DETEKSI PLAGIARISME PADA DOKUMEN TEKS BAHASA INDONESIA MENGGUNAKAN ALGORITMA *WINNOWING* DENGAN *STEMMING*

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh:

MILANI WINANGGA

NIM. 0910680026

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

MALANG

2014

LEMBAR PERSETUJUAN

DETEKSI PLAGIARISME PADA DOKUMEN TEKS BAHASA INDONESIA MENGGUNAKAN ALGORITMA *WINNOWING* DENGAN *STEMMING*

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh:

MILANI WINANGGA

NIM. 0910680026

Telah diperiksa dan disetujui oleh :

Pembimbing I	**Pembimbing II**
Drs. Marji, MT	**Drs. Achmad Ridok, M. Kom**
NIP. 19670801 199203 1 001	**NIP. 19680825 199403 1 002**

LEMBAR PENGESAHAN

DETEKSI PLAGIARISME PADA DOKUMEN TEKS BAHASA INDONESIA MENGGUNAKAN ALGORITMA *WINNOWING* DENGAN *STEMMING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh:

MILANI WINANGGA

NIM. 0910680026

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 11 Juni 2014

Penguji I	**Penguji II**
Nurul Hidayat, S.Pd., MSc.	**Ahmad Afif Supianto, SSi., M.Kom**
NIP. 19680430 200212 1 001	**NIK. 820623 16 1 1 0425**

Penguji III

Rekyan Regasari Mardi Putri, ST., MT.
NIK. 77041406120253

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP.19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 Juni 2014

Mahasiswa,

Milani Winangga

NIM. 0910680026

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Deteksi Plagiarisme pada Dokumen Teks Bahasa Indonesia menggunakan Algoritma *Winnowing* dengan *Stemming*". Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Informatika/Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer (PTIIK) Universitas Brawijaya Malang.

Saya menyadari bahwa Tugas Akhir ini dapat terselesaikan atas bantuan, petunjuk, dan bimbingan dari berbagai pihak yang telah membantu proses penyelesaiannya. Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan bantuan baik lahir maupun batin selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Kedua Orang Tua penulis, kakak dan adik tersayang, beserta seluruh keluarga besar atas segala nasihat, perhatian, dan kesabarannya dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan do'a demi terselesaikannya tugas akhir ini.
2. Drs. Marji, MT dan Drs. Achmad Ridok, M. Kom selaku dosen pembimbing tugas akhir penulis yang telah memberikan bimbingan dan arahan untuk kesempurnaan penulisan Tugas Akhir.
3. Seluruh Dosen dan Karyawan Informatika/Ilmu Komputer PTIIK Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
4. Dita Prima Oktasa, Mbak Anik Fitriyah, Arianty Anggraini, Fauziah Mayasari Iskandar, Anisa Aini Arifin, Austin Buya Oryza, Mamluatul Hani'ah, Novelia Kharisma, Winda Ayu Irianto, Ari Agustina, Hanifa Vidya Rizanti, Dian Arisandy, Adestiana Rahmawati, Krestian Apriliyanto, Ervin Yohanes, Nina Amalia Dewi, Sufia Adha Putri, Hendro Pramudyo Saputro, Aulia Meitika K., Tika Rahmadian, Deppy Rangga M., Astri Tika P. dan

Rindang Mazdarani Hakiki atas dukungan dan semangat yang diberikan selama menyelesaikan skripsi ini.

5. Teman-teman seperjuangan Angkatan 2009 Teknik Informatika Universitas Brawijaya, terimakasih atas segala bantuan yang diberikan selama menempuh studi di Informatika/Ilmu Komputer PTIIK Universitas Brawijaya.
6. Saul Schleimer, Arfian Hidayat, dan Mudafiq Riyan Pratama atas kesediaannya berbagi ilmu dan memberikan saran demi kelancaran penulis dalam menyelesaikan aplikasi Tugas Akhir ini.
7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikan-nya skripsi ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat. Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Saran dan kritik membangun dari para pembaca senantiasa penulis harapkan guna perbaikan bagi tugas akhir selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Informatika/Ilmu Komputer Universitas Brawijaya dan menjadi acuan dalam penelitian selanjutnya.

Malang, Juni 2014

Penulis

ABSTRAK

Milani Winangga. 2014: Deteksi Plagiarisme pada Dokumen Teks Bahasa Indonesia menggunakan Algoritma *Winnowing* dengan *Stemming*. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.
Dosen Pembimbing: Drs. Marji, MT. dan Drs. Achmad Ridok, M.Kom.

Perkembangan teknologi saat ini, memperbesar peluang seseorang untuk melakukan tindakan plagiarisme. Data digital sangat mudah untuk dijiplak, terutama dokumen digital. Kasus ini sering terjadi di kalangan mahasiswa dalam pengerjaan tugas. Namun demikian jarang ditemukan penjiplakan seluruh isi dokumen, terkadang mahasiswa hanya mengambil sebagian data, kemudian memodifikasinya dengan perubahan posisi kalimat atau penghilangan beberapa kata, sehingga sulit dalam mendeteksinya. Untuk memperkecil tindakan penjiplakan, diperlukan suatu sistem yang mampu mendeteksi penjiplakan tanpa mengabaikan perubahan posisi kalimat, pemotongan kata ataupun perubahan kata aktif menjadi kata pasif atau sebaliknya. Penelitian ini membuat sistem deteksi plagiarisme pada dokumen teks bahasa Indonesia menggunakan algoritma *Winnowing* dengan *stemming*. Algoritma *winnowing* akan mencari *fingerprint* dari tiap dokumen yang akan dibandingkan. Pada *preprocessing* terdapat penambahan *stemming* yang bertujuan untuk menghilangkan imbuhan kata. *Similarity* antar dokumen didasarkan pada kecocokan *substring* antar *fingerprint*, yang akan di hitung menggunakan prinsip *Jaccard Coefficient*. Nilai 5 untuk *k-gram*, nilai 25 untuk *threshold* dan 11 untuk *basis* sebagai hasil dari pengujian nilai parameter akan digunakan pada pengujian *similarity*, waktu eksekusi, dan *error* terhadap sistem dengan *stemming* dan tanpa *stemming*. Hasil pengujian menunjukkan bahwa sistem dengan *stemming* cenderung menghasilkan prosentase *similarity* yang kurang baik dibandingkan dengan sistem tanpa *stemming*. Lama waktu eksekusi pada pengujian sistem dengan *stemming* dan tanpa *stemming* tidak jauh berbeda. Sistem dengan *stemming* cenderung membutuhkan waktu lebih lama dibandingkan dengan sistem tanpa *stemming*. Pada pengujian *error*, terlihat bahwa sistem dengan *stemming* menghasilkan prosentase *error* yang lebih tinggi dibandingkan sistem tanpa *stemming*.

Kata kunci : plagiarisme, *Winnowing*, *fingerprint* dokumen, *Stemming*, *Jaccard Coefficient*

ABSTRACT

Milani Winangga. 2014 : ***Plagiarism Detection on Indonesian Text Document using Winnowing Algorithm with Stemming. Minor Thesis Informatics Engineering Program, Informatic Technology and Computer Science Program, University of Brawijaya.***
***Advisor*: Drs. Marji, MT. *and* Drs. Achmad Ridok, M.Kom.**

Nowadays, the development of technology increases the opportunity of people to perform plagiarism. Digital data is very easy to be duplicated, especially digital documents. This case has happened between students in performing their tasks. But rarely found plagiarized entire document, sometimes students just take a portion of data, and then change the position of the sentence or remove a few words, that's making it difficult to detect. To minimize plagiarism, we need a system that can detect plagiarism without ignoring changes the position of the sentences, cutting words or change the active word to passive or passive to active. This study makes plagiarism detection on Indonesian text document using Winnowing algorithm with stemming. Winnowing algorithm will look for the fingerprint of each document to be compared. In the preprocessing, we will add stemming which eliminate the affixes of the words. Similarity between documents based on a substring match between the fingerprints which will be calculated using the principle of Jaccard Coefficient. The value of 5 for k-grams, 25 for threshold, and 11 for the basis as a result on the testing before are being used in the similarity testing, execution time testing, and the error testing on system with stemming and without stemming. The results show that the system with stemming tends to produce a percentage of similarity which not better than the system without stemming. The execution time on system with and without stemming is not much different. Systems with stemming take a longer time than the system without stemming. In error testing, it appears that the system with stemming produce higher percentage than the system without stemming.

Keywords : plagiarism, Winnowing, fingerprint* dokumen, *Stemming, Jaccard Coefficient

DAFTAR ISI

LEMBAR PERSETUJUAN ii
LEMBAR PENGESAHAN iii
PERNYATAAN ORISINALITAS SKRIPSI iv
KATA PENGANTAR v
ABSTRAK vii
ABSTRACT viii
DAFTAR ISI ix
DAFTAR GAMBAR xii
DAFTAR TABEL xiv
BAB I PENDAHULUAN 1
1.1. Latar belakang 1
1.2. Rumusan Masalah 3
1.3. Batasan Masalah 3
1.4. Tujuan 3
1.5. Manfaat 4
1.6. Sistematika Pembahasan 4
BAB II KAJIAN PUSTAKA DAN DASAR TEORI 6
2.1 Kajian Pustaka 6
2.2 Plagiarisme 7
2.3 *Text Mining* 10
2.4 Algoritma *Stemming* Arifin Setiono 11
2.5 Algoritma *Winnowing* 13
2.6 Pengukuran Nilai *Similarity* 17
BAB III METODOLOGI DAN PERANCANGAN 19
3.1 Studi Literatur 19
3.2 Pengumpulan Data 20
3.3 Analisa dan Perancangan Sistem 20
3.3.1 Perancangan Sistem 20
3.3.2 Perancangan Proses 21

3.3.2.1 Preprocessing 22
3.3.2.2 Algoritma Winnowing 25
3.3.2.3 Perhitungan Similarity 27
3.3.3 Perhitungan Manual Sistem 30
3.3.4 Perancangan Antar Muka Sistem Deteksi Plagiarisme Dokumen Teks 34
3.4 Implementasi 35
3.5 Pengujian dan Analisa Sistem 35
3.5.1. Perancangan Pengujian 36
3.5.1.1. Bahan Pengujian 36
3.5.1.2. Tujuan Pengujian 36
3.5.1.3. Perancangan Tabel Hasil Uji Coba 37
3.5.1.4. Perancangan Dokumen Uji dan Dokumen Latih 40
3.6 Penarikan Kesimpulan 42
BAB IV IMPLEMENTASI 43
4.1. Spesifikasi Sistem 43
4.1.1. Spesifikasi Perangkat Keras 43
4.1.2. Spesifikasi Perangkat Lunak 43
4.2. Implementasi Proses Sistem 44
4.2.1. Proses *Preprocessing* 44
4.2.1.1. Proses Casefolding dan Tokenizing 44
4.2.1.2. Proses Filtering 45
4.2.2. Proses Algoritma *Winnowing* 45
4.2.2.1. Proses Pemotongan K-gram 45
4.2.2.2. Proses Hashing 46
4.2.2.3. Proses Windowing 46
4.2.2.4. Proses Fingerprinting 47
4.2.3. Proses Perhitungan Nilai *Similarity* 48
4.3. Implementasi Antar Muka 49
4.3.1. Tampilan Halaman Awal 49
4.3.2. Tampilan Halaman Multiple Documents 50
4.3.3. Tampilan Halaman *Plagiarism Detection* untuk *Multiple Documents* 51
4.3.4. Tampilan Halaman Two Documents 51

BAB V PENGUJIAN DAN ANALISIS 53
5.1. Pengujian nilai *k-gram*, *threshold*, dan *basis*. 53
5.2. Pengujian *Similarity*, Waktu Eksekusi, dan *Error* Sistem 58
5.3. Analisis Hasil 62
BAB VI PENUTUP 66
6.1. Kesimpulan 66
6.2. Saran 67
DAFTAR PUSTAKA DP-1

DAFTAR GAMBAR

Gambar 2. 1 Metode Pendeteksian Plagiarisme 9
Gambar 3.1 Diagram Alir Penelitian 19
Gambar 3.2 Arsitektur Sistem 20
Gambar 3.3 Diagram Alir Sistem 22
Gambar 3.4 Diagram Alir *Preprocessing* 22
Gambar 3.5 Diagram Alir *Casefolding* 23
Gambar 3.6 Diagram Alir *Tokenizing* 23
Gambar 3.7 Diagram Alir *Filtering* 24
Gambar 3.8 Diagram Alir Pemotongan *k-gram* 25
Gambar 3.9 Diagram Alir *Hashing* 25
Gambar 3.10 Diagram Alir *Window* 26
Gambar 3.11 Diagram Alir *Fingerprint* 26
Gambar 3.12 Diagram Alir Perhitungan *Similarity* 28
Gambar 3.13 Diagram Alir Perhitungan *Similarity* (Lanjutan) 29
Gambar 3.14 Ilustrasi Pengecekan *Similarity* dengan *Subtring* = 8 33
Gambar 3.15 Ilustrasi Pengecekan *Similarity* dengan *Substring* = 7 33
Gambar 3. 16 Rancangan Antar Muka Sistem 34
Gambar 4.1 Implementasi *Casefolding* dan *Tokenizing* 44
Gambar 4.2 Implementasi *Filtering* 45
Gambar 4.3 Implementasi Proses Pemotongan *K-gram* 46
Gambar 4.4 Implementasi Proses *Hashing* 46
Gambar 4.5 Implementasi Proses Pemotongan *Window* 47
Gambar 4.6 Implementasi Proses *Fingerprinting* 48
Gambar 4.7 Implementasi Proses Perhitungan Nilai *Similarity* 49
Gambar 4. 8 Tampilan Halaman Awal 50
Gambar 4. 9 Tampilan Halaman Multiple Documets 50
Gambar 4. 10 Tampilan *Form* Isian *Plagiarism Detection* 51
Gambar 4. 11 Tampilan Hasil *Plagiarism Detection* 51
Gambar 4. 12 Tampilan *Two Documents* 52

Gambar 4. 13 Tampilan Hasil *Plagiarism Detection* pada *Two Documents* 52
Gambar 5.1 Grafik Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Similarity* 54
Gambar 5.2 Grafik Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Error* 55
Gambar 5.3 Grafik Hasil Pengujian Nilai *threshold* terhadap Prosentase *Similarity* 56
Gambar 5.4 Grafik Hasil Pengujian Nilai *threshold* terhadap Prosentase *Error*.. 56
Gambar 5. 5 Grafik Hasil Pengujian Nilai *basis* terhadap Prosentase *Similarity*. 57
Gambar 5. 6 Grafik Hasil Pengujian Nilai *basis* terhadap Prosentase *Error*..... 58
Gambar 5.7 Grafik Hasil Pengujian *Similarity* 59
Gambar 5.8 Grafik Hasil Pengujian Waktu Eksekusi 60
Gambar 5.9 Grafik Hasil Pengujian *Error* 60

DAFTAR TABEL

Tabel 3.1 Skenario Pengujian Nilai *K-gram*, *Basis*, dan *Threshold* 38
Tabel 3.2 Pengujian Nilai *K-gram*, *Basis*, dan *Threshold* 38
Tabel 3.3 Pengujian *Similarity*, Waktu Eksekusi, dan *Error* Sistem 39
Tabel 5.1 Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Similarity* 54
Tabel 5.2 Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Error* 54
Tabel 5.3 Hasil Pengujian Nilai *threshold* terhadap Prosentase *Similarity* 55
Tabel 5.4 Hasil Pengujian Nilai *threshold* terhadap Prosentase *Error* 55
Tabel 5.5 Hasil Pengujian Nilai *basis* terhadap Prosentase *Similarity* 57
Tabel 5.6 Hasil Pengujian Nilai *basis* terhadap Prosentase *Error* 57
Tabel 5.7 Hasil Pengujian *Similarity*, Waktu eksekusi, dan *Error* Sistem 59

BAB I
PENDAHULUAN

1.1. Latar belakang

Peran teknologi dalam memperoleh informasi di masa kini, memperbesar peluang seseorang untuk melakukan tindakan plagiarisme. Data digital merupakan data yang sangat mudah untuk dijiplak, terutama dokumen digital. Hal ini sering terjadi di kalangan mahasiswa dalam pengerjaan tugas. Namun jarang ditemukan penjiplakan seluruh isi dokumen, terkadang mahasiswa hanya mengambil sebagian data, kemudian memodifikasinya dengan perubahan posisi kalimat, penghilangan beberapa kata, perubahan kata kerja pasif menjadi aktif ataupun sebaliknya, sehingga sulit dalam mendeteksinya. Oleh sebab itu diperlukan sebuah sistem yang mampu mendeteksi tindakan penjiplakan tanpa mengabaikan perubahan tersebut.

Dalam mendeteksi plagiarisme terdapat tiga metode, yaitu metode pencarian kata kunci, perbandingan teks lengkap dan dokumen *fingerprinting*. Pada metode pencarian kata kunci hanya akan mencari kata-kata yang sering muncul dalam dokumen untuk kemudian akan dibandingkan dengan dokumen lain, sehingga metode ini kurang cocok untuk menangani masalah di atas. Sedangkan metode perbandingan teks secara lengkap akan membandingkan seluruh isi teks. Metode ini cocok dalam menangani masalah penjiplakan dengan pengubahan posisi kalimat dan pemotongan beberapa kata. Namun demikian metode ini membutuhkan waktu yang sangat lama dalam mendeteksi adanya plagiarisme dibandingkan dengan metode dokumen *fingerprinting*. Oleh sebab itu, metode dokumen *fingerprinting* merupakan metode tercepat dan paling cocok dalam menangani masalah perubahan posisi kalimat dan pemotongan beberapa kata.

Terdapat tiga algoritma dalam metode dokumen *fingerprinting*, yaitu algoritma *Manber*, *Rabin Karp*, dan *Winnowing*. Penelitian pertama yang menerapkan metode dokumen *fingerprinting* pada koleksi dokumen dikembangkan oleh Richard M. Karp dan Michael O Rabin yaitu algoritma *Karp-Rabin*. Algoritma *Karp-Rabin* akan membandingkan seluruh pasang *k-gram* dalam dokumen. Algortima tersebut relatif lebih lama dibandingkan dengan

algoritma *Manber* dan *Winnowing*. Kemudian Udi Manber menerapkan *Karp-Rabin string matching* untuk mendeteksi kesamaan *file* pada *file system* dengan mendapatkan *fingerprint* dari setiap nilai *hash* hasil dari *0 mod p*, dengan *p* yang telah ditentukan [UDI-94]. Metode tersebut menciptakan *gap* yang terlalu jauh antara *fingerprint* satu dengan yang lainnya dan kecocokan di dalam *gap* tidak dapat terdeteksi [SCH-03]. Sedangkan pada algoritma *Winnowing* akan mengambil nilai *k-gram* terkecil sebagai *fingerprint* dokumen. Dengan mengambil nilai terkecil maka akan mengurangi *gap* antar *fingerprint*, dan jumlah *fingerprint* yang dihasilkan akan sesuai dengan besar dokumen yang bersangkutan [SCH-03].

Pada penelitian terdahulu oleh Diana Purwitasari dari Institut Teknologi Sepuluh Nopember Surabaya [PUR-11], algoritma *Winnowing* telah digunakan untuk mendeteksi kesamaan kalimat antar *file* teks atau *problem common subsequence*. Algoritma tersebut digunakan untuk mencari *fingerprint* dokumen dengan mengubah rangkaian *n-gram* dari teks menjadi kumpulan nilai-nilai *hash*. Penelitian tersebut melakukan uji coba untuk melihat kemampuan deteksi kalimat sama dengan perubahan parameter nilai *n* dari *n-gram*, bilangan prima *b* sebagai *basis hashing*, ukuran *window w* dan nilai ambang batas penentuan penjiplakan. Berdasarkan uji coba yang telah dilakukan terlihat bahwa konfigurasi nilai yang kurang tepat menyebabkan deteksi adanya kalimat mirip tidak terjadi [PUR-11].

Penelitian ini akan membuat sebuah sistem pendeteksi kesamaan kalimat dalam dokumen teks menggunakan metode dokumen *fingerprinting* dengan Algoritma *Winnowing*. Dokumen *fingerprinting* merupakan metode yang berfokus pada penjiplakan seluruh dokumen, maupun penjiplakan sebagian data dalam dokumen berjumlah besar [SCH-03]. Sebuah penelitian oleh Saul Schleimer mengatakan bahwa algoritma *Winnowing* merupakan cara yang paling cepat untuk menemukan *fingerprint* dokumen dan memberikan jaminan untuk mendeteksi kemiripan antar dokumen [SCH-03].

Penelitian ini sejalan dengan penelitian yang telah dilakukan oleh Diana Purwitasari dan Saul Schleimer, tetapi pendeteksian tingkat plagiarisme pada penelitian tersebut dilakukan tanpa melewati proses *stemming*, sehingga imbuhan dari suatu kata masih dibaca atau terproses. Pada penelitian ini, penulis akan

memberikan modifikasi, yaitu penambahan *stemming* pada proses *preprocessing* dan penambahan proses dalam perhitungan nilai kemiripan antar dokumen. Dengan penambahan proses *stemming*, algoritma *Winnowing* akan langsung memproses kata dasar saja, sehingga kata-kata berimbuhan yang sebelumnya merupakan kata aktif ataupun pasif akan bernilai sama. Dengan menghilangkan imbuhan kata dan hanya memproses kata dasar saja diharapkan dapat meningkatkan kinerja sistem, dengan asumsi kata aktif yang telah diubah menjadi kata pasif ataupun sebaliknya tetap dapat terdeteksi.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka dirumuskan masalah sebagai berikut:

1. Bagaimana menerapkan Algoritma *Winnowing* dalam pembuatan sistem pendeteksi plagiarisme berdasarkan kesamaan kalimat pada dokumen teks.
2. Bagaimana pengaruh *stemming* pada prosentase *similarity*.
3. Bagaimana pengaruh *stemming* pada waktu pemrosesan.
4. Bagaimana pengaruh *stemming* pada akurasi sistemnya.

1.3. Batasan Masalah

Batasan masalah dalam pembuatan tugas akhir ini ialah sebagai berikut:

1. Pengujian dilakukan pada data berupa dokumen berbentuk teks.
2. Sistem tidak memperhatikan kesalahaan ejaan atau penulisan pada dokumen.
3. Dokumen yang diinputkan bertipe .txt atau .docx.
4. Sistem tidak memperhatikan sinonim atau persamaan kata.
5. Sistem hanya memperhatikan awalan dan akhiran pada kata-kata berimbuhan.
6. Sistem ini menggunakan *stemming* Arifin-Setiono.
7. Data yang diuji menggunakan bahasa Indonesia.

1.4. Tujuan

Tujuan yang ingin dicapai dalam pembuatan tugas akhir ini ialah:

1. Menerapkan Algoritma *Winnowing* pada pembuatan sistem pendeteksi plagiarisme berdasarkan kesamaan kalimat pada dokumen teks.

2. Menganalisa pengaruh *stemming* pada prosentase *similarity*.
3. Menganalisa pengaruh *stemming* pada waktu pemrosesan.
4. Menganalisa pengaruh *stemming* pada akurasi sistem.

1.5. Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk semua pihak khususnya di ruang lingkup akademik. Adapun manfaat yang diharapkan ialah dapat membantu pengguna sebagai bahan pertimbangan dalam menentukan kemiripan pada dokumen teks, khususnya plagiarisme atau penjiplakan pada pengerjaan tugas kuliah.

1.6. Sistematika Pembahasan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut:

BAB I Pendahuluan

Menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan dalam pembuatan tugas akhir.

BAB II Kajian Pustaka dan Dasar Teori

Menguraikan dasar teori dan referensi yang mendasari pembuatan sistem pendeteksi plagiarisme berdasarkan kesamaan kalimat pada dokumen teks menggunakan Algoritma *Winnowing*.

BAB III Metodologi dan Perancangan

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri atas studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis, pengambilan kesimpulan dan saran serta membahas perancangan aplikasi yang dibuat.

BAB IV Implementasi

Membahas implementasi dari sistem pendeteksi plagiarisme berdasarkan kesamaan kalimat pada dokumen teks menggunakan Algoritma *Winnowing* yang sesuai dengan perancangan sistem yang telah dibuat.

BAB V Pengujian dan analisis

Memuat hasil pengujian dan analisis terhadap sistem pendeteksi plagiarisme pada dokumen teks yang telah direalisasikan.

BAB VI Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yag dikembangkan dalam tugas akhir ini serta saran-saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini akan membahas mengenai kajian pustaka dan teori dasar yang dibutuhkan dalam menyusun penelitian. Kajian pustaka membahas penelitian terdahulu dan perubahan yang diusulkan. Dasar teori yang akan dibahas meliputi, plagiarisme, konsep dasar *text mining* dengan *preprocessing* yang terdiri dari *casefolding, tokenizing/parsing*, dan *filtering*. Kemudian *stemming* Arifin-Setiono, cara kerja Algoritma *Winnowing*, dan penghitungan nilai *similarity* dengan menggunakan prinsip *Jaccard Coefficient*.

2.1 Kajian Pustaka

Pada penelitian yang dilakukan oleh Diana Purwitasari yang berjudul "Deteksi Keberadaan Kalimat Sama sebagai Indikasi Penjiplakan dengan Algoritma *Hashing* Ber*basis N-Gram*" pada tahun 2011, membahas tentang deteksi plagiarisme pada dokumen teks berbahasa Indonesia. Metode yang digunakan ialah algoritma Hashing ber*basis* N-Gram atau dapat juga disebut dengan algoritma *Winnowing*. Terdapat empat langkah utama dalam penelitian tersebut, yaitu pemotongan *n-gram*, *hashing*, *window*, dan terakhir penentuan *fingerprints*. Dalam penelitian tersebut, nilai *n-gram*, *window*, *basis*, dan *threshold* diinputkan oleh *user*. *Threshold* pada penelitian tersebut digunakan sebagai batasan penentu apakah sebuah dokumen disebut plagiat atau tidak. *Fingerprints* yang dihasilkan merupakan nilai *hash* terkecil dalam urutan *window*. Penelitian tersebut menguji keempat parameter tersebut dalam menghasilkan prosentase *similarity* [PUR-11].

Penelitian terdahulu oleh Saul Scleimer pada tahun 2003 yang berjudul *Winnowing: Local Algorithms for Document Fingerprinting*, membahas mengenai algoritma *Winnowing* dalam mendeteksi tingkat plagiarisme. Penelitian tersebut memberikan batas atas pada performa *Winnowing*, dalam menyatakan jumlah *fingerprint* dokumen yang akan dihasilkan dan kecocokan terkecil yang dijamin akan didapatkan. Dalam menyatakan kecocokan antar *substring*, digunakan *k* (*k-gram*) sebagai *noise threshold* dan *t* sebagai *guarantee threshold*. Nilai dari *k*

tidak lebih besar dari *t*. Kecocokan *substring* dibawah nilai *k-gram* dianggap tidak sah, sehingga perlu diperhatikan dalam penentuan nilai *k*. Jika nilai *k* terlalu kecil, maka ditakutkan kecocokan antar *fingerprints* merupakan sebuah kebetulan, mengingat *k* digunakan dalam pemotongan *k-gram*. Parameter nilai *k-gram*, *basis*, dan *threshold* diinputkan oleh *user*, sedangkan nilai *window* merupakan hasil perhitungan dari *t-k*+1 yang didasarkan teori *threshold* sebelumnya. Pengujian dilakukan pada 500.000 web page dari Stanford WebBase. Saul Schleimer menyimpulkan bahwa *Winnowing* merupakan algoritma yang effisien dan menjamin kecocokan dalam jarak tertentu [SCH-03].

Pada penelitian "Deteksi Plagiarisme pada Dokumen Teks Bahasa Indonesia menggunakan Algoritma *Winnowing* dengan *Stemming*" akan membahas penerapan algoritma *Winnowing* dalam mendeteksi tingkat plagiarisme pada dokumen teks. Penelitian menggunakan nilai *k* (*k-gram*), *t (threshold*) dan *b* (*basis*) sebagai inputan dari *user*. Nilai *window* akan didapatkan berdasarkan perhitungan pada penelitian yang dilakukan Saul Scleimer. Pengujian akan dilakukan pada sejumlah dokumen yang akan dipotong beberapa katanya, ditukar posisi kalimatnya, dan diubah jenis kata berimbuhannya. Prosentase *Similarity* antar dokumen yang dihasilkan, akan didasarkan pada kecocokan *substring* antar *fingerprint*, yang akan di hitung menggunakan prinsip *Jaccard Coefficient*. Kedua penelitian sebelumnya menggunakan *Winnowing* tanpa melewati proses *stemming*. Pada penelitian ini, *Stemming* Arifin-Setiono akan ditambahkan pada *preprocessing*. *Stemming* Arifin-Setiono dikenal cukup efektif untuk proses *stemming* pada teks berbahasa Indonesia. Dengan penambahan proses *stemming*, algoritma *Winnowing* akan langsung memproses kata dasar saja. Dengan menghilangkan imbuhan kata diharapkan dapat meningkatkan kinerja sistem, dengan asumsi kata aktif yang telah diubah menjadi kata pasif ataupun sebaliknya tetap dapat terdeteksi.

2.2 Plagiarisme

Plagiarisme secara umum merupakan tindakan menggunakan gagasan atau karya orang lain tanpa memberitahu kepada khalayak umum bahwa karya tersebut milik orang lain sehingga masyarakat mengganggap karya tersebut asli atau

orisinil. Sedangkan menurut Kamus Besar Bahasa Indonesia, plagiarisme adalah penjiplakan yang melanggar hak cipta. Penjiplakan ialah menggambar atau menulis garis-garis gambaran atau tulisan yang telah tersedia, mencontoh atau meniru (tulisan, pekerjaan orang lain), mencuri karangan orang lain dan mengakui sebagai karangan sendiri, mengutip karangan orang lain tanpa seizin penulisnya [KAM-13].

Penjiplakan karya dibidang karya ilmiah, dilakukan pada tingkat artikel sampai tingkat disertasi. Bentuk plagiarisme mencakupi penjiplakan secara utuh, sebagian teks, maupun gagasan. Alasan dilakukannya plagiarisme antara lain adalah rasa malas, kesulitan mendapatkan gagasan asli, kurang mahir dalam menulis, sampai ketidaktahuan bahwa yang bersangkutan telah melakukan plagiarisme [NUR-09]. Penjiplakan mempunyai beberapa tipe seperti [NUG-11]:

1. *Word for word plagiarism*

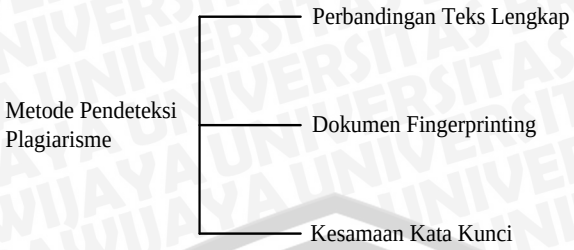
   Menyalin setiap kata secara lansung tanpa diubah sedikitpun.
2. *Plagirism of authorship*

   Mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencatumkan nama sendiri menggantikan nama pengarang sebenarnya.
3. *Plagiarism of ideas*

   Mengakui hasil pemikiran atau ide orang lain sebagai pemikirin diri sendiri.
4. *Plagiarism of sources*

   Jika seorang penulis menggunakan kutipan dari penulis tanpa mencantumkan narasumbernya.

Metode pendeteksi plagiarisme dibagi menjadi tiga bagian yaitu metode perbandingan teks lengkap, metode dokumen *fingerprinting*, dan metode kesamaan kata kunci. Metode pendeteksi plagiarisme dapat dilihat pada Gambar 2. 1 [STE-06]:



Gambar 2. 1 Metode Pendeteksian Plagiarisme

Berikut ini penjelasan dari masing-masing metode dan algoritma pendeteksi plagiarisme [STE-06]:

1. Perbandingan Teks Lengkap.

   Metode ini diterapkan dengan membandingkan semua isi dokumen. Dapat diterapkan untuk dokumen yang besar. Pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif, karena kumpulan dokumen yang diperbandingkan adalah dokumen yang disimpan pada penyimpanan lokal. Metode perbandingan teks lengkap tidak dapat diterapkan untuk kumpulan dokumen yang tidak terdapat pada dokumen lokal. Algoritma yang digunakan pada metode ini adalah algoritma *Brute-Force*, algoritma *edit distance*, algoritma *Boyer Moore* dan algoritma *lavenshtein distance*.

2. Dokumen *Fingerprinting*.

   Dokumen *fingerprinting* merupakan metode yang digunakan untuk mendeteksi keakuratan salinan antar dokumen, baik semua teks yang terdapat di dalam dokumen atau hanya sebagian teks saja. Prinsip kerja dari metode dokumen *fingerprinting* ini adalah dengan menggunakan teknik hashing. Teknik hashing adalah sebuah fungsi yang mengkonversi setiap string menjadi bilangan. Misalnya *Rabin-Karp, Winnowing* dan *Manber*.

3. Kesamaan Kata Kunci.

   Prinsip dari metode ini adalah mengekstrak kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen yang lain. Pendekatan yang digunakan pada metode ini adalah teknik *dot*.

2.3 *Text Mining*

Text mining merupakan salah satu turunan dari bidang keilmuan *Data Mining*. Menurut Feldman, R. Dan Sanger, J., *text mining* atau yang biasa disebut KDT (*Knowledge Discovery in Text*) merupakan sebuah proses pengetahuan intensif dimana pengguna berinteraksi dan bekerja dengan sekumpulan dokumen dengan menggunakan beberapa alat analisis [FEL-07]. *Text mining* mencoba untuk mendapatkan Informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi dari suatu pola. *Text mining* pada dasarnya tidak jauh berbeda dengan *Data Mining*, yang membedakan hanyalah data yang di-*mining*. Sumber data berupa sekumpulan dokumen dan pola menarik yang tidak ditemukan dalam bentuk *database record*, tetapi dalam data teks yang tidak terstruktur [IND-08].

Tahapan proses *text mining* dibagi menjadi 4 tahap utama. Masukan awal berupa suatu data teks dan setelah diproses akan menghasilkan keluaran berupa pola sebagai hasil tasfsiran. Berikut tahapannya [AUV-03]:

1. *Text Prepocessing*

   Tahap proses awal terhadap teks untuk mempersiapkan teks menjadi data yang akan diolah lebih lanjut. Sekumpulan karakter yang bersambungan (teks) harus dipecah-pecah menjadi unsur yang lebih berarti. Hal ini dapat dilakukan dalam beberapa tingkatan yang berbeda. Suatu dokumen dapat dipecah menjadi bab, sub-bab, paragraf, kalimat, kata, dan bahkan suku kata atau fonem. *Parsing/tokenizing* adalah proses memecah teks menjadi kalimat dan kata/token [FEL-07]. Fitur ini terdiri dari tipe kapitalisasi, keberadaan digit, tanda baca, karakter spesial dan lain sebagainya. Hasil keluaran dari proses *tokenizing* akan dipergunakan sebagai masukan dalam tahap transformasi teks.
2. *Text Transformation*

   Tahapan yang dipergunakan untuk mengubah kata-kata ke dalam bentuk dasar, sekaligus untuk mengurangi jumlah kata-kata tersebut. Pendekatan yang dapat dilakukan yaitu dengan *stemming* dan penghapusan *stopwords*. Teknik untuk meningkatkan performa, yaitu dengan cara menemukan variasi token dari token pencarian yang dimasukkan. *Stemming* dapat dilakukan pada saat *indexing* atau pencarian [FRA-92]. Keuntungan *stemming* saat *indexing* adalah efisiensi dan kompresi file. *Stoplist* berisi kumpulan kata yang ‘tidak relevan’,

tetapi seringkali muncul dalam sebuah dokumen. Dengan kata lain, *stoplist* berisi sekumpulan *stopwords* [HAN-01]. *Stopwords removal* adalah proses menghilangkan kata yang 'tidak relevan' dari sebuah dokumen teks dengan cara membandingkan dengan *stoplist* yang ada.

3. *Feature Selection*

   Walaupun teks sudah melalui tahapan transformasi teks, tetapi tidak semua kata yang tersisa menggambarkan isi dari dokumen. Tahap seleksi fitur (*feature selection*) bertujuan mengurangi dimensi dari suatu kumpulan teks. Dengan kata lain, menghapus kata-kata yang dianggap tidak penting atau menggambarkan isi dokumen berdasarkan frekuensi kemunculan kata tersebut.

4. *Pattern Discovery*

   Tahapan penemuan pola adalah tahap terpenting dari keseluruhan proses *text mining.* Merupakan penemuan pola atau pengetahuan dari keseluruhan teks.

2.4 Algoritma *Stemming* Arifin Setiono

Algoritma ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan :

1. Pemeriksaan semua kemungkinan bentuk kata. Setiap kata diasumsikan memiliki 2 Awalan (prefiks) dan 3 Akhiran (sufiks).Sehingga bentuknya menjadi :

   Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1

   Seandainya kata tersebut tidak memiliki imbuhan sebanyak imbuhan di atas, maka imbuhan yang kosong diberi tanda x untuk prefiks dan diberi tanda xx untuk sufiks.

2. Pemotongan dilakukan secara berurutan sebagai berikut :

   AW : AW (Awalan)

   AK : AK (Akhiran)

   KD : KD (Kata Dasar)

   a. AW I, hasilnya disimpan pada p1 (prefiks 1)
   b. AW II, hasilnya disimpan pada p2 (prefiks 2)
   c. AK I, hasilnya disimpan pada s1 (sufiks 1)

d. AK II, hasilnya disimpan pada s2 (sufiks 2)
e. AK III, hasilnya disimpan pada s3 (sufiks 3)

Pada setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Kalau pemeriksaan ini berhasil maka proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya. Contoh pemenggalan kata ”mempermainkannya”:

a. Langkah 1 :
   Cek apakah kata ada dalam kamus
   Ya : Success
   Tidak : lakukan pemotongan AW I
   Kata = permainkannya
b. Langkah 2 :
   Cek apakah kata ada dalam kamus
   Ya : Success
   Tidak : lakukan pemotongan AW II
   Kata = mainkannya
c. Langkah 3 :
   Cek apakah kata ada dalam kamus
   Ya : Success
   Tidak : lakukan pemotongan AK I
   Kata = mainkan
d. Langkah 4 :
   Cek apakah kata ada dalam kamus
   Ya : Success
   Tidak : lakukan pemotongan AK II
   Kata = main
e. Langkah 5 :
   Cek apakah kata ada dalam kamus
   Ya : Success
   Tidak : lakukan pemotongan AK III. Dalam hal ini AK III tidak ada, sehingga kata tidak diubah

Kata = main

f. Langkah 6

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : "Kata tidak ditemukan"

3. Namun jika sampai pada pemotongan AK III, belum juga ditemukan di kamus, maka dilakukan proses kombinasi. KD yang dihasilkan dikombinasikan dengan imbuhan-imbuhannya dalam 12 konfigurasi berikut :

a. KD

b. KD + AK III

c. KD + AK III + AK II

d. KD + AK III + AK II + AK I

e. AW I + AW II + KD

f. AW I + AW II + KD + AK III

g. AW I + AW II + KD + AK III + AK II

h. AW I + AW II + KD + AK III + AK II + AK I

i. AW II + KD

j. AW II + KD + AK III

k. AW II + KD + AK III + AK II

l. AW II + KD + AK III + AK II + AK I

Pemeriksaan 12 kombinasi ini diperlukan, karena adanya fenomena *overstemming* pada algoritma pemotongan imbuhan. Kelemahan ini berakibat pada pemotongan bagian kata yang sebenarnya adalah milik kata dasar itu sendiri yang kebetulan mirip dengan salah satu jenis imbuhan yang ada. Dengan 12 kombinasi itu, pemotongan yang sudah terlanjur tersebut dapat dikembalikan sesuai posisinya [AGU-02].

2.5 Algoritma *Winnowing*

Dalam melakukan pendeteksian penjiplakan terdapat kebutuhan mendasar yang harus dipenuhi oleh suatu algoritma penjiplakan ialah seperti berikut ini [SCH-03]:

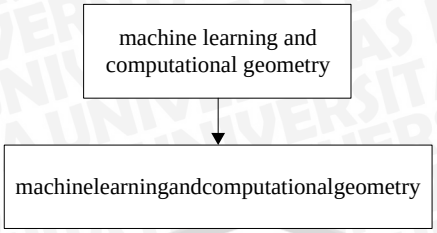
1. *Whitespace Insensitivity*, yang berarti dalam melakukan pencocokan terhadap file teks seharusnya tidak terpengaruh oleh spasi, jenis huruf (kapital atau normal), tanda baca dan sebagainya
2. *Noise Surpression*, yang berarti menghindari penemuan kecocokan dengan panjang kata yang terlalu kecil atau 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 / kesamaan tidak harus bergantung pada posisi kata-kata. Walau tidak dalam berada posisi yang sama pencocokan juga harus dilakukan.

Dalam menghasilkan *fingerprint* dokumen, sistem menggunakan *upper bound* dalam performansi *Winnowing*, agar *fingerprint* dokumen yang diperoleh dapat mendapatkan jarak terdekat yang menjamin terdeteksinya kemiripan antar *substring* [SCH-03]. Dalam sebuah dokumen, besaran *substring* yang baik harus memiliki kriteria sebagai berikut [SCH-03]:

a. Jika terdapat *substring* yang sama sepanjang *guarantee threshold*, *t*, maka kemiripan dianggap sah.
b. Dan kecocokan tidak dianggap sah jika dibawah batasan *noise threshold*, *k*.

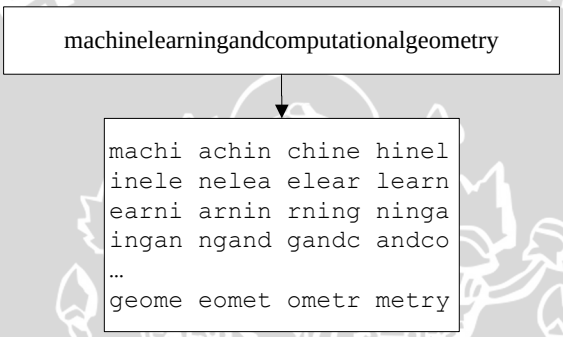
Cara kerja algoritma *Winnowing* ialah untuk mendeteksi adanya keberadaan kalimat sama sebagai suatu indikasi terjadinya penjiplakan menganut prinsip-prinsip tersebut. Pada detil bahasan berikut ditunjukkan keterkaitan antara teori-teori diatas dengan langkah-langkah yang dilakukan dalam algoritma *Winnowing*. Implementasi dari Algoritma *Winnowing* membutuhkan masukan file teks dan akan menghasilkan keluaran berupa sekumpulan nilai *hash* yang disebut *fingerprint*. Proses untuk menghasilkan *fingerprint* sebuah dokumen adalah sebagai berikut [PUR-11]:

1. Membuang karakter-karakter tidak relevan seperti tanda baca.
   Contoh teks:
   machine learning and computational geometry



Langkah tersebut terkait dengan *whitespace insensitivity* dan pembuangan kata-kata tidak penting seperti artikel atau kata sambung. Kemudian pembuangan kata-kata tidak penting atau sering disebut *stopword*. Tahap ini bertujuan menangani *noise surpression*.

2. Membentuk rangkaian *k-gram* dari teks, semisal *k*=5.



Pemotongan *k-gram* berdasarkan nilai *k*. *K-gram* pertama terbentuk dari karakter kesatu sampai dengan sejumlah *k*-karakter, *k-gram* selanjutnya terbentuk dimulai dari karakter kedua sampai sejumlah *k*-karakter, begitu seterusnya sampai berhenti pada *k-gram* terakhir.

Jumlah karakter teks diatas ialah 39 karakter. Dengan *k*=5, maka *k-gram* yang dihasilkan sebesar: *banyak karakter – k* +1, yaitu 35 *gram*.

3. Melakukan fungsi *hash* untuk setiap *k-gram*.

```
machi achin chine hinel      12756 11891 12203 12660
inele nelea elear learn      12809 13009 12441 12800
earni arnin rning ninga      12261 12350 13582 12141
ingan ngand gandc andco      12803 12994 12351 12135
…                            …
geome eomet ometr metry      12497 12578 13305 13063
```

Persamaan (2-1) adalah perhitungan fungsi *hash* dari algoritma *Winnowing* dengan *c* sebagai nilai ASCII, *b* sebagai nilai *basis* bilangan prima, dan banyak karakter *k* [SCH-03]:

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

Untuk menghitung *hash* dari *k-gram* $c_2 \ldots c_{k+1}$, dapat menggunakan Persamaan (2-2) [SCH-03]:

$$H_{(c2\ldots ck+1)} = \left(H_{(c1\ldots ck)} - c_1 * b^{(k-1)}\right) * b + c_{(k+1)} \quad (2\text{-}2)$$

Sebagai contoh *k-gram* dari "*machi*" dengan nilai $b = 3$ dan $k = 5$ memiliki nilai *hash*:

$$H_{(machi)} = ascii_{(m)} * 3^{(4)} + ascii_{(a)} * 3^{(3)} + ascii_{(c)} * 3^{(2)} + ascii_{(h)} * 3^{(1)} + ascii_{(i)} * 3^0$$

$$H_{(machi)} = 109 * 81 + 97 * 27 + 99 * 9 + 104 * 3 + 105 * 1 = 12756$$

4. Memilih *fingerprint* dari hasil *hashing* dengan pembagian hasil *hash* berdasarkan satu nilai *window*, *w*, nilai *window* merupakan hasil dari perhitungan *w* = *t*-*k*+1 [SCH-03]. Misal t=8, sehingga *w* = 8-5+1 = 4. Dalam satu ukuran *window* terdapat 4 nilai *hash* dengan pergeseran 1 *hash* setiap *window* baru, berhenti di *hash* ke *n*, dimana *n* = *jumlah hash* – *w* + 1. Jumlah *window* yang akan dibentuk ialah sebesar *jumlah banyak hash* - *nilai window* + 1. Pada kasus ini maka jumlah *window* yang terbentuk ialah 35 - 4 + 1, yaitu sebesar 32 *window*. Dan kemudian dipilih nilai hash terkecil sebagai *fingerprint* dokumen. Berikut contoh penggalan hasil dari proses pemotongan *window*:

[12756 **11891** 12203 12660]
[**11891** 12203 12660 12809]
[**12203** 12660 12809 13009]
[12660 12809 13009 **12411**]
[12809 13009 **12411** 12800]
[13009 12411 12800 **12261**]
…
[**12114** 12880 12497 12578]
[12880 **12497** 12578 13305]
[**12497** 12578 13305 13063]

Kemudian *fingerprint* yang dihasilkan adalah sejumlah 15 nilai *hash* dari 15 *window* yaitu: 11891 12203 12441 12261 12350 … 12450 13351 12377 12891 12114 12497. Sebagai catatan, untuk pasangan *window* 1-2

atau 4-5 atau *window* 31-32 yang memiliki nilai *hash* terkecil sama maka *window* kanan yang dipilih yaitu *window* 2, 5, dan 32. Kumpulan hash-hash terkecil tersebutlah yang disebut sebagai *fingerprint* dokumen.

2.6 Pengukuran Nilai *Similarity*

Setiap file teks memiliki *fingerprint* sejumlah *j window* yang telah ditentukan atau dokumen D = {w_1 ... w_j}. Kemudian setiap pasang dokumen akan dicari nilai yang sama dari set *window* tersebut dengan menggunakan Persamaan *Jaccard Coefficient* yang ditunjukkan pada Persamaan (2-3) [PR-11]:

$$Similarity(d_i, d_j) = \frac{|w(d_i) \cap w(d_j)|}{|w(d_i) \cup w(d_j)|} \quad (2\text{-}3)$$

Pencarian kesamaan tersebut menunjukkan bahwa algoritma *Winnowing* tidak bergantung pada posisi kata-kata dalam mencari adanya kesamaan atau disebut *position independence.*

Misalkan file teks D_1 dan D_2 dengan *fingerprint* D_1 = **11891 12203 12411 12261** 12350 12803 12351 121135 12211 12450 13351 12377 12891 12114 12497 dan *fingerprint* D_2 = 12232 12268 12411 12500 12195 12508 12756 **11891 12203 12411 12261**, maka sesuai dengan Persamaan (2-3) diperoleh nilai kesamaan (*Similarity*) sebesar 0.190476, kemudian dikalikan dengan 100% untuk menunjukkan dalam bentuk prosentase, yaitu sebesar 19.0476%.

$$Similarity = \frac{|11891\ 12203\ 12411\ 12261|}{21} = 0.190476$$

Prosentase *Similarity* = $0.190476\ x\ 100\% = 19.0476\%$

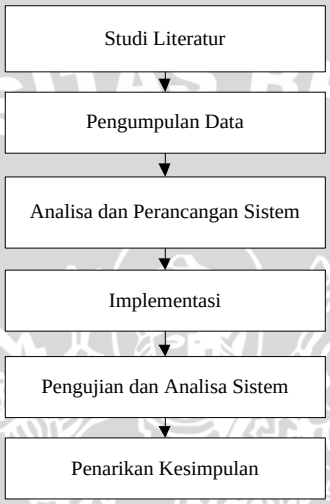
Untuk menentukan jenis plagiarisme antara dokumen yang diuji ada 5 jenis penilaian prosentase *similarity* [MUT-08]:

a. **Sebesar 0%:** Hasil uji 0% berarti kedua dokumen tersebut benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan.
b. **Kurang dari 15%:** Hasil uji 15% berarti kedua dokumen tersebut hanya mempunyai sedikit kesamaan.
c. **Antara 15% sampai 50%:** Hasil uji 15-50% berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang.

d. **Lebih dari 50%:** Hasil uji lebih dari 50% berarti dapat dikatakan bahwa dokumen tersebut mendekati plagiarisme.
e. **Sebesar 100%:** Hasil uji 100% menandakan bahwa dokumen tersebut adalah plagiat karena dari awal sampai akhir mempunyai isi yang sama persis.

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan sistem deteksi plagiarisme pada dokumen teks bahasa Indonesia menggunakan algoritma *Winnowing* dengan *stemming*. Langkah-langkah pengerjakannya dapat Gambar 3.1:



Gambar 3.1 Diagram Alir Penelitian

3.1 Studi Literatur

Pada tahap ini, dilakukan pengumpulan dasar teori yang mendukung pembuatan sistem deteksi plagiarisme. Dasar teori didapatkan dari buku, jurnal ilmiah, *e-book*, dan penelitian-penelitian terdahulu. Kemudian referensi-referensi tersebut dipelajari dan dianalisis untuk menyelesaikan masalah-masalah dalam pembuatan sistem pendeteksi plagiarism sebagai indikasi terjadinya penjiplakan. Adapun referensi tersebut membahas mengenai teori-teori berikut:

1. Pemahaman tentang struktur dokumen penelitian.
2. Pemahaman tentang *text mining.*
3. Pemahaman deteksi plagiarisme.
4. Pemahaman Algoritma *Winnowing*.
5. Algoritma *stemming* Arifin-Setiono.
6. Bahasa *script* HTML dan PHP.
7. *Basis* data MySQL.

3.2 Pengumpulan Data

Data yang dibutuhkan dalam penelitian ini ialah data tugas-tugas kuliah berbentuk dokumen teks tanpa gambar berbahasa Indonesia. Data lainnya ialah daftar kata dasar bahasa Indonesia dan daftar *stopword* (kata tidak penting). Data tersebut merupakan data sekunder. Data sekunder ialah data yang telah dikumpulkan oleh orang lain dan tidak dipersiapkan untuk kegiatan penelitian, tetapi dapat digunakan untuk tujuan penelitian. Data dokumen penelitian diambil dari internet.

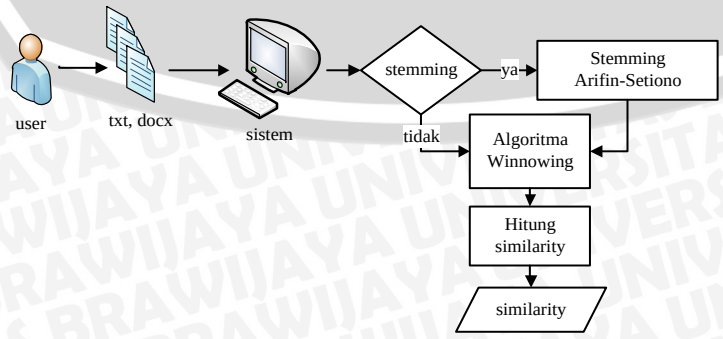
3.3 Analisa dan Perancangan Sistem

Pada tahap ini, dilakukan analisa kebutuhan sistem dan perancangan sistem untuk mendeteksi plagiarisme dengan menggabungkan metode yang telah dijelaskan sebelumnya.

3.3.1 Perancangan Sistem

Pada awal antar muka, sistem ini akan memberikan pilihan kepada *user* untuk menggunakan proses *stemming* atau tidak. Kemudian *user* akan memasukkan dokumen teks yang akan diuji kesamaannya. Setelah itu sistem akan menganalisa prosentase kemiripan (*similarity*) dan waktu pemrosesan dengan pilihan proses yang telah dipilih sebelumnya.

Data yang diuji dalam sistem ini adalah berupa dokumen teks. Dokumen teks dapat bertipe .txt, ataupun .docx. Dengan membandingkan hasil *similarity* dan waktu prosesnya dapat dianalisa efek dari ditambahkannya proses *stemming*, bagaimanakah perbedaan dari kedua keadaan tersebut. Arsitektur sistem dapat dilihat pada Gambar 3.2.



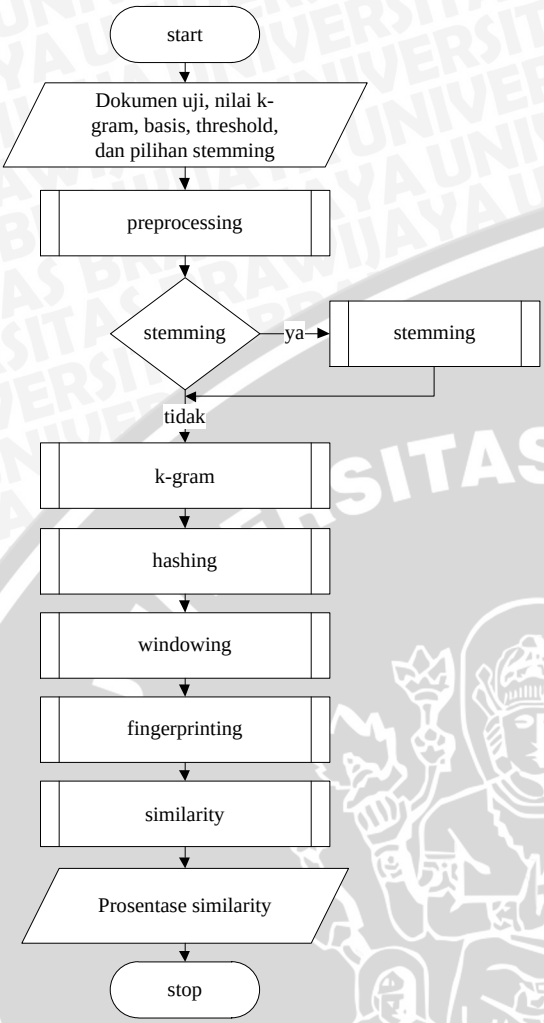
Gambar 3.2 Arsitektur Sistem

3.3.2 Perancangan Proses

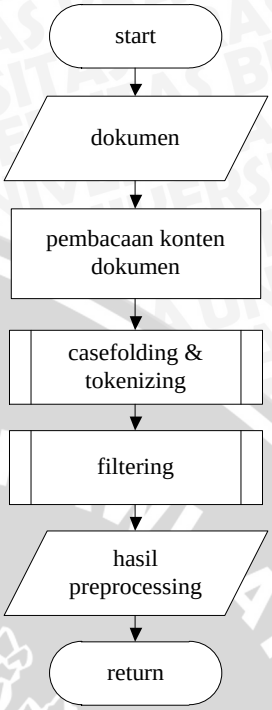
Sistem ini dirancang untuk menerima inputan berupa dokumen teks dengan format .txt, ataupun .docx. Dokumen teks tersebut kemudian akan mengalami beberapa proses untuk mendeteksi kemiripan berdasarkan *fingerprint* yang dimiliki setiap dokumen. Pada akhir proses, sistem akan mengeluarkan hasil prosentase *similarity* antar dokumen dan juga lama waktu pemrosesannya. Proses dapat dilihat pada Gambar 3.3.

Proses yang dilakukan pertama kali ialah proses pembacaan konten atau isi dari dokumen, keluaranya berupa kumpulan kalimat yang kemudian akan menjadi masukkan di proses selanjutnya. Tahap berikutnya ialah *preprocessing* yang meliputi proses *casefolding*, *tokenizing*, *filtering*, dan *stemming*, dapat dilihat pada Gambar 3.4. *Casefolding* merupakan proses menghilangkan huruf *capital*, sehingga semua huruf memiliki format yang sama yaitu *lowercase* (huruf kecil), dan juga menghilangkan karakter-karakter pengganggu, seperti tanda baca (.,:"/$%-+!?), spasi, dan juga angka, sehingga hanya menyisakan karakter a sampai z. Kemudian dilanjutkan pada proses *tokenizing*, yaitu proses membagi kalimat menjadi kata-kata. Proses *casefolding* dan *tokenizing*, berfungsi untuk menangani masalah *Whitespace Insensitivity*, sebagai syarat mendasar yang harus dipenuhi oleh suatu algoritma penjiplakan.

Selanjutnya ialah proses *filtering*, proses ini menghilangkan kata-kata tidak penting seperti "atau", "ialah", ataupun "yang". Proses ini lebih sering dikenal dengan *stopwords removal*. Setelah proses *filtering*, maka masalah *Noise Surpression* terselesaikan. Setelah proses *filtering*, kemudian dilakukan proses *stemming* yaitu proses untuk memperoleh kata dasar dari setiap kata pada dokumen. Proses ini hanya dilakukan, jika pada awal sistem, *user* memilih proses *stemming*. Kemudian dilanjutkan dengan metode *Winnowing*, diawali dengan pembagian *k-gram*, penghitungan *hashing*, kemudian proses mendapatkan *fingerprint* dan diakhiri dengan penghitungan prosentase *similarity* antar dokumen.



Gambar 3.3 Diagram Alir Sistem

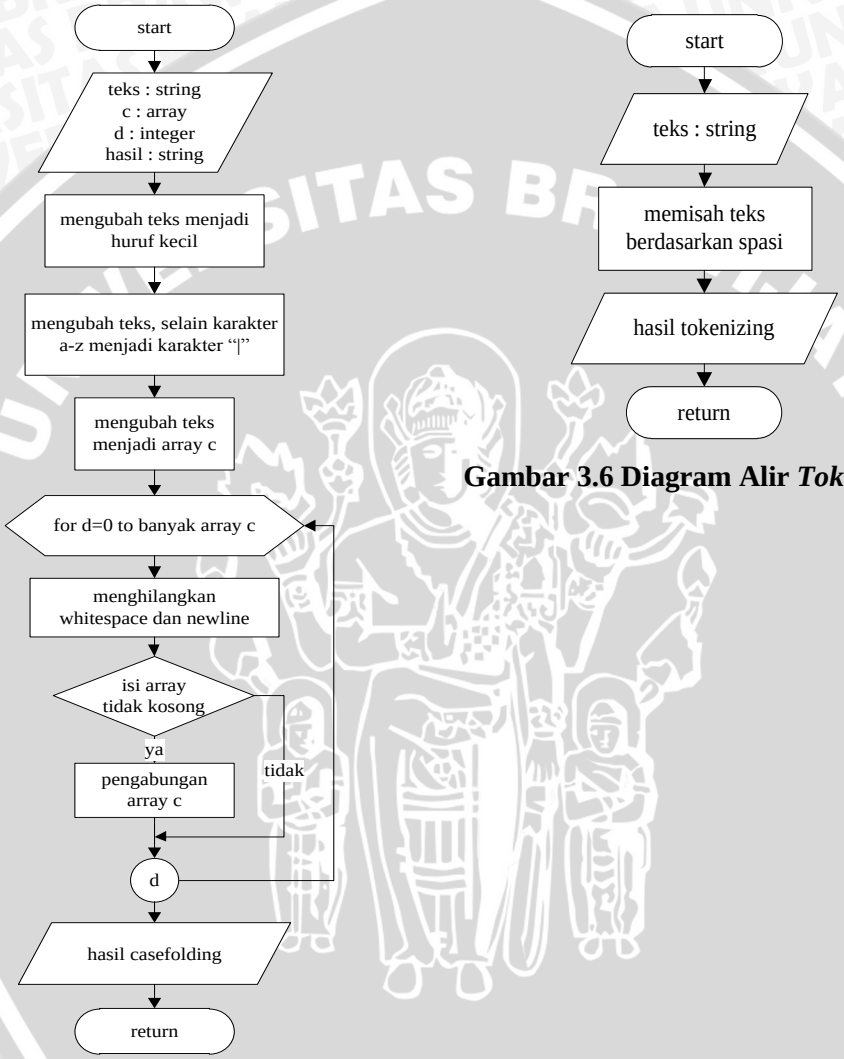


Gambar 3.4 Diagram Alir *Preprocessing*

3.3.2.1 *Preprocessing*

Pada tahap ini terdapat beberapa proses yang dilakukan oleh sistem terhadap dokumen yang diinputkan. Proses-proses tersebut ialah pembacaan konten dokumen, *casefolding*, *tokenizing*, dan *filtering*. Proses *casefolding* adalah proses merubah menjadi huruf kecil semua (*lowercase*) dan menghilangkan karakter yang mengganggu seperti, tanda baca, spasi, dan angka. Cara kerja *casefolding* dimulai ketika teks masuk kedalam sistem, kemudian seluruh karater teks akan diubah menjadi huruf kecil. Kemudian seluruh karakter selain a-z akan diubah menjadi karakter "|". Setelah itu kata akan dipisahkan berdasarkan karater "|" dan dimasukkan ke dalam *array c*. Kemudian dilakukan perulangan sebanyak jumlah *array c* untuk kemudian dilakukan penghapus *newline* dan *whitespace* di kiri-kanan isi *array*. Dan kemudian dilakukan pengecekan terhadap *array* yang

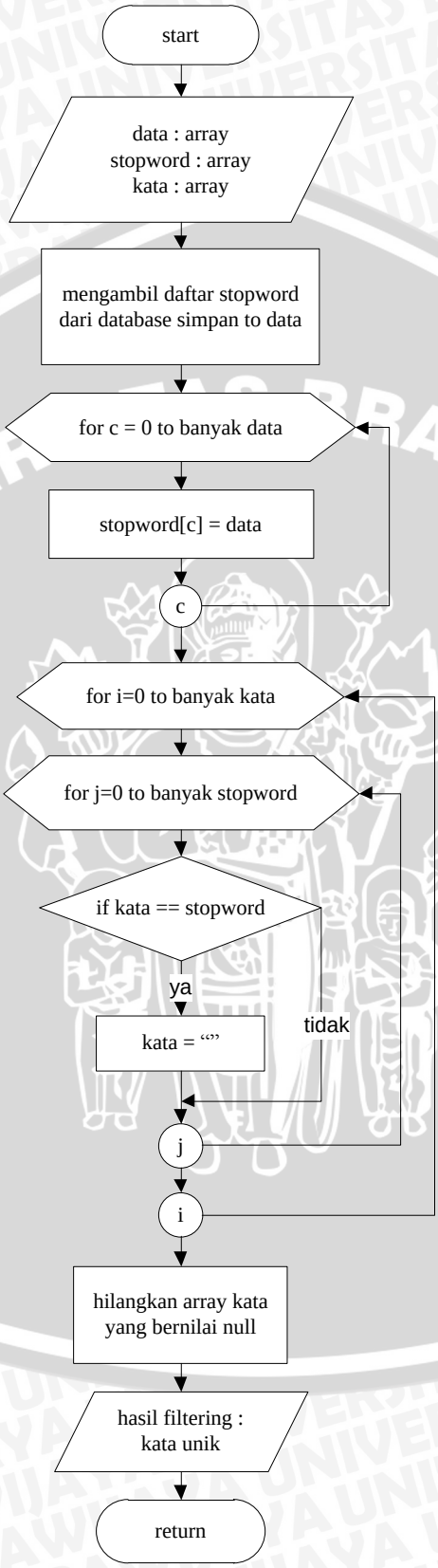
kosong, jika *array* tidak kosong, maka tidak disimpan pada hasil. Proses ini menghasilkan string hasil *casefolding* yang kemudian dijadikan inputan pada proses selanjutnya. Proses dapat dilihat pada Gambar 3.5. Kemudian proses *preprocessing* selanjutnya ialah *tokenizing*, yaitu proses memecah kalimat menjadi kata-kata. Proses kerjanya dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Alir *Tokenizing*

Gambar 3.5 Diagram Alir *Casefolding*

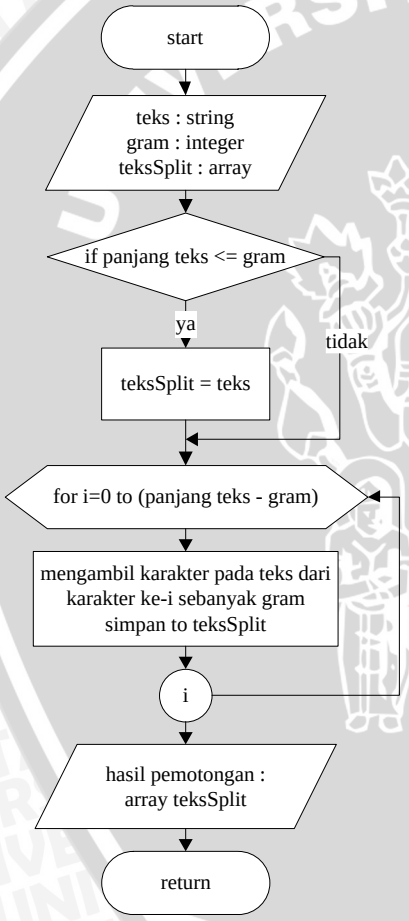
Dan dilanjutkan pada proses *filtering*, yaitu proses penghilangan kata-kata yang tidak penting. Kata-kata *stopword* akan diambil dari database yang kemudian akan disimpan dalam *array* data. Kemudian akan dilakukan pengecekan kata-kata dalam teks terhadap kata-kata *stopword*. Jika kata dalam teks sama dengan kata *stopword*, maka kata tersebut akan dihapus dalam teks. Setelah itu *array* yang kosong akan di hapus. Proses ini menghasilkan keluaran berupa hasil *filtering*. Proses kerjanya dapat dilihat pada Gambar 3.7.



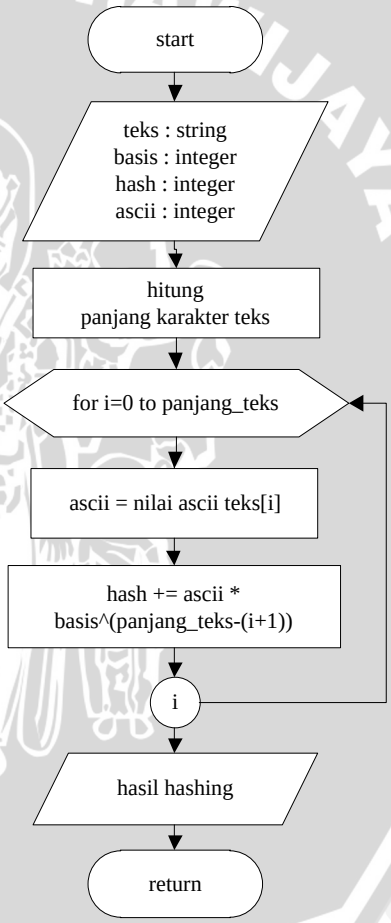
Gambar 3.7 Diagram Alir *Filtering*

3.3.2.2 Algoritma *Winnowing*

Algortima *Winnowing* diawali dengan proses pembagian teks menjadi *gram* sesuai inputan saat awal aplikasi. Proses dimulai dengan pengecekan jumlah karakter pada teks, ketika jumlah karakter lebih kecil atau sama dengan nilai *gram*, maka langsung disimpan pada *array teksSplit* sebagai hasil pemotongan *gram*. Namun jika jumlah karater teks lebih besar dari nilai *gram*, maka dilakukan perulangan untuk memotong teks dimulai dari karakter ke-*i* sebesar nilai *gram* dengan pergeseran satu karater dan disimpan pada *array teskSplit* sebagai hasil pemotongan *gram*. Proses ini akan terlihat pada Gambar 3.8.



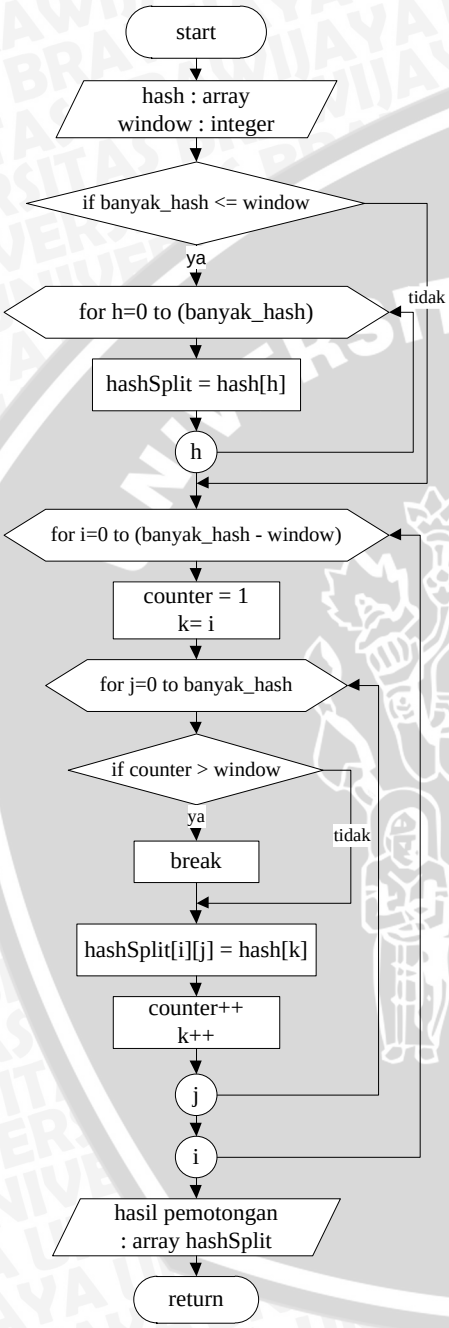
Gambar 3.8 Diagram Alir Pemotongan *k-gram*



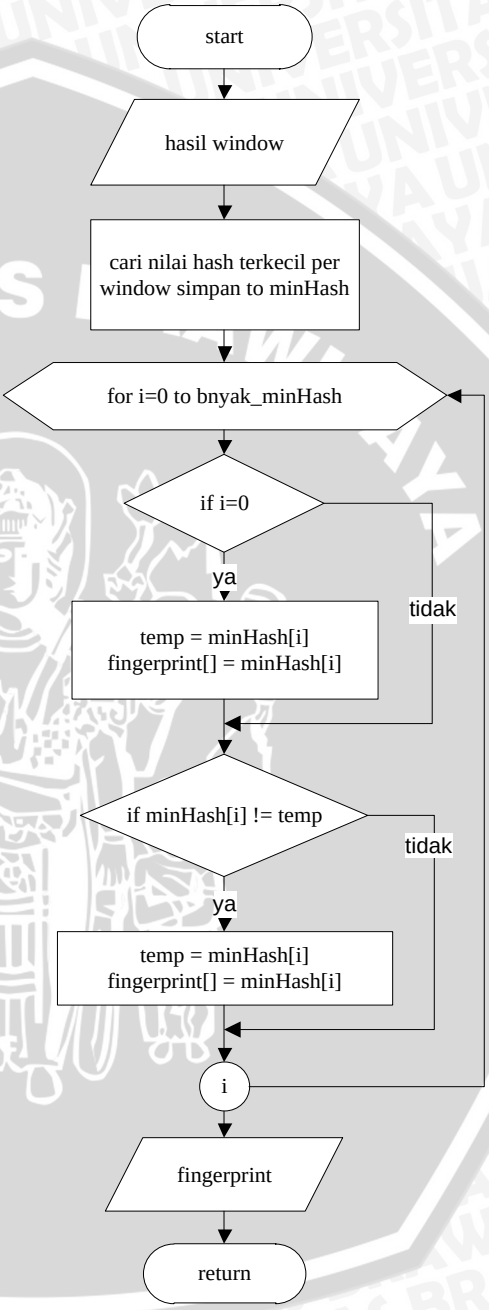
Gambar 3.9 Diagram Alir *Hashing*

Selanjutnya ialah proses *hashing*, yaitu proses perubahan setiap *string* hasil pemotongan *k-gram* menjadi nilai *hash*. Pada proses ini inputan berupa hasil dari pemotongan k-gram, setiap *k-gram* akan diiputkan untuk diubah menjadi nilai *hash*. Selanjutnya dicari panjang karakter *k-gram* dan kemudian dilakukan

perulangan sebanyak jumlah karakter *k-gram* tersebut, dan setiap karakter akan dicari nilai ASCII-nya, untuk kemudian dilakukan perhitungan menggunakan fungsi *hash* pada Persamaan (2-1). Alur kerjanya dijelaskan pada Gambar 3.9.



Gambar 3.10 Diagram Alir *Window*



Gambar 3.11 Diagram Alir *Fingerprint*

Kemudian setelah melewati proses penghitungan *hashing* untuk setiap *k-gram*, proses selanjutnya ialah pemotongan *window*. Hasil *hashing* akan menjadi inputan pada proses pemotongan *window*. Kemudian akan dihitung banyak isi *array hash*. Dan selanjutnya dilakukan perbandingan dengan nilai *window*, ketika

banyak *array hash* lebih kecil atau sama dengan nilai *window*, maka seluruh nilai *hash* langsung disimpan pada *array hashSplit* sebagai hasil dari *window*. Namun ketika banyak *array hash* lebih besar maka dilakukan perulangan untuk mengambil sejumlah nilai *hash* sebanyak nilai *window* dan disimpan pada *array hashSplit* sebagai hasil *window*. Alur prosesnya dapat dilihat pada Gambar 3.10.

Selanjutnya hasil dari pemotongan *window* akan menjadi masukan data pada proses selanjutnya yaitu pemilihan *fingerprint. Fingerprint* dokumen merupakan kumpulan nilai *hash* terkecil pada setiap *window*. Proses diawali dengan mencari nilai terkecil pada setiap *window*, kemudian disimpan pada *array minHash*. Jika ditemukan nilai terkecil yang sama, maka *fingerprint* yang dipilih ialah yang paling kanan. Kemudian dilakukan pengecekan kesamaan nilai *minHash*, pertama-tama akan dihitung banyak isi *array minHash*, untuk selanjutnya akan dilakukan perulangan sebanyak jumlah *minHash*. Nilai *minHash* pertama akan dimasukkan pada *array fingerprint* dan dimasukkan pula pada variabel *temp* untuk dibandingkan pada nilai *minHash* selanjutnya. Ketika nilai *minHash* selanjutnya tidak sama dengan nilai *temp* maka nilai *minHash* tersebut dimasukkan pada *array fingerprint* sebagai hasil *fingerprint*. Perbandingan dilakukan terus-menerus sampai pada isi *array minHash* terakhir. Proses kerja dapat dilihat pada Gambar 3.11. Hasil *fingerprint* inilah yang kemudian akan dicek kemiripannya.

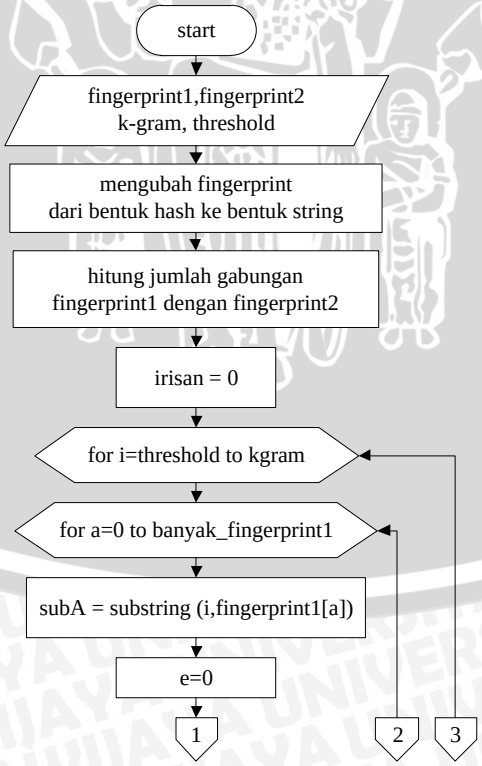
3.3.2.3 Perhitungan *Similarity*

Berbeda dengan penelietian sebelumnya, pada penelitian ini pada proses penghitungan nilai *similarity* akan dilakukan pengecekkan ulang pada *fingerprints* yang dihasilkan. Kemiripan pada *fingerprints* akan didasarkan pada kecocokan *subtring*, sehingga *fingerprints* dalam bentuk *hash* akan dikembalikan kedalam bentuk *substring*. Kecocokan substring menggunakan ambang batas dengan kriteria:

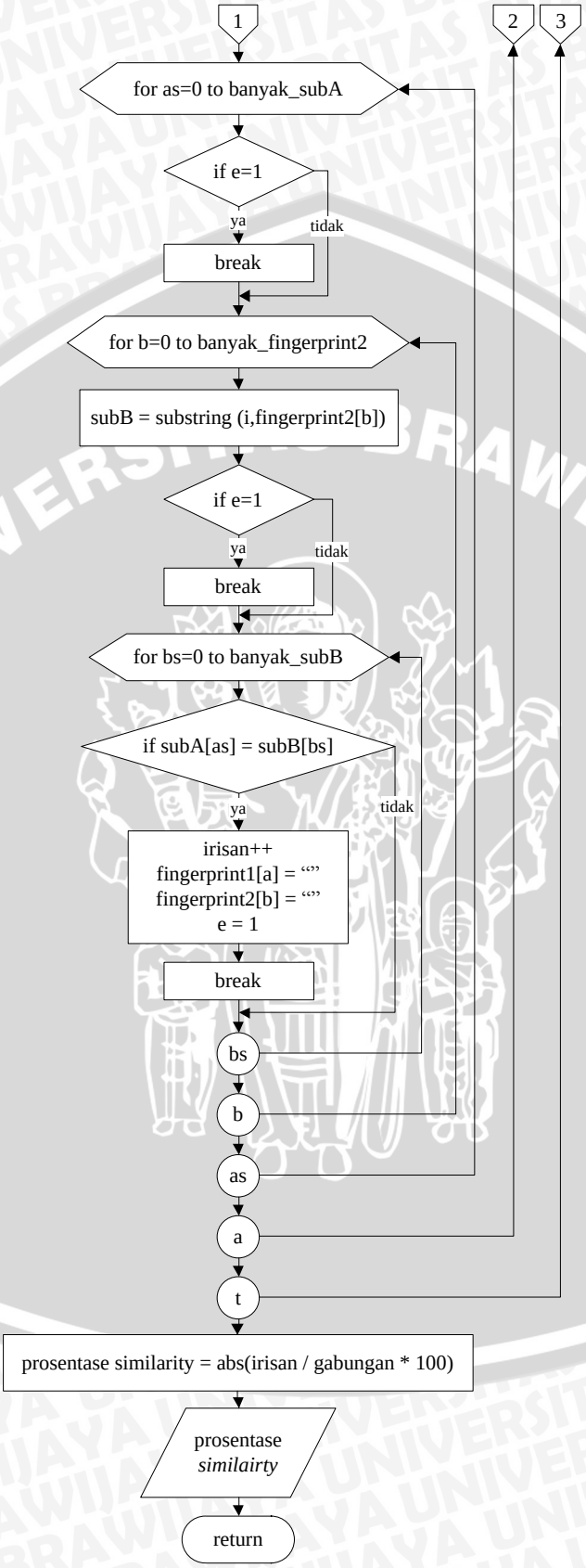
a. Jika terdapat *substring* yang sama sepanjang *guarantee threshold, t*, maka kemiripan dianggap sah.
b. Dan kecocokan tidak dianggap sah jika dibawah batasan *noise threshold*, *k*.

Prosentase *similarity* dihitung menggunakan metode *Jaccard Coefficient* dengan Persamaan (2-3). *Jaccard Coefficient* akan membandingan irisan *fingerprints* antar dua dokumen dengan gabungan dari *fingerprints* dua dokumen. Irisan *fingerprints* dari dua dokumen diperoleh dengan pengecekan kecocokan *substring* sebesar *guarantee threshold* (*t*) sampai dengan *noise threshold* (*k*). berdasarkan irisan dan gabungan yang diperoleh.

Prose diawali dengan mengubah *fingerprint* dari bentuk *hash* menjadi *string*. Kemudian mencari gabungan dari dua *fingerprint* dokumen. Selanjutnya pengecekan *substring* dimulai dari sepanjang nilai *threshold* sampai dengan nilai *k-gram*. Kemudian akan diambil isi pertama pada *array fingerprint1*, dan kemudian akan dibandingkan dengan isi *array* pertama pada *array fingerprint2* sebesar nilai *threshold* sampai dengan nilai *k-gram* dengan pergeseran 1 karakter. Jika ditemukan kesamaan maka varibel *irisan* akan ditambah 1. Hal ini dilakukan sampai pada seluruh *fingerprint* telah dicek. Kemudian setelah diperoleh nilai *irisan* dan *gabungan* maka akan dihitung prosentase *similarity* antara kedua dokumen. Alur dapat dilihat pada Gambar 3.12 dan Gambar 3.13.



Gambar 3.12 Diagram Alir Perhitungan *Similarity*



Gambar 3.13 Diagram Alir Perhitungan *Similarity* (Lanjutan)

3.3.3 Perhitungan Manual Sistem

Perhitungan manual sistem menjelaskan bagaimana langkah-langkah yang dilakukan sistem dari awal dokumen masuk sampai akhirnya menghasilkan prosentase *similarity document*. Teks yang digunakan ialah keseluruhan isi dari dokumen. Berikut contoh penerapan perhitungan dari metode *Winnowing* dalam mendeteksi tingkat plagiarisme pada penggalan isi dokumen teks:

Perhitungan Manual Sistem untuk Pemotongan kata secara tidak acak.

Misalkan *user* menginputkan nilai *k-gram* = 5, *basis* = 11, dan *threshlod* = 8.

a. **Dokumen A**:
   Simpelnya jaringan itu ialah penghubung atau saluran yang diibaratkan seperti pipa ledeng.
b. **Dokumen B**:
   Simpelnya jaringan itu ialah penghubung.

Hasil *preprocessing* (*casefolding, filtering,* dan *stemming*):

a. **Dokumen A**:
   simpeljaringhubungsalurpipaledeng
b. **Dokumen B**:
   simpeljaringhubung

Hasil pemotongan *k-gram*:

Pemotongan menggunakan *threshold*, dengan *t*=8

a. **Dokumen A**:
   simpelja impeljar mpeljari peljarin eljaring ljaringh jaringhu aringhub ringhubu inghubun nghubung ghubungs hubungsa ubungsal bungsalu ungsalur ngsalurp gsalurpi salurpip alurpipa lurpipal urpipale rpipaled pipalede ipaleden paledeng
b. **Dokumen B**:
   simpelja impeljar mpeljari peljarin eljaring ljaringh jaringhu aringhub ringhubu inghubun nghubung

Hasil perhitungan *hashing*

Perhitungan dilakukan pada setiap dokumen menggunakan Persamaan (2-1). Hasilnya ialah sebagai berikut:

Dokumen A :

2446381683 2258927312 2340518032 2380580433 2178190194 2309845257
2257538796 2110885468 2426928808 2259304564 2344667802 2211869027
2251594651 2474217645 2136405135 2493286261 2346159906 2228282161
2432139140 2102259322 2332041193 2501694076 2438645859 2388192116
2261918714 2373423452

Dokumen B:

2446381683 2258927312 2340518032 2380580433 2178190194 2309845257
2257538796 2110885468 2426928808 2259304564 2344667802

Hasil pemotongan *window* dan pemilihan *fingerprint* dokumen

Dengan *threshold* = 8, maka *window* = $t - k + 1$, *window* = 4

Hasil dibaca dari kolom sebelah kiri halaman sampai akhir halaman, kemudian dilanjutkan ke kolom sebelah kanan.

Dokumen A:

[2446381683 **2258927312** 2340518032 2380580433] [2251594651 2474217645 **2136405135** 2493286261]

[2258927312 2340518032 2380580433 **2178190194**] [2474217645 **2136405135** 2493286261 2346159906]

[2340518032 2380580433 **2178190194** 2309845257] [**2136405135** 2493286261 2346159906 2228282161]

[2380580433 **2178190194** 2309845257 2257538796] [2493286261 2346159906 **2228282161** 2432139140]

[2178190194 2309845257 2257538796 **2110885468**] [2346159906 2228282161 2432139140 **2102259322**]

[2309845257 2257538796 **2110885468** 2426928808] [2228282161 2432139140 **2102259322** 2332041193]

[2257538796 **2110885468** 2426928808 2259304564] [2432139140 **2102259322** 2332041193 2501694076]

[**2110885468** 2426928808 2259304564 2344667802] [**2102259322** 2332041193 2501694076 2438645859]

[2426928808 2259304564 2344667802 **2211869027**] [**2332041193** 2501694076 2438645859 2388192116]

[2259304564 2344667802 **2211869027** 2251594651] [2501694076 2438645859 2388192116 **2261918714**]

[2344667802 **2211869027** 2251594651 2474217645] [2438645859 2388192116 **2261918714** 2373423452]

[2211869027 2251594651 2474217645 **2136405135**]

Dokumen B:

[2446381683 **2258927312** 2340518032 2380580433] [2178190194 2309845257 2257538796 **2110885468**]

[2258927312 2340518032 2380580433 **2178190194**] [2309845257 2257538796 **2110885468** 2426928808]

[2340518032 2380580433 **2178190194** 2309845257] [2257538796 **2110885468** 2426928808 2259304564]

[2380580433 **2178190194** 2309845257 2257538796] [**2110885468** 2426928808 2259304564 2344667802]

Hasil *hashing* terkecil per *window* ditandai dengan "huruf tebal", namun yang dikatakan sebagai *fingerprint* dokumen ialah yang diberi "warna merah", karena ketika ditemukan nilai terkecil yang sama dan pada index yang sama pula, maka yang dipilih ialah yang paling sebelah kanan.

Dan hasil *fingerprint* dokumen dari kedua dokumen tersebut ialah:

fingerprint Dokumen A :

```
[2258927312,1]  [2178190194,4]  [2110885468,7]  [2211869027,11]
[2136405135,14]  [2228282161,17]  [2102259322,19]  [2332041193,20]
[2261918714,24]
```

dan *fingerprint* Dokumen B :

```
[2258927312,1] [2178190194,4] [2110885468,7]
```

Setelah mendapatkan *fingerprint* dokumen dari kedua dokumen, kemudian dilakukan penghitungan nilai *similarity* dari kedua dokumen tersebut.

Hasil *fingerprint* akan di ubah kembali ke bentuk *substring*. Hasil perubahan dari kedua dokumen ialah:

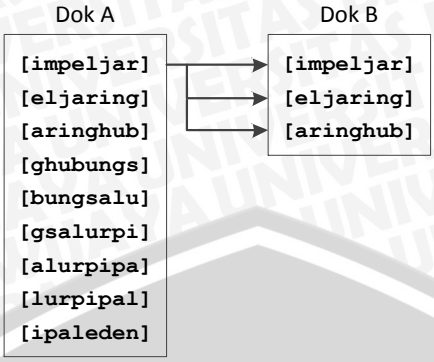
fingerprint Dokumen A :

```
[impeljar][eljaring][aringhub][ghubungs][bungsalu][gsalurpi]
[alurpipa][lurpipal][ipaleden]
```

dan *fingerprint* Dokumen B :

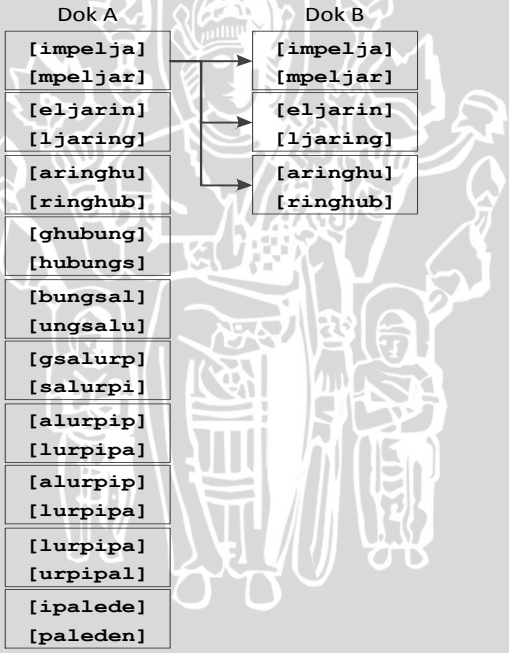
```
[impeljar][eljaring][aringhub]
```

Kemudian, *substring* tersebut akan di cek kemiripannya. Pengecekan kemiripan memiliki nilai ambang batas. Nilai *threshold* sebagai ambang batas atas dan nilai *k-gram* menjadi ambang batas bawah. Pengecekan dimulai dengan panjang subsring sebesar nilai threshold, yaitu pada kasus ini ialah 8.



Gambar 3.14 Ilustrasi Pengecekan *Similarity* dengan *Subtring* = 8

Pada Gambar 3.14, *substring* pertama pada dokumen A akan dicocokkan dengan seluruh *substring* pada dokumen B. Jika ditemukan kesamaan, maka irisan antar dokumen akan bertambah dan *substring* tersebut tidak akan di cek kembali. Pengecekan dilanjutkan sampai pada *substring* terakhir dokumen A.



Gambar 3.15 Ilustrasi Pengecekan *Similarity* dengan *Substring* = 7

Kemudian jika belum ditemukan kecocokan, pengecekan dilanjutkan untuk panjang *substring* = 7. Setiap *substring* akan di pecah sehingga panjang *substring* menjadi 7. Dapat dilihat pada Gambar 3.15, setiap *substring* dipecah menjadi 2 *substring* baru kemudian dicocokkan kesamaannya dengan *substring* dokumen lain. Jumlah substring dokumen A sebelumnya hanya 9, tetapi pada pengecekan substring = 7, jumlah substring bertambah menjadi 18. Hal ini juga

terjadi pada dokumen B, sehingga pengecekkan kecocokan substring bertambah lama.

Jika belum juga di temukan kecocokan pengecekkan *substring* terus berlanjut sampai nilai ambang batas bawah, yaitu 5. Pada pengecekkan terakhir akan diketahui berapa besar irisan dan gabungan *substring* antar dokumen. Nilai irisan dan gabungan tersebut kemudian akan diproses dengan metode *Jaccard Coeficient* untuk mendapatkan prosentase *similarity* antar dokumen.

$$Similarity(d_A, d_B) = \frac{|w\ (d_A) \cap\ w(d_B)|}{|w\ (d_A) \cup\ w(d_B)|}$$

$$Similarity(d_A, d_B) = \frac{|\text{impeljar eljaring aringhub}|}{|\text{impeljar eljaring aringhub ghubungs bungsalu gsalurpi alurpipa lurpipal ipaleden}|}$$

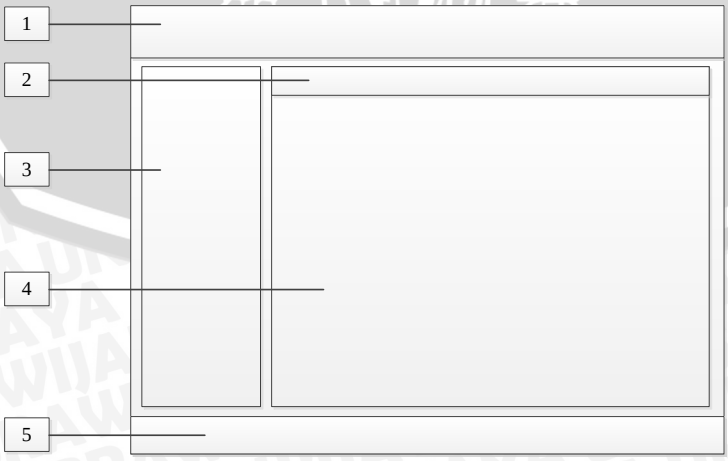
$$Similarity(d_A, d_B) = \frac{3}{9} = 0{,}333333$$

$$\text{Prosentase } Similarity(d_A, d_B) = \frac{3}{9}\ x\ 100 = 33{,}3333\%$$

Setelah proses diatas, didapatkan irisan sebesar 3, dan gabungan sebesar 9, sehingga berdasarkan Persamaan (2-3) maka akan diperoleh nilai similarity sebesar 0,333333. Jika diubah menjadi prosentase *similarity*, yaitu sebesar 33,3333%.

3.3.4 Perancangan Antar Muka Sistem Deteksi Plagiarisme Dokumen Teks

Perancangan antar muka sistem deteksi plagiarisme dapat dilihat pada Gambar 3. 16. Tujuan perancangan ini ialah kenyamanan *user* dalam menggunakan aplikasi.



Gambar 3. 16 Rancangan Antar Muka Sistem

Keterangan Gambar 3. 16:

1. *Header*
2. Menu
3. Folder
4. Konten
5. *Footer*

Desain antar muka sistem ini terdiri dari lima komponen utama sebagaimana yang dapat dilihat pada Gambar 3. 16. Folder pada antar muka sistem terdapat *icon add folder*. Setelah dibuat folder, maka konten menampilkan menu *add document*, dimana dokumen yang dapat diinputkan ialah dokumen yang bertipa .docx dan .txt. Pada halaman deteksi plagiarisme, *user* akan memasukkan nilai-nilai yang dibutuhkan pada proses deteksi dan pilihan mengenai penggunaan *stemming*. Setelah tombol deteksi ditekan, maka sistem akan menampilan hasil perbandingan dokumen berupa judul dokumen, prosentase *similarity*, dan waktu eksekusi dalam satu folder.

3.4 Implementasi

Pada tahap ini dilakukan penerapan dari rancangan yang telah dibuat di tahap sebelumnya. Sistem ini dibangun dengan *script* PHP yang dikoneksikan dengan *basis* data MySQL. *Basis* data hanya dipakai untuk menyimpan daftar kata dasar atau kamus dan juga daftar kata *stopword*, dan juga untuk menyimpan daftar dokumen serta teks yang telah diolah.

Implementasi sistem penentuan tingkat plagiarisme dokumen teks ini ialah sebagai berikut:

a. Penerapan komponen-komponen *text mining*.
b. Penerapan metode *stemming* arifin-setiono pada *text mining*.
c. Penerapan metode *Winnowing*.
d. Pembuatan *user interface* sistem penentuan tingkat plagiarisme dokumen teks untuk *end-user*.

3.5 Pengujian dan Analisa Sistem

Pengujian sistem dilakukan untuk mendapatkan nilai parameter *k-gram*, *basis*, dan *threshold* yang optimal dan juga untuk mengukur prosentase *similarity*

dan juga waktu pemrosesan. Nilai *similarity* adalah kemiripan antar dokumen. Pengukuran nilai *similarity* yang dibandingkan adalah nilai *similarity* yang didapat dari hasil keluaran sistem dengan menggunakan *stemming* dan tidak menggunakan *stemming*. Waktu proses ialah perbandingan seberapa lama waktu yang dibutuhkan oleh kedua jenis proses tersebut untuk melakukan seluruh proses dari awal hingga akhir sampai menghasilkan nilai *similarity*.

3.5.1. Perancangan Pengujian

Tahap ini mendeskripsikan bahan pengujian, tujuan pengujian, perancangan tabel uji, sampai pada pengukuran nilai *similarity* dan *running time* serta perancangan data uji.

3.5.1.1.Bahan Pengujian

Data yang digunakan berupa 20 dokumen teks yang mempunyai ekstensi .txt atau .docx. Data tersebut merupakan kumpulan tugas mata kuliah Jaringan Komputer, Jurusan Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya Malang. Dokumen yang diuji akan diproses sedemikian rupa untuk menguji tingkat plagiarisme dokumen dan bagaimana perbedaan *similarity* dan waktu pemrosesan dari sebelum dan setelah penambahan proses *stemming*.

3.5.1.2.Tujuan Pengujian

Tujuan dari pengujian sistem deteksi tingkat plagiarisme dokumen teks ini ialah sebagai berikut:

1. Menganalisa hasil *similarity* dan *error* sistem yang dihasilkan berdasarkan nilai *kgram*, *basis*, dan *threshold*.
2. Menganalisa pengaruh proses *stemming* terhadap nilai *similarity* dan waktu pemrosesan
3. Menganalisa prosentase *error* yang dihasilkan oleh sistem.

3.5.1.3.Perancangan Tabel Hasil Uji Coba

Perancangan tabel hasil uji coba diharapkan dapat memberikan Informasi mengenai perbedaan prosentase *similarity*, waktu eksekusi, dan *error* sistem antara sistem sebelum penambahan proses *stemming* dan setelah ditambahkan proses *stemming*. Pengujian juga dilakukan untuk menemukan nilai *k-gram*, *basis*, dan *threshold* yang optimal dalam mendeteksi plagiarisme.

a. Pengujian nilai *k-gram*, *basis*, dan *threshold*

Pengujian ini dilakukan untuk menemukan nilai *k-gram*, *basis*, dan *threshold* yang optimal dalam mendeteksi plagiarisme. *K-gram* ialah sebagai *noise threshold*, yaitu batasan terkecil kecocokan *substring*. *Basis* ialah bilangan prima yang dipakai dalam perhitungan *hashing. Threshold* merupakan *guarantee threshold* yang digunakan sebagai batas akhir pengecekan dalam menghitung prosentase *similarity*.

Berdasarkan dasar teori yang ada, ketiga parameter tersebut memiliki persyaratan sebagai berikut:

a. *K-gram* memiliki nilai lebih kecil dari atau sama dengan *threshold*.
b. Nilai dari *basis* merupakan bilangan prima.

Nilai yang akan diujikan ialah sebagai berikut:

1. *K-gram* : 5, 10, 15, 20, 25
2. *Threshold* : 5, 10, 15, 20, 25
3. *Basis* : 3, 5, 7, 11, 17,19

Pada pengujian ini akan dilakukan sesuai dengan skenario pengujian yang telah dirancang pada Tabel 3.1. Pengujian terbagi menjadi 3 tahapan :

1. Pengujian dilakukan pada nilai *k-gram*, nilai *threshold* dan *basis* yang digunakan ialah nilai yang paling besar, yaitu 25 untuk *threshold*, dan 17 untuk *basis*. Hal ini bertujuan untuk menganalisis nilai *k-gram* yang paling optimal.
2. Pengujian dilakukan pada nilai *threshold*, untuk nilai *k-gram* akan digunakan nilai terbaik, hasil dari pengujian tahapan 1. Nilai *basis* yang digunakan ialah nilai terbesar. Hal ini bertujuan untuk menganalisis nilai *threshold* yang paling optimal.

3. Pengujian dilakukan terhadap nilai *basis*, untuk nilai *k-gram* dan *threshold* yang digunakan merupakan hasil dari pengujian 1 dan 2. Pengujian ini bertujuan untuk menganalisis nilai *basis* paling optimal dalam mendeteksi tingkat plagiarisme pada dokumen teks.

Tabel 3.1 Skenario Pengujian Nilai *K-gram*, *Basis*, dan *Threshold*

Skenario	*K-gram*	*Threshold*	*Basis*	*Similarity* (%)	*Error* (%)
1	5	25	17		
	10	25	17		
	15	25	17		
	20	25	17		
	25	25	17		
2		5	17		
		10	17		
		15	17		
		20	17		
		25	17		
3			3		
			5		
			7		
			11		
			17		
			19		

Rancangan tabel hasil pengujian nilai *k-gram, basis,* dan *threshold* dapat dilihat pada

Tabel 3.2.

Tabel 3.2 Pengujian Nilai *K-gram, Basis,* dan *Threshold*

No.	File 1	File 2	*Similarity* (%)	*Error* (%)
1	Dok A	100% sama		
2		80% sama		
3		60% sama		
4		40% sama		
5		20% sama		
6		Tukar 20%		
7		Tukar 60%		
8		Ubah 10%		

Penjelasan Tabel 3.2:

- Kolom File 1 dan File 2 menunjukkan dokumen yang akan dibandingkan.
- Kolom *Similarity* menunjukan infromasi nilai kemiripan yang akan dihasilkan sistem.

- Kolom *Error* Sistem menunjukkan Informasi lama waktu proses sistem.

b. Pengujian *similarity* dokumen, lama waktu eksekusi sistem dan *error* sistem.

Prosentase *similarity* ialah nilai kemiripan antar dua dokumen yang ingin diketahui kemiripannya. Pengukuran nilai *similarity* yang dibandingkan ialah nilai *similarity* dari sistem dengan menggunakan *stemming* dan dengan hasil keluaran sistem yang tidak menggunakan proses *stemming*. Waktu eksekusi ialah perbandingan lama waktu yang dibutuhkan oleh kedua kondisi, sistem dengan *stemming* dan tanpa *stemming* dari awal proses sampai ditemukannya prosentase *similarity*. Pengujian juga dilakukan untuk membandingkan prosentase *similarity* yang dihasilkan oleh sistem dengan prosentase *similarity* yang diharapkan, sehingga diperoleh prosentase *error* sistem. Nilai dari *k-gram*, *threshold*, dan *basis* yang digunakan merupakan hasil dari pengujian pertama. Hal ini dilakukan untuk dapat melihat apakan sistem dapat berjalan dengan baik. Rancangan tabel hasil pengujian *similarit*y, waktu eksekusi, dan *error* sistem dapat dilihat pada Tabel 3.3.

Tabel 3.3 Pengujian *Similarity*, Waktu Eksekusi, dan *Error* Sistem

No.	File 1	File 2	*Similarity* (%)		*Error* Sistem (%)		Waktu Eksekusi (s)	
			Stemming	Tanpa *Stemming*	*Stemming*	Tanpa *Stemming*	*Stemming*	Tanpa *Stemming*
1	Dok A	100% sama						
2		80% sama						
3		60% sama						
4		40% sama						
5		20% sama						
6		Tukar 20%						
7		Tukar 60%						
8		Ubah 10%						

Penjelasan Tabel 3.3:

- Kolom File menunjukkan Informasi dokumen yang akan dibandingkan.
- Kolom *similarity* menunjukan Informasi hasil prosentase nilai *similarity* yang dihasilkan oleh sistem dengan *stemming* dan tanpa *stemming*.

- Kolom waktu eksekusi menunjukkan Informasi waktu yang dibutuhkan sistem dalam satuan detik (*second*) untuk melakukan deteksi dari awal proses hingga menemukan prosentase *similarity*.
- Kolom *Error* menunjukkan nilai *error* sistem dalam persen (%).

3.5.1.4. Perancangan Dokumen Uji dan Dokumen Latih

Dokumen latih yang digunakan merupakan hasil pengubahan dari dokumen uji dengan beberapa kondisi sebagai berikut:

1. 100% sama: Dokumen latih yang isi teksnya sama persis (100%) dengan dokumen uji.
2. 80% sama: dokumen uji yang isi teksnya dilakukan pemotongan sebanyak 20% kata secara acak sehingga menghasilkan 80% kata yang sama.
3. 60% sama: dokumen uji yang isi teksnya dilakukan pemotongan sebanyak 40% kata secara acak sehingga menghasilkan 60% kata sama.
4. 40% sama: dokumen uji yang isi teksnya dilkukan pemotongan sebanyak 60% kata secara acak sehingga menghasilkan 40% kata sama.
5. 20% sama: dokumen uji yang isi teksnya dilakukan pemotongan sebanyak 80% kata secara acak sehingga menghasilkan 20% kata sama.
6. Tukar 20%: dokumen uji yang isi kalimatnya ditukar sebanyak 20% dari kalimat keseluruhan
7. Tukar 60%: dokumen uji yang isi kalimatnya ditukar sebanyak 60% dari kalimat keseluruhan.
8. Ubah 10%: dokumen uji yang 10% kata kerja didalamnya diganti menjadi pasif atau sebaliknya serta perubahan partikel penyusun katanya tanpa mengubah posisi subjek, predikat, dan objeknya.

Jenis dokumen latih adalah dokumen uji yang telah dirubah sedemikian rupa untuk mengecek apakah sistem yang telah dibuat telah sesuai. Berikut ini adalah contoh data dokumen uji dan dokumen-dokumen latih:

a. Dokumen Uji (A):

> Sejak usia dini, anak-anak sudah dikenalkan dengan teknologi dan menggunakannya baik di sekolah maupun di lingkungan rumahnya, anak tersebut telah diajarkan baik secara langsung maupun secara tidak langsung pada berbagai macam teknologi. Jaringan komputer merupakan salah satu dari teknologi itu sendiri. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang terhubung dan memungkinkan berbagi sumber daya. Jaringan komputer banyak digunakan oleh seluruh lapisan masyarakat, dari yang muda sampai yang tua. Keberadaan jaringan komputer ini sangat menguntungkan baik bagi perseorangan maupun suatu kelompok tertentu.

b. Dokumen Latih Sama (A-00):

Sejak usia dini, anak-anak sudah dikenalkan dengan teknologi dan menggunakannya baik di sekolah maupun di lingkungan rumahnya, anak tersebut telah diajarkan baik secara langsung maupun secara tidak langsung pada berbagai macam teknologi. Jaringan komputer merupakan salah satu dari teknologi itu sendiri. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang terhubung dan memungkinkan berbagi sumber daya. Jaringan komputer banyak digunakan oleh seluruh lapisan masyarakat, dari yang muda sampai yang tua. Keberadaan jaringan komputer ini sangat menguntungkan baik bagi perseorangan maupun suatu kelompok tertentu.

c. Dokumen Latih 80%sama (A-P20)

Sejak dini, anak-anak dikenalkan dengan teknologi baik di sekolah maupun di rumahnya, anak tersebut telah diajarkan baik secara langsung maupun secara tidak langsung pada berbagai teknologi. Jaringan komputer merupakan salah satu dari teknologi. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang terhubung dan memungkinkan berbagi sumber daya. Jaringan komputer banyak digunakan oleh masyarakat, yang muda sampai yang tua. Keberadaan jaringan ini sangat menguntungkan baik bagi perseorangan maupun kelompok.

d. Dokumen Latih 60%sama (A-P40)

Sejak dini, sudah dikenalkan teknologi baik di sekolah maupun di lingkungan rumahnya, diajarkan baik secara langsung maupun tidak langsung. Jaringan dari teknologi itu sendiri. Jaringan hubungan dari dua komputer atau lebih dan memungkinkan berbagi. Jaringan seluruh lapisan masyarakat, muda sampai yang tua. jaringan komputer baik bagi perseorangan maupun suatu kelompok.

e. Dokumen Latih 40%sama (A-P60)

anak-anak dikenalkan teknologi di lingkungan rumahnya, diajarkan baik secara langsung maupun tidak langsung. Komputer satu dari teknologi. komputer hubungan dua komputer yang terhubung. Jaringan digunakan oleh seluruh lapisan masyarakat. Keberadaan jaringan ini sangat menguntungkan.

f. Dokumen Latih 20%sama (A-P80)

anak-anak sudah dikenalkan teknologi di sekolah. Jaringan komputer. Jaringan merupakan hubungan. banyak digunakan masyarakat. Keberadaan jaringan menguntungkan.

g. Dokumen Latih Tukar20% (A-T20)

> Sejak usia dini, anak-anak sudah dikenalkan dengan teknologi dan menggunakannya baik di sekolah maupun di lingkungan rumahnya, anak tersebut telah diajarkan baik secara langsung maupun secara tidak langsung pada berbagai macam teknologi. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang terhubung dan memungkinkan berbagi sumber daya. Jaringan komputer merupakan salah satu dari teknologi itu sendiri. Jaringan komputer banyak digunakan oleh seluruh lapisan masyarakat, dari yang muda sampai yang tua. Keberadaan jaringan komputer ini sangat menguntungkan baik bagi perseorangan maupun suatu kelompok tertentu.

h. Dokumen Latih Tukar60% (A-T60)

> Sejak usia dini, anak-anak sudah dikenalkan dengan teknologi dan menggunakannya baik di sekolah maupun di lingkungan rumahnya, anak tersebut telah diajarkan baik secara langsung maupun secara tidak langsung pada berbagai macam teknologi. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang terhubung dan memungkinkan berbagi sumber daya. Keberadaan jaringan komputer ini sangat menguntungkan baik bagi perseorangan maupun suatu kelompok tertentu. Jaringan komputer merupakan salah satu dari teknologi itu sendiri. Jaringan komputer banyak digunakan oleh seluruh lapisan masyarakat, dari yang muda sampai yang tua.

i. Dokumen Latih Ubah10% (A-U10)

> Sejak usia dini, anak-anak sudah mengenalkan dengan teknologi dan digunakannya baik di sekolah maupun di lingkungan rumahnya, anak tersebut telah mengajarkan baik secara langsung maupun secara tidak langsung pada berbagai macam teknologi. Jaringan komputer merupakan hubungan dari dua komputer atau lebih yang dihubung dan dimungkinkan berbagi sumber daya. Jaringan komputer dirupakan salah satu dari teknologi itu sendiri. Jaringan komputer banyak menggunakan oleh seluruh lapisan masyarakat, dari yang muda sampai yang tua. Keberadaan jaringan komputer ini sangat diuntungkan baik bagi perseorangan maupun suatu kelompok tertentu.

3.6 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap

terakhir dari penulisan adalah saran. Saran bertujuan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
IMPLEMENTASI

Tahap ini membahas mengenai implementasi sistem berdasarkan perancangan yang telah diuraikan sebelumnya pada bab 3. Implementasi meliputi implementasi proses dan juga implementasi antar muka. Bab ini juga membahas spesifikasi yang dibutuhkan dalam pengerjaan sistem ini, baik spesifikasi perangkat lunak maupun perangkat keras.

4.1. Spesifikasi Sistem

Spesifikasi sistem meliputi spesifikasi perangkat keras dan juga spesifikasi perangkat lunak.

4.1.1. Spesifikasi Perangkat Keras

Dalam perancangan dan pengembangan sistem deteksi tingkat plagiarisme dokumen teks ini digunakan sebuah komputer dengan spesifikasi sebagai berikut:

1. Processor Intel(R) Core(TM)2 Duo CPU T5470 @ 1.60GHz (2 CPUs)
2. RAM 2 GB
3. Hardisk 160 GB

4.1.2. Spesifikasi Perangkat Lunak

Pengembangan sistem deteksi tingkat plagiarisme dokumen teks ini menggunakan spesifikasi perangkat lunak sebagai berikut:

1. Sistem Operasi *Window*s 7 Ultimate
2. XAMPP 1.7.3
3. HTML dan PHP 5.3.1
4. Adobe Dreamwaver CS3 versi 9.0
5. MySQL client version: 5.1.41
6. Google Chrome Version 34.0.1847.131 m

4.2. Implementasi Proses Sistem

Tahap ini menjelaskan proses sistem dalam mengimplementasikan algoritma *Winnowing* dalam mendeteksi tingkat plagiarisme pada dokumen teks. Proses diawali dengan masuknya dokumen kemudian sistem memulai proses pembacaan konten dokumen baik yang bertipe txt, ataupun docx. Selanjutnya sistem menjalani beberapa proses, yaitu proses *preprocessing* dan proses algoritma *Winnowing*.

4.2.1. Proses *Preprocessing*

Tahap *preprocessing* memiliki beberapa proses, yaitu *casefolding*, *filtering*, dan *stemming*. *Casefolding* merupakan proses menghilangkan karakter yang mengganggu serta mengubah teks menjadi huruf kecil. *Filtering* merupakan proses menghilangkan kata-kata yang tidak memiliki arti kurang penting, seperti "dan", "ialah", "seperti", dan lain-lain. Dan proses *stemming* merupakan proses mengubah kata-kata pada teks menjadi kata dasar.

4.2.1.1.Proses *Casefolding* dan *Tokenizing*

Proses ini merupakan tahap awal dalam *preprocessing*. Teks yang dimasukkan diubah menjadi huruf kecil dan kemudian karakter-karakter yang tidak dibutuhkan akan dihilangkan. Dan dilanjutkan dengan proses memecah kalimat menjadi potongan kata-kata. Implementasi pada program dapat dilihat pada Gambar 4.1

```
1  //casefolding
2  $lowercase = strtolower("$file");
3  $a=preg_replace("/[^A-Za-z]/", "|", $lowercase); //mengganti
   karakter selain A-Z dan a-z dengan karakter "|"
4  $b = explode("|", $a); //misah string berdasarkan |
5  for ($c = 0; $c < count($b); $c++) {
6        if (trim($b[$c]) != "")
7              $d .= $b[$c] . "|";
8  }
9  $d = substr($d, 0, strlen($d) - 1);
10
11 //tokenizing
12 $pisah = explode("|", $d);
```

Gambar 4.1 Implementasi *Casefolding* dan *Tokenizing*

4.2.1.2.Proses *Filtering*

Proses ini sering disebut sebagai *stopword removal*. Pada tahap ini, teks hasil dari proses *casefolding* dan *tokenizing* sebelumnya akan dibandingkan dengan kata-kata *stopword* yang berada di database. Ketika kata dalam teks tadi sama dengan daftar kata *stopword*, maka kata tersebut akan dihapus, sehingga teks masukkan bersih dari kata stopword atau kata tidak penting. Proses tersebut ditunjukkan oleh Gambar 4.2

```
1  $counter=0;
2  $sql = mysql_query("select kata from stopwords");//mengambil
   daftar kata stopwords dari database
3  while ($data = mysql_fetch_row($sql))
4  {
5        $stopwords[$counter++] = $data[0];
6  }
7
8  for ($i=0; $i < count($pisah); $i++){
9        for ($j=0; $j < count($stopwords); $j++){
10             if ($stopwords[$j] == $pisah[$i]){
11                   $pisah[$i] = "";
12                   break;
13             }
14       }
15 }
16 $new_array=array_empty_remover($pisah, true);
17 $hasil_preprocessing = implode($new_array,"");
```

Gambar 4.2 Implementasi *Filtering*

4.2.2. Proses Algoritma *Winnowing*

Proses algoritma *Winnowing* terdiri dari empat tahapan, yaitu pemotongan *k-gram*, *hashing*, *windowing*, dan *fingerprinting*.

4.2.2.1. Proses Pemotongan *K-gram*

Tahapan pertama dalam algoritma *Winnowing* ialah pemotongan *k-gram*. Fungsi *k-gram* akan memproses teks hasil *preprocessing* ($teks) dengan nilai *t* inputan dari pengguna ($gram). Proses dapat dilihat pada Gambar 4.3

```
1  function kGram($teks, $gram) {
2        $length = strlen($teks);
3        $teksSplit;
4        if ($length < $gram) {
5              $teksSplit[] = $teks;
6        }
7        else {
8              for ($i=0; $i <= ($length - $gram); $i++) {
9                    $teksSplit[] = substr($teks, $i, $gram);
10             }
```

```
11          }
12          return $teksSplit;
13  }
```

Gambar 4.3 Implementasi Proses Pemotongan *K-gram*

Teks akan dipotong sebesar *t*, dan proses tersebut diulang dengan pergeseran 1 karakter. Dan proses akan berhenti pada karakter ke (*panjang teks* – *nilai t*).

4.2.2.2. Proses *Hashing*

Seperti yang sudah dijelaskan pada bab 3, hasil dari pemotongan *k-gram* akan dilanjutkan dengan proses *hashing*, sehingga setiap *k-gram* memiliki nilai *hash*. Fungsi hashing memiliki parameter nilai $string yaitu potongan *k-gram* yang akan diubah, dan $*basis* yaitu nilai bilangan yang diinputkan oleh pengguna di awal aplikasi pada halaman deteksi plagiarisme antar dokumen. Dapat dilihat pada Gambar 4.4

```
1   function hashing($string, $basis) {
2         $pjgKarakter = strlen($string);
3         $hash = 0;
4         for ($i = 0; $i < $pjgKarakter; $i++) {
5               $ascii = ord($string[$i]);
6               $hash += $ascii * pow($basis, $pjgKarakter - ($i
    + 1));
7         }
8         return $hash;
9   }
```

Gambar 4.4 Implementasi Proses *Hashing*

4.2.2.3. Proses *Windowing*

Proses selanjutnya ialah pemotongan *window*. Fungsi *window* memiliki parameter nilai $hash yaitu array yang berisi deretan nilai *hash* dari isi dokumen, dan $w yaitu nilai *window* yang akan dipakai untuk batasan isi banyak *hash* dalam satu *window*. Nilai *window* dimasukkan oleh pengguna. Implementasi dapat dilihat pada Gambar 4.5

```
1   function window($hash, $w) {
2         $bnykArray = count($hash);
3         if ($bnykArray <= $w) {
4               for($i=0; $i<$bnykArray; $i++){
5                     $hashSplit[0][$i] = $hash[$i].",".$i;
6               }
7         }
8         else{
```

```
 9              for ($i=0; $i <= ($bnykArray - $w); $i++) {
10                  $counter = 1;
11                  $k=$i;
12                  for ($j=0; $j<=$bnykArray; $j++){
13                      if ($counter > $w){
14                          break;
15                      }
16                      else{
                            $hashSplit[$i][$j] =
17  $hash[$k].",".$k;
18                      }
19                      $counter++;
20                      $k++;
21                  }
22              }
23          }
24          return $hashSplit;
    }
```

Gambar 4.5 Implementasi Proses Pemotongan *Window*

Hasil proses *window* berupa *array* yang berisi kumpulan *window* yang berisi deretean nilai *hash* dengan Informasi letak *hash* tersebut dalam teks.

4.2.2.4. Proses *Fingerprinting*

Proses *fingerprinting* diawali dengan pemilihan *hash* terkecil pada setiap *window*. Kemudian dilanjutkan dengan pengecekkan nilai *hash* terkecil tersebut, jika pada 2 *window* yang berurutan memiliki nilai *hash* terkecil yang sama dengan index yang sama pula, maka nilai *hash* yang dipilih adalah yang paling kanan. Hasil dari proses inilah yang disebut sebagai *fingerprint* dokumen. Implementasi dapat dilihat pada Gambar 4.6

```
1   function fingerprint($window){
2         //mendapatkan has terkecil per window.
3         for ($i=0; $i<count($window); $i++){
4               for($j=0; $j<count($window[$i]); $j++){
5                   $splitWindow[$i][$j] =
    explode(",",$window[$i][$j]);
6                   if ($j==0){
7                       $tempHash = $splitWindow[$i][$j][0];
8                       $combineWindow =
    implode(",",$splitWindow[$i][$j]);
9                   }
10                  else{
11                      if($tempHash <=
    $splitWindow[$i][$j][0]){
12                          $tempHash = $tempHash;
13                      }
14                      else{
15                          $tempHash =
    $splitWindow[$i][$j][0];
```

```
16                                  $combineWindow =
implode(",",$splitWindow[$i][$j]);
17                              }
18                      }
19              }
20              $minHash[$i] = $combineWindow;
21      }
22
23      //memilih hash paling kanan ketika nilai min hashnya
sama.
24      $temp;
25      for ($i=0;$i<count($minHash);$i++){
26              $split_minHash[$i] = explode(",",$minHash[$i]);
27              if ($i==0){
28                      $temp = $split_minHash[$i][1];
29                      $fingerprint[] = $minHash[$i];
30              }
31              else{
32                      if ($split_minHash[$i][1] != $temp){
33                              $temp = $split_minHash[$i][1];
34                              $fingerprint[] = $minHash[$i];
35                      }
36              }
37      }
38      return $fingerprint;
39  }
40
41
42
```

Gambar 4.6 Implementasi Proses *Fingerprinting*

4.2.3. Proses Perhitungan Nilai *Similarity*

Proses ini bertujuan untuk mendapatkan Informasi mengenai irisan dan gabungan dari dua dokumen, sehingga dapat diperoleh kesamaan atau *similarity* dari kedua dokumen tersebut. Seperti yang sudah dijelaskan pada bab 2, sistem ini menggunakan prinsip dari *Jaccard's Coefficient* untuk menghitung jarak kedekatan atau *similarity* antar 2 dokumen. Implementasi dapat dilihat pada Gambar 4.7

```
1   function similarity ($finger1,$finger2,$kgram,$threshold){
2       $array1 = array_unique($finger1);
3       $array2 = array_unique($finger2);
4       $gabungan = count(array_merge($array1,$array2));
5
6       $irisan = 0;
7       for ($i=$threshold; $i>=$kgram; $i--){
8               for($a=0; $a<count($array1); $a++){
9                       $substringA = substring($i, $array1[$a]);
10                      $e=0;
11                      for($as=0; $as<count($substringA); $as++){
12                              if($e==1){ break;}
13                              for($b=0; $b<count($array2); $b++){
```

```
14                                  $substringB = substring($i,
    $array2[$b]);
15                                  if($e==1){ break;}
16                                  for($bs=0;
17  $bs<count($substringB); $bs++){
18                                          if($substringA[$as] ==
    $substringB[$bs]){
19                                                  $irisan++;
20                                                  $array2[$b] ="";
21                                                  $array1[$a] ="";
22                                                  $e=1;
23                                                  break;
24                                          }
25                                  }
26                          }
27                  }
28          }
29      }
30      $pembilang = $irisan;
31      $penyebut = $gabungan-$pembilang;
32
33      //jaccard coefficient
34      $kesamaan = abs(round($pembilang/$penyebut*100,4));
35      return $kesamaan;
36  }
```

Gambar 4.7 Implementasi Proses Perhitungan Nilai *Similarity*

Proses perhitungan nilai *similarity* ini menggunakan nilai *threshold* dalam implementasinya. Seperti yang sudah dijelaskkan sebelumnya, *threshold* digunakan untuk menjamin terdeteksinya kemiripan antar *fingerprint*. Tentu *threshold* yang digunakan memenuhi syarat, yaitu kemiripan hanya ditentukan dalam batas *guarantee threshold*, untuk *substring* dibawah *noise threshold* akan dianggap sebagai *noise* dan tidak diproses.

4.3. Implementasi Antar Muka

Antar muka aplikasi deteksi plagiarisme ini bertujuan untuk memudahkan pengguna dalam berinteraksi dengan sistem. Antarmuka aplikasi ini terdiri dari halaman awal, halaman *multiple documents*, dan halaman *two documents*.

4.3.1. Tampilan Halaman Awal

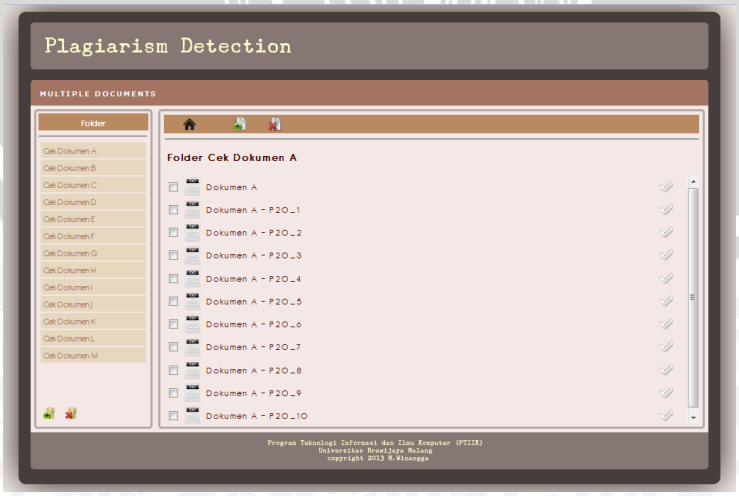
Tampilan awal aplikasi ini memberikan Informasi mengenai sistem ini sendiri, metode apa yang dipakai dan algoritma apa yang digunakan. Halaman ini juga memberikan pilihan kepada pengguna untuk memilih jenis deteksi, *multiple*

documents untuk deteksi banyak dokumen atau *two document* untuk deteksi plagiarisme antar dua teks saja. Berikut halaman awal aplikasi ini ditunjukkan pada Gambar 4. 8.

Gambar 4. 8 Tampilan Halaman Awal

4.3.2. Tampilan Halaman Multiple Documents

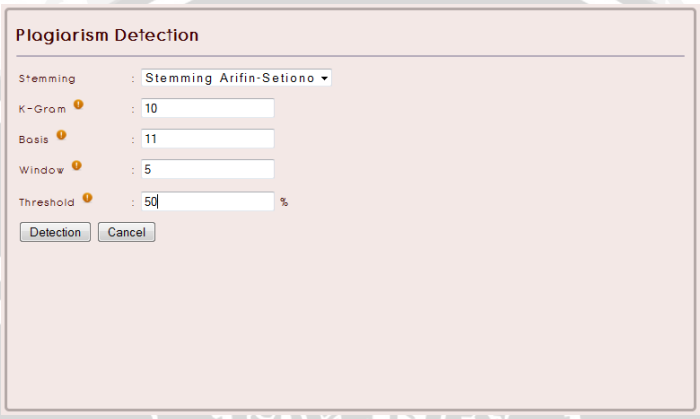
Halaman ini menampilkan daftar folder dari database. Folder berisi dokumen-dokumen yang diambil dari database pula. Halaman ini menyediakan tombol untuk kembali ke halaman awal atau home. Dan tombol untuk men-*delete* folder ataupun dokumen, dan juga memasukkan folder baru ataupun dokumen baru, serta tombol "*checklist*" untuk deteksi plagiarisme dokumen tersebut dengan dokumen lain pada folder tertentu. Tampilan dari halaman *multiple documents* dapat dilihat pada Gambar 4. 9.



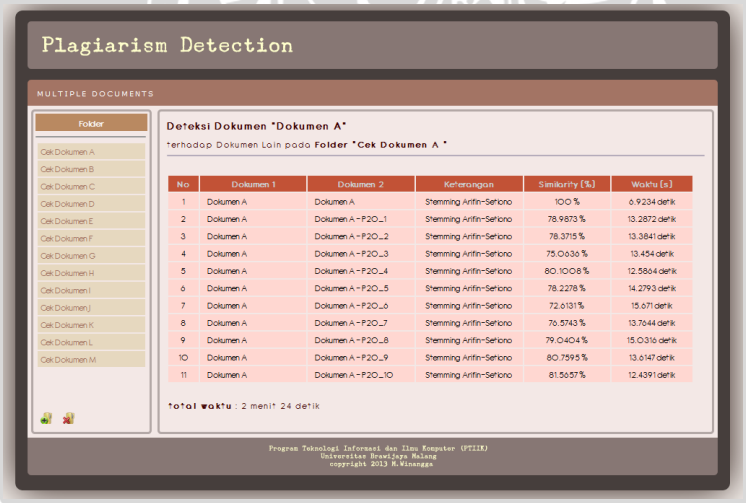
Gambar 4. 9 Tampilan Halaman Multiple Documets

4.3.3. Tampilan Halaman *Plagiarism Detection* untuk *Multiple Documents*

Untuk mendeteksi plagiarisme menggunakan algoritma *Winnowing*, sistem memberikan *form* isian kepada *user* berisi nilai-nilai parameter untuk proses penghitungan. Kemudian akan ditampilkan hasil penghitungan *similarity* antar dokumen tersebut dalam bentuk tabel. Berikut tampilan dari proses tersebut, ditunjukkan oleh Gambar 4. 10 dan Gambar 4. 11.



Gambar 4. 10 Tampilan *Form* Isian *Plagiarism Detection*



Gambar 4. 11 Tampilan Hasil *Plagiarism Detection*

4.3.4. Tampilan Halaman Two Documents

Halaman ini menampilkan *form* untuk memasukkan teks yang akan dicek *similarity*-nya. Teks dapat dimasukkan pada kotak "Teks1" dan kotak "Teks2". Kemudian pengguna dapat mengisi nilai-nilai pada *form* isian untuk deteksi plagiarisme. Keterangan tiap-tiap nilai dapat dilihat saat *mouse* mengarah pada

gambar lingkaran berwarna oren. Hasil dari penghitungan *similarity* juga dapat dilihat pada halaman ini setelah menekan tombol "deteksi". Berikut tampilan dari halaman *two documents* dapat dilihat pada Gambar 4. 12 dan tampilan hasil deteksi pada Gambar 4. 13.

Gambar 4. 12 Tampilan *Two Documents*

Gambar 4. 13 Tampilan Hasil *Plagiarism Detection* pada *Two Documents*

BAB V
PENGUJIAN DAN ANALISIS

Bab ini akan membahas mengenai pengujian dari aplikasi yang telah dikerjakan serta menganalisis hasilnya. Pengujian sistem dilakukan untuk menguji kelayakan sistem deteksi plagiarisme pada dokumen teks yang telah dibuat dan menunjukkan bahwa sistem yang telah dibuat dapat bekerja dengan baik sesuai dengan spesifikasi sistem yang telah ditentukan.

5.1. Pengujian nilai *k-gram*, *threshold*, dan *basis*.

Pengujian dilakukan terhadap 3 parameter nilai, yaitu *k-gram*, *threshold*, dan *basis* dengan kondisi sistem tanpa menggunakan *stemming*. Pengujian dilakukan berdasarkan skenario yang telah dirancang pada bab 3. Pengujian dilakukan pada 20 dokumen uji terhadap 1420 dokumen latih. Dokumen latih merupakan dokumen uji yang telah diberikan delapan perlakuan, yaitu 100% sama, 80% sama, 60% sama, 40% sama, 20% sama, tukar 20%, tukar 60%, dan ubah 10%. Perlakuan 100%sama, 80%sama, 60%sama, dan 20%sama merupakan pemotongan kata sebesar prosentase yang disebutkan terhadap dokumen uji. Perlakuan tukar 20% dan tukar60% merupakan pertukaran posisi kalimat pada dokumen uji sebesar prosentase yang disebutkan. Dan perlakuan ubah 10% merupakan pengubahan 10% kata kerja dari dokumen. Pengubahan tersebut ialah mengubah kata kerja berimbuhan dari kata kerja aktif menjadi kata kerja pasif ataupun sebaliknya tanpa mengubah posisi subjek dan objek dari kalimat. Setelah mendapatkan delapan perlakuan tersebut kemudian dokumen latih tersebut di-*random* sebanyak sepuluh kali, kecuali pada perlakuan 100%sama, sehingga menghasilkan 1420 dokumen latih. Hasil pengujian merupakan rata-rata prosentase *similarity* dari hasil perbandingan 20 dokumen uji terhadap 1420 dokumen latih.

Hasil pengujian nilai k-gram terhadap prosentase *similarity* dan prosentase *error* dapat dilihat pada Tabel 5.1 dan Tabel 5.2.

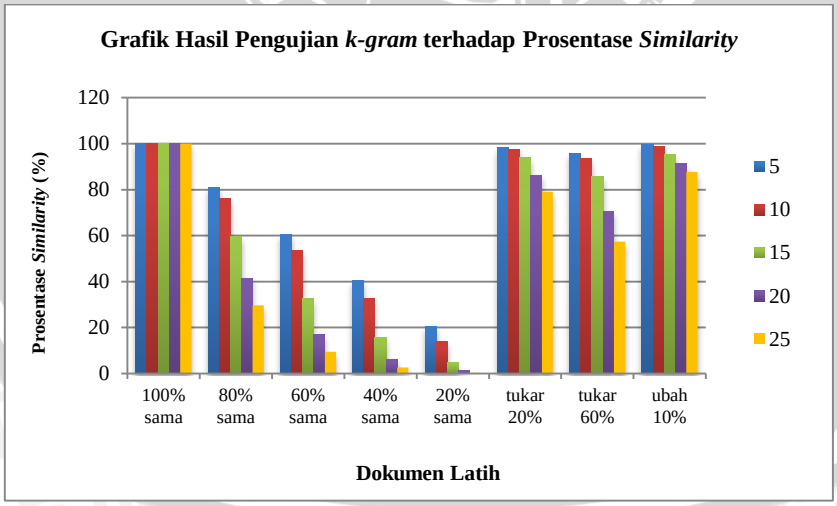
Tabel 5.1 Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Similarity*

Rata-rata Prosentase Simirality								
k	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
5	100	80.7251	60.6252	40.5649	20.2988	98.5236	95.778	99.3749
10	100	76.1866	53.4442	32.7133	13.9238	97.6582	93.4616	98.883
15	100	59.5588	32.3514	15.6953	4.7452	93.832	85.4676	95.3276
20	100	41.4066	17.0652	6.1594	1.1651	86.2986	70.2897	91.3912
25	100	29.5892	9.3798	2.5392	0.2544	79.1277	57.3483	87.7395

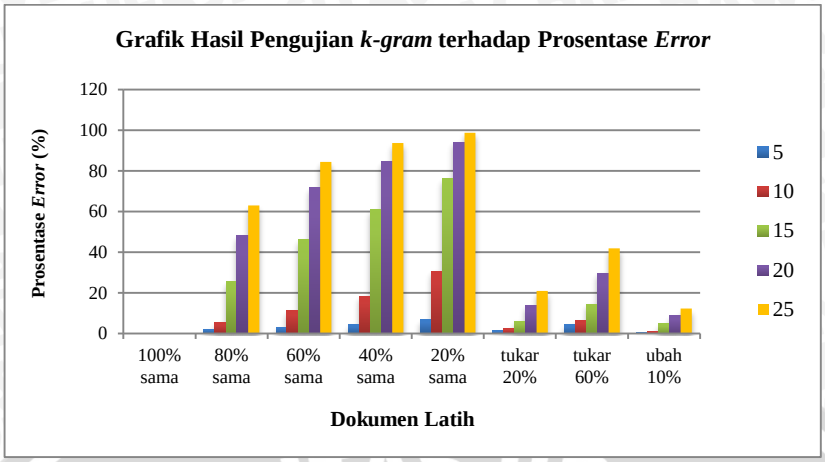
Tabel 5.2 Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Error*

Rata-rata Prosentase *Error*								
k	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
5	0	2.1846	3.2072	4.3133	6.7768	1.4764	4.2220	0.6251
10	0	5.3785	11.1826	18.3876	30.7460	2.3419	6.5384	1.1170
15	0	25.5515	46.0810	60.7618	76.2742	6.1680	14.5324	4.6724
20	0	48.2417	71.5580	84.6016	94.1744	13.7014	29.7103	8.6088
25	0	63.0135	84.3670	93.6519	98.7282	20.8724	41.9531	12.2605

Tabel 5.1 dan Tabel 5.2 dapat disajikan dalam bentuk grafik berikut ini:



Gambar 5.1 Grafik Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Similarity*



Gambar 5.2 Grafik Hasil Pengujian Nilai *k-gram* terhadap Prosentase *Error*

Grafik pada Gambar 5.1 dan Gambar 5.2, ditunjukkan bahwa nilai yang paling optimal untuk parameter *k-gram* adalah 5. Prosentase *error* yang dihasilkan merupakan yang paling kecil jika dibandingkan dengan hasil pada nilai parameter 10, 15, 20, dan 25. Prosentase *similarity* yang diperoleh merupakan yang paling baik, sehingga nilai parameter *k-gram* = 5, yang akan digunakan untuk pengujian *threshold*.

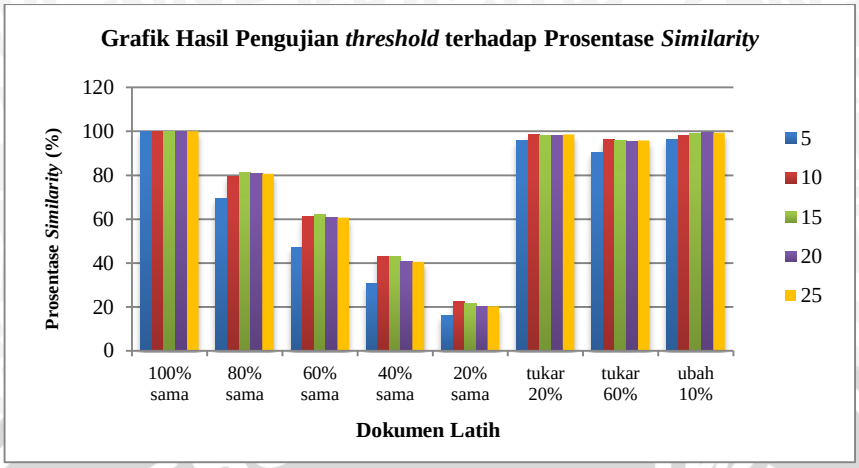
Tabel 5.3 Hasil Pengujian Nilai *threshold* terhadap Prosentase *Similarity*

Rata-rata Prosentase *Simirality*								
t	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
5	100	69.3971	47.1411	30.8428	15.8429	95.7562	90.4806	96.0822
10	100	79.4289	61.0507	43.1403	22.4174	98.4782	96.2835	98.1803
15	100	81.0659	62.2457	42.8718	21.6409	98.3586	95.6372	99.1657
20	100	80.6063	60.8692	40.7725	20.4565	98.1604	95.2917	99.2796
25	100	80.7251	60.6252	40.5649	20.2988	98.5236	95.7780	99.3749

Tabel 5.4 Hasil Pengujian Nilai *threshold* terhadap Prosentase *Error*

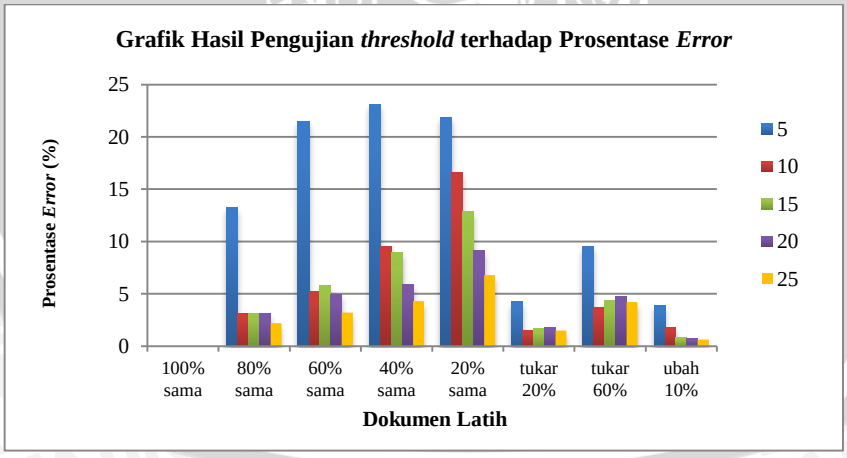
Rata-rata Prosentase *Error*								
t	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
5	0	13.2616	21.4315	23.0583	21.8378	4.2438	9.5194	3.9178
10	0	3.1015	5.2305	9.5806	16.5851	1.5218	3.7165	1.8197
15	0	3.1397	5.7782	8.9288	12.8403	1.6414	4.3628	0.8343
20	0	3.1520	4.9664	5.9158	9.1849	1.8396	4.7083	0.7204
25	0	2.1846	3.2072	4.3133	6.7768	1.4764	4.2220	0.6251

Tabel 5.3 dan Tabel 5.4 dapat disajikan dalam bentuk grafik berikut ini:



Gambar 5.3 Grafik Hasil Pengujian Nilai *threshold* terhadap Prosentase *Similarity*

Pada Gambar 5.3 dan Gambar 5.4 ditunjukkan bahwa nilai *threshold* yang paling baik ialah 25. Hal ini terlihat dari prosentase *error* yang dihasilkan merupakan yang paling kecil jika dibandingkan dengan nilai yang lain. Dari beberapa perlakuan, rata-rata *similarity* yang diperoleh dari nilai parameter 25, lebih baik jika dibandingkan dengan nilai yang lain. Oleh sebab itu, maka nilai 25 yang akan digunakan untuk *threshold* pada pengujian *basis*.



Gambar 5.4 Grafik Hasil Pengujian Nilai *threshold* terhadap Prosentase *Error*

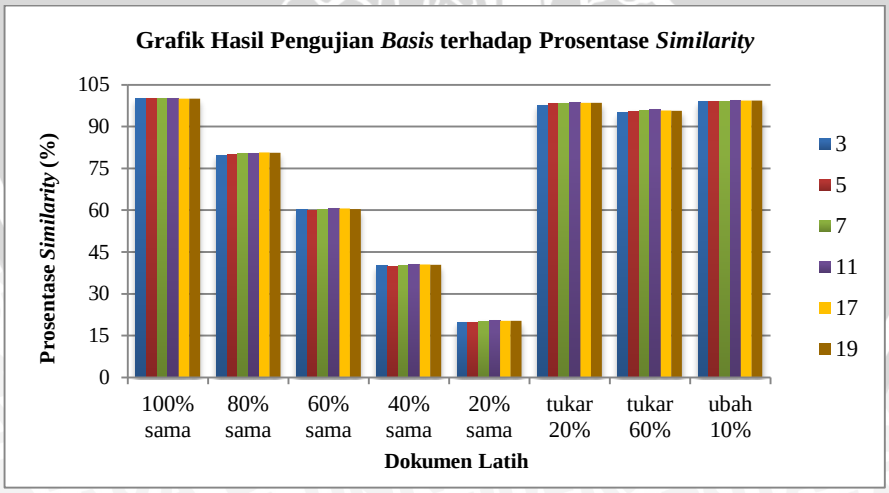
Tabel 5.5 Hasil Pengujian Nilai *basis* terhadap Prosentase *Similarity*

Rata-rata Prosentase *Simirality*								
b	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
3	100	79.8345	60.2203	40.0220	19.8378	97.7431	95.1166	98.9146
5	100	80.0889	59.9242	39.8726	19.8073	98.1438	95.5551	99.0918
7	100	80.3523	60.3113	40.1597	20.2395	98.3806	95.7809	99.0110
11	100	80.4041	60.4671	40.4711	20.3970	98.6746	96.1662	99.1717
17	100	80.7251	60.6252	40.5649	20.2988	98.5236	95.7780	99.3749
19	100	80.5695	60.4274	40.4144	20.2689	98.5031	95.6392	99.3475

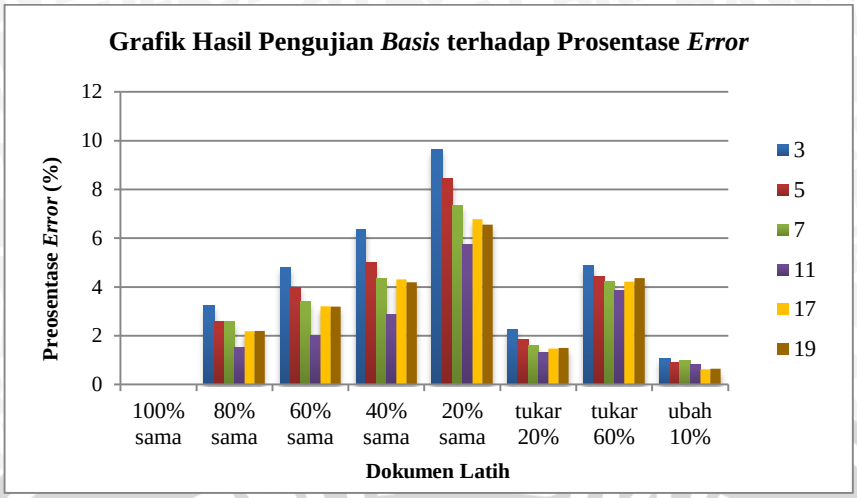
Tabel 5.6 Hasil Pengujian Nilai *basis* terhadap Prosentase *Error*

Rata-rata Prosentase *Error*								
b	**100% sama**	**80% sama**	**60% sama**	**40% sama**	**20% sama**	**tukar 20%**	**tukar 60%**	**ubah 10%**
3	0	3.2207	4.8113	6.3434	9.6320	2.2569	4.8834	1.0854
5	0	2.6003	3.9925	5.0191	8.4495	1.8562	4.4449	0.9082
7	0	2.5921	3.4013	4.3475	7.3472	1.6194	4.2191	0.9890
11	0	1.5070	2.0326	2.8683	5.7579	1.3254	3.8338	0.8283
17	0	2.1846	3.2072	4.3133	6.7768	1.4764	4.2220	0.6251
19	0	2.1952	3.1954	4.1980	6.5500	1.4969	4.3608	0.6525

Tabel 5.5 dan Tabel 5.6 dapat disajikan dalam bentuk grafik berikut ini:



Gambar 5. 5 Grafik Hasil Pengujian Nilai *basis* terhadap Prosentase *Similarity*



Gambar 5. 6 Grafik Hasil Pengujian Nilai *basis* terhadap Prosentase *Error*

Pengujian tahap ini diakhiri dengan uji coba pada nilai *basis*, hasil terlihat pada Gambar 5. 5 dan Gambar 5. 6. Jika dilihat dari rata-rata prosentase *similarity*, hasil yang didapat tidak terlalu menunjukkan perbedaan yang signifikan. Namun jika dilihat dari rata-rata prosentase *error*, *basis* = 11 memperoleh prosentase *error* yang lebih rendah jika dibandingkan dengan nilai yang lainnya.

Sehingga kombinasi nilai *k-gram*, *threshold*, dan *basis* yang paling optimal ialah *k-gram* = 5, *threshold* = 25, dan *basis* = 11. Kombinasi tersebut yang akan digunakan untuk pengujian *similarity*, lama waktu eksekusi, dan *error* sistem.

5.2. Pengujian *Similarity*, Waktu Eksekusi, dan *Error* Sistem

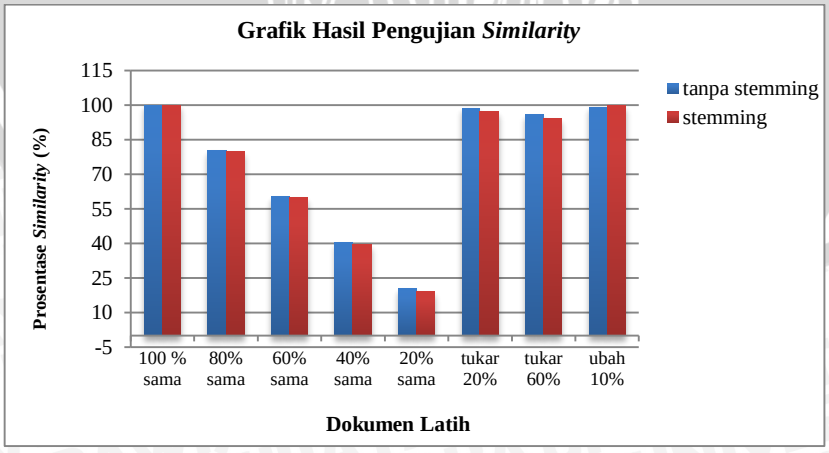
Pengujian *Similarity* dan Waktu Eksekusi ialah membandingkan hasil prosentase *similarity* dan lama waktu antara sistem dengan *stemming* Arifin-Setiono dan sistem tanpa menggunakan *stemming*. Pengujian dilakukan pada 20 dokumen uji terhadap 1420 dokumen latih. Dokumen latih merupakan dokumen uji yang telah diberikan delapan perlakuan, yaitu 100% sama, 80% sama, 60% sama, 40% sama, 20% sama, tukar 20%, tukar 60%, dan ubah 10%. Perlakuan 100%sama, 80%sama, 60%sama, dan 20%sama merupakan pemotongan kata sebesar prosentase yang disebutkan terhadap dokumen uji. Perlakuan tukar 20% dan tukar60% merupakan pertukaran posisi kalimat pada dokumen uji sebesar

prosentase yang disebutkan. Dan perlakuan ubah 10% merupakan pengubahan 10% kata kerja dari dokumen. Pengubahan tersebut ialah mengubah kata kerja berimbuhan dari kata kerja aktif menjadi kata kerja pasif ataupun sebaliknya tanpa mengubah posisi subjek dan objek dari kalimat. Setelah mendapatkan delapan perlakuan tersebut kemudian dokumen latih tersebut di-random sebanyak sepuluh kali, kecuali pada perlakuan 100%sama, sehingga menghasilkan 1420 dokumen latih. Hasil pengujian merupakan rata-rata prosentase similarity dari hasil perbandingan 20 dokumen uji terhadap 1420 dokumen latih. Hasil pengujian ditunjukkan pada Tabel 5.7.

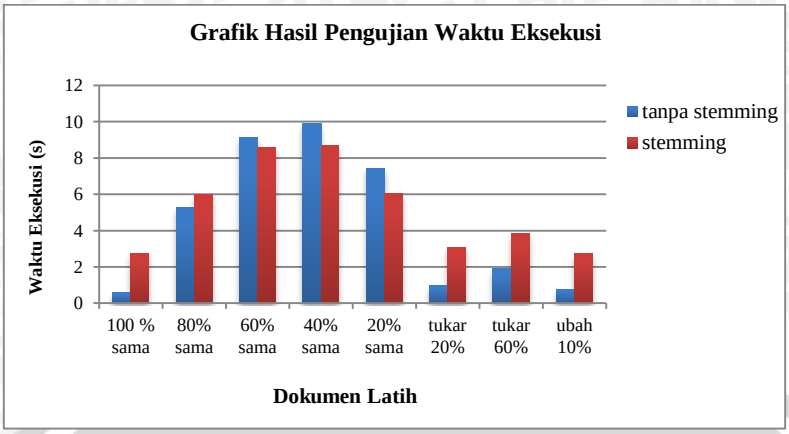
Tabel 5.7 Hasil Pengujian *Similarity*, Waktu eksekusi, dan *Error* Sistem

Dokumen	Rata-rata Prosentase Simirality		Rata-rata waktu		Rata-rata Prosentase *Error*	
	tanpa *stemming*	*stemming*	tanpa *stemming*	*stemming*	tanpa *stemming*	*stemming*
100 % sama	100	100	0.5673	2.7632	0	0
80% sama	80.4041	79.7606	5.2790	6.0011	1.5070	3.8681
60% sama	60.4671	60.0410	9.1224	8.6009	2.0326	6.0549
40% sama	40.4711	39.6392	9.8823	8.6693	2.8683	7.6450
20% sama	20.3970	19.0659	7.3985	6.0509	5.7579	12.4145
tukar 20%	98.7854	97.3143	0.9775	3.0580	1.2146	2.6857
tukar 60%	96.0373	94.1376	1.8865	3.8398	3.9627	5.8624
ubah 10%	99.1717	99.9163	0.7732	2.7367	0.8283	0.0837

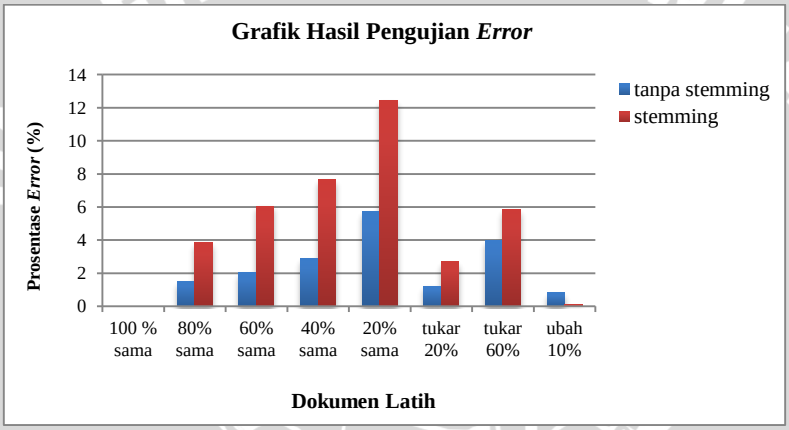
Tabel 5.7 dapat disajikan dalam bentuk grafik berikut ini:



Gambar 5.7 Grafik Hasil Pengujian *Similarity*



Gambar 5.8 Grafik Hasil Pengujian Waktu Eksekusi



Gambar 5.9 Grafik Hasil Pengujian *Error*

Berdasarkan hasil pada Tabel 5.7, prosentase *similarity* yang dihasilkan tidak persis sama dengan prosentase pemotongan kata, misal untuk perlakuan 80% sama, prosentase yang dihasilkan tidak persis sebesar 80%, begitu pula dengan jenis perlakuan yang lain. Hal ini dikarenakan dokumen mengalami proses *stopword removal* atau penghilangan kata-kata tidak penting. Faktor lain yang mempengaruhi prosentase kemiripan dokumen ialah pemotongan kata yang dilakukan secara *random*, begitu pula pada perlakuan tukar kalimat dan ubah kata kerja berimbuhan. *Random* pada kasus ini ialah pemotongan kata tidak berdasar pada kata penting saja, melainkan kemungkinan kata yang dihilangkan adalah kata yang termasuk *stopword*. Sehingga, prosentase kemiripan bisa persis 80%, lebih dari 80% atau kurang dari 80%. Hal ini terjadi juga pada jenis perlakuan yang lainnya.

Sebagaimana yang dapat dilihat pada Gambar 5.7, grafik tidak menunjukkan perbedaan yang signifikan antara sistem tanpa dan dengan menggunakan *stemming*. Namun demikian, prosentase *similarity* yang dihasilkan oleh sistem dengan *stemming* relatif lebih kecil. Tetapi, pada perlakuan ubah 10%, sistem dengan *stemming* menghasilkan prosentase *similiraty* yang lebih tinggi. Hal ini menunjukkan bahwa *stemming* berhasil meningkatkan prosentase *similarity* pada perlakuan ubah10% kata kerja berimbuhan, namun hal ini tidak terjadi pada jenis perlakuan yang lain.

Hasil waktu eksekusi yang terlihat pada Gambar 5.8 menunjukkan bahwa rata-rata lama waktu eksekusi untuk sistem dengan *stemming* relatif lebih lama dibandingkan tanpa menggunakan *stemming*. Walaupun terlihat pada jenis perlakuan 60% sama, 40% sama, dan 20% sama bahwa waktu eksekusi sistem dengan *stemming* lebih cepat. Hal ini dikarenakan akumulasi waktu yang tidak jauh berbeda antara proses dengan *stemming* maupun tidak. Proses *stemming* membutuhkan waktu lebih lama pada *preprocessing*, namun demikian ketika sistem menggunakan *stemming* maka jumlah karakter teks akan berkurang, sehingga proses *winnowing* dan penghitungan *similarity* akan lebih cepat daripada sistem tanpa menggunakan *stemming*. Hal ini menyebabkan hasil waktu dengan atau tanpa *stemming* tidak jauh berbeda.

Hasil rata-rata prosentase *error* sistem tanpa menggunakan *stemming* relatif lebih rendah daripada menggunakan *stemming*, dapat dilihat pada Gambar 5.9. Tetapi, pada perlakuan ubah 10%, hasil menunjukkan bahwa sistem dengan *stemming* menghasilkan prosentase *error* yang lebih rendah dibandingkan dengan sistem tanpa menggunakan *stemming*. Perlakuan ubah 10% merupakan perubahan 10% kata kerta berimbuhan menjadi bentuk sebaliknya, yaitu dari kata kerja aktif menjadi pasif ataupun sebaliknya. Perubahan tersebut dilakukan pada dokumen tanpa mengganti posisi subjek, predikat, ataupun objeknya. Hal ini membuktikan bahwa *stemming* hanya berhasil meningkatkan prosentase *similarity* pada teks tanpa perubahan posisi kata atau kalimat, namun ketika terjadi pemotongan kata ataupun pertukaran posisi kalimat secara *random*, *stemming* tidak berhasil meningkatkan prosentase *similarity*.

5.3. Analisis Hasil

Hasil analisa penelitian ini terdiri dari dua bahasan, yaitu hasil uji nilai *k-gram, threshold, basis* dan hasil uji *similarity,* waktu eksekusi, dan *error* sistem.

1. Hasil Uji Nilai *k-gram, threshold,* dan *basis*

Pengujian ini bertujuan untuk menganalisa nilai *k-gram, threshold,* dan *basis* yang optimal dalam mendeteksi adanya tindakan plagiat pada dokumen teks. Pengujian dilakukan pada 20 dokumen uji terhadap 1420 dokumen latih. Dokumen latih merupakan dokumen uji yang telah diberikan delapan perlakuan, yaitu 100% sama, 80% sama, 60% sama, 40% sama, 20% sama, tukar 20%, tukar 60%, dan ubah 10%. Perlakuan 100%sama, 80%sama, 60%sama, dan 20%sama merupakan pemotongan kata sebesar prosentase yang disebutkan terhadap dokumen uji. Perlakuan tukar 20% dan tukar60% merupakan pertukaran posisi kalimat pada dokumen uji sebesar prosentase yang disebutkan. Dan perlakuan ubah 10% merupakan pengubahan 10% kata kerja dari dokumen. Pengubahan tersebut ialah mengubah kata kerja berimbuhan dari kata kerja aktif menjadi kata kerja pasif ataupun sebaliknya tanpa mengubah posisi subjek dan objek dari kalimat. Setelah mendapatkan delapan perlakuan tersebut kemudian dokumen latih tersebut di-random sebanyak sepuluh kali, kecuali pada perlakuan 100%sama, sehingga menghasilkan 1420 dokumen latih. Hasil pengujian merupakan rata-rata prosentase similarity dari hasil perbandingan 20 dokumen uji terhadap 1420 dokumen latih.

Hasil uji nilai *k-gram* menunjukkan perbedaan yang cukup signifikan. Hal ini memperlihatkan bahwa peran *k-gram* dalam menghasilkan *fingerprint* dokumen cukup penting. Seperti yang dapat dilihat pada Tabel 5.2, nilai parameter 5 untuk *k-gram* berturut-turut menghasilkan prosentase *error* terendah untuk semua perlakuan. Namun berbeda dengan nilai parameter 25. Ternyata nilai 25 terlalu besar dan menyebabkan *error* yang sangat tinggi dengan *error* maksimal sebesar 98,7282% pada perlakuan potong kata, 41,9531% pada perlakuan tukar kalimat, dan 12,2605% pada perlakuan ubah kata kerja berimbuhan.

Semakin kecil nilai *k-gram* maka semakin baik prosentase *similarity* yang dihasilkan, hal ini terlihat pada Gambar 5.1. Hasil prosentase *error* juga memperlihatkan bahwa semakin kecil nilai *k-gram* maka semakin kecil pula prosentase *error* yang didapatkan, hal ini dapat dilihat pada Gambar 5.2. Berdasarkan kedua hasil tersebut dapat disimpulkan bahwa semakin rendah nilai *k-gram* maka akan memperoleh *similarity* yang baik dan *error* yang rendah.

Semakin kecil nilai *k-gram*, maka *error* sistem yang dihasilkan semakin rendah, dan prosentase *similarity* yang dihasilkan semakin tepat. Hal ini dikarenakan *k-gram* merupakan ambang batas bawah atau disebut sebagai *noise threshold* dalam menentukan kesamaan antar *fingerprint* dokumen. Semakin kecil nilai *k-gram*, maka pengecekan kesamaan antar *fingerprint* dokumen semakin banyak dan teliti.

Pengujian *threshold* menunjukkan bahwa nilai 25 untuk *threshold* menghasilkan prosentase *error* yang relatif rendah jika dibandingkan dengan nilai lainnya, dapat dilihat pada Tabel 5.4. Prosentase *error* maksimal diperoleh pada nilai parameter 5, yaitu sebesar 23.0583%. Hal ini menunjukkan bahwa nilai yang terlalu rendah untuk *threshold* akan menghasilkan *error* yang tinggi. Hal ini disebabkan karena *threshold* berperan dalam menentukan nilai *window*. Semakin kecil nilai *threshold* maka semakin kecil pula nilai *window*, sehingga memperkecil jarak ditemukannya *fingerpint*. Misal pada threshold = 25 dengan k-gram = 5, maka diperoleh *window* sebesar t-k+1 = 21, maka dalam 21 *hash* setidaknya akan diperoleh satu *fingerprint*. Namun demikian, jika terjadi *worstcase*, *hash* terkecil akan bertahan sepanjang 21*x*21 *hash*, jumlah *fingerprint* yang dihasilkan akan semakin kecil.

Semakin kecil nilai *window*, maka semakin meningkatkan jumlah *fingerprint* yang didapat. Pada kasus ini kinerja sistem terbaik diperoleh pada nilai 25 untuk *threshold*. Penggunaan nilai 5 pada *threshold* meningkatkan jumlah *fingerprint* yang dihasilkan, namun juga menyebabkan perbandingan jumlah *fingerprint* yang dihasilkan antara kedua dokumen tidak seperti yang diharapkan. Sedangkan untuk nilai 25, ternyata nilai ini cukup tepat dalam menghasilkan perbandingan *fingerprint* antara kedua dokumen, sehingga prosentase *similarity* yang didapat semakin tepat dan *error* sistem semakin kecil.

Pengujian nilai *basis* terhadap *similarity*, prosentase yang dihasilkan relatif sama, dapat dilihat pada Gambar 5. 5. Hal ini memperlihatkan bahwa nilai *basis* tidak cukup berpengaruh dalam mendeteksi adanya tindakan plagiat. Namun demikian, pada pengujian *basis* terhadap prosentase *error* menunjukkan bahwa nilai 11 memperoleh hasil yang relatif rendah.

2. Hasil Pengujian *Similarity*, Waktu Eksekusi, dan *Error* Sistem

Pada pengujian *similarity*, prosentase yang dihasilkan tidak persis sama dengan prosentase setelah pemotongan kata, misal untuk perlakuan 80%sama, prosentase yang dihasilkan tidak persis sebesar 80. Hal ini dikarenakan dokumen mengalami proses pemotongan kata secara *random*, dalam kasus ini tidak hanya kata penting saja yang terpotong melainkan terdapat kemungkinan kata yang dihilangkan merupakan kata *stopword*. Ketika proses *filtering* saat *preprocessing*, teks dimungkinkan untuk berkurang ataupun tetap, sehingga prosentase *similarity* bisa tetap, kurang atau lebih dari 80%.

Faktor lain yang mempengaruhi hasil prosentase *similarity* ialah perubahan posisi kata maupun kalimat setelah pemberian perlakuan seperti: pemotongan kata, pertukaran posisi kalimat, dan pengubahan jenis kata kerja berimbuhan. Pengubahan susunan kata ataupun kalimat yang dilakukan secara *random* menyebabkan hasil prosentase *similarity* kurang tepat. Hal ini dikarenakan algoritma *Winnowing* bekerja berdasarkan *k-gram* dalam memperoleh *fingerprint* dokumen. Cara kerja *k-gram* berdasarkan pada susunan karakter, sehingga ketika susunan berubah maka akan memperkecil kemiripan antar *fingerprint* dokumen.

Pengujian *similarity* tidak menunjukkan perbedaan yang signifikan antara sistem tanpa dan dengan menggunakan *stemming*, dapat dilihat pada Gambar 5.7. Namun demikian, prosentase *similarity* yang dihasilkan saat sistem menggunakan *stemming* relatif lebih rendah. Hal ini terjadi dikarenakan algoritma *winnowing* bekerja berdasarkan *substring* atau kumpulan karakter yang disebut sebagai *k-gram*. Proses pemotongan *k-gram* sangat memperhatikan susunan karakter, sehingga pengubahan susunan karakter akan sangat berpengaruh pada *fingerprint* dokumen yang akan dihasilkan. Ketika sistem menggunakan *stemming* maka

jumlah karakter teks akan semakin berkurang dan susunan karakter akan semakin berubah, sehingga akan memperkecil kemiripan antar dokumen dibandingkan dengan sistem tanpa *stemming*.

Hasil waktu eksekusi menunjukkan bahwa rata-rata lama waktu eksekusi untuk sistem yang menggunakan *stemming* relatif lebih lama dibandingkan dengan tanpa menggunakan *stemming*, dapat dilihat pada Gambar 5.8. Namun demikian, pada jenis perlakuan 60%sama, 40%sama, dan 20%sama menunjukkan bahwa waktu eksekusi sistem dengan *stemming* lebih cepat. Hal ini dikarenakan akumulasi waktu yang tidak jauh berbeda antara proses dengan *stemming* maupun tidak. Walaupun saat menggunakan *stemming*, sistem membutuhkan waktu lebih lama saat *preprocessing*, namun ketika sistem menggunakan *stemming* maka jumlah karakter pada teks akan berkurang, sehingga proses *winnowing* dan penghitungan *similarity* akan lebih cepat daripada sistem tanpa menggunakan *stemming*. Hal ini menyebabkan hasil waktu dengan atau tanpa *stemming* tidak jauh berbeda.

Hasil perhitungan prosentase *error* menghasilkan prosentase yang tidak seharusnya ada pada dokumen. Perhitungan nilai prosentase *error* didapat dari prosentase *similarity* terhadap prosentase perlakuan. Hasil rata-rata prosentase *error* sistem tanpa menggunakan *stemming* relatif lebih baik daripada menggunakan *stemming*, dapat dilihat pada Gambar 5.9. Hal ini terjadi dikarenakan perubahan posisi kata maupun kalimat setelah pemberian perlakuan seperti: pemotongan kata, pertukaran posisi kalimat, dan pengubahan jenis kata kerja berimbuhan. Pengubahan susunan kata ataupun kalimat yang dilakukan secara *random* memperkecil kemiripan *fingerprint* dokumen sehingga meningkatkan *error* sistem. Hal ini dikarenakan algoritma *Winnowing* bekerja berdasarkan *k-gram* dalam menentukan *fingerprint* dokumen. Proses pemotongan *k-gram* sangat memperhatikan susunan karakter. Ketika sistem menggunakan *stemming*, maka karakter teks akan berkurang dan sususan karakter akan semakin berubah, sehingga kemiripan antar *fingerprint* dokumen menurun dan prosentase *error* meningkat.

BAB VI
PENUTUP

Bab ini akan membahas mengenai kesimpulan dan saran yang dapat diambil dari pembuatan sistem deteksi plagiarisme pada dokumen teks dengan algoritma *Winnowing*.

6.1. Kesimpulan

Kesimpulan dari hasil penelitian sistem deteksi plagiarisme pada dokumen teks menggunakan algoritma *Winnowing* ialah sebagai berikut:

1. Sistem deteksi plagiarisme berdasarkan kesamaan kalimat pada dokumen teks menggunakan algortima *Winnowing* telah diterapkan sesuai dengan perancangan dan dapat digunakan untuk membandingkan kemiripan antar dua dokumen. Kinerja terbaik diperoleh pada konfigurasi parameter nilai kgram=5, *threshold*=25, dan *basis*=11 dengan hasil tingkat prosentase *error* yang berbanding lurus dengan prosentase perubahan dokumen. Proses perbandingan dokumen dilakukan pada dokumen dalam satu folder yang berisikan dokumen yang telah dimasukkan oleh *user* satu per satu.
2. Prosentase *similarity* pada pengujian sistem dengan *stemming* dan tanpa *stemming* relatif sama yaitu dengan perbedaan sebesar 0 - 1.8997%. Sistem menggunakan *stemming* cenderung menghasilkan prosentase *similarity* yang kurang baik dibandingkan dengan sistem tanpa *stemming*. Tetapi sistem dengan *stemming* menghasilkan prosentase *similarity* yang lebih baik pada kasus pengubahan jenis kata kerja berimbuhan dari aktif ke pasif ataupun sebaliknya tanpa mengubah susunan subjek, predikat, maupun objeknya. Hal ini dikarenakan pada pada kasus tersebut posisi kata dan kalimat tetap, sehingga dapat disimpulkan bahwa *stemming* hanya bekerja dengan baik ketika posisi dokumen tetap.
3. Lama waktu eksekusi pada pengujian sistem dengan *stemming* dan tanpa *stemming* tidak jauh berbeda yaitu dengan perbedaan sebesar 0 sampai 2.1959 detik. Sistem dengan *stemming* cenderung membutuhkan waktu lebih lama dibandingkan dengan sistem tanpa *stemming*.

4. Prosentase *error* yang dihasilkan, sistem dengan *stemming* cenderung menghasilkan prosentase *error* yang lebih tinggi dibandingkan dengan sistem tanpa *stemming*. Tetapi sistem dengan *stemming* menghasilkan prosentase *error* yang lebih kecil pada kasus pengubahan jenis kata kerja berimbuhan dari aktif ke pasif ataupun sebaliknya tanpa mengubah susunan subjek, predikat, maupun objeknya. Hal ini dikarenakan pada pada kasus tersebut posisi kata dan kalimat tetap, sehingga dapat disimpulkan bahwa *stemming* hanya bekerja dengan baik ketika posisi dokumen tetap.

6.2. Saran

Saran dari hasil penelitian ini untuk pengembangan lebih lanjut ialah sebagai berikut:

1. Dapat diteliti lebih lanjut mengenai metode dokumen *fingerprinting* yang berbasis kata, sehingga dapat cocok dengan *stemming*. Serta penambahan pengecekan sinonim kata dan pengecekan berdasarkan makna agar hasil yang didapatkan lebih optimal.
2. Bisa ditambahkan fitur *multiple* input dokumen dan diimplementasikan pada *website e-learning*, sehingga tugas-tugas yang di-*upload* dapat langsung dicek tingkat plagiarismenya dibandingkan dengan tugas teman-temannya dalam satu kelas ataupun lebih.

DAFTAR PUSTAKA

[AGU-02] Agus Zainal Arifin dan Ari Novan Setiono, (2002). *Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering*, *Proceeding of Seminar on Intelligent Technology and Its Applications* (SITIA), Teknik Elektro, Institut Teknologi Sepuluh Nopember.

[AUV-03] Auvil, L. & Searsmith, D., 2003, *Using Text Mining for Spam Filtering*. *Automated Learning Group, National Center for Supercomputing Applications,* University of Illinois.

[ELB-05] Elbegbayan, Norzima. 2005. *Winnowing, a Document Fingerprinting Algorithm*

[FEL-07] Feldman, R. & Sanger, J. (2007). *The Text Mining Handbook*. New York: Cambridge University Press.

[FRA-92] Frakes, W. B. & Baeza, R. (1992). *Information Retrieval Data Structure and Algorithms*. New Jersey: Prentice-Hall.

[HAN-01] Han, J. & Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

[IND-08] Indranandita, Amalia. 2008. *Sistem Klasifikasi dan Pencarian Jurnal dengan menggunakan Metode Naive Bayes dan Vector Space Model*. Universitas Kristen Duta Wacana.

[KAM-13] Kamus Besar Bahasa Indonesia. http://www. kbbi.web.id, diakses pada tanggal 13 Maret 2013.

[KUR-08] Kurniawati, Ana dan Wicaksana, I Wayan Simri. 2008. *Perbandingan Pendekatan Deteksi Plagiarism Dokumen dalam Bahasa Inggris*. Proceeding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008), Auditorium Universitas Gunardarma, Depok, 20-21 Agustus 2008

[MUT-08] Mutiara, Benny; Agustina, Sinta. 2008. *Anti Plagiarsm Application with Algorithm Karp-Rabin at Thesis in Gunadarma University*. Gunadarma University. Depok, Indonesia

[NUG-11] Nugroho, Eko. 2011. *Perancangan Sistem Deteksi Plgiarisme Dokumen Teks dengan menggunakan Algoritma Rabin-Karp*. Malang: Universitas Brawijaya.

[NUR-09] Nurhayati & Sungkar, Lubna A. 2009. *Pengetahuan dan Presepsi Mahasiswa atas Plagiarisme dalam Karya Ilmiah*. Semarang: Universitas Diponegoro Fakultas Ilmu Budaya

[PRI-11] Priantara, I Wayan Surya., Puswitasari, Diana., dan Yuhana, Umi Lailil. 2011. *Implementasi Deteksi Penjiplakan dengan Algoritma Winnowing pada Dokumen Terkelompok*. Surabaya: Institut Teknologi Sepuluh Nopember

[PUR-11] Purwitasari, Diana., Yuwono, Putu., dan Yuhana, Umi Lailil. 2011. *Deteksi Keberadaan Kalimat Sama sebagai Indikasi Penjiplakan dengan Algoritma Hashing Berbasis N-Gram*. Surabaya: Institut Teknologi Sepuluh Nopember

[SCH-03] Schleimer, Saul., Wilkerson, Daniel S., dan Aiken, Alex. 2003. *Winnowing : Local Algorithms for Document Fingerprinting*. San Diego.

[STE-06] Stein, Benno & Eissen, Sven Meyer Zu. 2006. *Fingerprint-based Similarity Search and its Applications*. Bauhaus-Universitat Weimar.

[UDI-94] Udi Manber. 1994. *Finding similar files in a large file system*. In Proceedings of the USENIX Winter 1994 Technical Conference, pages 1–10, San Fransisco, CA, USA, 17–21 1994