Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 9 (2): 337-349, 2015
ISSN 1819-4311 / DOI: 10.3923/jse.2015.337.349

A Component-Relation-Map Detection Algorithm for Text Similarity

Huajun Huang, Lili Xie and Jiaohua Qin
College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, 410004, China

Corresponding Author: Huajun Huang, College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, 410004, China

ABSTRACT

The conventional text similarity detection usually use word frequency vectors to represent texts. But it is high-dimensional and sparse. So in this research, a new text similarity detection algorithm using component relation map (CRM-TSD) is proposed. This method is based on the mathematical expression of Chinese characters with which Chinese characters can be split into components. Then each component's occurrence number will be counted for building the Component Histogram Map (CHM) in a text. All the relationship of correlated components which are split from a Chinese character will be depicted by the Component Relation Map (CRM). At last, the text similarity will be obtained through matching CRMs of two components and texts, respectively. The experiment results indicate that CRM-TSD achieves a good precision, recall and F1. It performs better than cosine theorem and Jaccard coefficient.

Key words: Similarity detection, Chinese text, component histogram map, component relation map

INTRODUCTION

As a branch of natural language processing, text similarity detection is more and more important for information security. It has been used in many fields such as Information Retrieval (IR), duplicated detection (Bao *et al*., 2003), data clustering and classification. In general, there are two ways for text similarity detection, one is that based on semantic similarity and the other one is non-semantic. Semantic similarity detection usually based on dictionary computation like HowNet (Zhang, 2013) and WordNet (Chen *et al*., 2011). Huang has ever proposed a method that combined the external dictionary with TF-IDF to compute text similarity (Huang *et al*., 2011). Some people also use a large-scale corpus for semantic similarity detection (Shi *et al*., 2013) but it's uncommon because of its disadvantages. Non-semantic similarity detection mostly use word frequency statistics and string comparison two methods. The most common used methods of word frequency statistics are VSM (Wang *et al*., 2011; Wang and Wu, 2011), the text similarity can be computed through cosine (Zhang *et al*., 2012) theorem or Jaccard coefficient (Thada and Joshi, 2011). In the other hand, Shingling (Zhao *et al*., 2011) and maximum string matching algorithm (Peng *et al*., 2010) is often used for string comparison. All of the methods above performance well in certain situations but there are also some shortcomings. For examples, the semantic method based on dictionary is too depending on person and the knowledge library to express the sense of a word exactly. Word frequency statistics is very high-dimensional and sparse (Sun *et al*., 2013).

From the above, a new Chinese text similarity detection method was proposed. This method used (CRM) Component Relation Map to avoid high-dimensional and sparse problem. Mathematical

expression of Chinese characters (Sun *et al.*, 2002), used to split Chinese characters into components was the basic theorem for this method. The components were taken as research object. Components are correlated with each other to compose Chinese characters, so these components are correlative. CRM was built with these correlated components. Then matching CRM of each text would get the text similarity. From the results, we can see that CRM-TSD performance well than cosine theorem and Jaccard coefficient.

RELATED THEORIES

In the process of text similarity detection, text feature representation and similarity detection are two very important steps (Wang *et al.*, 2011). VSM is the most common method for text feature representation. Assuring d_i is the i-th text, W_{ij} is the weight of the j-th word of d_i, then the i-th text can be represented as $\vec{d_i} = (W_{i,1}, W_{i,2}, .., W_{i,n})$ so all the texts in the experiment can compose a vector space $D = (\vec{d_1}, \vec{d_2}, ..., \vec{d_3})$. The similarity of each pair of text can be computed as two vectors' distance through cosine theorem. The equation is as follows:

$$\text{sim}\left(d_i, d_j\right) = \cos(\theta) = \frac{\vec{d_i} \bullet \vec{d_j}}{\left\|\vec{d_i}\right\| \times \left\|\vec{d_j}\right\|} = \frac{\sum_{t=1}^{n} W_{i,t} \times W_{j,t}}{\sqrt{\sum_{t=1}^{n} W^2_{i,t}} \times \sqrt{\sum_{t=1}^{n} W^2_{j,t}}} \quad (1)$$

where, $\left\|\vec{d_i}\right\|$ is norm of d_i. The value of cosine similarity between two vectors is between 0 and 1, 0 indicates the two texts are different and 1 indicates they are the same.

In the Chinese character library, there are 6763 common Chinese characters which encoded with Gb-2312. All these Chinese characters can be combined to thousands of words and even more. For example, the word segmentation software of Chinese Academy of Science (ICTCLAS), has extracted 130,000 commonly used words from the corpus provided by Sogo lab (Sun *et al.*, 2013). So it is obvious that the number of Chinese word in a corpus needs to be counted is quite big. This leads to high-dimensional and sparse vectors space. Therefore, a new text representation method based on component relation map has been proposed.

A mathematical expression of Chinese characters of a Chinese character is a formula for splitting Chinese characters into components. It composes of operators and components. Component is a part of a Chinese character and composes of strokes. Every component has a corresponding number as its identifier. There are two kinds of components, one is the ordinary components and the other called composed components consist of two or more components by certain structural rules. As shown in Fig. 1 Chinese characters are formed with the components by different structural rules (Sun *et al.*, 2002).

There are six operators of the mathematical expression of Chinese characters, lr(left right), ud(up down), we(whole embody), lu(left up), ld(left down), ru(right up). All these operators represent the combination mode of components. As shown in Fig. 1, the rectangle A and B are components (Zhang *et al.*, 2012). A lr B means that A is on the left and B is on the right. It has two results, a composed component and a Chinese character.

As mentioned above, Chinese characters are compose of components and correlated rules. In this research, we have selected 505 components which can form all the common used Chinese characters as research objects. As Chinese characters increasing, the number of each component

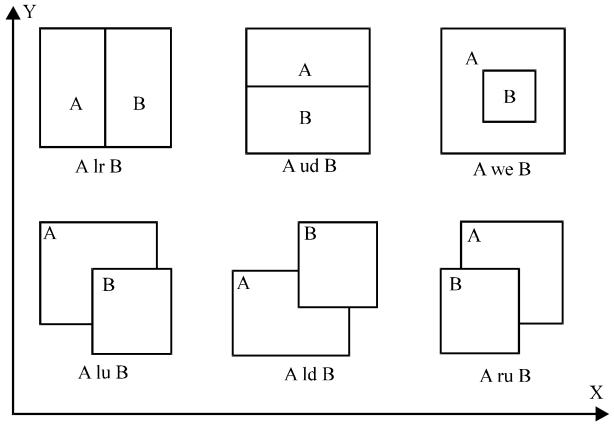

Fig. 1: Intuitive description of the operators

Table 1: Mathematical expression of chinese characters

Chinese characters	Mathematical expression
才	30
材	86 lr 30
柴	(99 lr 21) ud 86
闭	39 we 30
遍	500 ld (73 lu 375)
斌	68 r (336 ru 99)

will increase clearly but the number of kinds of components won't. Table 1 gives some examples of mathematical expression of Chinese characters.

SIMILARITY DETECTION MODEL

Similarity detection model divides into three modules: (1) Build the component histogram map, (2) Generate the component relation map and (3) Compute the text similarity. The framework of this model is as shown in Fig. 2. When two texts are prepared, the number, English characters and the stop and useless words are deleted first. So there are only Chinese characters retained. After the preprocessing, all Chinese characters in texts are split into components through the mathematical expression of Chinese characters. Then the occurrence number of each component will be counted for building the histogram maps. The component relation maps are constructed by the relationship of correlated components. At last, all the component relation maps of each pair of texts are matched to get the text similarity. The core module of this model is text similarity detection using component relation map.

Assuring that c is a component and T is a text, then c_i is the ith component and the text T can be regarded as a set of components, $T = \{c_1, c_2, c_3, \ldots, c_i, \ldots, c_n\}$:

- **Definition 1:** Component histogram map is defined as $CHM = \{Nc_1, Nc_2, Nc_3, \ldots\ldots, Nc_n\}$, where Nc_i is the number of the ith component in text T. Figure 3 is a CHM of the text extracted from experiment data corpus
- **Definition 2:** Component relation map is defined as CRM = <V, E>, where $V=\{c_1, c_2, c_3, \ldots, c_i, \ldots, c_n\}$ is the vertex set consist of all the components of text T, $E = \{\ldots, (c_i, c_j), \ldots\}$ $(1 \le i, j \le n)$ is an undirected edge set, where (c_i, c_j) means components c_i and c_j are correlated. They are called correlated components pair. The weight of (c_i, c_j) is the correlated times of them

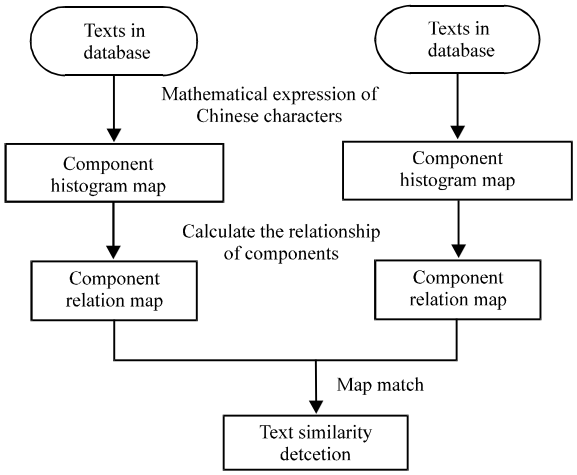

Fig. 2: Similarity detection model

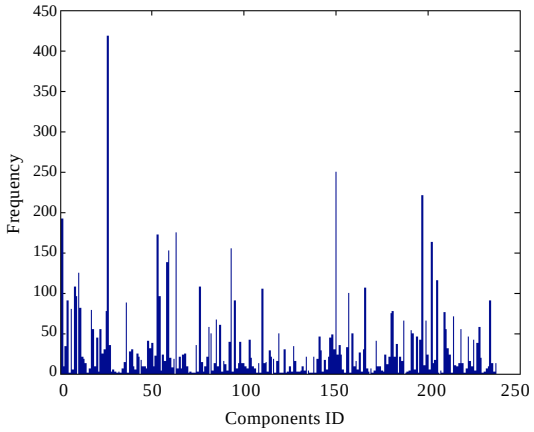

Fig. 3: Component histogram map

Fig. 4a-b are CRMs generated by the sentence of "文本相似度检测是度量两个或多个文本相似程度的主要途径". Figure 4a presents a CRM of the 86th component, it gives all the relationship of components that correlated with the 86th component to form a Chinese character in this sentence. Figure 4b is the CRM of the sentence, it represents all the relationship of all components in this sentence.

A component correlated with other different components in different mode can form different Chinese characters. Therefore, only considering the number of components is not precise enough to represent the content of a text. CRM which depicts the relationship of components decomposed in a text, can express the content precisely. The algorithm is as follow, Table 2 gives the notations and their meaning that used in follow context.

The similarity of CRM of a component occurs in TA and TB meanwhile could be computed through the correlated components pairs and corresponding weights. The specific computational method is shown as Eq. 2.

$$\mathrm{sim}(C_i, C_i') = \sum_{k=1}^{n} \frac{\min(W_{ik}, W_{ik}')}{\sum W_i} \times 1\{C_k = C_k'\} \quad (2)$$

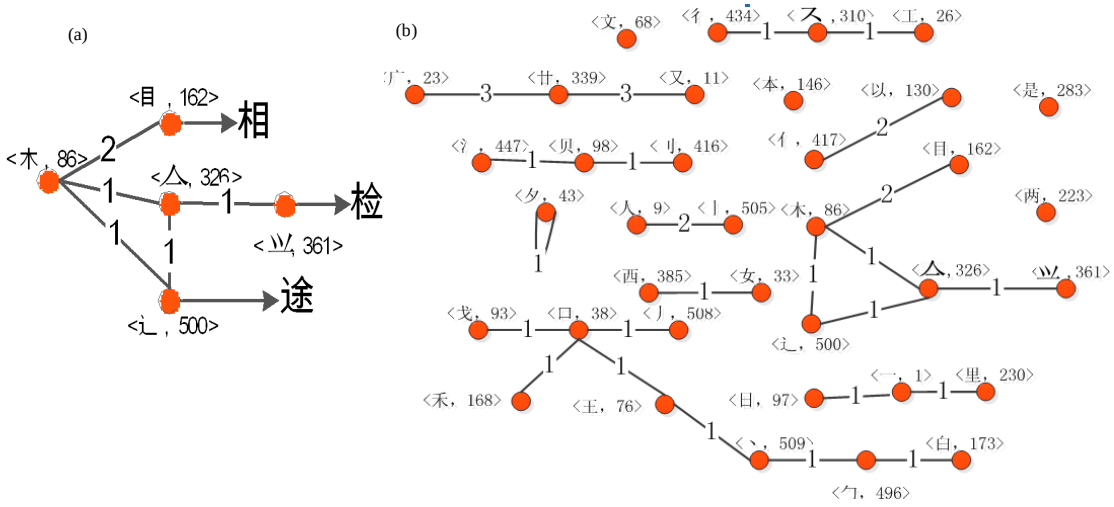

Fig. 4(a-b): CRM of the (a) 86th component and (b) sentence

Table 2: Notations and their meaning

Notations	Meaning
TA	Text in database
TB	Text to be detect
Ci	Component i in TA
Ci'	Component i in TB
Wik	Times of component i correlated with k in TA
Wik'	Times of component i correlated with k in TB
CRM (TA)	CRM of TA
CRM(TB)	CRM of TB
DCi	Sum of times of all components correlated with component i in TA, called degree
DCi'	Sum of times of all components correlated with component i in TB, called degree
FCi'	Occurrence number of the ith component in TB
Sim(Ci,Ci')	Similarity of CRM of the ith component occur in TA and TB meanwhile
Sim(C,C')	Sum of similarity of all CRM of a component in TA and TB
Sim(CRM(TA),CRM(TB))	Similarity of CRM of all component occur in TA and TB
Sim(TA, TB)	Similarity of TA and TB

where, n is the number of components in TB, ΣW_i equals to D_{Ci} and $\times 1(C_k\text{-}C_k')$ is 1 or 0 when the kth component occurs in TA and TB simultaneously or asynchronously.

The frequency of each component in a text can be get from CHM. So the similarity of the CRM of single component of a text can be got by synthesizing this frequency and similarity of CRM of each component. The specific computational method is shown as Eq. 3:

$$\mathrm{sim}(C,C') = \sum_{i=1}^{n} F_{C_i}' \times \mathrm{sim}(C_i, C_i') \tag{3}$$

The similarity of the CRMs of texts can be get by computing the ratio of the intersection and union set of each CRM, Eq. 4 shows the specific computational method:

$$\begin{aligned}&\mathrm{sim}\left(\mathrm{CRM}(\mathrm{TA}),\mathrm{CRM}(\mathrm{TB})\right)\\&=\frac{\left|\mathrm{CRM}(\mathrm{TA})\cap\mathrm{CRM}(\mathrm{TB})\right|}{\left|\mathrm{CRM}(\mathrm{TA})\cup\mathrm{CRM}(\mathrm{TB})\right|}\end{aligned}\tag{4}$$

where, $\left|\mathrm{CRM}(\mathrm{TA})\cap\mathrm{CRM}(\mathrm{TB})\right|$ and $\left|\mathrm{CRM}(\mathrm{TA})\cup\mathrm{CRM}(\mathrm{TB})\right|$ are the size of the set.

The intersection refers the same components that occurred in CRMs of TA and TB meanwhile and the small degree of these components. As Eq. 5 shows:

$$\mathrm{CRM(TA)}\cap\mathrm{CRM(TB)}=\sum\min(D_{c_i},D_{c_i'})\times i\times 1\{C_i=C_i'\}\tag{5}$$

The union set consists of all the components in CRM of TA and TB. When components occurred in CRM of TA and TB meanwhile, the large degree of them is selected to compute the similarity, otherwise, the degree of each component is selected directly. Equation 6 shows the specific computational method:

$$\mathrm{CRM(TA)}\cup\mathrm{CRM(TB)}=\sum\max(D_{c_i},D_{c_i'})\times i\times 1\{C_i=C_i'\}+\sum D_{C_j}\times j+\sum D_{C_k'}\times k\tag{6}$$

Text similarity has composited all these CRMs' similarity, the specific computational method is shown as Eq. 7:

$$\mathrm{sim}(\mathrm{TA},\mathrm{TB})=a\times\mathrm{sim}(C,C')+(1-a)\times\mathrm{sim}\left(\mathrm{CRM}(\mathrm{TA}),\mathrm{CRM}(\mathrm{TB})\right)\tag{7}$$

where, α indicates the proportion of the similarity of CRM of single component. It is a system parameter and will be ensured through experiment.

METHODOLOGY

Algorithm description and analysis

Algorithm description: The process of CRM-TSD can divided into three steps:

Step 1: Computing the similarity of CRM of single component through Eq. 1
Step 2: Computing the sum of similarity of CRM of all single components through Eq. 2
Step 3: Computing similarity of all CRMs to get the similarity of texts through the Eq. 4-6

We used an integrate array crm[ ] [ ] to present CRM component relation map, where the first dimension means all the components occurs in a text and the second dimension is all the components that correlated with the components in the first dimension. The value of each item in array is the weight of a correlated components pair. According to these, the pseudocode of the algorithm is as follows:

```
/*compute the similarity of CRM of each component, crmA [ ] [ ] is the CRM of TA and crmB [ ] [ ] is the CRM of TB, index is the number of the first dimension, ca is the number of component in TA, cb is the number of component in TB.*/
double Sim1(int crmA [ ] [ ], int crmB[ ] [ ], int index, int ca, int cb)
ci= GetRowValue(crmA, ca, index);
```

```
if(ci==0) then return 0;
for ( i = 0; i <= cb; i++ ) do
if(crmA[index,i]>0&&crmB[index, i]>0) then
if(crmA[index,i]<crmB[index,i]) then sum_ci+= crmA[index, i] / ci;
else
sum_ci += crmB[index,i] / ci;
        end if
        end if
       end for
for(i=0;i<=cb;i++) do
Sum_B += GetRowValue(crmB, cb, i);
end for
for (i = 0; i <= min(ca, cb); i++) do
Fi = GetRowValue(crmB, cb, i);
Fi = Fi / Sum_B;
sum += Fi * sum_ci;
       end for
return sum;
end Sim1
//compute the similarity of CRM of a text
double Sim2(int crmA [ ] [ ], int crmB[ ] [ ], int ca, int cb)
for (i = 0; i <= max(ca,cb); i++) do
rowA = GetRowValue(crmA, ca, i);
rowB = GetRowValue(crmB, cb, i);
if ( rowA==0 && rowB==0 ) then continue;
end if
v_max = rowA > rowB ? rowA : rowB;
v_min = rowA < rowB ? rowA : rowB;
if ( rowA > 0 && rowB > 0 ) then
sum_u += v_max * i;
sum_n += v_min * i;
        end if
if(rowA >0 && rowB <=0) then
sum_u += rowA * i ;
end if
if(rowA <= 0 && rowB > 0) then sum_u += rowB * I;
sum = sum_n / sum_u;
end if
return sum;
end Sim2
//compute the similarity of texts
double Sim_Text(int crmA [ ] [ ], int crmB [ ] [ ], int ca, int cb)
sum = a * Sim1(crmA, crmB, ca, cb) + (1 - a) * Sim2(crmA, crmB, ca, cb);
return sum;
end Sim_Text
//Calculating each component's degree
int GetRowValue(int crm [ ] [ ], int Colum, int index)
for (i = 0; i <= Colum; i++) do
sum += crm[index, i];
end for
return sum;
end GetRowValue
```

After description of the algorithm, time complexity and space complexity of this algorithm are analyzed as follows:

- **Time complexity:** Time complexity analysis is a way to evaluate the algorithm's performance from the aspects of time. The time an algorithm expended mainly means the executed times of sentences of the algorithm. As the pseudocode show above, the time complexity of Sim_Text( ) depends on Sim1( ) and Sim2( ). GetRowValue( ) which has a loop, is cycle called in Sim1( ), so these sentences are executed n^2 times. Then Sim2( ) was analyzed as the same and the result is n times. So the time complexity of the algorithm is as Eq. 8, where, n is the number that sentences are executed:

$$f(n) = n+n^2 \tag{8}$$

Then the same order of magnitudes of f(n) is n^2, so the algorithm's time complexity is as follows:

$$T(n) = O(n^2) \tag{9}$$

- **Space complexity:** Space complexity analysis refers to the resources the algorithm needed for running, including the memory space occupied by the algorithm itself, I/O data and the temporary data. As the pseudocode show above, the input data is a two-dimensional array and the common variables are storage in auxiliary space, so the algorithm's space complexity function is as Eq. 10, where n is the scale of the problem:

$$f(n) = n^2+1 \tag{10}$$

So, the algorithm's space complexity is as follows:

$$S(n) = O(n^2) \tag{11}$$

RESULTS AND DISCUSSION

The experiment corpus includes 200 texts collected from the internet. Then we analyze each text to generate a corresponding text to be detected and set a corresponding similarity value for each pair of text. So there are 200 pairs of texts altogether. The experiment tools include MATLAB 2012a and C#.

In this research, we use precision, recall and F1-Measure for analyzing the results of experiment. This three indexes are most commonly used in the field of information retrieval and natural language processing. Precision is the fraction of retrieved instances that are relevant while recall is the fraction of relevant instances that are retrieved. F1 is a synthesis evaluation parameter of precision and recall. The specific calculation equation are as follows:

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{false positives}} \tag{12}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{false negatives}} \tag{13}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (14)$$

Experiments' results analysis: We extracted the high-frequency words from texts in proportion of 20, 40, 60, 80 and 100%, respectively. As shown in Table 3 it presents the similarity values of 10 pairs of experiment's texts. We can see from the table, when there is a big difference in content between a text pair, the Chinese characters in texts also would be different. Then the CRMs of these texts are different. So the similarity value would be small, such as the results of the sixth pair of texts shown in Table 3. When a pair of texts is exactly the same, the data extracted from the texts and the results would be the same too, so the text similarity value will be 1, such as the results of the eighth pair of texts shown in Table 3. The results of the experiments that extract different proportion of high-frequency words from texts changes little.

Algorithm has two parameters to be confirmed: The extraction proportion and the coefficient α. But the criterion whether texts are similar or not depends on the similarity threshold, different thresholds have different effects on the analysis of experimental results. Therefore, we analyzed threshold at first. Figure 5a shows when the extraction proportion is 100% and α is 0.5, the impact will be quite good. When threshold is more than 0.6, F1 value decreases rapidly as the reduction of recall. The reason for this variation had been given as follows: The same components of similarity threshold on the performance of the algorithm. When the threshold is small, the result can compose different Chinese characters in different combination modes, only 505 components can form 6763 Chinese characters, so different Chinese characters could decomposed the same components in two text, then the text similarity would remain over a certain value. Therefore, when the threshold is very low, all the evaluation performance will be rather better. As the threshold increasing, the false negative will increase rapidly, the false positive decrease, then recall will decrease soon and precision increase little. So the F1 value will also decrease rapidly.

Through the analysis above, we selected 0.6 as the threshold to analyze the performance of the algorithm. Figure 5b shows the impact of coefficient α on experimental results under the condition of 100% extraction proportion. We can know from the figure, when α equals to 0.6, the F1 value is the best. Namely the similarity of CRM of single component occupied 60% in the text similarity and the CRM of text occupied 40%. Every component analyzed with CRM could express text similarity better.

Table 3: Similarity value of experiment results

	Words (%)				
Texts	20	40	60	80	100
1	0.6945497	0.7338968	0.7334071	0.7080256	0.7287503
2	0.747992	0.721931	0.7437472	0.7636345	0.7705754
3	0.9334247	0.9705478	0.977481	0.9874833	0.9915896
4	0.9590608	0.9796062	0.9887838	0.9909263	0.9939273
5	0.8371595	0.8556996	0.8744883	0.8922771	0.9029245
6	0.2966017	0.3145588	0.3136913	0.3114357	0.3017304
7	0.7393604	0.7721574	0.8074964	0.8054414	0.8093105
8	1	1	1	1	1
9	0.7439851	0.7732812	0.7587393	0.7544624	0.7795588
10	0.8907226	0.8845202	0.8977143	0.9138192	0.934136

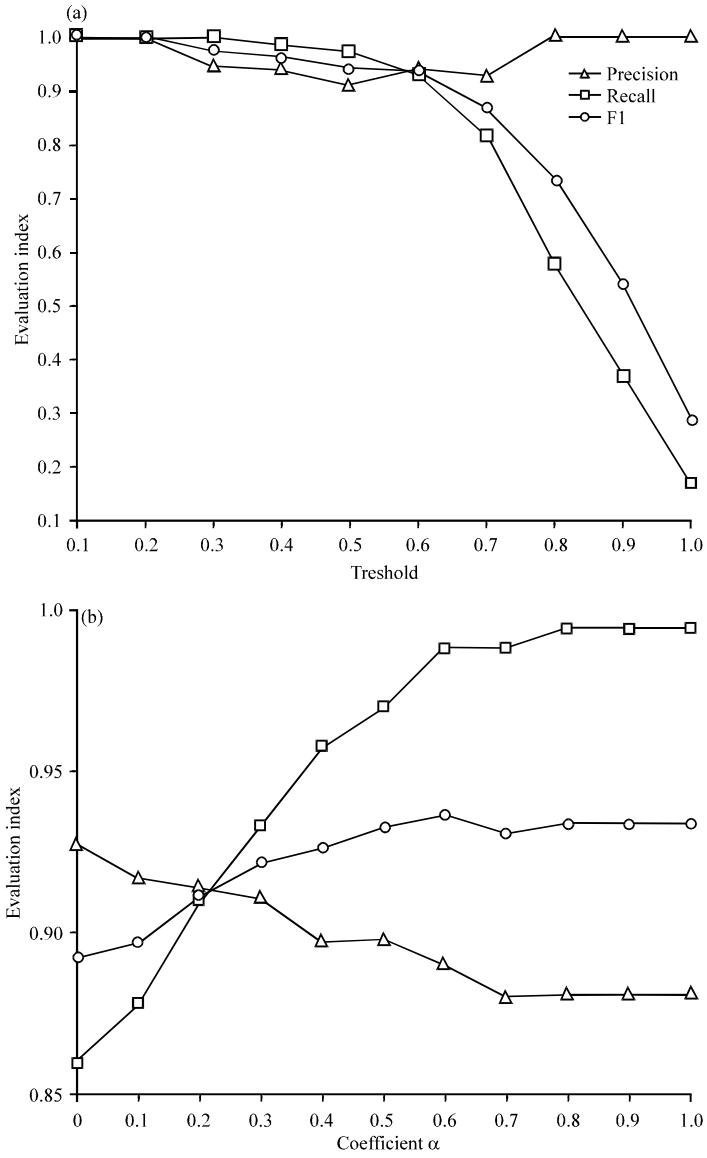

Fig. 5(a-b): Effects of (a) Threshold and (b) α on experiment result

Then, the same analyses are did under condition of 20, 40, 60 and 80% extraction proportion of high-frequency words, respectively and all the best combinations of parameters that lead to a best F1 value are obtained, as shown in Table 4.

We can see from Table 4, when coefficient α is 0.6 and the extraction proportion is 100%, F1 value is highest. This means the experiment performance is the best. The variation of conditions of different combination of parameters is not much. According to Table 3, we know that the extraction proportion affect experiment result and experiment performance little, F1 value also changed little.

After confirmed the best combination of parameters, we use CRM-TSD, cosine theorem and Jaccard coefficient to do the experiment on the same corpus to make a comparison. The cosine theorem and Jaccard coefficient methods used VSM to represent texts. The results are as shown in

Table 5, 6 and Fig. 6. Table 5 shows similarity values of experimental results, Table 6 presents the results of time and space complexity of these three algorithms and Fig. 5b shows the experimental performance.

Table 4: Group of parameters

Extraction ratio (%)	α	F1
20	1.0	0.9313
40	0.5	0.9321
60	0.4	0.9273
80	1.0	0.9333
100	0.6	0.936

Table 5: Comparison of similarity value

Text	Cosine	Jaccard coefficient	CRM-TSD
1	0.842535	0.694936	0.72875
2	0.834468	0.715955	0.770575
3	0.990039	0.978423	0.99159
4	0.998025	0.995984	0.993927
5	0.992871	0.985636	0.902925
6	0.655088	0.482757	0.30173
7	0.926866	0.863348	0.809311
8	1	1	1
9	0.882275	0.789107	0.779559
10	0.93018	0.869468	0.934136

Table 6: Comparison of algorithm analysis

Method	Time complexity	Space complexity	Time spent (sec)
CRM-TSD	O(n2)	O(n2)	1.638
Cosine	O(n3)	O(n2)	1.734
Jaccard coefficient	O(n3)	O(n2)	1.718

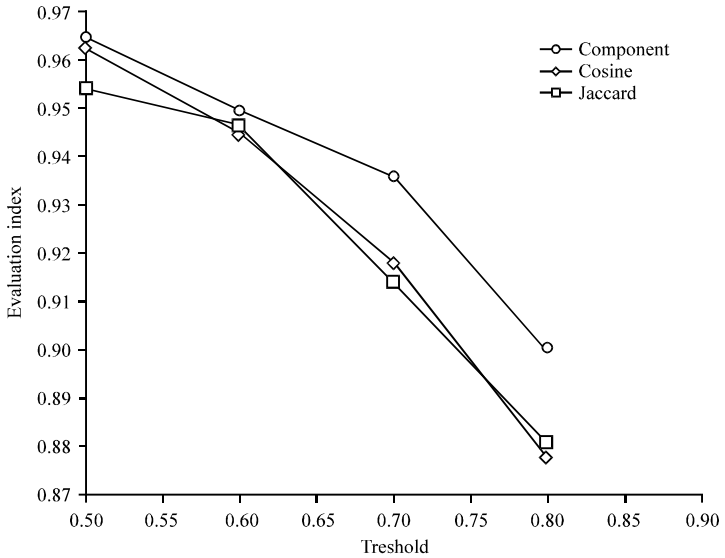

Fig. 6: Comparison of experiments' performance

We can see from Table 6, in time complexity analysis, CRM-TSD is much less than the other two methods. But the results of time spent are nearly. So we analyze the process of the experiment and found the time was mainly spent at the phrase of component decomposed from Chinese characters. Figure 6 shows that CRM-TSD achieves better F1 value than the others. This indicates that CRM-TSD perform well.

CONCLUSION

Text similarity detection is mainly used for copy detection and webpage check. It is also an effective ways for maintaining information quality. This study put forward a new algorithm of text similarity detection after the analysis and research on the structure of Chinese characters. CRM-TSD starts a new view of Chinese text similarity detection research. Chinese characters in text are split into components to build CHM and CRM. The texts similarity is obtained by computing the similarity of all CRM. The experimental results show that CRM-TSD performs better than cosine theorem (Zhang *et al*., 2012) and Jaccard coefficient (Thada and Joshi, 2011).

This study provides a new idea of the natural language processing. The method can be used for text copy detection and duplicated webpages deletion. In our future work, we will improve the efficiency of component decomposing and enhance the precision of the algorithm on the detection of the two texts that have a large variation on the number of words.

ACKNOWLEDGMENT

This study is supported by National Natural Science Foundation of China (No. 61304208, 61202496), Hunan Province Natural Science Foundation of China (No. 13JJ2031); Hunan Province Planned Science and Technology Key Project (No. 2013NK2017).

REFERENCES

Bao, J.P., J.Y. Shen, X.D. Liu and Q.B. Song, 2003. A survey on natural language text copy detection. J. Software, 14: 1753-1760.

Chen, C.L., F.S.C. Tseng and T. Liang, 2011. An integration of fuzzy association rules and WordNet for document clustering. Knowledge Inform. Syst., 28: 687-708.

Huang, C.H., J. Yin and F. Hou, 2011. A text similarity measurement combining word semantic information with TF-IDF method. Chin. J. Comput., 34: 856-864.

Peng, X.G., S.M. Liu and Y. Song, 2010. A copy detection tool for Chinese documents. Proceedings of the 2nd International Conference on Education Technology and Computer, June 22-24, 2010, Shanghai, pp: 125-129.

Shi, J., Y. Wu, L. Qiu and X.Q. Lv, 2013. Chinese lexical semantic similarity computing based on large-scale corpus. J. Chin. Inform. Proces., 27: 1-6.

Sun, X., J. Yin, H. Chen, Q. Wu and X. Jing, 2002. On mathematical expression of A Chinese character. J. Comput. Res. Dev., 39: 707-711.

Sun, C.N., C. Zhang and Q.S. Xia, 2013. Chinese text similarity computing based on LDA. J. Comput. Technol. Dev., 23: 217-220.

Thada, V. and S. Joshi, 2011. A genetic algorithm approach for improving the average relevancy of retrieved documents using jaccard similarity coefficient. Int. J. Res. IT Manage., 1: 50-55.

Wang, L.X., H.T. Geng, K. Sun and X. Zhang, 2011. Auto-detection technology of text divulgence based on natural language processing. J. Comput. Eng. Des., 32: 2600-2603.

Wang, Z.G. and M. Wu, 2011. Similarity checking algorithm in item bank based on vector space model. J. Comput. Syst. Applic., 19: 213-216.

Zhang, X., W. Xu, L. Gao and W. Liang, 2012. Combining content and link analysis for local web community extraction. J. Comput. Res. Dev., 49: 2352-2358.

Zhang, P.Y., 2013. A HowNet-based semantic relatedness kernel for text classification. TELKOMNIKA Indonesian J. Electr. Eng., 111: 1909-1915.

Zhao, D., L. Cai and P. Li, 2011. A similar text detection algorithm based on newshingling. J. Shenyang Jianzhu Univ., 27: 771-775.