ATLANTIS PRESS

Advances in Computer Science Research, volume 86

International Symposium on Communication Engineering & Computer Science (CECS 2018)

Similarity Retrieval Algorithm based on Multilevel Fingerprint Comparison Matrix

Qiang Lv[1, a], Feihu Duan[1, a], Yanchao Wu[2, b] and Jiaxing He[2, b]

[1]CNKI, Beijing 100089, China;

[2]School of Science and Technology of Hebei University, Shijiazhuang 050000, China.

[a]595215839@qq.com, [b]lvqiangqeen@163.com

Abstract. In order to carry out similarity retrieval in mass information accurately and efficiently, this paper proposes a similarity retrieval algorithm based on multilevel fingerprint comparison matrix. For mass text information, firstly, using the Simhash algorithm to generate multilevel fingerprints; secondly, selecting the similar texts, and constructing a comparison matrix; then, the similarity between texts is accurately marked by using the comparison matrix; Finally, the real data of a company is applied to verify the accuracy and efficiency of the proposed algorithm.

Keywords: Simhash algorithm, multilevel fingerprint, comparison matrix.

1. Related Research

Zhang Guangqing and his colleagues used the Simhash fingerprint method to generate binary fingerprints for documents, and proposed an optimization method for rapid search of massive similar documents with Hamming distance expression similarity [1]. Feng Gaolei and his colleagues proposed an algorithm that integrates the calculation of semantic similarity into the text similarity algorithm based on vector space model, and finally obtains the result of text similarity by semantic similarity and vector space model similarity [2]. Qi Feilong and his colleagues used the sentence center word as the benchmark to calibrate the relative position of the words and calculate the relative position offset of the word combination, and integrated the sentence length difference information, shallow hierarchical structure information and semantic information to calculate the sentence similarity [3].According to the requirement of Chinese text duplication checking, Lee Chan-long and his colleagues transformed the target text and duplicate samples into a participle matrix model by using the result of word segmentation, and proposed a duplicate checking algorithm [4]. Based on the vector space model and the matter-based knowledge representation model, Zhao Shijie and his colleagues proposed a similarity algorithm for the knowledge representation model of science and technology projects [5].

Through similar research on similar search algorithms, similar search can be divided into three steps: (1) text preprocessing; (2) feature extraction; (3) model construction and similarity discrimination [6]. Text preprocessing refers to the processing of text in a specified format such as word segmentation, clauses, and removal of stop words. Feature extraction is to extract the feature vector that can represent the full text from the pre-processed text, usually composed of feature words and weights. Common feature extraction methods include naive Bayes classifier, decision tree classifier and TF-IDF algorithm. Model construction establishes a model describing the content of the text through the feature vector. The model represents the method of discriminating the similarity. Usually, the TF-IDF algorithm is used to extract the feature words and weights with larger weights to form the space vector, and then the vector is similar by using the cosine of the angle. Degree [7]. Due to the lack of semantic representation in SVM, the similarity is calculated by combining keyword statistics and semantic network knowledge. The semantic distance is calculated by the method based on CNKI [8].

Based on many studies, this paper proposes a similar retrieval algorithm based on multilevel fingerprint matching matrix. The algorithm generates multilevel fingerprints by Simhash algorithm, then constructs the comparison matrix model by clause clauses and adds semantic relations to calculate the similarity. The algorithm improves the accuracy based on the traditional TF-IDF space vector and adds multi-stroke parallel computing to increase efficiency.

2. Main Ideas of the Algorithm

This paper proposes a similarity retrieval method based on multilevel fingerprints. The method includes: preprocessing the text to form a unified format; encoding the unified format text using simhash algorithm to form 64-bit binary multilevel fingerprint eigenvalues; The Hamming distance between the eigenvalue of the original text and the eigenvalue of the comparison text is selected, and the text whose Hamming distance is less than the threshold 3 is selected for secondary calculation; the original text and the comparative text are segmented and the comparison matrix is constructed, and the calculation is performed. Text similarity and similar content and mark the output; optimize the text similarity and similarity content calculation method, the optimization method starts multi-threading and uses parallel computing. The algorithm studied in this paper needs three theoretical foundations to better understand: (1) Simhash algorithm and Simhash query optimization; (2) The longest common subsequence algorithm; (3) The similarity calculation rule.

2.1 Simhash Algorithm and Simhash Query Optimization.

In this algorithm, Simhash is used to convert text into 64-bit multilevel fingerprints, that is, binary 0 and 1 are used to form an n-bit signature, so that the text becomes a series of numbers. After comparing the Hamming distance to calculate the text similarity, the Hamming distance is obtained: when XOR is used, the result is 1 only when the two compared bits are different, otherwise the result is 0, and the two binary "exclusive OR" are obtained. The number of 1 is the size of the Hamming distance. For 64-bit Simhash, the Hamming distance is less than 3 to determine that the two articles are similar.

The method of calculating the Hamming distance is simple, but it is unrealistic to perform XOR one by one when the amount of data is too large. For example, for a combination of all the three bits of the 64-bit Simhash value to be queried, you need to allocate 41,664 times of storage space [9]. In order to solve this problem, 64-bit Simhash is split and stored. This method is based on the drawer principle. If the Hamming distance of two Simhash values is within 3, they are divided into m ($m \leqslant 64$) blocks, and there must be l-3 blocks are equal. In order to improve the retrieval efficiency and consider the space overhead, this paper sets m=8, and divides each Simhash value into 8 blocks, such as the first block stores 0-7 bits, the second block stores 8-15 bits, and the third block. Blocks store 16-23 bits and so on. When searching for other Simhash values whose Hamming distance is within 3 according to a Simhash, the Simhash is divided into 8 blocks, each block is searched for a similar block at the corresponding position, and the Simhash set corresponding to the similar block is taken, and at least 5 blocks are selected. The Simhash values that appear in the corresponding collection, and then calculate the Hamming distance one by one. If Simhash is evenly distributed, this method reduces the number of Hamming distance calculations to the total of 0.375.

2.2 Longest Common Subsequence Algorithm.

The longest common subsequence can describe the "similarity" between two paragraphs of text, that is, their degree of similarity, which can be used to discern plagiarism. After modifying a paragraph of text, the longest common subsequence of the text before and after the change is calculated, and the part other than the subsequence is extracted. This method is often very accurate in judging the modified part [10].

The longest common subsequence (LCS) is a problem in which a sequence set (usually two sequences) is used to find the longest subsequence of all sequences. A sequence of numbers, if they are subsequences of two or more known series, and is the longest of all sequences that meet this condition, is called the longest common subsequence of the known sequence [11].

The solution of the longest common subsequence is as follows: for example, $X=<x_1, x_2, x_3,...,x_m>$ and $Y=<y_1,y_2,y_3,...,y_n>$,the two string sums are used. The LCS solution is to record the length of the LCS with a two-dimensional array like c[i][j], and then the state transition equation can be obtained:

$$c[i,j]=\begin{cases}0 & i=0\,\|\,j=0\\ c[i-1,j-i]+1 & i,j>0\,\&\,x_i=y_i\\ \max(c[i,j-1],c[i-1,j]) & i,j>0\,\&\,x_1\neq y_i\end{cases} \tag{1}$$

Finding the longest common subsequence of the two strings X and Y can be derived recursively in the following way, if $x_m=y_n$, finding the longest common subsequence of X_{m-1} and Y_{n-1} and then adding x_m to the end of it. One of the longest common subsequences of X and Y. If $x_m\neq y_n$, the two sub-problems must be solved, namely finding one of the longest common subsequences of X_{m-1} and Y and one of the longest common subsequences of X and Y_{n-1}. The longer of the two common subsequences is one of the longest common subsequences of X and Y.

2.3 Similarity Calculation Rules.

The determination of the similarity calculation rule determines the accuracy of the final output comparison text result. This section explains the following: This article can be used to represent a piece of text D as follows:

$$D=\{d_1,d_2,d_3,...,d_n\}=\{d_k\mid k=1,2,3,...,n\} \tag{2}$$

Where d_k is an element in text D, representing a sentence segmented according to punctuation, n is the number of sentences after the clause, Len(D) represents the length of the current text, and Len(d_k) represents the sentence d_k in the text. The length of D(k) represents the kth sentence. Define the set of clauses of D as follows:

$$D(i,I)=\{d_i,d_{i+1},d_{i+2},...,d_{i+I}\}\subset D \qquad i\geq 1, i\leq n, I\leq n-i \tag{3}$$

Where I represents the length of the clause, when there are two texts D_1 and D_2, and $D_1(i,I)\subset D_1$, $D_2(i,I)\subset D_2$. $D_1(i,I)$ is found from D_1, and $D_2(i,I)$ is found from D_2, which is a collection of similar sentences in the two texts. Then for the text D_1, its similarity compared with the text D_2 is:

$$Similar(D_1\otimes D_2)=\frac{Len(D_1(i,I))}{Len(D_1)} \tag{4}$$

In the formula, Len($D_1(i,I)$) is the sum of the words in all similar sentences of the texts D1 and D2, that is:

$$Len(D_1(i,I))=\sum_{i\leq k\leq i+I}Len(d_k) \qquad i\geq 1, i\leq n, I\leq n-i \tag{5}$$

3. Algorithm Implementation

3.1 Feature Extraction and Simhash Query.

First, through the program identification, the text in Word, PDF and Html format is cleaned and the valuable text is extracted to form a unified format text. Then the text is generated by the Simhash algorithm to generate 64-bit binary eigenvalue fingerprint for quick retrieval of similar text.

The Simhash algorithm is divided into five steps [12]:

(1) Participle, segmentation is performed for a given sentence to obtain a feature vector, and five levels of weights are set for each feature vector. For example, given a sentence: the first Digital China Conference was held in Beijing. The word segmentation and weighting each feature vector: the first (4) digital China (5) conference (3) is held (1) in Beijing (4). The number in parentheses represents the importance of the word in this sentence, and the larger the number, the more important it is.

(2) Hash, the hash value of each vector, and the hash value is an n-bit signature consisting of a binary number 01. For example, the first hash value is 100101, and the digital China hash value is 101011. In this way, the string becomes a series of numbers.

(3) Weighting, weighting all feature values based on the hash value, that is w=hash*weight, encountering 1 multiplied by a positive weight value, and encountering 0 times the negative weight value. For example, the weight is given to the first is w_1=100101*4=4-4-4+4-4+4, the weight is given

to the digital China is w_2=101011*5=5-5+5-5+5+5, and the other feature vector weighting methods are the same.

(4) Combining, the above-mentioned feature vector weighting result becomes a sequence, for example, the weighted result of the first two words is 9-9+1-1+1+9.

(5) Dimensionality reduction, if the result of the accumulation is greater than 0, it is recorded as 1. Else if it is less than 0, it is recorded as 0. For example, the result of the reduction is 101011. The hash value of this example is 6 bits. In the Simhash calculation, each feature vector is formed with a 64-bit hash value to make the final result more accurate.

The Simhash algorithm specifies that Hamming distance is less than or equal to 3, and is judged to be similar. To make the query more efficient, divide the 64-bit hash value into 8 blocks. To query the sql statement each time, only at least 5 parts of the same hash value need to be queried, and Select one of the different 1 to 3 blocks to calculate the Hamming distance.

The similar text is found by querying the Simhash value whose Hamming distance is less than 3, and the comparison matrix is constructed by the original text and the similar text.

3.2 Construction of the Comparison Matrix Model .

The construction of the comparison matrix model is the core of the algorithm. The flow chart of the similarity retrieval algorithm based on the multilevel fingerprint comparison matrix is as follows:

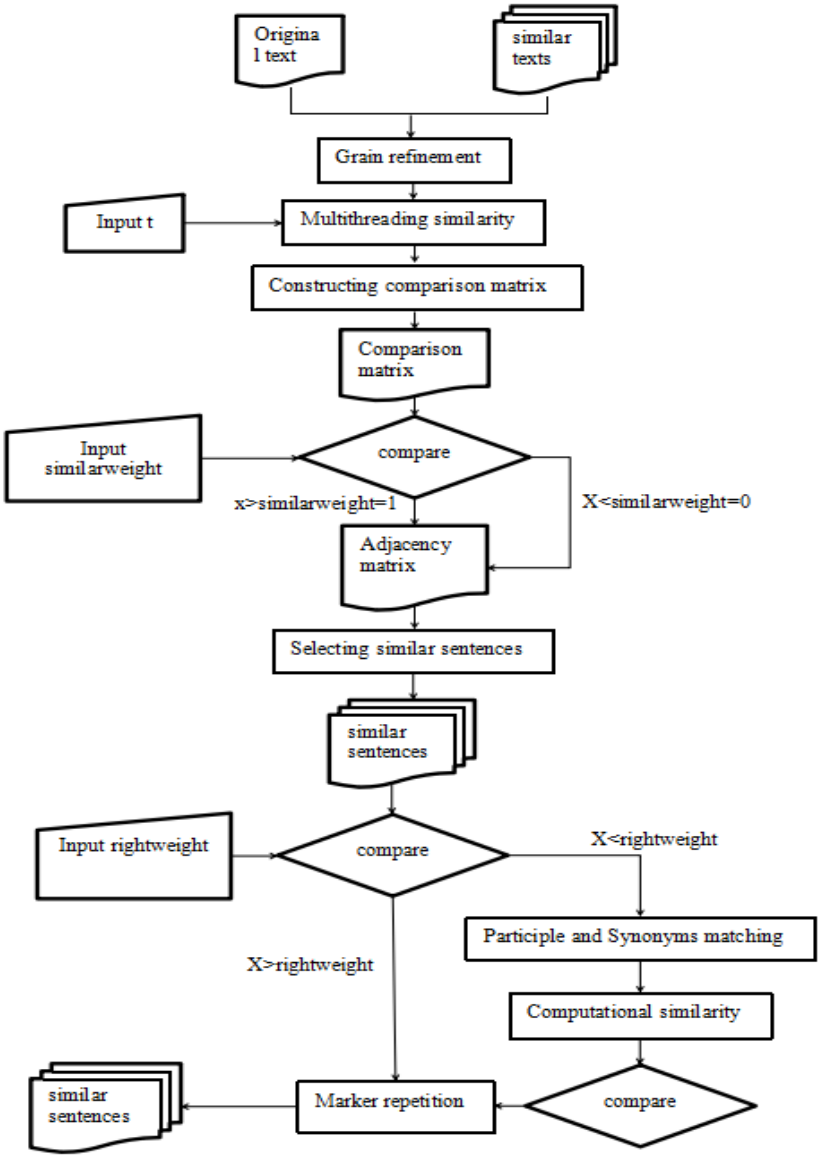

Fig.1 Flow chart of similarity retrieval method based on multilevel fingerprint

(1) Text granularity

The two texts are constructed to match the matrix. The text granularity is first refined, and the text is refined into statement granularity by punctuation. The comparison text and the compared text be: $D_1 = \{d_1, d_2, d_3, ..., d_n\} = \{d_k \mid k = 1,2,...,n\}$ and $D_2 = \{d'_1, d'_2, d'_3, ..., d'_m\} = \{d'_k \mid k = 1,2,...,m\}$

(2) Building a comparison matrix

D_1 and D_2 construct the comparison matrix P as:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1m} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2m} \\ p_{31} & p_{32} & p_{33} & \cdots & p_{3m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & p_{n3} & \cdots & p_{nm} \end{bmatrix} \tag{6}$$

p_{nm} calculates the similarity between the original text d_n and the comparison text d'_m. The formula is:

$$p_{nm} = \min\left(\frac{LCS(d_n, d'_m)}{Num(d_n)}, \frac{LCS(d_n, d'_m)}{Num(d'_m)}\right) \tag{7}$$

LCS (d_n,d'_m) is the number calculated using the longest common subsequence algorithm in d_n and d'_m. Num(d_n) is the number of words in statement d_n, Num(d'_m) is the number of words in statement d'_m, this number can be used to account for the ratio of the original sentence to the comparison statement. And taking the smaller ratio as the final result of p_{nm}.

(3) Setting the threshold similarWeight and constructing an adjacency matrix

The threshold value ranges from 0 to 1, and can be adjusted manually to cope with different requirements of different items for accuracy. All values in the comparison matrix P are compared with a threshold similarWeight, and greater than the threshold is set to 1, less than the threshold. Set to 0 to construct an adjacency matrix Q of n rows and m columns, for example:

$$Q = \begin{bmatrix} 0 & 1 & 0 & ... & 1 \\ 0 & 0 & 1 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 1 & 0 & 0 & ... & 1 \end{bmatrix} \tag{8}$$

(4) Finding similarities through adjacency matrix

Setting a threshold rightWeight, the threshold rightWeight is between 0-1 and greater than similarWeight. Checking the p_{nm} value of the adjacency matrix marker. Parts that are greater than the threshold are directly labeled as similar. and recording the position of similar sentences in two pairs of texts.

For parts smaller than the threshold rightWeight, finding the corresponding sentences through the position of the row and column and performing the word segmentation, the adding the thesaurus to perform semantic judgment. The judgment formula is as follows:

$$w_{nm} = \min\left(\frac{rightWord(d_n, d'_m) + similarWord_{d_n}(d_n, d'_m)}{Num(d_n)}, \frac{rightWord(d_n, d_m) + similarWord_{d'_m}(d_n, d'_m)}{Num(d'_m)}\right) \tag{9}$$

rightWord(d_n,d'_m) indicates the number of words in the same word after word segmentation. similarWord$_{dn}$(d_n,d'_m) indicates the number of similar words in the sentence d_n of the original text D_1. w_{nm} denotes the smaller proportion of similar text in the original text statement and comparing text statement after segmentation and adding the similar word library. When w_{nm} is greater than or equal to threshold rightWeight, it is labeled as similar. Counting all the labeled similar statements when the computation is completed, and using formula similar($D_1 \otimes D_2$) to find out the similarity of two texts.

(5) Efficiency optimization of comparison matrix model algorithm

When constructing the alignment matrix, we need to calculate the similarity of all the statements in the articles, that is, we need to calculate n*m times, and it will increase the operation time when

the length of the articles increases. However, when the thread is opened too much, the thread scheduling time will be too long to achieve the desired effect. After a large number of text tests, text comparison in the total number of words in each interval, selecting the optimal number of threads in each interval. Because of the different number of threads, it is necessary to segment the alignment matrix so that the original alignment matrix can be converted into a relatively small alignment matrix for parallel computation.

The conversion method is as follows:

The algorithm achieves the effect of limiting the number of threads by limiting the number of single-order comparison statements. If the number of single-order comparison statements is t, the calculation formula for the number of open threads v is:

$$v = Ceil(\frac{n}{t}) * Ceil(\frac{m}{t}) \tag{10}$$

The Ceil function is the integral function. As long as there are decimal places, add 1.

Comparison matrix segmentation method is:

$$P' = \begin{bmatrix} \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1t} \\ p_{21} & p_{22} & \cdots & p_{2t} \\ \cdots & \cdots & \cdots & \cdots \\ p_{t1} & p_{t2} & \cdots & p_{tt} \end{bmatrix} & \cdots & \begin{bmatrix} p_{1(t^* \ Ceil(\frac{m}{t})-1+1)} & p_{1(t^* \ Ceil(\frac{m}{t})-1+2)} & \cdots & p_{1m} \\ p_{2(t^* \ Ceil(\frac{m}{t})-1+1)} & p_{2(t^* \ Ceil(\frac{m}{t})-1+2)} & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{t(t^* \ Ceil(\frac{m}{t})-1+1)} & p_{t(t^* \ Ceil(\frac{m}{t})-1+2)} & \cdots & p_{tm} \end{bmatrix} \\ \cdots & \cdots & \cdots \\ \begin{bmatrix} p_{(t^* \ Ceil(\frac{n}{t})-1+1)1} & \cdots & p_{(t^* \ Ceil(\frac{n}{t})-1+1)t} \\ \cdots & \cdots & \cdots \\ p_{n1} & \cdots & p_{nt} \end{bmatrix} & \cdots & \begin{bmatrix} p_{(t^* \ Ceil(\frac{n}{t})-1+1)(t^*Ceil(\frac{m}{t})-1+1)} & \cdots & p_{(t^* \ Ceil(\frac{n}{t})-1+1)m} \\ \cdots & \cdots & \cdots \\ p_{n(t^*Ceil(\frac{m}{t})-1+1)} & \cdots & p_{nm} \end{bmatrix} \end{bmatrix} \tag{11}$$

In order to make full use of the space and time resources and improve the efficiency, all matrices in p' are added to the thread parallel computing at the same time.

4. Experimental Results and Analysis

Ten texts ranging from 500 words to 10,000 words were selected for the experiment. The experiment mainly focused on the accuracy and efficiency of the algorithm. In this paper, data pretreatment and feature extraction are carried out in 10 texts. Then generating 64-bit multilevel fingerprints. Data pre-processing results are shown in Figure 2. Generating Simhash fingerprints as shown in Figure 3.

f_article_title	f_article_content
As rainy season starts, UN health agency warns of cholera outbreak in dr	5 May 2017 &#150 Somalia is suffering from the largest
On Global Road Safety Week, UN health agency urges &#39Save Lives: S	5 May 2017 &#150 Spotlighting the relationship betwee
UN-backed network brings together thousands to help tackle mosquito-l	10 May 2017 &#150 In an effort to escalate the global f
China: Guangdong reported a H9N2 case during surveillance	Guangdong reported a human H9N2 case during serveillance on 25th, August 2016.
UN informed of Ebola virus outbreak in northern DR Congo	12 May 2017 &#150 An Ebola outbreak in north-easterr
UN health agency deploying technical experts to site of Ebola outbreak ir	15 May 2017 &#150 The United Nations health agency i
Deaths of 1.2M adolescents mostly preventable with improved health ser	16 May 2017 &#150 The deaths of more than 1.2 millior
UN aids Yemenis in embattled Mokha; warns of cholera' s spread amid	16 May 2017 &#150 The United Nations refugee agency
With barely enough money for food, conflict-displaced Ukrainians strugg	16 May 2017 &#150 Two-thirds of internally displaced U
Reporting of global death statistics improving, UN agency reports	17 May 2017 &#150 An increasing amount of deaths ha

Fig. 2 Data pre-processing results

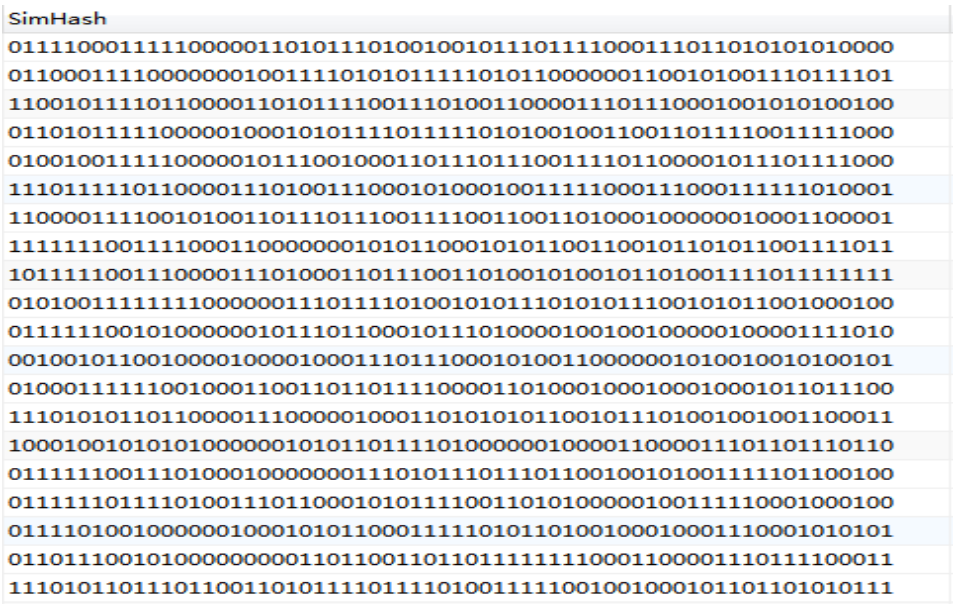

Fig. 3 64-bit Simhash multilevel fingerprints

The Simhash values of 10 texts is put into massive data to retrieve and find out 3 texts whose Hamming distance is less than or equal to 3.

4.1 Algorithm Accuracy Experiment.

The number of comparison statements is 10. The threshold similarweight is 0.6 and the threshold rightWeight is 0.8. The experimental texts are compared with the real similar texts. The results are shown in Table 1.

Table 1. Statistical results of similar texts

Test text volume	Experimental similar text	True similar amount of text
561	78	78
1129	112	128
2358	175	175
3136	286	309
4623	271	271
5716	241	264
7086	407	438
7897	329	329
8865	498	538
10849	594	659

According to the statistical results of similar text, we can see that the experimental similar text is basically close to the real similar text, with the increase of similar text detection may cause errors. According to the actual observation, under the condition of controlling the number of comparison sentences and the threshold value, the deviation of text similarity detection mainly depends on the perfection of the synonym lexicon. The more perfect the synonyms, the smaller the error of similarity detection.

4.2 Algorithm Running Efficiency Experiment.

In this paper, by controlling the number of text words and the comparison threshold, the number of comparison statements t (that is, changing the number of threads) is changed to compare the operational efficiency horizontally; By controlling the comparison threshold and the number of statements t, the number of words is changed to compare the operational efficiency longitudinally. The comparison results are shown in the following figure:

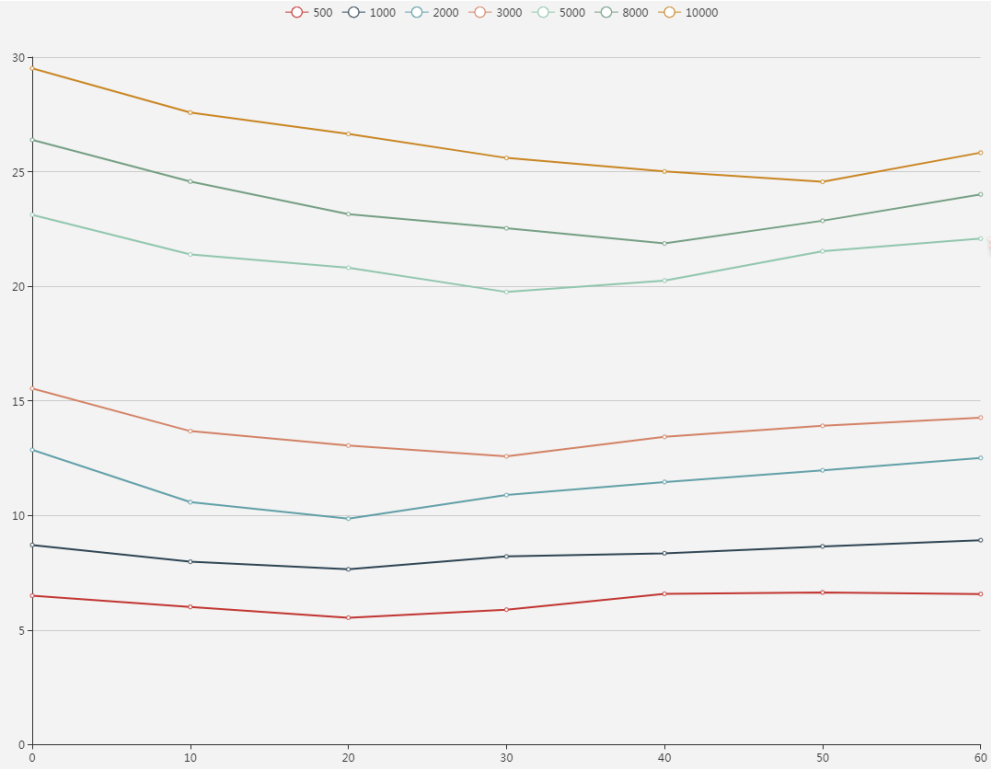

Fig. 4 Similarity comparison results

In Figure 4, abscissa denotes the number of threads opened, Left-hand label indicates the time (in seconds) of similarity comparison, and broken lines indicate the result of similarity comparison between different text quantities. From the results of similarity comparison, it can be seen that the time used for comparison and the number of threads opened are basically parabolic, so the best thread interval (that is, the shortest thread interval) for similarity comparison of current text can be determined according to the number of words in the text. When controlling other variables to change the number of words, the time of similarity comparison is proportional to the number of words in the text.

5. Conclusion

In this paper, a similarity retrieval algorithm based on multilevel fingerprint comparison matrix is proposed. Firstly, the text eigenvalues are extracted and the similar text is retrieved by Simhash algorithm. Then the comparison matrix is constructed to compare the similar texts and the final result is obtained. The algorithm is applied to the actual similarity detection. The accuracy and efficiency of similarity comparison are verified by experiments: (1) The accuracy of similarity retrieval depends on the determination of threshold and the perfection of similarity lexicon. (2) The efficiency of the algorithm is related to the number of rows and columns of the comparison matrix (i.e. the number of threads) and the number of words in the comparison text. When determining the range of text, the optimal number of threads can be derived to improve the efficiency of similarity comparison.

References

[1]. Zhang Guangqing, Ge Weiyi, He Chenglong. Fast Search and Optimization Method for Massive Similar Documents Based on Simhash[J]. Command Information System and Technology. Vol. 6(2015) No. 2, p. 61-65.

[2]. Feng Gaolei, Gao Yufeng. Text similarity algorithm based on vector space model combined with semantics[J]. Modern Electronic Technology. Vol. 41(2011) No. 11, p. 157-161.

[3]. Fei Feilong, Yan Huasong.Sentence similarity algorithm based on modified offset[J]. Computer Engineering. Vol. 43(2017) No. 9, p. 234-239.

[4]. Li Chenglong, Yang Dongju, Han Yanbo. Research on fuzzy matching check algorithm based on word segmentation matrix model[J]. Computer Science. Vol. 44(2017) No. 11A, p. 55-83.

[5]. Zhao Shijie, Xu Xiaoliang. Research on similarity calculation and clustering algorithm for science and technology projects [D]. Hangzhou: Hangzhou University of Electronic Science and Technology, 2015.

[6]. Li Shanqing, Xing Xiaozhao, Du Shengmei. Review of research on check methods of science and technology projects[J]. Science and Technology Management Research. Vol. 6(2018) No. 2, p. 197-201.

[7]. Fang Yanfeng. Improvement of the calculation method of TF-IDF value of characteristic words in scientific and technological project check[J]. Information Research. Vol.1(2012) No. 1, p.1 - 3.

[8]. Liu Qun, Li Sujian. Calculation of lexical semantic similarity based on HowNet[J]. Chinese Computational Linguistics. Vol.7(2002) No. 2, p. 59-76.

[9]. Chongshan, Shao Chunxia. Application of Simhash algorithm in checking the test questions[J]. Software Guide. Vol. 17(2018) No. 2, p. 151-157.

[10]. LIU Yu, WANG Qiandong. Judging of Non-synchronous Similar Trajectories Based on the Longest Common Subsequence[J]. Telecommunication Technology. Vol. 57(2017) No. 10, p. 1165-1170.

[11]. ZHENG Cuiling. Analysis and Implementation of the Longest Common Subsequence Algorithm[J]. Journal of Wuyi University. Vol. 29(2010) No. 2, p. 44-48.

[12]. Xu Jihui. Research on anti-cheat technology of massive documents based on Simhash algorithm[J]. Computer Technology and Development. Vol. 24(2014) No. 9, p. 103-107.