대용량 문서 집합에서 전역 사전을 이용한 빠른 유사문서 탐색 방법

A Fast Heuristic Search for Similar Documents
using Global Dictionary in Massive Document Repository

부산대학교 컴퓨터공학과
Park Sun Young
E-mail : `parksy@pusan.ac.kr`

Revised on 2010.09.12, Submitted to SAC 2011

ABSTRACT

Since plagiarism has become a social problem, there has been a recent surge in studies on plagiarism detection. Especially, because text plagiarism is growing rapidly with the development of the Internet, so many plagiarism detection algorithms have been proposed. However, most algorithms focus on the optimized one-to-one comparison rather than massive document comparison. The latter algorithms have a limitation in time performance when users carry out an exhaustive search on sets of many documents. In this paper, we propose an optimized preprocessing model to detect similar text in massive document repositories. This model uses an efficient data structure called GDIC (Global DICtionary). We experimentally prove the performance of the model. Our experiments show that the searching time via pre-process model is drastically more reduced than that without the preprocess model, and computation time is reduced to 70% or more while sensitivity is maintained at 75%. When we use this model, GDIC generation time accounts for a large proportion of all of detection time. In future work, we will optimize GDIC creation time to improve the performance of the entire system.

KEYWORDS text plagiarism, similar document, document repository, dictionary

1 Introduction

Plagiarism, as defined in the 1995 Random House Compact Unabridged Dictionary, is the "use or close imitation of the language and thoughts of another author and the representation of them as one's own original work."[1]. One of the main issues in Research ethics is Plagiarism. We can plagiarise more easily than before through the Internet, since the Internet provides us with much information. So, it is getting more difficult to detect text plagiarism due to the large quantity of texts to be inspected. Many automated detecting plagiarism algorithms are proposed, but most are suitable for one-to-one comparison rather than that for massive documents. Since plagiarism detection time depends on the number of pairs used for inspecting plagiarism, it is inefficient to detect all pairs of documents. Thus, if we can reduce the search space, we expect to save computing time. In this paper, we propose the preprocessing model using an efficient new data structure called GDIC (Global DICtionary) to detect similar texts in sets of numerous documents. Our proposed preprocessing model can reduce the time cost for detecting similar texts through extracting a suspicious documents set. We experimentally prove the efficiency of this model.

2 Related works

Studies on automatically detecting plagiarized documents have recently made steady progress. Many methods of finding similar texts have recently been introduced. They can be divided into two types, which are attribute counting and structure metric[2].

In the attribute counting method, similar texts are inspected through comparing suspicious documents based on the frequency of words in the texts[2]. In the latter method, document size hardly affects the effectiveness of plagiarism detection time, and it also has shorter search time. The attribute counting method, however, has limitations that it cannot report the specific location of corresponding sentences in two documents. It includes Blast[3], SCAM[4], fingerprinting algorithms, etc. In recent research, the most commonly used way based on the fingerprints algorithm. Fingerprints of a document yield the set of all possible document substrings of a certain length, called fingerprints, and fingerprinting is the process of generating fingerprints[5]. Many efforts[6, 7] have been made to find similar texts using fingerprints. The extension studies on fingerprints[8, 9, 10] are developed actively due to the limitation that attribute counting method cannot find a specific area. And studies about preprocessing to decrease inspection space are also carried out[11, 12].

In the Structure metric method, similar documents are detected by computing the similarity of Token String after analyzing the structure of the document[2]. In this method, the size of the document is proportional to the detection time of similar texts. However, the latter method is useful for detecting partially matched sentences, so it is usually used to detect programming code plagiarism. The extension studies of this method to find plagiarism[13] have been conducted. DeVAC[14] uses a combined algorithm that takes each strength of fingerprints and structure metric to detect plagiarism. The system reduces the domain that requires comparison through preprocessing, and it finds the specific location of the same sentences using the structure metric method[15].

However, these methods only consider time performance in one-to-one comparison or many-to-many comparison in a narrow set, so it has limited performance in terms of time. If we carry out exhaustive search on all the pairs in large document sets, the detection time increases rapidly. Let us assume that the number of documents in A set is n. If we carry out plagiarism detection, then the number of the pairs that needs inspection is n^2. Therefore, it is practically impossible to find similar texts since detection time increases exponentially if the number of documents falls outside a certain level. Therefore, we suggest the optimized pre-processing model to detect plagiarism in massive document repository. The proposed preprocess model reduces the total searching time by decreasing the inspected space. We use the DeVAC system to conduct experiments for measuring the performance of the preprocessing model since DeVAC shows high performance in a one-to-one comparison[15]. So, we will conduct an experiment on time performance by applying the preprocessing model.

3 Preprocessing Model

We propose a preprocessing model using the Global Dictionary to reduce search space. In the reduced space, we can dramatically reduce search time. An overview of the similar document search system that applied this model is shown in Figure 1.

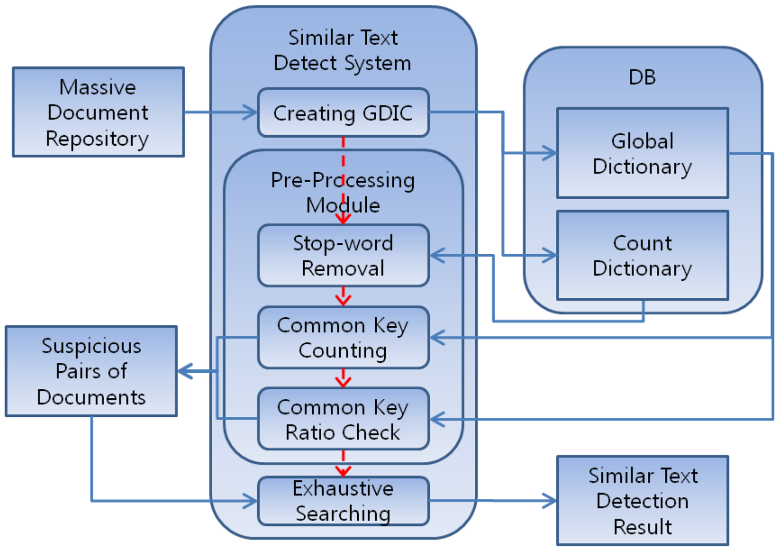


FIGURE 1. An overview of a similar document searching system that applied our preprocessing model. The dotted lines indicate the flow of execution, and the solid lines represent the flow of data. The entire process consists of three steps that are creating of GDIC, preprocessing, and exhaustive searching. The preprocessing model consists of three steps, which are stop-word removal, common key count, and common key ratio check. GDIC is used by the preprocessing module.

Our proposed system consists of three steps, including the Creation of GDIC(Global DICtionary), preprocessing, and exhaustive search. Each step is described as follows.

3.1 Creating GDIC(Global DICtionary)

Let us define some notations for a detailed explanation.
D : a set of documents.
d_i : a document consists of an ordered list of words.

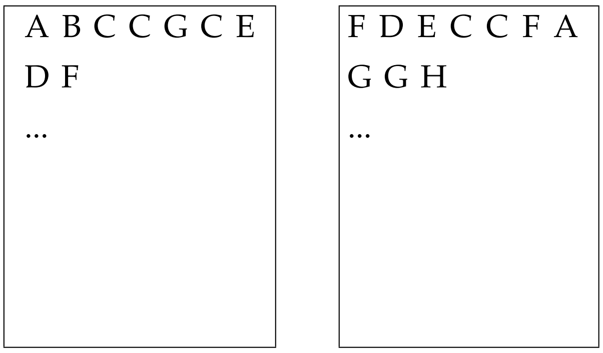
$$D = \{d_1, d_2, ..., d_i, ..., d_n\}, so|D| = n. \tag{1}$$

$$d_i =< w_{i,1}, w_{i,2}, ..., w_{i,j}, ..., w_{i,m} >, so|d_i| = m. \tag{2}$$

$w_{i,j}$: jth word in document d_i.
$Unique(d_i)$: the set of different words in d_i.

$$Unique(d_i) = \{u_k | u_k = \forall w_{i,j} \in d_i\} \tag{3}$$

$freq(d_i, w)$: the number of the word w that appeared in document d_i. For example, we suppose documents d_a and d_b as follows.



A B C C G C E
D F
...

F D E C C F A
G G H
...

(1)d_a (2)d_b

Then, $freq(d_a,$'C'$) = 3$ and $freq(d_b,$'F'$) = 2$.
$Unique_{set}(D)$: the set of different words in document d_i in a set D.

$$Unique_{set}(D) = \{us_j | \forall us_j \in d_i, \forall d_i \in D\} \tag{4}$$

$freq_{set}(D, w)$: the number of the document which has a word w. For example, D has d_a and d_b, $freq_{set}(D,$'C'$) = 2$ and $freq_{set}(D,$'B'$) = 1$.

First, the proposed system asked the user to enter D as input dataset. The system generated a Global Dictionary from D. Global DICtionary(GDIC) consists of two data structures, $Table_A$ and $Table_B$.

$$GDIC(D) =< Table_A(D), Table_B(D) > \tag{5}$$

$Table_A(D)$ is a data structure that consist of $d_i, u_{i,j}$ and $freq(d_i, u_{i,j})$.

$$Table_A(D) = \{A_1, A_2, A_3, ..., A_n\} \tag{6}$$

$$A_k =< d_i, u_{i,j}, freq(d_i, u_{i,j}) > \tag{7}$$

$Table_B$ has all word in D and $freq_{set}(D, word)$.

$$Table_B(D) = \{B_1, B_2, B_3, ..., B_n\} \tag{8}$$

$$B_k =< word \in D, freq_{set}(D, word) > \tag{9}$$

$Table_A(D)$ is shown in Table 1 and $Table_B$ is shown in Table 2. Generally, the size of a document set increases, and the generation time of GDIC is also increased because the GDIC creation process requires disk-access intensive computation.

doc.	word	$freq(d_i, w)$	doc.	word	$freq(d_i, w)$
d_a	A	1	d_b	F	2
d_a	B	1	d_b	D	1
d_a	C	3	d_b	E	1
d_a	G	1	d_b	C	2
d_a	E	1	d_b	A	1
d_a	D	1	d_b	G	2
d_a	F	1	d_b	H	1

TABLE 1. $Table_A(D)$ obtained from two documents d_a and d_b in D. keyword is a word in d_i, $freq(d_i, w)$ means the frequency of the word w in document d_i.

word	$freq_{set}(D, w)$	word	$freq_{set}(D, w)$
A	2	E	2
B	1	F	2
C	2	G	1
D	2	H	1

TABLE 2. $Table_B$ obtained from two documents d_A and d_B in D. $freq_{set}(D, w)$ means the number of the set of documents that have word w. For example, $freq_{set}(D,$'B'$)$ is 1 because the word 'B' in D_a only. But the word 'A' is in both A and B, so $freq_{set}(D,$'A'$)$ is 2. Such key that appeared in many documents will be a stop-word.

3.2 Preprocessing Method using GDIC

After generating GDIC, we can preprocess using this data structure. In this process, the system found suspicious pairs of documents. The preprocessing model consists of three steps that are stop-word removal, common key count, and common key ratio check.

First, the set of stopwords, Stop(D) is defined as follows.

$$Stop(D) = \{s_i | \frac{|freq_{set}(D, s_i)|}{|D|} > P_{stop}\} \tag{10}$$

The stop-word removal process prevents the use of the frequent words like 'have' or 'the'. So, we can use less written words like 'plagiarism' to find suspicious pairs of text. P_{stop} is proportional to stop-words. To perform a more efficient preprocessing, proportion of stop-words needs to be set to 0.995 or more. After the stop-word removal process is completes, we can preprocess using GDIC. Preprocessing is comprised of two steps. In the first step(step A), we use a number of common keys, and in the second step(step B), we use a percentage of common keys. $\mathbb{D}$ is the set of all unordered pairs of distinct elements in D.

$$\mathbb{D}(D) = \{(x, y) | x, y \in D \wedge (x \neq y)\} \tag{11}$$

In step A, we measure the number of appearing non-stop-word keys. If some keys that appear in each text are more than N_{match} times and these keys appear in two texts more than the total S_{match} times, then the system judges that the pair of documents need to exhaustively search each other. So, the pair of texts has entered the suspicious queue, D_{match}. The set D_{match}, subsets of $\mathbb{D}$, are defined as follows.

$$D_{match}(D) = \{(x, y) | word \in Stop(D)^c M(freq(x, word), freq(y, word))\} \tag{12}$$

,where

$$M(x, y) = (x \geq N_{match}) \wedge (y \geq N_{match}) \wedge (x + y \geq S_{match}) \tag{13}$$

N_{match} and S_{match} are configurable parameters in the system. Through the experiments, we uncovered the appropriate values N_{match} and S_{match}. The value of N_{match} is a proper 1-5, and the value of S_{match} is a proper 5-15. For example, we assume documents A and B. When N_{match} is 2 and S_{match} is 5, if a non-stop-word appears 3 times in A and B both, then D_{match} includes the pair of A and B becauseN_{match} = 2 ¡ 3 and S_{match} = 5 ¡ 3 + 3. If the higher $Stop(D)$ of the above, then the computing time is decreased, but sensitivity is lowered. So, it is important to use the appropriate settings.

After step A is done, almost all suspicious pairs of the documents are into the inspection queue. However, some short texts are not included in the inspection queue because the small text has only 1-2 times per key. To avoid this situation, we need a method to find the suspicious pairs of short texts. The set D_{common}, subsets of $\mathbb{D}$, are defined as follows.

$$D_{common} = \{(x, y) | \frac{|Unique(x) \cap Unique(y) \cap Stop(D)^c|}{\min\{|Unique(x)|, |Unique(y)|\}} \geq P_{common}\} \tag{14}$$

In step B, we measure the percentage of common non-stop-word keys. If the percentages of any pairs are greater than P_{common}, then the pairs go into the inspection queue D_{common}. The value of P_{common} is a proper 0.2-0.5. For example, when P_{common} is 0.4 and if the ratio of the common key is 40% or more, then D_{common} includes the pair of documents.

3.3 Exhaustive Search for Reduced Sets

The reduced set of unordered pairs of documents that will be finally compared is denoted by $D_{reduced}$, which is defined as follows.

$$D_{reduced}(D) = D_{match}(D) \cup D_{common}(D) \tag{15}$$

After preprocessing is completed, we conduct an exhaustive search on all the suspicious pairs of documents. In this process, we use the similar document detection engine of the DeVAC system. We do not mention the DeVAC system in this paper because the performance of the system has already been verified in previous articles. At the end of this process, we can see similar areas and similar values of each pair of text through the result file.

4 Experiments

4.1 Prepared Dataset

No.	No. of text	Size(MB)	No.	No. of text	Size(MB)
1	6,263	1,494.73	5	1,197	123.16
2	8,852	114.37	6	1,084	105.16
3	1,237	97.18	7	1,369	115.11
4	1,374	113.17	8	323	27.56

Table 3. Description of our input datasets. Set 1 has the largest size to evaluate the capacity of dealing with large size data. Although Set 2 has the largest number of texts, the size of its data is average. And Set 8 is the smallest set. The other sets have similar to each other.

We used 8 document sets of policy-related reports to measure the performance of our proposed model. We summarized the input datasets in Table 3. Especially, Set 1 is the largest set to evaluate the capacity of dealing with the large size data. And Set 2 has the most texts, but its size is average. Set 8 is the smallest set.

4.2 Experimental Methods

To compare of previous method with our method, we will measure the computing time and sensitivity of both methods. The experimental method is as follows. First, we do an exhaustive search on all pairs of documents using the DeVAC system without preprocessing. Moreover, we count the number of pairs of documents with absolute similarity as being over 500. Absolute similarity is the measured value of how similar the two documents are between specific areas. If this value is over 100, then the pair requires human exhaustive check. If the value is over 500, the we can judge that plagiarism has occurred in the area. Moreover, we measured processing time. Second, we will perform tests using the proposed model. We must set the appropriate parameters mentioned in the previous chapter. The proportion of stop-word was set to 0.995, N was set to 2, S to 7, and P to 0.02. The measurements were performed in the same way including detailed measurements of search time(generation of GDIC, preprocessing, and exhaustive search respectively).

4.3 Results and Analysis

No.	Previous method(sec.)	Our method(sec.)	γ (%)
1	144,886.92	25,788.27	82.21
2	217,126.33	28,126.53	87.05
3	4,332.81	1,253.44	71.07
4	4,731.23	1,328.13	71.93
5	4,011.98	1,218.38	69.63
6	3,826.36	1,181.21	69.13
7	5,283.65	1,012.26	80.84
8	632.59	436.02	31.07

TABLE 4. Experimental result on 8 document sets described in Table 3. We compared the computation time between our proposed method and the previous method. Gamma(γ) means decreasing rate of computing time. Our method can save 63.90-87.05% of the time costs in terms of search time. However, we can reduce only 31.02% of the time costs in the smallest 8th set because the generation time of GDIC is too large to preprocess with a small set.

The results of the measured time that was required for the operation are shown in Table 4. In most of input datasets, computation time was reduced by more than 63.90-87.05%. But in Set 8, the smallest set that has 323 elements, the width is reduced by 31.02%; it is low in comparison to the other sets. In other words, in a large set of documents consisting of more than 1000, we can benefit from preprocessing, but in a small set of 100-300, the benefits are not great. Table 5 shows a number of pairs of documents that have an absolute similarity of over 500.

No.	Number of document pairs		Sensitivity
	Previous method	Our method	
1	1,303	1,164	89.33%
2	1,246	1,062	85.23%
3	1,032	961	93.12%
4	1,420	1,181	83.17%
5	867	721	83.16%
6	782	604	77.24%
7	1,443	1,230	85.24%
8	452	432	95.58%

Table 5. The experimental result on 8 document sets described in Table 3. A comparison of sensitivity between the previous method and our method. The 'number of pairs of documents' value means the number of pairs that have an absolute similarity of over 500. This is the performance of 77.24-95.58% compared to those of conventional methods. Conversely, about 5-23% of the omissions may occur. We can improve a lot of performance of the system using our method. In contrast, the system miss a few suspicious pairs.

The results were as follows. In all sets, our proposed system found 77.24% to 95.58% of pairs of documents that were found by the previous method. It means that if the preprocessing model was applied, then the system could miss 20% of the results. Analysis of a missing pair of documents is necessary to reduce the miss ratio. Figure 2 shows the sensitivity and reduction of search time of our model. Table 6 shows the processing time required to perform every step in Set 1. In this set, each step of computation time using our method was measured as follows. Nearly half of the total search time(48.95%) is occupied with the GDIC creation time. Exhaustive search has progressed much of the optimization, and preprocessing with 13.4% of the entire process is relatively short. So, the GDIC creation process is essential for optimization. If GDIC generation time is reduced, then the overall system performance will improve even further.

5 Conclusion

In this paper, we proposed a preprocessing filter based on a new data structure called GDIC in order to improve the inspection capability for large numbers of documents. To preprocess, we used a data structure called GDIC. In order to decide whether or not to thoroughly inspect

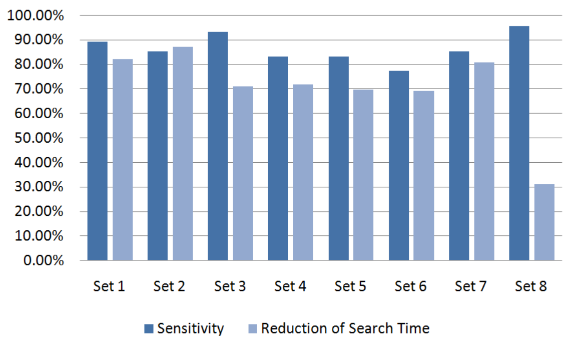


FIGURE 2. The experimental result of sensitivity and reduction of search time of our model. In the 8 document sets described in Table 3. In most of the sets, sensitivity was measured as being 77.24% or more. Search time reduction was measured as 63.90% or more. In other words, our model works effectively. Exceptionally, in Set 8, our system achieved only 31.07% of improved time costs in terms of search time because the processing time of generating GDIC added a load to the entire processing time.

documents, we measured the frequency of the occurrence of Key and the proportion of Key. We could filter the pairs that did not need to be inspected based on theses data. In our experiment, 6,200 documents were used and we measured the detecting time both with the proposed model and without preprocessing. The result shows that plagiarism detection time with preprocessing is significantly reduced than without. We could decrease the searching time in the 63.90% to 87.05% range maintaining sensitivity over 77.24%. In the process, it took a lot of time to generate GDIC.

	Previous method	Our method
creation time of GDIC	-	12,624.32s
preprocessing time	-	3,458.62s
exhaustive searching time	144,886.92s	9,705.33s
total searcing time	144,886.92s (40.24h)	25,788.27s (7.16h)

TABLE 6. In Set 1, a comparison between the previous methods and our methods in terms of computation time. Total search time, was reduced to 17.80% compared to that of the previous method, so it was 25,778.28 seconds. Creation time of GDIC accounted for 48.95% of the total time, and exhaustive search time was 37.64% of the total search time. And preprocessing time was 13.4% of the total. If we can optimize the GDIC generation process, then the system performance will be enhanced.

It required almost time in the 47.23% to 72.43% range of total operating time. If GDIC is already created or we do not need to make GDIC, then the searching time could be reduced to 50 to 80%. So in the future, we will study preprocessing with optimized and simplified GDIC and try to achieve process without GDIC.

References

1. "Dictionary.com unabridged," . http://dictionary.reference.com/browse/plagiarism
2. John L. Donaldson, Ann-Marie Lancaster, and Paula H. Sposato, "A plagiarism detection system," in *Proc. of ACM SIGSCE*, 1981, pp. 21–25.
3. Michael Cameron, H. E. Williams, and Adam. Cannane, "Improved gapped alignment in blast," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 1, no. 3, pp. 116–129, 2004.
4. N. Shivakumar and H. Garcia-Molina, "Scam: A copy detectino mechanizm for digital documents," in *Proc. pf DL*, 1995.
5. Rajiv Yerra and Yiu-Kai Ng, "A sentence-based copy detection approach for web documents," *Lecture Notes in Computer Science*, vol. 3614, no. Part 1, pp. 557–570, 2005.
6. Udi Manber, "Finding similar files in a large file system," in *Proc. of USENIX*, 1994, pp. 1–10.
7. S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," in *Proc. of ACM SIGMOD*, 1995, pp. 398–409.
8. Nevin Heintze, "Scalable document fingerprinting," in *Proc. of USENIX*, 1996.
9. Chow Kok Kent and Naomie Salim, "Features based text similarity detection," *Journal of Computing*, vol. 2, no. 1, pp. 53–57, January 2010.
10. Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken, "Winnowing: local algorithms for document fingerprinting," in *Proc. of ACM SIGMOD*. 2003, pp. 76–85, ACM.
11. Manuel Zini, Marco Fabbri, Massimo Moneglia, and Alessandro Panunzi, "Plagiarism detection through multilevel text comparison," in *Proc. of IEEE AXMEDIS*, Washington, DC , USA, 2006, pp. 181–185.
12. Zdenek Ceska and Chris Fox, "The influence of text pre-processing on plagiarism detection," in *Proc. of RANLP*. September 2009, pp. 55–59, Association for Computational Linguistics.
13. Takahisa Ota and Shigeru Masuyama, "Automatic plagiarism detection among term papers," in *Proc. of IUCS*, 2009, pp. 395–399.
14. Chang-Keon Ryu, Hyoung-Jun Kim, Su-Hyun Park, and Hwan-Gue Cho, "Devac(document evolution analysis center)," . http://devac.cs.pusan.ac.kr
15. Chang-Keon Ryu, Hyoung-Jun Kim, and Hwan-Gue Cho, "Developing of text plagiarism detection model using korean corpus data (in korean)," *Journal of KIISE : Computing Practices and Letters*, vol. 14, no. 231-235, pp. 2, 2008.