INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

Similarity Identification Based on Word Trigrams Using Exact String Matching Algorithms

Received:
20 Juni 2022
Accepted:
12 July 2022
Published:
13 August 2022

[1]Abdul Fadlil, [2]Sunardi, [3*]Rezki Ramdhani
[1,2]Electrical Engineering, Universitas Ahmad Dahlan,
[3]Master Program of Informatics, Universitas Ahmad Dahlan
E-mail: [1]fadlil@mti.uad.ac.id, [2]sunardi@mti.uad.ac.id,
[3]rezki2008048036@webmail.uad.ac.id
*Corresponding Author

Abstract—Several studies regarding excellent exact string matching algorithms can be used to identify similarity, including the Rabin-Karp, Winnowing, and Horspool Boyer-Moore algorithms. In determining similarities, the Rabin-Karp and Winnowing algorithms use fingerprints, while the Horspool Boyer-Moore algorithm uses a bad-character table. However, previous research focused on identifying similarities using these algorithms based on character n-gram. In contrast, identification based on the word n-gram to determine the similarity based on its linguistic meaning, especially for longer strings, had not been covered yet. Therefore, a word-level trigram was proposed to identify similarities based on the word trigrams using the three algorithms and compare each performance. Based on precision, recall, and running time comparison, the Rabin-Karp algorithm results were 100%, 100%, and 0.19 ms, respectively; the Winnowing algorithm results with the smallest window were 100%, 56%, and 0.18 ms, respectively; and the Horspool algorithm results were 100%, 100%, and 0.06 ms. From these results, it can be concluded that the performance of the Horspool Boyer-Moore algorithm is better in terms of precision, recall, and running time.

Keywords—string-matching; algorithm; performance; n-gram; similarity

Corresponding Author:

Author [Rezki Ramdhani],
Department [Master Program of Informatics],
Institution [Universitas Ahmad Dahlan],
Email [rezki2008048036@webmail.uad.ac.id]

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

I. INTRODUCTION

Identifying similarity using string matching algorithms has been widely applied as the first step in detecting plagiarism. These algorithms can be an exact string matching algorithm or an approximate string matching algorithm. The actual string matching algorithm requires a perfect match on each character being compared, while the approximate string matching algorithms can tolerate slight mismatches in personality being compared [1], [2].

The Rabin-Karp algorithm is an exact string matching algorithm that implements a hash function to find similarities between text and pattern strings. The text strings are converted into hash values then all the obtained hash values are selected as the fingerprints. The pattern strings are also converted into hash values and then chosen as the fingerprints. The selected fingerprints of both text-pattern are then compared to find any equal value. The equivalent value indicates a similarity between text and pattern [3], [4]. The Winnowing algorithm is an advanced version of the Rabin-Karp algorithm [5]. The application of the Winnowing algorithm to the Rabin-Karp algorithm is when determining fingerprints. In the Rabin-Karp algorithm, all the obtained hash values are considered fingerprints.

Meanwhile, in the Winnowing algorithm, fingerprints are determined by first grouping all hash values into windows based on a certain width. The rightmost minimum hash value obtained from each window is used as fingerprints [6], [7], [8], [9]. To prevent collisions in hash values, the Rolling Hash formula is used with the base number in the procedure using prime numbers [4], [10], [11]. A collision is an uneven distribution of keys in the hash table, which causes several hash values to have the same key. A hash table is an array-like data structure for storing data in the form of keys and values [12]. The complexity of the hash-based algorithms in the preprocessing is $O(m)$, while the complexity of matching is $O((n-m+1)*m)$, where the symbol of m is the length of the pattern and the character of n is the length of the text [2], [13].

The Horspool Boyer-Moore is one of the simplified versions of the Boyer-Moore algorithm [4], [14]. The Boyer-Moore algorithm is a character-based algorithm that applies a heuristic approach in the string matching process [4], [13], [15]. Pattern matching in this algorithm starts from right to left with two heuristic approaches wrong character heuristic and good suffix heuristic. A lousy character heuristic is applied to determine the value of the skip character if there is a mismatch during pattern matching. On the other hand, the excellent suffix heuristic is used to determine the skip value if the character being compared has a match on some of its characters at the time of pattern matching. The Boyer-Moore algorithm and its modified version have excellent performance in pattern matching and have been widely applied in various technology fields, including the Search and Replace features in operating systems [13], [16].

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

Several modified versions of Boyer-Moore algorithms have excellent performance, including Horspool Boyer Moore, Tuned Boyer Moore, Quick Search, Sunday Quick Search, and Berry-Ravindran [3].

The Horspool Boyer-Moore algorithm is one of the most effective algorithms among these algorithms, especially if the string of patterns is longer [4], [16], [17]. The Horspool algorithm removes the good suffix heuristic of the Boyer-Moore algorithm in the pattern matching process so that only the bad-character heuristic is used to compare characters [4], [14], [17], [18]. The Horspool Boyer-Moore algorithm was introduced by Nigel Horspool in 1980 [17]. This simplification of the algorithm allows a faster text-pattern matching process than the original algorithm [4], [14], [19], [20], [21]. The Horspool algorithm consists of two phases: the character matching phase and the sliding window shift phase [1]. The scanning process begins by aligning the sliding window of the pattern (needle) with the haystack string according to the length of the sliding window and then matching it from right to left [2], [17]. The sliding window size is as many as the number of characters used as a pattern, and the characters are compared from right to left (see Appendices section for more detail about the sliding window). The complexity of the Horspool Boyer Moore algorithm in forming the bad-character table is $O(\sigma+m)$. *In contrast,* the complexity in the matching process is $O(m*n)$, where σ is the size of the alphabet, m is the length of the needle, and n is the length of the haystack [2].

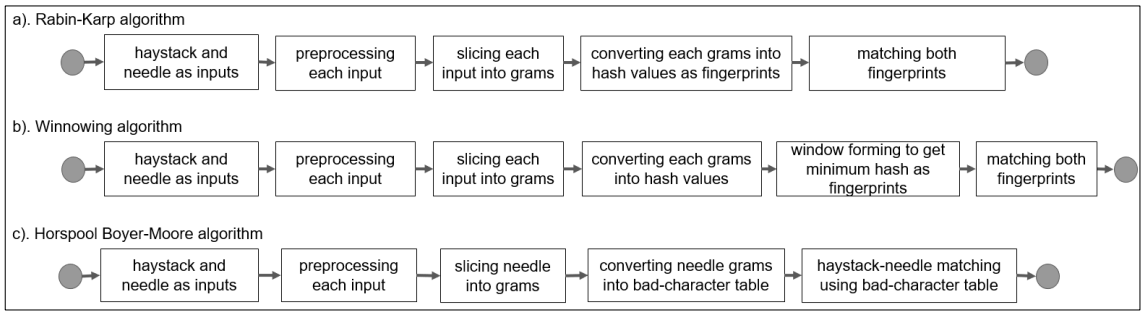


Figure 1. WORKFLOW OF ALGORITHMS

Figure 1 illustrates the different workflow of the Rabin-Karp algorithm, the Winnowing algorithm, and the Horspool Boyer-Moore algorithm. In Figure 1, it can be seen that the Rabin-Karp, Winnowing, and Horspool Boyer-Moore algorithms have to preprocess and split stages from the string into substrings (grams). These two hash-based algorithms split all inputs called haystacks and needles into grams. As for the Horspool, only the needle string is divided into substrings, while the haystack string is not split because the Horspool algorithm is a string-based pattern matching algorithm, and the substring pattern will be traced in the haystack string.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

Previous studies show that the Horspool and hash-based algorithms (Rabin-Karp and Winnowing algorithms) have good performance in string matching with the n-gram used in the patterns for identifying similarity based on character n-gram.

The Horspool algorithm had been used as a search method in the Javanese-Indonesian dictionary, where the algorithm had high accuracy (85.3%) with a short execution time (39.9 ms) [22]. The algorithm was superior to the original Boyer-Moore algorithm for multi-track string matching [18]. Another previous study to identify sensor devices using the Horspool algorithm also showed that the algorithm was efficient, especially if the length of the packet being matched was longer [19]. Another thing that is more interesting about this Horspool algorithm is that it can also be applied to detect malware on cloud networks; where the results showed that either used singly or integrated with other algorithms, the Horspool algorithm had a good performance because the number of attempts was less, thus speeding up the matching process [20]. Another excellence of the Horspool algorithm was implemented in the Network Intrusion Detection System (NIDS) because it could classify the types of attacks on the network well [21]. As for the Rabin-Karp and Winnowing algorithms as methods to identify word similarities between scientific work documents, the previous studies related to the application of both algorithms showed excellent results, either using in mono-language or multi-language forms and even using Chinese and Arabic letters [5], [9], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32].

Based on these results, it can be seen that the hash-function-based Rabin-Karp and Winnowing algorithms have good performance in identifying similarity using character n-gram. Likewise, the Horspool Boyer-Moore algorithm performs well in identifying similarity on complex objects and more extended patterns using the character n-gram. However, the use of these algorithms was still limited to identifying similarities using adjacent characters n-gram. In contrast, the use of adjacent words n-gram to determine similarity based on its linguistic meaning, especially for longer strings, had not been covered yet. Therefore, this research was carried out to identify similarities based on linguistic meaning using the word trigrams as the n-gram unit. The Rabin-Karp, Winnowing, and Horspool Boyer-Moore are used to determine the likeness so that the performance of the three algorithms in terms of precision and recall (sensitivity) parameters as well as the running times can be compared. Thus, it is expected that the results of this research will provide a clear picture of the algorithm with the most effective and efficient performance to be used as the foundation for developing a plagiarism detection system in future research.

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

II. RESEARCH METHOD

In this pattern matching research based on word-level trigrams using the Rabin-Karp algorithm and the Winnowing algorithm, the steps carried out consist of preprocessing, word-trigrams formation, hash values formation, and fingerprints comparison. Still, there was window formation before the comparison, especially in the Winnowing algorithm. The stages of the Horspool Boyer-Moore algorithm were preprocessing, needle word-trigrams formation, bad-character table formation, and haystack-needle matching. The illustration of the different stages of the Rabin-Karp, Winnowing, and Horspool Boyer-Moore algorithms [4], [14], [24], [26], [29], [30], [31], [32] in identifying similarity based on word-level trigrams is presented in Figure 2.

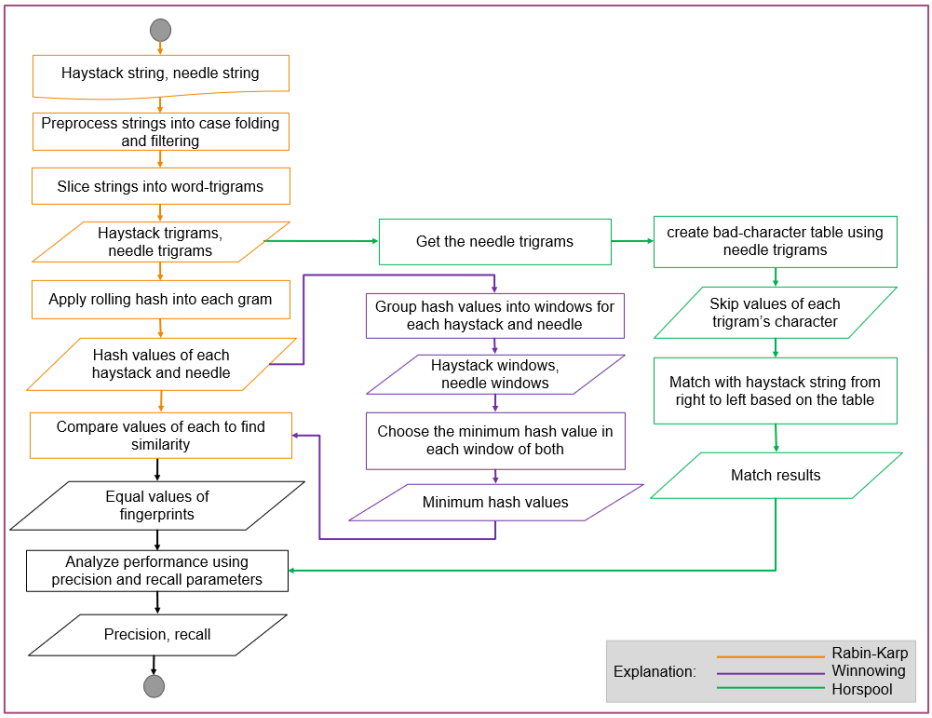


Figure 2. FLOWCHART OF ALGORITHMS

The illustration in Figure 2 shows that the detection process begins with the input preparation stage. In this study, two input strings were used. One input functions as a comparison text which is then called a haystack, while the other input functions as a pattern which is then called a needle. The performance measurement for the three algorithms uses parameters of precision and recall.

A. Preprocessing Stages

In this study, the preprocessing stage was applied to the Rabin-Karp, Winnowing, and Horspool Boyer-Moore. Preprocessing on the exact string matching algorithm generally consists of case folding, tokenizing, filtering, and stemming [26], [27], [32]. Case folding is a stage of converting all the letters on the haystack and needle into lowercase uniformly, tokenizing is a

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

stage of slicing string into substrings, filtering is a stage of removing meaningless symbols and letters in the two input texts, and stemming is a stage of converting affixed words into their original form (root word). In this study, only case folding, tokenizing, and filtering stages were carried out without stemming from shortening the running time. The number of alphabetic characters (c) used in this study is 27 characters consisting of characters [a-z] and the symbol of spaces [“ ”].

B. Grams Formation

A gram is a result of slicing a string into substrings using either character or word [3], [6], [33], [34], [35]. Unigrams, bigrams, or trigrams are units in n-gram terms that indicate the length of a gram. A trigram is a gram unit with a size of three adjacent characters at character-level n-gram or three adjacent words at the word-level n-gram. The n-gram based on word-trigram is generally applied in linguistics to interpret sentences based on the adjacent words [33], [34], [35]. For instance, slicing the string “searching for the same words” into substrings based on word level will result from unigrams as *[“searching”, “for”, “the”, “same”, “words”]*; bigrams as *[“searching for”, “for the”, “the same”, “same words”]*, and trigrams as *[“searching for the”, “for the same”, “the same words”]*. Using bigrams and trigram word levels to interpret sentences is more appropriate than using unigram [36].

C. Hash Values Formation

A hash-based algorithm uses hash values to identify similarity. The same hash value represents the same word. In the formation of hash values, collisions can occur, one of which is 258uet o the arrangement of characters. For instance, the substring of *“abcd”* can result in the same hash value as the substring of *“bcda”*. Therefore, to avoid collisions, sub-strings should be converted into the hash, valuessing the Rolling Hash formula [37]. The Rolling Hash formula is written as follows [29], [38]:

$$Hc_1..c_k = (c_1 * b^{k-1}) + (c_2\ x\ b^{k-2}) + \cdots + (c_{k-1}\ x\ b_k)) + c_k \quad (1)$$

$$Hc_2..c_{k+1} = (Hc_1..c_k - c_1\ x\ b^{k-1}) * b + c^{k+1} \quad (2)$$

Where:

$Hc_1..c_k$: the initial hash value

c_k : the ASCII code for each character

k : the unit of gram

b : Constanta (*in prime number*)

D. Window Formation by Winnowing Algorithm

Window formation is grouping hash values into a window based on a specific width to get the minimum hash value as fingerprints. The user can freely determine the window width (w), but it should be noted that the window width affects the algorithm's performance. In each window, the

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

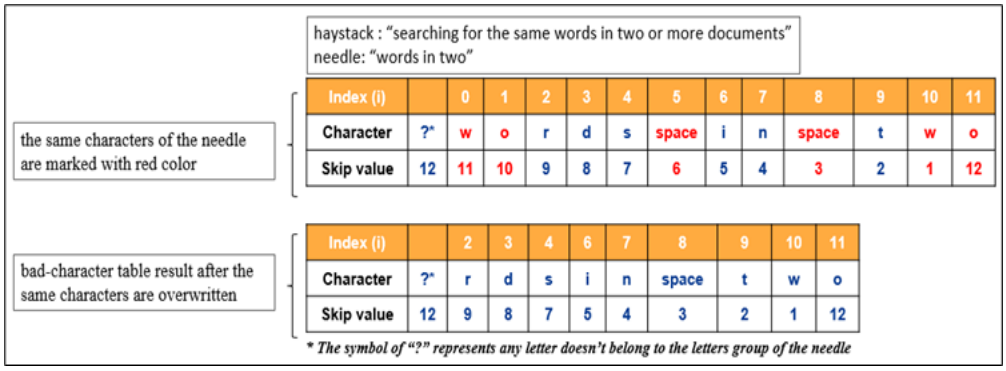
hash values of the haystack and the needle were grouped into a window with the number of members as many as the window's width. Then, the smallest value was taken from each window of the haystack and needle to determine fingerprints. When there were two or more equal minimum hash values from several windows of each haystack and hand, the value taken was the value on the rightmost of the window (the deal with the most extensive index) [6].

E. Fingerprints Comparison

In the Rabin-Karp algorithm, all the hash values obtained were considered fingerprints. In contrast, in the Winnowing algorithm, fingerprints were selected based on the minimum hash value obtained from each window applied to the hash value. Equal hash values obtained from fingerprints comparison between the fingerprints of the haystack and the fingerprints of the needle represented similar trigrams.

F. Bad-character table Horspool Boyer-Moore algorithm

In the Horspool algorithm, after the preprocessing stage was carried out, the next step was forming a bad-character table using the trigrams of the needle. The bad-character table is the central part of the Horspool algorithm in haystack-pattern matching. The bad-character table determines how far the sliding window must shift when there is a mismatch in the haystack pattern being compared so that the matching process becomes faster.



Index (i)		0	1	2	3	4	5	6	7	8	9	10	11
Character	?*	w	o	r	d	s	space	i	n	space	t	w	o
Skip value	12	11	10	9	8	7	6	5	4	3	2	1	12

Index (i)		2	3	4	6	7	8	9	10	11
Character	?*	r	d	s	i	n	space	t	w	o
Skip value	12	9	8	7	5	4	3	2	1	12

Figure 3. BAD-CHARACTER TABLE FORMATION

Figure 3 illustrates the formation of a bad-character table on the trigram of a needle so that the skip value of each character can be obtained to use in the next step. The pattern (arrow) to be searched for is *"words in two"*. The haystack is to be matched with the needle containing the sentence *"searching for the same words in two or more documents"*. Therefore, the formation of a bad-character table is then performed on the needle. In Figure 4, it shows that the characters in the bad-character table are *"w", "o", "r", "d", "s", space, "i", "n", 't"*, and *"?"*. This is because the characters that compose the needle are *"w", "o", "r", "d", "s", " ", "i", "n", " ", 't", " w"*, and *"o"* where the letters *"w"* and *"o"* and space symbol appear twice which causes the skip values to be overwritten. Therefore the value taken is the last skip value. The symbol of *"?"*

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

represents all the characters that compose the haystack that does not exist in the characters of the needle. The characters in the haystack that are not on the needle are the letters *"e", "a", "c", "h",* and so on. The skip value of *"?"* is 12 because the sliding window length is 12. The skip value of *"?"* is the same as the skip value of the letter "o" in the table because the letter *"o"* is the last character of the needle. Even though the letter "o" originally appeared at the beginning, it has been overwritten by the last *"o"*.

G. Haystack-Needle Matching by Horspool Boyer-Moore Algorithm

Figure 4 illustrates the pattern matching process between the plaintext of the haystack and a trigram of needle based on the bad-character tables of the Horspool algorithm (See Appendices section for more detail about the string matching the flow of the algorithm).

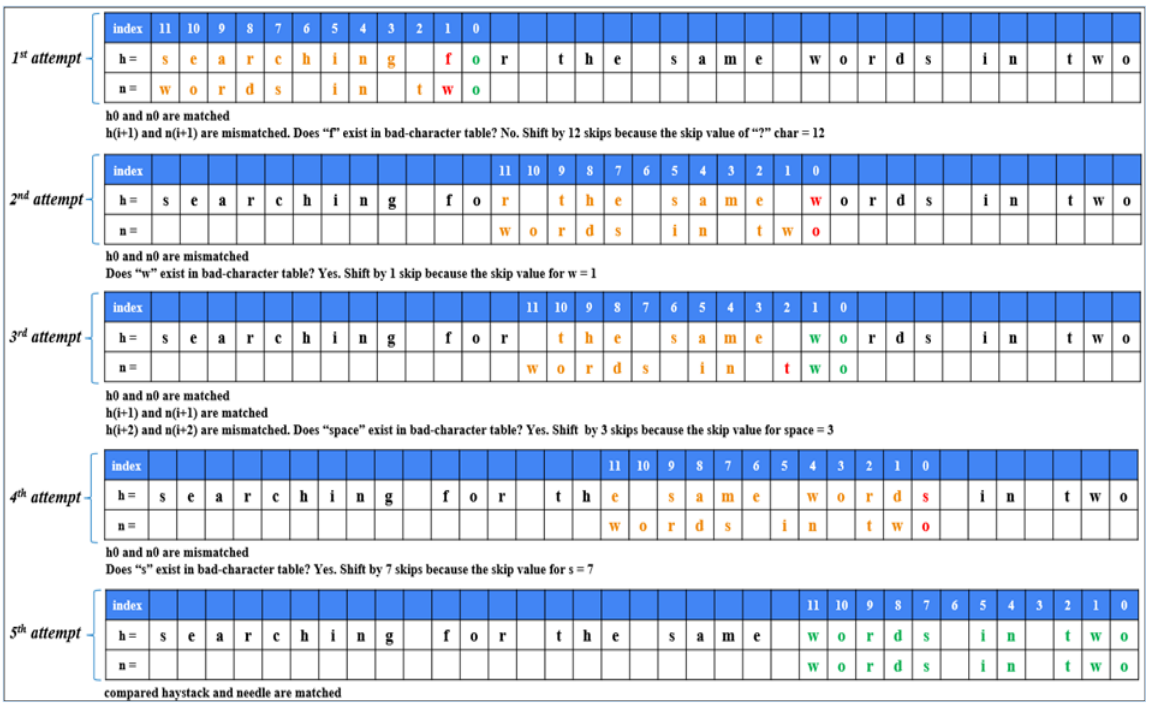


Figure 4. MATCHING PHASE OF HORSPOOL ALGORITHM

In Figure 4, it can be seen that to find a match between the Haystack string and the substring needle (pattern), it takes five attempts as follows. In the first attempt, after the alignment between the haystack and the hand and the pattern matching process has started from right to left, the results show that the characters h_0 and n_0 are matched; therefore, the matching continues on the following letter on the left side of h_0 and n_0, namely h_1 and n_1. However, h_1 and n_1 are mismatched. Therefore the character h_1 is traced in the bad-character table and checked whether its character, namely the letter *"f"*, is in the table or not. It turns out that because the character h_1 is not in the table, that h_1 is categorized as the character symbol of *"?"*. After the sliding window is shifted as long as 12 skips, the matching starts again from the first index of each character of the currently aligned haystack pattern from right to left.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

In the second attempt, h_0 and n_0 are mismatched; therefore, the character h_0, namely the letter *"w"*, is traced in the bad-character table. The skip value for the letter *"w"* is 1. After the sliding window is shifted as long as one skip, the matching starts again from the first index of each character of the currently aligned haystack pattern from right to left.

In the third attempt, h_0 and n_0 are matched to continue the matching on the next character, namely h_1 and n_1, which are compared. Matching continues on the next surface, namely h_2 and n_2 but is mismatched; therefore, the character h2, a *space symbol*, is traced in the bad-character table. The skip value for the *space symbol* is 3. After the sliding window has shifted as far as the skip value, the matching starts again from the first index of each character of the currently aligned haystack pattern from right to left.

In the fourth attempt, h_0 and n_0 are mismatched; therefore, the character h_0, namely the letter *"s"*, is traced in the bad-character table. The skip value for the letter *"s"* is seven, so the sliding window skips as far as that value. In the fifth attempt, all characters are matched; thus, the search for the first gram is found. The match is finished because it has reached the last character of the haystack string.

H. Performance Measurement

The parameters that are commonly used for algorithm performance in identifying similarity are precision and recall [39]. Precision is a parameter to measure the algorithm's accuracy, while memory is used to measure the algorithm's sensitivity. A harmonic mean (f-measure) is a parameter used to measure the balance between an algorithm's precision and recall. The formula for calculating precision, recall, and f-measure are as follows:

$$\text{Precision (P)} : \frac{TP}{TP+FP} \quad (3)$$

$$\text{Recall (R)} : \frac{TP}{TP+FN} \quad (4)$$

$$\text{F-measure (F*)} : \frac{2(P*R)}{P+R} = \frac{2\,TP}{2\,TP + FP + FN} \quad (5)$$

Where:

TP is a true positive, which means that the three adjacent words in the haystack aligned with the sliding window of the needle and considered *matched* by the algorithm are *matched*. *TN* is a true negative, which means that the three adjacent words in the haystack aligned with the sliding window of the needle and considered *unsuitable* by the algorithm are *mismatched*. *FP* is a false positive, which means that the three adjacent words in the haystack aligned with the sliding window needle are considered *matched* by the algorithm. They are *mismatched*. *FN* is a false negative, which means that the three adjacent words in the haystack aligned with the sliding window of the needle are considered *mismatched* by the algorithm. They are *matched*.

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

III. RESULT AND DISCUSSION

A. Experimental environment

The experiment was performed using *Acer Processor 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz, 2995 MHz, 2 Core(s), 4 Logical Processor(s)* with *RAM 8 GB/SSD 512 GB* and operating system *Windows 11 Home Single Language*. The programming language used was *Javascript* with *runtime* Node.js 16.14.0 and NPM 8.3.1.

B. Dataset and Preprocessing Results

The dataset used in this study was two plaintexts with different content of strings. One plaintext functions as the haystack and the other as the needle, as presented in Table 1.

Table 1. DATASET FOR EXPERIMENT

Before Preprocessing		**After Preprocessing**	
String Haystack	***String Needle***	***String Haystack***	***String Needle***
"Searching for the same words in two or more documents is the first step in the process of detecting plagiarism in scientific works."	"The first step in detecting plagiarism in scientific works is to look for the similar words in two or more documents."	"searching for the same words in two or more documents is the first step in the process of detecting plagiarism in scientific works"	"the first step in detecting plagiarism in scientific works is to look for the similar words in two or more documents"

C. Grams results

In the hash-based algorithms (Rabin-Karp and Winnowing), the trigram formation was applied to both plaintexts using word-level trigrams.

Table 2. TRIGRAMS FORMATION RESULTS

Trigrams of Haystack	**Trigrams of Needle**
['searching for the', 'for the same', 'the same words', 'same words in', 'words in two', 'in two or', 'two or more', 'or more documents', 'more documents is', 'documents is the', 'is the first', 'the first step', 'first step in', 'step in the', 'in the process', 'the process of', 'process of detecting', 'of detecting plagiarism', 'detecting plagiarism in', 'plagiarism in scientific', 'in scientific works']	['the first step', 'first step in', 'step in detecting', 'in detecting plagiarism', 'detecting plagiarism in', 'plagiarism in scientific', 'in scientific works', 'scientific works is', 'works is to', 'is to look', 'to look for', 'look for the', 'for the similar', 'the similar words', 'similar words in', 'words in two', 'in two or', 'two or more', 'or more documents']

Data in Table 2 show that the haystack trigrams obtained were 21 grams, and the trigrams needle were 19 grams—the trigrams formed by splitting the string into pieces of sub-strings. The trigrams formed on the hand were used not only to create fingerprints on the Rabin-Karp and Winnowing algorithm but also for the string matching process on the Horspool algorithm.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

D. Hash values Obtained from Rolling Hash Formulation

String matching using the Rabin-Karp and Winnowing algorithms began by converting the haystack and needle trigrams into hash values. Hash values were obtained by converting the haystack and needle trigrams into integer values using the Rolling Hash formula.

Table 3. HASH VALUES RESULTS

Hash Values of Haystack	Hash Values of Needle
[35375484, 12242798, 12038934, 15827180, 20864032, 8075538, 12932014, 8465958, 16392289, 36472303, 8257285, 12063909, 20647355, 16580372, 8071255, 12105173, 28612043, 8118645, 35629559, 39712737, 8213102]	[12063909, 20647355, 16581002, 8193143, 35629559, 39712737, 8213102, 39481651, 21125621, 8239028, 8540191, 16341245, 12243129, 12102817, 28156599, 20864032, 8075538, 12932014, 8465958]

Data in Table 3 show that the hash values of the haystack obtained were 21 hashes and the hash values of the needle were 19 hashes.

E. Windows Results of Winnowing algorithm

In the Winnowing algorithm, the hash values of each haystack and needle were first grouped into a window with a certain width. The hash minimum values obtained from each haystack window and needle window were then used as fingerprints.

Table 4. Windows Results By Winnowing Algorithm

Width of Window (w)	Total Windows of Haystack	Total Windows of Needle
w = 4	18 windows	16 windows
w = 3	19 windows	17 windows
w = 2	20 windows	18 windows

The data presented in Table 4 shows that the more comprehensive the width value set, the less the window is formed (see Appendices section for more detail).

Table 5. MINIMUM HASH VALUES FROM EACH WINDOW

Width of Window (w)	Haystack		Needle	
	Total of minimum hashes from Windows	***Hash in the rightmost window***	***Total of minimum hashes from Windows***	***Hash in the rightmost window***
w = 4	18	6	17	6
w = 3	19	8	18	7
w = 2	20	14	19	13

Data in Table 5 show that the wider the window, the less the hash values in the rightmost were selected as fingerprints because other hash values were not chosen as fingerprints, so they never became the minimum hash value in any window. (see Appendices section for more detail).

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

F. Fingerprints Comparison Results

1. Fingerprints Comparison Results of Rabin-Karp Algorithm

In the Rabin-Karp algorithm, the selected fingerprints were from all the haystack hash values and needle hash values. Table 6 presents the fingerprints used by the Rabin-Karp algorithm to identify the similarity between haystack and needle.

Table 6. FINGERPRINTS BY RABIN-KARP ALGORITHM

Fingerprints of Haystack	*Fingerprints of Needle*	*Equal values*	*Trigrams of equal values*
[35375484, 12242798, 12038934, 15827180, 20864032, 8075538, 12932014, 8465958, 16392289, 36472303, 8257285, 12063909, 20647355, 16580372, 8071255, 12105173, 28612043, 8118645, 35629559, 39712737, 8213102]	[12063909, 20647355, 16581002, 8193143, 35629559, 39712737, 8213102, 39481651, 21125621, 8239028, 8540191, 16341245, 12243129, 12102817, 28156599, 20864032, 8075538, 12932014, 8465958]	[20864032, 8075538, 12932014, 8465958, 12063909, 20647355, 35629559, 39712737, 8213102]	[‘words in two’, ‘in two or’, ‘two or more’, ‘or more documents’, ‘the first step’, ‘first step in’, ‘detecting plagiarism in’, ‘plagiarism in scientific’, ‘in scientific works’]

The data in Table 6 shows that from the comparison between haystack fingerprints and needles, nine fingerprints are equal, so the similarity between the two is relatively high.

2. Fingerprints Comparison Results of Winnowing Algorithm

Table 7 shows that the wider the window used in grouping hash values, the fewer fingerprints identified as similar.

Table 7. FINGERPRINTS BY WINNOWING ALGORITHM

Width of window (w)	**Fingerprints of Haystack**	**Fingerprints of Needle**	**Equals Fingerprints**	**Trigrams of equal**
w = 4	[12038934, 8075538, 8465958, 8257285, 8071255, 8118645]	[8193143, 8213102, 8239028, 8540191, 12102817, 8075538]	[8075538]	['in two or']
w = 3	[12038934, 8075538, 8465958, 8257285, 12063909, 8071255, 8118645, 8213102]	[12063909, 8193143, 8213102, 8239028, 8540191, 12102817, 8075538]	[8075538, 12063909, 8213102]	['in two or', 'the first step', ‘in scientific works’]
w = 2	[12242798, 12038934, 15827180, 8075538, 8465958, 16392289, 8257285, 12063909, 16580372, 8071255, 12105173, 8118645, 35629559, 8213102]	[12063909, 16581002, 8193143, 35629559, 8213102, 21125621, 8239028, 8540191, 12243129, 12102817, 0864032, 8075538, 8465958]	[8075538, 8465958, 12063909, 35629559, 8213102]	['in two or', ‘or more documents’, 'the first step', 'detecting plagiarism in', ‘in scientific works’]

As in a window width of 4 where the hash values were split into four groups, many other hash values were not selected as fingerprints because they did not become the smallest hash values in any window. But, when the window width is reduced, the hash value chosen as fingerprints also increases.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

G. Bad-Character Results of Horspool Boyer-Moore Algorithm

In the Horspool Boyer-Moore algorithm, the matching process was carried out by comparing the needle's trigrams with the haystack's plaintext. Therefore, in each needle trigram, a bad-character table was formed to determine the skip value of each character in each trigram before starting the haystack-needle matching.

Table 8. BAD-CHARACTER TABLE FOR EACH TRIGRAM

Trigrams of Needle	Skip Values
'the first step',	{ t: 2, h: 12, e: 1, ' ': 4, f: 9, i: 8, r: 7, s: 3 },
'first step in',	{ f:12, i: 1, r: 10, s: 6, t: 5, ' ': 2, e: 4, p: 3 },
'step in detecting',	{ s: 16, t: 3, e: 5, p: 13, ' ': 9, i: 2, n: 1, d: 8, c: 4 },
'in detecting plagiarism',	{ i: 2, n: 12, ' ': 10, d: 19, e: 16, t: 14, c: 15, g: 6, p: 9, l: 8, a: 4, r: 3, s: 1 },
'detecting plagiarism in',	{ d: 22, e: 19, t: 17, c: 18, i: 1, n: 15, g: 9, ' ': 2, p: 12, l: 11, a: 7, r: 6, s: 4, m: 3 },
'plagiarism in scientific',	{ p: 23, l: 22, a: 18, g: 20, i: 1, r: 17, s: 9, m: 14, ' ': 10, n: 5, c: 8, e: 6, t: 4, f: 2 },
'in scientific works',	{ i: 7, n: 11, ' ': 5, s: 15, c: 6, e: 12, t: 10, f: 8, w: 4, o: 3, r: 2, k: 1 },
'scientific works is',	{ s: 3, c: 9, i: 1, e: 15, n: 14, t: 13, f: 11, ' ': 2, w: 7, o: 6, r: 5, k: 4 },
'works is to',	{ w: 10, o: 9, r: 8, k: 7, s: 3, ' ': 2, i: 4, t: 1 },
'is to look',	{ i: 9, s: 8, ' ': 4, t: 6, o: 1, l: 3 },
'to look for',	{ t: 10, o: 1, ' ': 3, l: 7, k: 4, f: 2 },
'look for the',	{ l: 11, o: 5, k: 8, ' ': 3, f: 6, r: 4, t: 2, h: 1 },
'for the similar',	{ f: 14, o: 13, r: 12, ' ': 7, t: 10, h: 9, e: 8, s: 6, i: 3, m: 4, l: 2, a: 1 },
'the similar words',	{ t: 16, h: 15, e: 14, ' ': 5, s: 12, i: 9, m: 10, l: 8, a: 7, r: 2, w: 4, o: 3, d: 1 },
'similar words in',	{ s: 3, i: 1, m: 13, l: 11, a: 10, r: 5, ' ': 2, w: 7, o: 6, d: 4 },
'words in two',	{ w: 1, o: 10, r: 9, d: 8, s: 7, ' ': 3, i: 5, n: 4, t: 2 },
'in two or',	{ i: 8, n: 7, ' ': 2, t: 5, w: 4, o: 1 },
'two or more',	{ t: 10, w: 9, o: 2, ' ': 4, r: 1, m: 3 },
'or more documents.'	{ o: 8, r: 12, ' ': 10, m: 5, e: 4, d: 9, c: 7, u: 6, n: 3, t: 2, s: 1 }

The bad-character table formation, as presented in Table 8, shows that if there were the same characters in each trigram, the skip values would be overwritten by the last skip value of the same character.

H. Matching Results by Horspool Boyer-Moore Algorithm

The string matching results in Table 9 show similarities between the haystack and the needle because as many as nine trigrams were matched.

Table 9. MATCHING RESULTS BASED ON THE BAD-CHARACTER TABLE

String of Haystack	Trigrams of Needle	Matching Results of Horspool	Trigrams of Horspool Matching Results
"searching for the same words in two or more documents is the first step in the process of detecting plagiarism in scientific works."	['the first step',	[matched,	['the first step',
	'first step in',	matched,	'first step in',
	'step in detecting',	not,	-1,
	'in detecting plagiarism',	not,	-1,
	'detecting plagiarism in',	matched,	'detecting plagiarism in',
	'plagiarism in scientific',	matched,	'plagiarism in scientific',
	'in scientific works',	matched,	'in scientific works',
	'scientific works is',	not,	-1,
	'works is to',	not,	-1,
	'is to look',	not,	-1,
	'to look for',	not,	-1,
	'look for the',	not,	-1,
	'for the similar',	not,	-1,
	'the similar words',	not,	-1,
	'similar words in',	not,	-1,
	'words in two',	matched,	'words in two',
	'in two or',	matched,	'in two or',
	'two or more',	matched,	'two or more',
	'or more documents.']	matched]	'or more documents']

INTENSIF, Vol.6 No.2 August 2022
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

If the result obtained was matched during the matching process, then the algorithm returns the matched trigram string. However, when there was a trigram mismatched with the haystack, even if it was only one character, the algorithm returned the value "-1", meaning that there are no three adjacent words in the haystack that exactly match the trigram.

I. Performance Results of Algorithms

To measure the performance of the Horspool Boyer-Moore, Rabin-Karp, and Winnowing algorithms in identifying similarity based on the linguistic meaning using the word-level trigrams, the precision, recall and f-measures parameters were used. Based on the measurement, for the Horspool Boyer-Moore algorithms, it was found that there were 9 points of True Positive (TP), 10 points of True Negative (TN) points, but neither False Positive (FP) nor False Negative (FN) was found. Using the formula to calculate the precision and recall parameters, the results showed that the precision, the recall, and the f-measure values obtained were 100%, respectively. The Rabin-Karp algorithm also resulted from 9 points of True Positive (TP) and 10 points of True Negative (TN) issues. Still, neither False Positive (FP) nor False Negative (FN) was found so that the precision, the recall, and the f-measure values were 100%, respectively. The Winnowing algorithm with a window size of 2 resulted from 5 points of TP, 10 points of TN, 4 points of N) points and no FP was found by the algorithm; with a window size of 3 resulted from 2 points of TP, 10 points of TN, 6 points of FN points and still no FP was found; and with a window size of 4 resulted from only 1 point of TP, but the FN reached 8 points while the TN were still 10 points and the algorithm found no FP. Therefore, the precision, the recall, and the f-measure values obtained by the Winnowing algorithm with a window size of 2 were 100%, 56%, and 71%, respectively; with a window size of 3 were 100%, 25%, and 40%, respectively; and with a window size of 4 were 100%, 11%, and 20%, respectively (See Appendices section for more detail performance results of the algorithms).

In terms of the performance measurement using precision and recall parameters as presented in Figure 5, the Horspool Boyer-Moore performance was equal to the Rabin-Karp algorithm. In contrast, the Winnowing algorithm performance was lower in terms of sensitivity (recall).

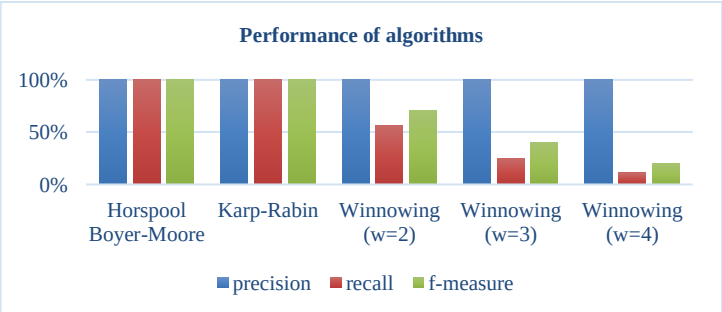


Figure 5. COMPARISON OF ALGORITHMS PERFORMANCE

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

It can be seen in Figure 5 that the wider the window of the Winnowing algorithm, the lower the sensitivity value. This was due to the effect of using windows on the determination of the fingerprints so that the wider the window width used in grouping hash values, the less the minimum hash value selected (for more detail, see Appendices section).

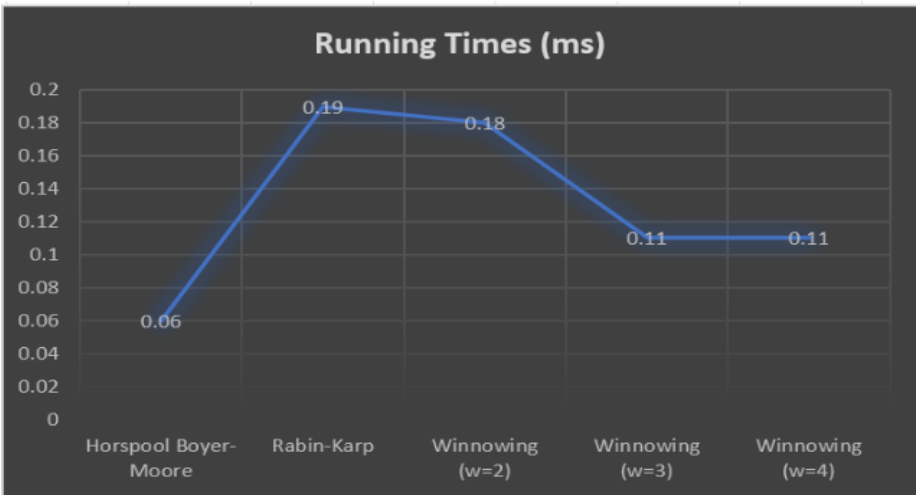


Figure 6. RUNNING TIMES OF ALGORITHMS

In identifying similarity using word-level trigrams, in terms of running time as presented in Figure 6, the Horspool Boyer-Moore took the shortest time, followed by the Rabin-Karp algorithm and then the Winnowing algorithm. The Horspool algorithm was faster than the other algorithms because of bad-character tables and more extended trigram patterns in identifying similarities. The bad-character table reduces the number of attempts of the algorithm to find matched trigrams. Moreover, using longer trigrams as patterns makes the pattern's sliding window shift further, making the performance faster when seeing the match trigrams. However, the running time of the two hash function-based algorithms is slightly different from the running time of Horspool Boyer-Moore because identification is made based on integers in the hash-based algorithms so that the processor only performs a single comparison for each hash value.

IV. CONCLUSION

Based on the research results regarding the identification of similarity using word-level trigrams in terms of precision, recall, and running time, it can be concluded that the Horspool Boyer-Moore algorithm is superior to other algorithms. The accuracy, recall, and running time of the Rabin-Karp algorithm were 100%, 100%, and 0.19 ms, respectively; the Winnowing algorithm with the smallest window was 100%, 56%, and 0.18 ms, respectively; and the Horspool algorithm were 100%, 100%, and 0.06 ms. Even though the three algorithms are exact string matching algorithms, the performance of each is still different. The Horspool Boyer-Moore algorithm requires a much faster time in the string matching process because of the

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

implementation of the bad-character table and the word trigrams in this algorithm so that the matching is only based on the skip value of each character pattern. The longer the practice is used in this algorithm, the further the sliding window of the procedure shifts forward when the mismatch is found. If there is a mismatch found in character, the sliding window of the algorithm will immediately skip as far as the skip value of the essence, making it more efficient. The results also show a significant difference between the performance of the Rabin-Karp algorithm and the Winnowing algorithm, even though both are based on hash functions caused by differences in the determination of fingerprints on the two algorithms. The use of word trigrams as the n-gram also affects the identification of similarity because the string matching between text patterns is based on the linguistic meaning of each sentence. Even if only one word is different, the three adjacent words of the aligned haystack will be considered as not similar. This research is open to further development using more extended word-gram units as n-gram and better algorithms in terms of resource cost and the running time so that it can be applied to large-scale databases and large documents.

REFERENCES

[1] I. Markić, M. Štula, M. Zorić, and D. Stipaničev, "Entropy-Based Approach in Selection Exact String-Matching Algorithms," Entropy, vol. 23, no. 1, pp. 1–19, Jan. 2021, doi: 10.3390/e23010031.

[2] R. K. Pandey and S. Taruna, "Prevalent exact string-matching algorithms in natural language processing: a review," in Journal of Physics: Conference Series, May 2021, vol. 1854, no. 1. doi: 10.1088/1742-6596/1854/1/012042.

[3] S. Hakak, A. Kamsin, P. Shivakumara, G. A. Gilkar, W. Z. Khan, and M. Imran, "Exact String Matching Algorithms: Survey, Issues, and Future Research Directions," IEEE Access, pp. 2169–3536, 2018.

[4] R. A. Baeza-Yates, "Algorithms for String Searching: A Survey," ACM SIGIR Forum, vol. 23, no. 3–4, pp. 34–58, Apr. 1989, doi: https://doi.org/10.1145/74697.74700.

[5] X. Duan, M. Wang, and J. Mu, "A plagiarism detection algorithm based on extended winnowing," in MATEC Web of Conferences, Oct. 2017, vol. 128. doi: 10.1051/matecconf/201712802019.

[6] S. D. Schleimer, "Winnowing: local algorithms for document fingerprinting," in Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, Jun. 2003, pp. 76–85.

[7] A. T. Wibowo, K. W. Sudarmadi, and A. M. Barmawi, "comparison between fingerprint and winnowing algorithm to detect plagiarism fraud on bahasa indonesia documents," in International Conference of Information and Communication Technology, 2013, pp. 128–133. doi: https://doi.org/10.1109/ICoICT.2013.6574560.

[8] B. Devore-Mcdonald and E. D. Berger, "Mossad: Defeating software plagiarism detection," in Proceedings of the ACM on Programming Languages, Nov. 2020, vol. 4, no. OOPSLA. doi: 10.1145/3428206.

[9] I. Widaningrum, D. Mustikasari, R. Arifin, and H. A. Pratiwi, "Evaluation of the accuracy of winnowing, rabin karp and knuth morris pratt algorithms in plagiarism detection applications," in Journal of Physics: Conference Series, May 2020, vol. 1517, no. 1. doi: 10.1088/1742-6596/1517/1/012093.

[10] M. Corman, "Why Should the Length of Your Hash Table Be a Prime Number?," Aug. 08, 2020. https://medium.com/swlh/why-should-the-length-of-your-hash-table-be-a-prime-number-760ec65a75d1 (accessed Apr. 14, 2022).

[11] E. Karimov, "Hash Table. In: Data Structures and Algorithms in Swift," Apress, Berkeley, CA, pp. 55–60, Mar. 2020, doi: https://doi.org/10.1007/978-1-4842-5769-2_7.

[12] X. Wang and L. Liu, "Image Encryption Based on Hash Table Scrambling and DNA Substitution," IEEE Access, vol. 8, pp. 68533–68547, 2020, doi: 10.1109/ACCESS.2020.2986831.

[13] M. Góngora-Blandón and M. Vargas-Lombardo, "State of the Art for String Analysis and Pattern Search Using CPU and GPU Based Programming," Journal of Information Security, vol. 03, no. 04, pp. 314–318, 2012, doi: 10.4236/jis.2012.34038.

[14] N. Horspool, "Practical fast searching in strings Cite this paper," Software:PracticeandExperience, vol. 10, pp. 501–506, 1980.

[15] R. Fitriyanto, A. Yudhana, and S. Sunardi, "Implementation SHA512 Hash Function And Boyer-Moore String Matching Algorithm For Jpeg/exif Message Digest Compilation," Jurnal Online Informatika, vol. 4, no. 1, p. 16, Sep. 2019, doi: 10.15575/join.v4i1.304.

[16] A. Thyab and A. Ajeeli, "Advanced Searching Algorithms and its Behavior on Text Structures," 2016. [Online]. Available: www.iiste.org

[17] M. Fisk, G. Varghese, and M. Gov, "Applying Fast String Matching to Intrusion Detection Applying Fast String Matching to Intrusion Detection," LA-UR-01-5459, Sep. 2021, doi: DOI:10.21236/ada406266.

[18] D. Hendrian, Y. Ueki, K. Narisawa, R. Yoshinaka, and A. Shinohara, "Permuted pattern matching algorithms on multi-track strings," Algorithms, vol. 12, no. 4, 2019, doi: 10.3390/a12040073.

[19] H. Min et al., "Pattern matching based sensor identification layer for an android platform," Wireless Communications and Mobile Computing, vol. 2018, 2018, doi: 10.1155/2018/4734527.

[20] E. Ahmed, A. Sorrour, M. Sobh, and A. Bahaa-Eldin, "A cloud-based malware detection framework," International Journal of Interactive Mobile Technologies, vol. 11, no. 2, pp. 113–127, 2017, doi: 10.3991/ijim.v11i2.6577.

[21] O. F. Rashid, Z. A. Othman, and S. Zainudin, "A Novel DNA Sequence Approach for Network Intrusion Detection System Based on Cryptography Encoding Method," International Journal on Advanced Science, Engineering and Information Technology, vol. 7, no. 1, 2017.

[22] Y. A. Gerhana, N. Lukman, A. F. Huda, C. N. Alam, U. Syaripudin, and D. Novitasari, "Comparison of search algorithms in Javanese-Indonesian dictionary application," Telkomnika (Telecommunication Computing Electronics and Control), vol. 18, no. 5, pp. 2517–2524, Oct. 2020, doi: 10.12928/TELKOMNIKA.v18i5.14882.

[23] A. P. U. Siahaan, R. Rahim, M. Aan, and D. Siregar, "K-Gram as a determinant of plagiarism level in rabin-karp algorithm," International Journal of Scientific & Technology Research, vol. 6, no. 7, pp. 350–353, 2017, [Online]. Available: www.ijstr.org

[24] A. D. Hartanto, A. Syaputra, and Y. Pristyanto, "Best parameter selection of rabin-Karp algorithm in detecting document similarity," in 2019 International Conference on Information and Communications Technology, ICOIACT 2019, Jul. 2019, pp. 457–461. doi: 10.1109/ICOIACT46704.2019.8938458.

[25] C. Supriyanto, S. Rakasiwi, and A. Syukur, "A Comparison of Rabin Karp and Semantic-Based Plagiarism Detection," in 3rd International Conferences on Soft Computing, Intelligent System and Information Technology, 2012, pp. 29–31. [Online]. Available: http://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v6i2.18141

[26] Y. Nurdiansyah, F. Nur Muharrom, and F. Firdaus, "Implementation of winnowing algorithm based k-gram to identify plagiarism on file text-based document," in MATEC Web of Conferences, Apr. 2018, vol. 164. doi: 10.1051/matecconf/201816401048.

[27] S. Sunardi, A. Yudhana, and I. A. Mukaromah, "Indonesia Words Detection Using Fingerprint Winnowing Algorithm," Jurnal Informatika, vol. 13, no. 1, Jan. 2019. doi: 10.26555/jifo.v13i1.a8452.

[28] D. B. Rahmawati, M. Luthfi Irfani, R. B. Purba, and I. Ranggadara, "Text Mining To Detect Plagiarism In E-Learning System Using Rabin Karp Algorithm," Iconic Research and Engineering Journals, vol. 3, no. 8, pp. 183-191, Feb. 2020.

[29] N. Ulinnuha, M. Thohir, D. C. R. Novitasari, A. H. Asyhar, and A. Z. Arifin, "Implementation of winnowing algorithm for document plagiarism detection," in Proceeding of EECSI , 2018, pp. 631–636.

[30] A. Yudhana, Sunardi, and I. A. Mukaromah, "Implementation of Winnowing Algorithm with Dictionary English-Indonesia Technique to Detect Plagiarism," International Journal of Advanced Computer Science and Applications, vol. 9, no. 5, pp. 183–189, 2018, [Online]. Available: www.ijacsa.thesai.org

[31] A. Aljohani and M. Mohd, "Arabic-English Cross-language Plagiarism using winnowing," Information Technology Journal, vol. 13, no. 14, pp. 2349–2355, 2014.

[32] R. Sutoyo et al., "Detecting documents plagiarism using winnowing algorithm and k-gram method," in 2017 IEEE International Conference on Cybernetics and Computational Intelligence, CyberneticsCOM 2017 - Proceedings, Mar. 2018, vol. 2017-November, pp. 67–72. doi: 10.1109/CYBERNETICSCOM.2017.8311686.

[33] M. Schonlau and N. Guenther, "Text mining using n-grams variables," The Stata Journal., vol. 17, no. 4, pp. 866–881, Dec. 2017.

[34] D. Wright, "Using word n-grams to identify authors and idiolects A corpus approach to a forensic linguistic problem," International Journal of Corpus Linguistics, vol. 22, no. 2, pp. 212–241, Sep. 2017, doi: https://doi.org/10.1075/ijcl.22.2.03wri.

[35] J. Alcañiz and J. Andrés, "Profiling Hate Spreaders using word N-grams Notebook for PAN at CLEF 2021," Sep. 2021. [Online]. Available: http://ceur-ws.org

[36] D. Johnson, V. Malhotra, and P. Vamplew, "More Effective Web Search Using Bigrams and Trigrams," Webology, vol. 3, no. 4, 2006.

[37] A. Violentyev, "Rolling Hash Function Tutorial Used by Rabin-Karp String Searching Algorithm," YouTube, Dec. 20, 2019.

[38] H. Jiang and S. J. Lin, "A Rolling Hash Algorithm and the Implementation to LZ4 Data Compression," IEEE Access, vol. 8, pp. 35529–35534, 2020, doi: 10.1109/ACCESS.2020.2974489.

[39] M. Paul and S. Jamal, "An improved SRL based plagiarism detection technique using Sentence ranking," in Procedia Computer Science, 2015, vol. 46, pp. 223–230. doi: 10.1016/j.procs.2015.02.015.