Aplikasi Pendeteksi Tingkat Kesamaan Dokumen Teks: Algoritma Rabin Karp Vs. Winnowing

Sugiono[1], Herwin[2], Hamdani[3], Erlin[4*]
[1,4] Program Studi Teknik Informatika, STMIK Amik Riau
[2] Program Studi Manajemen Informatika, STMIK Amik Riau
[3] Program Studi Teknologi Informasi, STMIK Amik Riau
Jl. Purwodadi Indah, Km.10, Pekanbaru
e-mail: [1]sugiono@stmik-amik-riau.ac.id, [2]herwin@stmik-amik-riau.ac.id,
[3]hamdani@stmik-amik-riau.ac.id, [4]erlin@stmik-amik-riau.ac.id

Abstrak

Tindakan copy paste dokumen teks sering terjadi dalam penulisan karya ilmiah tanpa memberikan kredit kepada yang mempunyai dokumen teks tersebut. Tindakan melanggar kode etik ini disebabkan karena tersedianya fasilitas menyalin dan menempel teks pada aplikasi pengolah kata. Tujuan dari penelitian ini adalah untuk membangun sebuah aplikasi yang mampu mendeteksi tingkat kesamaan dokumen teks dengan terlebih dahulu membandingkan tingkat kehandalan dari dua algoritma pendeteksi kesamaan teks yaitu algoritma rabin-karp dan algoritma winnowing. Perbandingan dilakukan terhadap dua variabel yaitu tingkat kemampuan mendeteksi dan waktu pemrosesan. Hasil menunjukkan bawah algoritma winnowing lebih unggul dibandingkan algoritma rabin-karp dari sisi tingkat akurasi maupun dari sisi waktu pemrosesan.

Kata kunci: *Rabin-Karp, Winnowing, Deteksi Plagiat, Kesamaan Teks*

Abstract

The behavior of copy pastes the text document often occurs in scientific writing without giving credit to those who have the text document. The behavior of this missing code of conduct due to the availability of facility to copy and paste the text in a word processing application. The purpose of this study is to build an application that can detect the index of similarity of text documents by first comparing the level of reliability of the two text similarity algorithms, i.e., Rabin-Karp and Winnowing. The comparison is measured based on two variables; the level of capability of detecting and processing time. The result shows that Winnowing algorithm outperforms Rabin-Karp in term of both accuracy and processing time.

Keywords: *Rabin-Karp, Winnowing, Plagiarism Detection, Text Similarity*

1. Pendahuluan

Perkembangan teknologi informasi yang semakin canggih mengakibatkan kemudahan disegala bidang, termasuk dalam pencarian informasi. Hal ini juga berdampak kepada penyimpanan data dan prosesnya yang mengalami perkembangan secara signifikan sehingga memudahkan orang untuk menyimpan, mencari, dan mengolah data dengan cepat. Kemudahan yang ditawarkan oleh media seperti *internet* untuk mendapatkan informasi merupakan salah satu dampak positif dari kemajuan teknologi. Namun perkembangan tersebut tidak terlepas dari dampak negatif yang hampir tidak dapat dihindari, salah satunya adalah *plagiarisme*. *Plagiarisme* adalah tindakan penyalahgunaan, dalam hal mencuri serta mempublikasikan

pemikiran, bahasa, gagasan, ide atau ekspresi penulis lain dan merepresentasikan semua itu sebagai milik sendiri [1].

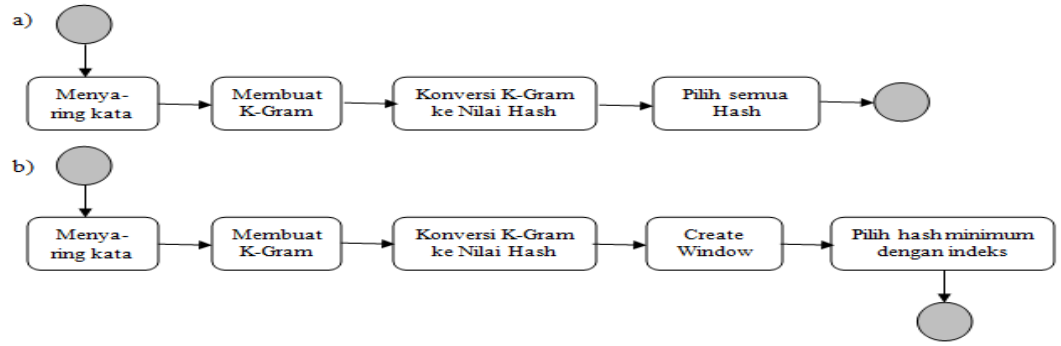
Praktek *plagiat* mudah terjadi diberbagai kalangan yang selalu berinteraksi dengan komputer mengingat adanya fasilitas untuk menyalin dan menempel teks (*copy paste*). Praktek ini lebih meningkat lagi dengan adanya dukungan fasilitas koneksi yang memungkinkan untuk mengakses hasil karya orang lain secara bebas melalui internet. *Plagiarisme* dapat mematikan kreatifitas seseorang karena tindakan ini tidak membutuhkan tenaga dan tidak harus berfikir keras. Oleh karena itu, tindakan preventif terhadap praktek *plagiarisme* harus dilakukan.

Membangun sebuah aplikasi yang dapat mendeteksi kesamaan dokumen teks menjadi penting untuk dilakukan. Saat ini tersedia beberapa aplikasi dan tools gratis maupun berbayar yang mampu mendeteksi kesamaan teks seperti Turnitin [2], Plagiarism Checker X [3], Grammarly [4], Dupli Checker [5] dan lainnya. Selanjutnya, beberapa penelitipun telah melakukan penelitian dengan menerapkan beberapa algoritma untuk mendeteksi tingkat kesamaan dokumen teks, seperti algoritma Brute-Force [6,7], Edit Distance [8,9], Boyer Moore [10,11] yang diimplementasikan untuk perbandingan teks lengkap. Selain itu, peneliti lain fokus pada tipe dokumen fingerprint yang digunakan untuk mendeteksi keakuratan salinan dokumen baik semua teks maupun sebagian. Algoritma yang termasuk dalam kelompok ini adalah algoritma Rabin-Karp [12,13,14] dan algoritma Winnowing [15,16,17].

Penelitian ini akan membangun sebuah aplikasi yang mampu mendeteksi tingkat kesamaan dokumen teks dalam bahasa Indonesia dengan ekstensi doc, html dan pdf. Aplikasi dibangun dengan membandingkan dua algoritma yaitu algoritma Rabin-Karp dan Algoritma Winnowing yang termasuk kedalam kelompok dokumen fingerprint. Kedua algoritma akan dibandingkan berdasarkan kinerja evaluasi terhadap dua variabel yaitu tingkat akurasi dan waktu yang diperlukan dalam proses mendeteksi kesamaan dokumen teks. Aplikasi dapat menentukan indeks plagiat berdasarkan tingkat kesamaan penggunaan kata yang muncul pada dua dokumen berurutan.

2. Metode Penelitian

Tahapan pembuatan aplikasi pendeteksi kesamaan teks berdasarkan perbandingan kedua algoritma ini dimulai dari tahap menyediaan dokumen asli dan dokumen uji. Dokumen uji selanjutnya akan diproses berdasarkan tahapan *preprocessing* yaitu melakukan analisis semantik (kebenaran arti) dan analisis sintaktik (kebenaran susunan) terhadap teks. Tujuan dari pemrosesan awal ini untuk mempersiapkan teks menjadi data yang akan diproses lebih lanjut. Hasil dari *preprocessing* dilanjutkan dengan membuat K-Gram, selanjutnya konversi K-Gram ke nilai hash. Untuk algoritma *Rabin-Karp* dilanjutkan dengan memilih nilai hash, sementara untuk algoritma winnowing dilanjutkan dengan meng-*create* window dan memilih hash minimum dengan indeks. Gambar 1 memperlihatkan perbandingan alur kerja dari kedua algoritma.



Gambar 1. a) Algoritma Rabin-Karp; b) Algoritma Winnowing

2.1. Pengujian Menggunakan Algoritma *Rabin-Karp*

Beberapa tahapan yang dilakukan untuk pengujian kesamaan dokumen teks dengan algoritma *Rabin-Karp* diuraikan sebagai berikut:

(a) Langkah Pertama

Tabel 1 memperlihatkan persiapan dokumen yang akan diuji dengan algoritma *Rabin-Karp.*

Tabel 1. Dokumen Asli dan Dokumen Uji

Proses	Dokumen Asli	Dokumen Uji
Isi String	Salah satu Mahasiswa “STMIK AMIK RIAU" adalah sugiono	sugiono Mahasiswa "STMIK AMIK RIAU"

(b) Langkah Kedua

Case folding digunakan untuk mengubah seluruh huruf kapital menjadi huruf kecil, hanya huruf “a” sampai “z” seperti diperlihatkan pada tabel 2.

Tabel 2. *Case Folding*

Proses	Dokumen Asli	Dokumen Uji
Isi String	salah satu mahasiswa "STMIK AMIK RIAU" adalah sugiono	sugiono Mahasiswa "STMIK AMIK RIAU"
Case Folding	salah satu mahasiswa "stmik amik riau" adalah sugiono	sugiono mahasiswa "stmik amik riau"

(c) Langkah Ketiga

Membuang semua tanda baca yang ada pada dokumen termasuk *enter* dan *spasi*. Hasil langkah 3 dapat dilihat pada tabel 3.

Tabel 3. Membuang Tanda Baca

Proses	Dokumen Asli	Dokumen Uji
Isi String	Salah satu Mahasiswa "STMIK AMIK RIAU" adalah sugiono	sugiono Mahasiswa "STMIK AMIK RIAU"
Case Folding	salah satu mahasiswa "stmik amik riau" adalah sugiono	sugiono mahasiswa "stmik amik riau"
Buang Tanda Baca	Salahsatumahasiswastmikamik riauadalahsugiono	sugionomahasiswastmikamikriau

(d) Langkah Keempat

Memotong *string* sepanjang *k*, untuk hasil *K-Gram* pertama sebanyak *k*. Untuk *K-Gram* selanjutnya diawali dari karakter kedua dari *K-Gram* yang pertama, sebanyak *k*. Misalnya nilai *k*=5. Tabel 4 memperlihatkan potongan sebahagian K-Gram dari algoritma Rabin Karp.

Tabel 4. Cuplikan *K-Gram Rabin-Karp*

Proses	Dokumen Asli	Dokumen Uji
K-Gram	{salah}{alahs}	{sugio}{ugion}
	{lahsa}{ahsat}	{giono}{ionom}
	{hsatu}{satum}	{onoma}{nomah}
	{atuma}{tumah}	{omaha}{mahas}
	{umaha}{hasis}	{ahasi}{hasis}
	{asisw}{siswa}	{asisw}{siswa}
	{iswas}{swast}	{iswas}{swast}
	{waste}{astem}	{waste}{astem}
	{stemi}{temik}	{stemi}{temik}

(a) Langkah Kelima

Tabel 5 memperlihatkan *hashing* terhadap seluruh pecahan string hasil dari proses *parsing K-Gram*.

Tabel 5. Proses *Hashing*

Proses	Dokumen Asli		Dokumen Uji	
Hashing	{1827061}	{1576921}	{1853171}	{1864126}
	{1724281}	{1573699}	{1662530}	{1699686}
	{1688859}	{1828254}	{1786288}	{1772611}
	{1590026}	{1868443}	{1783208}	{1738742}
	{1871054}	{1738742}	{1571708}	{1666956}
	{1571708}	{1666956}	{1587331}	{1838791}
	{1587331}	{1838791}	{1705951}	{1855222}
	{1705951}	{1855222}	{1886678}	{1588498}
	{1886678}	{1588498}	{1851636}	{1847238}
	{1851636}	{1847238}	{1637799}	{1749747}
	{1637799}	{1749747}	{1692763}	{1710145}
	{1692763}	{1710145}	{1579252}	{1749930}
	{1579252}	{1749930}	{1694768}	{1732210}
	{1694768}	{1732210}		
	{1821950}	{1681736}		
	{1588838}	{1855379}		
	{1566299}	{1607446}		
	{1576921}	{1724301}		
	{1573906}	{1691124}		
	{1853171}	{1864126}		
	{1662530}			

(b) Langkah Keenam

Fingerprint adalah proses membandingkan *string* dari dokumen teks yang berbeda yang mempunyai *hash value* yang sama. Proses *fingerprint* diperlihatkan pada tabel 6.

Tabel 6. Proses *Fingerprint*

Finger print	{1738742}	{1571708}	{1666956}	{1587331}	{1838791}
	{1705951}	{1855222}	{1886678}	{1588498}	{1851636}
	{1847238}	{1637799}	{1749747}	{1692763}	{1710145}
	{1579252}	{1749930}	{1694768}	{1732210}	{1853171}
	{1864126}	{1662530}			

(c) Langkah Ketujuh

Similarity digunakan sebagai acuan tingkat kesamaan antara dokumen asli dengan dokumen uji. Tabel 7 memperlihatkan hasil similarity index sebesar 65.67% yang menandakan bahwa tingkat kemiripan dokumen asli dengan dokumen uji sebesar 65.67% dengan waktu proses selama 0,09 detik.

Tabel 7. *Similarity Rabin-Karp*

Similarity Index	65.67%
Waktu	0.09 Detik

2.2. Pengujian Menggunakan Algoritma *Winnowing*

Tahapan *preprocessing* sebagaimana yang telah dilakukan pada algoritma *Rabin-Karp* juga dilakukan pada algoritma *Winnowing*. Lanjutan proses memotong string sepanjang *k* dalam proses perhitungan *K-Gram* dan lanjutan proses *Hashing* terhadap seluruh pecahan string juga sama antara kedua algoritma. Perbedaannya terletak langkah keenam dimana algoritma *Winnowing* tidak menggunakan semua nilai *hash* dari setiap rangkaian gram yang dibentuk. Nilai *hash* yang dibentuk pada tahap sebelumnya akan dibagi ke dalam *window* berukuran *w*. *Window* pertama berisi nilai *hash* pertama sampai nilai *hash* ke-*w*. *Window* kedua dibentuk dari nilai *hash* kedua sampai nilai *hash* ke-*w*+1 dan seterusnya sampai terbentuk *window* dari seluruh nilai *hash*. Pembentukan window dari hasil perhitungan nilai hash pada tahap sebelumnya dengan ukuran lebar window (w) = 6 diperlihatkan pada tabel 8.

Tabel 8. Cuplikan *Window Winnowing*

Dokumen Asli	Dokumen Uji
{1827061} {1576921} {1724281} {1573699} {1688859} {1828254}	{1853171} {1864126} {1662530} {1699686} {1786288} {1772611}
{1576921} {1724281} {1573699} {1688859} {1828254} {1590026}	{1864126} {1662530} {1699686} {1786288} {1772611} {1783208}
{1724281} {1573699} {1688859} {1828254} {1590026} {1868443}	{1662530} {1699686} {1786288} {1772611} {1783208} {1738742}
{1573699} {1688859} {1828254} {1590026} {1868443} {1871054}	{1699686} {1786288} {1772611} {1783208} {1738742} {1571708}
{1688859} {1828254} {1590026} {1868443} {1871054} {1738742}	{1786288} {1772611} {1783208} {1738742} {1571708} {1666956}
{1828254} {1590026} {1868443} {1871054} {1738742} {1571708}	{1772611} {1783208} {1738742} {1571708} {1666956} {1587331}
{1590026} {1868443} {1871054} {1738742} {1571708} {1666956}	{1783208} {1738742} {1571708} {1666956} {1587331} {1838791}
{1868443} {1871054} {1738742} {1571708} {1666956} {1587331}	{1738742} {1571708} {1666956} {1587331} {1838791} {1705951}

Setelah terbentuk *window* dari seluruh nilai *hash*, tahap selanjutnya adalah menentukan nilai *fingerprint* teks dengan memilih nilai *hash* terkecil dari setiap *window*. Pemilihan nilai *fingerprint* dari hasil pembentukan *window* pada tahap sebelumnya diperlihatkan pada tabel 9.

Tabel 9. Cuplikan *Fingerprint Winnowing*

Dokumen Asli	Dokumen Uji
{1827061} {1576921} {1724281}	{1853171} {1864126} {1662530}
{1573699} {1688859} {1828254}	{1699686} {1786288} {1772611}
{1576921} {1724281} {1573699}	{1864126} {1662530} {1699686}
{1688859} {1828254} {1590026}	{1786288} {1772611} {1783208}
{1724281} {1573699} {1688859}	{1662530} {1699686} {1786288}
{1828254} {1590026} {1868443}	{1772611} {1783208} {1738742}
{1573699} {1688859} {1828254}	{1699686} {1786288} {1772611}
{1590026} {1868443} {1871054}	{1783208} {1738742} {1571708}
{1688859} {1828254} {1590026}	{1786288} {1772611} {1783208}
{1868443} {1871054} {1738742}	{1738742} {1571708} {1666956}
{1828254} {1590026} {1868443}	{1772611} {1783208} {1738742}
{1871054} {1738742} {1571708}	{1571708} {1666956} {1587331}
{1590026} {1868443} {1871054}	{1783208} {1738742} {1571708}
{1738742} {1571708} {1666956}	{1666956} {1587331} {1838791}
{1868443} {1871054} {1738742}	{1738742} {1571708} {1666956}
{1571708} {1666956} {1587331}	{1587331} {1838791} {1705951}

Nilai *fingerprint* yang dibentuk dari algoritma *Winnowing* digunakan untuk mengukur presentase kemiripan. Tabel 10 memperlihatkan similarity winnowing yang dihasilkan.

Tabel 10. *Similarity Winnowing*

Dokumen Asli	Dokumen Uji
{1573699} {1590026} {1571708}	{1853171} {1662530}
{1587331} {1588498} {1692763}	{1571708} {1587331}
{1579252} {1588838} {1566299}	{1588498} {1637799}
{1573906}	{1579252}
= 10	= 7

$$= \frac{Jumlah\_fingerprint\_yang\_sama}{Jumlah\_seluruh\_fingerprint} 100\%$$

$$= \frac{7}{10} 100\% = 70\%$$

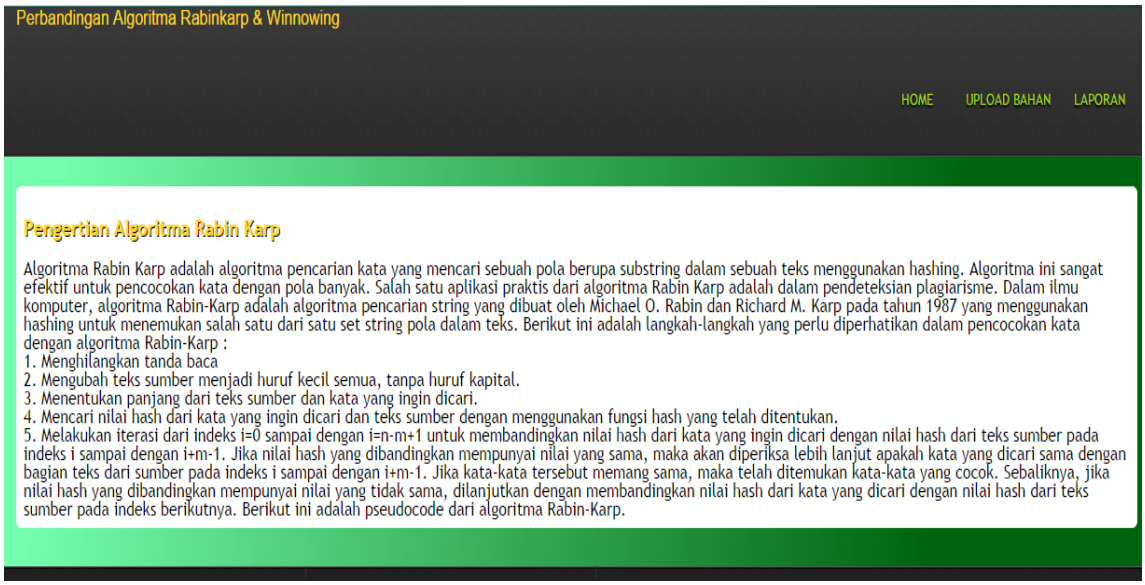
Berdasarkan hasil kesamaan kedua *fingerprint*, maka *presentase* kemiripan teks antara teks 1 dan teks 2 yang terbentuk yaitu sebesar 70%.

3. Hasil dan Pembahasan

Tampilan *output* program aplikasi terdiri dari tampilan halaman utama program aplikasi, halaman data laporan dan halaman *preview*.

3.1. Tampilan Halaman Utama

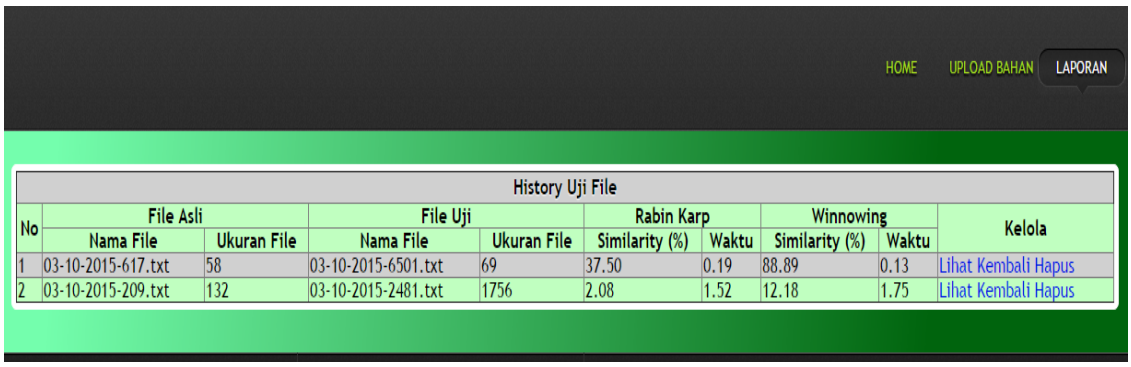
Tampilan halaman utama program aplikasi adalah sebuah informasi tentang pengertian masing-masing algoritma dan cara pemakaian aplikasi. Gambar 2 adalah tampilan dari halaman utama program aplikasi yang telah dibangun yang terdiri atas menu home, *upload* bahan dan laporan.



Gambar2: Tampilan halaman utama (interface)

3.2. Tampilan Halaman Data Laporan

Pada halaman *history* ditampilkan semua file hasil pengujian, memperlihatkan nama *file* asli dan *file* uji, persentase kemiripan teksnya (*similarity text*), serta hasil waktu yang diperlukan untuk memproses dari masing-masing metode. Kemudian pada halaman ini terdapat kontrol untuk melihat detail hasil pengujian dan kontrol untuk menghapus *file* hasil pengujian. Gambar 3 memperlihatkan tampilan halaman data laporan



No	File Asli		File Uji		Rabin Karp		Winnowing		Kelola
	Nama File	Ukuran File	Nama File	Ukuran File	Similarity (%)	Waktu	Similarity (%)	Waktu	
1	03-10-2015-617.txt	58	03-10-2015-6501.txt	69	37.50	0.19	88.89	0.13	Lihat Kembali Hapus
2	03-10-2015-209.txt	132	03-10-2015-2481.txt	1756	2.08	1.52	12.18	1.75	Lihat Kembali Hapus

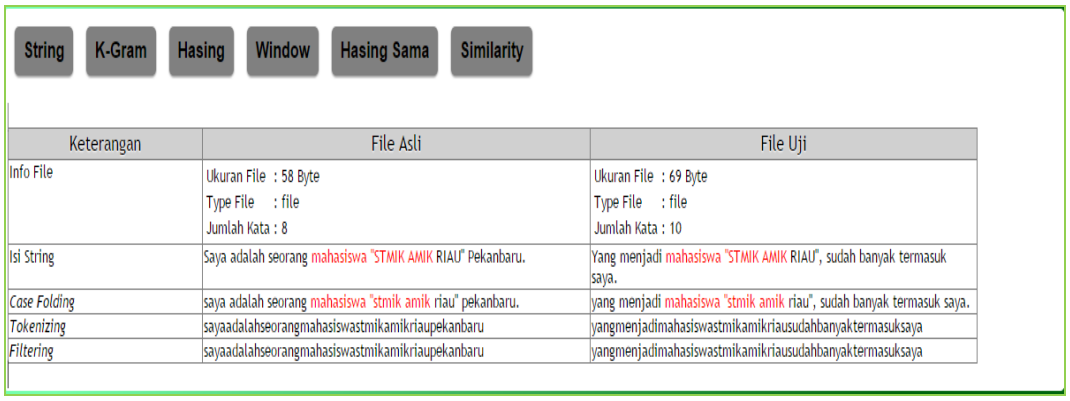
Gambar 3. Tampilan Halaman Data Laporan

3.3. Tampilan Halaman *Preview*

Pada kontrol “Lihat Kembali” akan ditampilkan halaman *preview*. Dimana data yang sudah kita proses diawal dapat kita lihat kembali tahapan-tahapannya, sehingga memberikan informasi yang kita butuhkan. Berikut tampilan sebagian halaman *preview*.

a) *String*

Pada halaman ini akan ditampilkan hasil proses *preprocessing* yang terdiri dari *case folding*, *tokenizing*, *filtering* dari file asli dan file uji, sehingga pada tampilan ini akan tampak perbedaan tahap per tahap yang dilewati pada proses pendeteksian dokumen teks. Gambar 4 berikut ini adalah tampilan untuk *string*

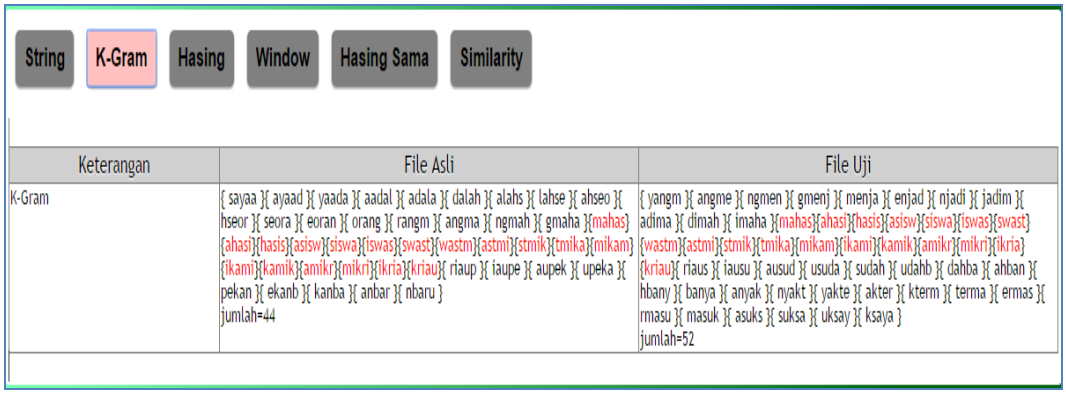


Keterangan	File Asli	File Uji
Info File	Ukuran File : 58 Byte Type File : file Jumlah Kata : 8	Ukuran File : 69 Byte Type File : file Jumlah Kata : 10
Isi String	Saya adalah seorang mahasiswa "STMIK AMIK RIAU" Pekanbaru.	Yang menjadi mahasiswa "STMIK AMIK RIAU", sudah banyak termasuk saya.
Case Folding	saya adalah seorang mahasiswa "stmik amik riau" pekanbaru.	yang menjadi mahasiswa "stmik amik riau", sudah banyak termasuk saya.
Tokenizing	sayaadalahseorangmahasiswastmikamikriaupekanbaru	yangmenjadimahasiswastmikamikriausudahbanyaktermasuksaya
Filtering	sayaadalahseorangmahasiswastmikamikriaupekanbaru	yangmenjadimahasiswastmikamikriausudahbanyaktermasuksaya

Gambar 4: Tampilan Halaman *String*

b) *K-Gram*

Tampilan proses pembentukan *K-Gram* dari masing-masing dokumen ditampilkan pada gambar 5 yang memperlihatkan dokumen asli dan dokumen uji. Tahap ini adalah proses pemotongan karakter sebanyak nilai *k* (*K-Gram*) yang sudah ditentukan, dan apabila terdapat kata yang sama dari pemotongan *K-Gram* tersebut langsung diberikan tanda merah

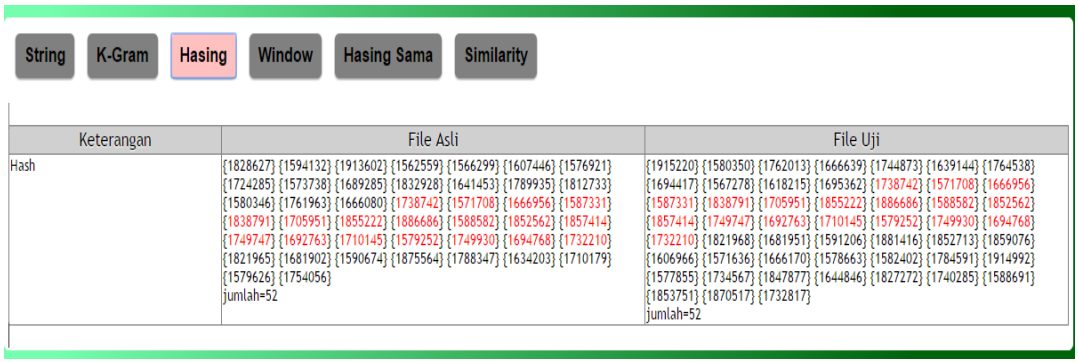


Keterangan	File Asli	File Uji
K-Gram	{ sayaa }{ ayaad }{ yaada }{ aadal }{ adala }{ dalah }{ alahs }{ lahse }{ ahseo }{ hseor }{ seora }{ eoran }{ orang }{ rangm }{ angma }{ ngmah }{ gmaha }{mahas}{ahasi}{hasis}{asisw}{siswa}{iswas}{swast}{wastm}{astmi}{stmik}{tmika}{mikam}{ikami}{kamik}{amikr}{mikri}{ikria}{kriau}{ riaup }{ iaupe }{ aupek }{ upeka }{ pekan }{ ekanb }{ kanba }{ anbar }{ nbaru } jumlah=44	{ yangm }{ angme }{ ngmen }{ gmenj }{ menja }{ enjad }{ njadi }{ jadim }{ adima }{ dimah }{ imaha }{mahas}{ahasi}{hasis}{asisw}{siswa}{iswas}{swast}{wastm}{astmi}{stmik}{tmika}{mikam}{ikami}{kamik}{amikr}{mikri}{ikria}{kriau}{ riaus }{ iausu }{ ausud }{ usuda }{ sudah }{ udahb }{ dahba }{ ahban }{ hbany }{ banya }{ anyak }{ nyakt }{ yakte }{ akter }{ kterm }{ terma }{ ermas }{ rmasu }{ masuk }{ asuks }{ suksa }{ uksay }{ ksaya } jumlah=52

Gambar 5: Tampilan Halaman *K-Gram*

c) *Hashing*

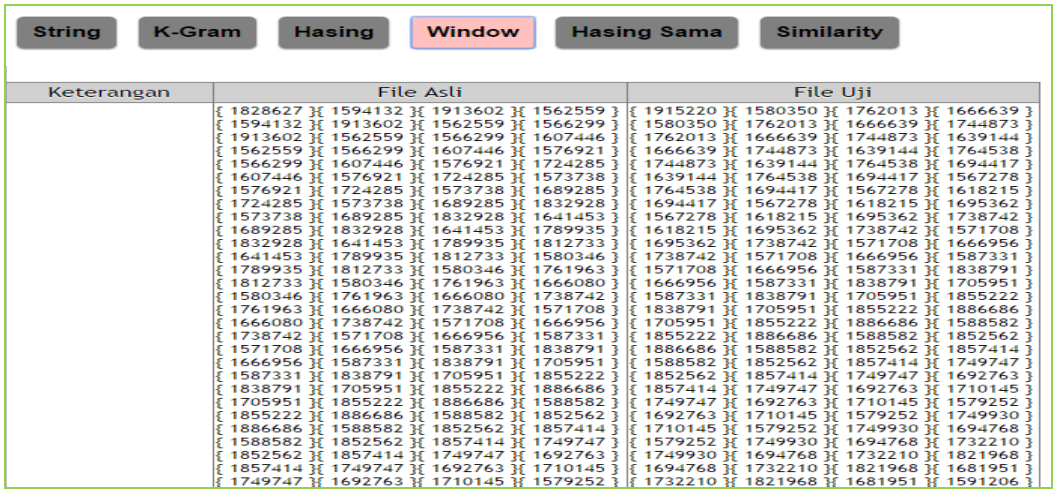
Pada halaman ini ditampilkan hasil proses *hashing*. Dimana pada proses ini, hasil *K-Gram* dicari nilai *ASCII*nya kemudian dihitung dengan rumus *hash* yang sudah dijabarkan pada bagian 2.1 sehingga menghasilkan angka-angka seperti yang diperlihatkan pada gambar 6



Gambar 6: Tampilan Halaman *Hashing*

d) *Windows*

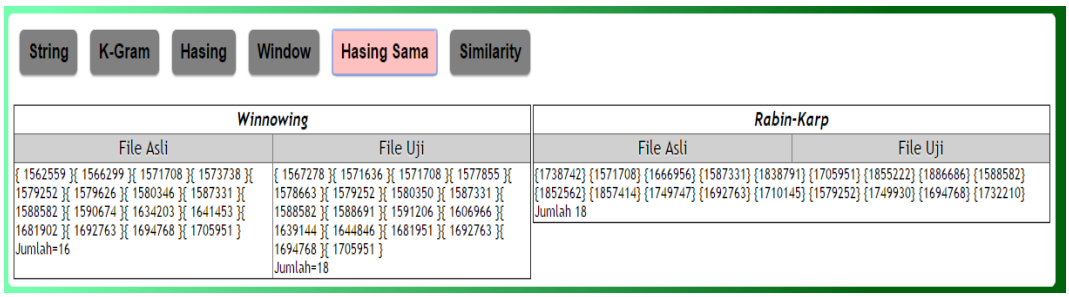
Gambar 7 memperlihatkan halaman proses *window*, yaitu memecah nilai hashing yang sama dengan menentukan berapa window yang akan di proses. Proses ini hanya dilakukan pada algoritma *winnowing*. *Window* yang digunakan adalah 4 *window* untuk masing-masing proses



Gambar 7: Tampilan Halaman *Windows*

e) *Hashing* Yang Sama

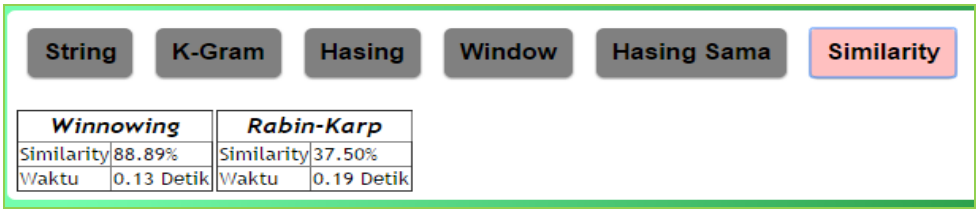
Dari hasil *hashing* pada proses sebelumnya, secara otomatis hasil *hash* yang memiliki kesamaan akan diberi tanda merah oleh sistem, agar dapat membedakan antara *hash* yang memiliki kesamaan dengan *hash* yang tidak memiliki kesamaan. Gambar 8 memperlihatkan tampilan hashing yang sama pada algoritma *rabin-karp*, sedangkan untuk algoritma *winnowing hashing* yang sama didapatkan dari *window*. Nilai *window* yang paling kecil dari masing-masing baris *window* adalah yang diambil sebagai nilai hash yang sama.



Gambar 8: Tampilan Halaman *Hashing* yang sama

f) *Similarity*

Gambar 9 merupakan tampilan penghujung dari aplikasi perbandingan pendeteksi kesamaan dokumen teks. Hasil akhir dari aplikasi ini adalah dapat diketahuinya tingkat persentase kemiripan (*similarity*) dokumen teks yang diuji antara dokumen asli dengan dokumen uji, dan juga dapat diketahui waktu yang dibutuhkan untuk proses pendeteksian kemiripan dokumen teks tersebut pada ukuran file yang sama. Pada aplikasi ini waktu untuk proses pendeteksian ditampilkan dalam satuan detik (*second*).



Gambar 9: Tampilan Halaman *Similiarity*

***g) Tampilan Halaman* Upload File**

Halaman *upload file* dokumen teks yang akan diuji diperlihatkan pada gambar 10. Tersedia 2 (dua) tempat *upload* file yaitu *upload* file asli dan *upload* file uji. Selanjutnya klik "proses" untuk memproses pengukuran similarity index dari kedua file tersebut

Gambar 10: Tampilan Halaman *Upload File*

h)* Hasil Perbandingan *Algoritma Rabin-Karp dan Winnowing

Tabel 11 memperlihatkan hasil akhir perbandingan antara algoritma Rabin-Karp dengan algoritma Winnowing berdasarkan akurasi dan waktu pemrosesan. Dari tabel dapat dilihat bahwa algoritma Winnowing lebih unggul dari tingkat akurasi dan lamanya waktu proses

yang dibutuhkan. Algoritma Rabin-Karp memiliki akurasi sebesar 37,5% dengan waktu proses selama 0,19 detik, sedangkan algoritma Winnowing memiliki tingkat akurasi sebesar 88,89% dengan waktu proses selama 0,13 detik.

Tabel 11. Perbandingan Algoritma Rabin-Karp dan Algoritma Winnowing

Algoritma	Tingkat Akurasi (%)	Waktu Proses (detik)
Rabin-Karp	37,50	0,19
Winnowing	88,89	0,13

4. Kesimpulan

Penelitian ini membuktikan bahwa algoritma Rabin-Karp dan algoritma Winnowing mampu mendeteksi tingkat kesamaan teks dengan level yang berbeda. Penelitian ini juga menggambarkan dan menguraikan tahapan penggunaan kedua algoritma yang diterapkan pada file yang sama. Hasil dari kedua algoritma dibandingkan berdasarkan 2 (dua) variabel yaitu tingkat akurasi dan waktu proses. Hasil percobaan membuktikan bahwa algoritma Winnowing lebih unggul dibandingkan dengan algoritma Rabin-Karp dari kedua variabel yang menjadi landasan perbandingan karena bukan hanya nilai hash saja yang dibandingkan, namun nilai hash yang ada akan dikonversikan terlebih dulu kedalam window baru sehingga dapat diketahui nilai hash yang sama. Aplikasi ini dapat mendeteksi kesamaan dokumen walaupun telah dilakukan perubahan posisi teks dokumen tersebut, meskipun mempengaruhi nilai similarity/persentase kesamaan menjadi turun.

Daftar Pustaka

[1] Oxford English Dictionary, https://en.oxforddictionaries.com/definition/plagiarism, retrieved 24 April 2018.
[2] Turnitin, https://www.turnitinuk.com/login_page.asp?lang=en_gb, retrieved 27 April 2018.
[3] Plagiarism Checker X. https://plagiarismcheckerx.com/, retrieved 27 April 2018.
[4] Grammarly, https://app.grammarly.com/, retrieved 27 April 2018.
[5] Dupli Checker, https://www.duplichecker.com/, retrieved 27 April 2018.
[6] J. Lin. *Brute Force and Indexed Approaches to Pairwise Document Similarity Comparisons with MapReduce.* SIGIR. Boston, Massachusetts, USA. 2009.
[7] F. Ture, T. Elsayed, J. Lin. *No Free Lunch: Brute Force vs. Locality-Sensitive Hashing for Cross-lingual Pairwise Similarity.* SIGIR. Beijing, China. 2011.
[8] G. Sidorov, D. Pinto. *Computing Text Similarity using Tree Edit Distance*. World Conference on Soft Computing (WconSC). Redmont, WA, USA. 2015.
[9] F. Hofmann. *Levenshtein Distance and Text Similarity in Python*. http://stackabuse.com/levenshtein-distance-and-text-similarity-in-python/, retrieved 28 April 2018.
[10] L. Salmela, J. Tarhio, P. Kalsi. Approximate Boyer-Moore String Matching for Small Alphabets. *Algorithmica*. 2010. Vol. 58:591.
[11] N. Ben Nsira, T. Lecrog, M. Elloumi. A Fast Boyer-Moore Type Pattern Matching Algorithm for Highly Similar Sequence. *International Journal of Data Mining and Bioinformatics*. 2015. Vol. 13(3). pp. 266-288.
[12] B. Leonardo, S. Hansun. Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winker Distance Algorithms. *Indonesian Journal of Electrical Engineering and Computer Science.* 2017. Vol. 5, No. 2. pp. 462-471.

[13] C. Supriyanto, S. Rakasiwi, A. Syukur. *A Comparison of Rabin Karp and Semantic-Based Plagiarism Detection.* 3rd International Conferences on Soft Computing, Intelligent System and Information Technology (ICSIIT). Bali, Indonesia. 2012.

[14] A. P. Utama Siahaan, Mesran, R. Rahim, D. Siregar. K-Gram as A Determinant of Plagiarism Level in Rabin-Karp Algorithm. *International Journal of Scientific and Technology Reserach.* 2017. Vol. 6, Issue 07.

[15] R. Sutoyo, I. Ramadhani, A. Dwi Ardiatma, et. al. *Detecting Documents Plagiarism using Winnowing Algorithm and K-Gram Method*. International Conference on Cybernetics and Computational Intelligence (CyberneticsCom). Phuket, Thailand. *2017*.

[16] *X. Duan, M. Wang, J. Mu.* A Plagiarism Detection based on Extended Winnowing. MATEC Web of Conferences. International Conference on Electronic Information Technology and Computer Engineering (EITCE). Zhuhai, China. 2017. Vol. 128.

[17] K. T. Tung, N. D. Hung, L. T. My Hanh. A Comparison of Algorithms used to Measure the Similarity between Two Documents. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*. 2015. Vol. 4 Issue 4.