HASIL CEK_Rezki Ramdhani, Abdul Fadlil, Sunardi

by Rezki Ramdhani, Abdul Fadlil, Sunardi Winnowing; Rolling-hash; Word-level; Trigrams; Kes

Submission date: 15-Aug-2022 11:15AM (UTC+0700)
Submission ID: 1882622398
File name: an_Word-Level_Triagrams_untuk_Mengidentifikasi_Kesamaan_Kata.pdf (480.68K)
Word count: 5558
Character count: 35416

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Penerapan Algoritma Winnowing dan Word-Level Trigrams Untuk Mengidentifikasi Kesamaan Kata

Rezki Ramdhani[1,*], Abdul Fadlil[2], Sunardi[2]

[1,*] Program Studi Magister Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia,
[2] Program Studi Teknik Elektro, Universitas Ahmad Dahlan, Yogyakarta, Indonesia,
Email: [1,*]rezki2008048036@webmail.uad.ac.id, [2]fadlil@mti.uad.ac.id, [3] sunardi@mti.uad.ac.id
Email Penulis Korespondensi: rezki2008048036@webmail.uad.ac.id
Submitted **21-04-2022**; Accepted **25-04-2022**; Published **29-04-2022**

Abstrak

Pencarian kata-kata yang sama pada dua teks atau lebih merupakan langkah awal dari proses deteksi tindakan plagiarisme. Software pendeteksi plagiarisme yang tersedia secara komersial relatif mahal. Meskipun ada beberapa software yang ditawarkan secara gratis, namun fitur yang disediakan sangat terbatas. Oleh karena itu, dibutuhkan sistem deteksi kesamaan kata yang dapat dijadikan alternatif bagi pengguna yang dapat diakses dengan lebih leluasa. Penerapan metode pencocokan pola (pattern matching) merupakan salah satu solusi yang dapat digunakan untuk menemukan kesamaan kata pada teks. Terdapat beberapa algoritma yang dapat digunakan sebagai metode untuk menemukan kesamaan kata pada teks diantaranya adalah algoritma Winnowing yang dikenal memiliki performa yang baik dalam mendeteksi kesamaan kata. Winnowing merupakan algoritma berbasis hashing-approach yang menerapkan hash-function dan pembentukan window untuk memperoleh fingerprints pada saat pencocokan pola. Berdasarkan fingerprints tersebut, tingkat kesamaan kata dapat dihitung. Penelitian-penelitian sebelumnya hanya menghitung tingkat kesamaan kata berdasarkan karakter (character-level), sedangkan perhitungan tingkat kesamaan berdasarkan kata (word-level) masih jarang dilakukan. Penelitian ini dilaksanakan dengan tujuan untuk mengukur tingkat kesamaan kata menggunakan algoritma Winnowing dan word-level trigrams. Hasil penelitian menunjukkan bahwa algoritma Winnowing yang diterapkan menggunakan word-level trigrams dapat mendeteksi kesamaan pada teks sebesar 76,84%, 52,29%, 37,40%, dan 19,29%. Dari hasil penelitian dapat disimpulkan bahwa metode pencocokan pola dengan algoritma Winnowing dan word-level trigrams dapat digunakan untuk mengukur tingkat kesamaan teks.

Kata Kunci: Winnowing; Rolling-Hash; Word-Level; Trigrams; Kesamaan

Abstract

Identifying the same words in two or more texts is the first step in the process of detecting plagiarism. Plagiarism detection software are commercially available but relatively expensive. Although some software is offered for free, the features provided are very limited. Therefore, a word similarity detection system is needed to be used as an alternative for users that can be freely accessed. The application of the pattern matching method is one of the solutions that can be used to find the similarity of words between documents. There are several algorithms that can be used as a method to find the similarity of words in the text, including the Winnowing algorithm which is known to have good performance in detecting similarity of words. Winnowing is a hashing-approach based algorithm that applies hash-function and window formation to obtain fingerprints during pattern matching. Based on these fingerprints, the word similarity level can be calculated. Previous studies have only calculated the level of similarity of words based on the character (character-level), while the calculation of the level of similarity based on words (word-level) is still limited. This research was carried out with the aim of measuring the level of similarity of words using the Winnowing algorithm and word-level trigrams. The results showed that the Winnowing algorithm which was applied using word-level trigrams could detect similarities in the text of 76.84%, 52.29%, 37.40%, and 19.29%, respectively. From the results of the study, it can be concluded that the pattern matching method with the Winnowing algorithm and word-level trigrams can be used to measure the level of similarity of the text.

Keywords: Winnowing; Rolling-Hash; Word-Level; Trigrams; Similarity

1. PENDAHULUAN

Plagiarisme merupakan tindakan yang bertentangan dengan nilai moral dan etika dalam bidang informatika namun lazim ditemukan di dalam dunia pendidikan sehingga dibutuhkan sistem yang dapat mengidentifikasi kesamaan kata antara satu dokumen dengan dokumen lainnya. Pendeteksian yang dilakukan juga diharapkan tidak hanya terbatas pada identik tidaknya isi dua dokumen atau lebih, melainkan juga perlu mempertimbangkan makna dari kalimat yang dianggap sama, karena kalimat *"saya ingin makan"* dan *"saya ingin malam"* sangatlah berbeda dari segi makna meskipun yang berbeda pada kedua kalimat ini hanya terletak pada tiga huruf di akhir kata terakhir.

Identifikasi pola yang yang sama pada dua atau lebih dokumen merupakan langkah awal yang dapat diterapkan dalam proses deteksi tindakan plagiarisme pada karya ilmiah. Terdapat beberapa algoritma yang dapat digunakan untuk menemukan kesamaan antar dokumen [1], [2]. Algoritma *Winnowing* adalah salah satunya, dimana algoritma ini menggunakan pendekatan *hashing-approach* approach yang menerapkan *hash-function* dan pembentukan *window* untuk memperoleh *fingerprints* dengan nilai *hash* minimum pada saat pencocokan pola [3], [4]. Algoritma ini bekerja dengan cara mengubah isi dokumen tersebut kedalam nilai-nilai *hash* (*hash values*) untuk kemudian dicari nilai-nilai yang sama. Nilai-nilai yang sama tersebut mewakili kata-kata yang sama pada kedua dokumen. Terdapat beberapa hasil penelitian terdahulu yang menunjukkan bahwa algoritma *Winnowing* memiliki performa yang baik dalam mendeteksi kesamaan kata pada dokumen-dokumen karya ilmiah, diantaranya diuraikan sebagai berikut.

Penelitian dilakukan oleh Yudhana, Sunardi, & Mukaromah (2018) mengenai deteksi kesamaan kata menggunakan algoritma *Winnowing* yang dipadukan dengan fitur *Dictionary English-Indonesia*. Teknik pra-pemrosesan meliputi tahap *case folding* dan *tokenizing* tanpa melakukan *filtering* maupun *stemming* pada teks. *Hash function*

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

menggunakan rumus *rolling hash*. Pembentukan *gram* menggunakan *character-level trigrams* dan pengukuran similaritas menggunakan rumus *Jaccard Coefficient*. Hasil penelitian menunjukkan bahwa algoritma *Winnowing* dapat mendeteksi kesamaan teks dengan hasil sebesar 45,21% [5].

Penelitian dilakukan oleh Sutoyo, dkk. (2017) untuk mendeteksi kesamaan kata pada beberapa dokumen menggunakan algoritma *Winnowing* dan *k-gram* berbasis karakter. Teknik pra-pemrosesan yang digunakan adalah *case folding*, *tokenizing*, dan *filtering*. *Hash function* menggunakan *Rolling Hash*. Pembentukan *gram* menggunakan *character-level 20-grams* dan 25-*grams*. Hasil penelitian menunjukkan bahwa algoritma Winnowing dapat digunakan untuk mendeteksi kesamaan kata pada beberapa dokumen yang diuji dengan parameter 100 – 1000 kata. Akurasi dari pendeteksian dengan character-level 25-grams adalah 7% sedangkan akurasi character-level 20-grams adalah 4% [6].

Penelitian dilakukan oleh Sunardi, Yudhana, & Mukaromah (2019) mengenai deteksi kesamaan teks berbahasa Indonesia menggunakan algoritma *Winnowing*. Teknik pra-pemrosesan yang digunakan adalah *case folding*, *tokenizing*, *filtering*, dan *stemming* dengan *stemmer* Nazief-Andriani. *Hash function* menggunakan *Rolling Hash*. Pembentukan *gram* menggunakan *character-level trigrams*. Pengukuran similaritas menggunakan *Jaccard Coefficient*. Hasil penelitian menunjukkan bahwa algoritma *Winnowing* dengan *character-level trigrams* dapat mendeteksi kesamaan kata dengan hasil sebesar 23,53% [7].

Penelitian dilakukan oleh Nurdiansyah, Muharom, & Firdaus (2017) untuk mengimplementasikan algoritma *Winnowing* untuk mengidentifikasi plagiarisme pada dokumen berbasis teks dengan tipe file *.txt*, *.doc,* dan *.docx*. Teknik pra-pemrosesan yang digunakan adalah *case folding*, *tokenizing*, *filtering*, dan *stemming* dengan *stemmer* Sastrawi. *Hash function* yang digunakan adalah *Rolling Hash*. Pembentukan *gram* menggunakan *character-level 7-grams*. Pengukuran similaritas menggunakan *Jaccard Coefficient*. Hasil penelitian menunjukkan bahwa persentase *error* menggunakan algoritma *Winnowing* berdasarkan karakter tersebut dalam mendeteksi similaritas adalah sebesar 0,69% [8].

Penelitian dilakukan oleh Duan, Wang, & Mu (2017) mengenai deteksi plagiarisme menggunakan algoritma *Extended Winnowing*. Inputan berupa dokumen berisi aksara Cina (*Han*), yang terdiri dari 1827 dokumen diperbandingkan dengan 3230 dokumen sumber. Pembentukan *gram* tidak disebutkan apakah menggunakan *character-level* ataupun word-level dengan jumlah unit disesuaikan dengan panjang *t threshold*. Hasil menunjukkan bahwa penggunaan algoritma *Extended Winnowing* untuk memeriksa 1 file berukuran 100 KB atau sekitar 100.000 karakter rata-rata membutuhkan waktu 1 detik. Algoritma *Extended Winnowing* mampu memberikan informasi posisi/lokasi dari teks yang diduga plagiat sehingga meningkatkan kemampuan algoritma *Winnowing* biasa [9].

Penelitian dilakukan oleh Ulinnuha, dkk. (2018) mengenai deteksi plagiarisme menggunakan algoritma *Winnowing* pada 100 dokumen abstrak. *Hash function* yang digunakan adalah *Rolling Hash*. Teknik pra-pemrosesan mencakup *case folding* dan *tokenizing*. Pembentukan gram menggunakan *character-level 7-grams*. Pengukuran similaritas menggunakan *Jaccard Coefficient*. Hasil penelitian menunjukkan bahwa sistem mampu mendeteksi plagiarisme dengan sensitivitas 82%, spesifisitas 100%, dan akurasi 91% [10].

Penelitian dilakukan oleh Sugiono, dkk. (2016) mengenai deteksi tingkat kesamaan dokumen teks menggunakan algoritma *Rabin-Karp* dan algoritma *Winnowing* dalam mendeteksi kesamaan teks pada judul skripsi dengan perbandingan antara dua variabel, yaitu kapabilitas dalam mendeteksi kesamaan teks dan waktu yang dibutuhkan. Teknik pra-pemrosesan yang digunakan adalah *case folding* dan *tokenizing*. Pembentukan *gram* menggunakan *character-level 5-grams*. Hasil penelitian menunjukkan bahwa kedua algoritma dapat mendeteksi kesamaan teks dengan akurasi 37,5% pada algoritma *Rabin-Karp* dan 88,89% pada algoritma *Winnowing* dengan waktu proses berturut-turut 0,19 detik dan 0,13 detik sehingga algoritma *Winnowing* lebih unggul dari segi kapabilitas maupun waktu pemrosesan [11].

Penelitian dilakukan oleh Sibarani, Magdalena, & Dharma (2019) mengenai perbandingan hasil deteksi kemiripan judul skripsi menggunakan algoritma *Winnowing* dan algoritma *Rabin-Karp*. Teknik pra-pemrosesan yang digunakan adalah *case folding* dan *tokenizing*. Pembentukan gram menggunakan *character-level 5-grams*. *Hash function* yang digunakan adalah *Rolling Hash*. Hasil penelitian menunjukkan akurasi kedua algoritma mendekati 75% [12].

Penelitian dilakukan oleh Alamsyah (2017) untuk membandingkan algoritma *Winnowing* dengan algoritma *Rabin-Karp* untuk mendeteksi plagiarisme pada kemiripan teks judul skripsi. Teknik pra-pemrosesan yang digunakan adalah *case folding* dan *tokenizing*. Pembentukan *gram* menggunakan *character-level* dengan unit bervariasi mulai dari 2-*grams* hingga 9-*grams*. *Hash function* yang digunakan adalah *Rolling Hash*. Pengukuran similaritas pada algoritma *Winnowing* menggunakan *Dice's Coefficient*, sedangkan pada algoritma *Rabin-Karp* menggunakan *Jaccard Coefficient*. Hasil penelitian menunjukkan performa algoritma *Winnowing* lebih baik daripada algoritma *Rabin-Karp* dengan waktu proses 0,0257 detik pada panjang k=9 [13].

Penelitian dilakukan oleh Widaningrum, dkk. (2019) mengenai algoritma *Winnowing*, *Rabin-Karp*, dan KMP dalam mendeteksi plagiarisme. Teknik pra-pemrosesan yang digunakan adalah *case folding* dan *tokenizing*. Pembentukan gram pada algoritma *Winnowing* menggunakan *character-level 5-grams*. Fungsi *Rolling Hash* digunakan untuk membuat nilai *hash* dari setiap *gram* yang terbentuk. *Jaccard Coefficient* digunakan untuk menghitung nilai similaritas pada algoritma *Winnowing* dan *Dice's Coefficient* pada algoritma *Rabin-Karp*. Hasil penelitian menunjukkan algoritma *Winnowing* lebih unggul daripada algoritma *Rabin-Karp* dan algoritma KMP karena algoritma *Winnowing* dapat mendeteksi plagiarisme yang kompleks [14].

Dari hasil-hasil penelitian tersebut dapat diketahui bahwa algoritma *Winnowing* memiliki performa yang baik, namun penelitian-penelitian sebelumnya hanya menghitung tingkat kesamaan kata berdasarkan karakter (*character-level*), sedangkan perhitungan tingkat kesamaan berdasarkan kata (*word-level*) masih jarang dilakukan padahal pendeteksian berbasis kata diperlukan untuk mengindetifikasi kalimat berdasarkan maknanya agar hasil deteksi tidak

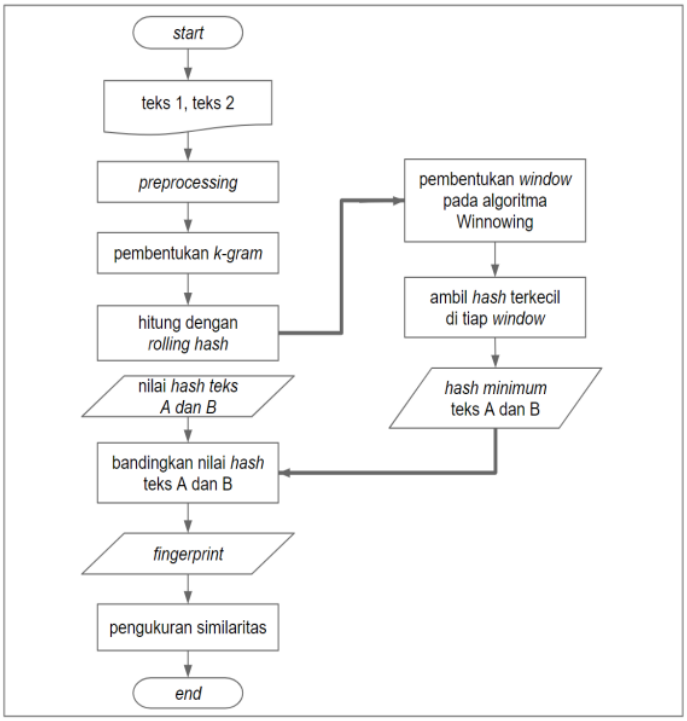
JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

hanya bertumpu pada kesamaan huruf-huruf di dalamnya. Oleh karena itu, penelitian ini dilaksanakan dengan tujuan untuk mengukur tingkat kesamaan dua dokumen menggunakan algoritma *Winnowing* dan *word-level trigrams*. Melalui penelitian ini diharapkan sistem yang ditawarkan dapat mengukur tingkat kesaaman kata tidak hanya berdasarkan karakter namun menjadikan makna kalimat sebagai faktor utama yang dipertimbangkan dalam memutuskan dua teks identik atau tidak.

2. METODOLOGI PENELITIAN

2.1 Alur Kerja Algoritma *Winnowing*

Proses deteksi plagiarisme menggunakan algoritma *Winnowing* secara umum diilustrasikan pada Gambar 1.



Gambar 1. Alur identifikasi kesamaan kata menggunakan algoritma *Winnowing* dan *character-level*.

Berdasarkan Gambar 1 diketahui bahwa inputan yang dibutuhkan adalah dua *strings* yang dapat berupa teks pendek maupun teks panjang. Tahap-tahap dalam mengidentifikasi tingkat kesamaan dua dokumen menggunakan algoritma *Winnowing* meliputi pra-pemrosesan data, pembentukan *grams*, pembentukan nilai *hash* dengan *Rolling Hash*, pembentukan *window*, kemudian dilanjutkan dengan pengukuran similaritas. Penjelasan mengenai tahap-tahapan penggunaan algoritma *Winnowing* pada penelitian ini adalah sebagai berikut.

2.1.1 Pra-pemrosesan Data (*preprocessing*)

Tahap pra-pemrosean data merupakan tahapan yang dilakukan untuk membersihkan data sebelum diolah. Tahapan ini dapat berupa *case folding* yakni mengubah seluruh huruf ke huruf kecil secara seragam; *tokenizing* yakni mengubah teks kedalam potongan-potongan subteks (*sub-strings*); *stemming* yakni mengubah setiap subteks ke kata dasarnya (*root word*) maupun *filtering* yakni menghilangkan kata-kata atau simbol-simbol yang tidak memiliki makna dalam kalimat [7], [8].

2.1.2 Pembentukan *Grams*

Suatu kata terdiri dari rangkaian huruf dan suatu kalimat terdiri dari rangkaian kata. Untuk mencari kesamaan pada suatu kata atau kalimat maka kata dapat diubah kedalam potongan-potongan kecil yang kemudian disebut sebagai *gram* (atau disebut juga dengan "*token*") untuk mempermudah kata tersebut dikonversi kedalam bilangan biner terutama untuk mendapatkan nilai-nilai *hash*. Istilah *k-gram* digunakan untuk menunjukkan rangkaian *substring* yang saling berdekatan, dengan *k* adalah panjang *gram* yang ditentukan oleh pengguna [4]. Untuk memudahkan memahami makna *k-gram* maka diberikan contoh penerapan *k-gram* dengan panjang gram k=3 pada kalimat berikut:

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

"sayabelajarjavascriptsejaksetahunlalusekarangsudahbisa"

Jika pemeriksaan dilakukan menggunakan basis karakter (*character-level*) maka sebelum diubah kedalam *gram* pada umumnya kalimat dihilangkan tanda spasinya terlebih dahulu sehingga *gram* yang terbentuk adalah *["say", "aya", "yab", "abe", "bel", ..., "isa"]*. Sedangkan pembentukan *gram* berbasis kata (*word-level*), gram yang terbentuk adalah *["saya belajar javascript", "belajar javascript sejak", "javascript sejak setahun", "sejak setahun lalu", "setahun lalu sekarang", "lalu sekarang sudah", "sekarang sudah bisa", "sudah bisa", "bisa"]*

Pada pembentukan gram, dikenal istilah "*unigram*", "*bigram*", "*trigram*" dan seterusnya. *Unigram* menunjukkan bahwa panjang *gram* adalah satu karakter atau satu kata, *bigram* menunjukkan panjang *gram* adalah dua karakter atau dua kata, dan demikian seterusnya. Pada bidang ilmu pemodelan bahasa atau yang disebut dengan linguistik komputasi, pembentukan rangkaian *gram* lebih umum menggunakan kata sebagai *gram* atau tokennya yang disebut dengan "*n-gram*" (atau dikenal juga dengan nama "*q-gram*"). Satuan *n* dapat berupa karakter maupun kata tergantung pada objek yang sedang diteliti. Satuan berupa karakter umumnya digunakan pada pencocokan DNA atau Protein (*DNA Sequencing* atau *Protein Sequencing*). Satuan berupa kata biasanya diterapkan bidang ilmu yang berkaitan dengan linguistik seperti *natural language processing, text mining* [15], [16], [17] karena lebih mudah diterapkan dalam proses pengolahan bahasa untuk menghitung probabilitas kata berdasarkan maknanya. Dengan menggunakan dua atau tiga satuan gram (*bigram* atau *trigram*) maka kata akan membentuk frasa sehingga kata-kata akan memiliki makna. Sebagai contoh pada rangkaian trigrams *"saya belajar javascript"* memiliki makna yang sangat berbeda dengan rangkaian trigrams *"sekarang belajar bersepeda"*.

Tabel 1. Contoh pembentukan *grams* dengan *word-level* dan *character-level*

Contoh Kata	Satuan	Unigram	Bigrams	Trigrams
saya belajar javascript.	Karakter	"s", "a", "y", "a", "b", "e", "l", "a", "j", "a", "r", "j", "a", "v", "a", "s", "c", "r", "i", "p", "t"	"sa", "ay", "ya", "ab", "be", "el", "la", "aj", "ja", "ar", "rj", "ja", "av", "va", "as", "sc", "cr", "ri", "ip", "pt"	"say", "aya", "yab", "abe", "bel", "ela", "laj", "jar", "arj", "rja", "jav", "avs", "vsc", "scr", "cri", "ipt".
saya belajar javascript.	Kata	Saya, belajar, javascript	"saya belajar" "belajar javascript"	"saya belajar javascript"

Tabel 1 menunjukkan proses pembentukan *grams* berbasis kata (*word-level*) dan pembentukan *grams* berbasis karakter (*character level*) dengan panjang *n* yang bervariasi yaitu *unigram* (*n*=1), *bigrams* (*n*=2), dan *trigrams* (*n*=3). Perbedaaan *grams* yang terbentuk terletak pada pemecahan *strings* berupa teks ke dalam *substrings* yang saling berdekatan. Pada pembentukan *grams* menggunakan satuan karakter maka teks *strings* dipecah berdasarkan huruf sedangkan pada satuan kata maka pemecahan teks *strings* berdasarkan kata.

2.1.3 Pembentukan *Hash Values*

Pada tahapan pembentukan nilai-nilai *hash* menggunakan *Rolling Hash*, kedua teks yang akan dibandingkan terlebih dahulu dikonversi kedalam bentuk *integer* dengan memanfaatkan kode ASCII dari setiap karakter tersebut. Namun jika hanya mengandalkan konversi ke dalam ASCII untuk mendapatkan nilai-nilai *hash* maka akan terjadi *collision*, yakni keadaan dimana dua nilai *hash* memiliki susunan huruf yang sama akibat persebaran *key* yang tidak merata pada *Hash table*. Hash table merupakan struktur data yang menyerupai array untuk penyimpanan data dalam bentuk *keys* and *values* [18]. Tujuan dari hash table adalah untuk mempercepat pencarian kembali dari banyak data yang disimpan Sebagai contoh susunan huruf "*abcd*" memiliki jumlah nilai ASCII yang sama dengan susunan huruf "*bcda*" sehingga nilai *hash* yang dihasilkan akan sama. Untuk menghindari *collision* karena urutan posisi karakter dari *string* yang teracak (permutasi) seperti ini maka solusi yang dapat dilakukan adalah mengalikan *hash value* tersebut dengan suatu nilai konstan yang dipangkatkan dengan panjang *n gram* sehingga pemeriksaan karakter dapat dilakukan dari arah sebaliknya, yaitu dari kanan ke kiri. Nilai basis sebaiknya menggunakan bilangan prima yang lebih besar dari kode ASCII semua huruf alfabet karena semakin besar bilangan prima maka semakin merata persebaran nilai *hash* di dalam *hash table* [19], [20], [21]
Rumus *Rolling Hash* dapat dituliskan dengan pemodelan matematika sebagai berikut [10]:

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

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

dimana:
$H_{(C1..Ck)}$ = nilai *hash* awal
C_k = nilai ASCII dari tiap karakter (C_0 ... $C_{(k-1)}$)
k = panjang *gram/token*
b = nilai basis (konstanta)

Berdasarkan rumus *Rolling Hash* tersebut maka pembentukan nilai hash $H_{(C1..Ck)}$ pada seri kata *"saya belajar javascript"* adalah sebagai berikut:

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

$$Hc_1..c_k = 430\ x\ 193^{3-1} + 721\ x\ 193^{3-2} + 1079\ x\ 193^{3-3} = 16157302$$
$$Hc_1..c_{k+1} = (16157302 - (430\ x\ 193^{3-1})\ x\ 193) + 526 = 27065302$$

Pada Tabel 2 disajikan hasil pembentukan nilai *hash* pada setiap *gram*. Nilai-nilai *hash* yang terbentuk disebut dengan *fingerprints*. Perlu diingat bahwa rumus *Rolling Hash* ini harus diterapkan pada kedua teks yang akan dibandingkan agar *fingerprints* dari masing-masing teks dapat dicari nilai yang sama. Namun, sebelum mencari nilai *fingerprints* yang sama, tahapan pembentukan *window* perlu dilakukan terlebih dahulu.

Tabel 2. Pembentukan nilai *hash* pada setiap *gram*

No	Seri *word-level trigrams*	Nilai Ascii per kata	Nilai *hash*
1.	"saya belajar javascript"	[430, 721, 1079]	16157302
2.	"belajar javascript sejak"	[721, 1079, 526]	27065302
3.	"javascript sejak setahun"	[1079, 526, 760]	40293949
4.	"sejak setahun lalu"	[526, 760, 430]	19740084
5.	"setahun lalu sekarang"	[760, 430, 844]	28393074
6.	"lalu sekarang sudah"	[430, 844, 533]	16180495
7.	"sekarang sudah bisa"	[844, 533, 415]	31541440
8.	"sudah bisa"	[533, 415]	19933812
9.	"bisa"	[415]	15458335

2.1.4 Pembentukan Window

Pembentukan *window* dilakukan untuk mencari nilai *hash* terkecil (*minumum hash*) di setiap jendela [4]. Banyaknya deretan *hash-values* di setiap jendela tergantung dari lebar *window* (w) yang ditetapkan. Pada penelitian ini, lebar *window* yang digunakan adalah $w = 4$ yang berarti pada setiap jendela terdapat empat nilai *hash*. Dengan demikian, *window* yang terbentuk dari *grams* menggunakan *word-level trigrams* tersebut adalah sebagai berikut:

$[\ [\ 16157302, 27065302, 40293949, 19740084\ ], (window\ I)$
$[\ 27065302, 40293949, 19740084, 28393074\ ], (window\ II)$
$[\ 40293949, 19740084, 28393074, 16180495\ ], (window\ III)$
$[\ 19740084, 28393074, 16180495, 31541440\ ], (window\ IV)$
$[\ 28393074, 16180495, 31541440, 19933812\ ], (window\ V)$
$[\ 16180495, 31541440, 19933812, 15458335\ ]\ (window\ VI)\ ]$

Pada setiap *window* diambil nilai *hash* terkecil sehingga hasil yang didapatkan adalah sebagai berikut:

[16157302, 19740084, 16180495, 15458335]

Pembentukan *window* harus dilakukan pada setiap teks yang akan dibandingkan karena kumpulan nilai *hash* minimum dari setiap teks tersebut yang akan dibandingkan untuk dihitung tingkat kesamaan atau similaritasnya.

2.1.5 Perhitungan Similaritas

Tingkat kesamaan kata dapat diihitung menggunakan rumus *Jaccard Coefficient*. Sebelum menggunakan rumus tersebut, nilai *hash* terkecil dari teks asli dengan teks uji harus dibandingkan terlebih dahulu agar didapatkan nilai-nilai *fingerprints* yang sama sehingga jika pada teks asli yang berisi teks *"saya belajar javascript sejak setahun lalu sekarang sudah bisa"* memiliki nilai *hash* terkecil:

[16157302, 19740084, 16180495, 15458335]

Dan pada teks uji yang berisi teks *"saya belajar golang sejak setahun lalu sekarang sudah bisa"* memiliki nilai *hash* terkecil:

[16156855, 19740084, 16180495, 15458335]

Maka nilai-nilai *fingerprints* yang sama dari kedua teks ini adalah:

[19740084, 16180495, 15458335]

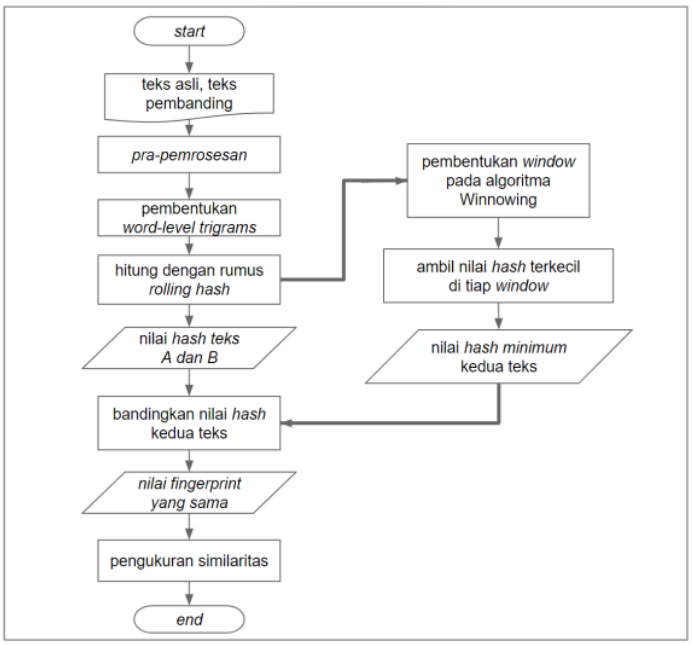
Persentase kemiripan diukur menggunakan rumus *Jaccard Coefficient* dengan membagi jumlah *fingerprints* yang sama dengan jumlah total nilai *hash* pada teks asli dan total nilai hash teks uji dikurangi dengan jumlah *fingerprints* yang sama yang dapat dimodelkan dengan persamaan berikut [13], [8], [10]:

$$Similaritas = \frac{total\ fingerprints\ yang\ sama}{(total\ hash\ dokumen\ asli + total\ hash\ dokumen\ uji) - fingerprints\ yang\ sama}\ x\ 100\ \% \qquad (3)$$

2.2 Alur Kerja Algoritma *Winnowing* dengan *Word-Level Trigrams*

Berdasarkan penjelasan mengenai tahap-tahapan penggunaan algoritma *Winnowing* berbasis karakter yang secara umum dilakukan untuk mendeteksi kesamaan kata tersebut, maka pada penelitian ini dapat diilustrasikan penggunaan algoritma *Winnowing* berdasarkan *word-level trigrams* dalam mengidentifikasi kesamaan kata sebagai berikut.

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 2. Alur identifikasi kesamaaan kata dengan algoritma *Winnowing* dan *word-level trigrams*

Gambar 2 mengilustrasikan proses identifikasi kesamaan kata menggunakan algoritma *Winnowing* dengan pembentukan *grams* menggunakan *word-level trigrams* pada masing-masing teks yang akan dibandingkan. Dengan demikian, pembentukan grams tidak lagi diperoleh dari pemecahan teks string berdasarkan huruf per huruf yang saling bersisian (*character-level*), melainkan *grams* diperoleh berdasarkan tiga kata yang saling bersisian (*word-level trigrams*).

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Algoritma *Winnowing* pada teks

Pada penelitian ini, tahap pra-pemrosesan data meliputi *case folding* untuk mengubah seluruh isi teks kedalam huruf kecil (*lowercase*) dan *filtering* dengan menghilangkan tanda simbol [+-] yang terdapat di dalam teks. Tahap *filtering* pada kata-kata tidak dilakukan dengan tujuan untuk mempermudah proses pendeteksian kesamaan berdasarkan makna kalimatnya. Dengan demikian, tanda spasi dibiarkan tetap ada (*case sensitivity ignored*), proses *stemming* atau pengembalian kata ke akar katanya juga tidak dilakukan sehingga kedua teks dibiarkan sesuai makna aslinya.

Setelah melalui tahap pra-pemrosesan, teks asli maupun teks pembanding kemudian diubah kedalam potongan-potongan *gram* menggunakan *word-level trigrams*. Setelah diubah kedalam bentuk *grams*, fungsi *Rolling Hash* diterapkan kepada setiap *gram* untuk mendapatkan nilai-nilai *hash* (*fingerprints*) dari kedua teks yang akan dibandingkan. Sebelum mengidentifikasi nilai-nilai *hash* yang sama diantara kedua teks, seluruh *fingerprints* pada masing-masing teks dikelompokkan kedalam *window* untuk mendapatkan nilai *hash* minimum. Pada penelitian ini lebar *window* yang digunakan adalah $w = 4$.

3.2 Hasil Pengujian

Berikut adalah teks yang diuji pada penelitian ini. Teks berisi 10 kalimat yang akan diuji dengan beberapa parameter yaitu tingkat pengubahan kalimat sebesar 20%, 40%, 60%, dan 80%.

> *searching for the same words in two or more documents is the first step in the process of detecting plagiarism in scientific works. this is very important because plagiarism is a violation of intellectual property rights, especially in the field of education. plagiarism is compiling a work by citing the work of others that has been made without adding citations to the quoted sentence. however, checking large numbers of documents manually will not be effective and efficient, while commercially available plagiarism detection software is relatively expensive. although some software is offered for free, the features provided are very limited. therefore, a pattern matching method is needed to make it easier to find the same words in the documents to be checked as a first step in detecting plagiarism. there are many algorithms that can be used as a method to find the similarity of words in documents, including the winnowing algorithm which is known to have good performance in detecting similarity of words.*

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

winnowing algorithm is a hashing approach based algorithm that applies hash function and window formation to obtain fingerprints during pattern matching. based on these fingerprints, the word similarity level between two documents can be calculated. previous studies have only calculated the level of similarity of words based on the character or character level, while the calculation of the level of similarity based on words or word level is still rarely done.

Setelah algoritma *Winnowing* dengan pembentukan *grams* menggunakan *word-level trigrams* diterapkan pada masing-masing teks maka diperoleh total *fingerprints* pada teks asli sebanyak 234 *fingerprints* dan total *fingerprints* pada masing-masing teks pembanding berturut-turut sebanyak 231, 227, 226, dan 224 *fingerprints* seperti dapat dilihat pada Tabel 3. Perbedaan jumlah *fingerprints* disebabkan oleh perbedaan *grams* yang terbentuk pada masing-masing teks. Pada saat pembentukan jendela (*windows*), terdapat perbedaan jumlah *windows* antara teks asli dengan teks-teks pembanding disebabkan oleh jumlah *fingerprints* dari masing-masing teks yang telah mengalami perubahan berdasarkan besar parameter yang ditetapkan.

Tabel 3. *Grams* dan *fingerprints* yang terbentuk pada teks asli dan teks yang dibandingkan

Inputan	Total *Grams*	Total *Fingerprints*	Total *Windows*
Teks Asli	234	234	84
Teks Pembanding 1	231	231	84
Teks Pembanding 2	227	227	82
Teks Pembanding 3	226	226	85
Teks Pembanding 4	224	224	83

Pembentukan *window* sangat penting dalam penerapan algoritma *Winnowing* karena *window* digunakan untuk mencari nilai *hash* minimum dari setiap jendela. Pada setiap kelompok *fingerprints* yang berada di dalam satu *window* terdapat nilai paling kecil diantara nilai-nilai *hash* lainnya dalam satu *window* tersebut. Nilai-nilai paling kecil dari kumpulan *window* teks asli dan teks-teks pembanding diambil untuk dihitung tingkat similaritasnya. Tabel 4 menunjukkan hasil perhitungan tingkat similaritas berdasarkan nilai *fingerprints* yang sama antara teks asli dengan teks-teks pembanding.

Tabel 4. Similaritas berdasarkan nilai *fingerprints* yang sama

Percobaan	Parameter	Fingerprints yang sama		Similaritas
		Teks Asli	Teks Pembanding	
Percobaan 0	Diubah sebesar 0%	84	84	100%
Percobaan 1	Diubah sebesar 20%	84	73	76,84%
Percobaan 2	Diubah sebesar 40%	84	57	52,29%
Percobaan 3	Diubah sebesar 60%	84	46	37,40%
Percobaan 4	Diubah sebesar 80%	84	27	19,29%

3.3 Analisis

Data hasil percobaan yang disajikan pada Tabel 4 dapat dijabarkan sebagai berikut.

a. Pada percobaan pertama, pada teks asli dilakukan perubahan makna sebesar 20% sehingga tingkat similaritas menjadi 76,84%. Hal ini dipengaruhi oleh perubahan yang terjadi pada teks pembanding yang mengalami perubahan isi sebesar 20% menyebabkan *fingerpints* juga berkurang hingga sekitar 20% yaitu menjadi 73 *fingerprints* saja yang sama dengan *fingerprints* teks asli. Perbedaan tingkat kesamaan *fingerprints* menyebabkan tingkat similaritas menjadi 76,84% saja.
b. Pada percobaan kedua, pada teks asli dilakukan perubahan makna sebesar 40% sehingga tingkat similaritas menjadi 52,29%. Hal ini dipengaruhi oleh perubahan yang terjadi pada teks pembanding yang mengalami perubahan isi sebesar 40% menyebabkan *fingerpints* juga berkurang hingga sekitar 40% menjadi yaitu 57 *fingerprints* sama dengan *fingerprints* teks asli. Perbedaan tingkat kesamaan *fingerprints* menyebabkan tingkat similaritas menjadi 52,29% saja.
c. Pada percobaan ketiga, pada teks asli dilakukan perubahan makna sebesar 60% sehingga tingkat similaritas menjadi 37,40%. Hal ini dipengaruhi oleh perubahan yang terjadi pada teks pembanding yang mengalami perubahan isi sebesar 60% menyebabkan *fingerpints* juga berkurang hingga sekitar 60% yaitu menjadi 46 *fingerprints* saja yang sama dengan *fingerprints* teks asli. Perbedaan tingkat kesamaan *fingerprints* menyebabkan tingkat similaritas menjadi 37,40% saja.
d. Pada percobaan keempat, pada teks asli dilakukan perubahan makna sebesar 80% sehingga tingkat similaritas menjadi 19,29%. Hal ini dipengaruhi oleh perubahan yang terjadi pada teks pembanding yang mengalami perubahan isi sebesar 80% menyebabkan *fingerpints* juga berkurang hingga sekitar 60% menjadi yaitu 27 *fingerprints* saja yang sama dengan *fingerprints* teks asli. Perbedaan tingkat tingkat kesamaan *fingerprints* menyebabkan tingkat similaritas menjadi 19,29% saja.

Hasil percobaan tersebut menunjukkan bahwa perubahan pada isi kalimat teks pembanding memberikan pengaruh yang cukup signifikan terhadap hasil perhitungan tingkat similaritas. Hal ini dipengaruhi oleh penerapan *word-level trigrams* pada teks. *Grams* yang terbentuk menggunakan potongan-potongan karakter akan berbeda dengan *grams* yang

JURIKOM (Jurnal Riset Komputer), Vol. 9 No. 2, April 2022
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v9i2.4060
Hal 427−435
http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

terbentuk akibat potongan kata. *Fingerprints* dipengaruhi oleh *grams* sehingga nilai-nilai *hash* yang diperoleh menggunakan rumus *Rolling Hash* tidak akan sama jika terjadi perbedaan antara *grams* dari teks asli dengan *grams* dari teks pembanding. Sebagai contoh, *fingerprints* dari *grams "searching for the same words in two or more documents"* berbeda dengan *fingerprints* dari *grams "to look for the similar words in two or more documents"* karena *grams* yang terbentuk pada teks asli adalah:

["searching for the", "for the same", "the same words",
"same words in". "words in two", "in two or",
"two or more", "or more documents"]

Grams yang terbentuk pada teks pembanding adalah:

["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"]

Ketika dilakukan pencocokan nilai-nilai *fingerprints* menggunakan rumus *Jaccard Coeefficient*, kedua teks yang telah dibandingkan tersebut tidak similar menurut sistem. Demikian seterusnya yang terjadi pada kalimat lainnya yang mengalami perubahan, dimana *grams* yang terbentuk berbeda dengan teks asli sehingga *fingerprints* yang dihasilkan juga mengalami perbedaan seiring dengan besarnya parameter perubahan yang ditetapkan.

4. KESIMPULAN

Berdasarkan hasil penelitian mengenai pencarian kata-kata yang sama pada dua teks dengan menerapkan algoritma *Winnowing* dan *word-level trigrams* maka metode tersebut dapat mendeteksi tingkat kesamaan teks sebesar 76,84%, 52,29%, 37,40%, dan 19,29%. Dengan mengubah isi kalimat pada teks pembanding sebesar 20% diperoleh tingkat similiritas sebesar 76,84%, demikian halnya ketika kalimat pada teks pembanding diubah sebesar 40% dihasilkan tingkat similaritas sebesar 52,29%, pada pengubahan isi kalimat sebesar 60% dihasilkan tingkat similaritas sebesar 37,40%, serta pada pengubahan isi kalimat sebesar 80% dihasilkan tingkat similaritas sebesar 19,29%. Perbedaan hasil perhitungan tingkat similaritas pada masing-masing teks yang dibandingkan dapat terjadi akibat penerapan *word-level trigrams* pada teks. Perubahan yang terjadi pada isi kalimat pada teks-teks pembanding menggunakan *trigrams* berbasis kata tersebut menyebabkan perubahan pada *grams*, *fingerprints*, dan *windows* yang terbentuk sehingga nilai-nilai *hash* yang sama antara teks yang dibandingkan juga menjadi berkurang seiring besarnya perubahan yang diterapkan. Dari hasil penelitian dapat disimpulkan bahwa metode pencocokan pola dengan algoritma *Winnowing* dan *word-level trigrams* dapat digunakan untuk mengukur tingkat kesamaan kata. Penelitian ini merupakan penelitian dasar yang dapat dikembangkan lagi untuk mendeteksi kesamaan kata pada dua dokumen karya ilmiah menggunakan algoritma pencocokan pola lainnya seperti algoritma *Boyer-Moore Horspool*, dan sebagainya, serta penelitian dapat diperluas dengan melakukan penandaan lokasi similaritas pada teks menggunakan algoritma *hybrid*.

REFERENSI

[1] 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.
[2] O. Hourrane and E. H. Benlahmar, "Survey of Plagiarism Detection Approaches and Big data Techniques related to Plagiarism Candidate Retrieval," in Proceedings of the 2nd international Conference on Big Data, Cloud and Applications (BDCA'17), Mar. 2017, pp. 1–6.
[3] N. Awale, M. Pandey, A. Dulal, and Timsina, "Plagiarism Detection in Programming Assignments using Machine Learning," Journal of Artificial Intelligence and Capsule Networks, vol. 2, no. 3, pp. 177–184, Jul. 2020.
[4] 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.
[5] A. Yudhana, Sunardi, and I. A. Mukaromah, "Implementation of Winnowing Algorithm with Dictionary English-Indonesia Technique to Detect Plagiarism," IJACSA) International Journal of Advanced Computer Science and Applications, vol. 9, no. 5, pp. 183–189, 2018, [Online]. Available: www.ijacsa.thesai.org
[6] 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.
[7] S. Sunardi, A. Yudhana, and I. A. Mukaromah, "Indonesia Words Detection Using Fingerprint Winnowing Algorithm," Jurnal Informatika, vol. 13, no. 1, p. 7, Jan. 2019, doi: 10.26555/jifo.v13i1.a8452.
[8] 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.
[9] 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.
[10] 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.
[11] Sugiono, Herwin, Hamdani, and Erlin, "Aplikasi Pendeteksi Tingkat Kesamaan Dokumen Teks- Rabin Ksrp vs Winnowing," Jurnal Teknologi Informasi & Komunikasi Digital Zone, vol. 9, no. 1, pp. 82–93, 2018.
[12] L. Sibarani, Magdalena, and A. Dharma, "Analisa Perbandingan Sistem Pendeteksian Kemiripan Judul Skripsi Menggunakan Algoritma Winnowing Dan Algoritma Rabin Karp," Riset dan E-Jurnal Manajemen Informatika Komputer, vol. 4, pp. 69–82, Oct. 2019.
[13] N. Alamsyah, "Perbandingan Algoritma Winnowing dengan Algoritma Rabin Karp untuk Mendeteksi Plagiarisme pada Kemiripan Teks Judul Skripsi," Technologia, vol. 8, no. 3, pp. 124–134, 2017.

[14] 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.
[15] M. Schonlau and N. Guenther, "Text mining using n-grams variables," The Stata Journal., vol. 17, no. 4, pp. 866–881, Dec. 2017.
[16] 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.
[17] 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
[18] 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.
[19] 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.
[20] 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).
[21] Stable Sort. Rolling Hash Function Tutorial Used by Rabin-Karp String Searching Algorithm. (Dec. 20, 2019). Accessed: Jan. 8, 2021. [Online Video]. Available: https://youtu.be/BfUejqd07yo

HASIL CEK_Rezki Ramdhani, Abdul Fadlil, Sunardi

ORIGINALITY REPORT

13%	13%	0%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.stmik-budidarma.ac.id Internet Source	8%
2	ejurnal.stmik-budidarma.ac.id Internet Source	5%

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		