Indonesian Journal of Electrical Engineering and Computer Science
Vol. 5, No. 2, February 2017, pp. 462 ~ 471
DOI: 10.11591/ijeecs.v5.i2.pp462-471

Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winkler Distance Algorithms

Brinardi Leonardo[1], Seng Hansun*[2]
Universitas Multimedia Nusantara, Indonesia
Corresponding author, e-mail: brinardileonardo@gmail.com[1],, hansun@umn.ac.id*[2]

Abstract

Plagiarism is an act that is considered by the university as a fraud by taking someone ideas or writings without mentioning the references and claimed as his own. Plagiarism detection system is generally implement string matching algorithm in a text document to search for common words between documents. There are some algorithms used for string matching, two of them are Rabin-Karp and Jaro-Winkler Distance algorithms. Rabin-Karp algorithm is one of compatible algorithms to solve the problem of multiple string patterns, while, Jaro-Winkler Distance algorithm has advantages in terms of time. A plagiarism detection application is developed and tested on different types of documents, i.e. doc, docx, pdf and txt. From the experimental results, we obtained that both of these algorithms can be used to perform plagiarism detection of those documents, but in terms of their effectiveness, Rabin-Karp algorithm is much more effective and faster in the process of detecting the document with the size more than 1000 KB.

Keywords*: Jaro-Winkler Distance; plagiarism; Rabin-Karp; string matching*

1. Introduction

The development of technology over the years has always evolved significantly. The number of internet users growing and had a great influence over science and world views [1]. Google as a search engine provides easiness in searching documents or scientific sources [2]. In this case, the user is facilitated in the search for documents, either as a source of information or in making a reference to scientific papers. It is one of the positive impacts in the advancement of technology. But the more convenience given to users, there are certainly disruptive issues; one of them is plagiarism [3].

Plagiarism is an act of fraud in the form of copying an article without any credit given to the original source [4, 5]. Neville [6] in his book entitled "The Complete Guide to Referencing and Avoiding Plagiarism" defines plagiarism as the action or practice which is considered by the university as a fraud by taking someone ideas or writing without mentioning the references and claimed as his/her own. Therefore, writing a referral or citation and resources is absolute in order for a work is not said to be a plagiarism.

In a document there is a set of strings that are strung together into a single word or sentence. There are many kinds of algorithms that can be used in string matching and each algorithm has their own complexities. By using string matching technique, it can be compared across documents if there is an indication of plagiarism or not.

Similar studies related to this research include a thesis prepared by Kornain, et al. [7], Nugroho [8], and Faranika, et al. [9]. Of these three studies we concluded that Rabin-Karp and Jaro-Winkler Distance algorithms can be used to detect the similarity of documents, such as the detection of plagiarism in documents. According to Wicaksono, et al. [10] to create a plagiarism detection system, it is required a good algorithm for multiple types of string matching patterns. One algorithm that is suitable for the problem of multiple string matching patterns is Rabin-Karp algorithm. The advantage of the Rabin-Karp algorithm compared to other string matching algorithm is the ability to search for multiple string patterns [11].

Jaro-Winkler Distance algorithm is an algorithm that uses a string metric approach, which do a string comparison by put it in certain Mathematical functions. Some algorithms which based on the string metric include the Levenshtein distance, TF/ IDF, Needleman-Wunsch distance, Jaro-Winkler distance, and so on. From all algorithms that have been mentioned,

Received September 2, 2016; Revised December 21, 2016; Accepted January 1, 2017

Jaro-Winkler Distance has good accuracy in a relatively short string matching [12]. According to Kurniawati, et al. [12], this algorithm has a quadratic runtime complexity that is very effective on a short string and can work faster.

Based on other researches that had been done; we have a basic conclusion that Rabin-Karp and Jaro-Winkler Distance algorithms can be used to detect plagiarism. In addition, both of these algorithms have advantages for each other with the same time complexity on the preprocessing phase that is $O(m)$, despite the complexity of search phase are different. It becomes an inspiration in doing this research to compare the performance of both algorithms in text documents plagiarism detection.

2. Research Method

2.1. Plagiarism

Plagiarism is the practice of abuse of intellectual property rights belonging to another person and the work is recognized invalid as a result of personal work [13]. According to Sastroasmoro [14], a classification of plagiarism based on the proportion or percentage of words, sentences, or paragraphs hijacked, is divided into three, i.e. light plagiarism: <30%, middle plagiarism: 30% - 70%, and severe or total plagiarism: >70%. There are many factors that cause the occurrence of plagiarism acts. Empirical studies by Hutton and French in Hartanto [15] suggest that the factors which cause plagiarism are their laziness themselves, because they feel stress, have confidence that the behavior will not be known, and the behavior is not a wrong thing to do nor harmful.

As for the types of plagiarism by Iyer et al. [16], namely:

1. Word-for-word plagiarism, is copying each word directly, without change at all.
2. Plagiarism of authorship, is to recognize the work of others as the work himself.
3. Plagiarism of ideas, is to recognize the results of thoughts or ideas of others.

Plagiarism of sources, if one author using quotations from other writers without acknowledgment.

2.2. Algorithm

Algorithm is derived from the name of an Arab Mathematician, Abu Ja'far Muhammad ibn Musa al-Khuwarizmi. Al-Khuwarizmi read by Westerners as Algorism [17]. Gradually the word algorithm is used to refer to the method of calculation (computing) in general, there by losing its original meaning [18].

According Sjukani [19], the algorithm is the flow of thought in completing a job that is put in writing. The first emphasis is the train of thought, so that one's algorithm can be different from the others' algorithm. While the second emphasis is written, which means it can be a phrase, an image, or a specific table. So it can be concluded that the algorithm is more of a line of thought to complete a task or a problem than the manufacture of computer programs. With an algorithm, a problem can be solved with logical sequence of steps. The logical steps are a stage of the process which had been known certainly by any measures that have been created [20].

According to Knuth [17], logical steps can be categorized as a good algorithm if it has the following requirements:

1. Finiteness, algorithms must end after doing a number of process steps.
2. Definiteness, every step of the algorithm should be defined properly and not to cause double meaning.
3. Input, an algorithm has zero or more input that is given to the algorithm before it is executed.
4. Output, each algorithm provides one or more of the output.
5. Effectiveness, algorithm steps are done within a "reasonable" time.

A problem can have many algorithms settlement. The algorithm used must not only be true, but it must also be efficient. The efficiency of an algorithm can be measured from the time of the execution of the algorithm and the needs of memory space. A quantity used to describe the model of measurement of time and space is the complexity of the algorithm [21]. According to Goldreich [22] the complexity of algorithm can be measured based on its performance by calculating the execution time of an algorithm. The execution time can be classified into three major groups, namely best-case, average-case, and worst-case.

2.3. String Matching

String matching is a method used to find results of one or more given text pattern. String matching is an important subject matter in Computer Science because the text is the main form of information exchange between human beings, for example in the literature, scientific papers, and web pages. String matching is an algorithm to search all string occurrences by looking for similarities. The principle of string matching is to find all occurrences of short strings called pattern $P[0 \ldots\ldots n-1]$ in a longer string of text called $T[0 \ldots m-1]$, where m and n is the length of the string. The second string is formed of a limited character set called the alphabet, denoted as Σ with size σ [23].

String matching algorithms can be classified into three types according to the direction of the search, i.e. [24],

1. From left to right

From the direction of the most natural, i.e. left to right, which is the direction to read. Algorithms included in this category are the Brute Force algorithm, Knuth Morris Pratt algorithm, etc.

2. From right to left

From right to left direction, a direction that usually produces the best results practically. Algorithm included in this category is the Boyer-Moore algorithm.

3. In a specific order

From a specified direction determined by the algorithm, this direction theoretically produces the best results. One example algorithm in this category is Colossi Crochemore-Perrin.

2.3.1. Rabin-Karp

Rabin-Karp algorithm is a string matching algorithm that using a hash function to compare between the string searched (m) with a substring in a text (n). If both hash values are same, then the comparison will be made once again against the characters. If the results of both are not same, then the substring will be shifted to the right. Shifting is done as many as ($n-m$). An efficient hash value calculation at the time of the shift will affect the performance of this algorithm [25].

Rabin-Karp algorithm was created by Michael O. Rabin and Richard M. Karp in 1987 that uses a hash function to find a pattern in a text. According to Abdeen and Rawan [26], in principle the Rabin-Karp algorithm computes a hash function to search for a pattern in a given text.

Each M character, subsequence of the text, will be compared. If the hash values are not the same, the algorithm will compute the hash value for the next subsequence M characters, and if the hash values are same then the algorithm will perform brute-force comparison between the pattern and M character subsequence. This way there would be only one comparison per text subsequence and brute-force required only if the hash values match or equal [27].

In general the algorithm characteristics as follows [28],

1. Using a hashing function.
2. The preprocessing phase time complexity is $O(m)$.
3. The search phase complexity is $O(mn)$.
4. The time it takes $O(n + m)$.

The hash function applied to this algorithm should provide at least four properties, namely [29],

1. Capable of performing computation efficiently.
2. Has high string discrimination.
3. The hash function ($y\,[j+1 \ldots j+m]$) should be easily computed from:

$$hash\,(y\,[j \ldots j+m-1])$$
$$hash\,(y\,[j+m])$$

2.3.2. Jaro-Winkler Distance

Jaro-Winkler Distance algorithm is an algorithm for measuring the similarity between two strings and most of this algorithm is used in the field of duplication detection [7]. This algorithm began from Jaro Distance algorithm found by Matthew A. Jaro which later been developed by William E. Winkler and Thibaudeau by modifying the Jaro Distance to give higher

weights to prefix the resemblance. The higher the Jaro-Winkler Distance value for two strings indicates higher similarity of both strings. Normal value is 0 which indicates no similarity and 1 that indicates the existence of exact similarities [12].

The basis of this algorithm has three parts, namely [12],

1. Calculate the length of the string,
2. Find the same number of characters in the two strings,
3. Find the amount of transposition.

In general, Jaro-Winkler Distance algorithm's characteristics as follows [30],

1. Preprocessing phase time complexity is $O(m)$.
2. Search phase is quadratic $O(n^2)$.
3. Required time complexity is $O(m + n^2)$.

Jaro-Winkler Distance formula is used to calculate the distance (d_j) between the two strings $S1$ and $S2$.

$$d_j = \frac{1}{3} \times \left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m} \right) \quad (1)$$

Where:

m is the same number of character
$|S_1|$ is the length of String 1
$|S_2|$ is the length of String 2
t is the amount of Transposition

Jaro-Winkler (dw) using prefix scale (p) which provides a prefix on a set of strings, with the following formula,

$$d_w = d_j + \left(\mathrm{lp}(1 - d_j) \right) \quad (2)$$

Where:

d_j is the result of string similarity calculation of S_1 and S_2.
l is the length of character or same prefix on string prefix before we found the existence of inequality with a maximum of up to four characters.
p is a constant scaling factor. The default value for the constant according to Winkler is $p = 0.1$.

2.4. System Design

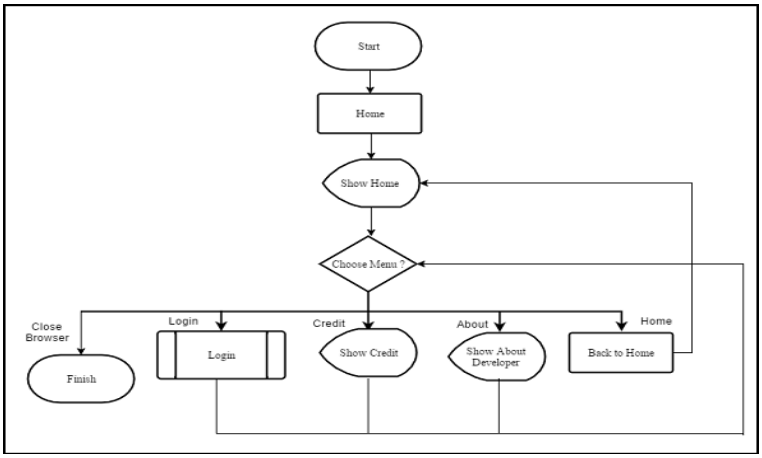

Figure 1. Main Flowchart of the System

To illustrate the process of the system built, we use the Flowchart diagram. Figure 1 shows the main flowchart in the plagiarism detection system. When the application is started,

the user will be directed to the Home page. After that, the user can specify a choice of several menu, such as About, Credit, and Login. Each option of the menu will display a page that is vary according to the function of the option's menu. To be able to use the system, users must log in first. However, if the user does not have any id to login, the user will be redirected to registering themselves first. After the user can login, the user will be redirected to the Home page with the navigation has changed.

Figure 2 describes the sub processes of the algorithms implemented on the system, i.e. the Rabin-Karp algorithm. This algorithm runs the process of checking the similarity first to seek the same hashing value between the characters. Then return to the checking process by comparing the string on the characters with the same hashing value.

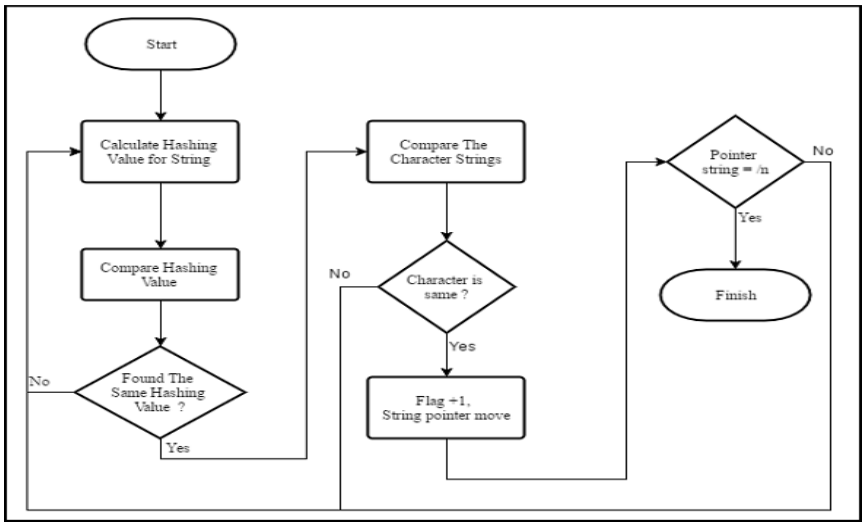

Figure 2. Rabin-Karp Algorithm

The Rabin-Karp algorithm implementation on the system is shown on the below program listing. It was written in PHP language. At the earliest stage, we initialized preparation before the algorithm is executed. The preparations include the retrieval of the contents of the documents in which have been made into text, then the data is purged from the special chars. After that the document will be processed in the form of an array.

```
function RabinKarp($Key, $Text){
$TextLength = strlen($Text);
$KeyLength = strlen($Key);
$tempSplitData = explode(" ",$Text);
$tempArrayData = array_filter($tempSplitData);
for($i=0;$i<count($tempSplitData);$i++){
  if(empty($tempArrayData[$i])){}
  else{
      if(Convert($tempArrayData[$i])==Convert($Key)){
  echo "<br/>".$tempArrayData[$i]." = ".Convert($tempArrayData[$i])."<br/> Same with
Key <br/>".$Key." = ".Convert($Key)."<br/>";
          if(CheckEqual($Key,$tempArrayData[$i])){
            $GLOBALS['temp'][$Key]+=1;
            $i++;
          }
          else continue;
      }
  }
}
}
```

On the Rabin-Karp function, documents stored in an array goes through the Convert() function. On this process, we would look for the hash values. Hash value is found by using primes power. Once the document is completed through the Convert() process, then it will be checked between the array and the hash values. Each hash value and array that has double similarity shall be deemed to be only one similarity.

Figure 3 describes the sub processes of the Jaro-Winkler Distance algorithm. The algorithm runs the checking process by using a string similarity metric approach that is incorporated into a Mathematical function.

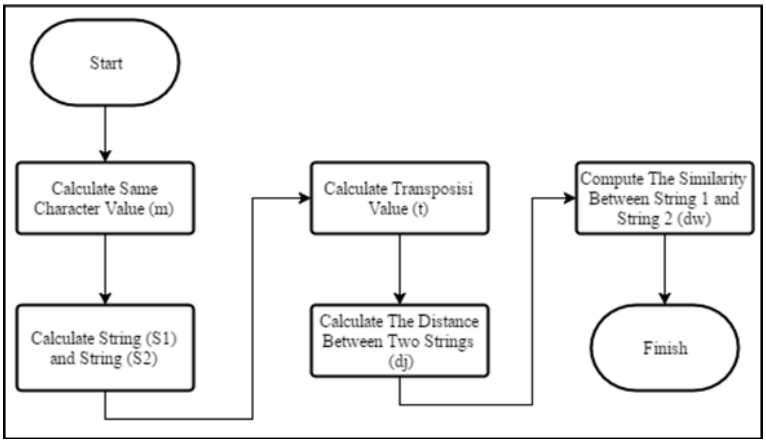

Figure 3. Jaro-Winkler Distance Algorithm

The Jaro-Winkler Distance algorithm implementation on the system is shown on the below program listing, which was written in PHP language. At this stage there are some function like Jaro() and getPrefixLength() functions to be run sequentially to obtain the required values. The first stage to run Jaro-Winkler Distance is to run Jaro-Winkler algorithm first, which demonstrated by Jaro() function. At the final stage, the values obtained from the implementation of Jaro-Winkler algorithm and other processes will be reprocessed simultaneously and produce a final value (d_w) of Jaro-Winkler Distance algorithm. The final value will determine the percentage similarity of documents that have been analyzed.

```
function JaroWinkler($string1,$string2,$PREFIXSCALE=0.1){
$JaroDistance = Jaro($string1,$string2);
$prefixLength = getPrefixLength($string1,$string2);
   $dw = $JaroDistance + ($prefixLength * $PREFIXSCALE * (1.0 - $JaroDistance));
Return $dw;
}
```

3. Results and Analysis

3.1. Implementation Results

Figure 4 shows the user interface of Member page of the system. On this page, three simple steps on how to use this system are given. First, the user could upload the documents to be analyzed to the system. Next, the user could choose which document to be analyzed from a set of documents that had been uploaded on the first step. Lastly, the user could get the comparison results of the chosen documents that had been processed using Rabin-Karp and Jaro-Winkler Distance.

Figure 5 shows the user interface of Results page. On this page, the documents to be analyzed using Rabin-Karp and Jaro-Winkler Distance algorithms are shown in tabular form. On the table, the name of the analyzed document and compared document are shown with the similarity percentage value of each algorithm implemented on the system.

Figure 4. Main Interface of the System

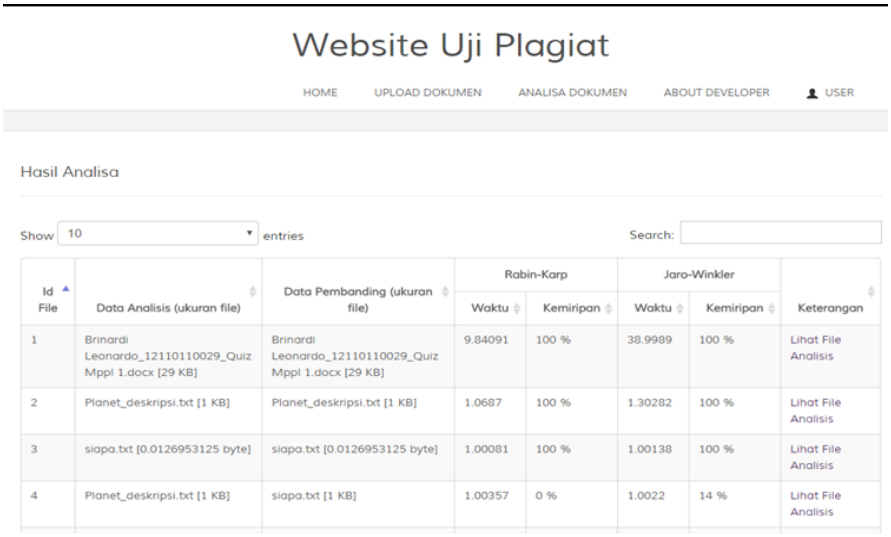

Figure 5. Results Page

Figure 6. Show Highlight Page

The user could see the text comparison results by opening the Show Highlight page. On this page, every similar word will be highlighted with yellow color between the two documents being analyzed. On this page, the information on how many similar words, number of all words found in the document, and the similarity percentage are also given to the user. Figure 6 shows the Show Highlight page of the system.

3.2. Analysis

After the implementation phase of the application had been completed, the next phase of testing is done through samples collected. The sample collection of data is collected randomly from private property and through Google search engine which had been modified. Data modifications made to run a scenario that will be done in testing. Roscoe in Sekaran [31] provides a common reference to determine the sample size of 30 samples data. Scenarios to be implemented, namely comparing data sample documents that are small in size (1 KB - 1000 KB) and large in size (> 1000 KB). It aims to obtain a conclusion regarding the performance comparison between the two algorithms that have been implemented on the application. Testing is done offline using the sample data collected.

From the experimental results, we obtained an average rating for each scenario as follows:

1. In the testing on text documents (.txt) with size <1000 KB, data been tested is 30 documents and we did not find any failure in finding the similarities. The average value of similarity is 52% on 0.118 minutes for Rabin-Karp algorithm and 45% on 0.228 minutes for Jaro-Winkler Distance algorithm.
2. In the testing on .doc or .docx documents with size <1000 KB, data been tested is 30 documents and we found six documents that had failed in finding the similarities using Jaro-Winkler Distance algorithm. So the average value of similarity is 59% on 0.724 minutes for Rabin-Karp algorithm and 42% on 1.156 minutes for Jaro-Winkler Distance algorithm.
3. In the testing on .pdf documents with size <1000 KB, data been tested is 30 documents and we found three documents that had failed in finding the similarities using Jaro-Winkler Distance algorithm. The average value of similarity is 55% on 0.773 minutes for Rabin-Karp algorithm and 45% on 2.044 minutes for Jaro-Winkler Distance algorithm.
4. In the testing on the documents with a size >1000 KB, data been tested is 30 documents and we found eight documents that had failed in finding the similarities with Jaro-Winkler Distance algorithm. The average value of similarity is 45% on 0.790 minutes for Rabin-Karp algorithm and 39% on 1.676 minutes for Jaro-Winkler Distance algorithm.
5. In the testing on different types of documents, data been tested as many as 30 documents and we found five documents that had failed in finding the similarities using Jaro-Winkler Distance algorithm. The average value of similarity is 47% on 0.341 minutes for Rabin-Karp algorithm and 37% on 0.670 minutes for Jaro-Winkler Distance algorithm.

From the exposure before, we obtained a conclusion that the Rabin-Karp Algorithm and Jaro-Winkler Distance managed to perform plagiarism detection of the document. From the scenario that has been designed, the two algorithms had different performance and processing time. In general both algorithms can perform detection although the size of the data being analyzed and compared being the same or different. Not only that, the Rabin-Karp algorithm and Jaro-Winkler Distance algorithm also have managed to perform detection of the document with the different file format.

4. Conclusion

In this research, the Rabin-Karp and Jaro-Winkler Distance algorithms had been successfully implemented on a web-based system. Based on the analysis of experimental results, both algorithms have their respective advantages. But in general it turns that Rabin-Karp algorithm is more effective than the Jaro-Winkler Distance algorithm, because on some trial this algorithm is more likely to get the higher percentage resemblance on the document being tested than Jaro-Winkler Distance. The scenario of Rabin-Karp algorithm has an average value of similarity percentage of 51% and 35% for Jaro-Winkler Distance algorithm. This is because the Jaro-Winkler Distance algorithm only able to examine the similarity of identical or similar documents only. In terms of processing time, Rabin-Karp algorithm has an average of 0.594 minutes while Jaro-Winkler Distance algorithm has an average of 0.992 minutes.

However, for some documents that have a large size, Rabin-Karp algorithm is much faster than Jaro-Winkler Distance algorithm.

It can be concluded that the two algorithms can perform the plagiarism detection of documents, but the performance of Rabin-Karp algorithm is much more effective than Jaro-Winkler Distance algorithm. In terms of processing speed, Rabin-Karp algorithm is faster than Jaro-Winkler Distance algorithm with the lapse of time for all the scenarios is about 0.389 minutes.

References

[1] Darma Jarot S, Ananda S. Buku Pintar Menguasai Internet. Jakarta: PT TransMedia. 2009.
[2] Sriyati T. Perkembangan Internet dan Strategi Pemanfaatannya di Perpustakaan Badan Penelitian dan Pengembangan Kehutanan. *Visi Pustaka.* 2009; 11(2): 20-24.
[3] Ulum S. Analisis Plagiarisme Penulisan Skripsi Mahasiswa Lulusan Tahun 2010 Jurusan Akuntansi Perguruan Tinggi X di Kota Malang. Thesis. Malang: Universitas Negeri Malang; 2014.
[4] Ezzikouri H, Erritali M, Oukessou M. Semantic Similarity/Relatedness for Cross Language Plagiarism Detection. *Indonesian Journal of Electrical Engineering and Computer Science*. 2016; 1(2): 371-374.
[5] Alfikri ZF, Purwarianti A. Detailed Analysis of Extrinsic Plagiarism Detection System using Machine Learning Approach (Naïve Bayes and SVM). *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(11): 7794-7804.
[6] Neville C. The Complete Guide to Referencing and Avoid Plagiarism. Berkshire: Open University Press. 2010.
[7] Kornain A, Yansen F, Tinaliah. Penerapan Algoritma Jaro-Winkler Distance untuk sistem Pendeteksi Plagiarisme pada Dokumen Teks Berbahasa Indonesia. Research Paper. [online] http://eprints.mdp.ac.id/1068/.
[8] Nugroho E. Perancangan Sistem Deteksi Plagiarisme Dokumen Teks dengan Menggunakan Algoritma Rabin-Karp. Thesis. Malang: Universitas Brawijaya; 2011.
[9] Faranika Y, Kurniawan H, Nikentari N. Sistem Pengukur Kemiripan Dokumen Menggunakan Algoritma Jaro-Winkler Distance. *E-Jurnal Universitas Maritim Raja Ali Haji*. 2013: 1-8.
[10] Wicaksono YA, Suyanto. Analisis dan Implementasi Algoritma Rabin-Karp dan Algoritma Stemming Nazief-Adriani pada Sistem Pendeteksi Plagiat Dokumen Teks Berbahasa Indonesia. Thesis. Bandung: Universitas Telkom; 2012.
[11] Cormen Leiserson TH, Rivest CE, Stein RL, Clifford. Introduction to Algorithm. USA: Massachusetts Institute of Technology. 2003.
[12] Kurniawati A, Puspitodjati S, Rahman S. *Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia*. Proceedings of Seminar Ilmiah Nasional KOMMIT 2010. Bali. 2010: 1-4.
[13] Sulianta F. Seri Referensi Praktis: Konten Internet. Jakarta: PT Elex Media Komputindo. 2007.
[14] Sastroasmoro S. Beberapa Catatan tentang Plagiarisme. *Majalah Kedokteran Indonesia*. 2007; 57(8): 239-244.
[15] Hartanto D. Menyontek: Mengungkap Akar Masalah dan Solusinya. Jakarta: Indeks. 2012.
[16] Iyer P Singh, Abhipsita. *Document Similarity Analysis for a Plagiarism Detection System*. Proceedings of 2nd Indian International Conference on Artificial Intelegence (IICAI-05). Pune. 2005: 2534-2544.
[17] Knuth DE. The Art of Computer Programming. Volume 1. USA: Addison-Wesley Company, Inc. 1973.
[18] Parsons TW. Introduction to Algorithms in Pascal. New York: Johns Wiley and Sons. 1995.
[19] Sjukani M. Algoritma dan Struktur Data dengan C, C++, dan Java. Jakarta: Mitra Wacana Media. 2005.
[20] Microsoft Press. Microsoft Press Computer Dictionary. 3rd Edition. USA: Microsoft Press. 1997.
[21] Azizah UN. Perbandingan Detektor Tepi Prewitt dan Detektor Tepi Laplacian Berdasarkan Kompleksitas Waktu dan Citra Hasil. Thesis. Bandung: Universitas Pendidikan Indonesia; 2013.
[22] Goldreich O. Computational Complexity: A Conceptual Perspective. New York: Cambridge University Press. 2008.
[23] Breslauer D. Efficient String Algorithmics. PhD Thesis. Columbia: CS Department; 1992.
[24] Charras C, Lecroq T. Handbook of Exact String Matching Algorithm. Oxford: University Press. 2001.
[25] Firdaus HB. Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp. *Jurnal Ilmu Komputer dan Teknologi Informasi*. 2003; 3(2): 1-5.
[26] Abdeen A Rawan. An Algorithm for String Searching Based on Brute-Force Algorithm. *International Journal of Computer Science and Network Security*. 2011; 11(7): 24-27.
[27] Jain S Rao, Nersimha AL, Agarwal P. A Relative Study of Pattern Matching Algorithms. *Journal of Computing Technologies*. 2012; 1(2).
[28] Fernando H. Perbandingan dan Pengujian Beberapa Algoritma Pencocokan String. Makalah IF2251

Strategi Algoritmik. [online] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2009-2010/Makalah2009/MakalahIF3051-2009-006.pdf.
[29] Novian D, Abdillah T, Tuloli MS, Yassin RMT. *Aplikasi Pendeteksi Plagiat pada Karya Ilmiah Menggunakan Algoritma Rabin-Karp*. Laporan Penelitian Pengembangan Fakultas dan Keilmuan Dana BOPTN Tahun Anggaran 2012, Jurusan Teknik Informatika, Universitas Negeri Gorontalo, Gorontalo. 2012.
[30] Wirawan T. Pencocokan String Menggunakan Algoritma Jaro-Winkler Distance. *Jurnal Ilmu Komputer dan Teknologi Informasi*. 2004; 3.
[31] Sekaran U. Metode Penelitian Bisnis. Jakarta: Salemba Empat. 2006.