Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **10**, Number **1**, February **2025**

English Sentence Similarity Algorithm Based on Vector Space Model

Qiao-Hua Lu*

School of Humanities
Guilin College, Guilin 541006, P. R. China
luqiaohua2023@163.com

David Huang

School of IT Art and Design
Sehan University, Jeollanam-do 58447, South Korea
vb9080@163.com

*Corresponding author: Qiao-Hua Lu
Received February 24, 2024, revised May 29, 2024, accepted July 28, 2024.

Abstract. *English text similarity computation provides an effective way to measure and compare the semantic similarity between texts, which can be widely used in natural language processing tasks such as information retrieval, recommender systems, question and answer systems, etc. However, most similarity algorithms based on vector space models suffer from unreasonable word weights and ignore the semantics of words. Therefore, an English utterance similarity on vector space model is proposed. Firstly, the principle of text similarity calculation based on vector space model is introduced, and the implementation method of word embedding model is given. Then, for the problem that the traditional TF-IDF algorithm does not consider the inter-class and intra-class distributional information of feature words, a TF-IDF algorithm (IGDTF-IDF) combining information gain and intra-class discretisation is proposed and introduced into the vector space model. Then, a monolingual Skip-Gram word embedding model is constructed by employing word alignment information, and a weighted summation is performed using IGDTF-IDF to achieve a vectorised representation of the text. Finally, combined with the classical cosine similarity calculation method, an English text similarity measure based on word embedding and co-occurrence correlation is proposed. Ten sample sets of different categories on the SICK dataset are selected for simulation testing. The results show that compared with TF-IDF, STF-IDF and TFPOS-IDF, IGDTF-IDF improve 6.69%, 3.22%, and 1.2% in terms of Accurary, respectively. When = 0.55, the Top10, Top20 and Top30 retrieval Recalls of the proposed English text similarity metrics reach the maximum value.*
Keywords: English text; Text similarity; Vector space modelling; Word embedding; Skip-Gram; TF-IDF

1. **Introduction.** In the field of Natural Language Processing, text similarity computation is a necessary and fundamental aspect, and its application in the fields of automatic answering, machine translation, information checking, text checking, etc. is already mature [1, 2]. It is worth noting that there is no standard definition of similarity at present, and it usually changes according to the scenarios. In machine translation, since a word can be interpreted by multiple words with the same or similar meanings, the focus is on calculating the similarity between two words [3]. In automatic question and answer systems, the corresponding answers are automatically obtained by measuring the semantic

match between the questions asked by the user and the existing questions in the question and answer database. In common checking systems, they first divide the checked text into paragraphs and their focus is to calculate the similarity between two paragraphs. In information retrieval, it considers how well the retrieved content matches the text content retrieved from a collection of texts, and a model is used to calculate the similarity between the texts. It follows that the meaning of similarity can vary from domain to domain [4].

English text similarity computation provides an effective way to measure and compare the semantic similarity between texts, which can be widely used in natural language processing tasks such as information retrieval, recommender systems, question and answer systems, etc [5, 6]. However, most similarity algorithms based on vector space models suffer from unreasonable word weights and ignore the semantics of words. The research objective of this work is to study and implement effective algorithms for word-item relevance metrics and text similarity metrics for English language corpus. In this paper, the word embedding model is trained through a monolingual corpus, and based on this, the vector representation of English text is implemented. The word co-occurrence relation model of English text is constructed through the monolingual corpus, and the model is used for English text similarity metrics.

1.0.1. *Related work.* Text similarity has been developed to a very mature level and is widely used in all major fields of natural language processing. Currently, there are many methods for calculating monolingual text similarity, which are generally classified into [7, 8, 9]: String-Based method, Corpus-Based method and Knowledge-Based method.

(1) String-Based Methods

String-based methods use the length or distance of character matches between two texts as a measure of similarity based on the string sequence or character combination form of the original text. Leonardo and Hansun [10] found that Levenshtein's algorithm can obtain better accuracy than Rabin-Karp's algorithm in calculating the hash distance in two texts. Zhou et al. [11] proposed the use of longest common sequence as a new text feature for text similarity detection, which improves the performance of automatic detection of reported duplication errors. Chen et al. [12] proposed a method to improve the N-gram language model using text generated by Recurrent Neural Network Language Model (RNNLM) to improve the performance of speech recognition.

(2) Knowledge-Based Approach

The knowledge base-based similarity calculation method obtains the amount of information through a manually constructed knowledge base with a system of rules, which is used to quantify the degree of semantic association between two texts. Knowledge base-based text similarity computation methods utilise external knowledge bases to enhance the accuracy and semantic understanding of text similarity computation. These knowledge bases can be structured, such as atlases or ontologies, or unstructured, such as large-scale text corpora. Kim et al. [13] proposed to improve the plain Bayesian text classifier using a Wikipedia-based semantic tensor space model to achieve near-perfect classification performance. Karuppaiah and Vincent [14] proposed a cross-language text similarity computation method based on WordNet. Wu et al. [15] proposed a lexical similarity calculation method based on Baidu encyclopedia entries, which improves the accuracy of lexical similarity calculation.

(3) Corpus-Based Approach

The corpus-based methods are completely dependent on the corpus, and measure the semantic similarity of the words in the corpus according to their co-occurrence frequency, and the words with similar contexts usually contain similar semantics. Based on the different ways of constructing text vectors, corpus-based approaches include Vector Space

Model (VSM), Topic Model, and Neural Network Model. Chen et al. [16] proposed a method for calculating text similarity based on topic model. Firstly, topic models (e.g. Latent Dirichlet Allocation, LDA) are used to learn the implied topic structure from the corpus. Then, the similarity between texts is computed by comparing the degree of similarity between texts in terms of topic distribution. This method is able to reveal the semantic relationships between texts and overcome some limitations in the traditional methods based on word frequency and word vectors. Han et al. [17] introduced a neural network based text similarity calculation method. By building an RNN model for two input texts separately and then performing similarity computation at the output layer, the method is able to capture the complex semantic relationships between texts. The model achieves excellent performance on several text similarity computation tasks. Agarwala et al. [18] proposed a framework for text similarity computation that combines vector space models and word embeddings. First, a vector space model is constructed based on TF-IDF and cosine similarity, and then pre-trained word embedding models (e.g., Word2Vec or GloVe) are used to enhance the semantic representation of the text. Finally, the similarity between texts is calculated by fusing the vector space model and the word embedding model. This method integrates lexical and semantic information and provides a more accurate text similarity calculation.

1.1. **Motivation and contribution.** It is observed that the Corpus-Based method above only counts the frequency of occurrence of words in the text, which is very simple to calculate, but it ignores the semantic information that the words are intended to express in the text and the contextual information of the words in the text, which leads to a decrease in the accuracy of the similarity value. In addition, linguistic features cannot be directly located in the same text similarity cannot be directly calculated for texts represented by features in different vector spaces. Therefore, this work is based on word embedding and improved vector space model for English text similarity metrics. The main innovations and contributions of this work include:

(1) An improved vector space model is proposed. In the TF-IDF weight calculation method, the lower the number of occurrences of a word in the text collection or the more times a word appears in the text, the more important the word is, without considering the influence of the distribution of the word within the class on its results, so this paper incorporates the information gain and intra-class discretisation into the TF-IDF algorithm and calls it IGDTF-IDF.

(2) The Skip-Gram word embedding model is trained and learnt for the word alignment information of text alignment, and the word vector representations of different language word items located in the same space are obtained. According to the co-occurrence correlation between English word items, the English text similarity calculation method based on IGDTF-IDF and co-occurrence correlation is proposed.

2. **Analysis of relevant principles.**

2.1. **Vector space models.** VSM is more maturely used in text processing [19, 20]. It assumes that the meanings of a text are related only to the words that can express those meanings and the frequency of those words in the text, but not to the position of those words in the text [21]. That is, the meaning of a text can be determined by the properties of the words in that text themselves and the frequency with which those words appear in the text. Therefore, the similarity between two texts can be calculated based on the characteristic words that are present in both texts and how often they appear in their respective texts. The purpose of the vector space model is to model all the features of

a text as elements of a vector space. These vectors obey some basic algebraic rules as follows:

$$\overline{x} + \overline{y} = \overline{y} + \overline{x} \tag{1}$$

where x and y denote lexical items; $\overline{x}$ and $\overline{y}$ denote vectors of lexical items.

Suppose $d_1, d_2, \ldots, d_n$ are all the features used to represent the text. For each feature d_i, there exists a vector $\overline{d_i}$ in the vector space. Let the unit length vector be $\overline{d_i}$. Now, suppose the text D_j is a vector represented by the unit vector $\overline{d_i}$, where $1 \leq j \leq m$. The expression for the text vector $\overline{D_j}$ is then [22].

$$\overline{D_j} = [w_{j1}, w_{j2}, \ldots, w_{jn}] \tag{2}$$

Each vector in a text vector space, including all text vectors, is a linear combination of unit vectors. Thus, the text vector $\overline{D_j}$ can be equivalently expressed as:

$$\overline{D_j} = \sum_{i=1}^{n} w_{ij}\overline{d_i} \tag{3}$$

where the coefficients w_{ij} are the weights on $\overline{d_i}$, $1 \leq i \leq n, 1 \leq j \leq m$.

The frequently implemented weighting technique for VSMs is TF-IDF [23]. The TF-IDF weights are calculated as follows:

$$tf\_idf = tf \times \frac{N}{df} \tag{4}$$

where N is the total number of training samples in the text set, and df is the number of samples in the training sample set in which feature t appears.

2.2. **Text similarity calculation based on VSM.** Nowadays, it is common to use $[0, 1]$ to indicate the degree of similarity between two texts or multiple texts [24]. If two texts are more similar in terms of paragraph structure and semantics, it means that their similarity is higher; on the contrary, if there is no relevant semantics and paragraph structure between them, their similarity is considered to be 0; if they are the same in terms of paragraph structure and semantics, their similarity is defined to be 1. After text data cleaning, feature engineering and feature selection, VSM will show it as a vector. Then, the similarity between texts is determined by the relationship between text feature vectors.

Suppose the feature vector of two texts d_i is $\overline{v_{d_i}} = (w_{i1}, w_{i2}, \ldots, w_{in})$, and the feature vector of d_j is $\overline{v_{d_j}} = (w_{j1}, w_{j2}, \ldots, w_{jn})$. If the angle between the vectors is θ, the similarity between d_i and d_j is calculated as shown below:

(1) Vector dot product

$$Sim(d_i, d_j) = \overline{v_{d_i}} \cdot \overline{v_{d_j}} = \sum_{k=1}^{n} w_{ik} w_{jk} \tag{5}$$

(2) Jaccard's coefficient method

$$\mathrm{Sim}\,(d_i, d_j) = \frac{\sum_{k=1}^{n} w_{ik} w_{jk}}{\sum_{k=1}^{n} w_{ik}^{\ 2} + \sum_{k=1}^{n} w_{jk}^2 - \sum_{k=1}^{n} w_{ik} w_{jk}} \tag{6}$$

(3) Dice coefficient method

$$Sim(d_i, d_j) = \frac{2\sum_{k=1}^{n} w_{ik} w_{jk}}{\sum_{k=1}^{n} w_{ik}^2 + \sum_{k=1}^{n} w_{jk}^2} \tag{7}$$

(4) Cosine coefficient method

$$Sim(d_i, d_j) = \cos\theta = \frac{\sum_{k=1}^{n} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{n} w_{ik}^2} * \sqrt{\sum_{k=1}^{n} w_{jk}^2}} \tag{8}$$

2.3. **Word embedding techniques.** Most blanc language processing and comprehension tasks involve words, but instead of representing words as independent symbols, they are generally represented as word representations reflecting their semantic relevance. Words with similar contexts have similar semantics. Therefore a number of word representations have been investigated, the vast majority of which can be described in terms of word-item-document matrices. Word embedding technique is to represent words as dense vectors.

The concept of Distributed Representation of words, also known as word embedding, was first introduced by Bengio et al. in 2003 [25]. Words are represented vectorially by training Neural Network Language Model (NNLM).

The NNLM is actually a three-layer fully connected neural network model, as shown in Figure 1. Firstly, the model maps the words in the dictionary to a dense space of fixed dimensions using the parameter matrix V. The mapping of a word in the dense space is the word vector representation of the word. Second, the model uses the *tanh* activation function to map the context corresponding to each word into the space of conditional probability distributions corresponding to all words of the dictionary. Finally, the model learns both the mapping relationship of the word vectors and the conditional probability of occurrence of the target word in the context during the training process, and does the normalisation using the $softmax$ function.

Word2Vec word embedding model is proposed on the basis of NNLM. Word2Vec model includes CBOW model [26] and Skip-Gram model [27]. Word2Vec word embedding model abandons the practice of considering only w_t above in NNLM, and considers the context of w_i instead.

The CBOW model predicts the current target word w_t by inputting the first n words and the next n words of the target word w_t through a neural network. The first n words and the last n words of the target word w_t are called the context of w_t, denoted as $Context(w_t)$. The CBOW model consists of three layers: an embedding layer, a mapping layer and an output layer.

The Skip-Gram model is a neural network that predicts the first n words and the next n words of the current target word w_t. The Skip-Gram model also consists of an embedding layer, a mapping layer, and an output layer, as shown in Figure 2.

(1) Embedding layer: The input is a vector representation $V(w_t)$ of the target word w_t.

(2) Mapping layer: project the target word w_t to $V(w_t)$.

(3) Output layer: The output layer is a fully connected neural network. The *softmax* function is used to normalise the output. The correct output of the neural network should be the subscript of the context word $Context(w_t)$ of the target word w_t.

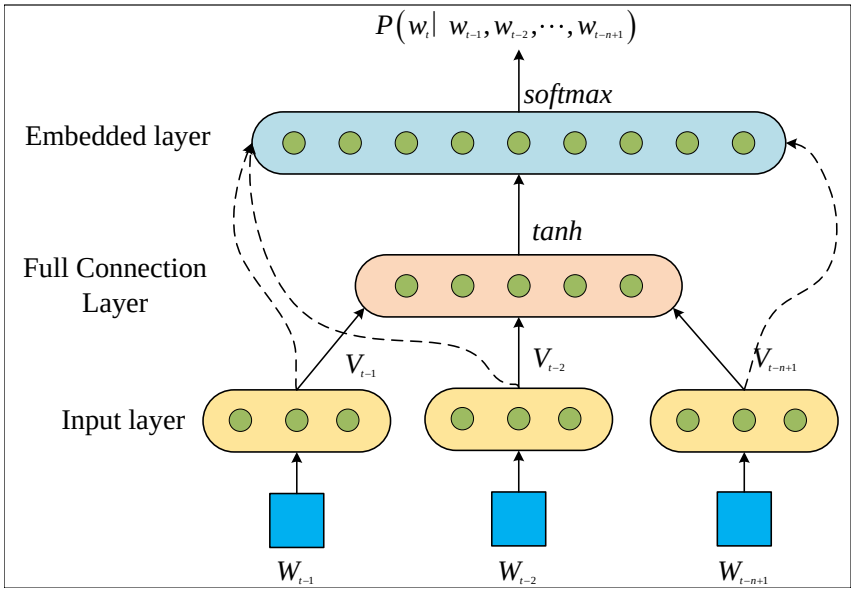

Figure 1. NNLM model

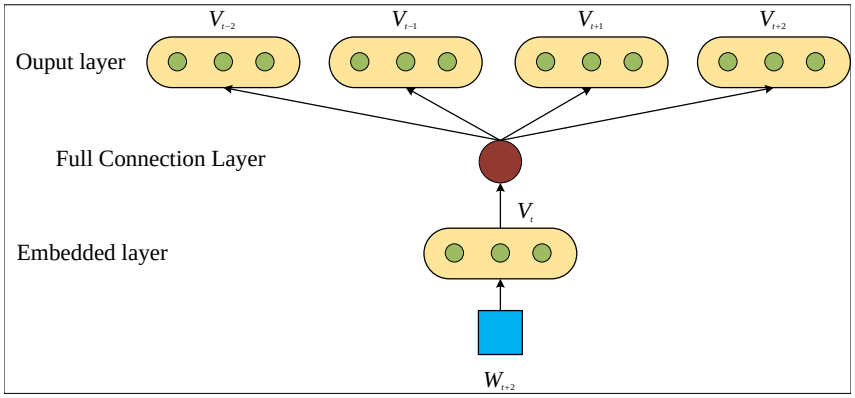

Figure 2. Skip-Gram model

The Skip-Gram model should output one word from the context $Context(w_t)$ each time a target word w_t is input. The sample $(Context(w_t), w_t)$ requires $2n$ input-output pairs $(w_t, Context(w_t))$ for all the words in the sample $(Context(w_t), w_t)$, where $i = 1, 2, \ldots, 2n$.

In fact, CBOW and Skip-Gram models face a multi-classification problem, which is usually solved by using the *softmax* function, whose time complexity is $\mathcal{O}(|V|)$, where $|V|$ is the size of the word list.

Compared to the CBOW model, the Skip-Gram model has advantages in handling rare and low-frequency words, large-scale corpora, and obtaining fine-grained word vector representations.

3. **Improved vector space modelling.**

3.1. **Information gain based TF-IDF.** The TF-IDF algorithm represents text as independent word-item vectors without taking into account the correlation between words and contextual information. This results in the TF-IDF algorithm failing to capture the information interactions between words and reducing the expressive power of the model when dealing with complex text data containing multiple words that are related to each other. Therefore, to address the problem that the TF-IDF algorithm does not consider

the inter-class distribution information of feature words, this paper introduces information gain (IG) to the TF-IDF algorithm as follows:

$$W = TF(t_k) * IDF(t_k) * IG(t_k) \tag{9}$$

In addition, the method takes the case that the feature words do not appear in the text into account. In order to attenuate the influence of low-frequency words on its calculation results, this paper removes the $P(\bar{t})$ factor from the traditional information gain calculation equation.

The improved information gain calculation method is described below:

$$IG(t) = H(C) - H(C|t) = -\sum_{i=1}^{n} P(c_i) \log P(c_i) + \lambda * P(t) \sum_{i=1}^{n} P(c_i|t) \log P(c_i|t) + \sum_{i=1}^{n} P(c_i|\bar{t}) \log P(c_i|\bar{t}) \tag{10}$$

$$\lambda = \max_{i=1}^{|c|} \frac{\sum_{j=1}^{n_i} tf_{c_i,j}}{\sum_{i=1}^{|C|} \sum_{j=1}^{n_i} tf_{c_i,j}} \tag{11}$$

where $\sum tf_{c_i,j}$ denotes the total number of word frequencies of the feature item t occurring in the category c_i, $\sum\sum tf_{c_i,j}$ denotes the total number of word frequencies of the feature item t occurring in all categories, and λ denotes the maximum word frequency ratio of the feature item t.

3.2. **TF-IDF based on intra-class discretisation.** Lack of consideration of intra-class distribution information may lead to partial loss of information. In some cases, documents within the same class may contain some key feature words, but due to the way feature weights are calculated in the TF-IDF algorithm, the importance of these feature words may be underestimated or overestimated, which affects the model's ability to characterise the text. Aiming at the problem that the TF-IDF algorithm does not consider the intra-class distribution information of feature words, this paper introduces dispersion to the TF-IDF algorithm.

The basic idea of dispersion is: in category C, if a word is evenly distributed in all the texts of the category, while the word t only sporadically occurs in the individual texts of the category, it indicates that the word is more representative of the category C than t_2, which means that it should be given a higher weight.

The key measure of the volatility of a set of data is dispersion, which is calculated from the variance of the data. The dispersion of the feature word t_k within the category c_i is calculated as shown below.

$$\overline{tf(t_k, c_i)} = \frac{1}{m} \sum_{j=1}^{m} tf(t_k, d_j) \tag{12}$$

$$D_{ii} = \frac{1}{m} \sum_{j=1}^{m} \left(tf(t_k, d_j) - \overline{tf(t_k, c_i)} \right)^2 \tag{13}$$

where m denotes the total number of texts in category c_i containing the feature word t_k; D_{ii} denotes the variance of t_k in category c_i; $\overline{tf(t_k, c_i)}$ denotes the mean of the frequency of occurrence of the feature word t_k in each text of category c_i, i.e., the ratio of the t_k is the ratio of the sum of word frequency of each text in category c_i to the total number of

texts in category c_i; $tf(t_k, d_j)$ denotes the frequency of occurrence of feature word t_k in text d_j in category c_i; and DI denotes the degree of discretisation of t_k within category c_i.

$$W = TF(t_k) * IDF(t_k) * (1 - DI(t_k)) \tag{14}$$

3.3. **IGDTF-IDF.** From the above analysis, it can be seen that the TF-IDF algorithm based on information gain can take the inter-class distribution information of feature words into account. Meanwhile, TF-IDF based on intra-class dispersion can fully consider the intra-class distribution information of feature words, enhance the correlation between features, and make the classifier accurately distinguish text data between different categories, thus affecting the classification accuracy. Therefore, in this paper, the IGDTF-IDF on information gain and intra-class discretisation is adopted by combining information gain and intra-class discretisation, and its calculation method is as follows.

$$WFW = TF(t_k) * IDF(t_k) * IG(t_k) * (1 - DI(t_k)) \tag{15}$$

The traditional TF-IDF method mainly gives vocabulary weight according to the word frequency and reverse document frequency in the document, but it does not consider the differences between different types of documents. IGDTF-IDF method combines information gain and intra-class dispersion, which can better reflect the differences between different categories of documents and improve the distinguishing ability of features. By introducing information gain, the improved TF-IDF method can more accurately evaluate the contribution of each feature to the document category, so as to select more representative and differentiated feature vocabulary. This is helpful to improve the accuracy of document classification.

4. English text similarity measure based on word embedding and IGDTF-IDF.

4.1. **Skip-Gram model based on word alignment.** The purpose of Word-Aligned is to find out the translation relationship between lexical units (usually words) in a sentence-aligned parallel corpus. Taking the English corpus as an example, Word-Aligned (WA) indexes the correspondence between each lexical item in an English text and each lexical item in another text.

In this paper, a word embedding Skip-Gram model is obtained from a monolingual Skip-Gram model, which relies on the word alignment information in the English corpus for training. Suppose $S = (s_1, s_2, \ldots, s_m)$ is a Chinese utterance of length m, where s_i is the i-th word in utterance S. $T = (t_1, t_2, \ldots, t_n)$ is an English statement of length n, where t_j is the j-th word in the statement T. If s_i and t_j are word-aligned, the upper and lower windows of the Skip-Gram model are $2n + 1$, and the number of negative samples is N. The Skip-Gram model can be used for the following purposes.

For the monolingual case, the prediction rules of the Skip-Gram model need to be designed. For the word s_i in the English utterance S, it is necessary to predict the context $Context(s_i)$ as $(s_{i-n}, \ldots, s_{i-1}, s_{i+1}, s_{i+n})$ of s_i by the target word s_i. The target word s_i contains $2n$ samples: (s_i, u), $u \in Context(s_i)$. The first item in each sample is the input value of the neural network, and the second item is the target value. To maximise the target value in each sample, the set of negative samples $NEG(u)$ of each target value is sampled, then for a sample (s_i, u) the maximised target value is.

$$g(u) = \prod_{z \in \{u\} \cup NEG(u)} \left\{ \left[\sigma\left(v(s_i)^\top \theta^z\right)\right]^{L^u(z)} \cdot \left[1 - \sigma\left(v(s_i)^\top \theta^z\right)\right]^{1 - L^u(z)} \right\} \tag{16}$$

$$L^u(z) = \begin{cases} 1, & z = u \\ 0, & z \neq u \end{cases} \tag{17}$$

where L^u is the context discriminant function.

4.2. **IGDTF-IDF based text vector representation.** After building a word embedding model based on word alignment information, we obtain linguistic word vectors. Next, how to use the word vectors obtained above for text representation will be described. In this paper, text vectorisation has been carried out using monolingual Skip-Gram word embedding model. For the obtained word embedding vectors, the proposed IGDTF-IDF is used for weighted summation, and the final text vectorised representation is achieved.

The set of lexical item vectors in the parallel corpus sets T^C and T^K after word embedding are V^C and V^K, defined as follows.

$$V^C = \left\{v(t_1^C), v(t_2^C), \ldots, v(t_m^C)\right\} \tag{18}$$

$$V^K = \left\{v(t_1^K), v(t_2^K), \ldots, v(t_n^K)\right\} \tag{19}$$

where $v(t_i^C)$ and $v(t_i^K)$ are the resultant vectors of word embeddings for the lexical item t_i^C and the lexical item t_i^K, respectively, both of dimension d.

The text vector representation d^C and the text vector representation d^K are constructed by weighted summation of word vectors using IGDTF-IDF, respectively.

$$d^C = \sum_{i=1}^{n_C} tf\_idf(t_i^C) v(t_i^C) \tag{20}$$

$$d^K = \sum_{i=1}^{n_K} tf\_idf(t_i^K) v(t_i^K) \tag{21}$$

where tf_idf is the computational function of the improved TF-IDF for a given lexical item.

4.3. **Text similarity calculation based on co-occurrence correlation.** The similarity of two texts is the degree of agreement between two text vectors, which can be described by the cosine similarity between text vectors. The cosine similarity of two texts is calculated as follows:

$$cos\_sim(d^C, d^K) = \frac{(d^C)^\top d^K}{\|d^C\| \|d^K\|} \tag{22}$$

The cosine similarity is based on the effective text vector representation, but as mentioned before, d^C and d^K are only approximate descriptions of the text, and Equation (20) only provides the similarity of the text as a vector, which is a kind of approximation of the similarity.

In this paper, in addition to the above vector similarity, the co-occurrence correlation is also proposed by using the word-item co-occurrence correlation, which is defined as follows.

$$coocc\_sim(d^C, d^K) = \frac{\sum_{i=1}^{n_C} \sum_{j=1}^{n_K} I\left[Corr(t_i^C, t_j^K) > \alpha Corr(t_i^C, t_j^K)_{\max}\right]}{n_C \cdot n_K} \tag{23}$$

where $I(\cdot)$ is the indicator function, $Corr(t_i^C, t_j^K)$ is the lexical item co-occurrence correlation, and α is the coefficient of determination of co-occurrence correlation, $0 < \alpha < 1$.

The meaning of co-occurrence relevance is the proportion of co-occurring related word-item pairs to the total word-items in different English texts. So the final improved English text similarity calculation method is obtained as follows:

$$sim(d^C, d^K) = \lambda coocc\_sim(d^C, d^K) + (1-\lambda) cos\_sim(d^C, d^K) \tag{24}$$

where $cos\_sim(d^C, d^K)$ is the cosine similarity between different English texts, given by Equation (20); $coocc\_sim(d^C, d^K)$ is the co-occurrence correlation, given by Equation (21); and λ is the fusion parameter, with the value ranging from $[0, 1]$, indicating the proportion of cosine similarity and co-occurrence correlation in the similarity calculation. When $\lambda = 0$, it means only cosine similarity is considered; when $\lambda = 1$, it means only co-occurrence correlation is considered; the actual value is determined by testing.

5. **Experimental results and analyses.**

5.1. **Experimental dataset and experimental setting.**

The dataset used in this experiment is The Sentences Involving Compositional Knowledge (SICK). The SICK dataset is a dataset for measuring the similarity of English sentences, containing text pairs as well as similarity scores for text matching and similarity measurement tasks. The SICK dataset contains sentences of various topics and sentences covering a wide range of domains such as daily life, news reports, and descriptive content. In this work, a sample set of 10 different categories was selected from the SICK dataset. From the 10 categories, 330 were randomly selected as the training set and 348 as the test set, and their data distributions are shown in Table 1.

Table 1. Distribution of experimental data

Form	**Training set**	**Test set**
C1 (Literature)	21	23
C2 (Education)	19	20
C3 (Philosophy)	33	35
C4 (Energy)	41	42
C5 (Electronics)	26	28
C6 (Communications)	26	27
C7 (Film and television)	33	34
C8 (Transport)	57	59
C9 (Law)	45	47
C10 (Public information)	29	33
Total number of texts	330	348

The experimental environment is shown in Table 2.

Table 2. Configuration of the experimental environment

Typology	**Detailed description**
Hardware Environment	CPU: E3-1230v3@3.30GHz GPU: NVIDIA Quadro K600 Running memory: 16GB
Software Environment	64-bit Windows 10 Education Edition Python 3.6

5.2. **Effectiveness of the improved VSM.** In order to validate the VSM algorithm based on the improved TF-IDF proposed in this paper, the traditional TF-IDF, STF-IDF [28], TF-POS-IDF [29], and this paper's IGDTF-IDF are used to select the feature words, and they are experimentally compared with each other through the KNN classification method. The classification effect of the whole text category will be evaluated by Macro-P, Macro-R, Macro-F1, and Accuracy, and the results are shown in Figure 3.

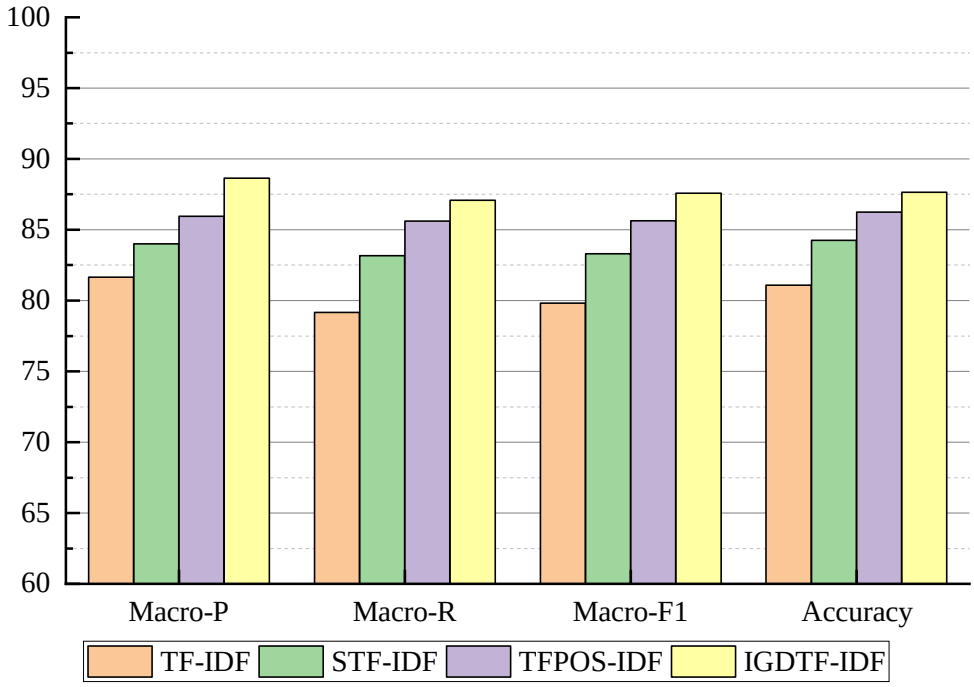

Figure 3. Number of matching feature points

It can be seen that, compared with the VSMs applying the traditional TF-IDF, STF-IDF and TFPOS-IDF, the VSMs applying the improved TF-IDF method proposed in this paper have been improved by 7%, 4.64%, and 2.7%, respectively, in Macro-P; 7.91%, 3.92%, and 1.47%, respectively, in Macro-R: 6.75%, 3.39%, and 1.4%, respectively, in Macro-F1; and in Accuracy, the improvements are 6.69%, 3.22%, and 1.2%, respectively. This is because the IGDTF-IDF method proposed in this paper fully takes into account the inter-class distribution information of feature words, intra-class distribution information, and rare feature words, and the accuracy of its feature word weights is improved to a large extent, which improves the accuracy of text classification. The experiment verifies the effectiveness of the IGDTF-IDF method proposed in this paper.

5.3. **Similarity measure test and result analysis.** In this paper, 330 pairs of English texts from the laboratory dataset are selected as the test set. In the following, the proposed English text similarity metrics will be tested separately and the effect of similarity calculation will be verified by the retrieval success rate.

For each English text, the similarity degree is calculated and ranked, and then the probability that its corresponding English text is in the Top10, Top20 and Top30 is recorded as λ grows from 0 to 1, and the test results are shown in Figure 4.

It can be seen that at the beginning the retrieval Recall of Top10, Top20 and Top30 increases with the increase of parameter λ. When $\lambda = 0.55$, the retrieval Recall of Top10, Top20 and Top30 reaches the maximum value of 0.492, 0.531 and 0.639, respectively. After the retrieval Recall reaches the peak value, the value of retrieval Recall starts to

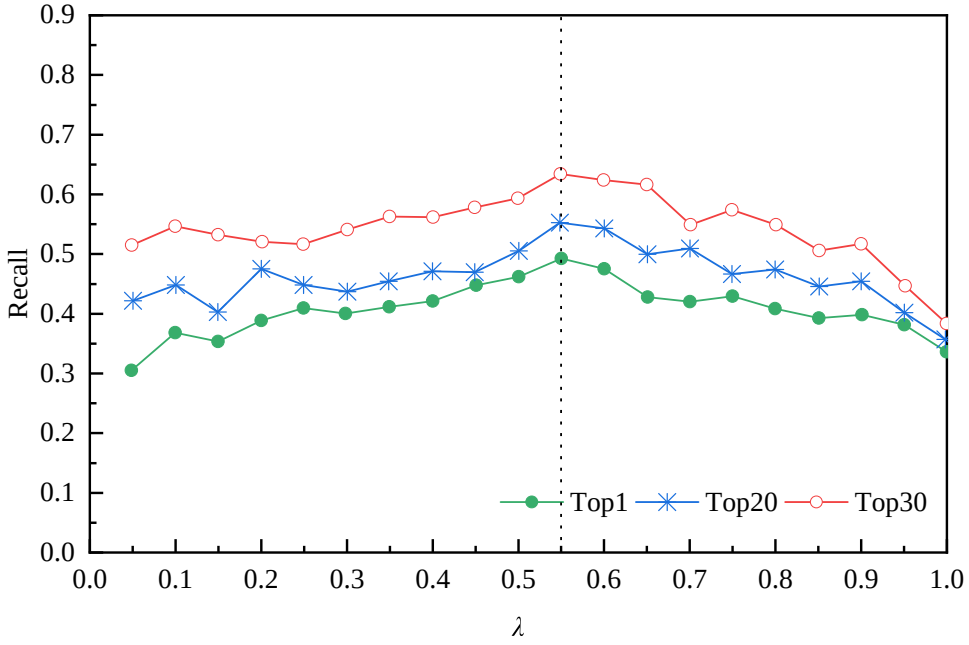

Figure 4. Retrieval recall rate

decrease with the increase of the parameter λ. When the parameter $\lambda = 1$, the retrieval Recall of Top20 and Top30 reaches the lowest. When the parameter $\lambda = 0$, the retrieval Recall of Top10 reaches the lowest. Overall, for the proposed English text similarity measure, the best parameter λ is 0.55. Therefore, the proposed method can better combine discourse vectors and discourse co-occurrences, and thus shows text commonality and text difference, and performs well on the task of English language information retrieval.

6. **Conclusion.** This work is based on word embedding and improved vector space modelling for English text similarity metrics. In the TF-IDF weight calculation method, the lower the number of occurrences of a word in a text collection or the more times a word occurs in the text, the more important the word is, without considering the influence of the distribution of the word within the class on its results, so this paper incorporates the information gain and intra-class discretisation into the TF-IDF algorithm and calls it IGDTF-IDF. Text-alignment oriented word alignment information, the Skip-Gram word embedding model is used for training and learning to obtain word vector representations of different language word items located in the same space. According to the co-occurrence correlation between English word items, the English text similarity calculation method based on IGDTF-IDF and co-occurrence correlation is proposed. And the effectiveness and practicality of the method are verified through retrieval experiments.

Acknowledgment. This work is supported by the Department of Education of Guangxi Province (No. 2022JGZ185).

REFERENCES

[1] G. Chen, X. Shi, M. Chen, and L. Zhou, "Text similarity semantic calculation based on deep reinforcement learning," *International Journal of Security and Networks*, vol. 15, no. 1, pp. 59-66, 2020.

[2] D. W. Prakoso, A. Abdi, and C. Amrit, "Short text similarity measurement methods: a review," *Soft Computing*, vol. 25, pp. 4699-4723, 2021.

[3] X. Quan, G. Liu, Z. Lu, X. Ni, and L. Wenyin, "Short text similarity based on probabilistic topics," *Knowledge and Information Systems*, vol. 25, pp. 473-491, 2010.

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

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

[6] T.-Y. Wu, J. Lin, Y. Zhang, and C.-H. Chen, "A Grid-Based Swarm Intelligence Algorithm for Privacy-Preserving Data Mining," *Applied Sciences*, vol. 9, no. 4, 774, 2019.

[7] N. Pradhan, M. Gyanchandani, and R. Wadhvani, "A Review on Text Similarity Technique used in IR and its Application," *International Journal of Computer Applications*, vol. 120, no. 9, pp. 29-34, 2015.

[8] A. Islam, and D. Inkpen, "Semantic text similarity using corpus-based word similarity and string similarity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 2, pp. 1-25, 2008.

[9] Y. Liu, and M. Chen, "Applying text similarity algorithm to analyze the triangular citation behavior of scientists," *Applied Soft Computing*, vol. 107, pp. 107362, 2021.

[10] B. Leonardo, and S. Hansun, "Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, no. 2, pp. 462-471, 2017.

[11] W. Zhou, H. Guo, and Y. Zhao, "A Novel Method for Detecting Similar Microblog Pages based on Longest Common Subsequence," *International Journal of Simulation, Systems, Science & Technology*, vol. 17, no. 8, pp. 111-118, 2016.

[12] M.-Y. Chen, H.-S. Chiang, A. K. Sangaiah, and T.-C. Hsieh, "Recurrent neural network with attention mechanism for language model," *Neural Computing and Applications*, vol. 32, pp. 7915-7923, 2020.

[13] H.-j. Kim, J. Kim, J. Kim, and P. Lim, "Towards perfect text classification with Wikipedia-based semantic Naïve Bayes learning," *Neurocomputing*, vol. 315, pp. 128-134, 2018.

[14] D. Karuppaiah, and P. D. R. Vincent, "Word sense disambiguation in Tamil using Indo-WordNet and cross-language semantic similarity," *International Journal of Intelligent Enterprise*, vol. 8, no. 1, pp. 62-73, 2021.

[15] M. Wu, T. Jiang, C. Bu, and B. Zhu, "Coarse-to-Fine Entity Alignment for Chinese Heterogeneous Encyclopedia Knowledge Base," *Future Internet*, vol. 14, no. 2, pp. 39, 2022.

[16] Y. Chen, B. Perozzi, and S. Skiena, "Vector-based similarity measurements for historical figures," *Information Systems*, vol. 64, pp. 163-174, 2017.

[17] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, "A survey on the techniques, applications, and performance of short text semantic similarity," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, pp. e5971, 2021.

[18] S. Agarwala, A. Anagawadi, and R. M. R. Guddeti, "Detecting semantic similarity of documents using natural language processing," *Procedia Computer Science*, vol. 189, pp. 128-135, 2021.

[19] P. D. Turney, and P. Pantel, "From frequency to meaning: Vector space models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, pp. 141-188, 2010.

[20] D. L. Lee, H. Chuang, and K. Seamons, "Document ranking and the vector-space model," *IEEE Software*, vol. 14, no. 2, pp. 67-75, 1997.

[21] K. Erk, "Vector space models of word meaning and phrase meaning: A survey," *Language and Linguistics Compass*, vol. 6, no. 10, pp. 635-653, 2012.

[22] S. Clark, "Vector space models of lexical meaning," *The Handbook of Contemporary Semantic Theory*, pp. 493-522, 2015.

[23] A. Hartawan, and D. Suhartono, "Using vector space model in question answering system," *Procedia Computer Science*, vol. 59, pp. 305-311, 2015.

[24] A. Manwar, H. S. Mahalle, K. Chinchkhede, and V. Chavan, "A vector space model for information retrieval: a matlab approach," *Indian Journal of Computer Science and Engineering*, vol. 3, no. 2, pp. 222-229, 2012.

[25] R. Collobert, Y. Bengio, and S. Bengio, "Scaling large learning problems with hard parallel mixtures," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 03, pp. 349-365, 2003.

[26] B. Liu, "Text sentiment analysis based on CBOW model and deep learning in big data environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 451-458, 2020.

[27] Y. Wang, L. Cui, and Y. Zhang, "Improving skip-gram embeddings using BERT," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1318-1328, 2021.

[28] A. Jalilifard, V. F. Caridá, A. F. Mansano, R. S. Cristo, and F. P. C. da Fonseca, "Semantic sensitive TF-IDF to determine word relevance in documents," in *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 2*, Springer, 2021, pp. 327-337.

[29] A. S. Alammary, "Arabic questions classification using modified TF-IDF," *IEEE Access*, vol. 9, pp. 95109-95122, 2021.