JOIN (Jurnal Online Informatika)
Volume 6 No.1 | June 2021: 88-95
DOI: 10.15575/join.v6i1.640
p-ISSN: 2528-1682
e-ISSN: 2527-9165

The Hybrid of Jaro-Winkler and Rabin-Karp Algorithm in Detecting Indonesian Text Similarity

Muhamad Arief Yulianto[1], Nurhasanah[2]
[1,2]Teknik Informatika, Universitas Pamulang, Indonesia
[1]dosen02547@unpam.ac.id, [2] dosen1123@unpam.ac.id

Article Info

Article history:

Received September 23, 2020
Revised November 19, 2020
Accepted December 02, 2020
Pubilshed June 17, 2021

Keywords:

Combination
Jaro-Winkler
Rabin-Karp
Text similarity

ABSTRACT

The String-matching technique is part of the similarity technique. This technique can detect the similarity level of the text. The Rabin-Karp is an algorithm of string-matching type. The Rabin-Karp is capable of multiple patterns searching but does not match a single pattern. The Jaro-Winkler Distance algorithm can find strings within approximate string matching. This algorithm is very suitable and gives the best results on the matching of two short strings. This study aims to overcome the shortcomings of the Rabin-Karp algorithm in the single pattern search process by combining the Jaro-Winkler and Rabin-Karp algorithm methods. The merging process started from pre-processing and forming the K-Gram data. Then, it was followed by the calculation of the hash value for each K-Gram by the Rabin-Karp algorithm. The process of finding the same hash score and calculating the percentage level of data similarity used the Jaro-Winkler algorithm. The test was done by comparing words, sentences, and journal abstracts that have been rearranged. The average percentage of the test results for the similarity level of words in the combination algorithm has increased. In contrast, the results of the percentage test for the level of similarity of sentences and journal abstracts have decreased. The experimental results showed that the combination of the Jaro-Winkler algorithm on the Rabin-Karp algorithm can improve the similarity of text accuracy.

Corresponding Author:

Muhamad Arief Yulianto,
Department of Information Engineering,
Universitas Pamulang,
Puspitek Buaran Road, South Tangerang Township 15310, Banten, Indonesia.

Email: dosen02547@unpam.ac.id

1. INTRODUCTION

A similarity technique is a widely used fundamental concept. For example, the similarity between the two subjects (A and B) is related to the similarity/difference between them[1]. There are several kinds of algorithms that use string-matching techniques to identify the level text similarity[2]. The algorithm of Rabin-Karp is an algorithm of string-matching type that is useful in several pattern searches[3]. This algorithm uses a hash function to determine the feature value of each word chunk[4]. This algorithm is very suitable to be used to find many patterns in strings[5]. However, this algorithm was not compatible with the process of searching for a single pattern[6]. The application of the algorithm Rabin-Karp has been done by previous research such as detecting documents' plagiarism by Hartanto, et al, 2019[7], Filcha, et al., 2019[8], Steveson, et al., 2018[9], Priambodo, J, 2018[10], Uji Cahyono, Drajad, 2018[11].

In 2017, Andyasah, et al.[12] analyzed the influence of the length of K-Grams in the Rabin-Karp algorithm determining the plagiarism level. The analysis concluded that the determination of the length of the K-Grams affected the percentage level of plagiarism. A higher K-Grams tended to present a lower degree of similarity. Besides, lower K-Grams was able to increase the percentage level of similarity.

Approximate string matching is used in searching *string* based on the *string* with the similarity of the writing on the dictionary. The algorithm can be used to search a string including Jaro-Winkler, Hamming, Damerau Levenshtein, and Levenshtein Distance[13]. The Jaro-Winkler algorithm is very suitable and gives the best results on matching two short strings[14]. The using of Jaro-Winkler algorithm has been done by

previous researchers, namely, the *autocorrect* and the *spelling suggestion*[15], [16] feature, detecting plagiarism document[17]–[26], stemming the word compensated is not standard English by Qulub, et al, 2020[27].

In 2016, Rochmawati, et al.[13] conducted research that compared the algorithm of the search *string* in the *approximate string matching* methods to identify errors typing text. The methods compared were the Jaro-Winkler, Levenshtein, Hamming, and Damerau Levenshtein Distance. Based on the testing results, it indicated that the algorithm of Jaro-Winkler is the best in checking the word with the value of the MAP is 0.87. The combination of *the string-matching* and *the approximate string-matching* algorithms has ever been done by Saberi, et al., 2018[6]. In the research, researchers combined the Levenshtein distance with the algorithm of Rabin-Karp to improve the accuracy of the degree of similarity of the document. The research concluded that the use of the Levenshtein algorithm to calculate the distance of the hash on these two documents will produce a better level of accuracy.

Based on the previous research, it was stated that the Rabin-Karp algorithm was less suitable in the process of finding a single pattern. Besides, the Jaro-Winkler algorithm is very suitable to be applied in the short string searching process. Thus, the researchers intended to combine these two algorithms to deal with the shortcomings of the Rabin-Karp algorithm. The stages start from preprocessing the data, followed by calculating the hash value of each piece of string by the algorithm of Rabin-Karp, then concluding with the calculation of the percentage of the similarity degree of the data using the Jaro-Winkler algorithm. The incorporation of those algorithms is expected to increase the accuracy of the percentage of the similarity degree in text.

2. METHOD

The stages of the research implementation can be seen based on the image flowchart below:

Figure 1. The Hybrid of Jaro-Winkler and Rabin Karp Flowchart

2.1. Data Acquisition

The data research used in this research was the abstract data of the study. This data was obtained from various journals or seminars that serve as research source references. The data was converted in the form of text.

2.2. PREPROCESSING

The Flowchart below represents the stages of the preprocessing process:

Figure 2. Preprocessing

2.2.1. Case Folding

The procedure of changing all letters in the text is all in lowercase[28]. The example (provided in Bahasa Indonesia) of the process of case-folding is in below:

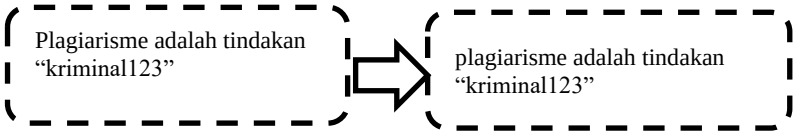

Figure 3. Case Folding

2.2.2. Tokenizing

The cutting process of the string is based on each of its constituent words as well as removing the delimiter character[29]. The example of the process of tokenizing is in below:

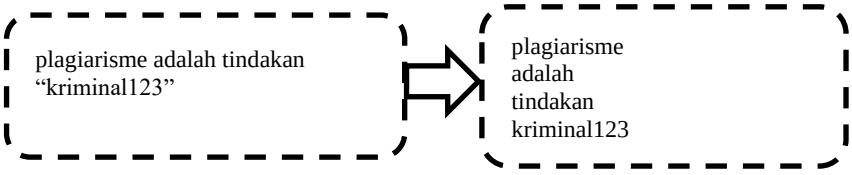

Figure 4. Tokenizing

2.2.3. Filtering

The procedure of selecting the important word by eliminating insignificant word, such as conjunction and pronoun [30]. Below is the process of filtering:

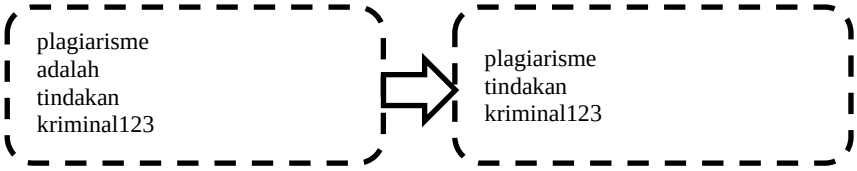

Figure 5. Filtering

2.2.4. Stemming

The procedure of discovering the basic word which is used in establishing the feature of the text [31]. The researcher used Nazief&Adriani algorithm in the stemming process. Below is the process of stemming:

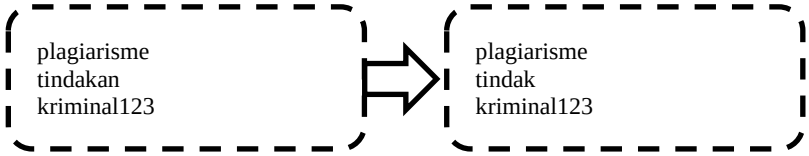

Figure 6. Stemming

2.3. Rabin-Karp Algorithm

The Algorithm of Rabin-Karp steps can be seen in the flowchart below:

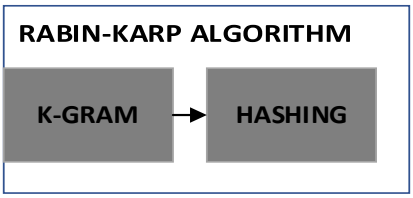

Figure 7. Rabin-Karp Algorithm Process

2.3.1. K-Gram

K-Gram is the process of forming the word pattern by continuously dividing the word as much as possible from the beginning to the end of the text [32]. Below is the example of the using of k-gram with k=3 value in the text "plagiarismetindakankriminal123":

[pla, lag, agi, gia, iar, ari, ris, ism, sme, met, eti, tin, ind, nda, dak, aka, kan, ank, nkr, kri, rim, imi, min, ina, nal]

2.3.2. Hashing

The process to change a string/text into a special value with a permanent length as the string feature. The function which produced the unique value is called as hash function and its result is labeled as value of hash. The similar function of hash is as follow [7]:

$$h(s) = \left(s[i] * b^{(n-1)} + s[i+1] * b^{(n-2)} + \cdots + s[i+n-1]\right) mod\ q \quad (1)$$

Where :

$h(s) = hash\ value\ (substring)$

$s = ASCII\ value\ each\ character$

$b = Prime\ Base\ Numbers$

$q = modulo$

Below is the example of hash calculation from the result of the previous k-gram with parameter value b = 397 and q = 1007:

[177, 797, 515, 375, 264, 854, 38, 356, 115, 901, 695, 60, 376, 1002, 693, 81, 287, 275, 777, 989, 32, 991, 469, 373, 829]

2.4. Jaro-Winkler Algorithm

The step of Jaro-Winkler Algorithm process can be seen in the flowchart below:

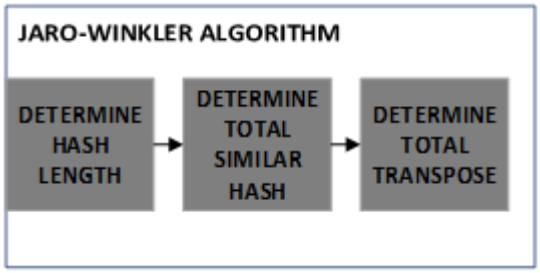

Figure 8. Jaro-Winkler Algorithm Process

2.4.1. Determine Hash Length

The program code which used to calculate the number of hash value at text is as follow:

```
nameList.size();
```

2.4.2. Determine The Number of Equal Hashes

The program code which used to establish the number of hash similar:

```
for (int x = 0; i < length1; x++)  {
       for (int y = Math.max(0, x - maxDist);y < Math.min(length2, x + maxDist + 1);
y++){
              if ((listOne.get(x)).equals(listTwo.get(y)) && hash2[y] == 0)  {
                     hash1[x] = 1;
                     hash2[y] = 1;
                     matched++;
                     break;
              }
       }
   }
```

2.4.3. Determine The Number of Transpositions

The program code that is used to determine the transposition total is as follows:

```
for (int x = 0; i < length1; x++){
         if (hash1[x] == 1) {
                     while (hash2[tip] == 0){
                     tip++;
                  }
                  int listOneTemp = listOne.get(x);
                  int listTwoTemp = listTwo.get(tip++);

                  if (listOneTemp != listTwoTemp ){
                     t++;
                  }
          }
    }
```

2.5. Similarity Score

The percentage level calculation of the text similarity in this research used Jaro-Winkler similarity. The determination of similarity text level also used Jaro-Winkler similarity based on Jaro-Winkler similarity is as follows [33]:

$$Jaro_{(s_1,s_2)} = \begin{cases} \frac{1}{3}.\left(\frac{m}{|s_1|}+\frac{m}{|s_2|}+\frac{m-t}{m}\right) : m > 0 \\ 0 \qquad\qquad\qquad\quad : otherwise \end{cases} \tag{2}$$

Where :

$|s_1|, |s_2| = both\ strings\ length(hash)$

$m \quad = matching\ characters\ number$(hash)

$t \quad = transpositions\ number$

The program code used in the calculation process of Jaro similarity is as follows:

```
public double jaro_distance(ArrayList<Integer> arrList1, ArrayList<Integer> arrList2)  {
  ….
  ….
  ….
  return (((double)match) / ((double)len1)  + ((double)match) / ((double)len2) + ((double)match - t) / ((double)match))  / 3.0;
}
```

The Jaro-Winkler similarity equation was obtained by adding the same prefix in the Jaro similarity value as follows:

$$JaroWinkler_{(s_1,s_2)} = \begin{cases} Jaro_{(s_1,s_2)} + l.p.\left(1 - Jaro_{(s_1,s_2)}\right) & : Jaro_{(s_1,s_2)} \geq b_t \\ Jaro_{(s_1,s_2)} & : otherwise \end{cases} \quad (3)$$

Where :

$Jaro_{(s_1,s_2)} = Jaro\ Distane\ Value$

$l = common\ prefix\ up\ to\ maks\ 4$

$p = prefix\ scale\ constant\ (p = 0.1)$

$b_t = boost\ treshold\ (b_t = 0.7)$

The program code used in the calculation process of Jaro-Winkler similarity is as follows:

```
public double jaro_winkler(ArrayList<Integer> arrList1, ArrayList<Integer> arrList2, double jaro_distance) {
   ....
   ....
   ....
    if (jaro_distance > 0.7){
           int prefix = 0;
           int len1 = listOne.size(), len2 = listTwo.size();
           for (int x = 0;x < Math.min(length1, length2); x++)
          {
                if ((listOne.get(x)).equals(listTwo.get(x)))
                   prefix++;
                else
                   break;
           }
           prefix = Math.min(4, prefix);
            jaroWinkler = jaro_distance + (0.1 * prefix * (1 - jaro_distance));
      }else {
            jaroWinkler = jaro_distance;
       }
       return jaroWinkler;
    }
```

3. RESULTS AND DISCUSSION

In this research, the researcher executed a process using three types of data input in the form of word, sentence and abstract of journal in Indonesian language. In the process of testing level of data similarity, a process of changing letter, word, and sentence arrangement was carried out againts the data that have been made. Meanwhile, journal abstracts were also tested on several journals with the same theme. The word similarity test data was obtained from the original data which changes the position of the letters by k-grams, while the sentence and journal similarity test data is obtained from the results of changes in the structure of words or sentences as much as n-grams. The values for the K-Gram, Basis and Modulo parameters for Rabin-Karp used in this study were 3, 397 and 1007 respectively. The merging process of algorithms started from pre-processing and forming the K-Gram data. Then, it was followed by the calculation of the hash value for each K-Gram by the Rabin-Karp algorithm. The process of finding the same hash score and calculating the percentage level of data similarity used the Jaro-Winkler algorithm. The result of tests are shown below:

3.1. Word Similarity Test Results

In the process of testing the level of similarity words have done by comparing between the same words, but the letter arrangement is changed. Meanwhile, the number of letters whose positions were exchanged in the testing process for each word is one alphabet, two alphabets, and three alphabets with the position of being replaced randomly. The data and test results are as follows:

Table 1. Word Similarity Test Data

Base Word	Testing Word	Number of Alphabets Swapping
Plagiarism	palgiarisme	One Alphabet
	palgairisme	Two Alphabets
	palgairisem	Three Alphabets
Konsultasi	konslutasi	One Alphabet
	konslutsai	Two Alphabets
	konslutsia	Three Alphabets
Dokumentasi	dokumentsai	One Alphabet
	dkoumentsai	Two Alphabets
	dkoumnetsai	Three Alphabets
Keterlambatan	keterlmabatan	One Alphabet
	kteerlmabatan	Two Alphabets
	kteerlmabtaan	Three Alphabets
Berorientasi	berorientsai	One Alphabet
	breorientsai	Two Alphabets
	breorientsia	Three Alphabets

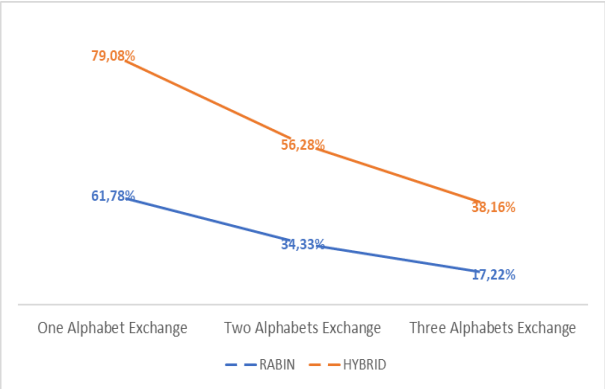

Figure 9. Word Similarity Test Results

The line chart above was obtained based on the testing between two similar words but changing the letter position. The changes of letter position were executed in three different variants, namely a change in the position of 1 letter, 2 letters, and 3 letters. The diagram results above shows that the combination algorithm have a higher accuration level than Rabin-Karp algorithm.

3.2. Sentence Similarity Test Results

In line with the process of testing the level of word similarity, testing the level of sentences similarity were also carried out by changing the word order in the same sentence. The number of words whose positions were exchanged in the testing process for each sentence is one word, two, three, and four words with the position of being replaced randomly. The data and test results are as follows:

Table 2. Sentence Similarity Test Data

Base Sentence	Testing Sentence	Number of Word Swapping
anton menendang kuda di lapangan	menendang anton kuda di lapangan	One Word
	menendang kuda anton di lapangan	Two Words
	menendang di lapangan kuda anton	Three Words
	di lapangan menendang kuda anton	Four Words
Budi memakan Daging Kemarin	Memakan Budi Daging Kemarin	One Word
	Memakan Daging Budi Kemarin	Two Words
	Daging Memakan Budi Kemarin	Three Words
	Kemarin Daging Memakan Budi	Four Words
Ayah Menulis Surat Undangan Sunat Kemarin	Kemarin Ayah Menulis Surat Undangan Sunat	One Word
	Kemarin Ayah Menulis Undangan Surat Sunat	Two Words
	Kemarin Ayah Menulis Undangan Sunat Surat	Three Words
	Kemarin Menulis Ayah Undangan Sunat Surat	Four Words

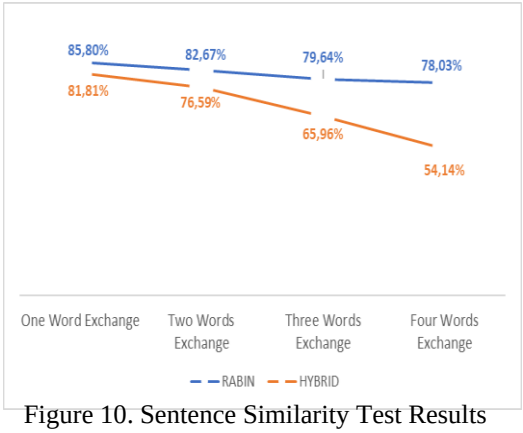

Figure 10. Sentence Similarity Test Results

The line chart above was obtained based on testing between the same two sentences but by changing the word position. Changes in word position were carried out in four different variants, namely changing the placement of one word, two, three and four words. Based on these results, it indicated that the combination algorithm possessed a lower level of accuracy than the Rabin-Karp algorithm.

3.3. Journal Abstract Similarity Test Results

The process of testing the similarity of Indonesian language journal abstract was not only done by testing the same journal by changing the order of words and sentences, but also testing it with journal abstracts that have the same theme. The data and test results are as follows:

Table 3. Journal Abstract Similarity Test Result

Abstract Journal Testing	Rabin-Karp Similarity	Hybrid Algorithm Similarity
Three Word Exchange	99.17%	93.42%
Six Word Exchange	98.34%	91.52%
Nine Word Exchange	98.06%	89.91%
Twelve Word Exchange	97.23%	89.29%
Fifteen Word Exchange	96.67%	88.32%
One Sentence Exchange	99.72%	93.50%
Two Sentence Exchange	99.58%	87.47%
Three Sentence Exchange	99.45%	82.41%
Similar Topics	78.88%	49.63%
Similar Topics	87.55%	52.28%
Similar Topics	93.86%	49.62%

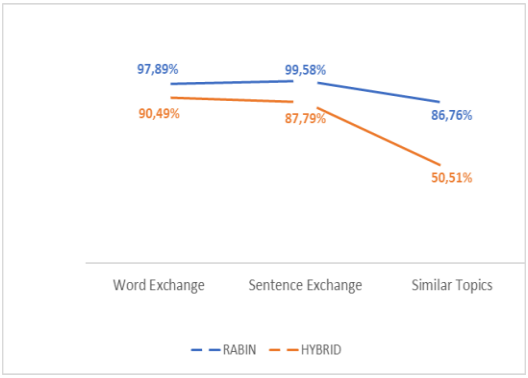

Figure 11. The Average Journal Abstract Similarity Test Results

The line chart above was obtained based on testing between the same two journal abstracts by changing the position of words and sentences. Changes in word position were carried out in five different variants, namely changing the placement of three words, six words, nine words, 12 words and 15 words. Meanwhile, changes in sentence position were carried out in three different variants, namely changing the placement of one sentence, two sentences and three sentences. The sentence and abstract similarity test data was formed based on changes in word and sentence structure as much as n-grams. This was done, because if you made changes to the arrangement of k-grams, the resulting test data will not change much from the original data. Moreover, the researcher also conducted the testing of journals which considered the same theme.

4. CONCLUSION

The test results above showed that it improved the presentation of the accuracy of the Rabin-Karp algorithm on the word similarity test by 20.06% through the implementation of the Jaro-Winkler algorithm on Rabin-Karp. However, the percentage level in testing the similarity of sentences and journal abstracts decreased by 15.20%. It showed that the implementation of the Jaro-Winkler algorithm on Rabin-Karp could improve the accuracy in detecting text similarities. Future research is expected to be able to analyze the implementation time in determining the level of text similarity using a web-based application.

ACKNOWLEDGEMENTS

The researcher would like to thank the Ministry of Research and Technology/National Agency for Research and Innovation for providing research grants. Dr. (HC) H. Darsono as Chairman of the Sasmita Jaya Foundation, Pamulang University. Dr. Ali Madinsyah and the LPPM team from Pamulang University who facilitated and provided guidance during the research implementation.

5. REFERENCES

[1] H. Ezzikouri, M. Erritali, and M. Oukessou, "Semantic Similarity / Relatedness for Cross Language Plagiarism Detection," vol. 1, no. 2, pp. 371–374, 2016, doi: 10.11591/ijeecs.v1.i2.pp371-374.

[2] B. Leonardo and S. Hansun, "Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 5, no. 2, pp. 462–471, 2017, doi: 10.11591/ijeecs.v5.i2.pp462-471.

[3] D. Leman, M. Rahman, F. Ikorasaki, B. S. Riza, and M. B. Akbbar, "Rabin Karp and Winnowing Algorithm for Statistics of Text Document Plagiarism Detection," *2019 7th Int. Conf. Cyber IT Serv. Manag. CITSM 2019*, 2019, doi: 10.1109/CITSM47753.2019.8965422.

[4] A. P. U. Siahaan, "Rabin-Karp Elaboration in Comparing Pattern Based on Hash Data," *Int. J. Secur. Its Appl.*, vol. 12, no. 2, pp. 59–66, 2018, doi: 10.14257/ijsia.2018.12.2.06.

[5] A. Bahrul Khoir, H. Qodim, B. Busro, and A. Rialdy Atmadja, "Implementation of rabin-karp algorithm to determine the similarity of synoptic gospels," *J. Phys. Conf. Ser.*, vol. 1175, no. 1, 2019, doi: 10.1088/1742-6596/1175/1/012120.

[6] A. P. U. Siahaan *et al.*, "Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level," *Int. J. Eng. Technol.*, vol. 7, no. 2 Special Issue 27, pp. 17–21, 2018, doi: 10.14419/ijet.v7i2.27.12084.

[7] A. D. Hartanto, A. Syaputra, and Y. Pristyanto, "Best parameter selection of rabin-Karp algorithm in detecting document similarity," *2019 Int. Conf. Inf. Commun. Technol. ICOIACT 2019*, no. February 2020, pp. 457–461, 2019, doi: 10.1109/ICOIACT46704.2019.8938458.

[8] A. Filcha and M. Hayaty, "Implementasi Algoritma Rabin-Karp untuk Pendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa," *JUITA J. Inform.*, vol. 7, no. 1, p. 25, 2019, doi: 10.30595/juita.v7i1.4063.

[9] D. Steveson, H. Agung, and F. Mulia, "APLIKASI PENDETEKSI PLAGIARISME TUGAS DAN MAKALAH PADA SEKOLAH MENGGUNAKAN ALGORITMA RABIN," *J. Algoritm. Log. dan Komputasi*, vol. 1, no. 1, pp. 12–17, 2018, [Online]. Available: https://journal.ubm.ac.id/index.php/alu.

[10] J. Priambodo, "Pendeteksian Plagiarisme Menggunakan Algoritma Rabin-Karp dengan Metode Rolling Hash," *J. Inform. Univ. Pamulang*, vol. 3, no. 1, p. 39, 2018, doi: 10.32493/informatika.v3i1.1518.

[11] D. Uji Cahyono, "Aplikasi Deteksi Dini Plagiarisme Judul Tugas Akhir Mahasiswa Sekolah Tinggi Ilmu Kesehatan Yayasan Rs. Islam Surabaya Dengan Algoritma Rabin-Karp," *Appl. Technol. Comput. Sci. J.*, vol. 1, no. 1, pp. 1–10, 2018, doi: 10.33086/atcsj.v1i1.3.

[12] A. P. U. Siahaan, R. Rahim, M. Mesran, and D. Siregar, "K-Gram As A Determinant Of Plagiarism Level in Rabin-Karp Algorithm," *Int. J. Sci. Technol. Res.*, vol. 06, no. 07, pp. 350–353, 2017, doi: 10.31219/osf.io/yxjnp.

[13] Y. Rochmawati and R. Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks," *J. Buana Inform.*, vol. 7, no. 2, pp. 125–134, 2016, doi: 10.24002/jbi.v7i2.491.

[14] D. Z. Putri, D. Puspitaningrum, and Y. Setiawan, "Konversi Citra Kartu Nama ke Teks Menggunakan Teknik OCR dan Jaro-Winkler Distance," *J. Teknoinfo*, vol. 12, no. 1, p. 1, 2018, doi: 10.33365/jti.v12i1.35.

[15] K. M. Suryaningrum and A. T, "Pengkoreksian dan Suggestion Word pada Keyword Menggunakan Algoritma Jaro-Winkler," *J. Teknol. Informasi-AITI*, vol. 13, no. 2, pp. 169–181, 2016.

[16] A. Prasetyo, W. M. Baihaqi, and I. S. Had, "Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 4, p. 435, 2018, doi: 10.25126/jtiik.201854780.
[17] S. C. Cahyono, "Comparison of document similarity measurements in scientific writing using Jaro-Winkler Distance method and Paragraph Vector method," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 662, no. 5, 2019, doi: 10.1088/1757-899X/662/5/052016.
[18] P. Novantara, "Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi," *Buffer Inform.*, vol. 3, no. 1, 2018, doi: 10.25134/buffer.v3i2.960.
[19] S. Christina, E. D. Oktaviyani, and B. Famungkas, "Mendeteksi Plagiarism Pada Dokumen Proposal Skripsi Menggunakan Algoritma Jaro Winkler Distance," *J. SAINTEKOM*, vol. 8, no. 2, p. 143, 2018, doi: 10.33020/saintekom.v8i2.68.
[20] I. E. Agbehadji, H. Yang, S. Fong, and R. Millham, "The Comparative Analysis of Smith-Waterman Algorithm with Jaro-Winkler Algorithm for the Detection of Duplicate Health Related Records," *2018 Int. Conf. Adv. Big Data, Comput. Data Commun. Syst. icABCD 2018*, 2018, doi: 10.1109/ICABCD.2018.8465458.
[21] T. Tinaliah and T. Elizabeth, "Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro-Winkler Distance dan Metode Latent Semantic Analysis," *J. Teknol. dan Sist. Komput.*, vol. 6, no. 1, pp. 7–12, 2018, doi: 10.14710/jtsiskom.6.1.2018.7-12.
[22] Jayanta, H. Mahfud, and T. Pramiyati, "Analisis pengukuran self plagiarism menggunakan algoritma Rabin-Karp dan Jaro-Winkler distance dengan stemming Tala," *Semin. Nas. Teknol. Inf. dan Multimed.*, vol. 5, no. 1, pp. 1–6, 2017.
[23] M. J. Tannga, S. Rahman, and Hasniati, "Analisis Perbandingan Algoritma Levenshtein Distance Dan Jaro Winkler Untuk Aplikasi Deteksi Plagiarisme Dokumen Teks," *Jtriste*, vol. 4, no. 1, pp. 44–54, 2017.
[24] L. Hakim, "Penggunaan N-Gram dan Jaro Winkler Distance pada Aplikasi Kelas Daring untuk Deteksi Plagiat," in *Seminar Nasional Sains dan Teknologi 2019*, 2019, pp. 1–10, [Online]. Available: jurnal.umj.ac.id/index.php/semnastek.
[25] S. Sugiono, H. Herwin, H. Hamdani, and E. Erlin, "Aplikasi Pendeteksi Tingkat Kesamaan Dokumen Teks: Algoritma Rabin Karp Vs. Winnowing," *Digit. Zo. J. Teknol. Inf. dan Komun.*, vol. 9, no. 1, pp. 82–93, 2018, doi: 10.31849/digitalzone.v9i1.1242.
[26] S. C. Cahyono, "Comparison of document similarity measurements in scientific writing using Jaro-Winkler Distance method and Paragraph Vector method," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 662, p. 52016, 2019, doi: 10.1088/1757-899x/662/5/052016.
[27] M. Qulub, E. Utami, and A. Sunyoto, "Stemming Kata Berimbuhan Tidak Baku Bahasa Indonesia Menggunakan Algoritma Jaro-Winkler Distance," *Creat. Inf. Technol. J.*, vol. 5, no. 4, p. 254, 2020, doi: 10.24076/citec.2018v5i4.218.
[28] A. F. Hidayatullah, "The influence of stemming on Indonesian tweet sentiment analysis," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2, no. August, pp. 127–132, 2015, doi: 10.11591/eecsi.v2i1.791.
[29] D. A. Putra and H. Sujaini, "Implementasi Algoritma Rabin-Karp untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah(CONTOH PLAGIAT)," *J. Sist. dan Teknol. Inf.*, vol. 4, no. 1, pp. 66–74, 2015, [Online]. Available: http://jurnal.untan.ac.id/index.php/justin/article/view/12411.
[30] M. V Bhosale and A. A. Vankudre, "Detection of Real-Time Traffic through Twitter Stream Analysis," *J. Adv. Eng. Sci.*, vol. 2, no. 2, pp. 124–126, 2017.
[31] P. M. Prihatini, I. K. G. D. Putra, I. A. D. Giriantari, and M. Sudarma, "Stemming Algorithm for Indonesian Digital News Text Processing," *Int. J. Eng. Emerg. Technol.*, vol. 2, no. 2, pp. 1–7, 2017.
[32] P. Internal, M. Algoritma, and T. Informatika, "SATIN – Sains dan Teknologi Informasi Sistem Pendeteksi Tingkat Kesamaan Teks pada Pengusulan Proposal," *SATIN - Sains dan Teknol. Inf.*, vol. 4, no. 2, pp. 84–91, 2018.