RANCANG BANGUN APLIKASI PENDETEKSI PENJIPLAKAN DOKUMEN MENGGUNAKAN ALGORITMA *BIWORD WINNOWING*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik
Pada Jurusan Teknik Informatika

Oleh :

Muhammad Ridho
10851002911

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SLTAN SYARIF KASIM
PEKANBARU
RIAU

2013

DESIGN DETECTION OF DOCUMENT PLAGIARISM APPLICATION BY USING BIWORD WINNOWING ALGORITHM

MUHAMMAD RIDHO

10851002911

Final Exam Date: February 11th, 2013

Graduation Ceremony Period: 2013

Information Engineering Department

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Plagiarism is taking the essays or opinions of others without acknowledgment of the source text, and make it as their own essays . Nowadays, we have many algorithms that discusses how to detect plagiarism text documents, such as Rabin-Karp algorithm, Winnowing, and edit distance. In this research, will developed of Winnowing algorithm in detecting plagiarism. Winnowing algorithm is an algorithm that uses the approach of k-grams in shaping the document fingerprint. Fingerprints have formed a character-based techniques. This research tries to use a different fingerprint techniques, i.e Phrase-based techniques. Phrase-based techniques will split a text document into tokens biword. Tokens are encrypted to MD5, that token has the same hash value and can be used as long as fingerprinting text documents. By applying the approach biword Winnowing algorithm, this algorithm can check each document phrases and then stored in an array. So that to display text that has same values, the algorithm can show the array value in the form of token biword as a fingerprint of a document.

***Keywords**: Fingerprint, Hash Value, Plagiarism Document Text, Token Biword.*

RANCANG BANGUN APLIKASI PENDETEKSI PENJIPLAKAN DOKUMEN MENGGUNAKAN ALGORITMA *BIWORD WINNOWING*

MUHAMMAD RIDHO

10851002911

Tanggal Sidang: 11 Februari 2013

Periode Wisuda : 2013

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Penjiplakan merupakan pengambilan karangan atau pendapat orang lain tanpa mencantumkan sumber tulisan, dan menjadikannya seolah-olah karangan atau pendapat sendiri. Saat ini sudah banyak algoritma yang membahas cara mendeteksi penjiplakan dokumen teks seperti Algoritma *Rabin-karp, winnowing, dan edit distance*. Pada penelitian ini, akan dilakukan pengembangan dari Algoritma *Winnowing* dalam mendeteksi penjiplakan. Algoritma *winnowing* merupakan algoritma yang menggunakan pendekatan *k-grams* dalam membentuk *fingerprint* dokumen. *Fingerprint* yang dibentuk memiliki teknik yang berbasis *character-based.* Penelitian ini mencoba menggunakan teknik *fingerprint* yang berbeda, yaitu teknik *Phrase-based.* Teknik yang berbasis *phrase* ini akan memecah dokumen teks menjadi token-token *biword*. Token-token ini akan dienkripsi menjadi nilai MD5, agar token tersebut memiliki nilai *hash* yang sama panjang dan dapat dijadikan sebagai *fingerprint* dokumen teks. Dengan menerapkan pendekatan *biword* pada algoritma *winnowing*, algoritma ini dapat melakukan pengecekan frasa setiap dokumen dan kemudian disimpan dalam sebuah *array*. Sehingga dalam menampilkan teks yang memiliki nilai yang sama, algoritma dapat menampilkan kembali nilai *array* yang berbentuk token *biword* yang dianggap sebagai *fingerprint* sebuah dokumen.

Kata kunci: *Fingerprint*, Nilai *Hash*, Penjiplakan Dokumen Teks, Token *Biword*,.

DAFTAR ISI

DAFTAR GAMBAR

Gambar **Halaman**

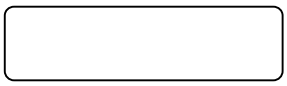
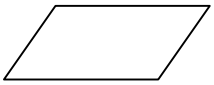
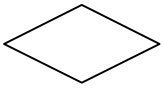
DAFTAR TABEL

Tabel **Halaman**

DAFTAR RUMUS

Rumus **Halaman**

DAFTAR SIMBOL

	Proses pada *flowchart*
	Start/Finish suatu proses pada *flowchart*
	Alur/langkah pada *flowchart* dan model data spasial
	Input/ Output pada *flowchart*
	Menyatakan kondisi pada *flowchart*
	Penghubung *flowchart* pada halaman berbeda

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi informasi yang semakin pesat telah membuat setiap orang untuk berusaha mengikutinya. Seiring dengan meningkatnya penggunaan komputer, sekarang sudah tidak lagi menyimpan berkas-berkas dokumen berupa *hardcopy* yang disimpan dalam lemari dan rak-rak buku, melainkan berupa *softcopy* atau dokumen digital. Semua dokumen yang memiliki file *softcopy* tersebut dapat diperbanyak dengan cara *copy-paste.* Namun kondisi tersebut memunculkan masalah baru dalam pelanggaran hak cipta, yaitu timbulnya tindakan *copy-paste* yang bermaksud untuk mengambil gagasan atau pendapat yang terdapat dalam dokumen tersebut, memperbanyak teori dan pendapat atau gagasan yang diambil tersebut tanpa mencantumkan sumbernya. Tindakan seperti ini disebut sebagai tindakan plagiarisme atau penjiplakan.

Banyak sekali tindakan yang dikatakan plagiat yang tidak disadari oleh setiap orang. Diantaranya adalah menggunakan gagasan, pendapat, pandangan atau teori tanpa mencantumkan sumbernya, mengubah kata-kata atau kalimat dari sebuah sumber tanpa menyebutkan rujukannya, bahkan mengakui karya orang lain sebagai karya sendiri. Dalam menyikapi tindakan tersebut, penulis ingin melakukan penelitian dengan mengembangkan sebuah aplikasi menggunakan algoritma yang dapat mengidentifikasi penjiplakan dokumen digital. Algoritma yang dimaksud adalah algoritma *winnowing.* Penelitian ini sebelumnya telah dilakukan oleh Diana Purwitasari dari Institut Sebelas Maret (ITS) pada tahun 2011. Namun disini penulis mencoba mengembangkan algoritma tersebut dengan tujuan yang sama yaitu mengetahui persentase kemiripan antar dokumen yang diuji.

Dalam penelitiannya dijelaskan bahwa algoritma *winnowing* ini digunakan karena telah memenuhi salah satu syarat algoritma penjiplakan, yaitu *whitespace intensitivity*, membuang karakter yang tidak relevan seperti tanda baca dan karakter lainnya. Algoritma ini menggunakan teknik *fingerprint* dalam mendeteksi kemiripan dokumen. Teknik *fingerprint* yang digunakan pada algoritma *winnowing* ini berbasiskan karakter. Teknik ini melakukan penelusuran teks dokumen menggunakan urutan karakter dalam dokumen teks. Pada tugas akhir ini penulis mencoba mengembangkan algoritma *winnowing* menggunakan teknik *fingerprint* yang berbasiskan frasa. Teknik ini mengelompokkan teks dokumen menjadi kumpulan dua buah kata atau disebut dengan *biword. Biword* yang dibentuk bertujuan untuk mempertahankan arti kata atau frasa pada teks dokumen. Konsep *biword* memberikan token kata lebih sedikit dibandingkan dengan *triword* maupun *quadword.* Dengan demikian *biword* lebih utama dalam mempertahankan frasa atau arti kata. Setelah dilakukan pembentukan beberapa *biword* pada masing-masing dokumen, selanjutnya dilakukan penelusuran dengan membandingkan *biword* antar dokumen. Kemudian akan dilakukan pengecekkan kumpulan *biword* yang telah dibentuk untuk menampilkan kata-kata yang memiliki *fingerprint* sama yang ada pada masing-masing dokumen.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka dapat ditarik sebuah rumusan masalah yaitu bagaimana merancang dan membangun sebuah aplikasi pendeteksi plagiarisme dokumen menggunakan algoritma *winnowing* dengan pendekatan *biword.*

1.3 Batasan Masalah

Batasan masalah dalam pembuatan tugas akhir ini adalah:

1. Dokumen yang diuji berupa dokumen teks digital.
2. Dokumen yang diuji memiliki perbandingan 1 : 1.
3. Tidak memperhatikan kalimat aktif dan pasif

4. Aplikasi ini membandingkan dokumen teks yang bersifat *verbatim copy*
5. Aplikasi tidak membandingkan pembentukan kata dengan konsep *triword* maupun *quadword*
6. Aplikasi tidak memberikan nilai sebuah dokumen sebagai dokumen penjiplak
7. Aplikasi tidak memperhatikan kata-kata semantik dan kata-kata yang memiliki makna sinonim.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam pembuatan tugas akhir ini adalah:

1. Terciptanya aplikasi pendeteksi plagiarisme dengan menggunakan algoritma *winnowing* dengan pendekatan *biword.*
2. Diketahuinya persentase kesamaan antara dokumen yang diuji dengan algoritma *biword-winnowing*
3. Diketahuinya kalimat yang memiliki nilai kesamaan antar dokumen yang diuji

1.5 Sistematika Penulisan

Bab I Pendahuluan

Dalam bab ini penulis memaparkan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian tugas akhir dan sistematika penulisan laporan.

Bab II Landasan Teori

Bab ini berisi mengenai studi literatur atau teori penunjang yang digunakan sebagai landasan dalam pembuatan tugas akhir. Diantaranya adalah definisi plagiarisme, metode mendeteksi plagiarisme, *information retrieval,* algoritma *winnowing,* ASCII, MD5, penerapan algoritma *winnowing,* dan penerapan konsep *biword* pada algoritma *winnowing.*

Bab III Metodologi Penelitian

Bab ini berisi tentang tahapan atau cara-cara yang dilakukan dalam mencapai tujuan penelitian tugas akhir. Yaitu: pengumpulan data, analisa aplikasi, perancangan aplikasi, implementasi, pengujian, kesimpulan dan saran.

Bab IV Analisa dan Perancangan

Bab ini berisi tentang analisa dalam pembuatan sistem dan juga menjelaskan cara perancangan sistem yang akan dibuat. Diantaranya adalah: analisa aplikasi pendeteksi plagiarisme, analisa algoritma *winnowing*, algoritma *winnowing* dengan pendekatan *biword*, dan perancangan aplikasi,

Bab V Implementasi dan Pengujian

Bab ini berisi tentang langkah-langkah pembuatan sistem pendeteksian plagiarisme dokumen dan menguji hasil dari sistem yang telah dibangun. Meliputi: tahapan implementasi, hipotesa pengujian aplikasi, pengujian aplikasi, hasil pengujian, kesimpulan pengujian.

Bab VI Penutup

Bab ini berisi tentang kesimpulan dan saran-saran yang ada dalam pembuatan sistem dan juga laporan penelitian tugas akhir ini.

BAB II
LANDASAN TEORI

2.1. Plagiarisme

Plagiarisme adalah mengambil ide-ide atau kata-kata orang lain dan mengakui sebagai milik sendiri. Plagiarisme adalah jenis pencurian intelektual. Plagiarisme dapat mengambil banyak bentuk, dari kecurangan yang disengaja untuk sengaja menyalin dari sumber tanpa pengakuan. (The Learning Centre,UNSW Sydney).

Sedangkan menurut Kamus Besar Bahasa Indonesia (KBBI), Plagiat merupakan pengambilan karangan atau pendapat orang lain dan menjadikannya seolah-olah karangan atau pendapat sendiri, misalnya menerbitkan karya tulis orang lain atas nama dirinya sendiri. (KBBI, Edisi III 2005).

Plagiarisme tidak selalu dilakukan dengan sengaja, ada kalanya perbuatan ini bersifat tidak disengaja, kebetulan dan dapat mencakup pencurian sendiri (*self stealing*). Berikut ini beberapa sifat plagiarisme (Steven, 2009):

1. Kebetulan (*accidental*)

   Praktik plagiarisme ini dapat terjadi karena kurangnya pengetahuan akan plagiarisme dan pemahaman mengenai penulisan referensi.
2. Tidak disengaja (*unintentional*)

   Ketersediaan informasi dalam jumlah yang sangat besar mempengaruhi pemikiran sehingga ide yang sama dapat dihasilkan secara tertulis maupun lisan sebagai milik pribadi.

3. Disengaja (*intentional*)

   Tindakan menyalin sebagian atau keseluruhan hasil karya orang lain secara sengaja tanpa mengikutsertakan nama pemilik hasil karya.

4. Diri sendiri (*self plagiarism*)

   Penggunaan hasil karya yang dibuat diri sendiri dalam bentuk lain tanpa menunjuk hasil karya asli.

2.2. Metode Mendeteksi Plagiarisme

Dalam melakukan pendeteksian penjiplakan terdapat tiga metode (Ana Kurniawati, 2008) yaitu:

1. Perbandingan Teks Lengkap

   Metode ini diterapkan dengan membandingkan semua isi dokumen. Namun pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif. Algoritma yang digunakan pada metode ini adalah algoritma *Brute Force*, algoritma *Edit Distance*, algoritma *Boyer Moore*.

   Pada algoritma *Edit Distance,* untuk mendeteksi plagiat pada dua teks sumber adalah dengan cara memasukkan isi tiap *file* sumber ke dalam string. Dalam proses memasukkan isi *file,* teks yang dimasukkan ke dalam string adalah teks program tanpa komentar.

   Jadi, ketika karakter yang dibaca dari teks sumber adalah penanda komentar maka teks dalam komentar tersebut tidak akan dimasukkan ke dalam string. Untuk mengecek keabsahan suatu teks digunakan persentase kemiripan (P) yang dapat dihitung dengan cara perbandingan hasil keluaran algoritma edit distance (D) dengan jumlah karakter terpanjang antara 2 masukan (T). Dari nilai persentase dapat diketahui besarnya tingkat kemiripan antara kedua teks masukan, apakah masih dalam batas toleransi (80%). Jika ternyata P lebih besar dari 80%, dapat diambil kesimpulan bahwa telah terjadi sebuah praktik

plagiat. Namun, hal di atas tidaklah mutlak karena pengguna dapat menentukan sendiri nilai batas toleransi.

2. Dokumen *Fingerprint*

   Dokumen *fingerprint* merupakan metode yang digunakan untuk mendeteksi keakuratan kesamaan antar dokumen. Prinsip kerja dari metode dokumen *fingerprint* ini dengan menggunakan teknik *hashing*. Teknik *hashing* adalah sebuah fungsi yang menkonversi setiap *string* menjadi bilangan. Algoritma yang digunakan pada metode ini seperti algoritma *Winnowing*, algoritma *Manber* dan algoritma *Rabin-Karp.*

3. Kesamaan Kata Kunci

   Prinsip dari metode kesamaan kata kunci adalah mencari kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen lain.

Sedangkan Untuk melakukan pendeteksian plagiarisme dokumen teks sebuah algoritma harus memenuhi salah satu persyaratan berikut ini (Schleimer dkk, 2003):

1. *Whitespace Insensitivity*, yang berarti dalam melakukan pencocokan terhadap dokumen teks seharusnya tidak terpengaruh oleh spasi, jenis huruf (kapital atau normal), tanda baca, simbol-simbol dan sebagainya.
2. *Noise Surpression*, yang berarti menghindari penemuan kecocokan dengan panjang kata yang terlalu kecil atau kurang relevan, misal: ‘the’. Panjang kata yang ditengarai merupakan penjiplakan harus cukup untuk membuktikan bahwa kata-kata tersebut telah dijiplak dan bukan merupakan kata yang umum digunakan.
3. *Position Independence*, yang berarti penemuan kecocokan atau kesamaan tidak harus bergantung pada posisi kata-kata. Meskipun berada pada posisi yang tidak sama, kecocokan atau kesamaan harus dapat ditemukan.

2.3. *Information Retrieval*

Information Retrieval adalah studi tentang sistem pengindeksan, pencarian, dan mengingat data, khususnya teks atau bentuk tidak terstruktur lainnya. (Manning, 2009). Information Retrieval merupakan bagian dari *computer science* yang berhubungan dengan pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri. *Information Retrieval* merupakan suatu pencarian informasi (biasanya berupa dokumen) yang didasarkan pada suatu *query* (masukkan *user*) yang diharapkan dapat memenuhi keinginan *user* dari kumpulan dokumen yang ada. Sedangkan, definisi *query* dalam *Information Retrieval* merupakan sebuah formula yang digunakan untuk mencari informasi yang dibutuhkan oleh *user* dan merupakan suatu *keywords* (kata kunci) dalam bentuk yang paling sederhana.

Dalam *Information Retrieval,* ada beberapa tahapan yang dapat dilakukan dalam pengelolaan dokumen teks (Manning, 2009), diantaranya:

2.3.1 *Preprocessing*

Pada sebuah teks dokumen, informasi yang akan digali berisi informasi-informasi yang strukturnya sembarang. Oleh karena itu, diperlukan proses pengubahan bentuk menjadi data yang terstruktur sesuai kebutuhannya yang di kenal dengan *prepocessing.*

Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan atau pembersihan teks yang dilakukan untuk mengubah data data berkulitas yaitu data yang telah memenuhi persyaratan untuk dieksekusi pada sebuah algoritma. Bentuk pembersihan teks ini seperti menghilangkan spasi, tanda baca, mengubah huruf kapital menjadi huruf kecil dan menghilangkan karakter-karakter yang tidak relevan lainnya.

2.3.2 Tokenisasi

Poses tokenisasi yaitu proses pemisahan kata dari teks dokumen secara dengan menggunakan karakter spasi sebagai tanda pemisahnya.

Dengan adanya pemisahan kata terlebih dahulu, *string* yang diinputkan akan terlihat lebih ringkas karena ditampilkan dalam bentuk tiap kata sesuai dengan spasi yang memisahkannya.

Pemisahan kata atau *fingerprint* pada teks dokumen memiliki beberapa teknik yang sesuai (Chow Kok Kent, 2010):

a. *Character-based*

   Teknik ini menggunakan urutan karakter untuk membentuk *fingerprint* pada semua dokumen.

   Misalnya, jika kita memiliki dokumen dengan panjang D= 5 yang memiliki satu kata "touch", maka kita dapat melihat bahwa "touc" dan "ouch" adalah semua kemungkinan *substring* panjang K = 4.

   Pada dasarnya, membandingkan dua dokumen menggunakan teknik ini adalah menghitung jumlah *substring* di kedua *fingerprint.*

b. *Phrase-based.*

   Teknik untuk menghasilkan sidik jari menggunakan mekanisme frase untuk mengukur kemiripan antara dua dokumen ini pertama kali diperkenalkan oleh Lyon et al pada tahun 2001. Pada tahap awal, kita harus mengkonversi setiap dokumen untuk satu set *bigram* (dua kata) atau *trigram* (tiga kata).

   Misalnya kalimat “aplikasi pendeteksi penjiplakan teks dokumen” akan dikonversi ke bentuk *triword,* sehingga menghasilkan “aplikasi pendeteksi penjiplakan, pendeteksi penjiplakan teks, penjiplakan teks dokumen”. Kemudian set dari *triword* untuk setiap dokumen dibandingkan dengan dokumen lain.

c. *Statement-based*

Pro dan kontra dari *character-based* dan *frase-based fingerprint* telah membuat munculnya *fingerprint* untuk setiap pernyataan. Meskipun setiap nilai K dapat dipertimbangkan, namun K = 4 dinyatakan sebagai pilihan ideal oleh Yerra dan Ng (2005). Hal ini karena nilai-nilai yang lebih kecil dari K (yaitu, K = 1, 2, atau 3) tidak memberikan diskriminasi yang baik antara kalimat.

2.3.3 Metode K-grams

Metode K-grams merupakan salah satu metode yang terdapat dalam proses tokenisasi. Metode *K-grams* ini digunakan untuk membentuk *substring* sepanjang *k* karakter dari sebuah *string*. Biasanya yang dijadikan *substring* adalah kata. Semakin kecil nilai karakter K , maka pencarian nilai similaritas semakin efektif.

2.4. Algoritma *Winnowing*

Algoritma *Winnowing* merupakan algoritma yang digunakan dalam deteksi penjiplakan termasuk bagian-bagian kecil yang mirip dalam dokumen yang berjumlah banyak. Input dari algoritma ini adalah dokumen teks yang diproses sehingga menghasilkan *output* berupa kumpulan nilai-nilai *hash*, kumpulan-kumpulan nilai *hash* tersebut selanjutnya disebut *fingerprint*. *Fingerprint* inilah yang dijadikan dasar pembanding antara *file-file* teks yang telah dimasukkan dan digunakan dalam deteksi penjiplakan (Schleimer dkk, 2003) .

Berikut Langkah-langkah yang dapat dilakukan dalam penerapan Algoritma Winnowing (Schleimer dkk, 2003):

2.4.1 *Prepocessing*

Menghilangkan karakter yang tidak relevan pada dokumen teks, seperti tanda baca, tanda spasi dan mengubah huruf besar menjadi kecil.

Contoh:

Diberikan sebuah kalimat “Teknik Informatika adalah salah satu jurusan yang terdapat di Fakultas Sains dan Teknologi”:

Setelah dilakukan proses *preprocessing*, sehingga terbentuk teks berikut:

“teknikinformatikaadalahsalahsatujurusanyangterdapatdifakultassainsdanteknologi”

2.4.2 Metode K-gram

Metode *K-gram* merupakan metode yang digunakan dalam proses tokenisasi atau pemisahan teks, dengan cara membentuk *substring* sepanjang *k* karakter dari sebuah *string.*

Contoh:

Memotong *string* sepanjang *k.* misalnya nilai *k* = 7, dari kalimat diatas, sehingga diperoleh hasil sebagai berikut:

```
tekniki eknikin knikinf nikinfo ikinfor kinform informa nformat
formati ormatik rmatika matikaa atikaad tikaada ikaadal kaadala
aadalah adalahs dalahsa alahsal lahsala ahsalah hsalahs salahsa
alahsat lahsatu ahsatuj hsatuju satujur atujuru tujurus ujurusa
jurusan urusany rusanya usanyan sanyang anyangt nyangte yangter
angterd ngterda gterdap terdapa erdapat rdapatd dapatdi apatdif
patdifa atdifak tdifaku difakul ifakult fakulta akultas kultass
ultassa ltassai tassain assains ssainsd sainsda ainsdan insdant
nsdante sdantek dantekn antekno nteknol teknolo eknolog knologi
```

Dari penelitian yang dilakukan oleh peneliti dari Institut Teknologi Sepuluh November, nilai *k* yang dianjurkan adalah bernilai 30, hal tersebut bertujuan untuk mendapatkan hasil yang maksimal. (Putu Yuwono dkk, 2009).

2.4.3 *Rolling Hash*

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula). Keluaran fungsi *hash* disebut juga nilai *hash* (*hash-value*) atau pesan ringkas (*message digest*).

Nama lain fungsi *hash* adalah:

- fungsi kompresi/kontraksi (*compression function*)
- cetak-jari (*fingerprint*)
- *cryptographic checksum*
- *message integrity check* (MIC)
- *manipulation detection code* (MDC)

Fungsi *hash* yang banyak dipakai di dalam aplikasi kriptografi adalah *MD5* dan *SHA*. Fungsi *hash* sering kali dihubungkan dengan perhitungan jumlah *bit* dari segmen pada data komputer yang dikalkulasi sebelum dan sesudah transmisi atau penyimpanan untuk memastikan bahwa data bebas dari kesalahan (*checksum*), pemeriksaan digit, fungsi acak, kode perbaikan kesalahan, dan fungsi *hash* kriptografi. Walaupun konsep-konsep tersebut saling melengkapi, setiap konsep mempunyai kegunaan dan persyaratannya sendiri.

Ada teori dari fungsi *hash* yang dikenal sebagai fungsi *rolling hash*. *Rolling hash* merupakan teknik yang digunakan untuk mendapatkan nlai *hash* dari rangkaian *grams* yang telah terbentuk dari metode *k-grams*. *Rolling hash* berfungsi untuk mempercepat komputasi nilai *hash* dari rangkaian *grams* selanjutnya yang telah terbentuk. Nilai *hash* yang baru dapat dengan cepat dihitung dari nilai *hash* yang lama dengan cara menghilangkan nilai lama dari kelompok *hash* dan menambahkan nilai baru ke dalam kelompok tersebut.

Berikut persamaan dari metode *hash*:

$$H_{(c1\ldots ck)} = c_1 * b^{(k-1)} + c_2 * b^{(k-2)} + \cdots + c_{(k-1)} * b^k + c_k \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Keterangan:

c: nilai *ascii* karakter (desimal)

b**:** basis (bilangan prima)

k: banyak karakter (indeks karakter)

Keuntungan dari *rolling hash* adalah untuk nilai *hash* berikutnya. Untuk mendapatkan nilai *hash* dari metode *k-grams* selanjutnya digunakan persamaan *rolling hash* dibawah ini:

$$H_{(c2\ldots ck+1)} = (H_{(c1\ldots ck)} - c_1 * b^{(k-1)}) * b + c_{(k+1)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

Dengan demikian tidak perlu melakukan iterasi dari indeks pertama sampai terakhir untuk menghitung nilai *hash* untuk *gram* ke-2 sampai terakhir. Hal ini tentu dapat mengehemat biaya komputasi saat menghitung nilai *hash* dari sebuah *gram.*

2.4.4 Pembentukan *Window*

Nilai-nilai *hash* yang telah terbentuk, selanjutnya dibentuk dalam beberapa *window* dengan ukuran *W. Window* merupakan pembagian atau pengelompokan beberapa nilai *hash* dengan ukuran yang ditentukan. Dari *window* yang telah dibentuk dilakukan pemilihan nilai *hash* terkecil pada tiap *window* untuk dijadikan *fingerprint* tiap dokumen.

2.4.5 *Jaccard Coefficient*

Jaccard Coefficient merupakan persamaan yang digunakan untuk menentukan tingkat kemiripan antara dua dokumen teks pada algoritma *winnowing*. Langkah ini,

dilakukan setelah melakukan perhitungan nilai *hash* dan memilih *fingerprint* yang terkecil dari dua dokumen teks (Schleimer dkk, 2003). Berikut persamaan *jaccard coefficient*:

$$\text{Similaritas}(d_i,d_j) = \frac{|W(di) \cap W(dj)|}{|W(di) \cup W(dj)|} \times 100\% \quad \text{.....(2.3)}$$

Keterangan:

$W(d_i)$: *fingerprint* terkecil dokumen teks 1

$W(d_j)$: *fingerprint* terkecil dokumen teks 2

2.5. *ASCII*

ASCII merupakan singkatan dari *American Standard Code for Information Interchange* merupakan standar yang berlaku di seluruh dunia untuk kode berupa angka yang merepresentasikan karakter-karakter, baik huruf, angka, maupun simbol yang digunakan oleh komputer. Terdapat 128 karakter standar ASCII yang masing-masing direpresentasikan oleh tujuh digit bilangan *biner* mulai dari 0000000 hingga 1111111. Dengan menstandarisasi nilai-nilai yang digunakan untuk karakter-karakter ini, *ASCII* memungkinkan komputer dan program komputer untuk saling bertukar informasi. *ASCII* menyediakan 256 kode yang dibagi ke dalam dua himpunan. Himpunan ini merepresentasikan total kombinasi dari 7 atau 8 *bit*, yang kemudian menjadi angka dari *bit* dalam 1 *byte*.

2.6. MD5 (*Message Digest* 5)

MD5 adalah sebuah fungsi matematika yang menerima input sebuah angka dan memberikan *output* berupa *hash value* dari *input*, dengan *hash value* 128-bit atau 32 karakter. Fungsi *hash* yang banyak dipakai di dalam aplikasi kriptografi adalah *MD5* dan SHA. Sedangkan SHA memberikan *hash value* sepanjang 40 karakter

Fungsi inti MD5 adalah untuk melindungi data dari modifikasi yang tidak terdeteksi, dapat dihitung hasil fungsi *hash* dari data tersebut, selanjutnya dapat menghitung hasil fungsi *hash*nya lagi dan membandingkannya dengan hasil fungsi *hash* yang sebelumnya apabila terjadi perubahan. Dalam arti lain, MD5 atau fungsi *hash* tidak ditujukan untuk merahasiakan data, namun untuk pengecekan kebenaran data.

Algoritma MD5 adalah fungsi *hash* satu arah yang dibuat oleh *Ron Rivest* dan merupakan pengembangan dari algoritma MD4 yang berhasil diserang oleh kriptanalis. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang (Rinaldi Munir, 2004).

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran (*message digest*) yang panjangnya tetap (*fixed*) dan biasanya dengan ukuran yang jauh lebih kecil dari ukuran *string* semula.

Contoh Aplikasi Enkripsi MD5:

Kata “supono” di enkripsi menggunakan MD5 akan berubah menjadi “9008a28a8a5d07db3091d9114a839268″. Jumlahnya akan menjadi 32 karakter, berapapun masukkannya akan menghasilkan *output* enkripsi sepanjang 32.

2.7. Contoh Penerapan Algoritma Winnowing

Secara garis besar, langkah-langkah dalam penerapan algoritma *Winnowing* adalah sebagai berikut:

1. Penghapusan karakter-karakter yang tidak relevan (*whitespace insensitivity*).
2. Pembentukan rangkaian *gram* dengan ukuran k.
3. Penghitungan nilai *hash.*
4. Membagi ke dalam *window* tertentu.
5. Pemilihan beberapa nilai *hash* menjadi *document fingerprinting*.

Berikut adalah contoh kalimat yang akan di deteksi menggunakan algoritma winnowing:

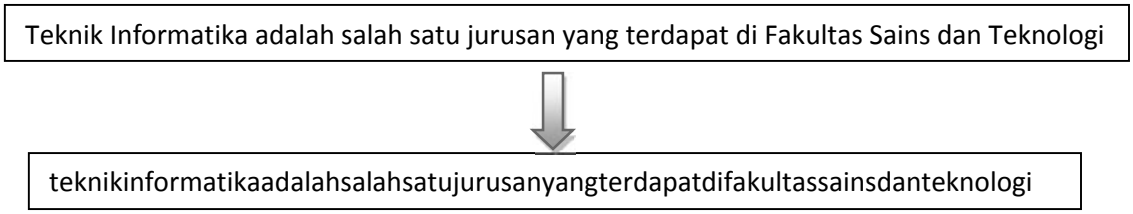
Kalimat 1: “Teknik Informatika adalah salah satu jurusan yang terdapat di Fakultas Sains dan Teknologi”

Kalimat 2: “Teknik Informatika adalah salah satu jurusan terfavorit di FASTE”

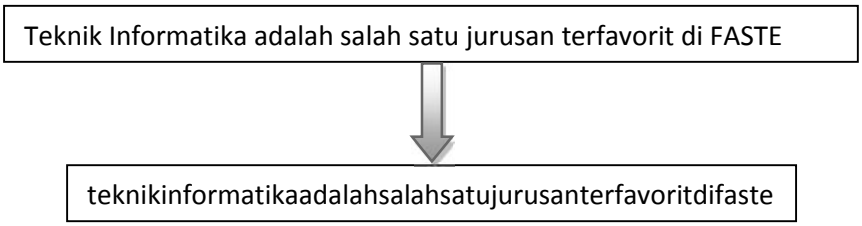
Langkah-langkah penerapan algoritmanya adalah:

1. *Whitespace Insensitivity*, membuang karakter-karakter dari isi dokumen yang tidak relevan seperti tanda baca dan huruf kapital.

Kalimat 1:



Kalimat 2:



2. Membentuk rangkaian *k-grams*, misalnya dengan ukuran 7

Untuk kalimat 1, terbentuklah rangkaian *k-grams* sebagai berikut:

```
tekniki eknikin knikinf nikinfo ikinfor kinform informa nformat
formati ormatik rmatika matikaa atikaad tikaada ikaadal kaadala
aadalah adalahs dalahsa alahsal lahsala ahsalah hsalahs salahsa
alahsat lahsatu ahsatuj hsatuju satujur atujuru tujurus ujurusa
jurusan urusany rusanya usanyan sanyang anyangt nyangte yangter
angterd ngterda gterdap terdapa erdapat rdapatd dapatdi apatdif
patdifa atdifak tdifaku difakul ifakult fakulta akultas kultass
ultassa ltassai tassain assains ssainsd sainsda ainsdan insdant
nsdante sdantek dantekn antekno nteknol teknolo eknolog knologi
```

Sedangkan untuk kalimat 2, membentuk rangkaian *k-grams* sebagai berikut:

```
tekniki eknikin knikinf nikinfo ikinfor kinform informa nformat
formati ormatik rmatika matikaa atikaad tikaada ikaadal kaadala
aadalah adalahs dalahsa alahsal lahsala ahsalah hsalahs salahsa
alahsat lahsatu ahsatuj hsatuju satujur atujuru tujurus ujurusa
jurusan urusant rusante usanter santerf anterfa nterfav terfavo
erfavor rfavori favorit avoritd voritdi oritdif ritdifa itdifas
tdifast difaste
```

3. Perhitungan nilai *hash* menggunakan persamaan 2.1, dimana b=2 dan k=7:

Perhitungan untuk Kalimat 1:

$H_{(tekniki)}$ = ascii(t) * $2^{(6)}$ + ascii(e) * $2^{(5)}$ + ascii(k) * $2^{(4)}$ + ascii(n) * $2^{(3)}$ + ascii(i) * $2^{(2)}$ + ascii(k) * $2^{(1)}$ + ascii(i) * $2^{(0)}$

= 116 * 64 + 101 * 32 + 107 * 16 + 110 * 8 + 105 * 4 + 107 * 2 + 105 * 1

= 7424 + 3232 + 1712 + 880 + 420 + 214 + 105

= 13987

$H_{(eknikin)}$ = (13987 - ascii(t) * $2^{(6)}$) * 2 + ascii(n) * $2^{(0)}$

= (13987 – 116 * 64) * 2 + 110 * 1

= (13987 – 7424) * 2 + 110 * 1

= (6563 * 2) + 110

= 13126 + 110

= 13236

$H_{(knikinf)}$ = (13236 - ascii(e) * $2^{(6)}$) * 2 + ascii(f) * $2^{(0)}$

= (13236 – 101 * 64) * 2 + 102 * 1

= (13236 – 6464) * 2 + 102 * 1

= (6772 * 2) + 102

= 13544 + 102

= 13646

Maka di peroleh hasil perhitungannya seperti berikut:

13987 **13236** 13646 13707 **13448** 13565 13531 13738 **13501** 14053 13995
13495 **13138** 13957 13174 **13005** **12418** **12535** **12751** **12810** 13301 **12882**
13463 13711 **12818** 13337 **12956** 13613 14028 **13453** 14605 14459 **14052**
14657 14435 14388 **13903** **13202** 14089 14212 **13036** 13753 13538 13989
13246 13664 **12841** **12984** 13649 13069 13839 **12938** 13192 **13041** **13141**
13981 14363 **13855** 13996 **13259** 14202 13781 **12952** 13604 13869 13765
12920 **13151** 13994 14019 **13293** 13763

Perhitungan untuk Kalimat 2:

$H_{(tekniki)}$ = ascii(t) * $2^{(6)}$ + ascii(e) * $2^{(5)}$ + ascii(k) * $2^{(4)}$ + ascii(n) * $2^{(3)}$ + ascii(i) * $2^{(2)}$ + ascii(k) * $2^{(1)}$ + ascii(i) * $2^{(0)}$

= 116 * 64 + 101 * 32 + 107 * 16 + 110 * 8 + 105 * 4 + 107 * 2 + 105 * 1

= 7424 + 3232 + 1712 + 880 + 420 + 214 + 105

= 13987

$H_{(eknikin)}$ = (13987 - ascii(t) * $2^{(6)}$) * 2 + ascii(n) * $2^{(0)}$

= (13987 – 116 * 64) * 2 + 110 * 1

= (13987 – 7424) * 2 + 110 * 1

= (6563 * 2) + 110

= 13126 + 110

= 13236

$H_{(knikinf)}$ = (13236 - ascii(e) * $2^{(6)}$) * 2 + ascii(f) * $2^{(0)}$

= (13236 – 101 * 64) * 2 + 102 * 1

= (13236 – 6464) * 2 + 102 * 1

= (6772 * 2) + 102

= 13544 + 102

= 13646

Maka di peroleh hasil perhitungannya seperti berikut:

13987 **13236** 13646 13707 **13448** 13565 13531 13738 **13501** 14053 13995
13495 **13138** 13957 13174 **13005** **12418** **12535** **12751** **12810** 13301 **12882**
13463 13711 **12818** 13337 **12956** 13613 14028 **13453** 14605 14459 **14052**
14652 14429 14380 **13886** **13149** 14000 14031 **13328** 13833 **13190** **13424**
14537 14072 14033 **13589** 13854 **12961**

4. Membentuk *window* dari nilai-nilai *hash* yang telah diperoleh dari perhitungan. Pada contoh dibawah ini menggunakan nilai $w = 4$:

[13987 **13236** 13646 13707]
[13236 13646 13707 13448**]**
[13646 13707 **13448** 13565]
[13707 13448 13565 13531]
[13448 13565 13531 13738]
[13565 13531 13738 **13501**]
[13531 13738 13501 14053]
[13738 13501 14053 13995]
[13501 14053 13995 **13495**]
[14053 13995 13495 **13138**]
[13995 13495 13138 13957]
[13495 13138 13957 13174]
[13138 13957 13174 **13005**]
[. . . . . .]
[. . . . . .]
[13869 13765 12920 13151]
[13765 12920 13151 13994]
[12920 13151 13994 14019]
[**13151** 13994 14019 13293]
[13994 14019 **13293** 13293]
[14019 13293 13293 13763]

Kalimat 2:

[13987 **13236** 13646 13707]
[13236 13646 13707 13448**]**

[13646 13707 **13448** 13565]
[13707 13448 13565 13531]
[13448 13565 13531 13738]
[13565 13531 13738 **13501**]
[13531 13738 13501 14053]
[13738 13501 14053 13995]
[13501 14053 13995 **13495**]
[14053 13995 13495 **13138**]
[13995 13495 13138 13957]
[13495 13138 13957 13174]
[13138 13957 13174 **13005**]
[13957 13174 13005 **12418]**
[13174 13005 12418 12535]
[13005 12418 12535 12751]
[12418 12535 12751 12810]
[12535 12751 12810 13301]
[12751 12810 13301 12882]
[12810 13301 12882 13463]
[13301 **12882** 13463 13711]
[12882 13463 13711 **12818]**
[13463 13711 12818 13337]
[. . . . . .]
[. . . . . .]
[**13424** 14537 14072 14033]
[14537 14072 14033 **13589**]
[14072 14033 13589 13854]
[14033 13589 13854 **12961**]

5. Pilih nilai *hash* minimum dengan posisi paling kanan dari setiap *window* yang telah terbentuk untuk dijadikan *fingerprint* (penanda), jika nilai tersebut berada pada posisi sama yaitu paling kanan maka kedua nilai *hash* tersebut dijadikan sebagai *fingerprint*. Berikut nilai *hash* yang dihasilkan berdasarkan indeks:

Untuk Kalimat 1:

[13236, 1] [13448, 4] [13501, 8] [13495, 11] [13138, 12] [13005, 15] [12418, 16] [12535, 17] [12751, 18] [12810,19] [12882, 21] [12818, 24] [12956, 26] [13453,29] [14052,32] [13903,36] [13202,37] [13036,40] [13246,44] [12841,46] [12984,47] [12938,51] [13041,53] [13141,54] [13855,55] [13259,57] [12952,62] [12920,66] [13151,67] [13293,70]

Sehingga *grams* yang digunakan:

eknikin ikinform formati atikaad tikaada adalahs dalahsa alahsal lahsala ahsalah lahsatu hsatuju tujurus urusany rusanya anyangt ngterda terdapa apatdif patdifa ifakult akultas kultass ultassa insdant antekno nteknol

13987 **13236** 13646 13707 **13448** 13565 13531 13738 **13501** 14053 13995
13495 **13138** 13957 13174 **13005** **12418** **12535** **12751** **12810** 13301 **12882**
13463 13711 **12818** 13337 **12956** 13613 14028 **13453** 14605 14459 **14052**
14652 14429 14380 **13886** **13149** 14000 14031 **13328** 13833 **13190** **13424**
14537 14072 14033 **13589** 13854 **12961**

Sedangkan untuk Kalimat 2:

[13236, 1] [13448, 4] [13501, 8] [13495,11] [13138, 12] [13005, 15] [12418, 13] [12535, 17] [12751, 18] [12810,19] [12882, 20] [12818, 21] [12956, 25] [13453,27] [14052,30] [13886,34] [13149,35] [13328,38] [13190,40] [13424,41] [13589,45] [12961,47]

Sehingga *grams* yang digunakan:

eknikin ikinform formati atikaad tikaada adalahs dalahsa alahsal lahsala ahsalah lahsatu hsatuju tujurus rusante usanter nterfav erfavor rfavori oritdif itdifas

Untuk menghitung kemiripan antar dokumen digunakan persamaan *jaccard coefficient*. Dokumen *fingerpritnt* 1 dan 2 adalah:

D1 = [13236] [13448] [13501] [13495] [13138] [13005] [12418] [12535] [12751] [12810] [12882] [12818] [12956] [13453] [14052] [13903] [13202] [13036] [13246] [12841] [12984] [12938] [13041] [13141] [13855] [13259] [12952] [12920] [13151] [13293]

D2 = [13236] [13448] [13501] [13495] [13138] [13005] [12418] [12535] [12751] [12810] [12882] [12818] [12956] [13453] [14052] [13886] [13149] [13328] [13190] [13424] [13589] [12961]

$$\text{Similaritas}(d_i,d_j) = \frac{|W(di) \cap W(dj)|}{|W(di) \cup W(dj)|} = \frac{|[13236][13448][13501][13495][13138][13005][12418][12535][12751][12810][12882][12818][12956][13453][14052]|}{37}$$

$$= \frac{15}{37} = 0.40 \;\; 40\%$$

2.8. Penerapan konsep *biword* pada Algoritma *winnowing*

Biword merupakan suatu teknik *matching* pada proses tokenisasi atau pemisahan kata (*fingerprint*) pada teks dokumen. Konsep *biword* itu sendiri menyerap teknik *phrase-based.* Pada tahap awal akan dilakukan konversi setiap dokumen untuk satu set *bigram* (dua kata) (Chow Kok Kent,2010).

Misalnya terdapat dua kalimat seperti berikut:

Kalimat 1: “Teknik Informatika adalah salah satu jurusan yang terdapat di Fakultas Sains dan Teknologi”,

Kalimat 2: “Informatika adalah salah satu jurusan terfavorit di FASTE “

Frasa yang terbentuk dari masing-masing kalimat adalah:

Frasa untuk kalimat 1:

```
teknik informatika
informatika adalah
adalah salah
salah satu
satu jurusan
```

```
jurusan yang
yang terdapat
terdapat di
di fakultas
fakultas sains
sains dan
dan teknologi
teknologi
```

Frasa yang terbentuk dari kalimat 2:

```
teknik informatika
informatika adalah
adalah salah
salah satu
satu jurusan
jurusan terfavorit
terfavorit di
di faste
```

Setelah membentuk kalimat ke dalam beberapa frasa (token *biword)* seperti proses di atas, kemudian dilakukan proses enkripsi frasa tersebut kedalam MD5. Proses ini dilakukan bertujuan agar setiap frasa yang memiliki panjang karakter yang berbeda-beda menjadi karakter yang memiliki panjang yang sama menggunakan fungsi MD5 dengan panjang 32 karakter.

Pada penelitian sebelumnya telah dilakukan pengujian panjang karakter yang terbaik, yaitu 32 karakter. Dibandingkan dengan SHA yang memiliki panjang 40 karakter , MD5 memiliki panjang karakter yang mendekati panjang karakter yang terbaik. Panjang karakter dapat mempengaruhi waktu proses dalam melakukan perhitungan nilai *hash.*

Berikut adalah hasil *eksport* token *biword* menjadi nilai MD5 menggunakan fungsi MD5:

Untuk Kalimat 1:

```
[0] => 5495ef670ce7ce89fae9677affba8b26
[1] => 7994e4e6ae098e4c6b3253c807369af9
[2] => d95e13f7daf667ba0fdbcab313b597b8
[3] => 085ed18c8e8b6377b4f0d6d685b7f323
[4] => 879b3588c428efb0be2d4fc0aca2737c
[5] => f354c10a4cd7f06c4792174a0f5e566f
[6] => f957126f6e2c13fc92f993f784413a74
[7] => c8bc10cfdb7fbee1afecc11fc7f7247d
[8] => 55248a5faa125bfccd9403da93de2486
[9] => af4fb3d9679b512d32c0524df6950050
[10] => bc25a55dfdbb252bb900ef66d2dc68bb
[11] => 5a4c9eaaab53a3a376526486d8a427e1
```

Untuk Kalimat 2:

```
[0] => 5495ef670ce7ce89fae9677affba8b26
[1] => 7994e4e6ae098e4c6b3253c807369af9
[2] => d95e13f7daf667ba0fdbcab313b597b8
[3] => 085ed18c8e8b6377b4f0d6d685b7f323
[4] => 879b3588c428efb0be2d4fc0aca2737c
[5] => 0dad2d77e81145f7f972f9256553081c
[6] => bef6f1adb539c8c4696f509f2e08cd5b
[7] => e4567e7a48b90cf247568be4c8e5cdca
```

Setelah terbentuk MD5, barulah dilakukan proses *hashing* untuk tiap-tiap token MD5 yang dibentuk. Proses *hashing* tersebut dilakukan dengan persamaan *rolling hash.* Berikut Perhitungan nilai *hash* menggunakan persamaan 2.1, dimana b=2 dan k=32:

Perhitungan untuk Kalimat 1:

$$H_{(\texttt{5495ef670ce7ce89fae9677affba8b26})} = \text{ascii}(5) * 2^{(31)} + \text{ascii}(4) * 2^{(30)} + \text{ascii}(9) * 2^{(29)} + \text{ascii}(5) * 2^{(28)} + \text{ascii}(e) * 2^{(27)} + \text{ascii}(f) * 2^{(26)} + \text{ascii}(6) * 2^{(25)} + \text{ascii}(7) * 2^{(24)} + \text{ascii}(0) * 2^{(23)} + \text{ascii}(c) * 2^{(22)} + \text{ascii}(e) * 2^{(21)} + \text{ascii}(7) * 2^{(20)} + \text{ascii}(c) * 2^{(19)} + \text{ascii}(e) * 2^{(18)} + \text{ascii}(8) *$$

$2^{(17)}$ + ascii(9) * $2^{(16)}$ + ascii(f) * $2^{(15)}$ + ascii(a) * $2^{(14)}$ + ascii(e) * $2^{(13)}$+ ascii(9) * $2^{(12)}$ + ascii(6) * $2^{(11)}$ + ascii(7) * $2^{(10)}$ + ascii(7) * $2^{(9)}$ + ascii(a) * $2^{(8)}$ + ascii(f) * $2^{(7)}$ + ascii(f) * $2^{(6)}$ + ascii(b) * $2^{(5)}$ + ascii(a) * $2^{(4)}$ + ascii(8) * $2^{(3)}$ + ascii(b) * $2^{(2)}$ + ascii(2) * $2^{(1)}$ + ascii(6) * $2^{(0)}$

= 238798777778

$H_{(7994e4e6ae098e4c6b3253c807369af9)}$ =

(238798777778 - ascii(5) * $2^{(31)}$) * 2 + ascii(9) * $2^{(0)}$

= 246686918097

$H_{(d95e13f7daf667ba0fdbcab313b597b8)}$ =

(246686918097 - ascii(7) * $2^{(31)}$) * 2 + ascii(8) * $2^{(0)}$

= 347445324656

Sehingga diperoleh nilai *hash* untuk kalimat 1:

[0] => 238798777778

[1] => 246686918097

[2] => 347445324656

[3] => 240196514691

[4] => 250757466581

[5] => 337232817858

[6] => 338068249934

[7] => 368322856610

[8] => 230497424406

[9] => 394976356810

[10] => 379498002150

[11] => 293245761735

Nilai *hash* kalimat 2:

[0] => 238798777778

[1] => 246686918097

[2] => 347445324656

[3] => 240196514691

[4] => 250757466581

[5] => 306863088181

[6] => 411366580532

[7] => 334320655167

Selanjutnya nilai-nilai *hash* yang telah diperoleh tersebut dibagi menjadi beberapa *window* dengan ukuran *w*.

Berikut adalah proses pembentukan *window* nilai *hash* dengan ukuran *w=4*:

Pembentukan *window* dokumen 1:

```
[238798777778 246686918097 347445324656 240196514691]
[246686918097 347445324656 240196514691 250757466581]
[347445324656 240196514691 250757466581 337232817858]
[240196514691 250757466581 337232817858 338068249934]
[250757466581 337232817858 338068249934 368322856610]
[337232817858 338068249934 368322856610 230497424406]
[338068249934 368322856610 230497424406 394976356810]
[368322856610 230497424406 394976356810 379498002150]
[230497424406 394976356810 379498002150 293245761735]
```

Pembentukan *window* Dokumen 2:

```
[238798777778 246686918097 347445324656 240196514691]
[246686918097 347445324656 240196514691 250757466581]
[347445324656 240196514691 250757466581 306863088181]
[240196514691 250757466581 306863088181 411366580532]
[250757466581 306863088181 411366580532 334320655167]
```

Proses selanjutnya adalah memilih nilai *hash* terkecil untuk dijadikan sebagai *fingerprint* dokumen. Pengambilan nilai *hash* terkecil dimulai nilai *hash* yang paling kanan. Jika ditemukan, maka akan dilanjutkan pada *index* berikutnya.

Berikut adalah nilai *hash* terkecil yang akan dijadikan sebagai *fingerprint:*

Fingerprint kalimat 1:

`[238798777778,1][240196514691,3][250757466581,4][230497424406,8]`

Fingerprint kalimat 2:

`[238798777778,0][240196514691,3][250757466581,4]`

Setelah diperoleh nilai *fingerprint* masing-masing dokumen, selanjutnya akan dilakukan perhitungan *similarity* menggunakan persamaan *jaccard coefficient:*

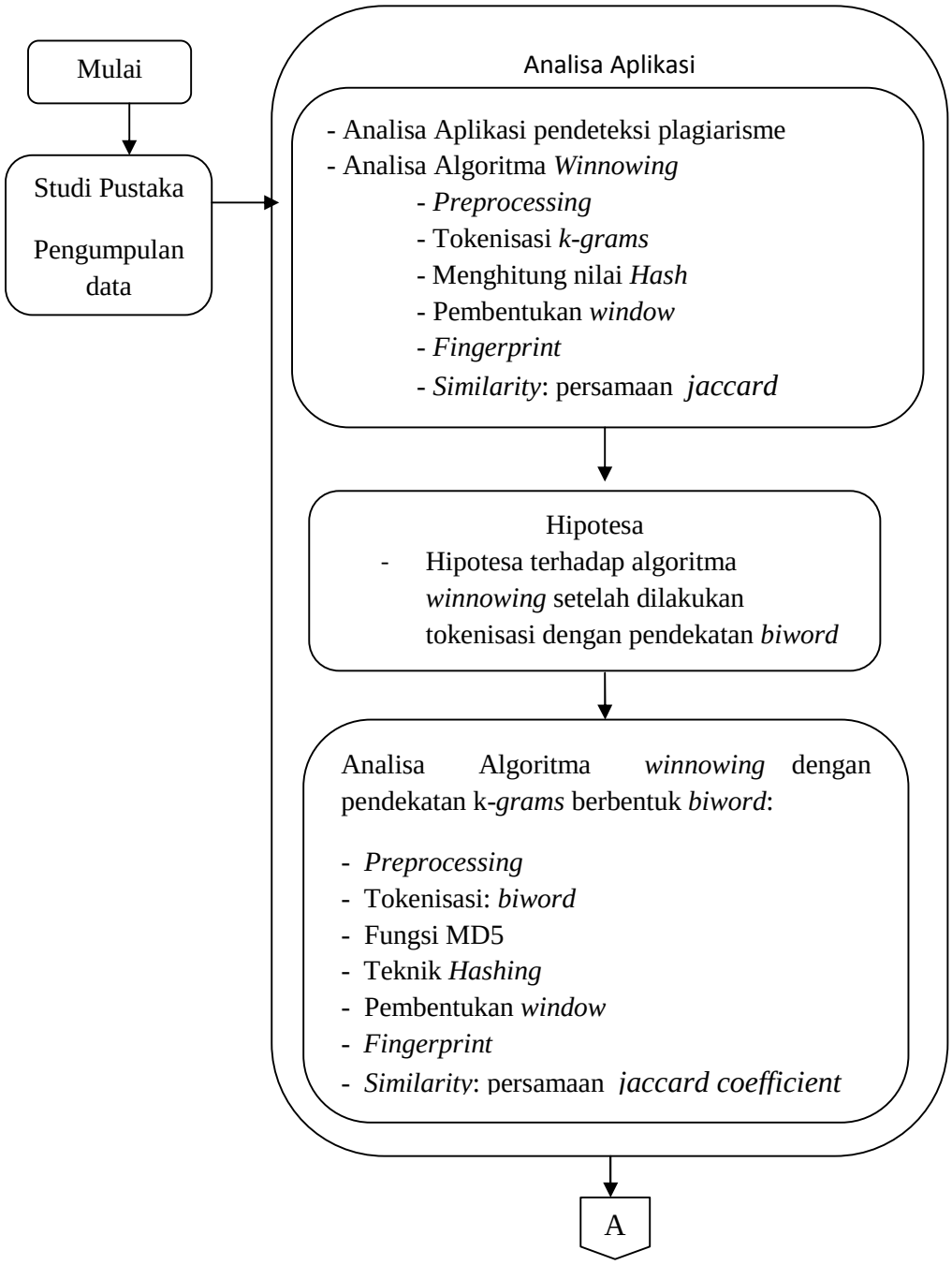
$$\text{Similaritas}(d_i,d_j) = \frac{|W(d1)\cap W(d2)|}{|W(d1)\cup W(d2)|} \text{ x } 100\%$$

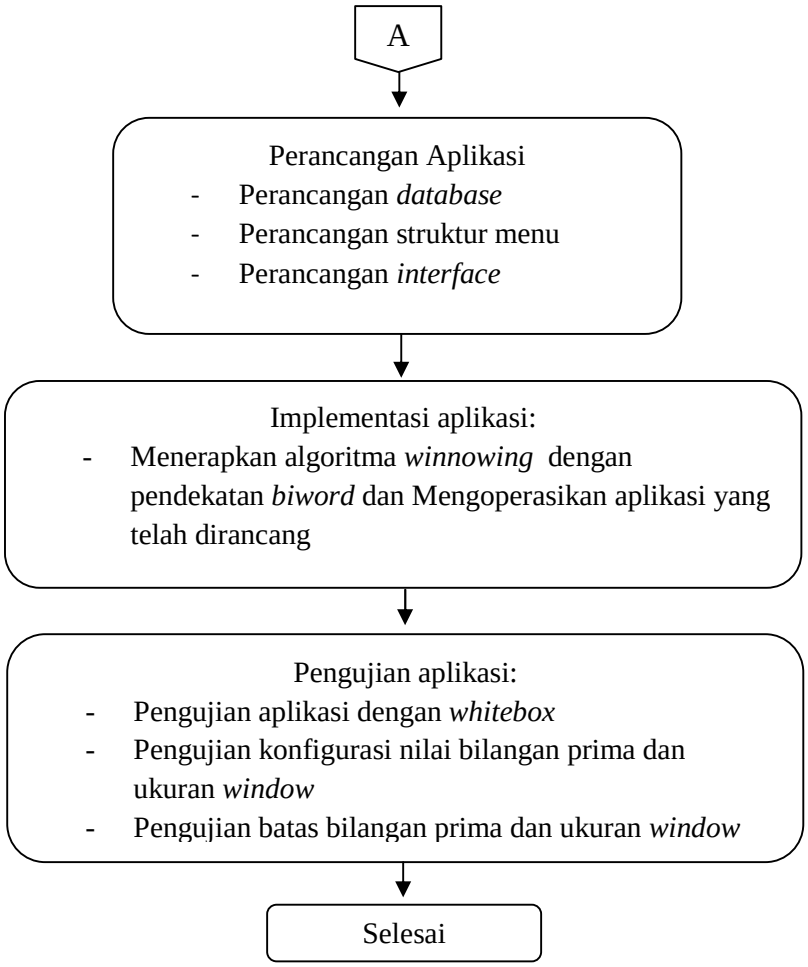
$$= \frac{|[238798777778][240196514691][250757466581]|}{[238798777778][240196514691][250757466581][230497424406]|}$$

$$= \frac{3}{4} \text{ x } 100\% = 70\%$$

BAB III
METODOLOGI PENELITIAN

Pada penelitian tugas akhir ini ada beberapa tahapan penelitian yang akan dilakukan seperti yang terlihat pada gambar 3.1:





Gambar 3.1 Tahapan Penelitian

3.1. Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data dengan cara mencari referensi-referensi terkait yang dibutuhkan untuk penelitian. Referensi tersebut dapat berupa buku-buku, jurnal-jurnal, tulisan penelitian dan juga artikel-artikel dari internet yang memiliki kaitan dengan kasus yang sedang dilakukan dalam penelitian.

3.2. Analisa Aplikasi

Setelah tahap pengumpulan data, langkah selanjutnya adalah menganalisa hal-hal yang berhubungan dengan aplikasi yang akan dibangun. Analisa aplikasi

berkaitan dengan mengidentifikasi kebutuhan dalam suatu penelitian. Beberapa tahapan dalam analisa yang akan dilakukan, adalah:

1. Analisa Aplikasi pendeteksi plagiarisme

   Memahami konsep aplikasi pendeteksi plagiarisme dan menganalisa data atau kebutuhan-kebutuhan yang diperlukan dalam membangun aplikasi
2. Analisa algoritma *winnowing*

   Pada tahap ini dilakukan pengidentifikasian terhadap syarat-syarat algoritma dan langkah-langkah algoritma *winnowing.*

   Adapun langkah-langkah algoritma *winnowing* adalah:

   a. *Preprocessing*
   b. Perhitungan nilai *hash*
   c. Pembentukan beberapa *window* dengan panjang *w*
   d. Memilih *fingerprint*
3. Jawaban sementara (hipotesa)

   Tahap hipotesa merupakan tahap memberikan jawaban sementara terhadap algoritma *winnowing* dengan pendekatan *biword* sebelum melakukan implementasi dan pengujian terhadap aplikasi.

   Diberikan sebuah hipotesa pada konfigurasi bilangan prima dan ukuran *window*. “Semakin kecil nilai bilangan prima, maka nilai *similarity* akan meningkat”, disebabkan karena kalkulasi dengan nilai prima terkecil dapat menghasilkan nilai *fingerprint* terkecil juga. Sementara itu, “semakin tinggi ukuran *window* yang digunakan akan meningkatkan hasil *similarity”*, disebabkan karena penerapan MD5 yang memiliki jumlah karakter yang panjang, yaitu 32 karakter.
4. Analisa algoritma *winnowing* dengan pendekatan *k-grams* berbentuk *biword.*

   Beberapa tahapan yang akan dilakukan adalah:

   a. *Preprocessing*.

Tahap ini merupakan proses yang pertama kali dilakukan setelah memasukkan dokumen yang akan diuji. *Preprocessing* yang dilakukan yaitu penghilangan karakter yang tidak relevan seperti mengubah huruf besar menjadi kecil.

b. Melakukan Tokenisasi

Setelah dilakukan proses *preprocessing*, selanjutnya akan dilakukan proses tokenisasi yaitu membagi dokumen menjadi token berbentuk *biword*.

c. Eksport menjadi nilai MD5

Setiap Nilai-nilai token *biword* yang telah dibentuk akan dieksport menjadi nilai MD5 menggunakan fungsi MD5. Hal ini bertujuan agar setiap token *biword* mempunyai panjang karakter yang sama.

d. Menghitung nilai *hash.*

Setelah nilai MD5 diperoleh dari masing-masing token *biword*, selanjutnya akan dihitung nilai *hash* masing-masing token menggunakan persamaan *Rolling Hash*.

e. Pembentukan *window*.

Nilai-nilai *hash* yang diperoleh akan dibagi ke dalam beberapa *window* dengan ukuran *w*. Ukuran *window* tersebut ditentukan oleh pengguna aplikasi.

f. Memilih *fingerprint*

Setelah pembagian nilai *hash* menjadi beberapa *window*, proses selanjutnya adalah memilih nilai *hash* terkecil dari tiap-tiap *window*. Nilai *hash* ini akan dijadikan sebagai *fingerprint* dokumen.

g. Menghitung *similarity.*

Setelah didapatkan token *fingerprint* dari masing-masing dokumen, selanjutnya akan dilakukan proses perhitungan *similarity* menggunakan persamaan *Jaccard coefficient.*

3.3. Perancangan Aplikasi

Pada tahap ini akan dilakukan perancangan aplikasi yang telah dianalisa sebelumnya. Tahap ini bertujuan untuk memberikan gambaran kepada pengguna terhadap aplikasi pendeteksi plagiarisme yang akan dibangun. Tahapan yang dilakukan adalah:

a. Perancangan Struktur Menu

   Merancang menu-menu pada aplikasi yang memiliki fungsi masing-masing sesuai tujuan yang akan dicapai.

   Menu-menu yang terdapat dalam aplikasi adalah:

   - Menu beranda: halaman yang memperkenalkan aplikasi kepada pengguna
   - Menu deteksi plagiat: halaman yang digunakan untuk memulai proses deteksi penjiplakan dokumen.
   - Menu hasil pengujian: halaman yang menampilkan hasil dari proses-proses deteksi penjiplakan yang telah dilakukan.
   - Menu bantuan: halaman yang memberikan informasi atau petunjuk cara penggunaan aplikasi kepada pengguna.

b. Perancangan *interface* aplikasi.

   Merancang atau mendesain tampilan antar muka aplikasi dengan pengguna. *Interface* yang akan dibangun adalah *interface input* dan output. Dengan demikian akan terlihat *interface* dari sistem dan dapat memberikan gambaran terhadap sistem yang akan dibangun,

3.4. Implementasi

Setelah dilakukan perancangan aplikasi, maka akan dilakukan tahap implementasi. Implementasi merupakan tahap dimana aplikasi siap untuk di operasikan sesuai dari hasil analisis dan perancangan yang telah dilakukan, sehingga akan diketahui apakah aplikasi yang dirancang benar-benar dapat menghasilkan tujuan yang ingin dicapai.

Implementasi pengembangan aplikasi deteksi penjiplakan dokumen ini akan dibangun pada spesifikasi *hardware* dan *software* sebagai berikut:

1. Perangkat keras

   Processor : *AMD C-60 With Radeon @ 1.0 GHz*

   Memori (RAM) : 2,00 GB

   Harddisk : 320 GB

2. Perangkat Lunak

   Sistem operasi : *Windows 7 Ultimate 32-bit OS*

   Bahasa pemrograman : *Hypertext Preprocessor (PHP)*

   Tool : *Notepad++*

3.5. Pengujian

Setelah tahap implementasi, selanjutnya akan dilakukan uji coba terhadap aplikasi yang telah dibangun. Dengan demikian dapat diketahui tingkat keberhasilan aplikasi, apakah telah mencapai tujuan yang diharapkan.

Pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian aplikasi dengan *whitebox*. Pengujian *whitebox* dilakukan dengan menguji proses-proses yang terdapat dalam algoritma, apakah telah diterapkan dengan benar ke dalam aplikasi. Sedangkan untuk pengujian *whitebox* dilakukan untuk mengetahui keberhasilan algoritma apakah telah sesuai dengan tujuan yang akan dicapai.
2. Pengujian konfigurasi bilangan prima dan ukuran *window* yang digunakan dalam menjalankan proses deteksi penjiplakan. Hal ini bertujuan untuk mendapatkan bilangan prima dan ukuran *window* terbaik dalam mendeteksi penjiplakan sehingga memperoleh hasil yang maksimal dengan nilai *similarity* yang tinggi.

3.6. Kesimpulan dan Saran

Pada tahapan ini berisi tentang kesimpulan yang dapat ditarik dari penelitian yang telah penulis lakukan, yang dimulai dari tahapan analisa hingga

tahapan pengujian aplikasi. Selanjutnya pada bagian saran berisi saran-saran yang yang penulis berikan untuk mengembangkan aplikasi yang penulis bangun dalam penelitian tugas akhir ini sehingga ke depannya dapat diteruskan dan menjadi lebih baik.

BAB IV
ANALISA DAN PERANCANGAN

4.1 Analisa Aplikasi Pendeteksi Plagiarisme

Pada tahap ini berisi tentang gambaran secara garis besar terhadap proses yang dilakukan aplikasi deteksi penjiplakan dokumen. Pada umumnya setiap algoritma pendeteksi penjiplakan memiliki tahapan-tahapan yang sama dalam pemrosesan dokumen teks. Tahapan yang dilakukan adalah adanya tahapan *input*, proses, dan *output*. Pada tahap *input* dilakukan pemasukan dokumen teks yang akan diuji. Selanjutnya dokumen akan diproses berdasarkan tahapan yang dimiliki oleh algoritma pendeteksi plagiarisme. Tahapan tersebut diantaranya adalah tahap *preprocessing* dan tokenisasi. Proses akan berlanjut dengan perhitungan tingkat *similarity* dokumen. Setelah proses utama dilakukan, selanjutnya aplikasi akan menghasilkan *output* informasi dokumen berupa hasil *similarity* dokumen dan kalimat yang telah diplagiasi.

4.2 Analisa Algoritma *Winnowing*

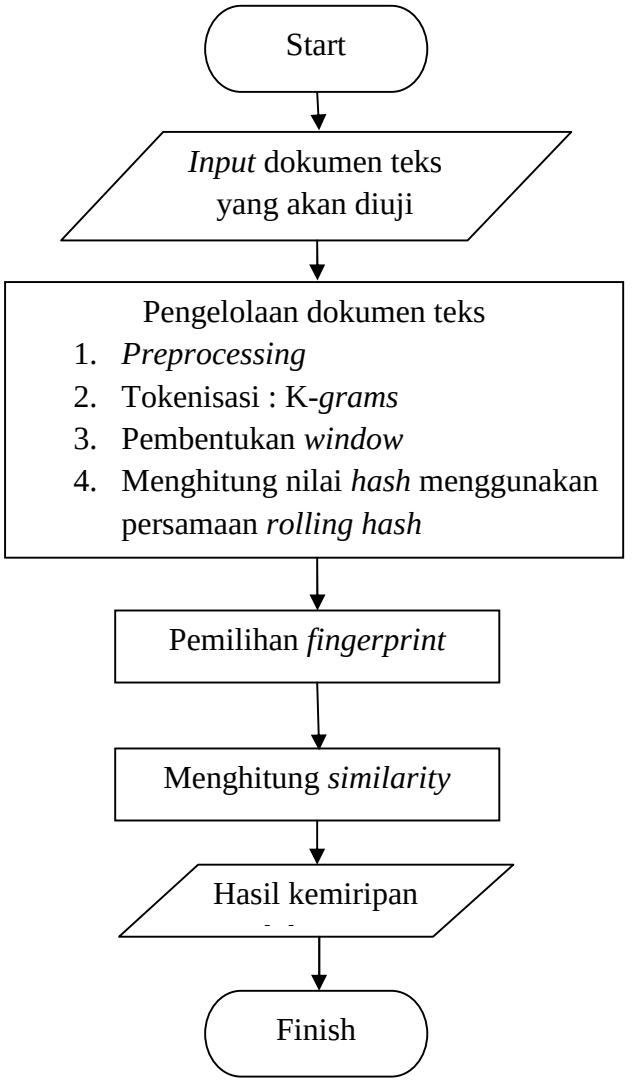
Pada tahap ini, analisa yang dilakukan adalah analisa terhadap algoritma *winnowing* asli yang belum dilakukan penambahan konsep *biword*. Prinsip kerja dari algoritma *winnowing* itu sendiri adalah menggunakan teknik *hashing*, yaitu mengkonversi setiap *string* menjadi bilangan dan kemudian bilangan tersebut akan disimpan sebagai *fingerprint*. Untuk memilih *fingerprint* dari hasil teknik *hashing*, dilakukan pembagian *fingerprint* ke dalam *window* tertentu dengan ukuran *w*.

Dari setiap *window* yang telah dibentuk, dipilih nilai *hash* yang paling minimum atau yang paling kecil. Jika terdapat nilai minimum lebih dari satu nilai, maka pilih dari *window* sebelah kanan. Kemudian simpan hasil *hash* yang telah dipilih yang merupakan *fingerprint* dokumen.

Secara umum prinsip kerja dari metode dokumen *fingerprinting* adalah sebagai berikut:

1. Hilangkan tanda baca dan spasi.
2. Sebelum melakukan fungsi *hash*, terlebih dahulu lakukan pembagian teks dokumen menjadi *k-gram*, dimana k merupakan parameter yang dipilih pengguna.
3. Menghitung nilai *hash* untuk setiap *k-grams*.
4. Memilih beberapa hasil *hash* menjadi dokumen *fingerprinting*.

Untuk lebih jelasnya, dapat dilihat dari *flowchart* berikut:



Gambar 4.1 *Flowchart* algoritma *winnowing*

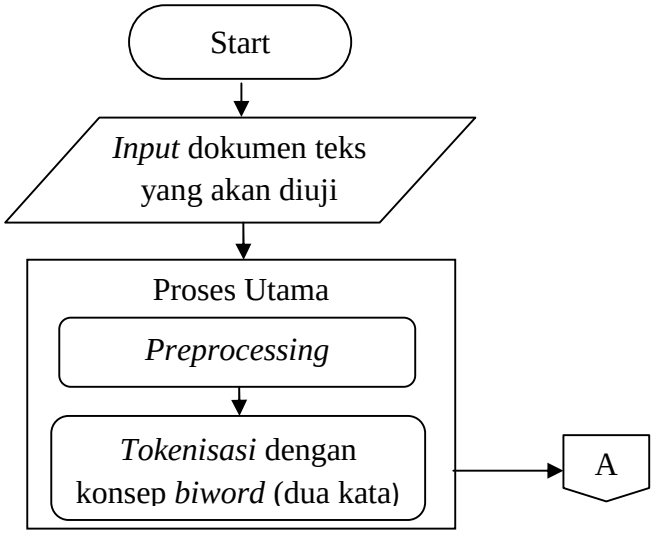
4.3 Algoritma *Winnowing* dengan pendekatan *biword*

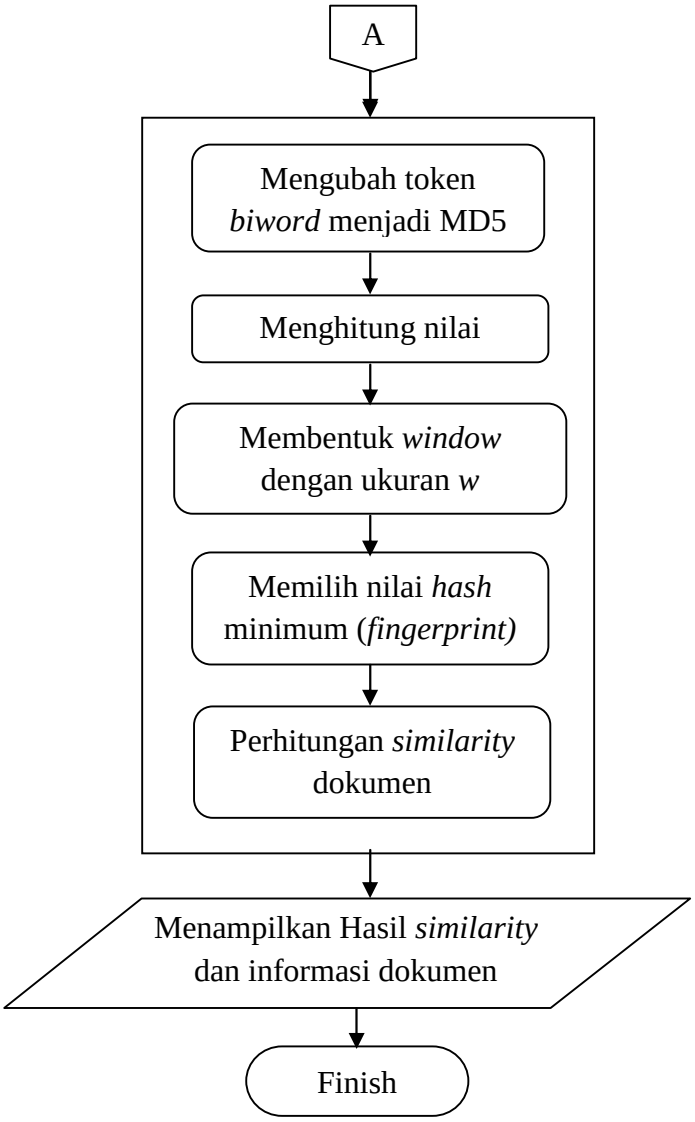
Pada penelitian ini dilakukan pengembangan algoritma *winnowing* dalam mendeteksi penjiplakan dokumen teks, yaitu dengan penerapan konsep *biword.* Algoritma *winnowing* yang biasanya menggunakan teknik *character-based* dalam proses tokenisasi dokumen, sekarang akan dilakukan menggunakan teknik *phrase-based.* Dengan demikian, akan terbentuk banyak frasa atau token *biword* dari masing-masing dokumen teks untuk perhitungan *similarity*. Konsep *biword* ini merupakan pendekatan *k-grams* untuk membentuk *substring* sepanjang k karakter atau kata. Pendekatan k-grams inilah yang digunakan dalam membentuk token *biword.*

Secara garis besar ada beberapa tahap dalam melakukan pendeteksian plagiarisme dokumen menggunakan pendekatan *biword winnowing*, diantaranya:

1. Melakukan pembersihan teks.
2. Melakukan pemotongan teks
3. Menghitung nilai *hash*
4. Membentuk *window* dengan ukuran *w*
5. Mendapatkan nilai *fingerprint*
6. Menghitung kemiripan dokumen dari nilai *fingerprint* yang diperoleh.

Berikut adalah *flowchart* proses-proses yang dilakukan pada algoritma *biword winnowing* dalam mendeteksi penjiplakan dokumen teks:





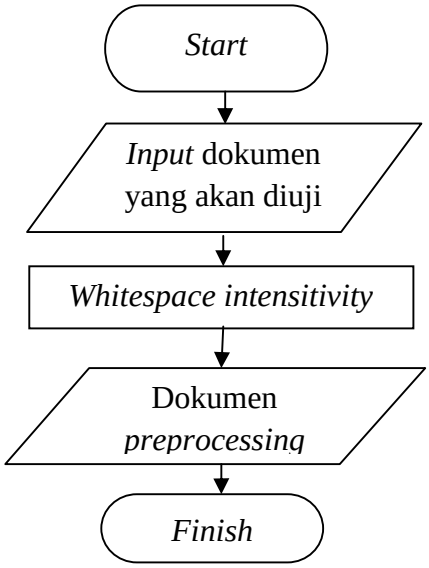
Gambar 4.2 *Flowchart* algoritma *winnowing* dengan pendekatan *biword*

Pada gambar 4.2 diatas dapat dilihat proses deteksi plagiarisme dokumen dengan menerapkan pendekatan *biword* (dua kata) pada algoritma *winnowing*. Proses-proses tersebut dapat dijelaskan sebagai berikut:

1. *Input* dokumen yang akan yang akan diuji kemiripannya pada aplikasi yang akan dibangun. Sehingga aplikasi akan memperoleh informasi dokumen yang akan diuji.
2. Dokumen yang telah dimasukkan akan diproses pada tahap *preprocessing*, yaitu menghilangkan karakter-karakter yang tidak relevan

seperti membuang tanda baca, mengubah huruf besar menjadi huruf kecil dan menghilangkan spasi.

Untuk lebih jelasnya, dapat dilihat pada *flowchart* berikut:

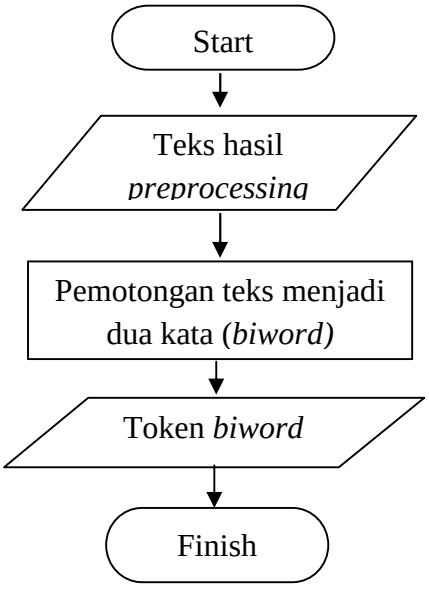


Gambar 4.3 *Flowchart* proses *preprocessing*

3. Tokenisasi dengan pendekatan *biword.*

Setelah memperoleh dokumen *preprocessing*, selanjutnya dilakukan proses tokenisasi kata menjadi *biword.*

Untuk lebih jelasnya dapat dilihat pada *flowchart* berikut:

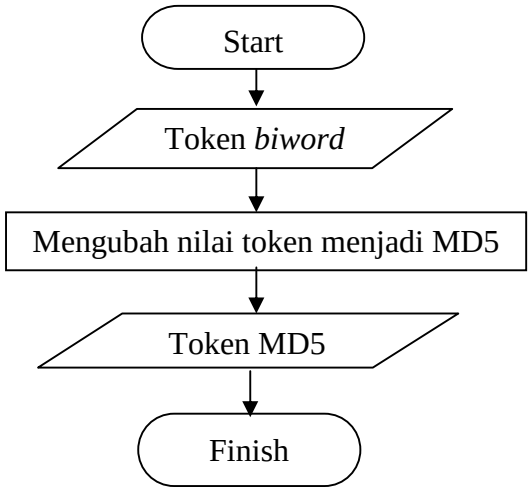


Gambar 4.4 *Flowchart* proses tokenisasi

4. Mengubah nilai token menjadi MD5

Setelah mendapatkan token kata *biword*, selanjutnya akan dilakukan konversi mengubah nilai token *biword* menjadi nilai MD5. Hal ini bertujuan agar token tersebut memiliki panjang karakter yang sama yaitu 32 karakter. Konversi token menjadi nilai MD5 dapat dilakukan dengan fungsi MD5.

Untuk lebih jelasnya dapat dilihat pada *flowchart* berikut:

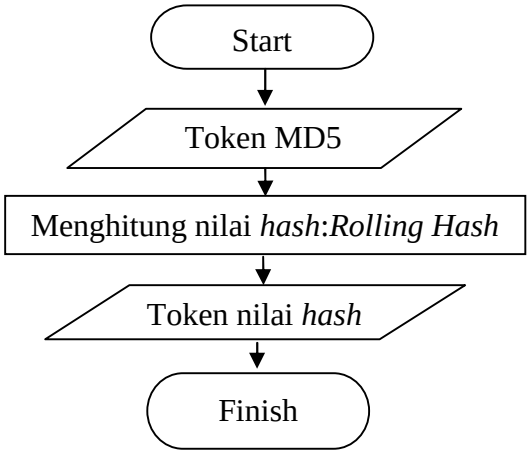


Gambar 4.5 *Flowchart* proses mendapatkan nilai MD5

5. Menghitung nilai *hash* masing-masing token.

Token-token yang telah diubah menjadi MD5, selanjutnya akan diproses menggunakan persamaan *rolling hash* untuk mendapatkan nilai *hash* dokumen. Nilai *hash* ini nantinya akan dijadikan *fingerprint* dokumen.

Untuk lebih jelasnya dapat dilihat pada *flowchart* berikut:

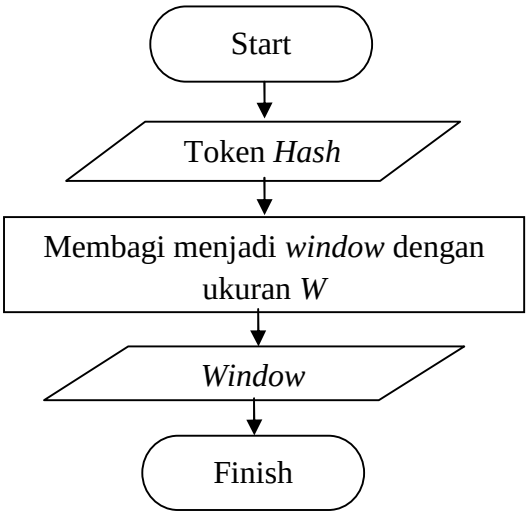


Gambar 4.6 *Flowchart* proses hitung nilai *hash*.

6. Membagi ke dalam beberapa *window*.

Token-token yang telah diperoleh, akan dibagi dalam beberapa *window* dengan ukuran *w*. Ukuran *window* ditentukan oleh pengguna aplikasi.

Berikut adalah *flowchart* pembentukan *window*:

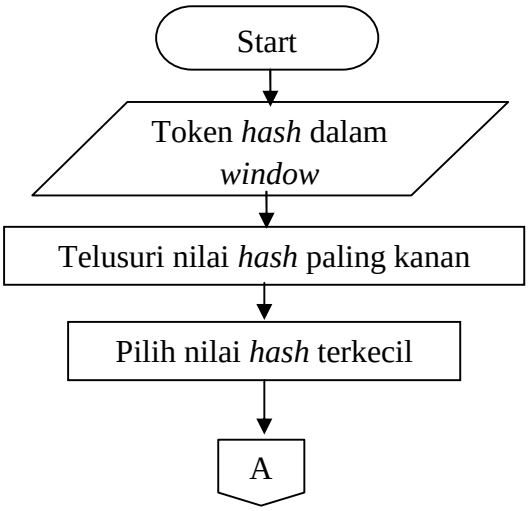


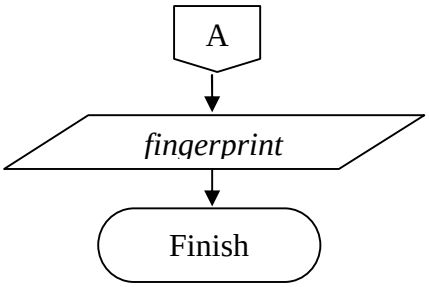
Gambar 4.7 *Flowchart* proses pembentukan *window*.

7. Mencari nilai *hash* minimum.

Dari nilai-nilai *hash* yang telah dibentuk menggunakan persamaan *rolling hash*, selanjutnya akan ditelusuri nilai-nilai *hash* terkecil untuk dijadikan *fingerprint* dokumen. Penelusuran nilai *hash* terkecil adalah dimulai dari nilai *hash* yang paling kanan dalam suatu *window*.

Berikut adalah *flowchart* pencarian nilai *hash* terkecil:



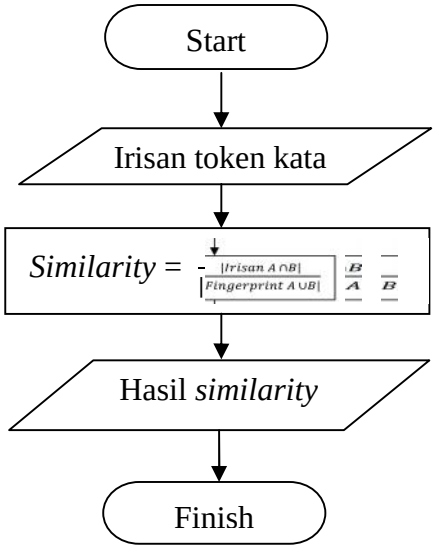


Gambar 4.8 *Flowchart* proses memilih *fingerprint.*

8. Perhitungan *similarity* dokumen

   Nilai *fingerprint* yang diperoleh akan digunakan untuk menghitung *similarity* dokumen. Proses perhitungan dilakukan menggunakan persamaan *jaccard coefficient.*

   Untuk lebih jelasnya dapat dilihat pada *flowchart* berikut:



Gambar 4.9 *Flowchart* proses hitung *similarity*

9. Selanjutnya akan diperoleh hasil dari proses utama berupa informasi dokumen yaitu nama dokumen, ukuran dokumen, waktu proses dan hasil *similarity* dokumen teks.

Untuk lebih jelasnya, berikut adalah contoh penerapan token *biword* pada algoritma *winnowing*:

Terdapat 2 buah kalimat sebagai berikut:
Kalimat 1: “Teknik Informatika adalah jurusan yang memiliki peminat tertinggi di Universitas Islam Negeri Sultan Syarif Kasim Riau”

Kalimat 2: “Teknik Informatika adalah jurusan terfavorit di Universitas Islam Negeri Sultan Syarif Kasim Riau”

Tahapan-tahapan yang dilakukan adalah:

1. *Whitespace Intensitivity* atau *preprocessing*, yaitu menghilangkan karakter yang tidak relevan seperti menghilangkan tanda baca dan mengubah huruf besar menjadi kecil. Sehingga terbentuk kalimat:
   Kalimat 1:
   teknik informatika adalah jurusan yang memiliki peminat tertinggi di universitas islam negeri sultan syarif kasim riau
   Kalimat 2:
   teknik informatika adalah jurusan terfavorit di universitas islam negeri sultan syarif kasim riau

2. Proses selanjutnya adalah tokenisasi, yaitu pemotongan kata berbentuk *biword*. Sehingga terbentuk token kata *biword*:

Tabel 4.1 Hasil token *biword*

Token kalimat 1	Token kalimat 2
[0] => teknik informatika	[0] => teknik informatika
[1] => informatika adalah	[1] => informatika adalah
[2] => adalah jurusan	[2] => adalah jurusan
[3] => jurusan yang	[3] => jurusan terfavorit
[4] => yang memiliki	[4] => terfavorit di
[5] => memiliki peminat	[5] => di universitas
[6] => peminat tertinggi	[6] => universitas islam
[7] => tertinggi di	[7] => islam negeri
[8] => di universitas	[8] => negeri sultan

[9] => universitas islam [10] => islam negeri [11] => negeri sultan [12] => sultan syarif [13] => syarif kasim [14] => kasim riau	[9] => sultan syarif [10] => syarif kasim [11] => kasim riau

3. Mengubah token *biword* menjadi nilai MD5.

Untuk mengubah sebuah token *biword* menjadi MD5, dapat dilakukan dengan fungsi MD5 yang terdapat dalam bahasa pemrograman PHP. Setelah masing-masing token *biword* diubah menjadi MD5, akan diperoleh hasil sebagai berikut:

```
[0]=> 5495ef670ce7ce89fae9677affba8b26
[1]=> 7994e4e6ae098e4c6b3253c807369af9
[2]=> bc4680f26e956ebd658eff70b4c43963
[3]=> f354c10a4cd7f06c4792174a0f5e566f
[4]=> 83d01da1003236447620e49d25ca7b06
[5]=> f83647b8c3d631b0436415cfd6dbf6b9
[6]=> bdd124e4a0ff7d22de054dbad8e28ef7
[7]=> 5c7e04c243f6c2cba376a9ac87939895
[8]=> 2215e9f11df2c3846cb644d06f398c74
[9]=> 5a0e390acfaf38a119d387a617ecb610
[10]=> 0fa24c32a6c0c0f6d7106095dbadc6df
[11]=> 0be258e165e21d11799332a479cfa9f6
[12]=> 1b0df4db1412519805a297a9c8628445
[13]=> b1c7a66f83f1791bb893061c4229c6d7
[14]=> a8f150992ae13b30ea64d52636ab5005
```

Kalimat 2:

```
[0]=> 5495ef670ce7ce89fae9677affba8b26
```

```
[1] => 7994e4e6ae098e4c6b3253c807369af9
[2] => bc4680f26e956ebd658eff70b4c43963
[3] => 0dad2d77e81145f7f972f9256553081c
[4] => bef6f1adb539c8c4696f509f2e08cd5b
[5] => 2215e9f11df2c3846cb644d06f398c74
[6] => 5a0e390acfaf38a119d387a617ecb610
[7] => 0fa24c32a6c0c0f6d7106095dbadc6df
[8] => 0be258e165e21d11799332a479cfa9f6
[9] => 1b0df4db1412519805a297a9c8628445
[10] => b1c7a66f83f1791bb893061c4229c6d7
[11] => a8f150992ae13b30ea64d52636ab5005
```

Setelah didapatkan nilai MD5 masing-masing token *biword* yang dibentuk, selanjutnya akan dihitung nilai *hash* menggunakan persamaan *Rolling Hash.* Nilai-nilai *hash* ini akan dipilih nantinya untuk dijadikan *fingerprint.*

Berikut adalah hasil perhitungan nilai *hash* masing-masing token *biword*:

Tabel 4.2. Nilai *hash* token *biword*

Token kalimat 1	Token kalimat 2
[0] => 4.9667213130 [1] => 5.1825664534 [2] => 8.6378083570 [3] => 7.9287077961 [4] => 5.3781971911 [5] => 7.9915518913 [6] => 8.9731000098 [7] => 5.9809251960 [8] => 4.6778057251 [9] => 5.8928789443 [10] => 5.9167894083 [11] => 5.8555161140 [12] => 5.7054682866 [13] => 7.9633002980 [14] => 8.0165130733	[0] => 4.9667213130 [1] => 5.1825664534 [2] => 8.6378083570 [3] => 5.9891910630 [4] => 9.0587209216 [5] => 4.6778057251 [6] => 5.8928789443 [7] => 5.9167894083 [8] => 5.8555161140 [9] => 5.7054682866 [10] => 7.9633002980 [11] => 8.0165130733

4. Pembentukan *window* dari nilai *hash* yang telah diperoleh.

Misalkan ukuran *window w* yang digunakan adalah 4, maka diperoleh hasil pembagian token *hash* sebagai berikut:

Kalimat 1:

[**4.9667213130** 5.1825664534 8.6378083570 7.9287077961]

[**5.1825664534** 8.6378083570 7.9287077961 5.3781971911]

[8.6378083570 7.9287077961 **5.3781971911** 7.9915518913]

[7.9287077961 5.3781971911 7.9915518913 8.9731000098]

[5.3781971911 7.9915518913 8.9731000098 5.9809251960]

[7.9915518913 8.9731000098 5.9809251960 **4.6778057251**]

[8.9731000098 5.9809251960 4.6778057251 5.8928789443]

[5.9809251960 4.6778057251 5.8928789443 5.9167894083]

[4.6778057251 5.8928789443 5.9167894083 5.8555161140]

[5.8928789443 5.9167894083 5.8555161140 **5.7054682866**]

[5.9167894083 5.8555161140 5.7054682866 7.9633002980]

[5.8555161140 5.7054682866 7.9633002980 8.0165130733]

Kalimat 2:

[**4.9667213130** 5.1825664534 8.6378083570 5.9891910630]

[**5.1825664534** 8.6378083570 5.9891910630 9.0587209216]

[8.6378083570 5.9891910630 9.0587209216 **4.6778057251**]

[5.9891910630 9.0587209216 4.6778057251 5.8928789443]

[9.0587209216 4.6778057251 5.8928789443 5.9167894083]

[4.6778057251 5.8928789443 5.9167894083 5.8555161140]

[5.8928789443 5.9167894083 5.8555161140 **5.7054682866**]

[5.9167894083 5.8555161140 5.7054682866 7.9633002980]

[5.8555161140 5.7054682866 7.9633002980 8.0165130733]

Maka diperoleh nilai *hash* minimum masing-masing dokumen:

Kalimat 1:

[4.9667213130, 0] [5.1825664534, 1] [5.3781971911, 4] [4.6778057251, 8] [5.7054682866, 12]

Kalimat 2:

[4.9667213130, 0] [5.1825664534, 1] [4.6778057251, 8] [5.7054682866, 12]

Nilai-nilai *hash* minimum (*fingerprint)* yang diperoleh berdasarkan posisi indeks nya, jika diubah kembali menjadi token *biword* akan terlihat frasa mana yang memiliki *fingerprint* yang sama antara kedua kalimat yang diuji.

Berikut ini adalah *biword* yang dianggap memiliki nilai *fingerprint* yang sama.

Tabel 4.3 Token *biword* dengan *fingerprint* yang sama

Kalimat 1	Kalimat 2
[0] => teknik informatika [1] => informatika adalah [4] => yang memiliki [8] => di universitas [12] => sultan syarif	[0] => teknik informatika [1] => informatika adalah [8] => di universitas [12] => sultan syarif

5. Proses selanjutnya adalah menghuitung *similarity*. Perhitungan *similarity* dapat dilakukan dari hasil token terurut yang diperoleh pada table 4.2 dan gabungan hasil token kata pada tabel 4.1.menggunakan persamaan 2.3

   Similaritas(d_i,d_j) = $\frac{|1(di) \cap 2(dj)|}{|1(di) \cup 2(dj)|} x\ 100\% = \frac{4}{5} = 80\%$

4.4 Perancangan Aplikasi

Pada tahap ini akan dibahas tentang perancangan aplikasi pendeteksi penjiplakan berdasarkan tahapan analisa yang telah dilakukan sebelumnya. Adapun perancangan yang akan dibuat adalah perancangan basis data, perancangan struktur menu dan perancangan *interface.*

4.4.1 Perancangan Database

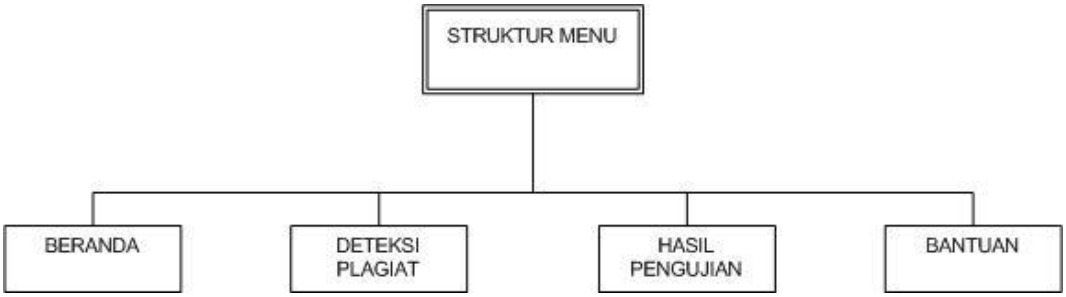
Perancangan database yang dibangun pada aplikasi bertujuan untuk menyimpan hasil proses deteksi penjiplakan dokumen. *Conceptual* data model digunakan untuk mengetahui tipe-tipe data yang digunakan dalam database aplikasi pendeteksi plagiarisme dokumen. Berikut adalah *Conceptual* data model yang dirancang pada aplikasi ini.

Tabel 4.4 *Conceptual* Data Model Tabel pengujian

Field	Type	Null
Id	*Int (11)*	No
Nama1	*Varchar(50)*	No
Ukuran1	*Int(10)*	No
Nama2	*Varchar(50)*	No
Ukuran2	*Int(10)*	No
Bil_prima	*Int(8)*	No
Ukuran_*window*	*Int(8)*	No
Waktu	*Decimal(20,4)*	No
Similarity	*Decimal(10,2)*	No

4.4.2 Perancangan Struktur Menu

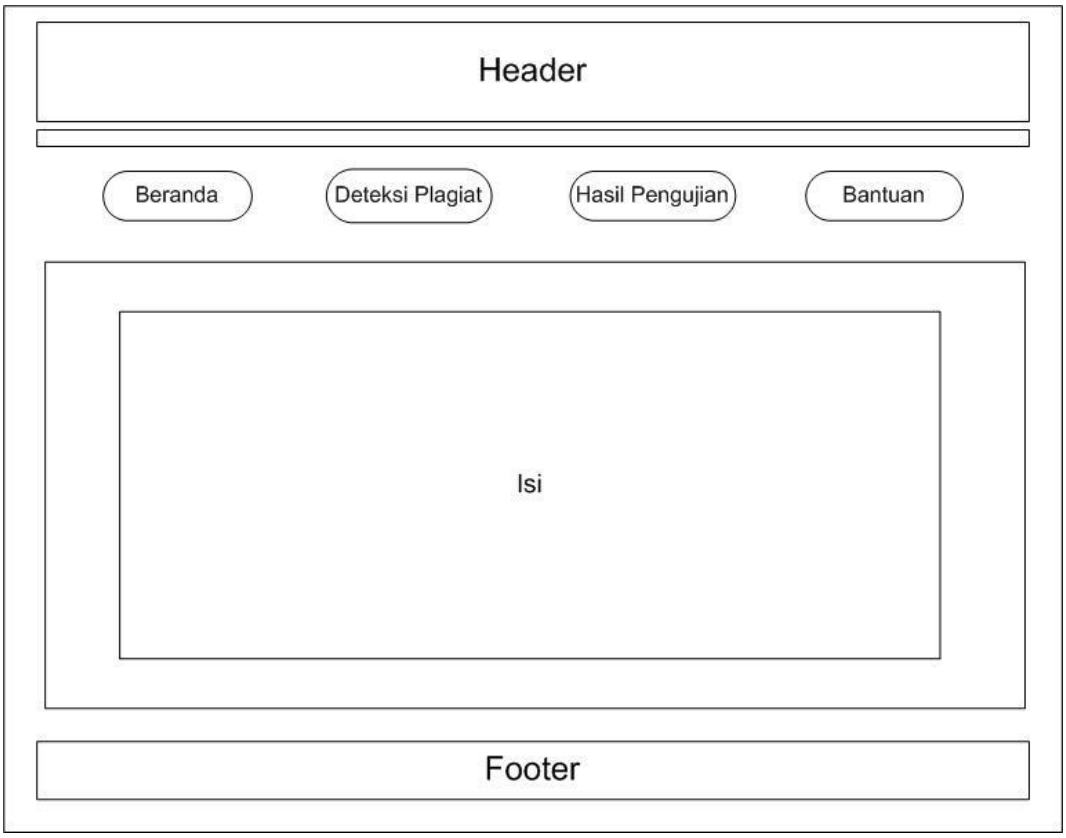
Perancangan struktur menu adalah tahap merancang menu-menu yang dapat digunakan pengguna untuk menjalankan aplikasi, sehingga dapat memudahkan pengguna dalam memilih proses yang akan dijalankannya. Untuk lebih jelasnya dapat dilihat pada gambar strktur menu berikut:



Gambar 4.10 Rancangan struktur menu

4.4.3 Perancangan *Interface*

Interface adalah sarana pengembangan sistem yang digunakan untuk membuat komunikasi lebih mudah, konsisten antara aplikasi dengan pemakaiannya. Penekanan *interface* meliputi tampilan yang baik dan mudah dipahami. Berikut adalah rancangan *interface* yang akan dibangun:

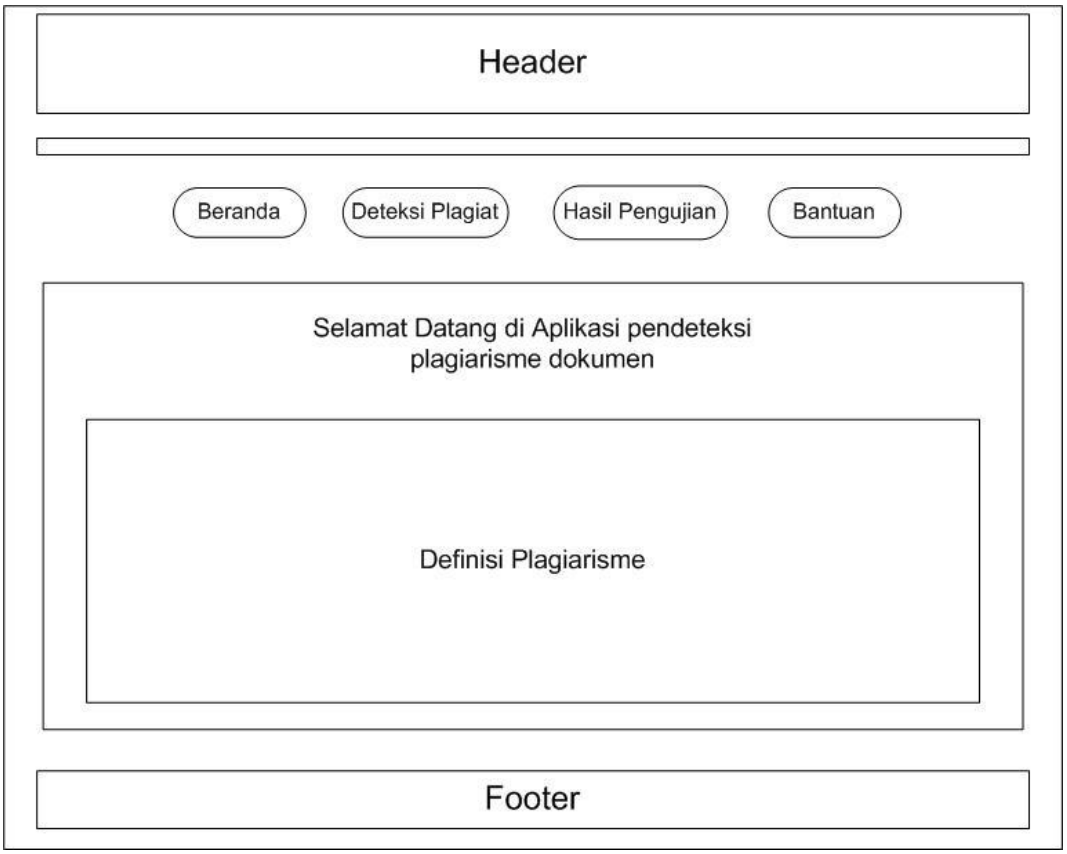


Gambar 4.11 Rancangan *interface*

4.4.3.1 Rancangan menu Halaman utama (beranda)

Menu home adalah halaman utama yang ditampilkan aplikasi kepada pengguna. Halaman ini berisi tentang definisi atau sekilas penjelasan tentang plagiarisme. Halaman ini juga mendeskripsikan tujuan dari aplikasi pendeteksi penjiplakan dokumen.

Berikut adalah rancangan menu halaman utama:

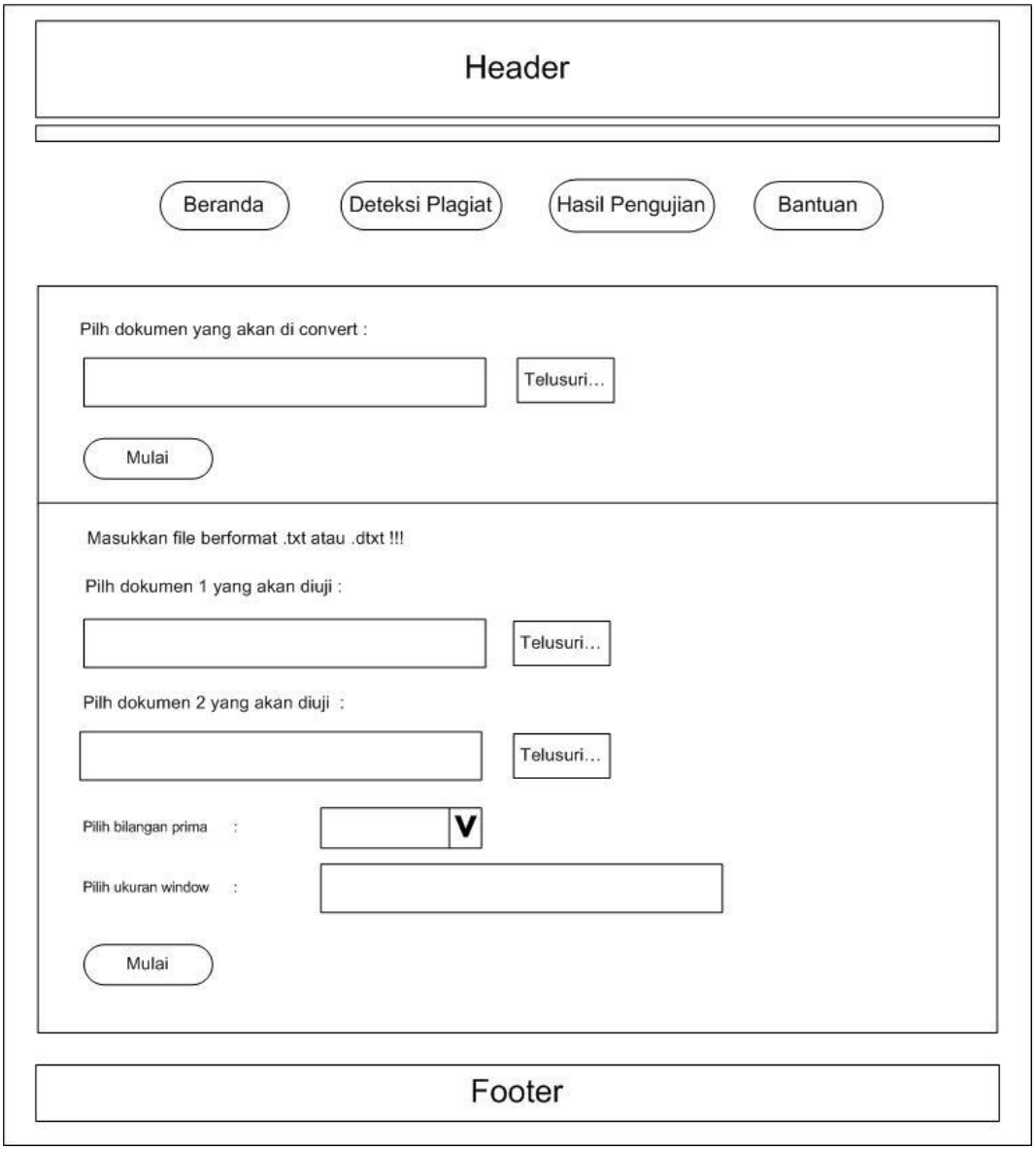


Gambar 4.12 *Interface* Halaman utama

4.4.3.2 Rancangan menu deteksi plagiarisme

Halaman ini digunakan oleh pengguna untuk melakukan proses deteksi plagiarisme dokumen. Pada halaman ini terdapat dua proses yang dapat dilakukan pengguna aplikasi. Yaitu proses *converter* atau mengubah dokumen teks asli menjadi dokumen teks yang berformat *plaintext* (.txt atau .dtxt). Selanjutnya pada menu ini terdapat 2 buah tombol *submit* yang berfungsi untuk memulai proses *converter* dan proses deteksi plagiat.

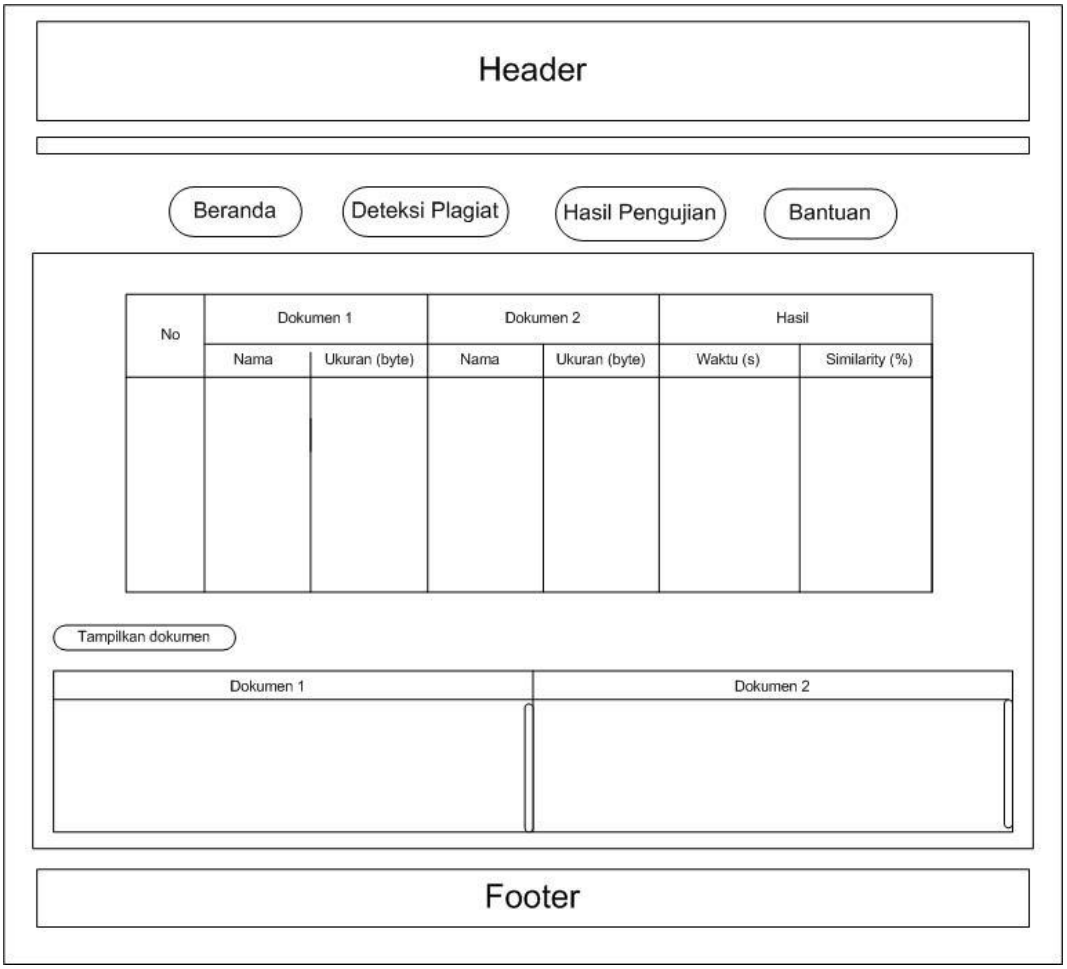
Berikut adalah rancangan menu deteksi plagiarisme yang akan dibuat:



Gambar 4.13 *Interface* Menu deteksi plagiarisme

Ketika tombol submit untuk proses deteksi plagiat dijalankan, selanjutnya halaman akan menampilkan hasil deteksi plagiat berupa tabel informasi penjiplakan dan dokumen teks yang memiliki nilai *fingerprint* yang sama.

Berikut adalah tampilan halaman hasil proses deteksi plagiat:

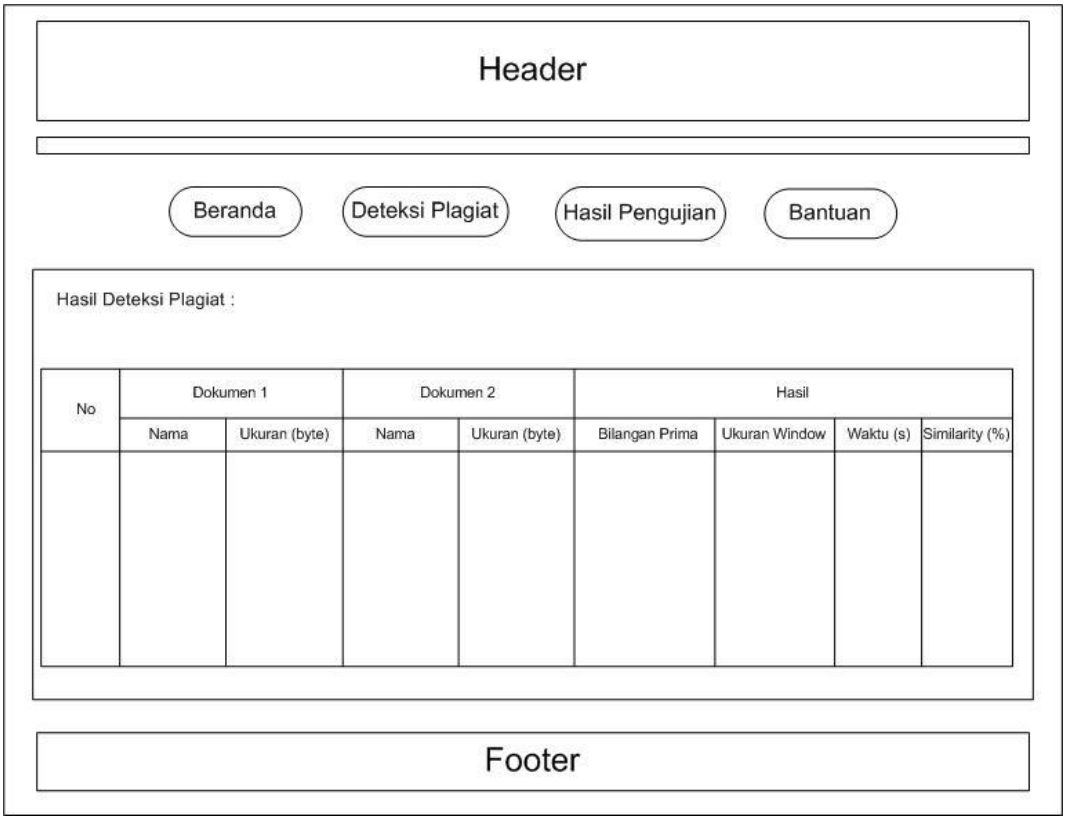


Gambar 4.14 *Interface* Hasil menu deteksi plagiarisme

4.4.3.3 Rancangan menu Hasil Pengujian

Untuk perancangan tampilan menu hasil, akan ditampilkan berupa sebuah tabel. Pada tabel tersebut memberikan informasi dokumen berupa nama dokumen, ukuran, waktu proses, bilangan prima dan ukuran *window* yang digunakan, serta hasil *similarity* pengujian dokumen.

Untuk lebih jelasnya dapat dilihat pada gambar berikut:



Gambar 4.15 *Interface* Menu hasil pengujian

4.4.3.4 Rancangan menu bantuan

Menu bantuan merupakan halaman yang berisi tentang petunjuk penggunaan aplikasi pendeteksi plagiarisme. Halaman ini memberikan informasi mengubah file dokumen menjadi sebuah file *plaintext* dan informasi tentang cara mendeteksi plagiarisme dokumen.

Berikut ini adalah gambar rancangan untuk menu bantuan:

Gambar 4.16 *Interface* Menu bantuan

BAB V
IMPLEMENTASI DAN PENGUJIAN

5.1. Tahapan Implementasi

Implementasi merupakan tahap pembuatan aplikasi agar dapat mengetahui apakah aplikasi yang telah dirancang dapat menghasilkan tujuan yang ingin dicapai sesuai dengan analisa dan perancangan yang telah dilakukan sebelumnya.

Pembahasan yang akan dijelaskan pada tahapan implementasi ini adalah sebagai berikut.

5.1.1. Batasan Implementasi

Adapun batasan implementasi pada aplikasi pendeteksi penjiplakan ini adalah sebagai berikut:

1. Bahasa pemrograman yang digunakan adalah PHP dengan mesin *database mySQL*.
2. Masukkan dokumen teks yang dapat dideteksi aplikasi penjiplakan ini adalah dokumen yang memiliki format *plaintext (.txt* dan *.dtxt)*.
3. Dalam proses deteksi plagiat, bilangan prima yang disediakan antara 2-11. Disebabkan karena kalkulasi perhitungan nilai *hash* sudah melewati *Error* diatas 34. Selanjutnya untuk ukuran *window* diberi batas rentang nilai antara 1-32. Hal ini agar mendapatkan hasil yang maksimal.

5.1.2. Lingkungan Implementasi

Lingkungan implementasi adalah lingkungan dimana aplikasi ini dikembangkan. Lingkungan implementasi aplikasi pendeteksi plagiarisme dokumen ini terdiri atas dua lingkungan yaitu lingkungan perangkat keras dan perangkat lunak. Berikut ini merupakan spesifikasi lingkungan tersebut:

1. Perangkat keras

   Berikut spesifikasi dari perangkat keras yang digunakan:

   Processor : *AMD C-60 With Radeon @ 1.0 GHz*

   Memori (RAM) : 2,00 GB

   Harddisk : 320 GB

2. Perangkat Lunak

   Berikut spesifikasi dari perangkat lunak yang digunakan:

   Sistem Operasi : *Windows 7 Ultimate* 32 bit

   Bahasa Pemrograman : PHP

   DBMS : mySQL

   Tools Perancangan : Notepad++

5.1.3. Implementasi Antarmuka Aplikasi

Tahapan ini merupakan tahap implementasi aplikasi dari hasil analisa yang telah diperoleh dan mengimplementasikan hasil perancangan antarmuka yang telah dibuat. Pada aplikasi pendeteksi plagiarisme dokumen ini memiliki empat menu, yaitu menu beranda, menu deteksi plagiat, menu hasil pengujian dan menu bantuan. Proses mendeteksi penjiplakan dokumen dapat dilakukan pada menu deteksi plagiat. Berikut ini merupakan implementasi aplikasi pendeteksi plagiarisme dokumen sesuai dengan menu yang ada pada aplikasi:

1. Implementasi Antarmuka Menu Beranda

Halaman beranda merupakan halaman yang berisi tentang definisi plagiarisme dan juga menerangkan fungsi perancangan aplikasi plagiarisme.
Berikut adalah tampilan antarmuka menu beranda:

Gambar 5.1 Antarmuka Menu Beranda

2. Implementasi Antarmuka Menu Deteksi Plagiat

Halaman deteksi plagiat ini merupakan halaman proses memulai melakukan deteksi plagiat dokumen teks.

Proses diawali dengan mengkonversi file dokumen teks dari file .doc menjadi file .txt atau .dtxt. Pilih file yang akan di *convert* dan klik telusuri. Hasil konversi dokumen tersebut akan disimpan pada tempat yang sama. Langkah selanjutnya adalah proses deteksi plagiat. Proses di awali dengan memasukkan dokumen yang akan diuji yang memiliki format .txt atau .dtxt.

Setelah dokumen dipilih, kemudian pilih bilangan prima dan ukuran *window* yang akan dijadikan untuk proses tokenisasi dan perhitungan nilai *hash*. Untuk bilangan prima yang disediakan adalah bernilai antara 2-11, sedangkan untuk ukuran *window* adalah antara 1-32. Selanjutnya klik tombol ‘mulai’ untuk memulai proses deteksi plagiat.

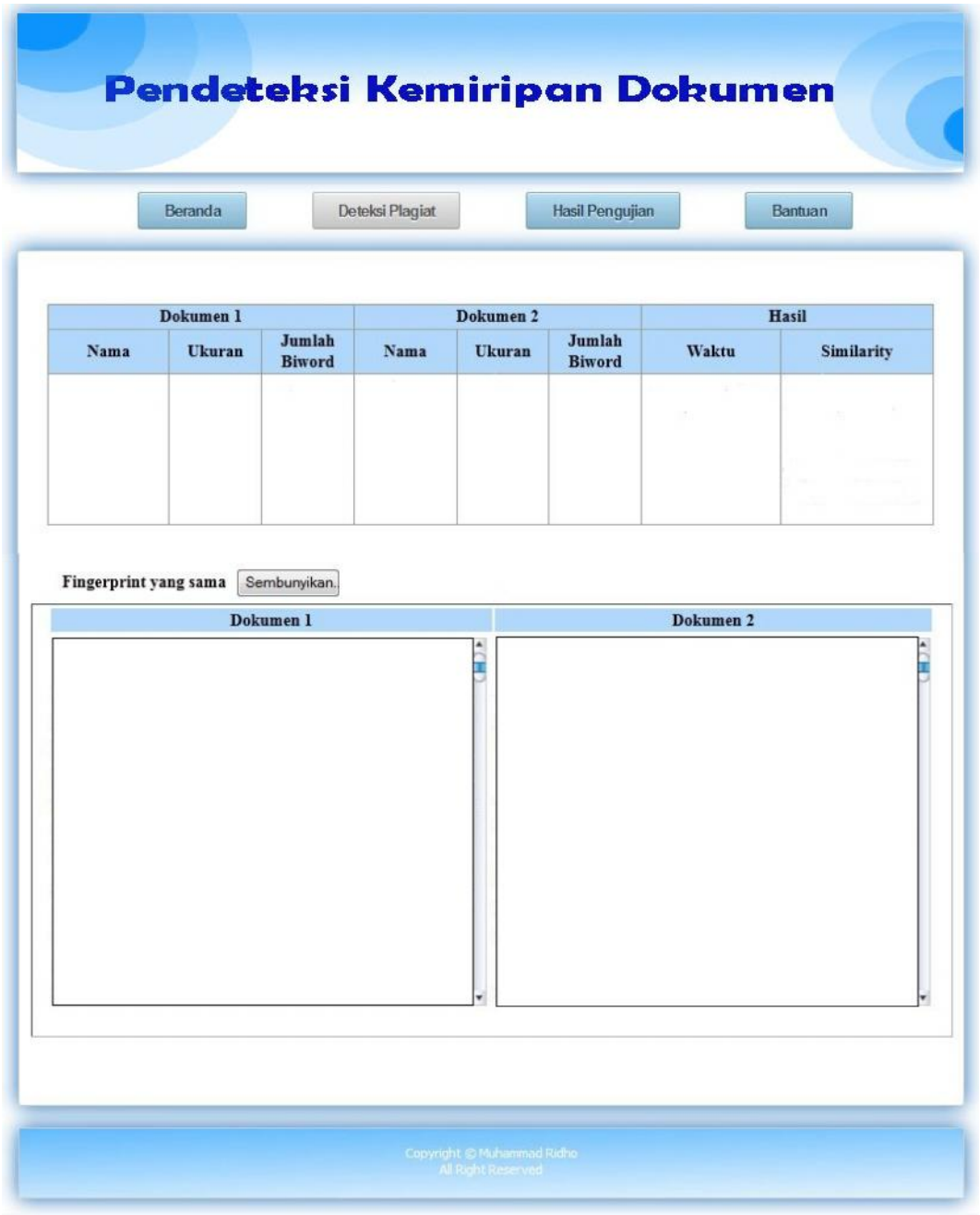
Untuk lebih jelasnya dapat dilihat pada gambar berikut ini:

Gambar 5.2 Antarmuka Menu Deteksi Plagiat

Pada gambar di atas dapat dilihat proses yang dapat dilakukan pada menu aplikasi plagiat. Yaitu proses mengubah dokumen teks menjadi dokumen yang memiliki format *plaintext.*

Proses selanjutnya adalah proses pendeteksian penjiplakan dokumen teks. Pada proses ini dilakukan penelusuran dokumen mana yang diuji. Kemudian dilanjutkan dengan mongkonfigurasi bilangan prima yang dipilkih dan ukuran *window* yang akan dipakai.

Jika proses pendeteksian telah dilakukan, aplikasi akan menampilkan informasi dokumen yang telah diuji dan juga *fingerprint* yang terdeteksi pada masing-masing dokumen. Berikut adalah antarmuka hasil proses pendeteksian penjiplakan:



Gambar 5.3 Antarmuka Hasil proses Menu Deteksi Plagiat

3. Implementasi antarmuka Menu Hasil Pengujian.

Menu hasil pengujian merupakan halaman yang menampilkan tabel hasil pengujian yang telah dilakukan pengguna. Hasil ini akan diperoleh oleh seitiap pengguna yang menjalankan aplikasi pendeteksi penjiplakan. Tabel hasil pengujian ini memberikan informasi berupa nama file yang diuji, ukuran file,

bilangan prima dan ukuran *window* yang dipakai, waktu proses deteksi, dan hasil *similarity* kemiripan dokumen.

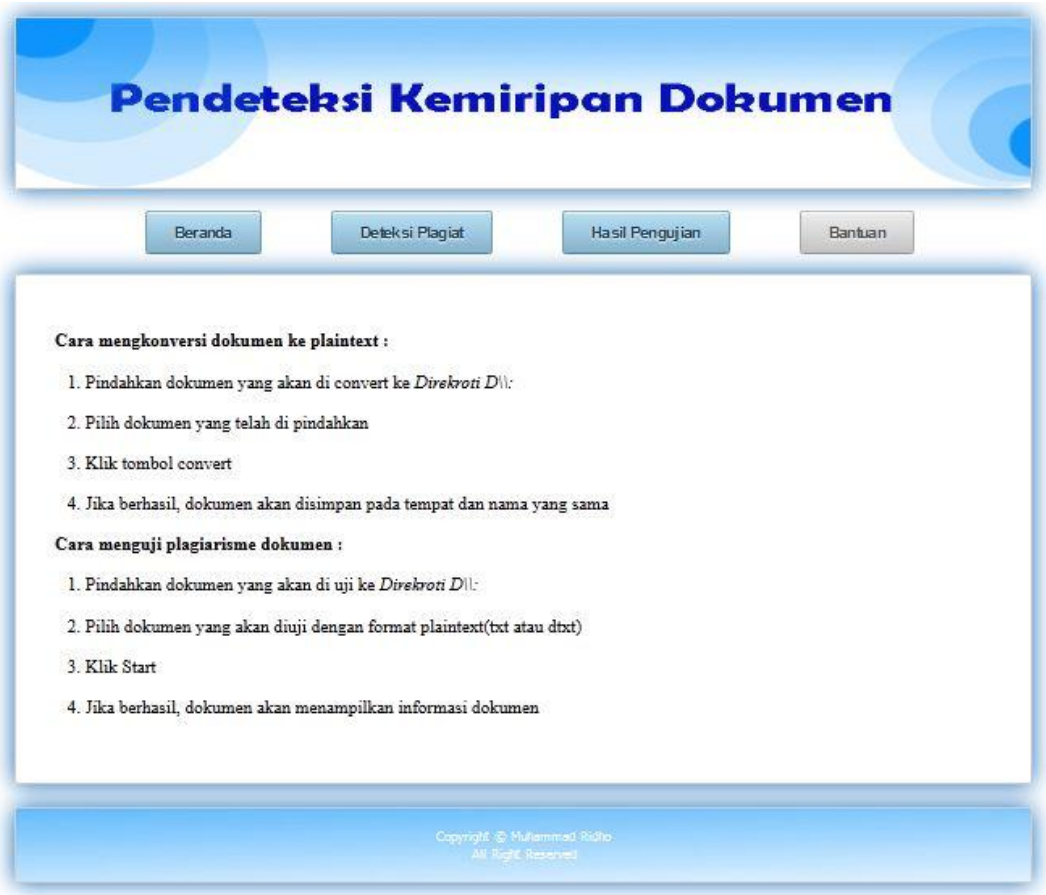
Berikut adalah tampilan interface Menu hasil pengujian:

Gambar 5.4 Antarmuka Menu Hasil Pengujian

4. Implementasi Antarmuka Menu Bantuan.

Menu bantuan merupakan menu yang menampilkan halaman berupa penjelasan cara menjalankan aplikasi deteksi plagiarisme. Pada menu bantuan ini, pengguna akan mendapatkan informasi cara mengkonversi dokumen teks menjadi dokumen yang memiliki format *plaintext* (.txt atau .dtxt). Selain itu, pengguna juga mendapatkan informasi cara menjalankan proses deteksi penjiplakan dokumen, sehingga pengguna tidak canggung lagi menggunakan aplikasi ini.

Berikut ini adalah tampilan antarmuka menu bantuan:



Gambar 5.5 Antarmuka Menu Bantuan

5.2. Hipotesa pengujian aplikasi

Tahap hipotesa merupakan tahap memberikan jawaban sementara terhadap algoritma *winnowing* dengan pendekatan *biword* sebelum melakukan implementasi dan pengujian terhadap aplikasi.

Diberikan sebuah hipotesa pada konfigurasi bilangan prima dan ukuran *window*. "Semakin kecil nilai bilangan prima, maka nilai *similarity* akan meningkat", disebabkan karena kalkulasi dengan nilai prima terkecil dapat menghasilkan nilai *fingerprint* terkecil juga. Sementara itu, "semakin tinggi ukuran *window* yang digunakan akan meningkatkan hasil *similarity"*, disebabkan karena penerapan MD5 yang memiliki jumlah karakter yang panjang, yaitu 32 karakter.

5.3. Pengujian aplikasi

Tahap pengujian aplikasi ini merupakan tahap yang dilakukan setelah tahap implementasi selesai. Tujuan tahap pengujian ini adalah untuk menjamin apakah aplikasi yang sudah dibangun telah sesuai dengan analisa dan perancangan yang dilakukan sehingga tujuan yang diinginkan tercapai.

5.3.1 Rencana Pengujian

Rencana pengujian yang akan dilakukan adalah sebagai berikut:

1. Pengujian *white box* yang bertujuan untuk mengetahui apakah proses-proses yang terdapat dalam algoritma yang digunakan telah sesuai dan telah diterapkan dengan benar.
2. Menguji konfigurasi nilai bilangan prima dan ukuran *window*. Pengujian ini bertujuan untuk mengetahui konfigurasi mana yang lebih baik dan menghasilkan *similarity* tertinggi dalam mendeteksi plagiarisme dokumen teks.

5.3.1.1 Pengujian *whitebox*

Pada pengujian ini dilakukan pengujian terhadap proses-proses yang terdapat dalam algoritma, apakah telah diterapkan dengan benar pada aplikasi pendeteksi penjiplakan dokumen ini.

1. *Whitespace Intensitivity.*

   Syarat yang terdapat dalam proses ini diantaranya adalah menghilangkan spasi, mengubah huruf capital menjadi kecil, dan membuang karakter yang tidak relevan.

Berikut adalah hasil pengujian *whitespace intensitivity* yang telah dilakukan pada sebuah dokumen *plaintext*:

Tabel 5.1 Pengujian *whitespace intensitivity*

Nama	Ukuran	Hasil *Preprocesssing*
t1.dtxt	417 bytes	universitas islam negeri sultan syarif kasim merupakan universitas yang terdapat di provinsi riau universitas ini memiliki 8 fakultas yang di dalamnya memiliki banyak jurusan baik jurusan yang berbasis islamic maupun bersifat umum salah satu fakultas yang terdapat di universitas ini adalah fakultas sains dan teknologi teknik informatika adalah salah satu jurusan yang terdapat di fakultas sains dan teknologi

2. Tokenisasi.

Dokumen teks yang telah melewati proses *preprocessing*, selanjutnya akan dibagi-bagi menjadi token *biword.*

Berikut adalah hasil pengujian tokenisasi pada sebuah dokumen *plaintext*:

Tabel 5.2 Pengujian proses Tokenisasi

Nama	Ukuran	Hasil Tokenisasi
t1.dtxt	417 bytes	universitas islam islam negeri negeri sultan sultan syarif syarif kasim kasim merupakan merupakan universitas universitas yang yang terdapat terdapat di di provinsi provinsi riau universitas ini

		ini memiliki memiliki 8 8 fakultas fakultas yang yang di di dalamnya dalamnya memiliki memiliki banyak banyak jurusan jurusan baik baik jurusan jurusan yang yang berbasis berbasis islamic islamic maupun maupun bersifat bersifat umum umum salah salah satu satu fakultas fakultas yang yang terdapat terdapat di di universitas universitas ini ini adalah adalah fakultas fakultas sains sains dan dan teknologi teknologi teknik

		teknik informatika informatika adalah adalah salah salah satu satu jurusan jurusan yang yang terdapat terdapat di di fakultas fakultas sains sains dan dan teknologi

3. Perhitungan nilai *hash.*

Setelah proses tokenisasi dilakukan, selanjutnya akan dilakukan perhitungan nilai *hash*. Setiap token biword yang dibentuk akan di cari nilai *hash*-nya menggunakan persamaan *rolling hash.*

Berikut adalah hasil pengujian perhitungan nilai *hash* tiap-tiap token biword pada sebuah dokumen *plaintext*:

Tabel 5.3 Pengujian perhitungan nilai *hash*

Nama	Ukuran	Hasil Tokenisasi
t1.dtxt	417 bytes	5.8928789443634E+16 5.9167894083407E+16 5.85551611404E+16 5.7054682866089E+16 7.9633002980822E+16 5.3204988856013E+16 5.6992993076577E+16 5.7456370603087E+16 8.0218640338309E+16 8.2292093218188E+16 7.8576215812583E+16 5.0614616437096E+16

		7.9294974765488E+16
		7.8210782487192E+16
		4.7523087684335E+16
		6.1200644445106E+16
		5.9826123045259E+16
		6.4196655167606E+16
		4.8552062298227E+16
		6.2855266977963E+16
		5.4274101429662E+16
		5.5440130588783E+16
		7.9830111355219E+16
		4.6942587080859E+16
		4.9056851588685E+16
		7.9287077961835E+16
		4.831781555593E+16
		7.9095378174655E+16
		4.9980873242189E+16
		4.6671177809127E+16
		6.0209560964465E+16
		5.091058325301E+16
		4.8098175164531E+16
		4.916039179393E+16
		6.4196655167606E+16
		8.0218640338309E+16
		8.2292093218188E+16
		4.6778057251224E+16
		7.8210782487192E+16
		4.6896535821728E+16
		5.5083799718002E+16
		8.7803038570548E+16
		8.6520953534561E+16
		5.9355769907327E+16
		5.0645798241092E+16
		4.9667213130659E+16
		5.1825664534441E+16
		8.0070162531149E+16
		4.8098175164531E+16

		5.2666312138911E+16 7.9287077961835E+16 8.0218640338309E+16 8.2292093218188E+16 4.9030322413805E+16 8.7803038570548E+16 8.6520953534561E+16 5.9355769907327E+16

4. Pemilihan *hash* terkecil (*fingerprint).*

Nilai-nilai *hash* yang telah diperoleh pada tabel 5.3 selanjutnya akan dibentuk dalam beberapa *window* dengan ukuran *w*. Kemudian dipilih nilai-nilai *hash* terkecil dari setiap *window*. Nilai *hash* terkecil ini akan dijadikan *fingerprint* sebuah dokumen.

Berikut adalah hasil pengujian pemilihan *fingerprint* dari sebuah dokumen *plaintext*:

Tabel 5.4 Pengujian pemilihan *fingerprint*

Nama	Ukuran	Hasil Tokenisasi
t1.dtxt	417 bytes	5.7054682866089E+16 5.3204988856013E+16 5.6992993076577E+16 5.7456370603087E+16 5.0614616437096E+16 4.7523087684335E+16 4.8552062298227E+16 5.4274101429662E+16 4.6942587080859E+16 4.831781555593E+16 4.6671177809127E+16 4.916039179393E+16 4.6778057251224E+16 4.6896535821728E+16 5.5083799718002E+16 5.0645798241092E+16

		4.9667213130659E+16 4.8098175164531E+16 5.2666312138911E+16 4.9030322413805E+16

Pengujian diatas bertujuan untuk mengetahui nilai *hash* mana yang akan dijadikan *fingerprint*, sehingga dapat dilakukan perhitungan *similarity* dokumen.

5.3.1.2 Pengujian Konfigurasi

Pada pengujian ini dilakukan pengujian terhadap aplikasi pendeteksi penjiplakan dokumen dengan berbagai konfigurasi pada nilai bilangan prima dan ukuran *window*. Pengujian ini dilakukan untuk mendapatkan konfigurasi yang paling baik terkait dengan bilangan prima dan ukuran *window* yang digunakan. Pengujian ini dilakukan pada dokumen teks yang telah dieksport menjadi file *plaintext.*

a. Pengujian I:

Pengujian tahap pertama yang dilakukan ini akan membandingkan dua buah dokumen teks yang memiliki topik yang sama, yaitu membahas tentang **analisa terhadap sistem peramalan emas**, dan kemudian beberapa kata atau kalimat-kalimat yang terdapat dalam dokumen telah dimodifikasi sebesar 10% pada paragraf awal masing-masing dokumen. Pengujian pertama ini dilakukan dengan enam buah konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda.

Pada pengujian pertama dilakukan pada dua buah dokumen *plaintext* dengan mengkonfigurasi nilai bilangan prima yang berbeda-beda yang dipilih secara acak. yaitu:

Konfigurasi 1:

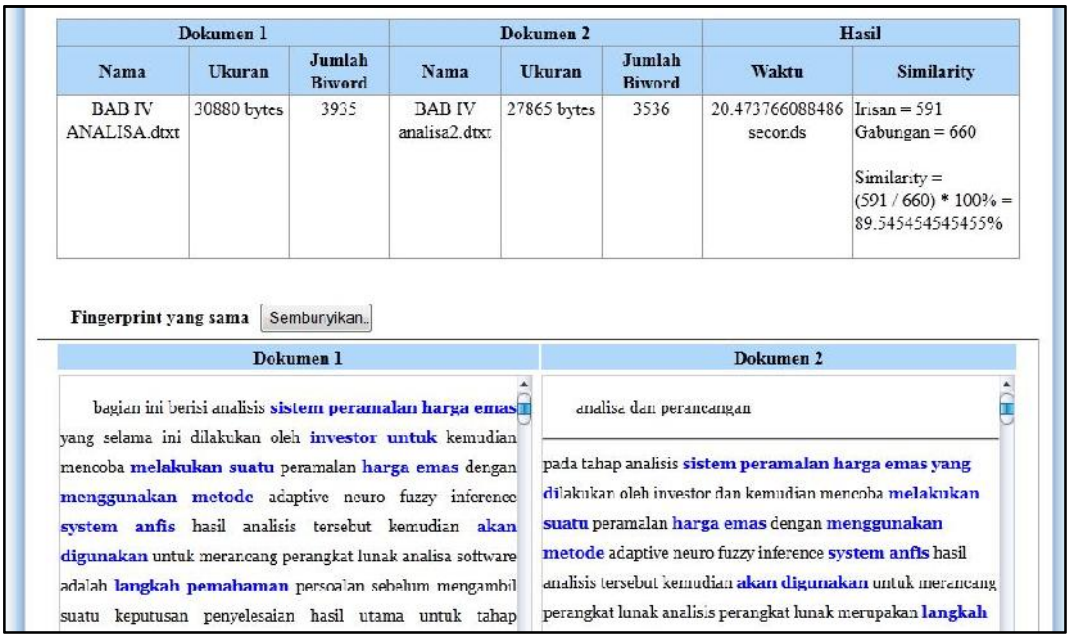
- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Setelah pengujian dilakukan, diperoleh hasil pengujian sebagai berikut:

Tabel 5.5 Hasil pengujian I konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	20.473	89.55

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 1:



Gambar 5.6 *Screen-shoot* pengujian 1 konfigurasi 1

Konfigurasi 2:

- Bilangan prima: 5
- Ukuran *window*: 8
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

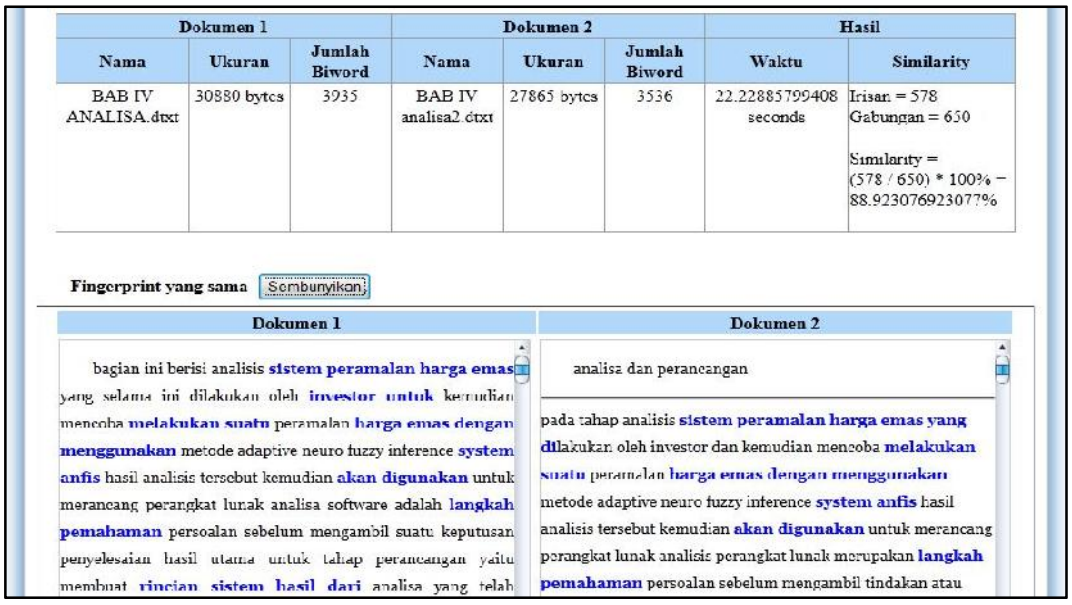
Pengujian dengan konfigurasi diatas digunakan untuk mengetahui seberapa besar dampak dari perubahan bilangan prima terhadap *similarity* yang dihasilkan oleh aplikasi ini.

Setelah pengujian dilakukan, diperoleh hasil pengujian sebagai berikut:

Tabel 5.6 Hasil pengujian I konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	22.228	88.92

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 2:



Gambar 5.7 *Screen-shoot* pengujian 1 konfigurasi 2

Pada konfigurasi 2 ini telah dilakukan peningkatan nilai bilangan prima dari konfigurasi 1 sebelumnya dari b=2 menjadi b=5. Namun berdasarkan hasil pada pada tabel 5.6 diatas dapat dilihat bahwa nilai *similarity* mengalami penurunan setelah dilakukan konfigurasi bilangan prima yang berbeda, yaitu dari 89.55% menjadi 88.92%.

Konfigurasi 3:

- Bilangan prima: 11
- Ukuran *window*: 8
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

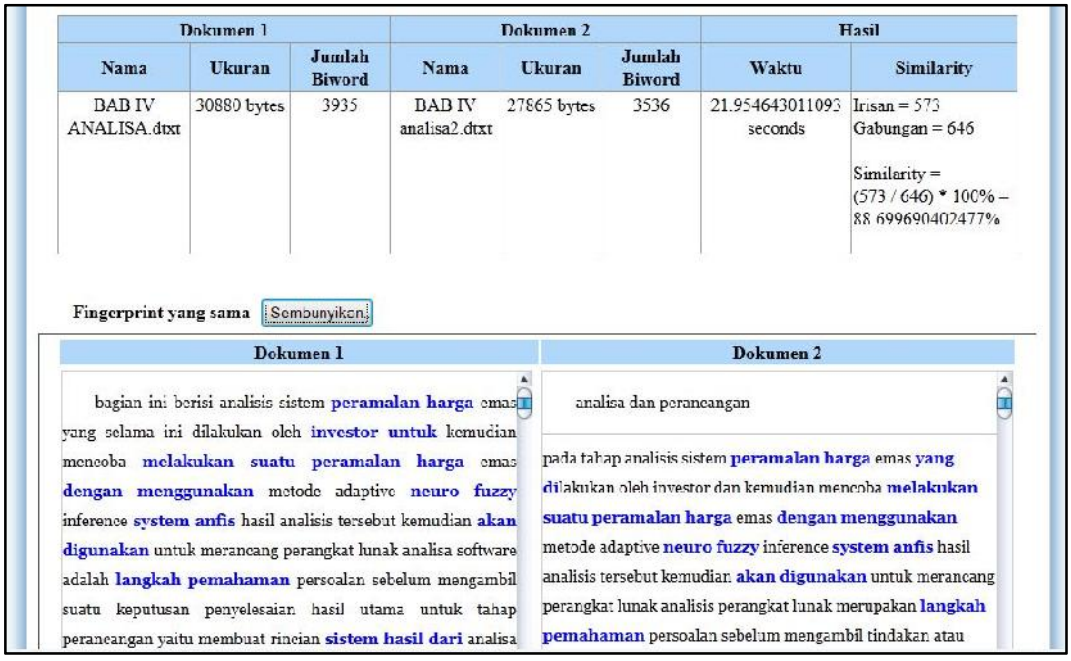
Pengujian dengan konfigurasi diatas digunakan untuk mengetahui seberapa besar dampak dari perubahan bilangan prima terhadap *similarity* yang dihasilkan oleh aplikasi ini.

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.7 Hasil pengujian I konfigurasi 3

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	21.954	88.70

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 3:



Gambar 5.8 *Screen-shoot* pengujian 1 konfigurasi 3

Pada konfigurasi 3 ini telah dilakukan peningkatan nilai bilangan prima dari konfigurasi 2 sebelumnya dari b=5 menjadi b=11. Namun berdasarkan hasil pada pada tabel 5.7 diatas dapat dilihat bahwa nilai *similarity* masih mengalami penurunan setelah dilakukan konfigurasi bilangan prima yang berbeda, yaitu dari 88.92.55% menjadi 88.69%.

Konfigurasi 4:

Pada pengujian konfigurasi 1-3, telah dilakukan perubahan konfigurasi terhadap nilai bilangan prima. Selanjutnya pada pengujian konfigurasi 4-6 akan dilakukan dengan melakukan perubahan konfigurasi terhadap ukuran *window* agar dapat mengetahui seberapa besar perubahan nilai *similarity* yang dihasilkan.

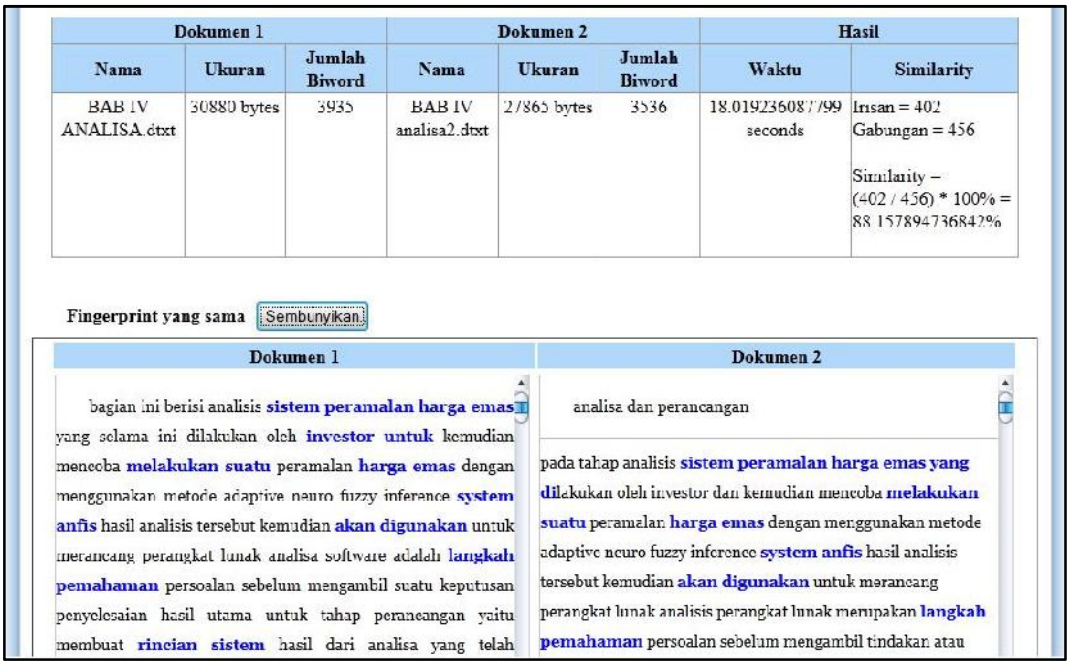
- Bilangan prima: 3
- Ukuran *window*: 12
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.8 Hasil pengujian II konfigurasi 4

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	18.019	88.16

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 4:



Gambar 5.9 *Screen-shoot* pengujian I konfigurasi 4

Konfigurasi 5:

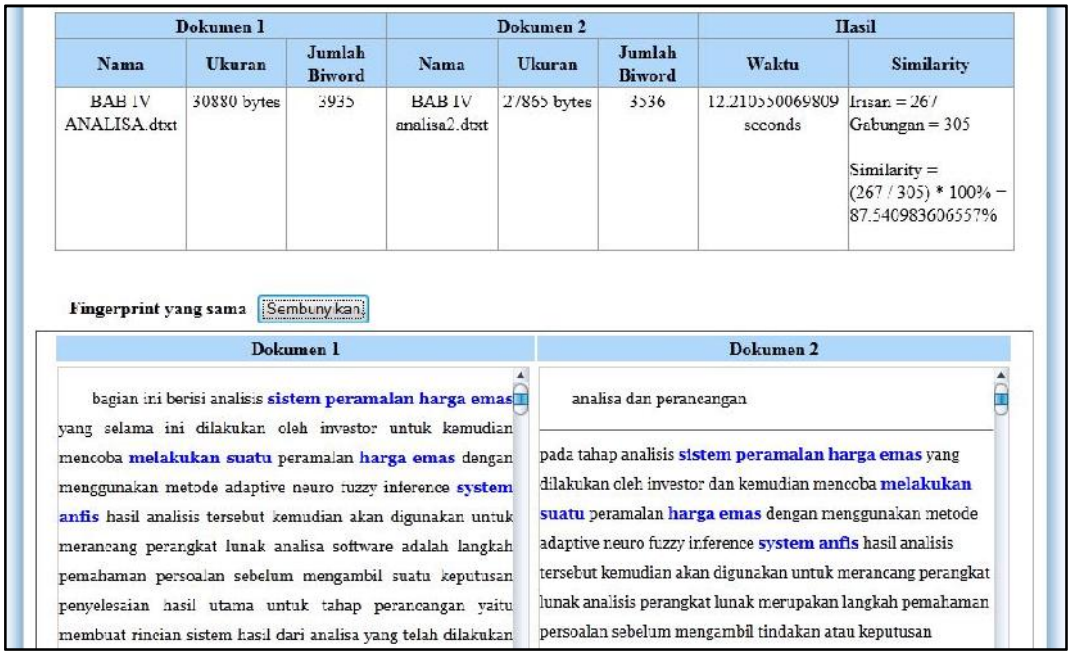
- Bilangan prima: 3
- Ukuran *window*: 20
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.9 Hasil pengujian II konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	12.210	87.54

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 5:



Gambar 5.10 *Screen-shoot* pengujian I konfigurasi 5

Pada pengujian 2 konfigurasi 6 telah dilakukan perubahan konfigurasi terhadap ukuran *window* dengan menaikkan nilainya. Maka diperoleh penurunan nilai *similarity* dari 88.16% menjadi 87.54%. Dengan demikian pemilihan ukuran *window* teritinggi tidak memberikan nilai *similarity* yang tinggi juga.

Konfigurasi 6:

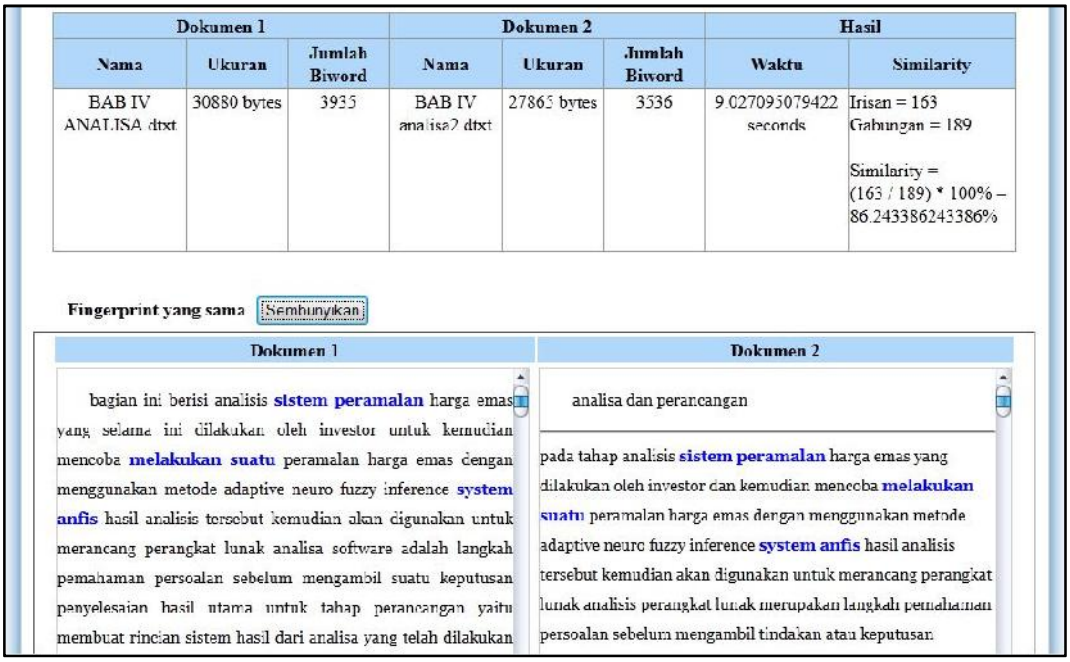
- Bilangan prima: 3
- Ukuran *window*: 32
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.10 Hasil pengujian II konfigurasi 6

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	9.027	86.24

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 6:



Gambar 5.11 *Screen-shoot* pengujian 1 konfigurasi 6

Pada konfigurasi ke-6, juga dilakukan konfigurasi dengan menaikkan ukuran *window* dari konfigurasi 5 sebelumnya, namun hasil *similarity* yang diperoleh masih mengalami penurunan.

b. Pengujian II:

Pada pengujian tahap kedua, dilakukan pengujian dengan dokumen yang berbeda. Pengujian dilakukan dengan membandingkan dua buah dokumen teks yang memiliki bidang yang sama, yaitu dokumen surat lamaran pekerjaan yang memiliki format penulisan yang hampir sama. Dokumen telah dimodifikasi pada sebagian kata atau kalimat-kalimat yang terdapat dalam dokumen sebesar 30%. Pengujian kedua ini juga dilakukan dengan enam buah konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda

Pada pengujian kedua, dilakukan pada dua buah dokumen *plaintext* dengan mengkonfigurasi nilai bilangan prima yang berbeda-beda yang dipilih secara acak. yaitu:

Konfigurasi 1:

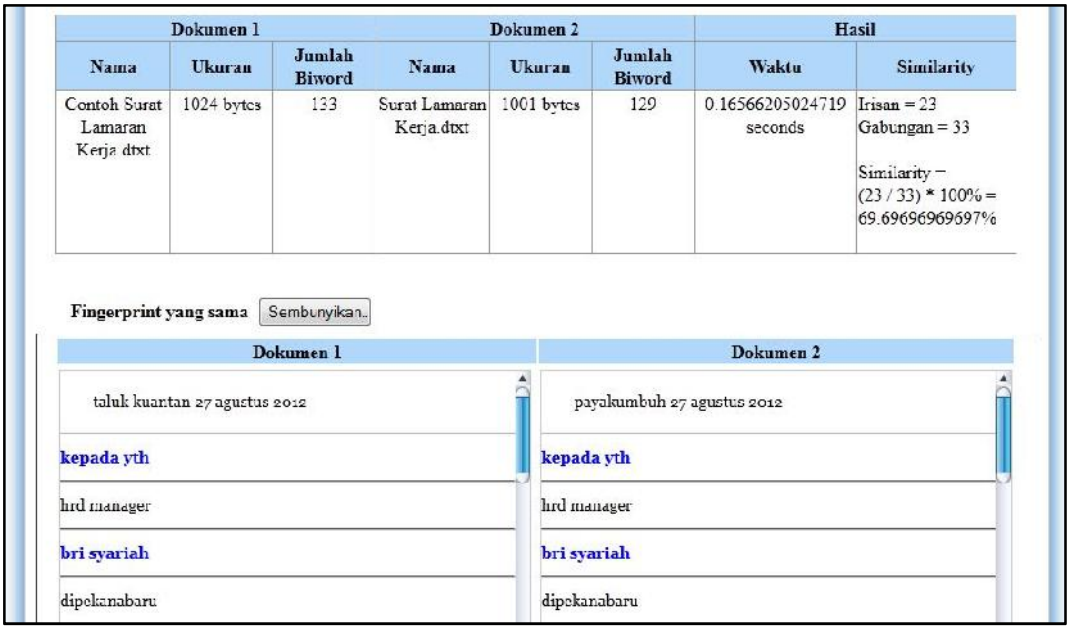
- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.11 Hasil pengujian II konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1656	69.69

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 1:



Gambar 5.12 *Screen-shoot* pengujian II konfigurasi 1

Pada konfigurasi ke-2, telah dilakukan perubahan konfigurasi terhadap nilai bilangan prima. Dengan menaikkan nilai bilangan prima, diperoleh nilai *similarity* yang lebih rendah dibandingkan konfigurasi sebelumnya.

Konfigurasi 2:

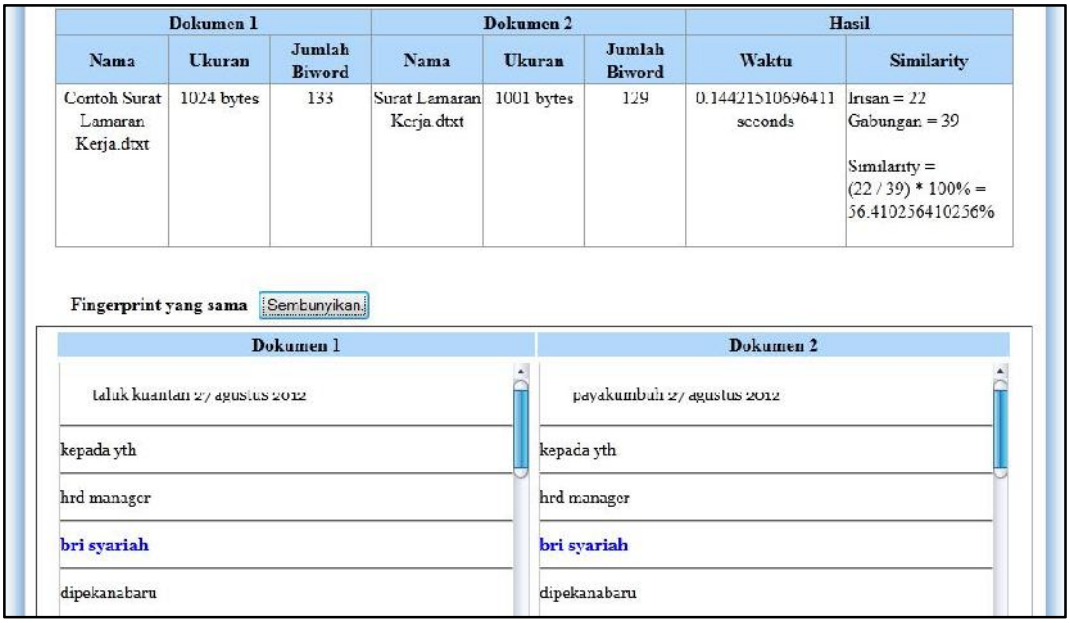
- Bilangan prima: 5
- Ukuran *window*: 8
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.12 Hasil pengujian III konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1442	56.41

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 2:



Gambar 5.13 *Screen-shoot* pengujian II konfigurasi 2

Konfigurasi 3:

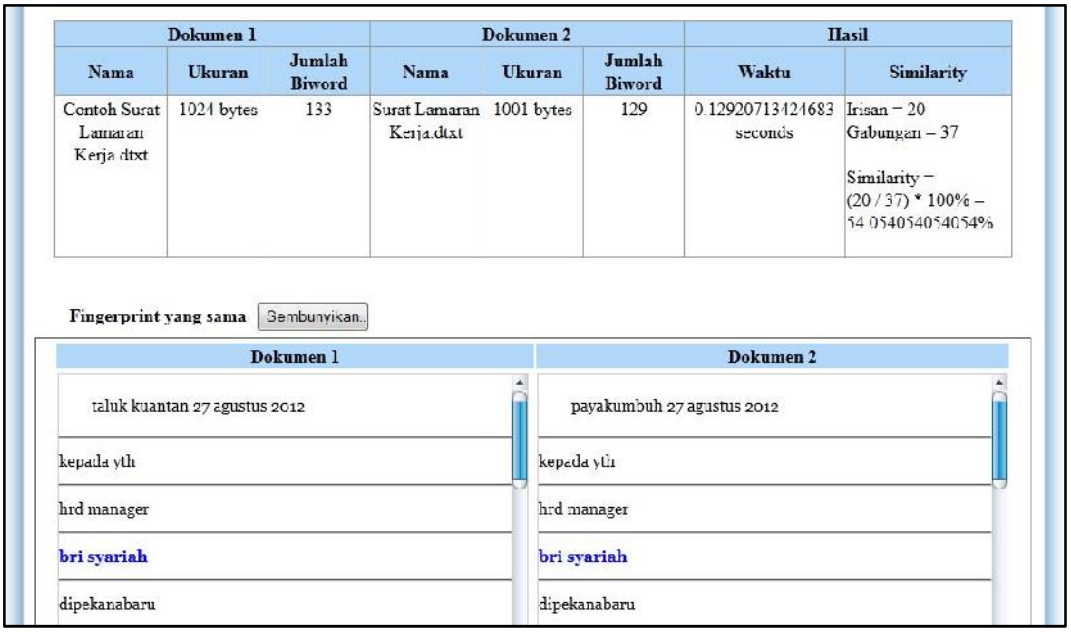
- Bilangan prima: 11
- Ukuran *window*: 8
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.13 Hasil pengujian II konfigurasi 3

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1269	54.05

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 3:



Gambar 5.14 *Screen-shoot* pengujian II konfigurasi 3

Pada konfigurasi 3, dilakukan konfigurasi bilangan prima dengan menaikkan nilainya, sehingga diperoleh nilai *similarity* 54.05%. Nilai ini mengalami penurunan dari nilai *similarity* pada konfigurasi sebelumnya.

Konfigurasi 4:

Pengujian dengan konfigurasi ke-4 ini dilakukan pada dua buah dokumen *plaintext* yang berbeda dengan mengkonfigurasi ukuran *window* yang berbeda-beda yang dipilih secara acak. yaitu:

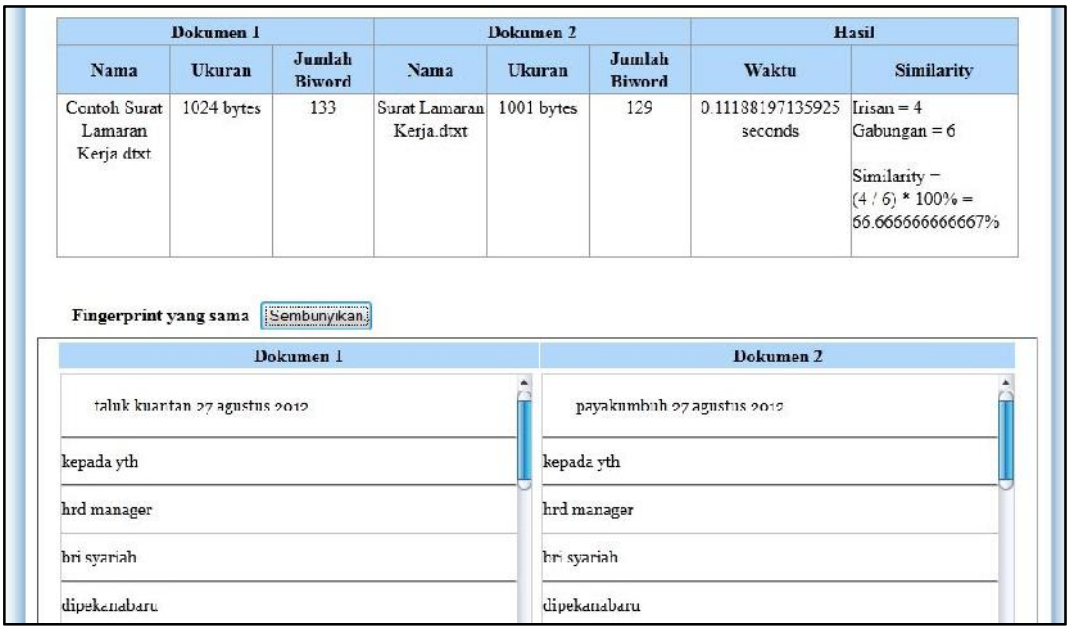
- Bilangan prima: 3
- Ukuran *window*: 32
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.14 Hasil pengujian II konfigurasi 4

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1111	66.67

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 4:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
Contoh Surat Lamaran Kerja.dtxt	1024 bytes	133	Surat Lamaran Kerja.dtxt	1001 bytes	129	0.11188197135925 seconds	Irisan = 4 Gabungan = 6 Similarity = (4 / 6) * 100% = 66.666666666667%

Gambar 5.15 *Screen-shoot* pengujian II konfigurasi 4

Konfigurasi 5:

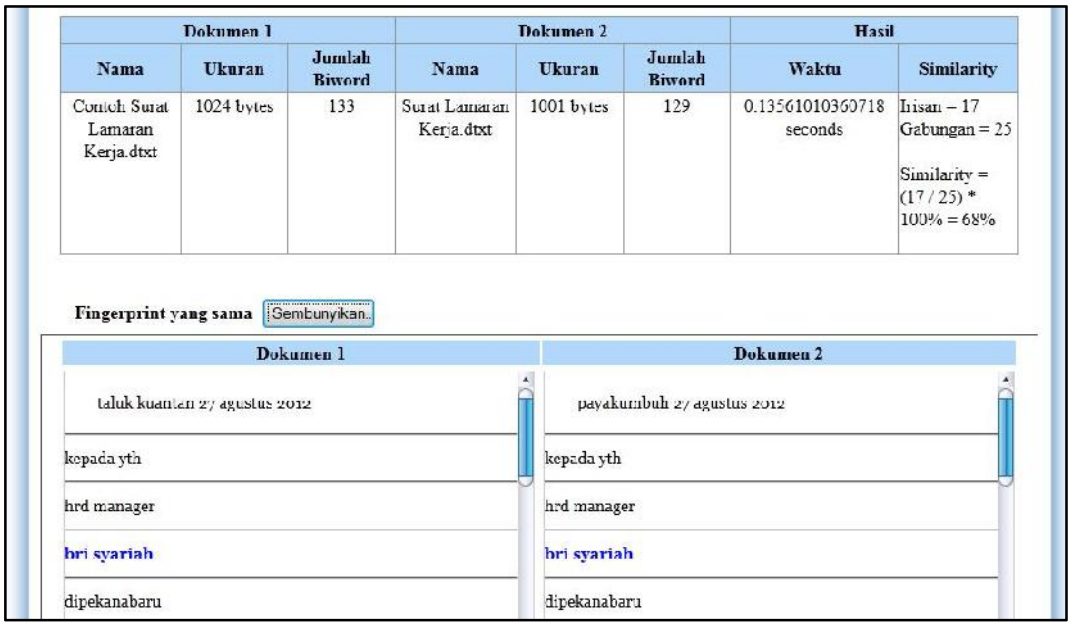
- Bilangan prima: 3
- Ukuran *window*: 12
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.15 Hasil pengujian II konfigurasi 5

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1356	68

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 5:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
Contoh Surat Lamaran Kerja.dtxt	1024 bytes	133	Surat Lamaran Kerja.dtxt	1001 bytes	129	0.13561010360718 seconds	Irisan = 17 Gabungan = 25 Similarity = (17 / 25) * 100% = 68%

Gambar 5.16 *Screen-shoot* pengujian II konfigurasi 5

Pada konfigurasi 5 telah dilakukan pengujian dengan memilih ukuran *window* yang lebih tinggi dari konfigurasi sebelumnya. Ternyata nilai *similarity* yang diberikan mengalami penurunan. *Similarity* mengalami penurunan dari 68% menjadi 63%.

Konfigurasi 6:

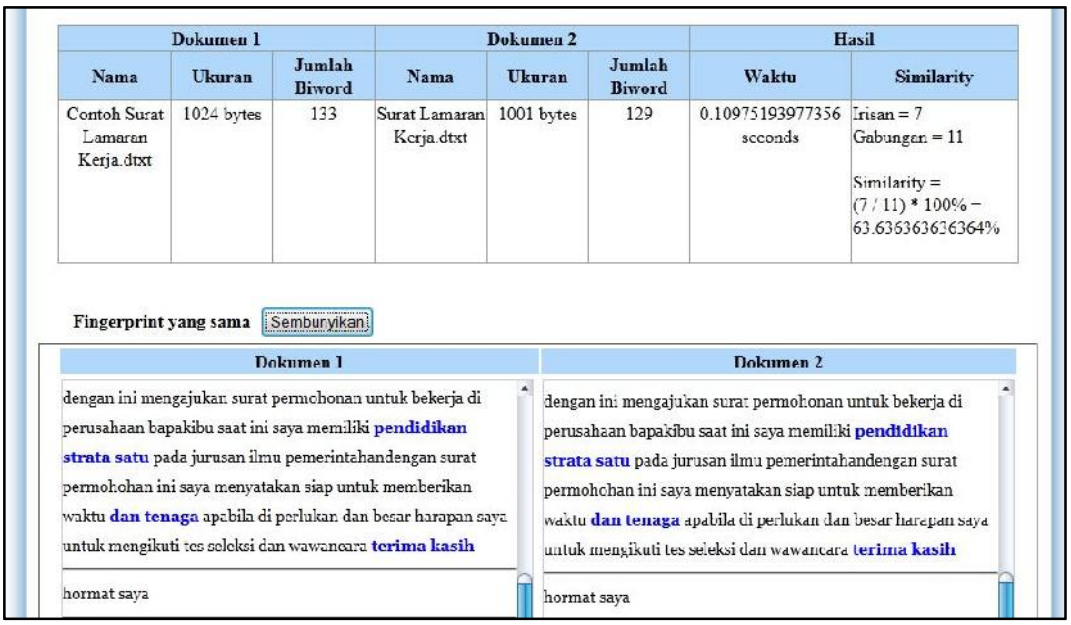
- Bilangan prima: 3
- Ukuran *window*: 20
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.16 Hasil pengujian II konfigurasi 6

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Contoh Surat	1024	Surat Lamaran	1001	0.1097	63.64

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 6:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
Contoh Surat Lamaran Kerja.dtxt	1024 bytes	133	Surat Lamaran Kerja.dtxt	1001 bytes	129	0.10975193977356 seconds	Irisan = 7 Gabungan = 11 Similarity = (7 / 11) * 100% = 63.636363636364%

Gambar 5.17 *Screen-shoot* pengujian II konfigurasi 6

Pada pengujian dengan konfigurasi 6 juga dilakukan pemilihan ukuran *window* yang lebih tinggi dari konfigurasi sebelumnya. Namun nilai *similarity* juga tidak memberikan nilai tertinggi dibanding nilai *similarity* pada konfigurasi 4.

c. Pengujian III:

Pada pengujian ketiga, dilakukan pengujian terhadap dokumen teks yang memiliki isi yang sama, namun posisi kata dalam teks telah di ubah sebagian. Pengujian ketiga ini dilakukan dengan 4 buah konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda

Untuk konfigurasi 1 dan 2, dilakukan konfigurasi denga ukuran *window* yang sama dan nilai bilangan prima yang berbeda-beda yang dipilih secara acak. yaitu:

Konfigurasi 1:

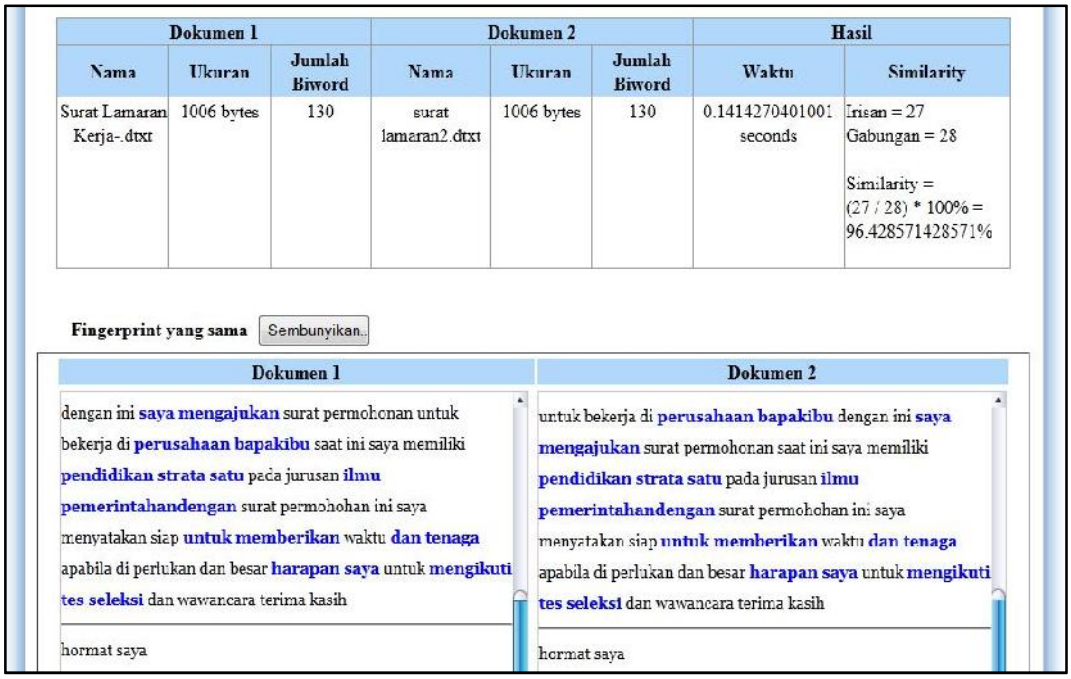
- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: Surat Lamaran Kerja.dtxt dan surat lamaran2

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.17 Hasil pengujian III konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Surat Lamaran Kerja	1006	Surat Lamaran2	1006	0.1414	96.43

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 1:



Gambar 5.18 *Screen-shoot* pengujian III konfigurasi 1

Konfigurasi 2:

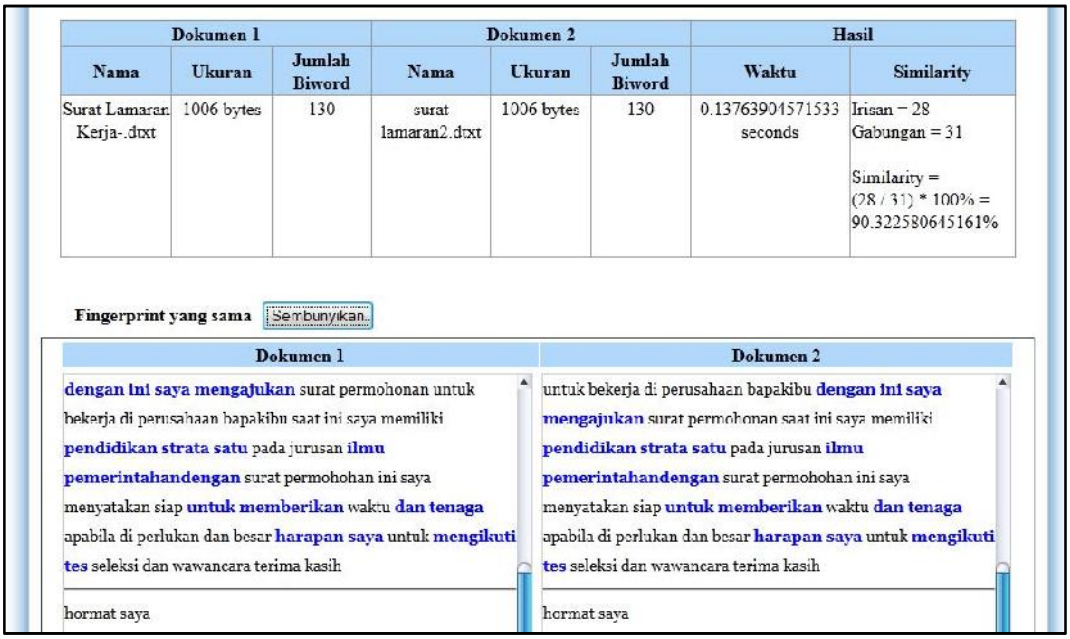
- Bilangan prima: 5
- Ukuran *window*: 8
- Dokumen plaintext: Surat Lamaran Kerja.dtxt dan surat lamaran2

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.18 Hasil pengujian III konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Surat Lamaran Kerja	1006	Surat Lamaran2	1006	0.1376	90.32

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 2:



Gambar 5.19 *Screen-shoot* pengujian III konfigurasi 2

Pada konfigurasi ke-2, telah dilakukan perubahan konfigurasi terhadap nilai bilangan prima. Dengan menaikkan nilai bilangan prima, diperoleh nilai *similarity* yang lebih rendah dibandingkan konfigurasi sebelumnya.

Konfigurasi 3:

Pada pengujian konfigurasi ini, dilakukan pengujian dengan mengkonfigurasi ukuran *window* yang berbeda-beda yang dipilih secara acak yaitu:

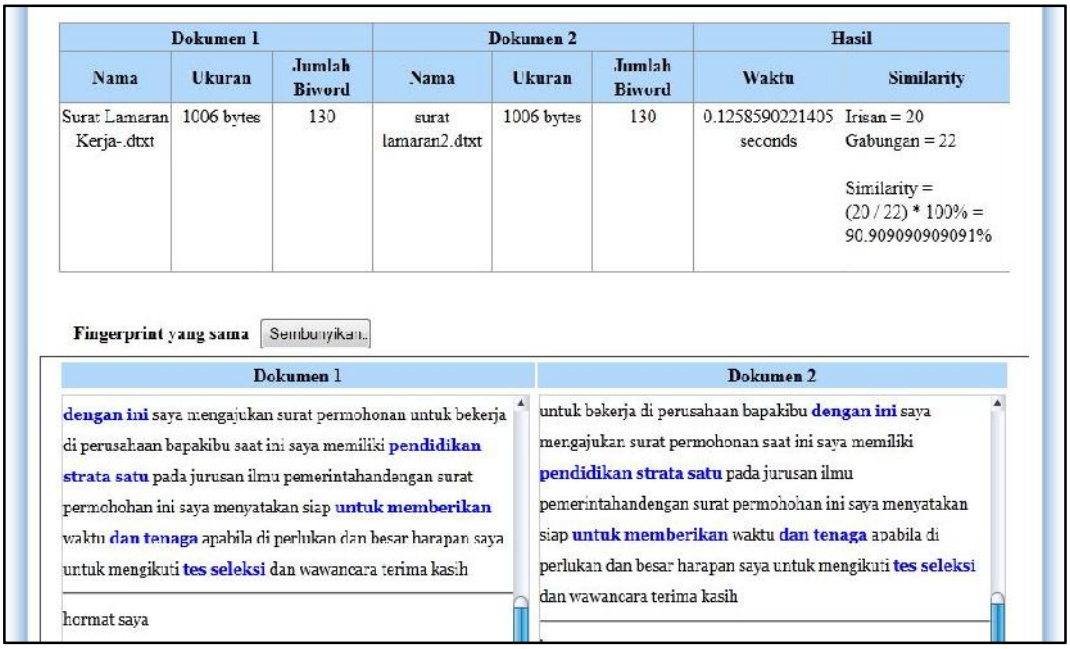
- Bilangan prima: 3
- Ukuran *window*: 12
- Dokumen plaintext: Surat Lamaran Kerja.dtxt dan surat lamaran2

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.19 Hasil pengujian III konfigurasi 3

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Surat Lamaran Kerja	1006	Surat Lamaran2	1006	0.1258	90.91

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 3:



Gambar 5.20 *Screen-shoot* pengujian III konfigurasi 3

Konfigurasi 4:

Pengujian dengan konfigurasi ke-4 ini juga dilakukan dengan mengkonfigurasi ukuran *window* dengan nilai bilangan prima yang sama yaitu:

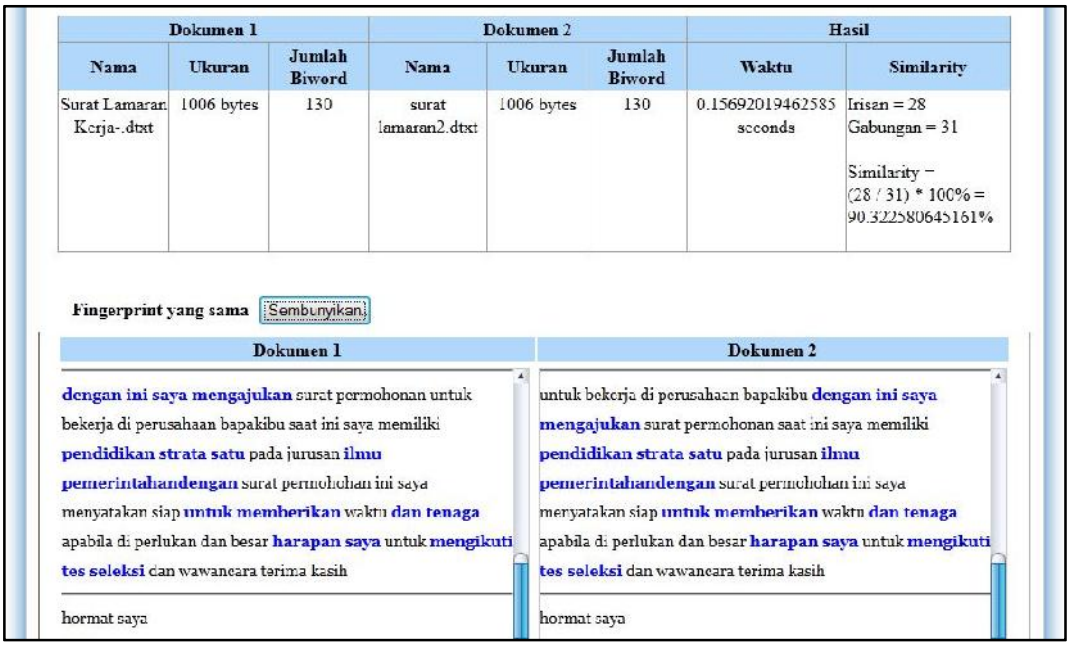
- Bilangan prima: 3
- Ukuran *window*: 8
- Dokumen plaintext: Surat Lamaran Kerja.dtxt dan surat lamaran2

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.20 Hasil pengujian III konfigurasi 4

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Surat Lamaran Kerja	1006	Surat Lamaran2	1006	0.1569	90.32

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 4:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
Surat Lamaran Kerja-.dtxt	1006 bytes	130	surat lamaran2.dtxt	1006 bytes	130	0.15692019462585 seconds	Irisan = 28 Gabungan = 31 Similarity = (28 / 31) * 100% = 90.322580645161%

Gambar 5.21 *Screen-shoot* pengujian III konfigurasi 4

Pada konfigurasi 4, dilakukan konfigurasi dengan mengubah menurunkan nilai ukuran *window* dari konfigurasi sebelumnya, sehingga diperoleh nilai *similarity* 90.32%.

Setelah dilakukan pengujian III, dapat ditarik sebuah kesimpulan bahwa aplikasi bisa menemukan nilai *fingerprint* yang sama dalam teks dokumen meskipun telah dilakukan perubahan posisi kata dalam teks, asalkan posisi teks yang di ubah masih dalam kondisi *biword* yang sama, namun nilai *similarity* yang diberikan mengalami penurunan disebabkan karena beberapa token *biword* mengalami perubahan nilai sehingga *fingerprint* yang ditemukan bukan *fingerprint* dengan nilai terkecil dalam suatu *window*.

d. Pengujian IV:

Pada pengujian keempat, dilakukan pengujian terhadap perubahan bahasa yang dilakukan pada dokumen teks yang diuji. Dokumen yang akan diuji memiliki pembahasan yang sama, namun memiliki bahasa yang berbeda yaitu berbahasa Indonesia dan bahasa melayu. Pengujian ini dilakukan pada dokumen teks *subtitle* sebuah film. Pengujian kelima ini dilakukan dengan 2 buah konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda

Berikut adalah konfigurasi bilangan prima dan ukuran *window* yang berbeda-deda yang dipilih secara acak.

Konfigurasi 1:

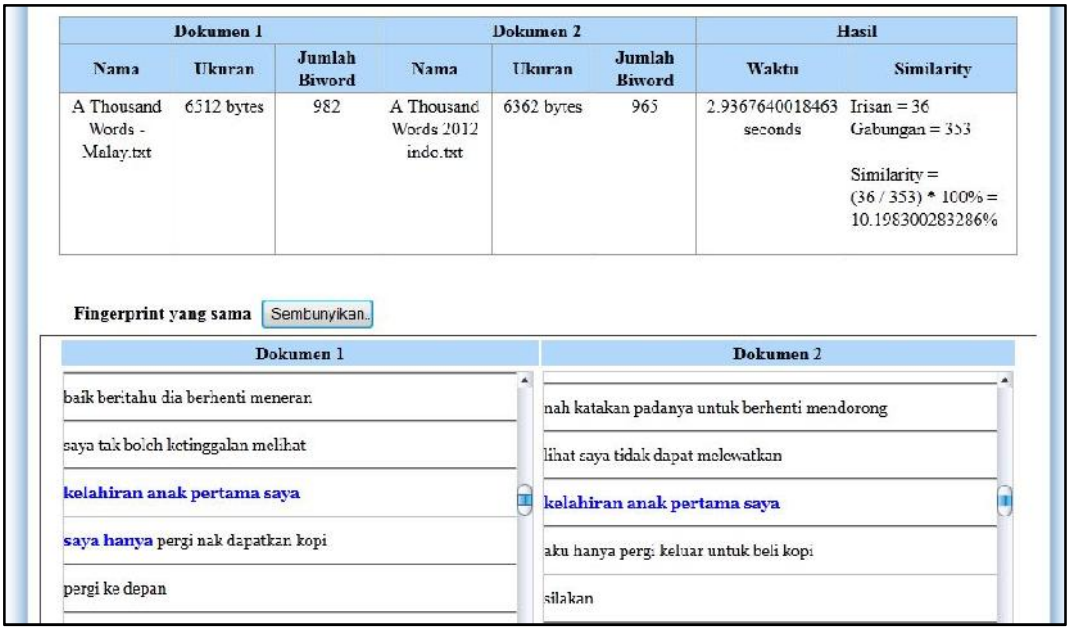
- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: A Thousand Words - Malay.txt dan A Thousand Words 2012 indo.txt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.21 Hasil pengujian IV konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
A Thousand Words - Malay.txt	6512	A Thousand Words 2012 indo.txt	6362	2.936	10.19

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 1:



Gambar 5.22 *Screen-shoot* pengujian IV konfigurasi 1

Konfigurasi 2:

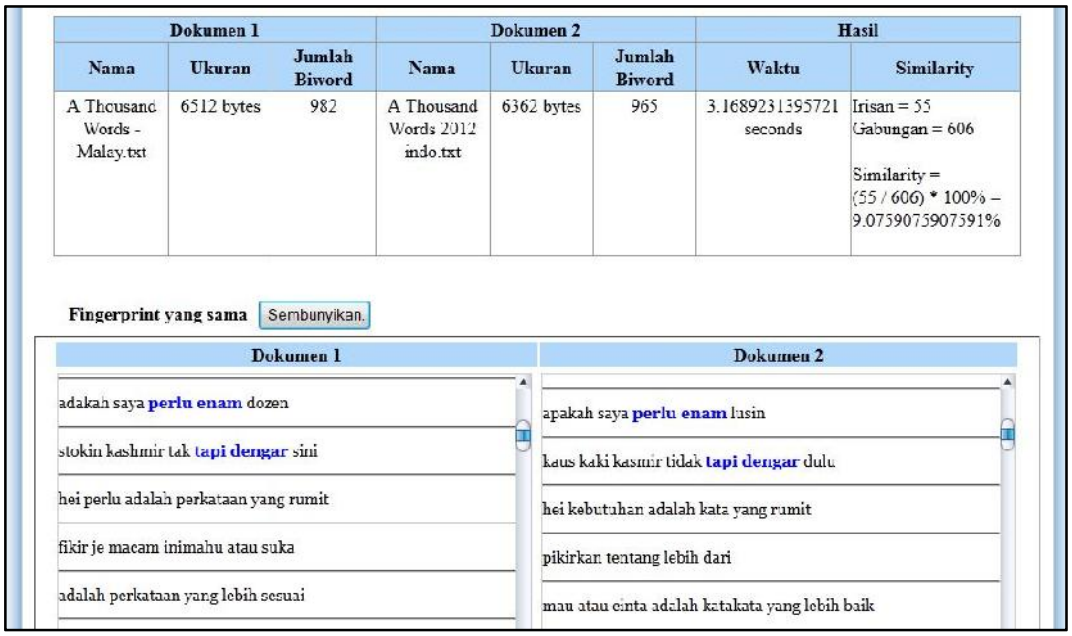
- Bilangan prima: 3
- Ukuran *window*: 4
- Dokumen plaintext: A Thousand Words - Malay.txt dan A Thousand Words 2012 indo.txt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.22 Hasil pengujian IV konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
A Thousand Words - Malay.txt	6512	A Thousand Words 2012 indo.txt	6362	3.168	9.07

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 2:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
A Thousand Words - Malay.txt	6512 bytes	982	A Thousand Words 2012 indo.txt	6362 bytes	965	3.1689231395721 seconds	Irisan = 55 Gabungan = 606 Similarity = (55 / 606) * 100% = 9.075907590759%

Gambar 5.23 *Screen-shoot* pengujian IV konfigurasi 2

Pada pengujian keempat ini telah dilakukan pengujian deengan dua buah konfigurasi yang memiliki bilangan prima dan ukuran *window* yang berbeda-beda. Pada gambar 5.22 dan 5.23 dapat dilihat bahwa nilai *similarity* yang diperoleh dari kedua konfigurasi hanya beberapa persen saja. Aplikasi hanya dapat menemukan sedikit *fingerprint* yang memiliki nilai yang sama. Hal ini disebabkan bahasa yang digunakan pada dokumen memiliki konteks atau tulisan yang berbeda meskipun bahasa tersebut memiliki rumpun yang sama. Sehingga kata-kata yang berbeda penulisannya namun memiliki makna yang sama tidak bisa dideteksi oleh aplikasi.

d. Pengujian V:

Pengujian kelima ini bertujuan untuk mengetahui pengaruh nilai *similarity* dokumen jika dilakukan perubahan terhadap pembentukan token *biword* menggunakan fungsi SHA. Dengan demikian nilai token akan terbentuk sepanjang 40 karakter. Pengujian dilakukan pada beberapa dokumen teks yang dikonfigurasi dengan nilai prima dan ukuran *window* yang berbeda. Pengujian

kelima ini dilakukan dengan 8 buah konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda.

Untuk konfigurasi 1-4, dilakukan pengujian pada dokumen *plaintext* BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt. Sedangkan untuk konfigurasi 5-8 dilakukan pengujian pada dokumen *plaintext* BAB II profil perusahaan.txt dan BAB II PROFIL PT CHEVRON PACIFIC INDONESIA.txt. Konfigurasi dengan ukuran *window* dan nilai bilangan prima yang berbeda-beda yang dipilih secara acak. yaitu:

Konfigurasi 1:

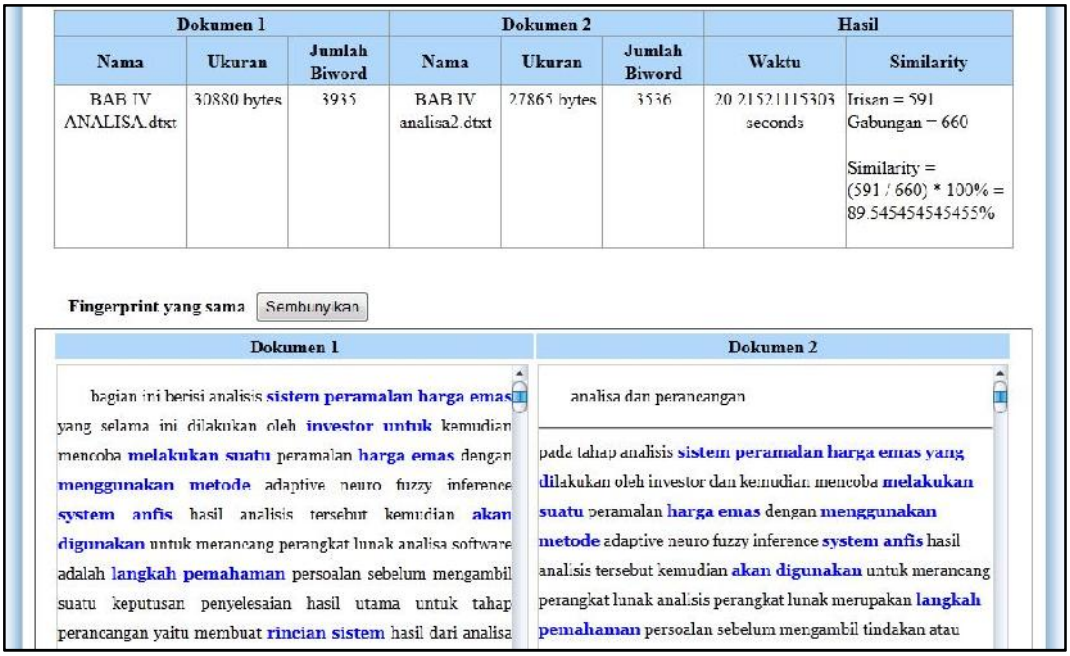
- Bilangan prima: 2
- Ukuran *window*: 8
- Token *biword* : fungsi MD5
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.23 Hasil pengujian V konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	20.2152	89.55

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 1:



Gambar 5.24 *Screen-shoot* pengujian V konfigurasi 1

Konfigurasi 2:

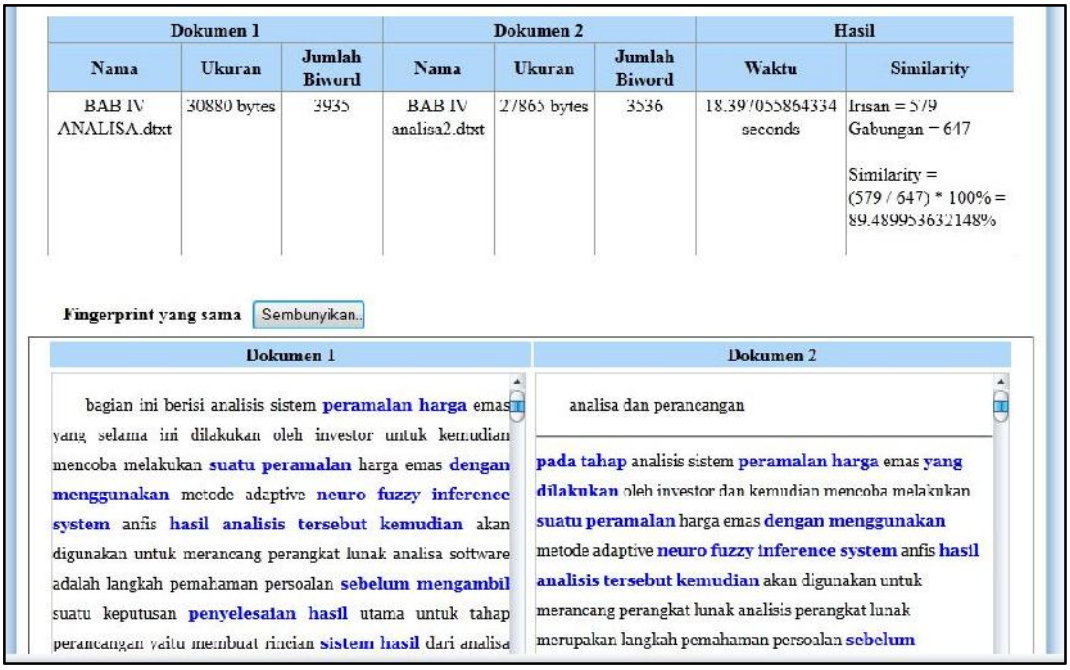
- Bilangan prima: 2
- Ukuran *window*: 8
- Token *biword*: fungsi SHA
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.24 Hasil pengujian V konfigurasi 2

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	18.3971	89.48

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 2:



Gambar 5.25 *Screen-shoot* pengujian V konfigurasi 2

Pada konfigurasi ke-2, telah dilakukan pengujian dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama dengan konfigurasi 1, tetapi telah dilakukan perubahan terhadap kalkulasi nilai token *biword.* Nilai *similarity* yang diperoleh dari konfigurasi 2 mengalami penurunan dari konfigurasi sebelumnya, yaitu 89,55% menjadi 89,48%

Konfigurasi 3:

Pada pengujian konfigurasi ini, dilakukan pengujian pada dokumen teks yang sama dengan mengkonfigurasi nilai bilangan prima dan ukuran *window* yang berbeda-beda yang dipilih secara acak yaitu:

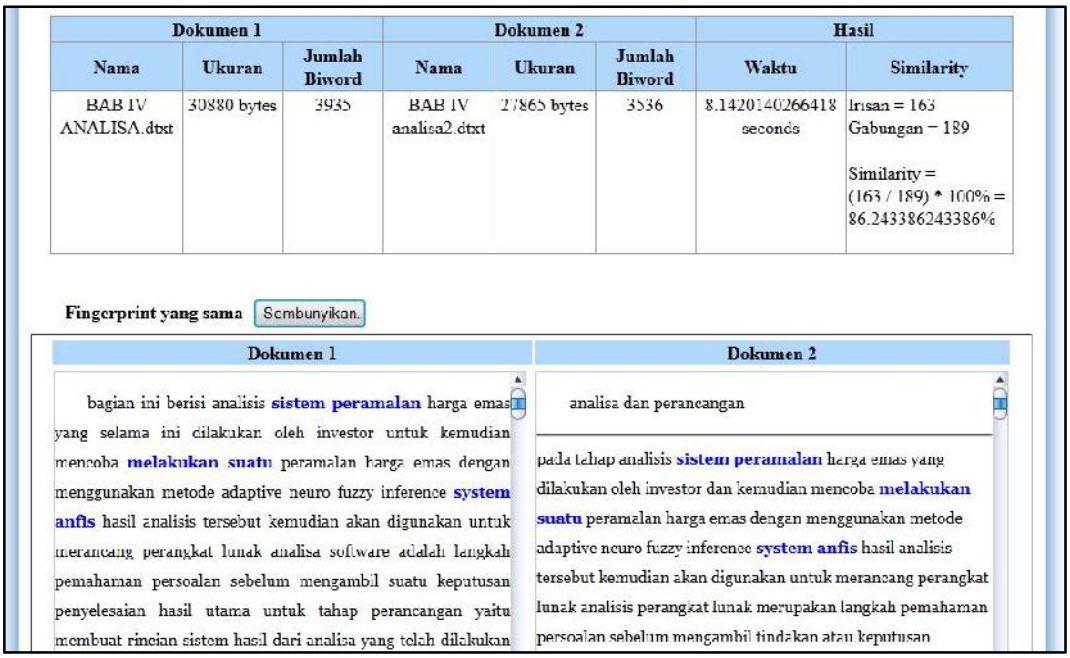
- Bilangan prima: 3
- Ukuran *window*: 32
- Token *biword*: fungsi MD5
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.25 Hasil pengujian V konfigurasi 3

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	8.1420	86.24

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 3:



Gambar 5.26 *Screen-shoot* pengujian V konfigurasi 3

Konfigurasi 4:

Pada pengujian konfigurasi ini, dilakukan pengujian pada dokumen teks yang sama dengan mengkonfigurasi nilai bilangan prima dan ukuran *window* yang berbeda-beda yang dipilih secara acak yaitu:

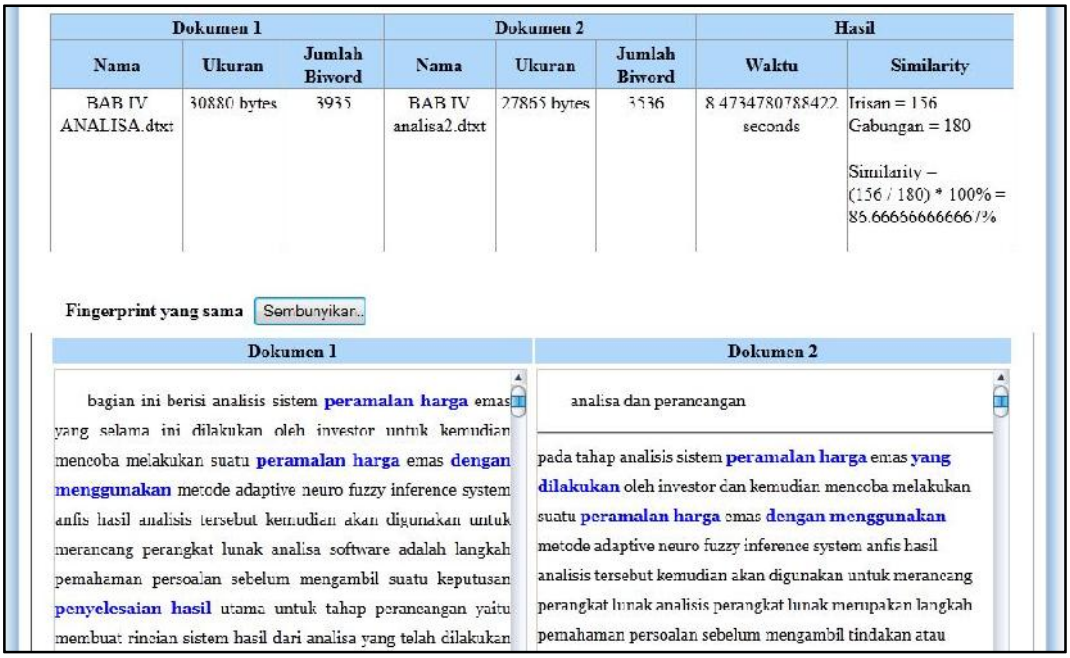
- Bilangan prima: 3
- Ukuran *window*: 32
- Token *biword*: fungsi SHA
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.26 Hasil pengujian V konfigurasi 4

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	8.4735	86.67

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 4:



Gambar 5.27 *Screen-shoot* pengujian V konfigurasi 4

Pada konfigurasi ke-4, telah dilakukan pengujian dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama dengan konfigurasi 3, tetapi telah dilakukan perubahan terhadap kalkulasi nilai token *biword* dari MD5 menjadi SHA. Nilai *similarity* yang diperoleh dari konfigurasi 4 mengalami penurunan dari konfigurasi sebelumnya, yaitu 86,24% menjadi 86,67%

Konfigurasi 5:

Pengujian konfigurasi 5-8 dilakukan pada dokumen *plaintext* yang berbeda dengan pengujian konfigurasi 1-4. Pengujian ini dilakukan dengan mengkonfigurasi nilai bilangan prima dan ukuran *window* yang berbeda-beda yang dipilih secara acak yaitu:

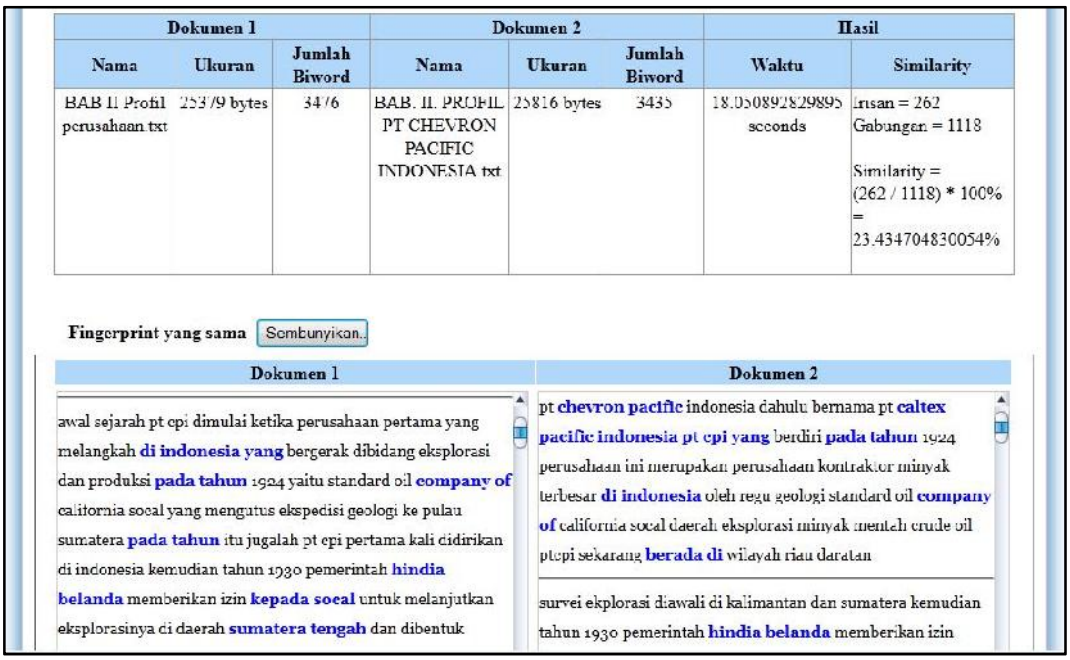
- Bilangan prima: 2
- Ukuran *window*: 8
- Token *biword*: fungsi MD5
- Dokumen plaintext: BAB II Profil perusahaan.txt dan BAB II PROFIL PT CHEVRON PACIFIC INDONESIA.txt.

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.27 Hasil pengujian V konfigurasi 5

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB II Profil perusahaan.txt	25379	Bab II PROFIL PT CHEVRON PACIFIC INDONESIA .txt	25816	18.0509	23.43

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 5:



Gambar 5.28 *Screen-shoot* pengujian V konfigurasi 5

Konfigurasi 6:

Pengujian konfigurasi 6 dilakukan dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama dengan konfigurasi 5, namun telah dilakukan perubahan terhadap nilai token *biword* yaitu:

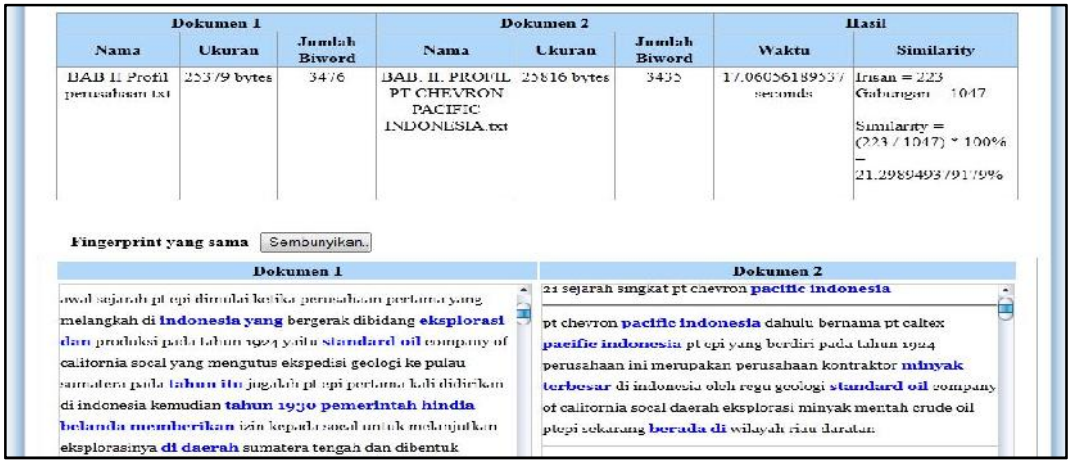
- Bilangan prima: 2
- Ukuran *window*: 8
- Token *biword*: fungsi SHA
- Dokumen plaintext: BAB II Profil perusahaan.txt dan BAB. II. PROFIL PT CHEVRON PACIFIC INDONESIA.txt.

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.28 Hasil pengujian V konfigurasi 6

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB II Profil perusahaan.txt	25379	Bab II PROFIL PT CHEVRON PACIFIC INDONESIA .txt	25816	17.0606	21.30

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 6:



Gambar 5.29 *Screen-shoot* pengujian V konfigurasi 6

Pada konfigurasi 5 dan konfigurasi 6 telah dilakukan pengujian dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama, tetapi nilai token *biword* telah diubah dari MD5 menjadi SHA. Nilai *similarity* yang diperoleh dari konfigurasi 4 mengalami penurunan dari konfigurasi sebelumnya, yaitu 23.43% menjadi 21.30%

Konfigurasi 7:

Pengujian konfigurasi 5-8 dilakukan pada dokumen *plaintext* yang berbeda dengan pengujian konfigurasi 1-4. Pengujian ini dilakukan dengan mengkonfigurasi nilai bilangan prima dan ukuran *window* yang berbeda-beda yang dipilih secara acak yaitu:

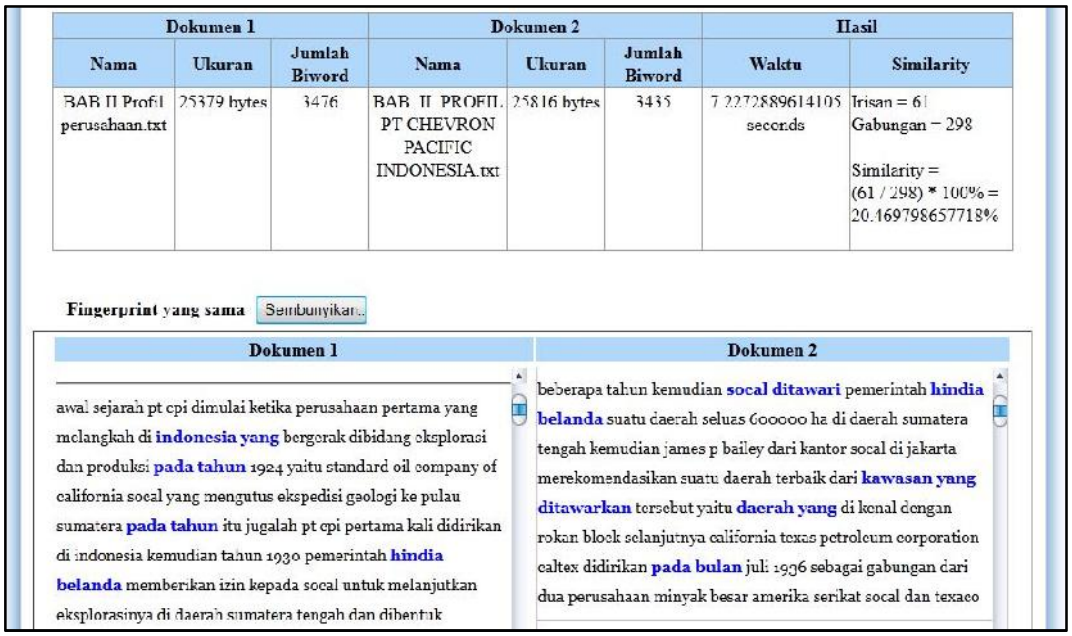
- Bilangan prima: 3
- Ukuran *window*: 32
- Token *biword*: fungsi MD5
- Dokumen plaintext: BAB II Profil perusahaan.txt dan BAB II PROFIL PT CHEVRON PACIFIC INDONESIA.txt.

Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.29 Hasil pengujian V konfigurasi 7

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB II Profil perusahaan.txt	25379	Bab II PROFIL PT CHEVRON PACIFIC INDONESIA .txt	25816	7.2273	20.47

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 7:



Dokumen 1			Dokumen 2			Hasil	
Nama	Ukuran	Jumlah Biword	Nama	Ukuran	Jumlah Biword	Waktu	Similarity
BAB II Profil perusahaan.txt	25379 bytes	3476	BAB. II. PROFIL. PT CHEVRON PACIFIC INDONESIA.txt	25816 bytes	3435	7.2272889614105 seconds	Irisan = 61 Gabungan = 298 Similarity = (61 / 298) * 100% = 20.469798657718%

Fingerprint yang sama | Sembunyikan..

Dokumen 1	Dokumen 2
awal sejarah pt cpi dimulai ketika perusahaan pertama yang melangkah di **indonesia yang** bergerak dibidang eksplorasi dan produksi **pada tahun** 1924 yaitu standard oil company of california socal yang mengutus ekspedisi geologi ke pulau sumatera **pada tahun** itu jugalah pt cpi pertama kali didirikan di indonesia kemudian tahun 1930 pemerintah **hindia belanda** memberikan izin kepada socal untuk melanjutkan eksplorasinya di daerah sumatera tengah dan dibentuk	beberapa tahun kemudian **socal ditawari** pemerintah **hindia belanda** suatu daerah seluas 600000 ha di daerah sumatera tengah kemudian james p bailey dari kantor socal di jakarta merekomendasikan suatu daerah terbaik dari **kawasan yang ditawarkan** tersebut yaitu **daerah yang** di kenal dengan rokan block selanjutnya california texas petroleum corporation caltex didirikan **pada bulan** juli 1936 sebagai gabungan dari dua perusahaan minyak besar amerika serikat socal dan texaco

Gambar 5.30 *Screen-shoot* pengujian V konfigurasi 7

Konfigurasi 8:

Pengujian konfigurasi 8 dilakukan dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama dengan konfigurasi 5, namun telah dilakukan perubahan terhadap nilai token *biword* yaitu:

- Bilangan prima: 3
- Ukuran *window*: 32
- Token *biword*: fungsi SHA
- Dokumen plaintext: BAB II Profil perusahaan.txt dan BAB. II. PROFIL PT CHEVRON PACIFIC INDONESIA.txt.

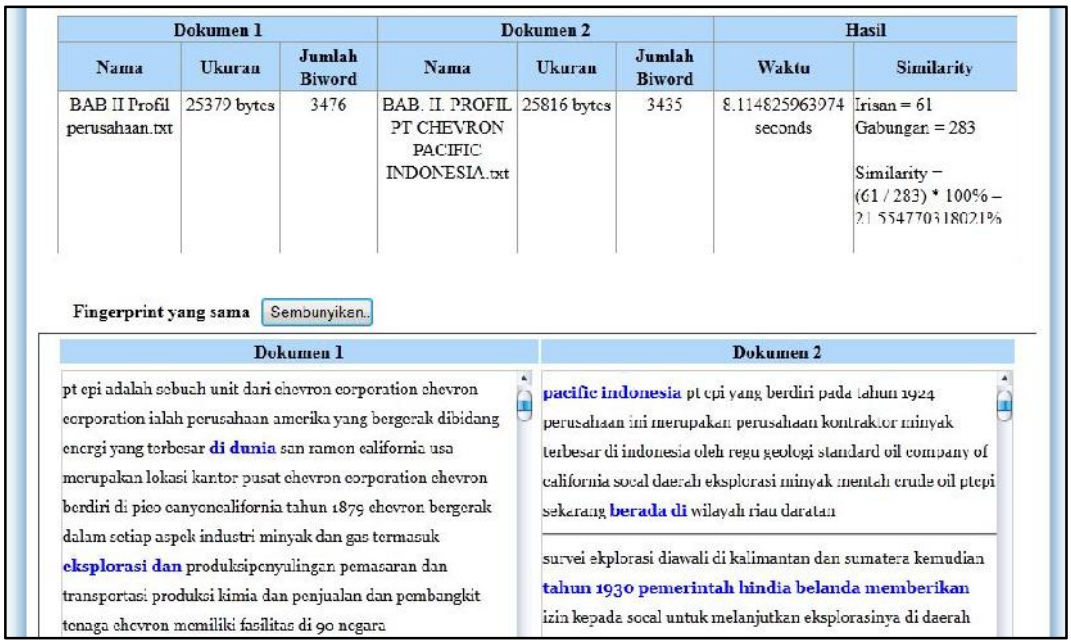
Selanjutnya pengujian dilakukan dengan hasil pengujian sebagai berikut:

Tabel 5.30 Hasil pengujian V konfigurasi 8

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB II Profil perusahaan.txt	25379	Bab II PROFIL PT CHEVRON	25816	8.1148	21.55

		PACIFIC INDONESIA .txt			

Berikut adalah hasil *printscreen* pengujian aplikasi konfigurasi 8:



Gambar 5.31 *Screen-shoot* pengujian V konfigurasi 8

Pada konfigurasi 7 dan konfigurasi 8 telah dilakukan pengujian dengan konfigurasi nilai bilangan prima dan ukuran *window* yang sama, tetapi nilai token *biword* telah diubah dari MD5 menjadi SHA. Nilai *similarity* yang diperoleh dari konfigurasi 8 mengalami kenaikan dari konfigurasi 7, yaitu 20.47% menjadi 21.55%

Setelah dilakukan pengujian V, dapat ditarik sebuah kesimpulan bahwa dengan menggunakan fungsi SHA untuk mendapatkan nilai *fingerprint*, maka nilai *similarity* yang diperoleh mayoritas akan mengalami penurunan. Hal ini disebabkan panjang karakter *string* yang dihasilkan berjumlah 40 karakter. Semakin panjang karakter yang digunakan maka akan kecil kemungkinan untuk mendapatkan nilai *fingerprint.* Sehingga fungsi MD5 akan menjadi nilai yang terbaik dalam membentuk nilai token *biword.*

5.3.2 Hasil Pengujian

Setelah dilakukan beberapa pengujian pada beberapa dokumen dan dengan beberapa konfigurasi bilangan prima dan ukuran *window* yang berbeda-beda, maka dapat di ambil kesimpulan hasil uji coba aplikasi pendeteksi plagiarisme. Hasil pengujian untuk nilai bilangan prima dan ukuran *window* tersebut dapat dilihat pada tabel berikut:

Tabel 5.31 Hasil pengujian secara keseluruhan

No	Dokumen 1		Dokumen 2		Hasil			
	Nama	Size	Nama	Size	Prima	*Window*	Wkt (s)	Sm (%)
Pengujian I								
1	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	**2**	**8**	**20.473**	**89.55**
2	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	5	8	22.228	88.92
3	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	11	8	21.954	88.70
4	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	3	32	9.027	86.24
5	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	3	12	18.019	88.16
6	Bab IV Analisa.dtxt	30880	Bab IV Analisa2	27865	3	20	12.210	87.54
Pengujian II								
7	Contoh Surat.dtxt	1024	Surat Lamaran	1001	**2**	**8**	**0.1656**	**69.70**
8	Contoh Surat.dtxt	1024	Surat Lamaran	1001	5	8	0.1442	56.41
9	Contoh Surat.dtxt	1024	Surat Lamaran	1001	11	8	0.1269	54.05

10	Contoh Surat.dtxt	1024	Surat Lamaran	1001	3	32	0.1111	66.67
11	Contoh Surat.dtxt	1024	Surat Lamaran	1001	3	12	0.1356	68.00
12	Contoh Surat.dtxt	1024	Surat Lamaran	1001	3	20	0.1097	63.64
Pengujian III								
13	Surat Lamaran Kerja.dtxt	1006	Surat lamaran 2	1006	**2**	**8**	**0.1414**	**96.43**
14	Surat Lamaran Kerja.dtxt	1006	Surat lamaran 2	1006	5	8	0.1376	90.32
15	Surat Lamaran Kerja.dtxt	1006	Surat lamaran 2	1006	3	12	0.1258	90.91
16	Surat Lamaran Kerja.dtxt	1006	Surat lamaran 2	1006	3	8	0.1569	90.32
Pengujian IV								
17	A Thousand Words - Malay.txt	6512	A Thousand Words 2012 indo.txt	6362	2	8	2.936	10.19
18	A Thousand Words - Malay.txt	6512	A Thousand Words 2012 indo.txt	6362	3	4	3.168	9.07

Pengujian V									
No	Nilai *biword*	Dokumen 1		Dokumen 2		Hasil			
		Nama	Size	Nama	Size	Prima	*Window*	Wkt (s)	Sm (%)
19	MD5	Bab IV Analisa	30880	Bab IV Analisa2	27865	2	8	20.215	89.55
20	SHA	Bab IV Analisa	30880	Bab IV Analisa2	27865	2	8	18.397	89.49
21	MD5	Bab IV Analisa	30880	Bab IV Analisa2	27865	3	32	8.47.3	86.67
22	SHA	Bab IV Analisa	30880	Bab IV Analisa2	27865	3	32	8.1420	86.24
22	MD5	Bab II Profil perusahaan.dtxt	25379	Bab II Profil PT Chevron pacific Indonesia .dtxt	25816	2	8	18.0509	23.43
24	SHA	Bab II Profil perusahaan.dtxt	25379	Bab II Profil PT Chevron pacific Indonesia .dtxt	25816	2	8	17.0609	21.30
25	MD5	Bab II Profil perusahaan.dtxt	25379	Bab II Profil PT Chevron pacific Indonesia .dtxt	25816	3	32	7.2273	20.47

26	SHA	Bab II Profil perusahaan.dtxt	25379	Bab II Profil PT Chevron pacific Indonesia .dtxt	25816	3	32	8.1148	21.55

5.3.3 Kesimpulan Pengujian

Pada tabel 5.31 di atas, dapat disimpulkan konfigurasi yang paling baik dalam mendeteksi penjiplakan dokumen dan menghasilkan nilai *similarity* tertinggi dari semua pengujian, yaitu:

Pengujian I dengan konfigurasi 1:

- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: BAB IV ANALISA.dtxt dan BAB IV analisa2.dtxt

Tabel 5.32 Hasil pengujian I konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
BAB IV ANALISA.dtxt	30880	Bab IV Analisa2.dtxt	27865	20.473	89.55

Pengujian II dengan konfigurasi 1:

- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: Contoh Surat.dtxt dan Surat Lamaran.dtxt

Tabel 5.33 Hasil pengujian II konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu	*Similarity*
Contoh Surat	1024	Surat Lamaran	1001	0.1656	69.70

Pengujian III konfigurasi 1:

- Bilangan prima: 2
- Ukuran *window*: 8
- Dokumen plaintext: Surat Lamaran Kerja.dtxt dan surat lamaran2

Tabel 5.34 Hasil pengujian III konfigurasi 1

Dokumen 1		Dokumen 2		Hasil	
Nama	Ukuran	Nama	Ukuran	Waktu(s)	*Similarity*(%)
Surat Lamaran Kerja	1006	Surat Lamaran2	1006	0.1414	96.43

Pada ketiga tabel hasil pengujian diatas dengan pengujian yang berbeda-beda dapat ditarik sebuah kesimpulan bahwa bilangan prima yang terbaik adalah bilangan prima terkecil yang bernilai 2, sedangkan untuk ukuran *window* terbaik mempunyai nilai 8. Dengan demikian, konfigurasi diatas dianjurkan untuk mendeteksi penjiplakan agar hasil yang di dapat lebih maksimal dengan nilai *similarity* tertinggi. Sedangkan pada pengujian IV dapat dilihat bahwa fungsi MD5 dapat memberikan nilai *similarity* terbaik dibandingkan fungsi SHA. Dengan demikian fungsi MD5 merupakan fungsi terbaik dalam perhitungan nilai token *biword.*

Berdasarkan hipotesa yang diberikan sebelumnya, yaitu “semakin kecil nilai bilangan prima yang digunakan, maka nilai *similarity* akan meningkat, dan semakin tinggi ukuran *window* yang digunakan juga akan meningkatkan hasil *similarity”*. Namun setelah dilakukan beberapa pengujian, ternyata hipotesa tersebut bernilai salah. Pengujian membuktikan bahwa konfigurasi bilangan prima dengan nilai terkecil dapat menghasilkan nilai *similarity* tertinggi. Pada tabel 5.31 dapat dilihat bahwa angka 2 merupakan bilangan prima terkecil yang memberikan nilai *similarity* yang tinggi. Pada pengujian 1 kongfigurasi 1 diperoleh nilai *similarity* tertinggi 89,55%. Pada pengujian II konfigurasi 2 diperoleh nilai *similarity* tertinggi 69,70%, dan pada pengujian III juga diperoleh nilai *similarity* tertinggi 96,43%. Namun, ukuran *window* terbesar tidak menghasilkan nilai

similarity yang tinggi juga. Hal tersebut juga dapat dilihat pada tabel 5.31, bahwa nilai 32 merupakan ukuran *window* terpanjang tidak bisa memberikan nilai *similarity* terbaik.

BAB VI
PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari tahap-tahap penelitian Tugas Akhir ini adalah sebagai berikut:

a. Aplikasi ini dapat mendeteksi persentase kesamaan antar dokumen menggunakan algoritma *winnowing* dengan pendekatan *biword.*
b. Pada *ouput* hasil pengujian, aplikasi ini dapat menampilkan teks yang memiliki nilai yang sama antar dokumen berdasarkan nilai *fingerprint* yang dihasilkan.
c. Dalam proses deteksi penjiplakan dokumen, bilangan prima terkecil yang digunakan dapat menghasilkan nilai kemiripan yang tinggi antar dokumen.
d. Dalam proses deteksi penjiplakan dokumen, ukuran *window* tertinggi yang digunakan tidak dapat menghasilkan nilai kemiripan yang tinggi antar dokumen. Hal ini bertolak belakang dengan hipotesa yang diberikan pada tahap analisa.
e. Aplikasi dapat mendeteksi penjiplakan meskipun telah dilakukan perubahan posisi teks (berbentuk *biword)* dalam teks dokumen.
f. Dari pengujian yang dilakukan, fungsi MD5 pada pembentukan nilai *hash* dapat memberikan nilai *similarity* terbaik dibandingkan fungsi SHA. Dengan demikian fungsi MD5 merupakan fungsi terbaik dalam perhitungan nilai token *biword.*
g. Dari pengujian yang dilakukan, pendeteksian pada dokumen teks yang memiliki bahasa yang serumpun memberikan nilai *similarity* yang rendah, karena teks yang terdapat pada dokumen memiliki penulisan kata yang berbeda sehingga menghasilkan nilai *fingerprint* yang berbeda pula.

6.2 Saran

Untuk pengembangan aplikasi ini di masa yang akan datang, maka diberikan beberapa saran sebagai berikut:

a. Aplikasi pendeteksi penjiplakan dokumen ini akan lebih baik apabila dikembangkan dengan penambahan metode *clustering*. Yaitu membandingkan satu dokumen dengan banyak dokumen.
b. Aplikasi ini dapat dikembangkan dengan membandingkan proses pembentukan kata dengan konsep *triword* dan *quadword.*

DAFTAR PUSTAKA

Goenawan , Willy , Ronald Augustinus, Krisantus Sembiring. *Penerapan Algoritma Edit Distance Pada Pendeteksian Praktik Plagiat.* Bandung : Laboratorium Ilmu danRekayasa Komputer Departemen Teknik Informatika, Institut Teknologi Bandung.
Diakses, Mei, 17, 2012

Kadek, Versi Yana Yoga. *Pengembangan Aplikasi Pendeteksi Plagiarisme Pada Dokumen Tteks Menggunakan Algoritma Rabin-Karp*. singaraja: Jurusan Pendidikan Teknik Informatika, Fakultas Teknik Kejuruan Universitas Pendidikan Ganesha Singaraja. 2012.
Diakses Mei, 08, 2012

Kok, Chow Kent, Naomie Salim. *Features Based Text Similarity Detection*. Malaysia: Faculty of Computer Science and Information Systems, University Teknologi Malaysia. 2010

Kusmawan, Putu Yuwono, Umi Laili Yuhana, Diana Purwitasari. *Aplikasi Pendeteksi Penjiplakan Pada File Teks Dengan Algoritma Winnowing.* Surabaya: Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh November Surabaya. 2010.
Available:
http://digilib.its.ac.id/public/ITS-Undergraduate-10278-Paper.pdf , diakses Mei, 17, 2012

Kurniawati, Ana, Wicaksana, I Wayan Simri. *Perbandingan Pendekatan Deteksi Plagiarism Dokumen Dalam Bahasa Inggris*. Jakarta: Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma. 2008.
Available:
http://research.mercubuana.ac.id/proceeding/OSSOC_26.pdf, diakses Mei, 25, 2012

Manning, Christopher D, Prabhakar Raghavan dan Hinrich Schütze. *An Introduction to Information Retrieval*. England: Cambridge University Press. 2009.

Munir, Rinaldi. *Algoritma Dan Pemograman*. Bandung: Informatika Bandung. 2007

Nugroho, Eko. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp*. Malang: Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijaya. 2011.
Available:
http://blog.ub.ac.id/ecoorner/files/2011/03/Bab12345.pdf, diakses Mei, 17, 2012

Purwitasari, Diana, Putu Yuwono Kusmawan, Umi Laili Yuhana. *Deteksi Keberadaan Kalimat Sama Sebagai Indikasi Penjiplakan Dengan Algoritma Hashing Berbasis N-gram*. Surabaya: Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh November Surabaya. 2011.
Available:
http://kursor.trunojoyo.ac.id/wp-content/uploads/2012/03/vol6_no1_p5.pdf , diakses Mei, 17, 2012

Schleimer, Saul, Daniel S. Wilkerson, dan Alex Aiken. *Winnowing: Local Algorithms for Document Fingerprinting.* San Diego: In Proceedings of the ACM SIGMOD International Conference On Management Of Data. 2003
Available:
http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf, diakses Mei, 17, 2012

Steven. *Perancangan Program Aplikasi Pendeteksian Plagiarisme Dokumen Berbasis Teks Menggunakan Algoritma Rabin-Karp.* Jakarta: Program Ganda Teknik Informatika Dan Matematika, Universitas Bina Nusantara. 2009.
Available:
http://thesis.binus.ac.id/Asli/Cover/2009-1-00394-MTIF%20Cover.pdf,
http://thesis.binus.ac.id/Asli/Bab1/2009-1-00394-MTIF%20Bab%201.pdf
http://thesis.binus.ac.id/Doc/Bab2/2009-1-00394-MTIF%20Bab%202.pdf,
http: //thesis.binus.ac.id/Asli/Bab3/2009-1-00394-MTIF%20Bab%203.pdf,
http://thesis.binus.ac.id/Asli/Pustaka/2009-1-00394-MTIF%20Pustaka.pdf,
diakses Mei, 25, 2012