J Inf Process Syst, Vol.15, No.6, pp.1503~1516, December 2019
https://doi.org/10.3745/JIPS.02.0124

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

SSF: Sentence Similar Function Based on word2vector Similar Elements

Xinpan Yuan*, Songlin Wang*, Lanjun Wan*, and Chengyuan Zhang**

Abstract
In this paper, to improve the accuracy of long sentence similarity calculation, we proposed a sentence similarity calculation method based on a system similarity function. The algorithm uses word2vector as the system elements to calculate the sentence similarity. The higher accuracy of our algorithm is derived from two characteristics: one is the negative effect of penalty item, and the other is that sentence similar function (SSF) based on word2vector similar elements doesn't satisfy the exchange rule. In later studies, we found the time complexity of our algorithm depends on the process of calculating similar elements, so we build an index of potentially similar elements when training the word vector process. Finally, the experimental results show that our algorithm has higher accuracy than the word mover's distance (WMD), and has the least query time of three calculation methods of SSF.

Keywords
Long Sentence Similarity, Similar Element, System Similarity, WMD, Word2vector

1. Introduction

Sentence similarity calculation is widely used in many fields and plays a very important role [1]. It has always been a key issue in text information processing technology and natural language processing (NLP) [2]. In different application fields, the calculation of sentence similarity expresses different meanings and functions. Therefore, it can be widely used in many applications, such as recommendation system [3], intelligent retrieval [4], paraphrase generation [5], automatic summarization [6-8], ontology building [9], digital library systems [10], question-answer systems [11], and so forth.

In recent years, the rapid development of machine learning, deep learning, and other technologies has led to the rapid progress of natural language processing and related technologies, and the fast updating iteration of the underlying technology also makes the method of sentence similarity calculation change constantly. Among them, the method of using word vector to calculate word similarity and combine with similarity function is favored by many researchers because of its higher accuracy than other methods. Guo and Xing [12] proposed a method to calculate sentence similarity by using editing distance and Jaccard coefficient with the vector as the basic element. Li et al. [13] proposed the use of word vectors to represent the meaning of words and considers the influence of multiple factors such as word meaning,

Manuscript received November 7, 2018; first revision February 27, 2019; accepted March 30, 2019.
Corresponding Author: Lanjun Wan (wanlanjun2008@163.com)
* School of Computer Science, Hunan University of Technology, China (xpyuan@hut.edu.cn, {WangSL110, wanlanjun2008}@163.com)
** School of Information Science and Engineering, Central South University, China (2561838940@qq.com)

word order and sentence length on the calculation of sentence similarity. Arora et al. [14] proposed model sentences based on word vector and unsupervised methods. Mrabet et al. [15] proposed a completely different approach to the calculation of sentence similarity based on word vector, the algorithm draws on the idea of DNA matching in biology, and fully considers the influence of word order on sentence similarity in sentence structure. Xu [16] proposed a method based on the similarity calculation function and word vector to calculate the text similarity, in this paper, this method is called WJ method, which can only calculate the sentence similarity after the change of simple synonyms. Kusner et al. [17] proposed the word mover's distance (WMD), which can solve the synonym problem well, but it has a poor effect on the similarity calculation of the new long sentence of the "short-long" type or the "long-long" type formed by the original sentence. Guan et al. [18] believe that everything in the information world can be regarded as a system, the system is composed of elements, and the system has similarity, the system similarity function can be used to calculate the similarity between the two systems.

From some prior work, we find the main problems in the current sentence similarity calculation: (1) adjusting the word order, the passive sentence and the active sentence to exchange sentence patterns; (2) sentence synonym replacement or word-of-speech words are replaced with each other; (3) extending the original text to the long text. Aiming at the above problems, the main idea of sentence similarity calculation is: words are the basic elements of the sentence, and word vectors can calculate word similarity, so the mainstream sentence similarity algorithm uses the word vector to replace the word as the expression of the sentence, then compare the expression's similarity. In order to solve the problem of original sentence extended growing sentences and their synonyms or words with the same part of speech in sentence similarity calculation, a sentence similarity algorithm based on word vector is proposed in this paper under the framework of system similarity function [19], and in order to reduce the time complexity, an SSFP (index of potential similar elements) algorithm is proposed on the basis of algorithm SSF (sentence similar function).

Inspired by the influence of system similarity function and application of word vector in sentence similarity algorithm. Our main contributions are as follows:

1. Our paper proposes a sentence similarity function based on word2vec as a system element, and calculating the similarity of sentences based on a system similarity function.
2. In order to speed up the calculation of similar elements, we propose to construct indexes of potential similar elements in the process of word vector training. This greatly reduces the calculation time.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 describes the SSF. Section 4 discusses the optimization of calculating similar elements. In Section 5, the accuracy and time consumption of SSF and SSFP is experimentally verified on three data set. Section 6 gives conclusions.

2. Related Work

With the development of NLP, the related methods of text similarity calculation are evolving. In recent years, the combination of word vector [16,17] and other similarity measurement functions has been widely concerned. One of the most famous is the WMD algorithm proposed by Kusner et al. [17]. In this paper, the method proposed by Xu [16] is called WJ algorithm. In the experimental part, we will prove the superiority of SSF by comparing the performance of WMD, WJ.

WMD combines word vector and the earth mover's distance (EMD) to calculate text distance. WMD uses the word frequency of two sentences to do the EMD algorithm respectively. Therefore, WMD conforms to the characteristics of positive qualitative, homogeneous and inequality, and also ensures that the input and output of weight value are equal, which satisfies the fourth constraint condition.

$$WMD = \min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i,j)$$
$$S.t. \begin{cases} \sum_{j=1}^{n} T_{ij} = d_i \ \ \forall \ i \in \ \{1,\dots n\} \\ \sum_{i=1}^{n} T_{ij} = d_j' \ \ \forall \ j \in \{1,\dots,n\} \end{cases} \tag{1}$$

$$c(i,j) = \| \boldsymbol{a} - \boldsymbol{b} \|_2 \tag{2}$$

where $T_{i,j}$ is corresponding weight matrix, d_i, d_j' is the frequency of words, $c(i, j)$ is the distance between the i-th word and the j-th word, a represents the i-th word vector and b represents the j-th word vector.

WJ uses word vector to realize the calculation process of word similarity in $Max(a_i, b_j)$, then the similarity of sentences is calculated by the formula (3):

$$Sim(A,B) = \frac{\sum_{i=1}^{m} w_i Max(a_i, b_j) + \sum_{j=1}^{n} w_j Max(b_j, a_i)}{\sum_{i=1}^{m} w_i + \sum_{j=1}^{n} w_j} \tag{3}$$

where m and n represent the number of elements contained in the word set A and the word set B, respectively. w_i and w_j represent the weight of the i-th word and the j-th word. Max represents the maximum similarity.

3. Sentence Similarity Function

3.1 Definition of SSF

After the word segmentation and stop word processing, sentence A could be converted into word vector set S_A by loading the word vector table. For any two sentences A and B, their similarity is expressed as the similarity of word vector sets S_A and S_B, and the sentence similarity function, $SSF(A, B)$, is used to calculate the similarity of the two sentences, which can be regarded as the cosine value of the angle α between vector X and Y:

$$X = (x_1, x_2, \dots, x_{\mathrm{m}}, 0, \dots, 0)_N,$$
$$Y = (x_1 u_1, x_2 u_2, \dots, x_{\mathrm{p}} u_{\mathrm{p}}, 0, \dots, 0, y_{\mathrm{p}+1}, y_{\mathrm{p}+2}, \dots, y_{\mathrm{n}})_N,$$

where N is a positive integer, the corresponding range of values is $N \geq m + n$; so the SSF value range is [0, 1].

- The former m terms $x_i (1 \leq i \leq m)$ in the vector X is the word's weight of S_A;
- The n terms y_j $(1 \leq j \leq n)$ in the vector Y is the word's weight of S_B;

- The former p terms u_i of $x_i u_i (1 \leq i \leq p)$ in the vector Y represent the similarity of similar elements of S_A and S_B;
- The former p terms x_i of $x_i u_i (1 \leq i \leq p)$ in the vector Y represent the weight of similar elements;
- The latter terms $y_k (1 + p \leq k \leq n)$ represent the weight of non-similar elements;

The similarity calculation of any two sentences can be expressed by *SSF*(*A*, *B*), as shown in formula (4).

$$SSF(A,B) = \frac{\sum_{i=1}^{p} u_i {x_i}^2}{\sqrt{\sum_{i=1}^{m} x_i^2} \sqrt{\sum_{i=1}^{p} u_i^2 x_i^2 + \sum_{i=p+1}^{n} y_i^2}} \tag{4}$$

In formula (5), x_i and y_i represent the weight of the word, and the weight of the word is calculated by using the frequency of the word.

$$x_i = \frac{e_i}{\sum_{i=1}^{m} e_i} \quad y_i = \frac{e_i}{\sum_{y=1}^{n} e_i} \tag{5}$$

where e_i represents the number of times that the i-th word appears in the sentence.

3.2 The Features of SSF

It is easy to prove that *SSF*(*A*, *B*) satisfies the following conditions:

(a) *SSF*(*A*, *B*) is a monotonically increasing function for the weights of similar elements;

(b) *SSF*(*A*, *B*) is a monotonically decreasing function for the weights of non-similar elements;

(c) *SSF*(*A*, *B*) is a monotonically increasing function for the similarity of similar elements;

3.2.1 Monotonic

Through the above three available conditions of (a) (b) (c), the SSF algorithm has monotonicity, and the feature of the function ensures that the similarity of two sentences with the parameters of the linear monotone increasing or decreasing.

3.2.2 Does not satisfy the exchange law

SSF(*A*, *B*) ≠ *SSF*(*B*, *A*) could be verified. This feature of SSF has great advantages in calculating two sentences with a large difference in length. For example, in the search engine domain question answering system or the intelligent retrieval system, the question text is usually short but the high-quality answer text or the search result text is usually long, therefore, it does not satisfy the exchange law to calculate the text similarity based on "left" (the first text), which is more consistent with the application of the actual question and answer and retrieval system.

3.2.3 Penalty items

In general, some words (such as negative words, interference words, etc.) in the text have negative or reverse effects on the expression of the whole text. The SSF has a penalty effect on non-similar elements (including single words, negative words, noise words, etc.) according to the condition (b). This feature of the SSF will show good accuracy for the similarity calculation of long sentences.

3.3 The Calculation of Similar Elements

The time complexity of SSF is mainly affected by the calculation of similar elements, that is, if the time of similar element calculation becomes shorter, the calculation time of SSF will be shortened. on the contrary, the calculation time of SSF will increase. u_i represents the similarity of a similar element, as shown in formula (6).

$$u_i = \begin{cases} \arg\max\limits_{b_j \in S_B} (sim(a_i, b_j)), & if > u_0 \\ 0 \quad , & if < u_0 \end{cases} \tag{6}$$

where the sim (a_i, b_j) represents the cosine of the angle between two vectors as the calculation of similar elements.

u_0 is the judgment of the similarity threshold. Guo and Xing [12] provides the following relevant conclusions: [0, 0.25) is not similar, [0.25, 0.4) is similar, [0.4, 0.5) is very similar, and [0.5, 1] is basically equivalent.

We give an intuitive way to calculate similar elements as shown in Table 1. In this paper, the calculation method of this double cycle is called SSFN (SSF naive).

Table 1. SSF of calculating similar elements u

Input: S_A, S_B, u_0
Required variables: S=[ ] is similar element set ; NS=[ ] is non-similar set of elements, $maxsim$=0;
Output: u=[]

```
1: For $a_i$ in $S_A$ do
2:     For $b_i$ in $S_B$ do
3:     if $cos(a_i, b_i) > maxsim$
4:       $maxsim = cos(a_i, b_i)$;
5:     End for
6:     if $maxsim > \mu_0$ then
7:        $S$.add($a_i$)
8:        $u$.add($maxsim$)
9:     else
10:        $NS$.add($a_i$)
11         $u$.add(0);
12: End for
```

4. The Optimization of Calculating Similar Elements

In the context of big data, the text-similarity performance test system requires an efficient sentence similarity algorithm, the time complexity of the SSF focuses on the optimization of calculating similar elements.

4.1 SSFT (SSF Temporary Index)

To reduce the calculation of a double cycle to one cycle, a further approach is to construct an index of S_B for each vector a_i in S_A.

According to experience, the dimension of the word vector is generally 200–300 dimensions to get better results, using the open-source project FAISS to do this index job.

For a vector a_i in S_A, we search for the vector b_j with the highest similarity in the temporary *index*$_i$, so that the process requires only one similarity calculation.

The n times calculations of similar elements $<a_i, b_j>$ are reduced to vector searching, thereby reducing the execution time of *SSF*.

But there is a flaw that when every time similar elements of a sentence are calculated, a temporary index needs to be built once, and the index cannot be reused.

The method of a temporary index is called the SSFT algorithm (SSF Temporary Index), as shown in Fig. 1.

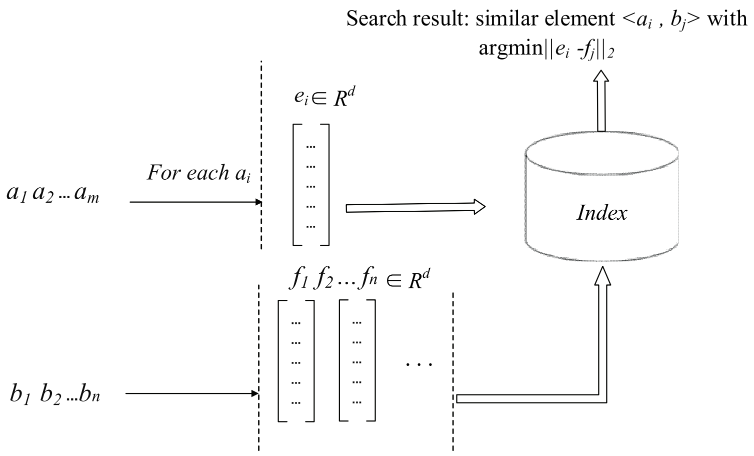

Search result: similar element $<a_i , b_j>$ with $\mathrm{argmin}||e_i - f_j||_2$
$e_i \in R^d$
For each a_i
$a_1\, a_2 \dots a_m$
Index
$f_1\, f_2 \dots f_n \in R^d$
$b_1\, b_2 \dots b_n$

Fig. 1. Similar elements calculation process of SSFT.

4.2 SSFP (Index of Potential Similar Elements)

In order to solve the problem that the index can't be reused, we off-line establish an index of potential similar elements in the process of word vector training. We could search for the index to complete the calculation of similar elements without having to repeatedly create a temporary index.

The steps for creating the index of potential similar elements:

Step 1: Establishing an index for all the word vector set by trained word vector model.

Step 2: Traversing any vector v to search the index to get a return set. In this set, the potential similar elements have abstained with the similarity is greater than the threshold u_0, in similarity descending order.

Step 3: The physic index of potential similar elements could be implemented by a Huffman tree.

According to the hierarchical Softmax strategy in word2vec, an original word2vec Huffman tree constructed on the basis of the word frequency, and each node (except the root node) represents a word and its corresponding vector.

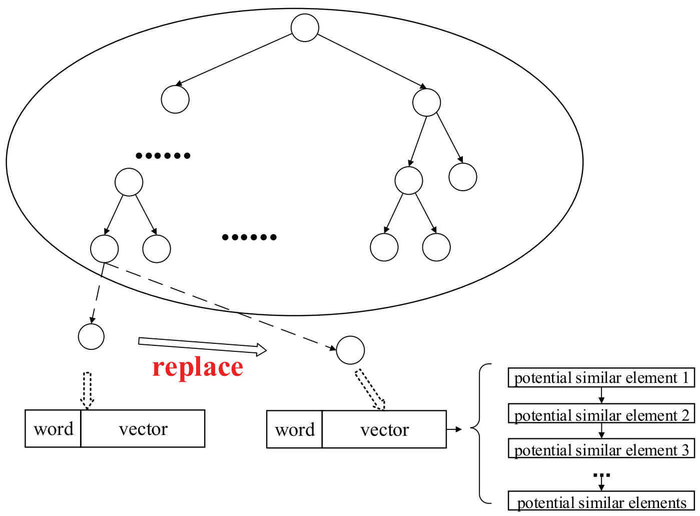

replace
word
vector
word
vector
potential similar element 1
potential similar element 2
potential similar element 3
potential similar elements

Fig. 2. The physic data structure of potential similar elements.

We try to replace the vector with potential similar elements. So each node of the tree represents a word and its corresponding potential similar elements instead, as shown in Fig. 2. This method via the global index of potential similar elements is called to be SSFP, as shown in Table 2.

Table 2. SSFP of calculating similar elements u

Input : S_A, S_B, u_0
Required variables: S=[ ] is similar element set ; NS=[ ] is non-similar set of elements, P=[] is potential similar elements; $maxsim$=0;
Output: u=[]

```
1: For $a_i$ in $S_A$ do
2:  $P$[]= $searchHuffman(a_i)$;
3:  For $b_j$ in $S_B$ do
4:     For $p_k$ in $P$[] do
5:       if $b_i$ .equals($p_k$.vector)
6:          $S$.add($a_i$)
7:          $u$.add($p_k.sim$)
8:          Break to loop $a_{i+1}$;
9:     End for
10:  End for
11:     $NS$.add($a_i$)
12:     $u$.add(0);
13:  End for
```

4.3 Time Complexity Analysis of SSFN, SSFT, and SSFP

Suppose that the number of sentence pairs to be similarly calculated is L, the average number of word vectors of the word vector set S_A is m, and the average number of word vectors of the word vector set S_B is n, d represents the dimension of the vector.

(1) In SSFN, whether all elements which constitute similar elements (u_i) are calculated once by using

formula (6), and the time consumption of calculation is determined by the number of vector dimension, the time complexity of SSFN is $O(Lmn\times Tcos)$, where $Tcos$ is the time of $cos(\theta)$ between vector a_i and b_j , and equals d. so the time complexity of SSFN is $O(Lmn\times d)$.

(2) In SSFT, in order to reduce the number of similar element calculations in SSFN, the method of constructing an index is used, the index is equivalent to fuzzy search, and then the similar element is calculated to determine whether it constitutes a similar element, the time complexity of SSFT is $O\ (LmTindex +Lnlog(m)d)$, where $Tindex$ is the time to index each word vector, and $log(m)$ is the number of times to look up in the index. Since the $Tindex$ value is small that $LmTindex$ can be ignored, compared with $Lnlog(m)d$. the time complexity of SSFT approximately equals to $O(Ln\times log(m)d)$.

Suppose d (the number of dimensions) is 300, m (the number of word vectors of the word vector set S_A) is 50, and K (the total number of the dictionary) is 160,000. We can deduce that the time of $Tcos$ is seen like 300 times of CPU operation, the time of *searching Huffman tree* $log(K)$ is seen like 400 times of CPU operation. So the time proportion of SSFP:SSFT:SSFN equals to 400:2100:15000. So the time complexity of SSF is shown in Table 3.

Table 3. The time complexity of SSF

Algorithm	Time complexity	Proportion
SSFN	$O(Ln\times md)$	150
SSFT	$O(Ln\times log(m)d)$	21
SSFP	$O(Ln\times log(K))$	4

5. Experimental Evaluation

5.1 Experimental Preparation

5.1.1 Word vector training corpus

The data set of training word vector model is Wikipedia Chinese corpus [20] which requires data cleansing, traditional and simplified conversion, and so on. The amount of Wikipedia Chinese corpus downloaded is about 1.12 GB and 4 million articles. The model of training word vector is the combination of the Skip-gram model and hierarchical Softmax strategy, and the code of training word vector is the word vector toolkit [21]. In general, the threshold value of calculating similar elements is $\mu_0 = 0.5$, the dimension number of word vector is 200.

5.1.2 Evaluated algorithms

The algorithms involved in the time complexity comparison experiment are SSFN, SSFT, and SSFP, since SSFN, SSFT and SSFP only calculate the similarity element in different ways, the error of the similarity calculation results of the three is zero, and the similarity accuracy is consistent. SSFN, WMD and WJ algorithms were selected to participate in the experiment of precision comparison. An overview of the algorithms involved in the accuracy comparison is shown in Table 4.

Table 4. Relevant algorithms for participation in the accuracy comparison

Algorithm	Brief description	Ref.
WMD	Calculating the moving distance of word vector	[22]
SSF	Calculating the similarity by similar elements and no-similar elements	[23]
WJ	Take the weight of words into account in the similarity calculation	[24]

5.1.3 Experimental data

The experimental dataset is divided into the following three categories:

– News: from Guangming Daily, People's Daily, and other News articles (400 news), the average length of news is 500 words;
– Wikis: from Wikipedia and other encyclopedia texts (400 wikis), the average length of the wiki is 2000 words;
– Papers: from papers and periodicals, etc. (400 papers), the average length of a paper is 20,000 words.

Each kind of data set which is randomly selected from the three categories is divided into four groups, each group contains 100 pieces. The first group has 100 titles of 10 words, the second group has 100 titles of 25 words, the third group has 100 titles of 35 words, and the fourth group has 100 titles of 50 words (Table 5).

Table 5. The groups of the experimental dataset

	Dataset		
	News	Wikis	Papers
Number of data	400	400	400
Average number of words	500	2,000	20,000
Average number of sentences	20	80	800
Group	Group 1 (100 titles of 10 words), Group 2 (100 titles of 25 words), Group 3 (100 titles of 35 words), Group 4 (100 titles of 50 words)		

5.2 Evaluation Indicator

The first is the accuracy of similarity calculation comparison, the essence of WMD is to calculate the distance between texts, and WJ and SSF are to calculate the similarity directly. The distance of the text and similarity can swap, the smaller the distance, the greater the similarity, the greater the distance, the smaller the similarity. In order to make a reasonable evaluation, we used sentence similarity calculation in [2] as an experimental method.

As shown in Fig. 3, the experimental process is divided into two stages: calculation and result comparison:

1. Calculation step: Traversing the similarity (or distance) between the title sentence and the S_i in the sentence set S.
2. Results comparison step: The sentence with the highest similarity (or the smallest distance) compared with the artificially marked sentence.

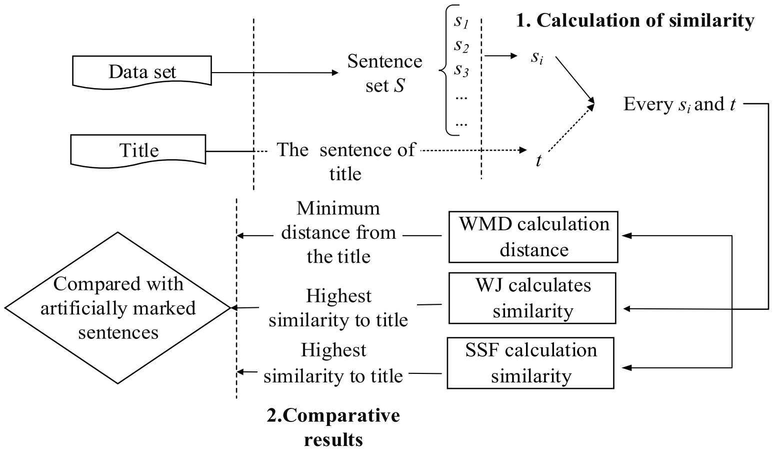

1. Calculation of similarity
Data set
Sentence set S
s_1
s_2
s_3
…
…
s_i
Every s_i and t
Title
The sentence of title
t
Minimum distance from the title
WMD calculation distance
Compared with artificially marked sentences
Highest similarity to title
WJ calculates similarity
Highest similarity to title
SSF calculation similarity
2.Comparative results

Fig. 3. Experiment process.

5.2.1 Accuracy

First of all, a graduate student of Chinese language literature is invited to manually mark the most similar sentences which are the most similar to the title in each type of data set. The algorithm then calculates the sentences that are most similar to the title. For example, we take the title as the original sentence, to calculate the minimum distance with the original sentence on WMD, to calculate the most similar to the original sentence on WJ and SSF. Finally, the accuracy of the algorithm is examined by comparing the difference of accuracy between the results of the algorithm and the result of artificial markers.

In each class of data sets, 100 articles are selected according to the length of the title 10, 25, 35, and 50 words and then divided into 4 groups. Each algorithm is carried out in each group of data sets, and then the average accuracy of each algorithm in each group is calculated. Three kinds of datasets are selected and 4 groups, each algorithm calculates the average accuracy of 12 times, and there are 3 algorithms involved in the comparison of similarity accuracy. In total, the average accuracy was calculated to 36 times.

5.2.2 Query time

Time performance is evaluated by the average cost time comparison of algorithms (SSFN, SSFT, and SSFP) hits in each group, there are 4 algorithms involved in the comparison of query time. In total, the average query time was calculated to 48 times.

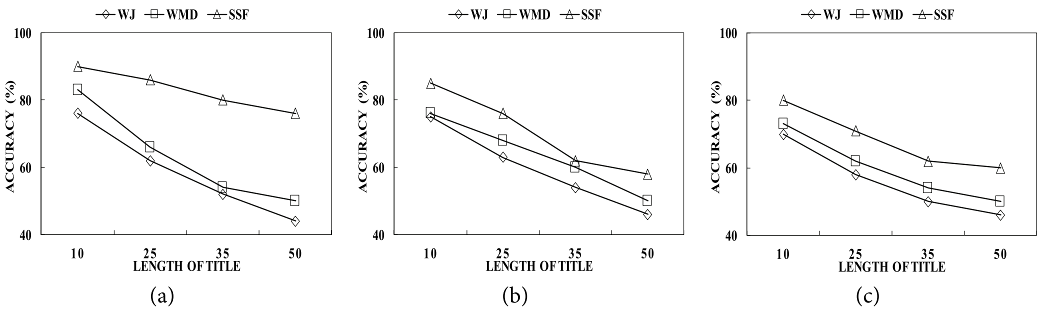


Fig. 4. The comparison of the average accuracy: (a) News, (b) Wikis, and (c) Papers.

5.3 Accuracy Evaluation

The comparison of the average accuracy on algorithms (WMD, SSF, and WJ) is shown in Fig. 4. There are the following discussions:

(1) The average accuracy of each algorithm is different under the same header length. The general trend is that as the number of sentences increases, the average accuracy tends to decrease. And the accuracy comparison of the 3 datasets shows that the larger the dataset, the smaller the accuracy. This is normal because the increase in data size will always increase the error hit, leading to a decrease accuracy.

(2) WJ's accuracy reduces with the length increased, since WJ only considers the influence of the word's weight, the similarity of words and other factors on the overall similarity of sentences, and it couldn't effectively excavate the deep semantic information of the sentence and lacks the penalty term, so it has not solved the similarity of the long sentence.

(3) SSF's accuracy is always higher than WMD in the five groups of each data set. We try to explain the results with the theoretical analysis:

- The negative effect of penalty item: SSF's penalty items composed of noise words, single words, and non-similar elements. SSF takes full account of the harmful or undesirable influences of penalty terms on the overall similarity of sentences. However, WMD does not consider the negative effects of penalty terms on the calculation of sentence similarity.
- Not satisfying the exchange rule: WMD satisfies the exchange rules. Taking the title sentence as the standard, SSF compares the text sentence to the title sentence. When the length of the two sentences is basically the same, the two algorithms do not show any significant difference. When two sentences differ greatly in length, the SSF algorithm will show great advantages.

In conclusion, SSF takes the first sentence as the standard, would be ideal for two sentences with large length differences and partial content similarity, good at calculating "short-long" sentence similarity. At the same time, SSF takes full account of the negative effect of penalty terms on the overall sentence similarity, the accuracy of similarity calculation of "long-long" sentences is improved, which also makes SSF have high accuracy.

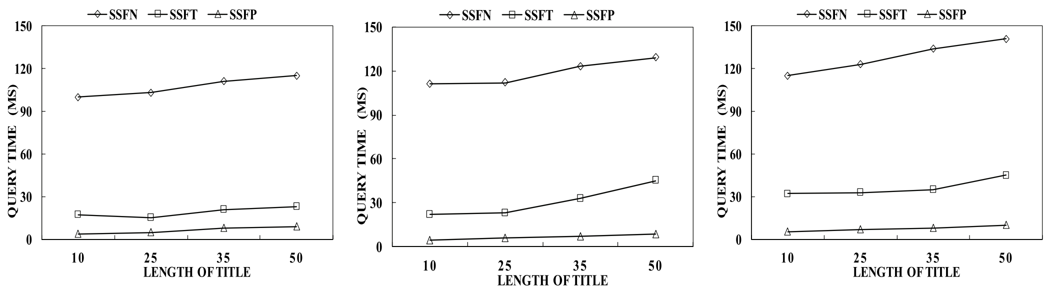
5.4 Query Time Evaluation

This experiment is a comparative experiment of the average time-consuming index of SSFN, SSFT, and SSFP under three sets of data sets, and the results measured in the experimental data sets are shown in Fig. 5. There are the following discussions:

(1) SSFN takes the most time because SSFN needs mn times of similar element calculation. SSFT builds temp index to lookup to complete the calculation of similar elements, takes less time than SSFN. The disadvantage of SSFT is that every time the similarity of a sentence is calculated, an index should be set up, the index cannot be reused. Using the idea of finding similar elements instead of calculating them, the difference of SSFP is that SSFP establishes the global index for all potential similar element of word vector by training the word vector model meanwhile. So the global index could be reused, SSFP takes less time than SSFT.

(2) As the length of the title increases in the same data set, the average time spent by each algorithm increases. This is because the number of times the similar elements are calculated in long sentences is more than that of short sentences, so it takes more time to calculate the similarity of long sentences than

short sentences. At the same time, as the number of sentences included in the data set increases, the average time-consuming time of each algorithm tends to increase. For example, the Paper set contains more sentences than News and Wiki, so the average time taken in the group of the same title length is high.



(a) (b) (c)

Fig. 5. Time-consuming results of SSFN, SSFT, and SSFP: (a) News, (b) Wikis, and (c) Papers.

6. Conclusions

This paper presented and analyzed some possible approaches for sentence similarity functions. We proposed the SSF that uses word2vector as the system elements to calculate the similarity of sentences. The accuracy results show our novel SSF generally performs superior. The time complexity analysis and time experimental evaluation show the query time performance of SSFP is optimal.

Acknowledgement

We are grateful to the support of research fund of Hunan Province (2018JJ2099, 19C0558, CX1911), Hunan open fund (Project Similarity detection cloud platform; Clothing material aggregation and prediction), Hunan Education Reform fund (The whole process performance management and evaluation system of innovation and entrepreneurship center).

References

[1] M. U. Devi and G. M. Gandhi, “Query expansion on the role of word and sentence similarity for domain ontology driven fuzzy retrieval systems,” *Journal of Computational and Theoretical Nanoscience*, vol. 14, no. 6, pp. 2612-2619, 2017.

[2] W. Yin, K. Kann, M. Yu, and H. Schutze, “Comparative study of CNN and RNN for natural language processing,” 2017; https://arxiv.org/abs/1702.01923.

[3] D. Zhang, T. He, Y. Liu, S. Lin, and J. A. Stankovic, “A carpooling recommendation system for taxicab services,” *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 254-266, 2014.

[4] J. R. Lin, Z. Z. Hu, J. P. Zhang, and F. Q. Yu, “A natural-language-based approach to intelligent data retrieval and representation for cloud BIM,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 1, pp. 18-33, 2016.

[5] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual LSTM networks," 2016; https://arxiv.org/abs/1610.03098.

[6] M. A. Boudia, A. Rahmani, M. E. Rahmani, A. Djebbar, H. A. Bouarara, F. Kabli, and M. Guandouz, M. "Hybridization between scoring technique and similarity technique for automatic summarization by extraction," *International Journal of Organizational and Collective Intelligence*, vol. 6, no. 1, pp. 1-14, 2016.

[7] P. W. McBurney and C. McMillan, "Automatic source code summarization of context for Java methods," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 103-119, 2015.

[8] S. K. Bharti and K. S. Babu, "Automatic keyword extraction for text summarization: a survey," 2017; https://arxiv.org/abs/1704.03242.

[9] Y. C. Lee, C. M. Eastman, and W. Solihin, "An ontology-based approach for developing data exchange requirements and model views of building information modeling," *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 354-367, 2016.

[10] J. Muralikumar, S. A. Seelan, N. Vijayakumar, and V. Balasubramanian, "A statistical approach for modeling inter-document semantic relationships in digital libraries," *Journal of Intelligent Information Systems*, vol. 48, no. 3, pp. 477-498, 2017.

[11] G. Zhou, J. Zhao, T. He, and W. Wu, "An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities," *Knowledge-Based Systems*, vol. 66, pp. 136-145, 2014.

[12] S. Guo and D. Xing, "Sentence similarity calculation based on word vector and its application research," *Modern Electronics Technique*, vol. 39, no. 13, pp. 99-102, 2016.

[13] F. Li, J. Hou, R. Zeng, and C. Ling, "Research on multi-feature sentence similarity computing method with word embedding," *Journal of Frontiers of Computer Science and Technology*, vol. 11, no. 4, pp. 608-618, 2017

[14] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[15] Y. Mrabet, H. Kilicoglu, and D. Demner-Fushman, "TextFlow: a text similarity measure based on continuous sequences," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017, pp. 763-772.

[16] S. Xu, "Research and implementation of paraphrasing recognition technology for question-and-answer system," Harbin Institute of Technology, Harbin, China, 2009.

[17] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015, pp. 957-966.

[18] Y. Guan, X. Wang, and Q. Wang, "A new measurement of systematic similarity," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 4, pp. 743-758, 2008.

[19] Y. Guan, X. Wang, and Q. Wang, "Measurement of system similarity," in *Proceedings of China Computational Linguistics Conference (CCL)*, Nanjing, China, 2005, pp. 341-347.

[20] Wikimedia Chinese corpus [Online]. Available: https://dumps.wikimedia.org/zhwiki/latest/zhwiki-latest-pages-articles.xml.bz2.

[21] Word2VEC_java [Online]. Available: https://github.com/NLPchina/Word2VEC_java.

[22] wmd4j is a Java library for calculating Word Mover's Distance (WMD) [Online]. Available: https://github.com/crtomirmajer/wmd4j.

[23] Word Sentence Similarity Code [Online]. Available: https://download.csdn.net/download/u011001835/9849524.

[24] Word2VEC [Online]. Available: https://github.com/jsksxs360/Word2Vec.

Xinpan Yuan https://orcid.org/0000-0001-9509-0755

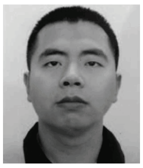
He received Ph.D. degree in computer science from the Central South University, in 2012. Currently, he is a lecturer in School of Computer Science of Hunan University of Technology, China. His research interests include information retrieval, data mining and NLP.

Songlin Wang https://orcid.org/0000-0001-7952-0479

He received B.S. degree in computer science from Hunan University of Technology, in 2017. Currently, he is a master student in School of Computer Science of Hunan University of Technology, China. His current research interests include information retrieval and NLP.

Lanjun Wan https://orcid.org/0000-0001-7236-3589

He received his Ph.D. degree in Computer Science from Hunan University, in 2016. He is currently an assistant professor of Computer Science at Hunan University of Technology. His research interests include high-performance computing, parallel computing, heterogeneous computing and parallel programming model.



Chengyuan Zhang https://orcid.org/0000-0003-2721-6867

He received PhD degree in computer science from the University of New South Wales. Currently, he is a lecturer in School of Information Science and Engineering of Central South University, China. His main research interests include information retrieval, query processing on spatial data and multimedia data.