

IMPLEMENTASI N-GRAM DAN K-GRAM PADA ALGORITMA *WINNOWER* UNTUK MENDETEKSI KEMIRIPAN LAPORAN AKHIR PRAKTIKUM

Qonita Abdah¹, Sigit Widiyanto²

^{1,2}Program Studi Magister Sistem Informasi, Universitas Gunadarma

Jl. Margonda Raya No. 100, Kec. Pondok Cina, Kel. Beji, Kota Depok, Jawa Barat

e-mail: gunitaro@gmail.com, sigitwidiyanto@staff.gunadarma.ac.id

Abstrak

Penelitian ini bertujuan untuk menganalisis efektivitas metode *n*-gram berbasis karakter dan *k*-gram berbasis kata pada Algoritma *Winnower* dalam mendeteksi tingkat kemiripan laporan akhir praktikum di Laboratorium Informatika. Proses penelitian meliputi tahap pra-pemrosesan teks, pembentukan *n*-gram dan *k*-gram, perhitungan nilai hash menggunakan rolling hash, pembentukan jendela (*window*), pemilihan *fingerprint*, serta perhitungan similaritas menggunakan Jaccard Coefficient. Pengujian dilakukan dengan membandingkan dokumen asli dan dokumen yang telah dimodifikasi sebanyak 20%, 40%, 60%, dan 80% dari isi aslinya. Hasil penelitian menunjukkan bahwa metode *n*-gram dengan parameter $n = 5$ dan *window size* = 4 menghasilkan nilai similarity masing-masing sebesar 1,0000; 0,7684; 0,5229; 0,3740; dan 0,1929. Sementara itu, metode *k*-gram dengan parameter $k = 3$ dan *window size* = 3 menghasilkan nilai similarity sebesar 1,0000; 0,7391; 0,5000; 0,3158; dan 0,1579.

Kata kunci: Algoritma *Winnower*, Plagiarisme, Deteksi Kemiripan Dokumen, *N-Gram*, *K-Gram*

Abstract

This study analyzes the effectiveness of the *n*-gram character-based method and the *k*-gram word-based method on the *Winnower* algorithm for detecting similarities in final practicum reports submitted by Informatics students. The research process involved text pre-processing, *n*-gram and *k*-gram formation, hash calculation using the rolling hash function, window generation, fingerprint selection, and similarity measurement using the Jaccard Coefficient. Experiments were conducted by comparing an original report with modified dummy documents containing 20%, 40%, 60%, and 80% changes in content. The experimental results show that the *n*-gram method with parameter $n = 5$ and *window size* = 4 produced similarity scores of 1.0000, 0.7684, 0.5229, 0.3740, and 0.1929. Meanwhile, the *k*-gram method with parameter $k = 3$ and *window size* = 3 produced scores of 1.0000, 0.7391, 0.5000, 0.3158, and 0.1579.

Keywords: *Winnower* Algorithm, Plagiarism, *N-Gram*, *K-Gram*

1. PENDAHULUAN

Pendidikan tinggi tidak hanya berupa perkuliahan teori, tetapi juga mencakup kegiatan praktikum yang bertujuan untuk mengasah keterampilan mahasiswa dalam menerapkan konsep yang telah dipelajari. Berdasarkan Peraturan Menteri Pendidikan dan Kebudayaan Republik Indonesia tentang Standar Nasional Pendidikan Tinggi (SNPT) Tahun 2020 Pasal 19 Ayat 3, pembelajaran di perguruan tinggi tidak terbatas pada kuliah, tetapi juga mencakup kegiatan praktikum. Di perguruan tinggi, laboratorium menjadi tempat penting bagi mahasiswa untuk berlatih dan mengembangkan kemampuan praktis mereka. Salah satunya adalah Laboratorium Informatika yang mengelola kegiatan praktikum bagi mahasiswa Jurusan Informatika.

Dalam kegiatan praktikum, mahasiswa diwajibkan membuat laporan akhir sebagai bentuk dokumentasi dan evaluasi atas kemampuan serta pemahaman yang diperoleh. Laporan ini

berfungsi sebagai penilaian akademik sekaligus latihan dalam menuangkan gagasan secara sistematis. Namun, dalam praktiknya, laporan akhir praktikum sering menjadi sasaran tindakan plagiarisme. Plagiarisme didefinisikan sebagai tindakan menyalin atau mengadaptasi ide, analisis, atau hasil kerja orang lain tanpa memberikan atribusi yang layak [2]. Dengan kemajuan teknologi internet, mahasiswa dengan mudah dapat menemukan dan menyalin laporan yang sudah ada milik rekan mereka, baik sebagian maupun seluruhnya. Perilaku ini tidak hanya mengurangi nilai keaslian karya akademik, tetapi juga menghambat proses pembelajaran dan pengembangan keterampilan kritis yang menjadi tujuan utama dari praktikum [3].

Untuk mengatasi hal ini, laboratorium telah menerapkan regulasi pembagian nilai bagi mahasiswa yang terdeteksi melakukan plagiarisme. Namun, implementasi manual memiliki kelemahan karena proses penilaian laporan di satu kelas seringkali dibagi antara 2 hingga 3 asisten yang berbeda. Situasi ini menciptakan celah pengawasan; misalnya, mahasiswa dengan nomor absen 1 menyalin laporan mahasiswa nomor absen 30 tidak akan teridentifikasi karena laporan mereka diperiksa oleh asisten yang berbeda.

Salah satu langkah komprehensif untuk mendeteksi plagiarisme adalah menggunakan konsep kemiripan dokumen melalui penerapan *Text Mining*. *Text mining* diaplikasikan untuk mengeksplorasi dan mengukur kesamaan teks dengan membandingkan elemen-elemen kata yang serupa. Dalam pengembangan sistem deteksi kemiripan teks, algoritma Winnowing merupakan salah satu pilihan yang cukup populer. Beberapa penelitian terdahulu telah mengkaji penggunaan algoritma Winnowing. Penelitian oleh Ramdhani dkk. [7] mengeksplorasi penggunaan word-level trigrams dengan algoritma Winnowing yang terbukti dapat mendeteksi kesamaan teks yang bervariasi bergantung pada skenario modifikasi. Penelitian lain oleh Hidayat dkk. [8] membandingkan performa k-gram berbasis karakter dan n-gram berbasis kata untuk dokumen Bahasa Indonesia, di mana parameter pemotongan sangat mempengaruhi hasil similarity.

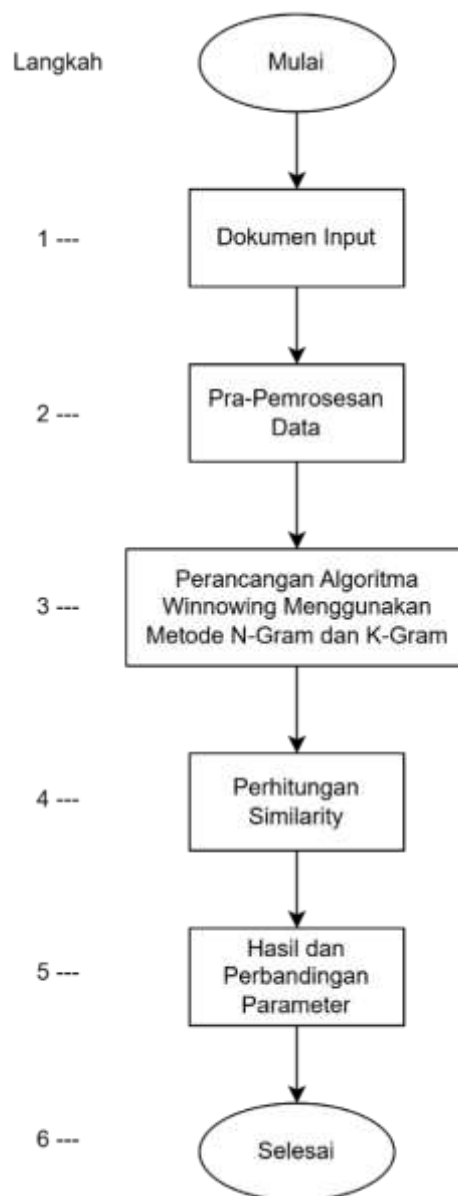
Dalam konteks ini, Algoritma Winnowing menjadi salah satu metode yang sangat efektif karena algoritma ini memenuhi syarat utama deteksi plagiarisme, yaitu whitespace insensitivity, noise suppression, dan position independence [5]. Karakter yang tidak relevan dihilangkan, menyisakan substansi teks untuk diekstraksi menjadi *fingerprint* [6]. Algoritma ini bekerja dengan cara mengubah teks menjadi representasi numerik yang disebut *fingerprint*, kemudian membandingkan *fingerprint* antar dokumen untuk menghitung tingkat kemiripannya [7]. Proses pembentukan fingerprint dalam Algoritma Winnowing dapat dilakukan melalui dua pendekatan, yaitu *n-gram* yang berbasis karakter dan *k-gram* yang berbasis kata. Kedua pendekatan ini memiliki karakteristik berbeda dalam mendeteksi kemiripan teks. Berdasarkan hal tersebut, penelitian ini dilakukan untuk mengimplementasikan dan membandingkan kedua metode tersebut.

2. METODE PENELITIAN

Penelitian ini melakukan lima tahapan dalam proses pembuatan Algoritma Winnowing, dimulai dari persiapan dataset sampai evaluasi hasil similaritas [8]. Berikut tahapan penelitian secara umum pada Gambar 1.

2.1 Dokumen Input

Data yang digunakan berupa 45 dokumen laporan akhir praktikum mahasiswa Program Studi Informatika pada mata kuliah Praktikum Kecerdasan Artifisial format PDF. Selain data asli, dirancang 5 dokumen uji tambahan (data dummy) secara manual yang memiliki modifikasi isi (pengubahan kalimat) sebesar 0% (identik), 20%, 40%, 60%, dan 80% terhadap salah satu dokumen asli. Data dummy ini digunakan untuk menguji sensitivitas dan akurasi algoritma terhadap berbagai tingkat manipulasi teks.



Gambar 1. Tahapan Penelitian

2.2 Pra- Pemrosesan Data

Tahapan awal adalah ekstraksi dokumen PDF menjadi plain text dengan menghapus halaman cover untuk membuang metadata seperti nama, NPM, dan asisten. Tahapan pra-pemrosesan difokuskan pada dua proses utama yaitu *case folding* dan *filtering* [9]. Penelitian ini tidak menggunakan tahap tokenisasi maupun stemming karena algoritma Winnowing yang digunakan bersifat *whitespace-insensitive* dan bekerja pada level substring, bukan pada level kata atau akar kata sehingga teks dibiarkan sesuai makna aslinya. *Case folding* yaitu proses normalisasi teks dengan mengubah seluruh huruf kapital menjadi huruf kecil (*lowercase*). Tahap berikutnya adalah *filtering*, yaitu yaitu proses pembersihan data dari elemen-elemen yang bersifat *noise* atau tidak relevan. *Filtering* dalam penelitian ini lebih difokuskan pada penghilangan simbol, tanda baca, dan karakter non-alfanumerik, bukan penghapusan kata-kata umum atau *stopwords*. Hal ini dilakukan karena algoritma Winnowing berbasis pada unit karakter atau kata (*gram*) dan sensitif

terhadap struktur internal teks. Menghapus kata justru dapat menyebabkan kehilangan informasi penting dalam proses deteksi kemiripan [10].

2.3 Perancangan Algoritma Winnowing

Winnowing merupakan algoritma yang digunakan dalam proses document fingerprinting. Prinsip kerja metode ini didasarkan pada penerapan fungsi hashing, yaitu suatu teknik yang mengubah rangkaian string menjadi bilangan. Langkah-langkah algoritma winnowing sebagai berikut [11]:

2.3.1 Pembentukan Grams

Implementasi algoritma Winnowing dalam penelitian ini diawali dengan tahap pra-pemrosesan dokumen melalui dua pendekatan pemotongan teks (*gramming*). Pendekatan pertama menggunakan metode N-Gram yang berfokus pada tingkat karakter (*character-level*). Pada metode ini, teks dipotong secara sekuensial dan berkesinambungan menjadi substring sepanjang n karakter. Proses pemotongan dilakukan melalui pergeseran (*sliding window* atau *shift*) sebanyak satu karakter pada setiap iterasinya, sehingga menghasilkan irisan karakter yang saling tumpang tindih (*overlapping*). Secara fungsional, penggunaan N-Gram ini bertujuan untuk memberikan sensitivitas deteksi yang tinggi. Metode ini sangat efektif dalam menangkap modifikasi minor yang dilakukan pada laporan akhir praktikum, seperti kesalahan ketik, penyisipan karakter, atau manipulasi kecil pada sintaks kode program.

Pendekatan kedua menggunakan metode K-Gram yang beroperasi pada tingkat kata (*word-level*). Berbeda dengan N-Gram, proses pemotongan pada K-Gram dilakukan dengan mengelompokkan teks menjadi kombinasi k kata yang berurutan. Dalam proses ini, spasi dipertahankan secara eksplisit dan difungsikan sebagai batas pemisah untuk mengidentifikasi pergantian antar kata, dengan pergeseran pemotongan dilakukan per satu kata. Fungsi utama K-Gram adalah menjaga konteks semantik kalimat sekaligus meminimalisir *false positive* yakni deteksi kemiripan keliru akibat kesamaan format baku atau *header* standar yang lazim ditemukan dalam dokumen laporan. Selain itu, K-Gram meningkatkan efisiensi waktu komputasi karena memproduksi unit teks yang lebih ringkas. Hasil pemotongan dari kedua metode ini bertindak sebagai unit dasar representasi teks. Setiap gram yang terbentuk kemudian dikonversi menjadi representasi numerik menggunakan fungsi hash. Kumpulan nilai hash diskrit inilah yang selanjutnya diproses ke dalam jendela-jendela (*windows*) algoritma Winnowing untuk mengekstrak *fingerprint* akhir dari masing-masing dokumen laporan.

2.3.2 Perhitungan Fungsi Hash

Dalam penelitian ini digunakan algoritma Polynomial Rolling Hash untuk proses hashing karena dinilai efisien dalam menghasilkan nilai hash yang unik dengan kemungkinan kolisi yang sangat rendah [12]. Hash collision terjadi ketika dua gram berbeda menghasilkan nilai hash yang sama. Algoritma ini mengubah setiap karakter dalam sebuah gram menjadi nilai ASCII-nya, lalu menggabungkannya secara matematis menggunakan bilangan dasar (*base*) 257 dan operasi modulo terhadap bilangan prima besar 10^9+7 . Penggunaan *base* dan modulo ini bertujuan agar nilai hash tersebar merata dan tetap dalam batas ukuran yang terkendali. Jumlah nilai hash yang dihasilkan pada setiap jendela tergantung pada lebar jendela yang ditentukan [13]. Berikut Persamaan (1) untuk menghitung *hash value*:

$$H(c_1 \dots c_k) = c_1 \times b^{(k-1)} + c_2 \times b^{(k-2)} + \dots + c_{k-1} \times b^{(k)} \bmod q \quad (1)$$

Keterangan:

$H(c_1 \dots c_k)$	= Nilai <i>hash</i> awal
$c_{(k)}$	= Nilai <i>ascii</i> per-karakter
b	= Nilai basis (konstanta) bilangan prima
k	= Panjang <i>gram/token</i>
Mod q	= Module bilangan prima

2.3.3 Pembukaan Window

Setelah dilakukan perhitungan fungsi hash terhadap setiap potongan n-gram dan k-gram, langkah selanjutnya adalah pembentukan *window* sebagai bagian dari mekanisme pemilihan *fingerprint* dalam algoritma Winnowing. Windowing adalah proses dari algoritma Winnowing untuk menyimpan hasil dari *rolling hash* yang telah dihasilkan sebelumnya. Banyaknya deretan hash values di setiap jendela tergantung dari lebar *window* (w) yang ditetapkan [14].

2.3.4 Pemilihan Fingerprint

Pemilihan *fingerprint* dilakukan dengan mencari nilai hash minimum dari setiap window [15]. Jika dalam window ada nilai minimum lebih dari satu, hash yang berada di paling kanan akan dipilih sebagai *fingerprint*. Nilai-nilai ini akan dipasang ke dalam *fingerprint*.

2.3.5 Pengukuran Similaritas

Tingkat kesamaan kata dapat dihitung menggunakan rumus Jaccard Coefficient. Algoritma ini secara umum digunakan dalam proses perbandingan dokumen dengan menghitung rasio antara jumlah elemen yang sama (irisan) dan jumlah total elemen unik dari kedua dokumen (gabungan). Nilai Jaccard Coefficient selalu berkisar antara 0 dan 1 atau, jika diekspresikan dalam persentase, antara 0% dan 100%. Berikut persamaan Jaccard Coefficient [16] :

$$J(X, Y) = \frac{X \cap Y}{X \cup Y} \times 100\% \quad (2)$$

Keterangan:

$|X \cap Y|$ adalah jumlah *fingerprint* yang sama atau identik yang muncul di kedua himpunan X dan Y.

$|X \cup Y|$ adalah jumlah total *fingerprint* unik yang terdapat di kedua himpunan X dan Y. Nilai ini dapat dihitung sebagai $|X| + |Y| - |X \cap Y|$.

3. HASIL DAN PEMBAHASAN

Pada tahap awal implementasi, proses *import* data dilakukan terhadap kumpulan dokumen laporan akhir praktikum mahasiswa yang telah disimpan dalam format PDF pada direktori Google Drive. Sebanyak 46 file laporan akhir diimpor sebagai data utama dan 5 file tambahan sebagai data uji (*dummy*) dengan tingkat pengubahan kalimat sebesar 20%, 40%, 60%, dan 80%.

3.1 Hasil Pra-Pemrosesan Data

Pra-pemrosesan yang dilakukan pada penelitian ini ada dua tahap utama, yaitu *case folding* dan *filtering*. Filtering dilakukan dengan menghapus kata kunci administratif dan metadata seperti pertanyaan dan "jawab", serta mengeliminasi symbol dan tanda baca. Selain itu, penomoran soal dan struktur kalimat pertanyaan juga dihilangkan agar sistem fokus hanya pada isi jawaban yang diberikan.



Gambar 2. Hasil Pra-Pemrosesan Data Metode N-Gram

Seperti pada Gambar 2 ditunjukkan teks hasil pra-pemrosesan berbasis karakter (*character-level*) yang digunakan untuk metode n-gram dengan seluruh spasi telah dihapus untuk membentuk aliran karakter yang kontinu. Sebaliknya, pada Gambar 3 ditampilkan teks berbasis kata (*word-level*) untuk metode k-gram dengan mempertahankan spasi sebagai pemisah kata untuk menjaga struktur dan makna semantik antar kata. Spasi ganda atau lebih yang mungkin muncul selama proses pembersihan dinormalisasi menjadi spasi tunggal untuk menjaga konsistensi.



Gambar 3. Hasil Pra-Pemrosesan Data Metode K-Gram

3.2 Hasil Pembentukan Grams

Ilustrasi hasil pembentukan n-gram & k-gram ditunjukkan dalam Gambar 4. Pada metode n-gram, teks digabung tanpa spasi. Sebagai contoh, frasa "kecerdasan buatan" diubah menjadi n-gram ($n=5$) yaitu "kecer", "ecerd", "cerda", "erdas", dan seterusnya. Pada metode k-gram, spasi dipertahankan. Frasa dipotong per 3 kata ($k=3$) yaitu "kecerdasan artifisial ai", "artifisial ai adalah", dan seterusnya.

Contoh Hasil Pembentukan N-Gram dan K-Gram dari File: Data Uji 5.txt

5-Gram (Karakter)	3-Gram (Kata)
kecer	kecerdasan artifisial ai
ecerd	artifisial ai adalah
cerda	ai adalah simulasi
erdas	adalah simulasi proses
rdasa	simulasi proses kecerdasan
dasan	proses kecerdasan manusia
asana	kecerdasan manusia oleh
sanar	manusia oleh mesin
anart	oleh mesin khususnya
narti	mesin khususnya sistem
...	...
erbag	gambaran awal tentang
rbaga	awal tentang bagaimana
bagai	tentang bagaimana ai
agaib	bagaimana ai dikembangkan
gaibi	ai dikembangkan dan
aibid	dikembangkan dan digunakan
ibida	dan digunakan dalam
bidan	digunakan dalam berbagai
idang	dalam berbagai bidang

Gambar 4. Pembentukan Grams

3.3 Hasil Perhitungan Nilai Hash

Untuk menjelaskan proses perhitungan fungsi *hash* secara konseptual, diberikan contoh pada metode *n-gram* "kecer" yang terdiri dari 5 karakter. Masing-masing karakter pada string ini diubah ke dalam bentuk nilai ASCII. Selanjutnya, dilakukan proses iteratif dengan basis bilangan $b = 257$ dan modulus $q = 10^9 + 7$, menggunakan rumus *rolling hash* Persamaan (1). Dengan

memasukkan nilai-nilai tersebut ke dalam rumus, diperoleh nilai *hash* untuk "kecer" sebesar 505328446.

$$H = (ASCII(k) \times 257^{(5-1)} + ASCII(e) \times 257^{(4-1)} + ASCII(c) \times 257^{(3-1)} + ASCII(e) \times 257^{(2-1)} + ASCII(r) \times 257^{(1-1)}) \bmod 10^{(9)} + 7$$

$$H = (107 \times 257^4 + 101 \times 257^3 + 99 \times 257^2 + 101 \times 257^1 + 114 \times 257^0) \bmod 10^9 + 7$$

$$H = (107 \times 257^4 + 101 \times 257^3 + 99 \times 257^2 + 101 \times 257^1 + 114 \times 257^0) \bmod 10^9 + 7$$

$$H = (107 \times 257^4 + 101 \times 257^3 + 99 \times 257^2 + 101 \times 257^1 + 114 \times 257^0) \bmod 10^9 + 7$$

$$H = (107 \times 257^4 + 101 \times 257^3 + 99 \times 257^2 + 101 \times 257^1 + 114 \times 257^0) \bmod 10^9 + 7$$

$$H = 468505331722 \bmod 10^9 + 7$$

$$H = 505328446$$

Hasil Hashing N-Gram (n=5) dari file: Data Uji 5.txt

N-Gram	ASCII per Huruf	Nilai Hash
kecer	[107, 101, 99, 101, 114]	505328446
ecerd	[101, 99, 101, 114, 100]	296692461
cerda	[99, 101, 114, 100, 97]	606555944
erdas	[101, 114, 100, 97, 115]	551240953
rdasa	[114, 100, 97, 115, 97]	25517926
.....		
ibida	[105, 98, 105, 100, 97]	729859948
bidan	[98, 105, 100, 97, 110]	311058499
idang	[105, 100, 97, 110, 103]	763283318

Gambar 5. Hasil Hasing N-Gram

Fungsi *Rolling Hash* diimplementasikan dalam Bahasa pemrograman Python dengan pendekatan iteratif, di mana setiap karakter pada string dikonversi ke nilai ASCII menggunakan fungsi `ord(char)`. Nilai tersebut dikombinasikan melalui operasi perkalian dengan base sebesar 257 dan penjumlahan kumulatif, kemudian distabilkan menggunakan operasi modulus dengan bilangan prima besar 10^9+7 untuk mencegah overflow serta menjaga distribusi nilai hash tetap merata. Hasil perhitungan ini menghasilkan nilai seperti pada Gambar 5.

Hasil Hashing K-Gram (k=3) dari file: Data Uji 5.txt

K-Gram	ASCII per Karakter	Nilai Hash
kecerdasan artifisial ai	[107, 101, 99, 101, 114, 100, 97, 115, 97, 110, 32, 97, 114, 116, 105, 102, 105, 115, 105, 97, 108, 32, 97, 105]	961000181
artifisial ai adalah	[97, 114, 116, 105, 102, 105, 115, 105, 97, 108, 32, 97, 105, 32, 97, 100, 97, 100, 97, 108]	67618397
ai adalah simulasi	[97, 105, 32, 97, 100, 97, 108, 97, 108, 32, 115, 105, 108, 117, 108, 97, 115, 105, 32, 112, 114, 111, 115, 101, 115]	836518417
adalah simulasi proses	[97, 100, 97, 108, 97, 108, 32, 115, 105, 108, 117, 108, 97, 115, 105, 32, 112, 114, 111, 115, 101, 115]	824959466
simulasi proses kecerdasan	[115, 105, 108, 117, 108, 97, 115, 105, 32, 112, 114, 111, 115, 101, 115, 32, 107, 101, 90, 101, 114, 100, 97, 115, 97, 110]	354081502
.....		
dan digunakan dalam	[100, 97, 110, 32, 100, 105, 103, 117, 110, 97, 107, 97, 110, 32, 100, 97, 108, 97, 100]	714137720
digunakan dalam berbagai	[100, 105, 103, 117, 110, 97, 107, 97, 110, 32, 100, 97, 108, 97, 109, 32, 90, 101, 114, 98, 97, 105, 97, 105]	989221621
dalam berbagai bidang	[100, 97, 108, 97, 108, 32, 98, 101, 114, 98, 97, 103, 97, 105, 32, 98, 105, 100, 97, 110, 103]	191580401

Gambar 6. Hasil Nilai Hash

3.4 Hasil Pembentukan Window dan Pencarian *Fingerprint*

Pada pendekatan *n-gram* berbasis karakter, penelitian ini menggunakan parameter $n = 5$ dengan *window size* $w = 4$. Berdasarkan Gambar 6 pada Window 1 yang terdiri dari potongan “kecer”, “ecerd”, “cerda”, dan “erdas”. Nilai *hash* terkecil yang terpilih adalah 296692461 dari potongan “ecerd” yang kemudian dipilih sebagai *fingerprint* pertama. Sementara itu, proses serupa diterapkan pada pendekatan *k-gram* berbasis kata dengan parameter $k = 3$ dan *window size* $w = 3$. Sebagai contoh, pada Window 1 terdiri dari potongan "kecerdasan artifisial ai", "artifisial ai adalah", dan seterusnya, nilai *hash* terkecil yang dipilih sebagai *fingerprint* adalah 67618397, yang berasal dari potongan “artifisial ai adalah”.

Window 1: kecer + 505328446 ecerd + 296692461 cerda + 606555944 erdas + 551240953 ↳ Min hash: 296692461 (dari "ecerd") Window 2: ecerd + 296692461 cerda + 606555944 erdas + 551240953 rdasa + 25517926 ↳ Min hash: 25517926 (dari "rdasa") Window 1800: gaibi + 987944110 aibid + 948458081 ibida + 729859948 bidan + 311058499 ↳ Min hash: 311058499 (dari "bidan") Window 1801: aibid + 948458081 ibida + 729859948 bidan + 311058499 idang + 763283318 ↳ Min hash: 311058499 (dari "bidan")	Pembentukan Window dan Pemilihan Fingerprint: Window 1: kecerdasan artifisial ai + 991988581 artifisial ai adalah + 67618397 ai adalah simulasi + 856518417 adalah simulasi proses + 824959466 ↳ Min hash: 67618397 (dari "artifisial ai adalah") Window 2: artifisial ai adalah + 67618397 ai adalah simulasi + 856518417 adalah simulasi proses + 824959466 simulasi proses kecerdasan + 874941582 ↳ Min hash: 67618397 (dari "artifisial ai adalah") Window 274: ai dikembangkan dan + 662663761 dikembangkan dan digunakan + 379113174 dan digunakan dalam + 714137329 digunakan dalam berbagai + 909221621 ↳ Min hash: 379113174 (dari "dikembangkan dan digunakan") Window 275: dikembangkan dan digunakan + 379113174 dan digunakan dalam + 714137329 digunakan dalam berbagai + 909221621 dalam berbagai bidang + 191580491 ↳ Min hash: 191580491 (dari "dalam berbagai bidang")
---	---

Gambar 7. Pembentukan *Window* dan Hasil *Fingerprint* Pada Metode N-Gram & K-Gram

3.5 Hasil Perhitungan Similaritas

Pengukuran kemiripan antar dokumen ini dilakukan menggunakan rumus *Jaccard Similarity Coefficient* yang terdapat pada Persamaan (2). Hasil dari perhitungan ini disajikan pada Gambar 7 menunjukkan penggunaan metode *n-gram* menghasilkan skor kemiripan lebih tinggi dibanding *k-gram*, terutama pada dokumen dengan modifikasi rendah hingga sedang. *N-gram* mencatat nilai tertinggi 0,7318 antara LA33.txt dan LA45.txt, dengan sebagian besar pasangan memiliki skor di atas 0,5, menunjukkan sensitivitas tinggi terhadap kesamaan karakter lokal, bahkan saat terjadi perubahan kata atau kalimat. Sebaliknya, *k-gram* menghasilkan skor maksimum hanya 0,5284 dan turun drastis pada banyak pasangan dokumen.

--- 20 HASIL KEMIRIPAN TERATAS (N-Gram) ---			--- 20 HASIL KEMIRIPAN DOKUMEN (K-Gram) ---		
Dokumen 1	Dokumen 2	Similarity Score	Dokumen 1	Dokumen 2	Similarity Score
LA33.txt	LA45.txt	0.7318	LA33.txt	LA4.txt	0.5284
LA4.txt	LA33.txt	0.6974	LA33.txt	LA45.txt	0.5260
LA4.txt	LA45.txt	0.6493	LA4.txt	LA45.txt	0.5244
LA7.txt	LA33.txt	0.6220	LA4.txt	LA7.txt	0.4435
LA7.txt	LA45.txt	0.6060	LA45.txt	LA7.txt	0.4378
LA4.txt	LA7.txt	0.6058	LA33.txt	LA7.txt	0.4337
LA33.txt	LA42.txt	0.5720	LA4.txt	LA42.txt	0.4050
LA4.txt	LA42.txt	0.5673	LA42.txt	LA45.txt	0.3916
LA7.txt	LA42.txt	0.5546	LA33.txt	LA42.txt	0.3888
LA42.txt	LA45.txt	0.5417	LA23.txt	LA4.txt	0.3688
LA4.txt	LA23.txt	0.5231	LA23.txt	LA45.txt	0.3610
LA23.txt	LA33.txt	0.5209	LA23.txt	LA33.txt	0.3518
LA23.txt	LA45.txt	0.5183	LA42.txt	LA7.txt	0.3420
LA7.txt	LA23.txt	0.5072	LA23.txt	LA7.txt	0.3183
LA23.txt	LA42.txt	0.4792	LA31.txt	LA4.txt	0.2988
LA31.txt	LA42.txt	0.4142	LA31.txt	LA45.txt	0.2981
LA7.txt	LA31.txt	0.4071	LA31.txt	LA33.txt	0.2859
LA31.txt	LA45.txt	0.4056	LA23.txt	LA42.txt	0.2857
LA4.txt	LA31.txt	0.4053	LA31.txt	LA7.txt	0.2717
LA31.txt	LA33.txt	0.3996	LA31.txt	LA42.txt	0.2450

Gambar 8. Hasil Perhitungan Similaritas 20 Data Tertatas Pada Metode N-Gram dan K-Gram

3.6 Evaluasi dan Perbandingan

Untuk menguji keandalan kedua metode terhadap upaya manipulasi (parafrasa atau penyuntingan sebagian), dilakukan uji silang dokumen asli (LA31) dengan 5 data dummy. Hasil pengujian

ditunjukkan pada Gambar 9 menunjukkan bahwa semakin besar tingkat modifikasi dokumen, semakin menurun jumlah *fingerprint* yang sama antara dokumen asli dan dokumen yang telah diubah, serta nilai skor kemiripan (*similarity score*). Pada percobaan awal (percobaan 0), dokumen tidak mengalami perubahan (diubah sebesar 0%), sehingga jumlah *fingerprint* yang sama mencapai 1972, dengan skor kemiripan sempurna sebesar 100,00%, yang menandakan kedua dokumen identik. Pada percobaan 1 (modifikasi 20%), terjadi penurunan jumlah *fingerprint* yang sama menjadi 1429, dengan skor kemiripan turun menjadi 68,77%. Hal ini menunjukkan bahwa meskipun hanya 20% teks yang diubah, cukup banyak bagian dari struktur teks yang berubah secara lokal, sehingga mengakibatkan hilangnya sejumlah *fingerprint* yang unik.

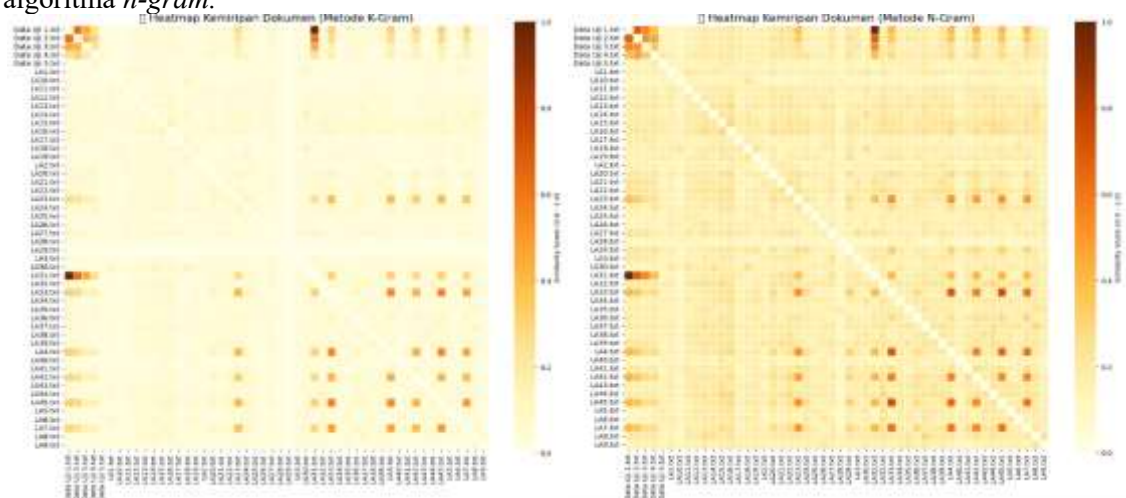
Percobaan	Parameter	Total Fingerprints Asli	Total Fingerprints Pembanding	Fingerprints Sama (Irisan)	Similaritas
Percobaan 0	Diubah sebesar 0% (Identik)	576	576	576	100,00%
Percobaan 1	Diubah sebesar 20%	576	434	388	62,38%
Percobaan 2	Diubah sebesar 40%	576	451	319	45,06%
Percobaan 3	Diubah sebesar 60%	576	277	170	24,89%
Percobaan 4	Diubah sebesar 80%	576	137	19	2,74%

Gambar 9. Perbandingan Skor Similaritas pada Data Uji *Dummy*

Dengan meningkatnya tingkat modifikasi hingga 40% (percobaan 2), jumlah *fingerprint* yang sama turun lebih drastis menjadi 1286, dan skor kemiripan menurun menjadi 55,43%. Ini mengindikasikan bahwa modifikasi pada tingkat sedang mulai mengganggu pola karakter yang ditangkap oleh algoritma n-gram, sehingga memengaruhi kemampuan deteksi kemiripan. Pada percobaan 3 (modifikasi 60%), jumlah *fingerprint* yang sama turun menjadi 845, dengan skor kemiripan 38,68%, menunjukkan bahwa sebagian besar isi dokumen telah berubah, namun masih terdapat bagian-bagian yang memiliki kesamaan struktur karakter. Sementara itu, pada percobaan 4 (modifikasi 80%), jumlah *fingerprint* yang sama sangat rendah (286) dan skor kemiripan turun menjadi 12,81%, yang menunjukkan bahwa hampir seluruh konten telah direvisi secara signifikan, sehingga hanya tersisa sedikit pola karakter yang serupa.

3.7 Distribusi Heatmap Kemiripan Dokumen

Pengukuran menggunakan Jaccard Similarity dilakukan untuk menyilang 45 dokumen laporan asli. Distribusi visualisasi matriks (*heatmap*) memperlihatkan bahwa metode n-gram memiliki cakupan deteksi (warna gelap) yang jauh lebih merata pada banyak pasangan dokumen. Sementara metode k-gram hanya memunculkan warna gelap pada dokumen-dokumen spesifik yang benar-benar memiliki struktur kalimat identik tanpa modifikasi berarti. Ini menegaskan bahwa mahasiswa sering menyalin dokumen namun menyisipkan pengubahan struktur kata, yang mana hal ini berhasil meloloskan dokumen dari deteksi *k-gram*, namun tetap terjebak oleh algoritma *n-gram*.



Gambar 9. Visualisasi Heatmap Metode N-gram dan K-gram

3.8 Optimalisasi Parameter

Optimalisasi parameter pada algoritma Winnowing bertujuan untuk menemukan kombinasi nilai *n-gram*, *k-gram*, serta ukuran *window* (*w*) yang dapat menghasilkan tingkat akurasi deteksi kemiripan dokumen yang optimal dengan efisiensi waktu komputasi yang tetap terjaga. Variasi parameter yang diuji mencakup nilai *n* untuk pemotongan berbasis karakter, nilai *k* untuk pemotongan berbasis kata, serta ukuran *window* yang digunakan untuk memilih *fingerprint* dari sekumpulan *hash*. Lebar jendela (*window size*) ditetapkan sebesar *w*=4 untuk *n-gram* dan *w*=3 untuk *k-gram* berdasarkan hasil uji pendahuluan. Perbedaan ukuran *window* ini disesuaikan dengan kerapatan data; pada *n-gram* yang memotong teks menjadi unit karakter yang lebih kecil, jendela yang sedikit lebih lebar (*w*=4) diperlukan untuk menyaring nilai *hash* agar *fingerprint* yang dihasilkan tidak terlalu padat dan tidak membebani komputasi. Sebaliknya, pada *k-gram* yang berbasis kata, potongan teks sudah memiliki tingkat keunikan leksikal yang lebih tinggi, sehingga jendela *w*=3 dinilai paling efisien dalam menangkap karakteristik dokumen secara representatif tanpa menghilangkan jejak kemiripan teks yang esensial.

Tabel 1. Hasil Uji Optimalisasi Paramater N-Gram

Nilai <i>n</i>	Window <i>w</i>	Rata-rata Similaritas	Waktu Komputasi (detik)
2	4	0.8921	8.14
3	4	0.8456	7.92
4	4	0.8013	7.35
5	4	0.9147	6.28
6	4	0.7429	7.11
7	4	0.6958	7.56
8	4	0.6532	8.23
9	4	0.6124	8.95
10	4	0.5776	9.34

Pada metode *n-gram* berbasis karakter (dengan *w*=4), hasil pengujian menunjukkan pada konfigurasi *n*=5 tidak hanya menghasilkan rata-rata nilai similaritas tertinggi sebesar 0,9147, tetapi juga mencatatkan waktu komputasi yang paling cepat yaitu 6,28 detik. Angka ini jauh lebih optimal dibandingkan penggunaan *n*=2 yang memakan waktu hingga 8,14 detik dengan akurasi 0,8921, atau *n*=10 yang akurasinya anjlok ke 0,5776 dengan waktu 9,34 detik.

Tabel 2. Hasil Uji Optimalisasi Paramater K-Gram

Nilai <i>k</i>	Window <i>w</i>	Rata-rata Similaritas	Waktu Komputasi (detik)
2	3	0.8017	7.32
3	3	0.8724	6.41

Nilai k	Window w	Rata-rata Similaritas	Waktu Komputasi (detik)
4	3	0.7415	7.01
5	3	0.6959	7.66
6	3	0.6528	8.17
7	3	0.6142	8.76
8	3	0.5831	9.11
9	3	0.5519	9.74
10	3	0.5236	10.25

Pada metode k-gram Kombinasi k=3 terbukti paling optimal dengan menyentuh rata-rata nilai similaritas tertinggi sebesar 0,8724 dan waktu komputasi yang efisien yaitu 6,41 detik. Penggunaan k=1 justru menghasilkan similaritas yang lebih rendah (0,7652) dan waktu yang lebih lama (7,85 detik), yang mengindikasikan bahwa pemotongan per satu kata terlalu membebani sistem dengan *noise* yang tidak bermakna. Sebaliknya, penggunaan k=10 membuat sistem mengabaikan banyak kesamaan frasa dengan similaritas yang merosot ke 0,5236 dan waktu komputasi membengkak menjadi 10,25 detik.

3.9 Pembahasan

Penelitian ini mengimplementasikan dua pendekatan gramming dalam kerangka Algoritma Winnowing, yaitu *n-gram* berbasis karakter dan *k-gram* berbasis kata. Perbedaan mendasar keduanya terletak pada unit dasar pembentukan *gram*. Perbedaan ini berimplikasi langsung pada hasil deteksi, di mana *n-gram* cenderung menghasilkan nilai similaritas yang lebih tinggi karena mampu menangkap kesamaan lokal antarkarakter meskipun terjadi perubahan pada level kata atau struktur kalimat, sementara *k-gram* lebih rentan terhadap perubahan leksikal seperti penggunaan sinonim atau parafrasa. Hasil pengujian dengan data *dummy* menunjukkan pola penurunan yang konsisten pada kedua metode seiring meningkatnya tingkat modifikasi dokumen. Pada dokumen identik, keduanya menghasilkan nilai sempurna 1,0000. Pada modifikasi 20% dan 40%, kedua metode masih mampu mendeteksi kemiripan dalam kategori berat hingga sedang. Namun pada modifikasi 60% dan 80%, selisih antara keduanya semakin lebar, dengan *n-gram* mempertahankan nilai 0,3740 dan 0,1929 dibandingkan *k-gram* yang lebih cepat turun ke 0,3158 dan 0,1579, mencerminkan bahwa *k-gram* lebih mudah dikelabui oleh modifikasi struktural kalimat. Dari perspektif optimalisasi parameter, konfigurasi n=5 dengan *window size* w=4 dan k=3 dengan w=3 terbukti paling optimal, menghasilkan rata-rata similaritas tertinggi sekaligus waktu komputasi paling efisien.

4. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa metode n-gram berbasis karakter dan k-gram berbasis kata dalam kerangka Algoritma Winnowing terbukti mampu mendeteksi kemiripan dokumen laporan akhir praktikum Laboratorium Informatika. Pada skenario dokumen identik, kedua metode mencapai nilai similaritas maksimal sebesar 1,0000, menunjukkan kemampuan yang kuat dalam mengenali salinan langsung atau praktik *copy-paste* secara utuh. Pada pengujian dengan dokumen yang dimodifikasi sebesar 20%, 40%, 60%, dan 80%, metode n-gram menunjukkan nilai similaritas yang secara bertahap menurun menjadi 0,7684; 0,5229; 0,3740; dan 0,1929. Sementara itu, metode k-gram menghasilkan nilai 0,7391; 0,5000; 0,3158; dan 0,1579. Pola penurunan yang

lebih lambat pada n-gram mengindikasikan sensitivitasnya yang lebih tinggi terhadap kesamaan lokal pada level karakter, sehingga lebih responsif terhadap plagiarisme parsial meskipun hanya sebagian kecil teks yang identik. Sebaliknya, k-gram lebih bergantung pada kesamaan unit kata dan struktur kalimat, sehingga lebih rentan terhadap modifikasi seperti parafrase atau penggantian kosakata, yang menyebabkan penurunan nilai similaritas lebih cepat. Oleh karena itu, metode n-gram lebih sesuai digunakan sebagai sistem deteksi awal kemiripan dokumen, terutama dalam konteks Laporan Akhir Praktikum di Laboratorium Informatika yang menekankan deteksi plagiasi berbasis teks secara langsung.

Daftar Pustaka

- [1] Y. Apridiansyah, A. Wijaya, dan A. Purjiawan, "Penerapan Fungsi Metode Rolling Hash Pada Algoritma Winnowing Untuk Mendeteksi Kemiripan Teks Abstrak Berbasis Web," *Jurnal Media Infotama*, vol. 18, no. 1, hal. 128-133, 2022.
- [2] I. Gunawan, "Deteksi Plagiarisme Abstrak Skripsi dengan Menggunakan Algoritma Rabin Karp," *Techno Xplore: Jurnal Ilmu Komputer dan Teknologi Informasi*, vol. 6, no. 2, hal. 75-81, 2021.
- [3] M. Hadid dan A. Triawan, "Penerapan Algoritma Winnowing dan Cosine Similarity untuk Penilaian Jawaban Esai," *Jurnal SAINTEKOM*, vol. 1, no. 1, hal. 66-72, 2025.
- [4] A. Pratomo dan A. Suryotomo, "Implementasi pengecekan plagiarisme proposal tugas akhir mahasiswa teknik informatika," *Seminar Nasional Informatika (SEMNASIF)*, vol. 1, no. 1, hal. 221-229, 2020.
- [5] W. A. Negoro, F. Amalia, dan E. Santoso, "Pengembangan Aplikasi Resep Masakan dengan Rekomendasi berdasarkan Bahan-Bahan Makanan Berbasis Web," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 3, hal. 9212-9221, 2019.
- [6] H. Pamungkas dan Fitrianiingsih, "Deteksi Similaritas Dokumen Ilmiah Menggunakan Algoritma Rabin-Karp," *Jurnal Ilmiah Informatika Komputer*, vol. 24, no. 3, hal. 209-219, 2019.
- [7] R. Ramdhani, A. Fadlil, dan S. Sunardi, "Penerapan Algoritma Winnowing dan Word-Level Trigrams Untuk Mengidentifikasi Kesamaan Kata," *JURIKOM (Jurnal Riset Komputer)*, vol. 9, no. 2, hal. 427, 2022.
- [8] W. Hidayat, E. Utami, dan A. D. Hartanto, "Pemilihan Parameter Terbaik pada Algoritma Winnowing dalam Mendeteksi Tingkat Kesamaan Dokumen Bahasa Indonesia," *Creative Information Technology Journal*, vol. 7, no. 2, hal. 119-132, 2021.
- [9] S. Purwaningrum, A. Susanto, dan N. W. A. Prasetya, "Comparison of Dice Similarity and Jaccard Coefficient Against Winnowing Algorithm For Similarity Detection of Indonesian Text Documents," *J. Appl.*, vol. 6, no. 1, hal. 10-22, 2021.
- [10] M. I. Rahayu dan A. P. Hidayatullah, "Implementasi Algoritma Winnowing untuk Deteksi Kemiripan Proposal Skripsi pada Sistem Informasi KP dan Skripsi (SIKPS)," *Jurnal Bangkit Indonesia*, vol. 14, no. 1, hal. 11-17, 2025.
- [11] M. Rijoly, W. Pramudita, B. Tomasouw, dan Z. Leleury, "Perancangan Sistem Deteksi Plagiarisme Skripsi (Judul Dan Abstrak) Berbasis Matlab Menggunakan Algoritma Winnowing," *Tensor: Pure and Applied Mathematics Journal*, vol. 2, no. 2, hal. 67-76, 2021.
- [12] R. Said, M. Muliadi, dan E. Sofyan, "Sistem Pendeteksi Kemiripan Judul Skripsi Menggunakan Algoritma Winnowing," *Journal Pegguruang: Conference Series*, vol. 3, no. 1, hal. 144-147, 2021.
- [13] A. K. Saputra, E. Erlangga, dan T. Tanjung, "Analisis Kinerja Algoritma Winnowing pada Pendeteksian Plagiarisme," *Explore: Jurnal Sistem Informasi dan Telematika*, vol. 13, no. 2, hal. 125-130, 2022.
- [14] S. Shrestha, S. Gautam, K. Sharma, dan A. Bhandari, "Winnowing Algorithm: A Powerful Tool for Identifying Plagiarism in Assignments," *Journal of Trends in Computer Science and Smart Technology*, vol. 5, no. 2, hal. 168-189, 2023.

- [15] L. Sugiarto, C. Mulyadi, dan S. Rihastuti, “Analisa Algoritma String Matching Dan Winnowing Untuk Deteksi Kemiripan Judul Tugas Akhir,” *Jurnal Teknologi Informasi*, vol. 6, no. 2, hal. 97-106, 2020.
- [16] A. Suroni dan E. S. Setyati, “Klasifikasi Bank Soal Berdasarkan Kompetensi Menggunakan Algoritma Winnowing,” *Jurnal Ilmiah Teknik Informatika*, vol. 15, no. 2, hal. 163-173, 2021.
- [17] S. Syaiful, “Implementasi Algoritma Winnowing Pada Aplikasi Ensiklopedia Alam,” *Bulletin of Information System Research*, vol. 1, no. 1, hal. 23-29, 2022.
- [18] I. Widaningrum, D. Mustikasari, R. Arifin, dan E. D. Cahyani, “Analisa Penggunaan K-Gram pada Karakter, Kata dan Kalimat untuk Mendeteksi Kesamaan Dokumen,” *Jurnal Ilmiah Komputer*, vol. 8, hal. 10-18, 2020.



ZONasi: Jurnal Sistem Informasi

Is licensed under a [Creative Commons Attribution International \(CC BY-SA 4.0\)](https://creativecommons.org/licenses/by-sa/4.0/)