# A Plagiarism Detection Algorithm based on Extended Winnowing

Xuliang DUAN[1,a], Mantao WANG[1], Jiong MU[1]

[1]*College of Information Engineering, Sichuan Agricultural University, Ya'an 625014, China*

**Abstract.** Plagiarism is a common problem faced by academia and education. Mature commercial plagiarism detection system has the advantages of comprehensive and high accuracy, but the expensive detection costs make it unsuitable for real-time, lightweight application environment such as the student assignments plagiarism detection. This paper introduces the method of extending classic Winnowing plagiarism detection algorithm, expands the algorithm in functionality. The extended algorithm can retain the text location and length information in original document while extracting the fingerprints of a document, so that the locating and marking for plagiarism text fragment are much easier to achieve. The experimental results and several years of running practice show that the expansion of the algorithm has little effect on its performance, normal hardware configuration of PC will be able to meet small and medium-sized applications requirements. Based on the characteristics of lightweight, high efficiency, reliability and flexibility of Winnowing, the extended algorithm further enhances the adaptability and extends the application areas.

## 1 Introduction

Plagiarism detection, also known as duplicate or copy detection is to detect whether the contents of one file copy from one or more others. Strictly speaking, plagiarism means more than intact copy, and also includes the transformation of transposing, synonym substitution and restatement of original ideas without reference [1]. Plagiarism detection is an important method for academic misconduct, author identification and digital products intellectual property protection. The PAN (Plagiarism Detection Author Identification Author Profiling) founded by EU digital library project hold the annual competition on plagiarism detection, author identification and author profiling which involves the most advanced method and technology in this field [2]. In China, several major bibliographic databases such as CNKI, Wan Fang Data, CQVIP constructed their own plagiarism detection system that effectively prevent the occurrence of academic misconduct.

Plagiarism is an unavoidable topic in digital age. "In surveys, nearly 70% of college students admit to having taken material from the internet without properly crediting its source" [3]. And in recent decades, various forms of plagiarism and plagiarism are increasing year by year [4]. In terms of financial and time costs, the commercial detection system is hard to meet the demand of lightweight, real-time, and normalization for students' assignments plagiarism detection. In recent years, plenty of research in this field has been done and some results are obtained. Zhong et al. transformed the C source code to XML document and then extracted the XML structure information to detect code block and statement rearrangement etc. [5]. Qin et al. presented an algorithm based on local word-frequency fingerprint and improved the simple digital fingerprinting and word frequency based algorithms [6]. Zhang et al. compared the Winnowing fingerprint algorithm and the dynamic programming in assignment plagiarism detection and found that the Winnowing has better anti-interference ability for noise words and it is more effective and reliable [7-8].

This paper presents an approach to extend the Winnowing fingerprint algorithm, for each text fragment in a document, the location and length are preserved while generating the fingerprint. The fingerprint extra information is used to locate and mark plagiarism text in source documents that extends the application field and practicability of the Winnowing algorithm.

## 2 Related Technologies

### 2.1 Plagiarism detection

The research of text copy detection can be traced back to the code similarity detection which is used to prevent the large-scale program copying in 1970s. Natural language copy detection technology appeared in the 1990s. Till now, the plagiarism detection research is broadly classified into two categories: the internal and external plagiarism detection. The internal plagiarism detection aims to determine whether the text is plagiarism mainly based on the author's writing style model, while external algorithms compare the suspicious document with a closed documents setting to check the similarity whether

[a] Corresponding author: duanxuliang@sicau.edu.cn

over a certain threshold. The general meaning of plagiarism detection refers to the external detection.

According to the text feature extraction, there are mainly three kinds of methods of plagiarism detection [9]: 1) Grammar based methods. These algorithms focus on document grammatical structure and measure the document similarity based on string matching, such as the LCS (Longest Common Substring), Karp-Rabin string matching algorithm [10], and the MOSS text block fingerprint detection algorithm. The grammar based methods have higher efficiency and precision for direct replication. 2) Semantic based methods. Based on vector space model, these methods first construct document feature vector using term frequency or TF-IDF (Term Frequency–Inverse Document Frequency), and then calculate vectors' similarity using dot product or the cosine of vector angle. While it's generally not easy for these approaches to determine the location of the plagiarism text. 3) The hybrid of semantic and grammatical. Integrate the above two methods in order to complement one another and achieve better results.

The Winnowing [11] algorithm is a fingerprint based text similarity detection method, proposed by Schleimer et al in 2003. The basic idea of Winnowing comes from the Karp-Rabin algorithm which using overlapping *k*-gram and moving window for string matching. Winnowing chooses the minimize hash value in each window to compose the document fingerprint, and then compares documents' fingerprints using pair-wise method to find the copied text. Winnowing is a lightweight and flexible similarity detection method, it is robust for sentence and text block rearrangement, and the influence of interference words can be effectively reduced through reasonable parameter setting.

There are three parameters, $t$, $k$, $w$, used to control the detection granularity and fingerprints density, adjustment of these parameters makes Winnowing be capable of different detection requirements. Parameter $t$ is the guarantee threshold, any substring match at least as long as $t$ would be detected, and any matches shorter the noise threshold $k$ will not be detected. Bigger $k$ value means longer $k$-gram substring and once detected plagiarism the result is more irrefutable, but larger $k$ may reduce the detection sensitivity. Parameter $w$ is the scan window size, smaller $w$ will lead to higher detection sensitivity, but more hashes causes greater storage requirements and the processing efficiency will be reduced. Schleimer et al. proved that the expect selected density is $d = \frac{2}{w+1}$, and parameters can be adjusted according different application requirements.

The basic process of Winnowing fingerprint selection algorithm is as follows:

a. Text preprocessing, including the case conversion, and removal of blank characters, stop words, meaningless symbols, etc.;

b. Using $k$-gram method to segment the text and calculate the hash of each block, so there will be a hash sequence representing the document;

c. Using the $w$ width window to sample the document hashes. The selection principles are as follows: in each window select the minimum hash value, if there is more than one hash with the minimum value, select the rightmost occurrence;

d. Save all selected hashes as the document fingerprint for further processing.

### 2.2 Plagiarism Text Location

Text location is an important components of plagiarism detection technology. The purpose of plagiarism detection is not only to calculate document similarity, but also to get the copied text location and be marked in original documents. At present, copied text location is largely based on character comparison method, such as the longest common subsequence. For example, Zaslavsky et al. constructed MDR (Match Detect Reveal) [12] system for Spanish similar text location, Sediyono et al. proposed the LCCW (Longest Commonly Consecutive Word) [13] algorithm to find and locate similar contents of a document. In order to improve the efficiency, the character comparison method is mostly indexed by words. But for Chinese text, there are few clear separators between words, word segmentation must be proceeded before indexing so the processing efficiency is relatively low.

## 3 Extended-Winnowing Algorithm

Winnowing is a high flexibility algorithm, the detection sensitivity and fingerprints density could be adjusted to different application scenarios. However, in practice, due to the lack of location information, the similar text matching and marking are costly. This paper presents a method of keeping the original text block location to extend the basic Winnowing algorithm, which can locate anyone fingerprint in original document and improve the adaptability of the detection algorithm.

### 3.1 Extracting Fingerprints with Location Information

The first step in Winnowing is to remove irrelevant features from source text, such as punctuation, spaces, control characters, stop words. After cleaning, $seg_i$ indicates the $i$-th $k$-gram text block, and the fingerprint is the hash value of this text block, that is $h_i = hash(seg_i)$. It could be seen that from the above process, text length has been changed and the location of segment hash is lost as well.

In extended Winnowing, in order to accurately locate the text segment in source document, any preprocessing that may change text length will not execute any more and the text cleanup operation is implemented in processing the $k$-gram segment. The ith $k$-gram text segment $seg_i$ contains $k$ useful characters, as the meaningless characters are removed so the length of text segment in origin document is not less than $k$, therefor $len_i \geq k$. The extended fingerprint is made up of triples

$h_i = \{hash(seg_i), loc_i, len_i\}$

In which $loc_i$ is the start location of text segment $seg_i$ in origin document, and the $len_i$ is the source length of $seg_i$ in origin document.

After obtaining the hash sequence of a document, the next step is to select the "proper" hashes according to a certain rule and then generate a new sequence of hashes which represents the origin document. Algorithm 1 gives an extension method to calculate all the *k*-gram fingerprint sequences of a document. Algorithm 2 describes the process of winnowing and selecting the fingerprints sequence.

**Algorithm 1.** Fingerprint extraction algorithm of extended Winnowing

**INPUT**: the *text* in document, *stopwords* List, length of *k*-gram(value of *k*)

**OUTPUT**: the *k*-gram hashes list *H*[ ] (without Winnowing)

**Function** GetTextFingers(*text, stopwords, k*)
**for** (*loc* = 0; *loc* < *text.length* - *k*; *loc*++) **do**
*len* ← *k*; //the default length of *k*-gram is *k*
*kgram* ← *substring*(text, loc, len); //get *k*-gram substring
*kgram* ← *replace*(*kgram*, *stopwods*, '');
//clear stop word characters
**while** *length*(*kgram*) < k **do**
//the length of *k*-gram may less than *k*,
//continue until the length is *k*
*len* ← *len* + ( *k* - *length*(*kgram*) );
*kgram* ← *substring*(*text*, *loc*, *len*);
*kgram* ← *replace*(*kgram*, *stopwods*, '');
**endwhile**
*H*[ ].*add*(<*hash*(*kgram*), *loc*, *len*>);
//calculate each *k*-gram hash,
//and record the location and length
**endfor**
**return** *H*[ ] //return all *k*-gram hashes list
**endfunction**

**Algorithm 2.** Fingerprints Sampling algorithm for extended Winnowing

**INPUT**: all *k*-gram hashes list *H*[ ], threshold *t*, *k* of *k*-gram

**OUTPUT**: sampled hashes list *HS*[ ]

**Function** SamplingFingers(*H*[ ], *t, k*)
*w* ← *t* + 1 - *k*; //scan window width
*n* ← *H*[].*count;* //all hashes count
*minIndex* ← -1;
*preMinIndex* ← -1;
*tmpMin*;
**for** (*i* = 0; *i* ≤ *H*[].*count* - *w*; *i*++) **do**
*tmpMin* = -1;
**for** ( *j* = *i*; *j* < *i* + *w*; *j*++ ) **do**
**if** *H*[*j*].*hash* ≤ *tmpMin* **then**
//if there are more than one minimum in a window,
*minIndex* ← j; // take the rightmost one
*tmpMin* ← *H*[*j*].*hash*;
**endif**
**endfor**
**if** *minIndex* ≠ *preMinIndex* **then**
//the same value only select once
*preMinIndex* ← *minIndex*;
*HS*[ ].*add*( *H* [*minIndex*] );
**endif**
**endfor**
**return** *HS*[ ] //return sampled hashes list
**endfunction**

### 3.2 Text Similarity and Plagiarism Rate

After Winnowing, the document is represented as a series of discrete hashes, and Jaccard distance [14] can be used to measure the similarity between documents. There are many variants of Jaccard for practical application, the basic definition for two sets *X* and *Y* is the quotient of intersection elements number and union elements number, as shown in formula (1):

$$J(X,Y)=\frac{|X\cap Y|}{|X\cup Y|}=\frac{|X\cap Y|}{|X|+|Y|-|X\cap Y|} \quad (1)$$

In practical, too much difference between text size may lead an inaccurate result for Jaccard distance, for example, if *A* is a proper subset of *B*, although *A* is totally plagiarized while the Jaccard distance will less than 1.

In order to improve the precision and recall ratio, we set the minimum elements number of *A* and *B* as the denominator, as shown in formula (2), where *F*(*A*) and *F*(*B*) are the fingerprints set of documents *A* and *B*:

$$sim(A,B)=\frac{|F(A)\cap F(B)|}{\min(|F(A)|,|F(B)|)} \quad (2)$$

TF-IDF (Term Frequency–Inverse Document Frequency) could also be introduced to compute the documents similarity. Documents are presented by vectors in VSM (Vector Space Model), and the angle cosine value indicates similarity of two vectors.

Since the fingerprint weight calculated by the TF-IDF increases in proportion to the number of occurrences it appears in the file and decreases inversely with the frequency it appears in all documents, so the similarity results are more reasonable. For example, assuming that each document has a fingerprint corresponding to the string "*int main ()*", the term frequency *tf* of this fingerprint is 1, and the inverse document frequency *idf* is 0, so the TF-IDF weight $w = tf \cdot idf$ is 0, which means that the string fragment "*int main ()*" will not be considered in computing documents similarity. The TF-IDF results are more accurate, while in the case of large documents the corresponding VSM may be very sparse and facing very low processing efficiency.

As each document is represented by a discrete hash sequence, the document plagiarism rate is also relatively simple, formula (3) shows the calculation method, *F(A)* represents all fingerprints of a document, *F'(A)* represents the fingerprints also appear in other documents (suspected plagiarism fingerprints):

$$P(A)=\frac{|F'(A)|}{|F(A)|} \quad (3)$$

### 3.3 Location and Marking

Essentially, the plagiarism detection is to calculate the intersection of a specific document from each file in source documents. In processing, for every same hash

value (suggest the copied text fragment) there will be a plagiarism hash index generated,

*p-index* ={*pid*, *ploc* , *plen*, *sid*, *sloc*, *slen*}

where *pid* is the current detection document ID and *sid* is the source document ID, *ploc* and *plen* are respectively the location and length of copied text fragment in current detection document, correspondingly *sloc* and *slen* are parameters in source document.

Plagiarism text marking separately processing in current detection document and source document, the principle is basically the same, the marking implementation process is as follows:

1) Sorting. Sort the plagiarism hash index ascending order by *ploc*;

2) Merging. If two adjacent plagiarism hash index are derived from a same source document and the interval is less than a certain threshold, the two indexes could be merged. Algorithm 3 describes the detail processing of the merging adjacent plagiarism hash index;

3) Marking. According to the position and length information in plagiarism hash index, respectively mark plagiarism text segments in detection and source documents.

**Algorithm 3.** Merging Adjacent Plagiarism Hash Index

**INPUT**: plagiarism indexes *P*[ ] of a document, the interval threshold *spacer*

**OUTPUT**: merged plagiarism indexes *P*[ ]

**Function** MergeFingerIndex( *FI*[ ], *spacer*)

*P*[ ] ← *sort*( *P*[ ] ) //sort by *p_loc* in ascending order

**for** ( *i*= *P*[ ].*count* -1; *i* > 0; *i*-- ) **do**

//check the adjacent indexes from behind

//if the two adjacent indexes have the same document id

//and the interval is less than threshold, merge the two into one

**if** *P*[*i*].*pid* = *P*[*i* - 1].*pid* && *P*[*i*].*sid* = *P*[*i* - 1].*sid* **then**

**if** *P*[*i* - 1].*ploc* + *P*[*i* - 1].*plen* + *spacer*≥ *P*[*i*].*ploc* && *P*[*i*-1].*sloc* + *P*[*i*-1].*slen* + *spacer* ≥*P*[*i*].*sloc* **then**

*P*[*i* - 1].*plen*= *P*[*i*].*ploc*- *P*[*i* - 1].*ploc*+ *P*[i].*plen*;

// update the length

*P*[*i* - 1].*slen*= *P*[*i*].*sloc*- *P*[*i* - 1].*sloc*+ *P*[i].*slen*;

// update the length

*delete P*[*i*];

//merging, update the previous one and delete the after one

**endif**

**endif**

**endfor**

**return** *P*[ ];

**endfunction**

Figure 1 describes the merge process of adjacent fingerprints with an example. The first 3 indexes have the same *pid* and *sid*, and the interval is less than threshold, so they are combined into one index; the interval between 3rd and 4th indexes is greater than threshold and will not be processed; the 4th and 5th are merged; although interval between 5th and 6th is less than threshold, but as the *sid* is different, skip the merge process. It should be noted that, as the algorithm skips the space and other stop words in the fingerprint extraction process, the *plen* and *slen* is not necessarily exactly the same in one fingerprint index. The merged fingerprint index contains the plagiarized text source and its exact location in the original document, which can be used for marking and positioning.

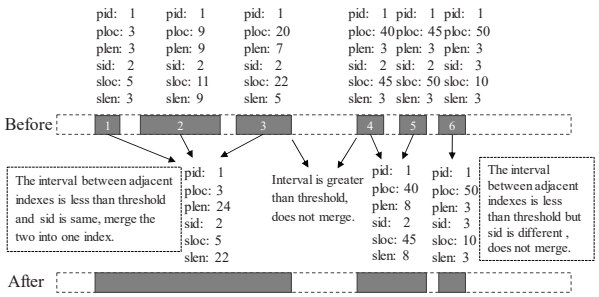


**Figure 1.** An example of merging neighboring fingerprint indexes

## 4 Experiment and Analysis

The data used in the experiment are derived from the PAN2013 corpus, there are total 3230 source text documents, and the file size ranges from 1KB to 507KB, in total 13.8MB. There are 1827 documents waiting to be checked, the file size ranges from 1KB to 101KB, in total 15.1MB. We run a PC with Intel E7500 2.93GHz CPU and 2GB physical memory and WIN7 for test platform. The algorithms are implemented by C#.

Winnowing algorithm as a highly efficient and flexible plagiarism detection algorithm, many scholars have confirmed its effectiveness in plagiarism detection [8, 15]. The extended Winnowing algorithm does not change the principle of the algorithm itself, therefore, the precision and efficiency are not affected.

The extended Winnowing algorithm performance is first tested. According to the fingerprint sampling method described above, the 3230 source documents are first *k*-gram hashed, then process the fingerprint sampling, and then get time-consuming according to the average file size. Figure 2 shows the relationship between file size time-consuming. The abscissa represents the text file size (KB), and the ordinate indicates the average time-consuming in milliseconds(ms). In order to preserve the text location information, the stop words removing is processed in the *k*-gram splitting stage, which is the main reason for the decrease of the algorithm efficiency. Before extending, the efficiency is about 270KB/s, and the extended algorithm efficiency is reduced to about 200KB/s, that is, about 200,000 characters per second.

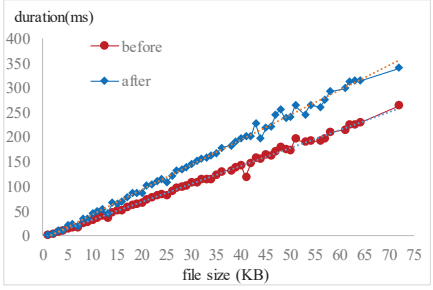


**Figure 2.** Relation between file size and time-consuming of original/extended finger computing algorithm

Operational efficiency test shown in Figure 3, the figure can be seen, time-consuming and overall file size linear relationship.

The following test is about plagiarism detection algorithm. For the 1827 waiting checked documents, each file is examined to determine whether it is suspected plagiarism from the 3230 source documents, and then collect statistics of average time-consuming for each file. Figure 3 shows the efficiency, the abscissa is the file size (KB), and the vertical axis represents the average time-consuming (ms). It can be found that time-consuming and file size overall show a linear trend. Checking a file of 100KB with all 3230 documents in source takes about 1 second.

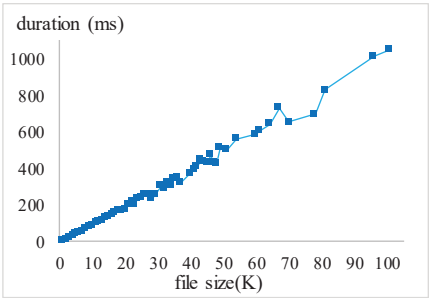


**Figure 3.** Relation between file size and time-consuming of plagiarism detection

Figure 4 shows the testing result of the file "suspicious-document00119.txt" in PAN2013 corpus. The result is displayed as a web page, and the plagiarism text sections are highlighted. When the mouse hover over the plagiarism text above, the source document will be displayed.

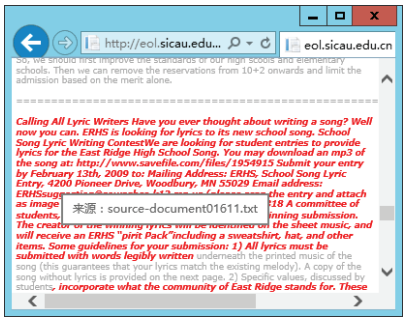


**Figure 4.** Marking of plagiarism text in html page

## 5 Conclusion

This paper introduces the basic principles and methods of extending the Winnowing algorithm. Based on the characteristics of the original lightweight, high efficiency, reliability and flexibility of the algorithm, the extended algorithm well reserve the position information of plagiarized text in detecting process, improves the algorithm function and further expand the adaptability and usability of the algorithm. Experiments show that the extended algorithm has no significant effect on the performance of the original algorithm, and the execution efficiency can meet the requirements of small and medium-sized applications in general hardware configuration, and has certain practical value.

The algorithm has been running over 3 years in our learning platform, some algorithms implement and parameters can be further optimized, and the fingerprint index space occupancy remains to be further optimized. In the future work, we will further optimize and improve the algorithm, and constantly enhance the algorithm adaptability, in order to achieve better results.

## Acknowledgements

The research is supported by the Sichuan Provincial Department of Education natural science project of “Construction of Big Data Hierarchical Storage Model for Agricultural Internet of Things” (15ZB0017).

## References

1. Junpeng Bao, Junyi Shen, Xiaodong Liu, et al. Journal of Software, **14**,1753-1760 (2003)
2. PAN 2015, Overview of the 6th international competition on plagiarism detection, http://www.uni-weimar.de/medien/webis/events/pan-15/pan15-web/index.html
3. Blum S. D. *My word! Plagiarism and college culture* (New York: Cornell University Press, 2009)
4. Martin D F. Journal of Education for Business, **80**,149-152 (2005)
5. Mei Zhong, Liping Zhang, Dongsheng Liu. Computer Engineering and Applications, **47**,215-218 (2011)
6. Yuping Qin, Qiangkui Leng, Xiukun Wang, et al, Computer Engineering, **37**, 193-194 (2011)
7. Huidong Ma, Guohua Liu, Xu Li, et al. Computer Engineering and Science, **29**,63-64(2007)
8. Liang Zhang, Xiumin Liu, Xiujuan Liu. Computer Engineering and Science, **31**,147-149(2009)
9. Wang T, Fan X Z, Liu J. *2008 International Conference on Machine Learning and Cybernetics*, 2574-2579(2008).
10. Karp R M, Rabin M O. IBM Journal of Research and Development, **31**,249-260 (1987)
11. Schleimer S, Wilkerson D S, Aiken A. *2003 ACM SIGMOD International Conference on Management of Data*, 76-85( 2003)
12. Zaslavsky A, Bia A, Monostori K. *The 5th European Conference on Research and Advanced Technology for Digital Libraries*, 103-114(2001)
13. Sediyono A, Ku-Mahamud K R. *International Conference on Digital Information Management*, 253 - 259(2008).
14. Wikipedia. Jaccard index, https://en.wikipedia.org/wiki/Jaccard_index
15. Zou D, Long W J, Ling Z. *2010 International Conference on Internet Technology and Applications*, 1-5(2010)
16. Chuanbo Huang, Xiaoqin Hu, Xiaoxu Ma, et al. Journal of Sichuan University (Natural Science Edition), **49**,535-542(2012)