kapitanov, A., Kapitanova, I., Troyanovskiy, V., Ilyushechkin, V., & Dorogova, E. (2019, January). Clustering of Word Contexts as a Method of Eliminating Polysemy of Words. In 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (pp. 1861-1864). IEEE.
[11] Jain, D. K., Eyre, Y. G. M., Kumar, A., Gupta, B. B., & Kotecha, K. (2024). Knowledge-based data processing for multilingual natural language analysis. ACM Transactions on Asian and Low-Resource Language Information Processing, 23(5), 1-16.
[12] Opitz, J., Wein, S., & Schneider, N. (2025). Natural language processing relies on linguistics. Computational Linguistics, 1-23.
[13] Nematzadeh, A., Meylan, S. C., & Griffiths, T. L. (2017). Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 39).
[14] Kumari, A., & Lobiyal, D. K. (2022). Efficient estimation of Hindi WSD with distributed word representation in vector space. Journal of King Saud University-Computer and Information Sciences, 34(8), 6092-6103.
[15] Zhou, S., Xu, X., Liu, Y., Chang, R., & Xiao, Y. (2019). Text similarity measurement of semantic cognition based on word vector distance decentralization with clustering analysis. IEEE Access, 7, 107247-107258.
[16] Wang, Y., Hou, Y., Che, W., & Liu, T. (2020). From static to dynamic word representations: a survey. International Journal of Machine Learning and Cybernetics, 11, 1611-1630.
[17] Ilias Kalouptsoglou,Miltiadis Siavvas,Apostolos Ampatzoglou,Dionysios Kehagias & Alexander Chatzigeorgiou. (2025). Transfer learning for software vulnerability prediction using Transformer models. The Journal of Systems & Software,227,112448-112448.
[18] Mishra Ravita & Rathi Sheetal. (2022). Enhanced DSSM (deep semantic structure modelling) technique for job recommendation. Journal of King Saud University - Computer and Information Sciences,34(9),7790-7802.
[19] M. Jayamohan & S. Yuvaraj. (2025) .A novel human action recognition using Grad-CAM visualization with gated recurrent units. Neural Computing and Applications,(prepublish),1-16.