International Journal of Innovative Computing, Information and Control
Volume **14**, Number **5**, October **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 1893–1903

INDONESIAN TEXT DOCUMENT SIMILARITY DETECTION SYSTEM USING RABIN-KARP AND CONFIX-STRIPPING ALGORITHMS

Deardo Dibrianto Sinaga and Seng Hansun

Computer Science Department
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang, Banten 15811, Indonesia
simadanaga@gmail.com; hansun@umn.ac.id

Received January 2018; revised May 2018

Abstract. *Nowadays, negative impact, such as plagiarism, may arise along with faster and easier ways in finding information. There are many software and websites that can be used to check the occurrence of plagiarism, but unfortunately, they are not really suitable for scientific papers which are written in Bahasa Indonesia because it is designed for text in English. Therefore, a document similarity detection system that is more suitable for papers written in Bahasa Indonesia is needed. Rabin-Karp is an algorithm that can be used in checking the similarity between documents, while Confix-Stripping is an algorithm that can perform basic word search in Bahasa Indonesia. This research has successfully implemented Rabin-Karp and Confix-Stripping algorithms very well. Tests performed with various document scenarios as well as algorithms have given some performance results of the system in terms of time and similarity level. The system with the pure Rabin-Karp can provide the best system performance, both in terms of time and accuracy with an average total processing time speed of 0.0123 second and the average similarity rate of 89.1967%. The accuracy level given by the system is 0.7. The system that has been added with a stemming process or N-Gram can also improve some test results in terms of processing time and similarity level.*

Keywords: Rabin-Karp, Confix-Stripping, Similarity, Plagiarism, Text document

1. Introduction.

Along with the development of technology and information so rapidly, the search for information becomes easier and faster. It has a positive impact for the various levels of the society in broadening their insights and knowledge. On the other hand, the ease in the search for information also has negative impacts for people who misuse it. The misuse of the ease in the search for information often occurs in terms of education, especially in doing daily tasks or scientific work done by students, such as plagiarizing the work of others without citing the source. It is widely known to the public as plagiarism.

According to Ministry of Education of the Republic of Indonesia [1], plagiarism is defined as "the act of intentionally or unintentionally obtaining or attempting to obtain credit or value for a scientific work, by quoting part or all of the work and/or scientific work of other people and recognized it as their own scientific work, without declaring the source appropriately and adequately". In addition, plagiarism is also described as the act of theft of other people's work and makes the work as if it was his own work [2]. Plagiarism according to Tudesman et al. [3] can be overcome by prevention and detection acts. Prevention means keeping or preventing plagiarism from being done. Prevention should be done as early as possible, especially in the educational and moral systems of the

DOI: 10.24507/ijicic.14.05.1893

community, while detection means trying to find the plagiarism action that has been done. Therefore, a plagiarism detection system is required in a computerized text document. There has been many software and websites that can be used to check plagiarism on text documents, but the software and websites are less suitable for papers or scientific papers written in Bahasa Indonesia as they are designed for English text [4]. Therefore, there is a need for a system to detect similarities in text documents which is more suitable for writing structures in Bahasa Indonesia.

In the next section, we will describe some related works and studies that become the background of this study. Then some basic theories such as plagiarism, Rabin-Karp, Confix-Stripping stemming algorithm, and N-Gram method will be discussed in Sections 3 to 6 respectively. The main results of this study then will be explained on the next section, and closed by a conclusion at the end of this paper.

2. Related Works.

This research uses the Confix-Stripping stemming algorithm which is the development of the Nazief-Adriani stemming algorithm. This study refers to the scientific paper written by Wicaksono et al. [5] and Lede et al. [6] who had successfully implemented the Rabin-Karp algorithm as well as the Nazief-Adriani stemming algorithm to detect similarities in Bahasa Indonesia text documents. Based on these studies, this research develops further using Confix-Stripping stemming algorithm. Confix-Stripping stemming algorithm itself is a modification algorithm of the Nazief-Adriani stemming algorithm performed by Asian et al. [7]. Stemming is a way to get the basic word from a word that is affixed. The results of the research conducted by Asian et al. [7] state that the Nazief-Adriani algorithm has a stemming error rate under 1 of 21 words. After modifying the algorithm using the Confix-Stripping algorithm, the stemming error rate becomes below 1 of 38 words. Thus, the use of Confix-Stripping algorithm is proven to be better in performing stemming process for Bahasa Indonesia text.

This research will apply N-Gram method which will be inserted in Rabin-Karp process to increasing the similarity value of document testing. The hashing technique used in this research is Rolling Hash, while the formula used to measure the similarity level of two documents after the Rabin-Karp process performed is Dice's Similarity Coefficient. In this research, the accuracy of the system built will also be measured using Confusion Matrix.

3. Plagiarism.

Kurniawati and Wicaksana [8] explained that: "Plagiarism is an act of abuse, theft or robbery, publication, declaration, or declaring others own thoughts, ideas, writings, or creations as the property of his own". In addition, plagiarism is also described as acts of theft of other people's work and makes the work as if it was his own work [2].

According to Parvati and Abhipsita in Lede et al. [6], plagiarism is grouped into six forms, namely:

1) Word-of-word plagiarism; copying sentences directly from a text document without quoting or permissions.
2) Authorship plagiarism; the acknowledgment of the work of others as self-possession by including the name of oneself which replaces the name of the actual author.
3) Idea plagiarism; reusing an original idea or thought of a text source regardless of the form of the source text.
4) Secondary source plagiarism; citing original source which is obtained from a secondary source by ignoring the original text from the actual source.
5) Plagiarism of the source structure; copying or plagiarizing the structure of an argument from a source.

6) Plagiarism paraphrase; rewriting a text by changing the word or syntax, but the original text is still recognizable.

Based on the proportions and degree of similarity of documents, plagiarism is classified as follows [9].

1) Light plagiarism, if the similarity level is below 30 percent ($< 30\%$).
2) Moderate plagiarism, if the similarity level is between 30 to 70 percent.
3) Severe plagiarism, if the similarity level is above 70 percent ($> 70\%$).

4. Rabin-Karp.

Rabin-Karp algorithm is a string matching algorithm found by Michael Rabin and Richard Karp. This algorithm compares the hash value between the input string and the substring in the text, if the hash value is the same, then the comparison will be done once again to the characters, but if the hash value is not the same, the substring will shift to the right. The efficient calculations of hash value during a shift will affect the performance of this algorithm [10].

The following figures illustrate the process of the Rabin-Karp algorithm derived from [10]. Given "cab" as an input and "aabbcaba" as a text. The hash function used for example will add the value of each letter in alphabet (a = 1, b = 2, etc.) and then modulo with 3. The "cab" hash value will be 0 and the first three characters in the text, which is "aab" will be 1.

Figure 1 shows the comparison results are not the same, and then the substring on the text will shift one character to the right. The algorithm will not recalculate the substring hash value. Here is what the so-called rolling hash does, which is reducing the value of the outgoing character and add the value of the incoming character so that there will be a relatively constant time complexity at each shift [10].

After the shift, the hash value of the fingerprint "abb" (abb = aab − a + b) becomes two ($2 = 1 - 1 + 2$) as can be seen in Figure 2.

Figure 3 shows the results of comparison are also not the same, so that shifting should be done. Similarly, the third comparison should be done the same. In the fourth comparison, the same hash value is obtained.

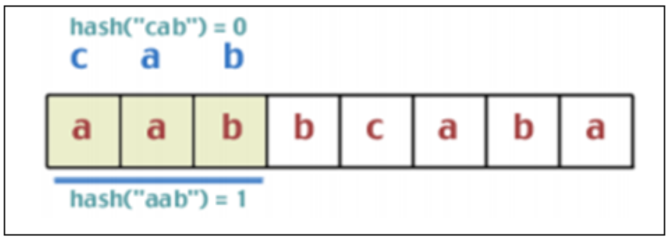


FIGURE 1. Initial fingerprint [10]

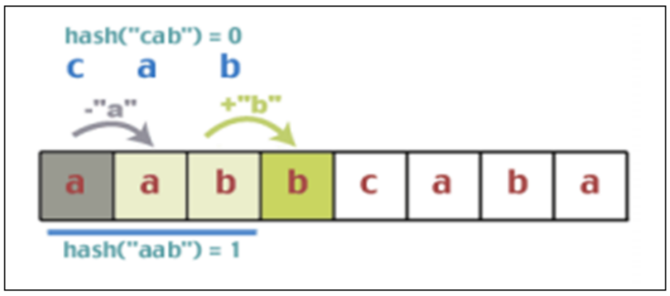


FIGURE 2. Shifting fingerprint [10]

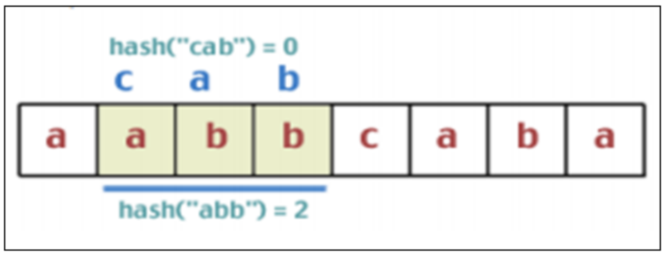


FIGURE 3. Second comparison [10]

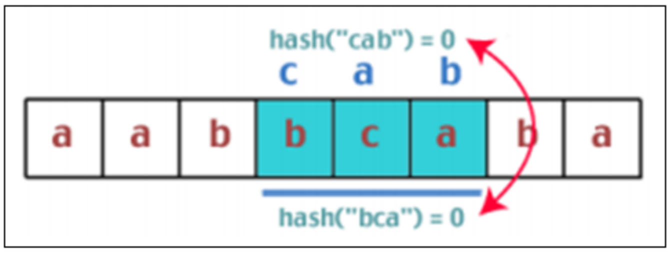


FIGURE 4. Fourth comparison (same Hash value) [10]

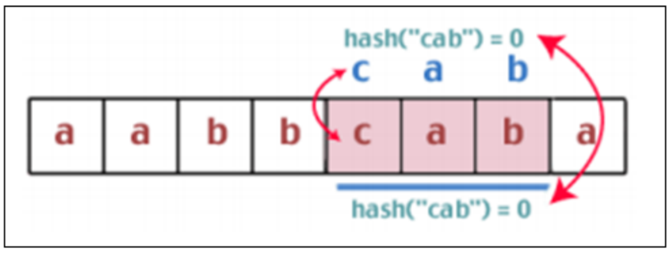


FIGURE 5. Fifth comparison (*String* found) [10]

Since the hash values are the same, then a string of characters per character is performed between "bca" and "cab". The result is that both strings are not the same. Again, substring shifted to the right that can be seen in Figure 4.

In the fifth comparison, both hash values and string-forming characters are appropriate, which means the solutions are found. From the calculation, the time complexity required is O(m + n) where m is the length of the input string and n is the number of iterations performed to find the solution. This result, shown in Figure 5, is much better than the time complexity obtained using a brute-force algorithm that is O(mn) [10].

5. **Confix-Stripping.**

Based on Adriani et al. in Kurniawan et al. [11], Confix-Stripping stemming algorithm is an algorithm that is used to perform stemming process to the affixed words. The Confix-Stripping stemming algorithm has an affixed rule with the following model.

$$[[[AW+]AW+]AW+] \text{ Kata Dasar } [+AK][+KK][+P] \tag{1}$$

where

AW: Awalan (Prefix)
AK: Akhiran (Suffix)
KK: Kata ganti kepunyaan (Possessive Pronoun)
P: Partikel (Particle)
Kata Dasar: Basic Word

The steps of the Confix-Stripping stemming algorithm are as follows [11].

1) The word that has not been stemmed is compared to the basic word dictionary database. If it is found, then the word is assumed as the basic word and the algorithm stops. If the word does not exist then continue to step 2).
2) If the input word has a prefix-suffix "be-lah", "be-an", "me-i", "di-i", "pe-i", or "te-i" then the next stemming step is 5, 3, 4, 5, 6, but if the inputed word has no such prefix-suffix pair, the stemming step runs normally, where the steps will be 3, 4, 5, 6.
3) Eliminate the particles and possessive pronouns. First, remove the particles ("-lah", "-kah", "-tah", "-pun"). After that remove also the possessive pronouns ("-ku", "-mu", or "-nya"). For example: for the word "bajumulah", the first stemming process becomes "bajumu" and the second stemming process becomes "baju". If the word "baju" is in the dictionary then the algorithm stops. In accordance with the affixed model, it becomes:

$$[[[AW+]AW+]AW+] \text{ Kata Dasar } [+AK] \tag{2}$$

4) Eliminating also the suffix ("-i", "-an", and "-kan"), according to the affixed model, then it becomes:

$$[[[AW+]AW+]AW+] \text{ Kata Dasar} \tag{3}$$

5) The elimination of prefix ("be-", "di-", "ke-", "me-", "pe-", "se", and "te-") will follow these following steps.
   a. The algorithm will stop if:
      i. The prefix identified as the form of a pair of unadjusted affixes with the suffix (based on Table 1) removed in step 3).
      ii. Identified that the prefix is now identical to the prefix that has been deleted before.
      iii. The word already has no prefix.
   b. Identify the prefix type and its decay if necessary. The prefix type is specified by the following rules:
      i. If the prefix of the word is "di-", "ke-", or "se" then the prefix can be immediately omitted.
      ii. Delete the "te-", "be-", "me-", or "pe-" prefixes that use the decay rule described in Table 2. For example, for the word "menangkap", after removing the "me-" prefix, then the word obtained is "nangkap". Since the word "nangkap" is not found in the database, then the character "n" is replaced with character "t" so that the word "tangkap" is obtained. The word "tangkap" is corresponding with the basic word database, and then the algorithm will stop.
6) If all the steps fail, then the word tested on this algorithm is considered as the basic word.

TABLE 1. Prefix and suffix combinations that are not allowed

Prefix	**Suffix**
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

Table 3 shows the modification or additional decay rules for basic words of Table 2 for the Confix-Stripping stemming algorithm obtained from [12].

Table 2. Basic words' decay rules

Rules	**Prefix**	**Decay**
1	berV...	ber-V... | be-rV...
2	berCAP...	berCAP... where C! = "r" & P! = "er"
3	berCAerV...	ber-CaerV... where C! = "r"
4	belajar...	bel-ajar
5	beC_1erC_2...	be-C_1erC_2... where C_1! = {'r'|'l'}
6	terV...	ter-V... |te-rV...
7	terCerV...	ter-CerV... where C! = 'r'
8	terCP	ter-CP... where C! = "r" and P! = "er"
9	teC_1erC_2	te-C_1erC_2... where C_1! = 'r'
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe{r|l}...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... |me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... |me-tV...
16	meng{g|h|q}...	meng-{g|h|q}...
17	mengV...	meng-V... |meng-kV...
18	menyV...	meny-sV...
19	mempV...	mem-pV... where V! = 'e'
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... |pe-rV...
22	perCAP	per-CAP... where C! = "r" and P! = "er"
23	perCAerV	per-CAerV... where C!= "r"
24	pem{b|f|V}...	pem-{b|f|V}...
25	pem{rV|V}...	pe-m{rV|V}... |pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... |pe-tV...
28	peng{g|h|q}	peng-{g|h|q}...
29	pengV	peng-V... |peng-kV...
30	penyV...	peny-sV...
31	pelV...	pe-lV...; except the word "pelajar" will become "ajar"
32	peCP	pe-CP... where C! = {r|w|y|l|m|n} and P! = 'er'
33	perCerV	Per-CerV... where C!={r|w|y|l|m|n}

Here C: Consonants; V: Vowels; A: Vowels or consonants; P: Particle of fragment from a word, for example "er"

Table 3. Modification and additional rule of Table 2

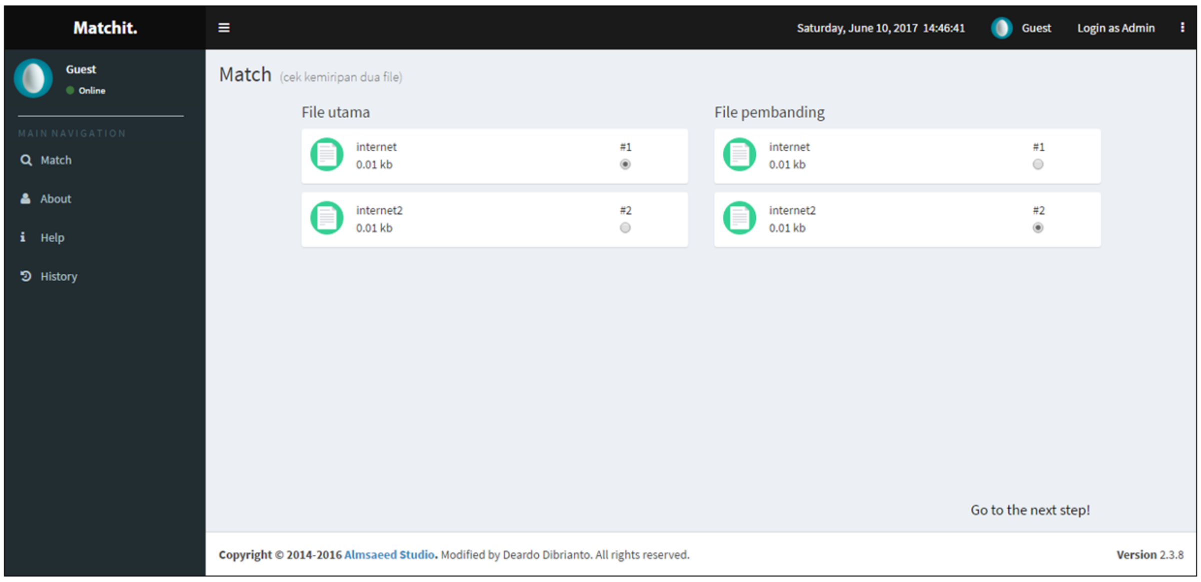
Rules	**Word Formats**	**Fragmentation**
12	mempe...	mem-pe...
16	meng{g|h|q|k}...	meng-{g|h|q|k}...
34	terC1erC2...	ter-C1erC2... where C1! = "r"
35	peC1erC2...	pe-C1erC2... where C1! = {r|w|y|l|m|n}

6. **N-Gram.** N-Gram is a way to get N sliced character of a sentence based on the number of N specified [13]. An example of using N-Gram can be seen in Table 4 with the example of the phrase "buku saya", where "_" is represented as a white space.

TABLE 4. Example of using N-Gram in sentence

Gram	Result
Uni-Gram	b, u, k, u, _, s, a, y, a
Bi-Gram	bu, uk, ku, u_, _s, sa, ay, ya
Tri-Gram	buk, uku, ku_, u_s, _sa, say, aya
Quad-Gram	buku, uku_, ku_s, u_sa, _say, saya

7. **Main Results.** Figure 6 shows the home page of the main features in the application. "Match Page" is functionally used to select two documents to calculate the similarity level. After the user selects the two documents that he wants to calculate the level of similarity, then the user only needs to continue the process by pressing the "Go to the next step" button.



Copyright © 2014-2016 Almsaeed Studio. Modified by Deardo Dibrianto. All rights reserved.

FIGURE 6. Match page result

The tests conducted in this research focus on the comparison of results to the system, both in terms of time and similarity level. The amount of the original document used in this research is three documents obtained from the journal of Putra and Susetyo [14] with art content, the journal of Munir et al. [15] with medical content, and the journal of Apsari [16] with social content. Those three documents are modified and cited from Haryadi [17]. The deletion of tables, images, abstracts, and bibliography is also done in the document modification process. The result of the modification for all documents can be viewed by accessing the following link: https://tinyurl.com/yccjh9bf.

The trial process involves the previously discussed document scenarios as well as the algorithm scenarios. The algorithm scenarios done in this research consist of four types, which are system with Rabin-Karp algorithm hereinafter referred to as type 1, Rabin-Karp with N-Gram as type 2, Rabin-Karp plus Confix-Stripping algorithm as type 3, and Rabin-Karp with N-Gram added with Confix-Stripping as type 4. The default value of N-Gram in this research is 5, while the base value used is 11. The basic dictionary database used in this research amount to 28,526 words obtained from Adikara [18].

Figure 7 shows a graph of the average result of the average speed of processing time of the Rabin-Karp algorithm based on the tests performed on documents A, B, and C. The results shown in the graph indicate that the fastest processing time position of the Rabin-Karp algorithm is presented in the algorithm scenario type 3 with an average time of 0.0103 second. Algorithm scenario type 1 is on the second position with an average time of 0.0123 second. In the third position, there is algorithm type 4 with an average time of 0.0776 second, while the fourth is algorithm scenario type 2 with an average time of 0.0909 second.

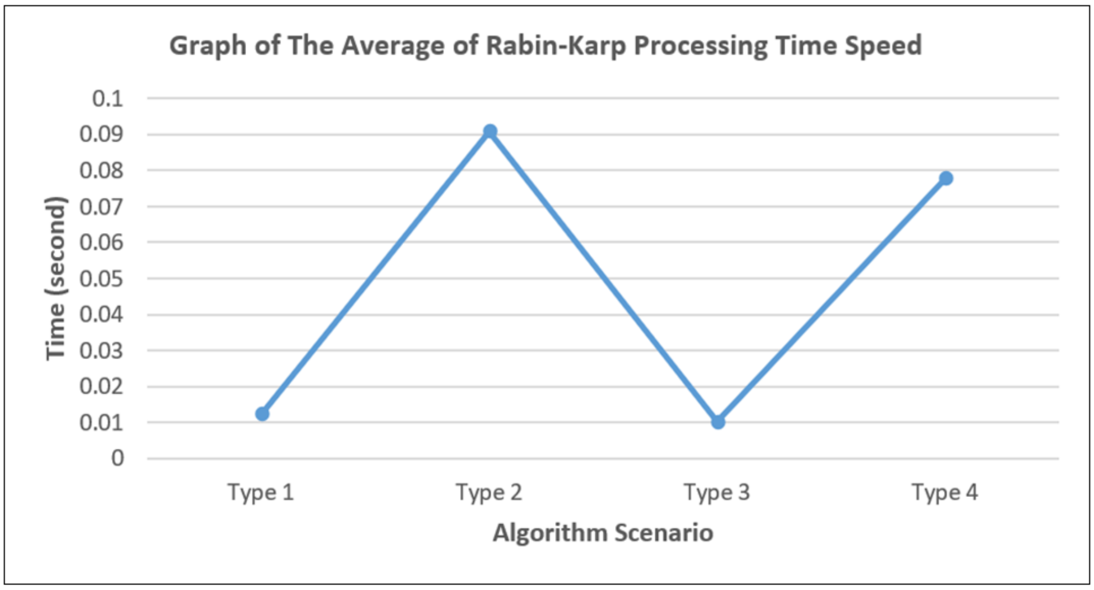


Figure 7. Graph of the average of Rabin-Karp processing time speed

Type 3 algorithm scenario produces the fastest Rabin-Karp processing time due to the fact that the Rabin-Karp algorithm in that scenario does not require additional N-Gram processing time in it, besides before the calculation process with Rabin-Karp algorithm is performed, the system will do the stemming process first to the words in the document that speeds up the final calculation in the Rabin-Karp algorithm.

Figure 8 shows a graph of the average result of the average total processing time based on the tests performed on documents A, B, and C. The graph results show the order of the total processing time, starts with algorithm scenario type 1 on the first position with an average processing time 0.0123 second. The second position placed by algorithm scenario type 2 with an average processing time of 0.0909 second. Algorithm scenario type 3 is on the third position with an average processing time of 44.4580 second, while the fourth position is algorithm scenario type 4 with an average processing time of 44.4919 second. System with the algorithm type 1 results in the fastest average total processing time because the system does not require additional time to run the N-Gram or Confix-Stripping process.

Figure 9 shows a graph showing the average results of the average similarity level based on the tests performed on the documents A, B, and C. The graph results show the order of similarity level, starts with algorithm scenario type 1 and type 3 on the first position with the same average similarity level of 89.1967%. For the algorithm scenario with N-Gram, the scenario algorithm type 2 results in a higher average similarity level of 89.1313% rather than the scenario algorithm type 4 with an average similarity level of 89.1290%.

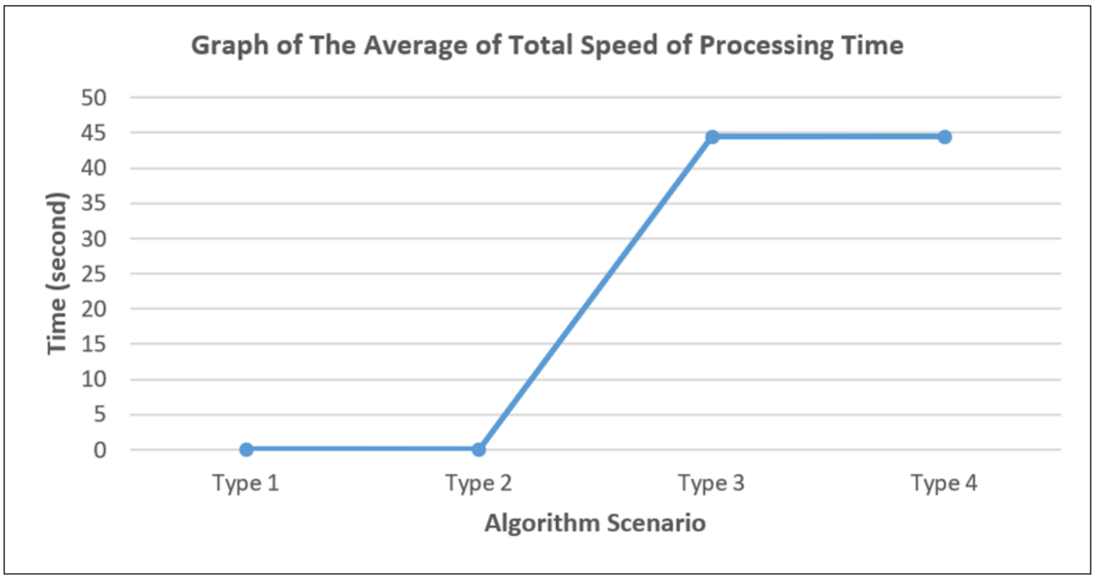


Figure 8. Graph of the average of total speed of processing time

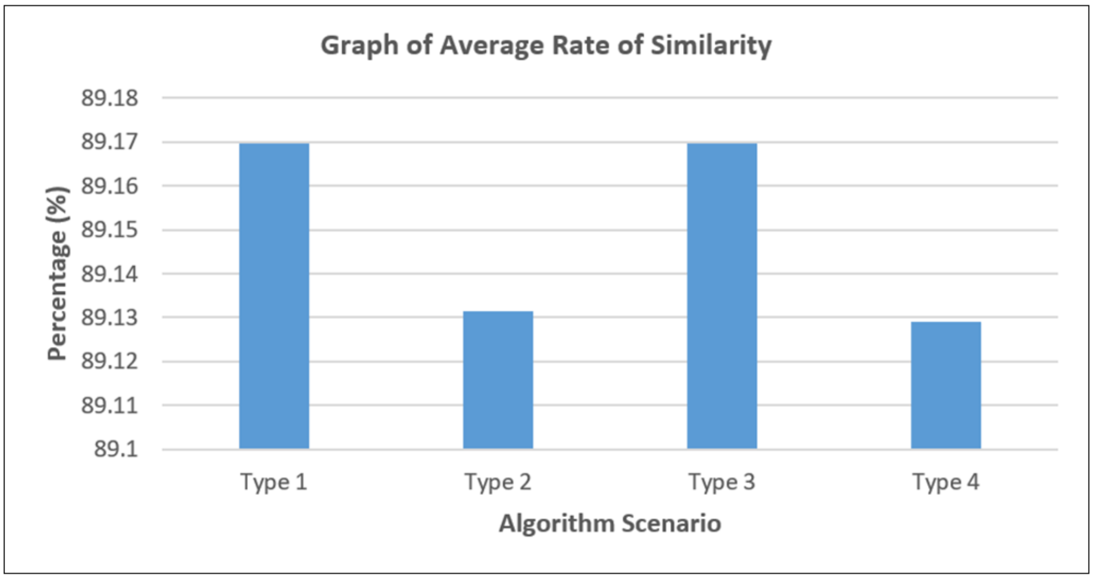


Figure 9. Graph of average rate of similarity

This research also performed calculations of the system accuracy in terms of similarity level, and the calculation of accuracy is done by using the results obtained from the system that has been made and the results obtained from using another similarity detection software named Plagiarism Checker X. Plagiarism Checker X is a desktop application that can be used to detect the similarity between documents online as well as offline.

Table 5 is a table that shows the Confusion Matrix from a plagiarism classification “heavy”. Based on the results of the table, it is known that the value of its Precision calculation is 0.7, the value of the Recall is 1, the value of its F Measure is 0.8235, and the value of its Accuracy is 0.7.

Table 6 shows a Confusion Matrix of “medium” plagiarism classification. Based on the results shown in the table, the calculation value of Precision, Recall, and F Measure is 0, while the value of Accuracy is 0.7.

Table 5. Heavy classification Confusion Matrix

		Predicted (Research System)	
		Heavy	**Not-Heavy**
Actual (Plagiarism Checker X)	**Heavy**	(TP) 7	(FN) 0
	Not-Heavy	(FP) 3	(TN) 0

Table 6. Medium classification Confusion-Matrix

		Predicted (Research System)	
		Medium	**Not-Medium**
Actual (Plagiarism Checker X)	**Medium**	(TP) 0	(FN) 3
	Not-Medium	(FP) 0	(TN) 7

8. Conclusions.

The conclusions obtained in this research are that Rabin-Karp and Con-fix-Stripping algorithms have been successfully implemented in a website based application with PHP programming language to detect similarity level between text documents. The analytical results from this research show that the scenario algorithm that gives best results, both in terms of time and similarity level, is system with a pure Rabin-Karp algorithm with average speed of total processing time of 0.0123 second because the process does not take additional time to run the N-Gram or stemming process, and with an average similarity level of 89.1967%. System with the fastest Rabin-Karp calculation is obtained by using the Rabin-Karp algorithm without N-Gram through stemming process with an average processing time of 0.0103 second, because stemming process shortens the affixed word, so that the hashing process in Rabin-Karp gets faster. Systems with Rabin-Karp without N-Gram can detect up to 100% for word-changing in documents, whereas the Rabin-Karp plus N-Gram system can not detect up to 100% because when the systems include N-Gram in it, the changing of word location can make a difference of some word fractions, so the system can not find some of the same hash values even though all the words of both documents are identical. The system has also been able to detect 100% similarities by using four different algorithms' scenarios. The level of system accuracy is good against the classification of "medium" and "heavy" plagiarism of 0.7.

Based on this research, there are some suggestions for application development and further research, and some of them are:

1) Adding another string-matching algorithm such as Boyer-Moore which matches characters from right to left to get more system performance comparison results in terms of time or similarity level, either by adding N-Gram or stemming process. In addition, document scenarios can also be further developed.
2) Development of string matching or stemming algorithms used to obtain systems with better processing in terms of process speed, similarity, and system accuracy.

REFERENCES

[1] Ministry of Education of the Republic of Indonesia, *Minister of National Education Regulation No. 17 Year 2010 on Plagiarism Prevention and Control in Higher Education*, Jakarta, 2010.

[2] S. A. Setiyati, *Qualitative Descriptive Study on the Behavior of Plagiarism on Students Who Develop Thesis in the Faculty of Psychology Universitas Muhammadiyah Purwokerto*, Bachelor Thesis, Universitas Muhammadiyah Purwokerto, Purwokerto, 2015.

[3] Tudesman, E. Oktalina, Tinaliah and Yoannita, Indonesian document plagiarism detection system using vector space model method, *Jurnal Online Program Studi Teknik Informatika*, 2014.

[4] A. Wibowo, Preventing and overcoming plagiarism in education, *Kesmas National Public Health Journal*, vol.6, no.5, pp.195-200, 2012.

[5] Y. A. Wicaksono, Suyanto and Suyanto, *Analysis and Implementation of Rabin-Karp Algorithm and Nazief-Adriani Stemming Algorithm on Plagiarism Detection System of Indonesian Language Text Document*, Universitas Telkom, Bandung, 2012.

[6] P. A. R. L. Lede, A. Fanggidae and Y. T. Polly, Implementation of Rabin-Karp algorithm to detect plagiarism allegations based on similarity levels of words in text documents, *Jurnal Komputer dan Informatika (J ICON)*, vol.2, no.1, pp.50-64, 2014.

[7] J. Asian, H. E. Williams and S. M. M. Tahaghoghi, Stemming Indonesian, *Proc. of the 28th Australasian Conference on Computer Science*, Australia, pp.1-8, 2005.

[8] A. Kurniawati and I. W. S. Wicaksana, Comparison of document plagiarism detection approach in English language, *Proc. of Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008)*, Universitas Gunadarma, Depok, Indonesia, 2008.

[9] N. H. Ariyani, Sutardi and R. Ramadhan, Similarity of text file contents detection application using levenshtein distance method, *Jurnal Teknik Informatika*, vol.2, no.1, pp.279-286, 2016.

[10] H. B. Firdaus, Detecting document plagiarism using Rabin-Karp algorithms, *Jurnal Ilmu Komputer dan Teknologi Informasi*, vol.3, no.2, pp.1-5, 2003.

[11] B. Kurniawan, S. Effendi and O. S. Sitompul, Classification of news content by text mining method, *Jurnal Dunia Teknologi Informasi*, vol.1, no.1, pp.14-19, 2012.

[12] A. D. Tahitoe and D. Purwitasari, *Implementation of Modified Enhanced Confix Stripping Stemmer for Indonesian with Corpus Based Stemming Method*, Institut Teknologi Sepuluh Nopember, Surabaya, 2010.

[13] W. B. Cavnar and J. M. Trenkle, N-gram-based text categorization, *Ann Arbor MI Journal*, vol.48113, no.2, p.162, 1994.

[14] A. P. Putra and B. Susetyo, Performing arts shape of Angklung Carang Wulung, *Jurnal Seni Musik Jurusan Sendratasik*, pp.2-3, 2012.

[15] B. Munir, A. A. Nasution and Y. Purnamasari, Determinants affecting the depression of infarction post stroke patients in Saiful Anwar Hospital, *Malang Neurology Journal*, vol.2, no.2, pp.60-62, 2016.

[16] N. C. Apsari, Assistance for children with thalassemia and their families, *Social Work Journal*, vol.6, no.2, pp.154-157, 2016.

[17] D. Haryadi, *Winnowing Algorithm Implementation with Preprocessing Stage in Text Document Plagiarism Detection Application*, Bachelor Thesis, Universitas Multimedia Nusantara, Tangerang, 2012.

[18] P. P. Adikara, *Indonesian Basic Word Dictionary and Stopword List*, http://hikaruyuuki.lecture.ub.ac.id/kamus-kata-dasar-dan-stopword-list-bahasa-indonesia.