

**SISTEM DETEKSI KEMIRIPAN KATA PADA DUA DOKUMEN  
MENGUNAKAN ALGORITMA RABIN-KARP**



**SKRIPSI**

**Disusun Sebagai Salah Satu Syarat  
Untuk Memperoleh Gelar Sarjana Komputer  
Pada Jurusan Ilmu Komputer/ Informatika**

**Disusun oleh :  
ANNIS PRASTYANTI  
J2F009075**

**JURUSAN ILMU KOMPUTER/ INFORMATIKA  
FAKULTAS SAINS DAN MATEMATIKA  
UNIVERSITAS DIPONEGORO  
2014**

## HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini :

Nama : Annis Prastyanti

NIM : J2F009075

Judul : Sistem Deteksi Kemiripan Kata pada Dua Dokumen Menggunakan Algoritma *Rabin-Karp*

Dengan ini saya menyatakan bahwa dalam tugas akhir/ skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan di dalam daftar pustaka.

Semarang, 15 Oktober 2014



Annis Prastyanti

J2F009075

## HALAMAN PENGESAHAN

Judul : Sistem Deteksi Kemiripan Kata pada Dua Dokumen Menggunakan  
Algoritma *Rabin-Karp*  
Nama : Annis Prastyanti  
NIM : J2F009075

Telah diujikan pada sidang Tugas Akhir pada tanggal 1 Oktober 2014 dan dinyatakan lulus  
pada tanggal 14 Oktober 2014

Semarang, 15 Oktober 2014

Mengetahui,

Ketua Jurusan Ilmu Komputer/ Informatika  
SM Universitas Diponegoro



Nurdin Bahtiar, S.Si, MT  
NIP. 19790720 200312 1 002

Panitia Penguji Tugas Akhir  
Ketua,

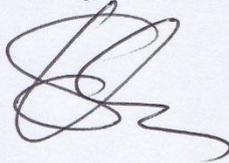
Drs. Kushartantya, MI. Komp.  
NIP. 19500301 197903 1 003

## HALAMAN PENGESAHAN

Judul : Sistem Deteksi Kemiripan Kata pada Dua Dokumen Menggunakan  
Algoritma *Rabin-Karp*  
Nama : Annis Prastyanti  
NIM : J2F009075

Telah diujikan pada sidang tugas akhir pada tanggal 1 Oktober 2014.

Pembimbing Utama,

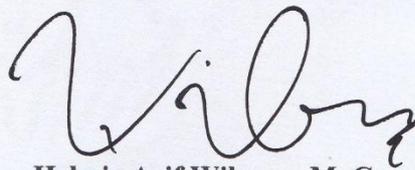


Sukmawati Nur-Endah, S.Si, M.Kom

NIP. 19780502 200501 2 002

Semarang, 15 Oktober 2014

Pembimbing Anggota,



Helmie Arif Wibawa, M. Cs

NIP 19780516 200312 1 001

## ABSTRAK

Maraknya berbagai kasus plagiarisme yang terjadi di beberapa perguruan tinggi menjadi masalah yang harus diatasi. Dengan pencegahan dan pendeteksian dapat mengurangi kemiripan dokumen. Pendeteksian secara manual sangat sulit jika dilakukan, sehingga dibutuhkan pendeteksian secara cepat dan tepat untuk mendeteksi kemiripan dokumen. Oleh karena itu, diperlukan sistem untuk mencocokkan dua dokumen secara terkomputerisasi. Metode yang digunakan untuk pembuatan sistem yaitu metode *String Matching* dengan menggunakan Algoritma *Rabin-Karp*. Algoritma tersebut mencocokkan rangkaian kata *5-gram* yang telah diubah menjadi nilai-nilai *hash*. Nilai hash yang sama menghasilkan persentase kemiripan kata. Pencocokan kalimat sama digunakan untuk mengindikasikan keberadaan kalimat sama. Data uji menggunakan 10 dokumen skripsi dari berbagai sumber yang memiliki keterkaitan judul. Pengujian sistem sebanyak 20 dokumen percobaan menghasilkan tingkat keakuratan sebesar 95%.

Kata Kunci : plagiarisme, Algoritma *Rabin-Karp*, *string matching*

## **ABSTRACT**

The abundant case of plagiarism committed in colleges is a problem which needs to be resolved. Prevention and detections might help to reduce similarity documents. Detection will be very difficult if it's done manually. Rapid and precise detection is essential to identify documents' similarities. Therefore, a computerized system is needed to do a comparison between two documents. The method proposed in the system development is the String Matching which used the Rabin-Karp algorithm. The algorithm compare two sequences of 5-gram words which have been converted to hash values. The same hash values result in the percentage of the word similarities. The same sentence matching is used to indicate the similar sentences. Ten thesis documents from various sources which has related titles are used as the test data. The system testing from 20 experiments documents resulted in 95% of accuracy.

Keyword : plagiarism, Rabin-Karp Algorithm, string matching

## KATA PENGANTAR

Segala puji Penulis ucapkan kehadirat Allah SWT karena limpahan rahmat dan hidayah-Nya Tugas Akhir yang berjudul “**Sistem Deteksi Kemiripan Kata pada Dua Dokumen menggunakan Algoritma Rabin-Karp**” dapat terselesaikan. Penulisan Tugas Akhir ini dimaksudkan untuk memperoleh gelar sarjana strata satu (S1) Jurusan Ilmu Komputer/ Informatika di Fakultas Sains dan Matematika Universitas Diponegoro Semarang.

Dalam penyusunan Tugas Akhir ini, Penulis mendapat bantuan dan dukungan dari banyak pihak. Atas peran sertanya dalam membantu penyelesaian Tugas Akhir ini, Penulis ingin mengucapkan terima kasih kepada :

1. Dr. Muhammad Nur, DEA selaku Dekan FSM Universitas Diponegoro.
2. Nurdin Bahtiar, S.Si., M.T. selaku Ketua Jurusan Ilmu Komputer/ Informatika
3. Indra Waspada, ST. MTI. selaku Koordinator Tugas Akhir Jurusan Ilmu Komputer/ Informatika.
4. Sukmawati Nur Endah, S.Si, M.Kom selaku dosen wali sekaligus dosen pembimbing I yang telah meluangkan waktu, tenaga, dan pikiran untuk membimbing dan mengarahkan Penulis dalam menyelesaikan Tugas Akhir ini.
5. Helmie Arif Wibawa, M. Cs. selaku dosen pembimbing II yang telah meluangkan waktu, tenaga, dan pikiran untuk membimbing dan mengarahkan Penulis dalam menyelesaikan Tugas Akhir ini.
6. Seluruh dosen Jurusan Ilmu Komputer/Informatika Fakultas Sains dan Matematika Universitas Diponegoro yang telah memberikan ilmu pengetahuan kepada Penulis.
7. Semua pihak yang telah membantu, yang tidak dapat Penulis sebutkan satu persatu.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan laporan Tugas Akhir ini, untuk itu Penulis mohon maaf dan mengharapkan saran serta kritik yang membangun dari pembaca. Semoga laporan Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu dan pengetahuan, khususnya pada bidang komputer.

Semarang, 15 Oktober 2014

Penulis

## DAFTAR ISI

HALAMAN JUDUL .....	i
HALAMAN PERNYATAAN KEASLIAN SKRIPSI.....	ii
HALAMAN PENGESAHAN .....	iii
HALAMAN PENGESAHAN .....	iv
ABSTRAK .....	v
ABSTRACT .....	vi
KATA PENGANTAR.....	vii
DAFTAR ISI .....	viii
DAFTAR GAMBAR.....	xii
DAFTAR TABEL .....	xiv
DAFTAR KODE .....	xv
DAFTAR LAMPIRAN .....	xvi
BAB I PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	2
1.3. Tujuan dan Manfaat.....	2
1.4. Ruang Lingkup .....	3
1.5. Sistematika Penulisan.....	3
BAB II DASAR TEORI .....	5
2.1. Plagiarisme .....	5
2.1.1. Bentuk Plagiarisme.....	6
2.1.2. Metode Pendeteksi Plagiarisme.....	7
2.2. Metode Pencocokkan <i>String</i> .....	8
2.2.1. Algoritma pada Metode Pencocokkan <i>string</i> .....	9

2.2.1.	Pencocokkan <i>String</i> dengan Algoritma <i>Rabin-Karp</i> .....	10
2.3.	<i>Rolling Hash</i> .....	14
2.4.	Tahap <i>Text Preprocessing</i> .....	16
2.5.	Persentase Kemiripan .....	19
2.6.	<i>Parsing K-Gram</i> .....	18
2.7.	Tahap Pengembangan Sistem.....	19
2.8.	<i>Data Flow Diagram (DFD)</i> .....	22
2.9.	<i>Entity Relation Diagram (ERD)</i> .....	22
2.10.	<i>Flowchart</i> .....	23
2.11.	MySQL.....	24
2.12.	PHP.....	25
2.13.	Pengukuran dan Kesalahan .....	25
BAB III SPESIFIKASI, ANALISIS, DAN PERANCANGAN .....		27
3.1.	Gambaran Umum Sistem .....	27
3.2.	Spesifikasi dan Analisis Kebutuhan Sistem .....	34
3.2.1.	Spesifikasi Sistem.....	34
3.2.2.	Permodelan Data.....	34
3.2.2.	Permodelan Fungsional .....	35
3.2.2.1.	Data Context Diagram.....	35
3.2.2.2.	DFD level 1 .....	36
3.2.2.3.	DFD level 2 Sub Proses Preprocessing .....	38
3.3.	Desain Sistem .....	39
3.3.1.	Perancangan Proses Sistem Deteksi Kemiripan Kata .....	39
3.3.1.1.	Proses Upload dan Ekstraksi .....	41
3.3.1.2.	Proses Text Preprocessing.....	41
3.3.1.2.1.	Proses <i>Filtering</i> .....	42
3.3.1.2.2.	Proses <i>Whitespace Insensitivity</i> .....	43

3.3.1.2.3.	Proses <i>Parsing 5-gram</i> .....	43
3.3.1.3.	Proses Rolling Hash .....	44
3.3.1.4.	Proses Pencocokkan String dengan Algoritma Rabin-Karp .....	47
3.3.1.5.	Proses Perhitungan Kemiripan .....	48
3.3.1.6.	Proses Pencocokkan Kalimat Sama .....	48
3.3.2.	Perancangan Antarmuka Sistem .....	49
3.3.2.1.	Antarmuka Halaman Depan .....	50
3.3.2.2.	Antarmuka Upload Dokumen .....	50
3.3.2.3.	Antarmuka Pemrosesan Dokumen .....	51
3.3.2.4.	Antarmuka Hasil Pengujian .....	51
3.3.2.5.	Antarmuka Daftar Dokumen yang Diuji .....	52
3.3.2.6.	Antarmuka Stopwords .....	52
3.3.2.7.	Antarmuka Bantuan .....	53
<b>BAB IV</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b> .....	<b>54</b>
4.1.	Implementasi .....	54
4.1.1.	Spesifikasi Perangkat .....	54
4.1.2.	Implementasi Basis Data .....	54
4.1.3.	Implementasi Fungsi .....	55
4.1.3.1.	Fungsi Upload dan Ekstraksi .....	55
4.1.3.2.	Fungsi Text Preprocessing .....	57
4.1.3.3.	Fungsi Rolling Hash .....	58
4.1.3.4.	Fungsi Pencocokkan String .....	59
4.1.3.5.	Fungsi Perhitungan Kemiripan .....	59
4.1.3.6.	Fungsi Pencocokan Kalimat Sama .....	60
4.1.4.	Implementasi Antarmuka .....	61
4.1.4.1.	Implementasi Antarmuka Halaman Depan .....	61
4.1.4.2.	Implementasi Antarmuka Upload Dokumen .....	61

4.1.4.3.	Implementasi Antarmuka Pemrosesan Dokumen .....	62
4.1.4.4.	Implementasi Antarmuka Hasil Pengujian.....	62
4.1.4.5.	Implementasi Antarmuka Riwayat Pengujian.....	63
4.1.4.6.	Implementasi Antarmuka Stopwords .....	63
4.1.4.7.	Implementasi Antarmuka Bantuan.....	64
4.2.	Pengujian .....	65
4.2.1.	Lingkungan Pengujian.....	65
4.2.2.	Rencana Pengujian .....	65
4.2.3.	Pelaksanaan Pengujian .....	67
4.2.4.	Analisis Hasil Pengujian.....	71
BAB V PENUTUP .....		72
5.1.	Kesimpulan.....	72
5.2.	Saran.....	72
DAFTAR PUSTAKA.....		73
LAMPIRAN - LAMPIRAN .....		75

## DAFTAR GAMBAR

Gambar 2.1. Skema Metode Pendeteksi Plagiarisme .....	7
Gambar 2.2. Contoh <i>String Matching</i> .....	10
Gambar 2.3. <i>Fingerprint</i> awal .....	11
Gambar 2.4. Menggeser <i>fingerprint</i> .....	11
Gambar 2.5. Perbandingan kedua .....	12
Gambar 2.6. Perbandingan ketiga.....	12
Gambar 2.7. Perbandingan keempat (nilai <i>hash</i> sama) .....	12
Gambar 2.8. Perbandingan kelima ( <i>string</i> ditemukan).....	13
Gambar 2.9. Contoh <i>Whitespace Insensitivity</i> .....	16
Gambar 2.10. Contoh <i>Parsing</i> .....	16
Gambar 2.11. Contoh <i>Filtering</i> .....	17
Gambar 2.12. Contoh <i>Stemming</i> .....	17
Gambar 2.13. <i>Parsing 5-gram</i> .....	18
Gambar 2.14. Model <i>Waterfall</i> .....	20
Gambar 3.1 Gambaran Umum Sistem Deteksi Kemiripan Kata.....	27
Gambar 3.2 ERD Sistem Deteksi Kemiripan Kata.....	35
Gambar 3.3 <i>Data Context Diagram</i> .....	35
Gambar 3.4. <i>Data Flow Diagram</i> level 1 .....	36
Gambar 3.5. DFD level 2 Proses <i>Preprocessing</i> .....	38
Gambar 3.6. <i>Flowchart</i> Sistem Deteksi Kemiripan pada Dua Dokumen .....	40
Gambar 3.7 <i>Flowchart</i> Proses <i>Upload</i> dan Ekstraksi.....	41
Gambar 3.8. <i>Flowchart</i> Proses <i>Preprocessing</i> .....	42
Gambar 3.9. <i>Flowchart</i> Proses <i>Filtering</i> .....	42
Gambar 3.10. <i>Flowchart</i> Proses <i>Whitespace Insensitivity</i> .....	43
Gambar 3.11. <i>Flowchart</i> Proses <i>Parsing 5-gram</i> .....	44
Gambar 3.12. <i>Flowchart</i> Proses <i>Hashing</i> .....	45
Gambar 3.13. <i>Flowchart</i> Proses <i>Rolling Hash</i> .....	45
Gambar 3.14. <i>Flowchart</i> Proses Pergeseran <i>Hash</i> .....	46
Gambar 3.15 <i>Flowchart</i> Perancangan Proses Pencocokkan String Algoritma <i>Rabin-Karp</i> .....	47
Gambar 3.16. <i>Flowchart</i> Proses Hitung Kemiripan .....	48
Gambar 3.17. <i>Flowchart</i> Proses Pencocokkan Kalimat Sama .....	49

Gambar 3.18. Desain Antarmuka Halaman Depan .....	50
Gambar 3.19. Desain Antarmuka <i>Upload</i> Dokumen .....	50
Gambar 3.20. Desain Antarmuka Pemrosesan Dokumen .....	51
Gambar 3.21. Desain Antarmuka Hasil Pengujian.....	52
Gambar 3.22. Desain Antarmuka Daftar Dokumen yang Diuji .....	52
Gambar 3.23. Desain Antarmuka <i>Stopwords</i> .....	53
Gambar 3.24. Desain Antarmuka Bantuan.....	53
Gambar 4.1. Implementasi Antarmuka Halaman Depan .....	61
Gambar 4.2. Implementasi Antarmuka <i>Upload</i> Dokumen.....	62
Gambar 4.3. Implementasi Antarmuka Pemrosesan Dokumen.....	62
Gambar 4.4. Implementasi Antarmuka Hasil Pengujian .....	63
Gambar 4.5. Implementasi Antarmuka Riwayat Pengujian .....	63
Gambar 4.6. Implementasi Antarmuka <i>Stopwords</i> .....	64
Gambar 4.7. Implementasi Antarmuka Bantuan .....	64

## DAFTAR TABEL

Tabel 2.1 Simbol-simbol pada DFD .....	22
Tabel 2.2 Tabel Notasi <i>Entity Relation Diagram</i> (ERD) .....	23
Tabel 2.3 Simbol-simbol pada <i>Flowchart</i> .....	24
Tabel 3.1. Contoh Dokumen .....	30
Tabel 3.2. Dokumen hasil <i>filtering</i> .....	30
Tabel 3.3 Dokumen hasil <i>whitespace insensitivity</i> .....	30
Tabel 3.4 <i>Term</i> hasil <i>Parsing 5-gram</i> Dokumen Asli .....	31
Tabel 3.5 <i>Term</i> hasil <i>Parsing 5-gram</i> Dokumen Uji .....	31
Tabel 3.6. Nilai ASCII untuk kata “algor” dan “lgori” .....	31
Tabel 3.7 <i>Term</i> hasil <i>Rolling Hash</i> Dokumen Asli .....	32
Tabel 3.8. <i>Term</i> hasil <i>Rolling Hash</i> Dokumen Uji .....	33
Tabel 3.9 Hasil Pencocokkan Kalimat Sama .....	34
Tabel 3.10. Spesifikasi Kebutuhan Fungsional .....	34
Tabel 4.1. Daftar Tabel Sistem Deteksi Kemiripan .....	55
Tabel 4.2. Struktur Tabel DOKUMEN .....	55
Tabel 4.3. Struktur Tabel SIMILARITY .....	55
Tabel 4.4. Struktur Tabel STOPWORDS .....	55
Tabel 4.5 Rencana Pengujian Fungsional Sistem Deteksi Kemiripan .....	66
Tabel 4.6. Data Pengujian Validitas Sistem Deteksi Kemiripan .....	66
Tabel 4.7. Hasil dan Evaluasi Pengujian Sistem Deteksi Kemiripan .....	68
Tabel 4.8 Hasil Pengujian Validitas Sistem Deteksi Kemiripan .....	70

## DAFTAR KODE

Kode 2.1. <i>Pseudocode</i> Algoritma <i>Rabin-Karp</i> .....	13
Kode 4.1. Fungsi <i>Upload</i> dan Ekstraksi .....	56
Kode 4.2. Fungsi <i>Filtering</i> .....	57
Kode 4.3. Fungsi <i>White Insensitivity</i> .....	57
Kode 4.4. Fungsi <i>Parsing 5-gram</i> .....	58
Kode 4.5 Fungsi <i>Rolling Hash</i> .....	59
Kode 4.6. Fungsi Pencocokkan <i>String</i> .....	59
Kode 4.7. Fungsi Perhitungan Kemiripan .....	60
Kode 4.8. Fungsi Pencocokkan Kalimat Sama .....	60

## DAFTAR LAMPIRAN

Lampiran 1. Tabel Daftar <i>Stopwords</i> .....	76
Lampiran 2. Daftar Tanda Baca .....	77
Lampiran 3. Flowchart Sistem Deteksi Kemiripan Kata Pada Dua Dokumen menggunakan Algoritma <i>Rabin-Karp</i> .....	78

# BAB I

## PENDAHULUAN

Bab ini membahas latar belakang, rumusan masalah, tujuan dan manfaat, dan ruang lingkup, serta sistematika penulisan tugas akhir mengenai Sistem Deteksi Kemiripan Kata pada Dua Dokumen menggunakan Algoritma *Rabin-Karp*.

### 1.1. Latar Belakang

Mahasiswa sudah semestinya mempunyai etika dan moral akademik yang kokoh untuk dapat membentuk karakter yang kuat. Para mahasiswa yang tidak memiliki standar nilai moral dan etika yang baik akan memanfaatkan lembaga institusi atau perguruan tinggi sebagai alat untuk menipu banyak hal. Menurut Astuti, dalam penelitiannya menyatakan bahwa salah satu kasus yang merebak di negeri ini adalah maraknya berbagai kasus plagiarisme yang terjadi di beberapa perguruan tinggi baik perguruan tinggi negeri maupun perguruan tinggi swasta (Astuti et al., 2012). Bahkan, Menteri Pendidikan Nasional, Muhammad Nuh menyebut, “*maraknya tindak plagiarisme menunjukkan lemahnya pendidikan karakter, budaya, dan moral insan di dunia akademik*” (Mulyana, 2010). Hal ini menjadi contoh buram dalam potret dinamika pendidikan.

Plagiarisme terjadi dalam karya ilmiah dan tugas akhir mahasiswa dikarenakan memiliki kemiripan yang hampir sama pada karya satu dengan karya lain. Kemiripan sebuah karya ilmiah, karya seni maupun hasil sebuah pencapaian akan mudah bagi seseorang menyebutnya sebagai plagiarisme. Namun, bagi seorang ahli atau yang menggeluti bidang tersebut akan memiliki beberapa argumen sebelum seseorang menyebutnya sebagai plagiarisme atau memang karya atau hal baru. Kemiripan kata khususnya akan sangat membantu dalam menyimpulkan apakah karya ilmiah atau dokumen ilmiah tersebut masuk dalam kategori plagiarisme atau tidak, karena plagiarisme memiliki syarat-syarat khusus untuk sebuah dokumen atau karya ilmiah untuk bisa masuk dalam kategorinya (Djafar. 2014).

Upaya untuk mengatasi masalah kemiripan tidak cukup dengan mengingatkan bahwa tindakan tersebut melanggar hak kekayaan intelektual. Pencegahan dan

pendeteksian merupakan salah satu cara yang paling efektif untuk meningkatkan kualitas pendidikan dan mengurangi kemiripan dokumen satu dengan yang lain. Akan tetapi, pendeteksian sangat sulit jika dilakukan secara manual, sehingga diperlukan suatu sistem cerdas yang mampu mendeteksi kemiripan kata. Pada penelitian Surahman, disarankan untuk mengembangkan algoritma *Rabin-Karp* berbasis web (Surahman, 2013). Sehingga diperlukan sistem untuk mendeteksi karya lebih mudah dengan menggunakan sistem berbasis web. Agar mempermudah pengguna untuk mengecek karya tersebut melalui internet.

Salah satu metode dalam sistem cerdas yang dapat digunakan yaitu metode *string matching* dengan menggunakan Algoritma *Rabin-Karp*. Algoritma *Rabin-Karp* adalah *multiple pattern search* yang sangat efisien untuk mencari *string* dengan pola banyak dan melakukan proses pendeteksian kemiripan kata dengan menghasilkan nilai persentase kemiripan (Nugroho, 2011). Sistem cerdas dengan algoritma *Rabin-Karp* dalam penelitian Nugroho, mampu menampilkan hasil akurasi kemiripan yang sama dan rata-rata proses lebih baik, terutama dokumen teks yang mempunyai *size/* ukuran file yang besar (Nugroho, 2011). Namun dalam penelitian Nugroho, belum menunjukkan keberadaan kalimat yang memiliki kemiripan. Dalam penelitian yang lain, Purwitasari mampu mengindikasi kemiripan kalimat yang sama menggunakan Algoritma Winowwing (Purwitasari et al., 2011). Sehingga pada penelitian ini, mengutip penelitian yang dilakukan Nugroho ditambahkan dengan mengindikasi kalimat sama.

Berdasarkan uraian di atas, dilakukan penelitian tugas akhir berupa pembuatan sebuah sistem yang berguna untuk mendeteksi kemiripan kata pada dua dokumen menggunakan algoritma *Rabin-Karp*.

## **1.2. Rumusan Masalah**

Berdasarkan uraian latar belakang di atas, dapat dirumuskan permasalahan yang dihadapi yaitu bagaimana mengimplementasikan algoritma *Rabin-Karp* untuk sistem deteksi kemiripan kata pada dua dokumen.

## **1.3. Tujuan dan Manfaat**

Tujuan dari penelitian tugas akhir ini adalah mengimplementasikan algoritma *Rabin-Karp* untuk sistem deteksi kemiripan kata pada dua dokumen.

Manfaat dari penelitian tugas akhir ini adalah untuk mempermudah dan mempercepat pengecekan terhadap dokumen yang kemungkinan memiliki kemiripan kata dengan dokumen dari hasil penelitian yang sudah pernah ada, dengan ketelitian yang lebih tinggi dan usaha yang lebih kecil dibandingkan pendeteksian manual.

#### **1.4. Ruang Lingkup**

Ruang lingkup dalam tugas akhir ini, antara lain :

1. Sistem dapat mendeteksi kemiripan kata antara dua dokumen skripsi, terdiri dari pendahuluan, pembahasan, dan kesimpulan.
2. Bagian dokumen teks yang diproses adalah file teks digital yang bersifat plain teks, yaitu *file* teks yang terdiri dari huruf dan angka, tanpa mencakup gambar, tabel dan sejenisnya.
3. Dokumen *input* adalah dokumen yang berformat Ms. Office 2007 ( \*.docx).
4. Sistem hanya memproses dokumen teks berbahasa Indonesia.
5. Nilai *k* pada *parsing k-gram* ditentukan sebanyak 5 ukuran perkata.
6. Sistem ini diimplementasikan berbasis *web*, dengan bahasa pemrograman PHP (*Hypertext Preprocessor*) dan sistem basis data MySQL.

#### **1.5. Sistematika Penulisan**

Sistematika penulisan yang digunakan dalam tugas akhir ini terbagi menjadi beberapa pokok bahasan, yaitu :

##### **BAB I PENDAHULUAN**

Bab ini berisi uraian tentang latar belakang masalah, rumusan masalah, tujuan dan manfaat, ruang lingkup, dan sistematika penulisan tugas akhir mengenai sistem deteksi kemiripan kata pada dua dokumen menggunakan Algoritma *Rabin-Karp*.

##### **BAB II DASAR TEORI**

Bab ini berisi penjelasan singkat konsep-konsep yang mendukung pengembangan sistem, meliputi Plagiarisme, Metode Pencocokkan *String*, *Rolling Hash*, Tahap *Text Preprocessing*, *Parsing K-Gram*, Persentase Kemiripan, Tahap Pengembangan Sistem, *Data Flow Diagram* (DFD),

*Entity Relationship Diagram (ERD), Flowchart, MySQL, PHP, dan Pengukuran dan Kesalahan.*

### BAB III SPESIFIKASI, ANALISIS, DAN PERANCANGAN

Bab ini membahas proses pengembangan sistem pada tahap gambaran umum sistem, spesifikasi dan analisis kebutuhan sistem, dengan hasilnya berupa desain dan rancangan sistem yang dikembangkan.

### BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas hasil pengembangan sistem pada tahap implementasi dan menerangkan rincian pengujian sistem yang dibangun dengan metode *black box*.

### BAB V PENUTUP

Bab ini berisi kesimpulan yang diambil berkaitan dengan sistem yang dibangun dan saran untuk pengembangan lebih lanjut.

## **BAB II**

### **DASAR TEORI**

Bab ini memaparkan dasar teori yang mendukung proses pengembangan sistem meliputi Plagiarisme, Metode Pencocokkan *String*, *Rolling Hash*, Tahap *Text Preprocessing*, *Parsing K-Gram*, Persentase Kemiripan, Tahap Pengembangan Sistem, *Data Flow Diagram* (DFD), *Entity Relationship Diagram* (ERD), *Flowchart*, MySQL, PHP, dan Pengukuran dan Kesalahan.

#### **2.1. Plagiarisme**

Plagiarisme merupakan perbuatan secara sengaja atau tidak sengaja dalam memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu karya ilmiah, dengan mengutip sebagian atau seluruh karya ilmiah orang lain, tanpa menyatakan sumber secara tepat dan memadai istilah tersebut sesuai dengan Permendiknas No 17 tahun 2010, Pasal 1 Ayat 1 (Mulyana, 2010). Sedangkan pendapat Ir. Balza Achmad, M. Sc. E adalah berbuat sesuatu seolah-olah karya orang lain tersebut adalah karya kita dan mengakui hasil karya tersebut milik kita (Astuti et al., 2012). Menurut Moeliono dalam referensi Mulyana, plagiarisme adalah pengambilan karya orang lain, dan dipublikasikan sebagai karya miliknya. Pada kenyataannya, di dunia ilmiah, pengambilan tulisan atau mengutip karya orang lain tersebut, kadang-kadang dianjurkan namun dengan aturan dan norma yang telah berlaku dan disepakati secara luas di dunia akademik. Persoalannya, seberapa besar kadar pengambilan kutipan, bagaimana cara mengutip, dan apakah pihak pengutip menyertakan sumber kutipannya atau tidak adalah sesuatu yang muskil dan sulit ditelusuri (Mulyana, 2010).

Di Indonesia perlindungan hak cipta diatur dalam Undang-Undang Republik Indonesia Nomer 19 Tahun 2002 Tentang Hak Cipta. Oleh karena itu, kegiatan plagiarisme atau yang lebih dikenal dengan kata plagiat harus dihindari. Plagiarisme dapat dianggap sebagai tindak pidana karena mencuri hak cipta orang lain. Di dunia pendidikan, pelaku plagiarisme akan mendapat hukuman berat seperti dikeluarkan dari sekolah/ universitas. Pelaku plagiarisme disebut sebagai plagiator (Astuti et al., 2012).

Plagiarisme tidak hanya melanggar hak cipta atau kepemilikan tetapi tindakan penipuan dan menimbulkan kesalahpahaman mengenai orisinalitas dari penulis yang sebenarnya. Para siswa/ mahasiswa dan peneliti diperbolehkan untuk menciptakan suatu karya baru yang timbul dari pengembangan ide orang lain. Tetapi pemanfaatan ide orang lain tanpa membubuhkan pernyataan sumber merupakan tindakan yang tidak dapat diterima (Mulyana, 2010). Namun, Mulyana menyatakan bahwa cara mencantumkan relevansi mengarahkan mahasiswa untuk melakukan duplikasi atau plagiarisme. Sadar atau tidak, cara mengutip yang dilakukan telah mendekatkan skripsi mereka pada skripsi orang lain. Dari sinilah antara lain gejala plagiarisme muncul (Mulyana, 2010).

### **2.1.1. Bentuk Plagiarisme**

Dalam penelitian Gipp bentuk-bentuk praktek plagiarisme yang sering terjadi seperti yang di bawah ini (Gipp et al., 2011) :

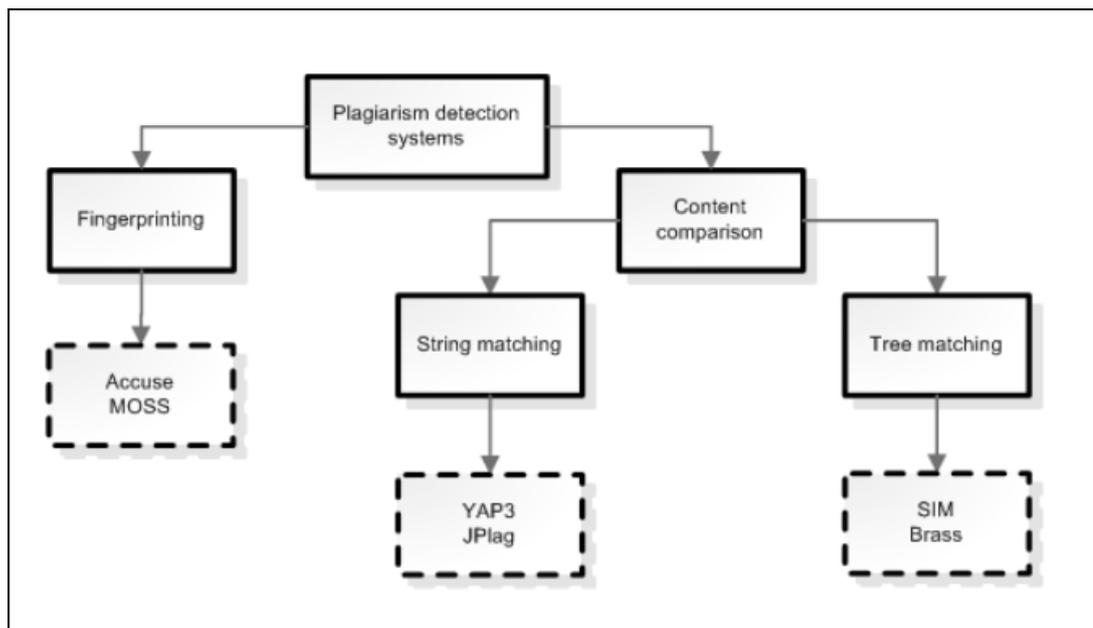
1. *Copy-Paste plagiarism*, menyalin setiap kata tanpa perubahan.
2. *Disguised plagiarism*, plagiarisme yang sering terjadi dibidang akademis tergolong ke dalam praktek menutupi bagian yang disalin, teridentifikasi ke dalam empat teknik, yaitu :
  - a. *Shake-paste*, dengan menyalin dan menggabungkan kalimat atau paragraf dari berbagai sumber dengan sedikit penyesuaian yang diperlukan untuk membentuk teks yang koheren.
  - b. *Expensive plagiarism*, penyisipan kata tambahan.
  - c. *Contractive plagiarism*, menggambarkan ringkasan dengan pemangkasan bahan yang disalin.
  - d. *Mosaic plagiarism*, penggabungan segmen kata dari sumber yang berbeda dengan mengubah urutan kata, mengganti kata-kata dengan sinonim atau menambah/ menghapus kata-kata.
3. *Technical disguise*, teknik meringkas untuk menyembunyikan konten plagiarisme dari deteksi otomatis dengan memanfaatkan kelemahan dari metode analisis teks dasar, misal dengan mengganti huruf dengan simbol huruf asing.
4. *Undue paraphrasing*, sengaja menuliskan ulang pemikiran asing dengan pemilihan kata dan gaya plagiator dengan menyembunyikan sumber asli.

5. *Translated plagiarism*, mengkonversi konten dari satu bahasa ke bahasa lain.
6. *Idea plagiarism*, penggunaan ulang suatu gagasan/ pemikiran asli dari sebuah sumber teks tanpa bergantung bentuk teks sumber.
7. *Self plagiarism*, penggunaan sebagian atau keseluruhan tulisan pribadi yang tidak dibenarkan secara ilmiah.

Bila dilihat dari berbagai macam bentuk-bentuk praktek plagiarisme di atas, dapat disimpulkan bahwa tindakan plagiarisme yang terjadi di dunia akademis lebih cenderung kepada tindakan menggunakan kembali suatu bagian dokumen teks. Kalimat/ kata dari suatu sumber yang tidak mengikuti tata aturan hak cipta, seperti aturan pengutipan maupun ketidakjelasan sumber/ pengarang asli (Purwitasari et al., 2010).

### 2.1.2. Metode Pendeteksi Plagiarisme

Metode pendeteksi plagiarisme dibagi menjadi tiga metode yaitu metode *Fingerprint*, metode *String-Matching*, dan metode *Tree-Matching*. Skema jenis-jenis metode pendeteksi plagiarisme dapat dilihat pada Gambar 2.1 (Muzgovoy, 2007).



Gambar 2.1. Skema Metode Pendeteksi Plagiarisme (Muzgovoy, 2007)

Berikut ini penjelasan dari masing-masing metode pendeteksi plagiarisme (Muzgovoy, 2007) :

1. Metode *Fingerprinting*

Metode *Fingerprinting* merupakan metode yang digunakan untuk mendeteksi beberapa atribut dan struktur dari dokumen. Atribut tersebut diantaranya jumlah kata per baris, jumlah kata unik, dan jumlah kutipan pendek. Contoh sistem yang menggunakan metode *fingerprinting* yaitu MOSS dan Accuse.

2. Metode *String-Matching*

Metode string-matching atau pencocokkan string adalah membandingkan string dokumen dengan string dokumen lain dan dihitung kesamaannya. Contoh sistem yang menggunakan metode pencocokkan string yaitu YAP3 dan Jplag.

3. Metode *Tree-Matching*

Metode *Tree-Matching* adalah dokumen yang dibandingkan harus memiliki aturan struktur yang sama. Contoh sistem yang menggunakan metode *Tree-Matching* yaitu SIM dan Brass.

Pada tugas akhir ini, metode pendeteksi plagiarisme yang digunakan adalah metode pencocokkan string untuk mendeteksi kemiripan kata.

## 2.2. Metode Pencocokkan *String*

Pencocokkan *string* atau *string matching* merupakan suatu metode yang digunakan untuk menemukan suatu keakuratan/ hasil dari satu atau beberapa pola teks yang diberikan. Pencocokkan *string* merupakan pokok bahasan yang penting dalam ilmu komputer karena teks merupakan bentuk utama dari pertukaran informasi antar manusia, misalnya pada literatur, karya ilmiah, dan halaman *web* (Cormen et al., 2009). Persoalan pencarian string dirumuskan sebagai berikut (Cormen et al., 2009) :

1. Teks (*text*), yaitu (*long string*) yang panjangnya  $n$  karakter.
2. *Pattern*, yaitu *string* dengan panjang  $m$  karakter ( $m < n$ ) yang akan dicari di dalam teks.

Pencocokkan *string* digunakan dalam lingkup yang bermacam-macam, misalnya pada pencarian dokumen, pencocokkan DNA *sequences* yang direpresentasikan dalam bentuk *string* dan juga pencocokkan *string* yang dapat dimanfaatkan untuk mendeteksi adanya kemiripan kata dalam karya seseorang. Pencocokkan *string* fokus pada

pencarian satu, atau lebih umum, semua kehadiran sebuah kata (lebih umum disebut *pattern*) dalam sebuah teks.

### 2.2.1. Algoritma pada Metode Pencocokkan *string*

Ada beberapa macam algoritma yang digunakan dalam metode pencocokkan string diantaranya (Adam et al., 2009) :

#### 1. Algoritma *Brute Force*

Algoritma *Brute Force* dalam penerapan pencocokkan string didefinisikan sebagai berikut: Jika diasumsikan teks adalah sebuah table karakter maka *pattern* yang juga merupakan table karakter dengan panjang lebih kecil dari teks, maka algoritma *Brute Force* diimplementasikan sebagai berikut :

- a. Mula-mula mencocokkan *pattern* diawal teks.
- b. Dengan bergerak dari kiri ke kanan bandingkan setiap karakter di dalam *pattern* dengan karakter yang sesuai dengan teks sampai :
  1. Semua karakter yang dibandingkan sesuai dengan *pattern* yang dicari.
  2. Ditemui sebuah ketidakcocokkan karakter.
  3. Bila ditemui ketidakcocokkan karakter pada *pattern* dan teks belum berakhir geser *pattern* satu karakter ke kanan.

#### 2. Algoritma *Knuth Morris Pratt*

Pada algoritma ini memelihara informasi yang digunakan untuk melakukan sejumlah pergeseran. Algoritma ini menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti pada algoritma *Brute Force*.

#### 3. Algoritma *Boyer-Moore*

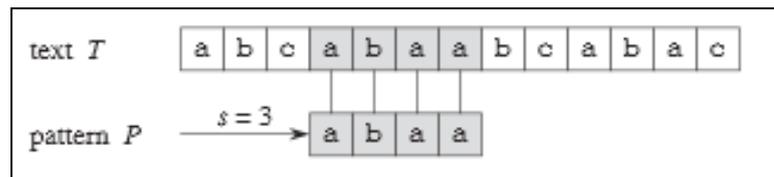
Algoritma ini akan bertambah cepat jika kata yang dicari panjang. Setiap pencocokkan yang gagal antara teks dan kata yang dicari, algoritma ini menggunakan informasi yang didapat dari proses awal untuk melewati karakter-karakter yang tidak cocok.

#### 4. Algoritma *Rabin-Karp*

Algoritma ini adalah algoritma acak sederhana yang cenderung berjalan dalam waktu linier. Pada penelitian Adam, algoritma *Rabin-Karp* adalah pilihan tepat

untuk pencocokkan dengan banyak pola (Adam et al., 2009). Kasus terburuk dari algoritma ini seperti pada algoritma *Brute Force*.

Semua algoritma yang akan dibahas mengeluarkan semua kehadiran pola dalam teks. Sebuah contoh pada Gambar 2.2. menunjukkan pola/ *pattern*  $P = abaa$  dalam *text*  $T = abcabaabcabac$ . Pola ini hanya terjadi satu kali pergeseran(s) dalam setiap teks. Nilai  $s = 3$  karena melakukan pergeseran sebanyak 3 kali. Sebuah garis vertikal yang menghubungkan setiap string dari pola untuk pencocokkan string dalam teks (Cormen et al., 2009). Algoritma *string matching* yang digunakan dalam tugas akhir ini adalah Algoritma *Rabin-Karp*.



Gambar 2.2. Contoh *String Matching*

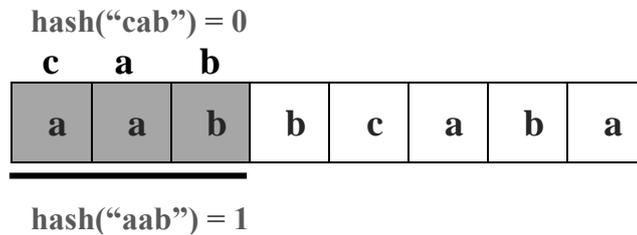
### 2.2.1. Pencocokkan *String* dengan Algoritma *Rabin-Karp*

Pada tahun 1987 Michael O. Rabin dan Richard M. Karp menciptakan sebuah algoritma pencocokkan string yang diberi nama Algoritma *Rabin-Karp*. Algoritma *Rabin-Karp* digunakan untuk pencocokkan kata dengan pola tunggal (*single pattern search*), namun lebih efektif jika digunakan untuk pencocokkan kata dengan pola banyak (*multi pattern search*) (Riyanti, 2009). Algoritma ini mempercepat pengecekan kata pada suatu kata dengan menggunakan fungsi *hash*. Fungsi *hash* adalah fungsi yang mengkonversikan suatu kata menjadi nilai yang disebut nilai *hash* (*hash value*) (Atmopawiro, 2007). Fungsi *hash* yang digunakan algoritma Rabin-Karp adalah *rolling hash*.

Pada dasarnya, algoritma *Rabin-Karp* akan membandingkan nilai hash dari string masukkan (*pattern*) dan *substring* pada teks. Apabila sama, maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila tidak sama, maka *substring* akan bergeser ke kanan. Kunci utama performa algoritma ini adalah perhitungan yang efisien terhadap nilai *hash substring* pada saat penggeseran dilakukan (Surahman, 2013). Gambar 2.3 hingga Gambar 2.7 menjelaskan cara kerja algoritma *Rabin-Karp* dengan fungsi *hash*.

Berikut penjelasan contoh cara kerja algoritma *Rabin-Karp* (Riyanti, 2009) :

1. Diberikan masukkan “cab” dan teks “aabbcaba”. Fungsi *hash* yang dipakai misalnya akan menambahkan nilai keterurutan setiap huruf dalam alfabet (a = 1, b = 2, c=3, dst.) dan melakukan modulo 3. Didapatkan nilai *hash* dari “cab” adalah 0 dan tiga karakter pertama pada teks yaitu “aab” adalah 1. Seperti pada Gambar 2.3.

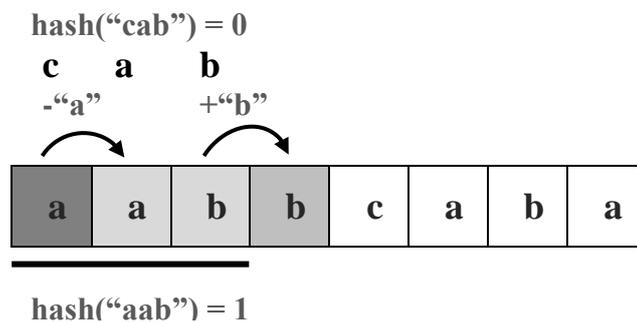


Gambar 2.3. *Fingerprint* awal

$$\text{hash(“cab”)} = 3 + 1 + 2 \text{ modulo } 3 = 6 \text{ modulo } 3 = 0$$

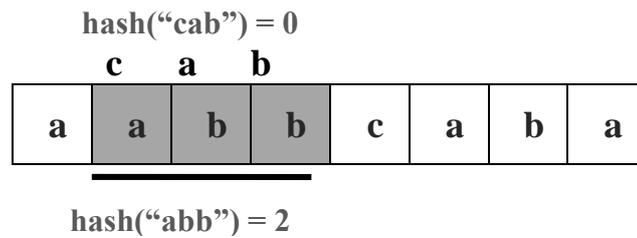
$$\text{hash(“aab”)} = 1 + 1 + 2 \text{ modulo } 3 = 4 \text{ modulo } 3 = 1$$

2. Pada Gambar 2.4, substring pada teks bergeser satu karakter ke kanan dikarenakan hasil perbandingan tidak sama. Algoritma melakukan *rolling hash* yaitu mengurangi nilai karakter yang keluar dan menambahkan nilai karakter yang masuk sehingga didapatkan waktu yang relatif konstan pada setiap kali pergeseran.



Gambar 2.4. Menggeser *fingerprint*

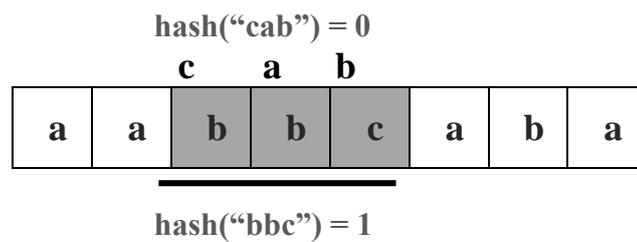
Nilai *hash* dari “abb” ( $abb = aab - a + b$ ) hasil menjadi dua ( $2 = 1 - 1 + 2$ ) setelah dilakukan pergeseran. Hasil dapat dilihat pada Gambar 2.5.



Gambar 2.5. Perbandingan kedua

$$\text{hash}(\text{"abb"}) = (\text{aab} - \text{a} + \text{b}) \text{ modulo } 3 = (1 - 1 + 2) \text{ mod } 3 = 2 \text{ mod } 3 = 2$$

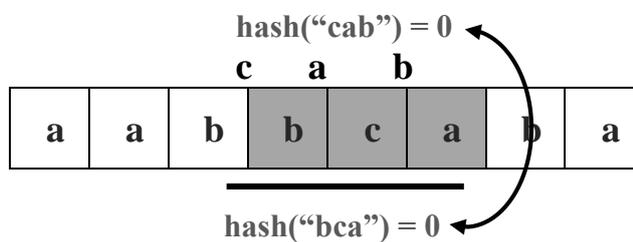
3. Pergeseran akan terus dilakukan jika nilai *hash* belum sama seperti Gambar 2.6.



Gambar 2.6. Perbandingan ketiga

$$\text{hash}(\text{"bbc"}) = (\text{abb} - \text{a} + \text{c}) \text{ modulo } 3 = (2 - 1 + 3) \text{ mod } 3 = 4 \text{ mod } 3 = 1$$

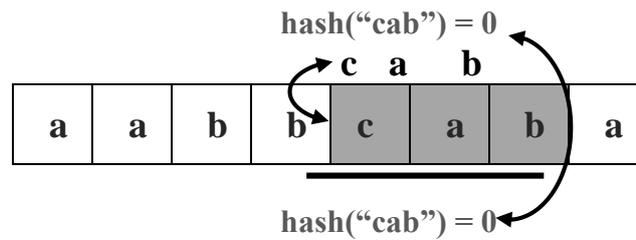
4. Pada perbandingan keempat, didapatkan nilai *hash* yang sama. Pada Gambar 2.7 dilakukan perbandingan *string* karakter per karakter antara "bca" dan "cab" karena nilai *hash*nya sama. Hasil perbandingan *string* tersebut tidak sama, sehingga *substring* kembali digeser ke kanan.



Gambar 2.7. Perbandingan keempat (nilai *hash* sama)

$$\text{hash}(\text{"bca"}) = (\text{bbc} - \text{b} + \text{a}) \text{ modulo } 3 = (1 - 2 + 1) \text{ mod } 3 = 0 \text{ mod } 3 = 0$$

5. Pada perbandingan yang kelima, ditemukan nilai *hash* yang sama. Kedua nilai *hash* dan karakter pembentuk *string* sesuai, sehingga solusi ditemukan. Gambar 2.8 adalah akhir perbandingan karena perbandingan kedua *string* tersebut sama dengan nilai *hash* yang sama.



Gambar 2.8. Perbandingan kelima (*string* ditemukan)

$$\text{hash}(\text{"cab"}) = (\text{bca} - \text{b} + \text{b}) \text{ modulo } 3 = (0 - 2 + 2) \text{ mod } 3 = 0 \text{ mod } 3 = 0$$

Algoritma *Rabin-Karp* menggunakan fakta bahwa, jika dua kata merupakan kata yang sama maka nilai *hash* dari kedua kata tersebut akan sama juga. Oleh karena itu, pemeriksaan kecocokan kata hanya memerlukan perhitungan nilai *hash* dari *substring* yang akan dicari dengan kata-kata yang memiliki nilai *hash* sama. Akan tetapi, permasalahan akan timbul karena jenis kata yang berbeda sangat banyak. Beberapa kata yang perlu diberi nilai *hash* yang sama untuk menjaga *hash* tetap kecil. Jika nilai *hash* sama maka kata tersebut belum tentu sama (Atmopawiro, 2007). Kode 2.1 merupakan *pseudocode* dari algoritma *Rabin-Karp*.

```

function RabinKarp (input s: s[1..m], teks:
string[1..n]) → boolean
{ Melakukan pencarian string s pada string teks dengan
  algoritma Rabin-Karp}
Deklarasi
  i : integer
  ketemu = boolean
Algoritma
  ketemu ← false
  hs ← hash(s[1..m])
  hsub ← hash(teks[1..i+m-1])
  for i ← 1 to n do
    if hsub = hs then
      if teks[i..i+m-1] = s then
        ketemu ← true
      else
        hsub ← hash(teks[i+1..i+m])
    endfor
  return ketemu

```

Kode 2.1. *Pseudocode* Algoritma *Rabin-Karp* (Riyanti, 2009)

Pada Kode 2.1 Algoritma pada baris 2 dan 3 adalah fungsi perhitungan *hash* hanya dieksekusi sekali dan baris 6 hanya dieksekusi bila *hash*-nya sama, yang kemungkinan akan

terjadi lebih dari satu kali. Baris 5 dieksekusi  $n$  kali, dengan waktu yang konstan. Baris 8 jika melakukan komputasi ulang nilai *hash* untuk  $(\text{teks}[i+1.. i+m])$ , akan membutuhkan waktu karena dieksekusi setiap loop. Trik untuk mengatasi ini dengan menggunakan variabel *hsub* telah mempunyai nilai *hash*  $(\text{teks}[i+1.. i+m])$ . Jika nilai *hash* berikutnya dapat dihitung konstan maka permasalahan terselesaikan. Caranya dengan menggunakan *rolling hash*.

### 2.3. Rolling Hash

*Hashing* adalah suatu cara untuk mentransformasi sebuah *string* menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda (*signature*) *string* tersebut. Panjang *string* sesuai dengan nilai *k-gram* yang ditentukan. *Hash function* atau fungsi *hash* adalah suatu cara menciptakan *fingerprint* dari berbagai data masukkan. *Hash function* akan mengganti atau mentransformasikan data tersebut untuk menciptakan *fingerprint*, yang biasa disebut *hash value* (Purwitasari et al., 2010).

Nilai *hash* yang akan dicari dengan fungsi *hash* dalam algoritma *Rabin-Karp* merupakan representasi dari nilai *ASCII* (*American Standar Code for Information Interchange*) yang menempatkan angka numerik pada karakter, angka, tanda baca dan karakter-karakter lainnya. *ASCII* menyediakan 256 kode yang dibagi ke dalam dua himpunan standar dan diperluas yang masing-masing terdiri dari 128 karakter. Himpunan ini merepresentasikan total kombinasi dari 7 atau 8 *bit*, yang kemudian menjadi angka dari *bit* dalam 1 *byte*. *ASCII* standar menggunakan 7 bit untuk tiap kode dan menghasilkan 128 kode karakter dari 0 sampai 127 (heksadesimal 00H sampai 7FH). Himpunan *ASCII* yang diperluas menggunakan 8 bit untuk tiap kode dan menghasilkan 128 kode tambahan dari 128 sampai 255 (heksadesimal 80H sampai FFH) (Purwitasari et al., 2010).

*Rolling hash* adalah sebuah fungsi *hash* yang *input*-nya dikelompokkan ke dalam suatu blok yang digerakkan melewati input secara keseluruhan. Beberapa fungsi *hash* memungkinkan *rolling hash* untuk dikomputasi dengan cepat. Nilai *hash* yang baru dapat dengan cepat dihitung dari nilai *hash* yang lama dengan cara menghilangkan nilai lama dari kelompok *hash* dan menambahkan nilai baru ke dalam kelompok tersebut. (Surahman, 2013).

Kunci dari performa algoritma *Rabin-Karp* adalah komputasi yang efektif dari nilai *hash* dari *substring-substring* yang berurutan pada teks. Algoritma *Rabin-Karp* melakukan perhitungan nilai *hash* dengan memperlakukan setiap *substring* sebagai sebuah angka dengan basis tertentu (Adam et al., 2009). Perhitungan nilai *hash* dengan *rolling hash* dapat dilihat pada Persamaan 2.1. Keuntungan dari *rolling hash* adalah untuk nilai *hash* berikutnya dapat dilakukan dengan Persamaan 2.2. Dengan begitu tidak perlu melakukan iterasi dari indeks pertama sampai terakhir. Hal ini tentu menghemat nilai *hash* dari sebuah *string* (Surahman, 2013).

$$H_{(c_1 \dots c_k)} = c_1 * b^{(k-1)} + c_2 * b^{(k-2)} + \dots + c_{(k-1)} * b^k + c_k \dots \dots \dots (2.1)$$

$$H_{(c_2 \dots c_{k+1})} = (H_{(c_1 \dots c_k)} - c_1 * b^{(k-1)}) * b + c_{(k+1)} \dots \dots \dots (2.2)$$

Keterangan :

- H : *substring*
- c : nilai ascii per-karakter
- b : basis
- k : banyak karakter

Sebagai contoh, jika *substring* yang ingin dicari adalah “machio”. Karakter (k) yang ditentukan sebanyak 5 sehingga *substring* machio terbagi menjadi 2 yaitu “machi” dan achio. Sedangkan basis (b) yang digunakan adalah 3, rumus perhitungan nilai *hash* menggunakan Persamaan 2.1.

Contoh 2.1 :

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

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

Nilai *hashing* pada *substring* “machi” bernilai 12756. Untuk *substring* selanjutnya menggunakan persamaan 2.2. Sebagai contoh, “achio” k = 5 dan b = 3.

$$H_{(achio)} = (H_{(machi)} - \text{ascii}(m) * 3^{(4)}) * 3 + \text{ascii}(o)$$

$$H_{(achio)} = (12756 - 109 * 81) * 3 + 111 = 11892$$

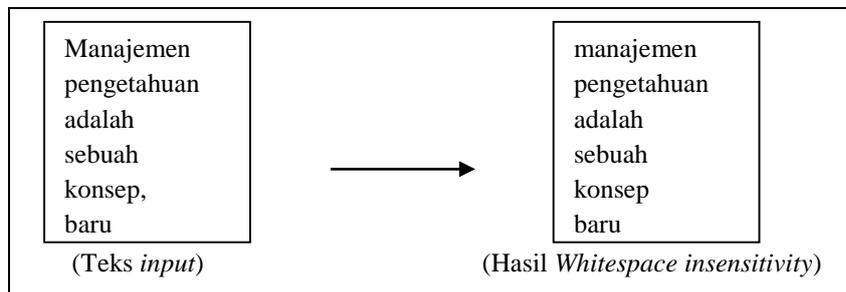
Nilai *hashing* pada *substring* “achio” bernilai 11892.

## 2.4. Tahap *Text Preprocessing*

Pada umumnya data berupa teks memiliki dimensi yang tinggi, terdapat *noise* pada data, dan struktur kalimat yang tidak baik. Untuk itu dilakukan pemecahan dokumen menjadi kalimat, kata sehingga dapat dilakukan pemrosesan data dengan lebih baik. Sebenarnya ini merupakan bagian dari *text preprocessing*. Secara umum dalam tahap – tahap preprocessing yaitu (Mooney, 2006) :

### 1. *Whitespace insensitivity*

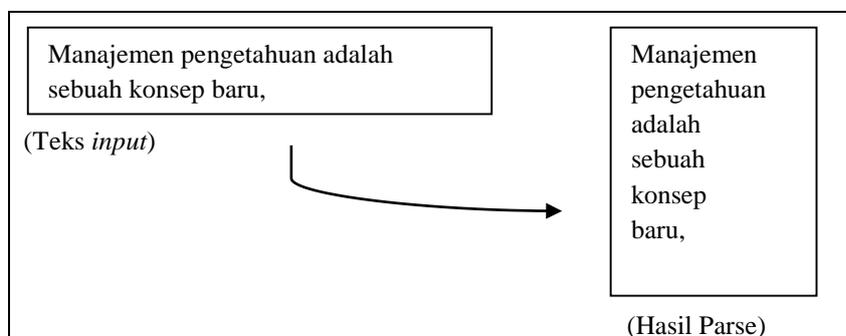
*Whitespace insensitivity* merupakan tahapan yang menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*). Contoh proses *Whitespace insensitivity* dapat dilihat pada Gambar 2.9.



Gambar 2.9. Contoh *Whitespace Insensitivity*

### 2. *Parsing*

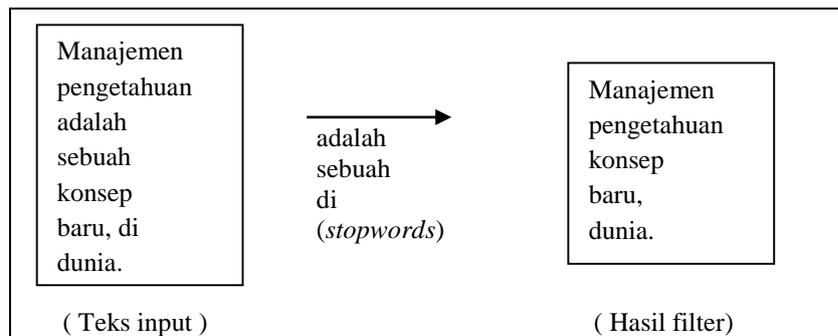
*Parsing* merupakan pemecahan kalimat menjadi kata. Contoh proses *Parsing* terdapat pada Gambar 2.10.



Gambar 2.10. Contoh *Parsing*

### 3. *Filtering*

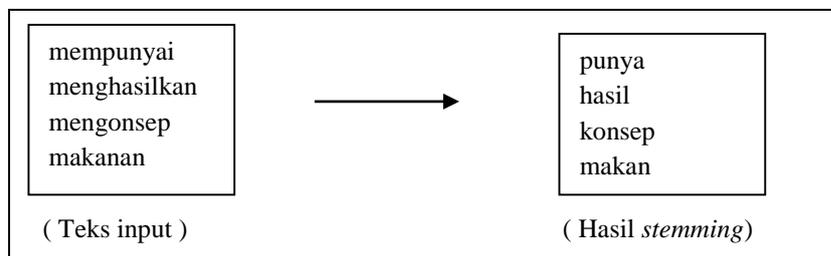
*Filtering* merupakan tahap pengambilan kata-kata penting dari hasil token. Pengambilan kata-kata ini dapat dilakukan dengan membuang kata yang kurang penting (*stopwords*) atau menyimpan kata-kata penting (*wordlist*). *Stoplist/ stopwords* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bags-of-words*. Contoh *stopwords* adalah “yang”, “dan”, “di”, “dari” dan sebagainya. Contoh proses *filtering* terdapat dalam Gambar 2.11.



Gambar 2.11. Contoh *Filtering*

### 4. *Stemming*

*Stemming* merupakan tahap mencari *root* kata dari tiap kata hasil *filtering*. Pada tahap ini dilakukan proses pengembalian bentukan kata ke dalam representasi yang sama. Pada teks berbahasa Indonesia, *stemming* sulit diterapkan karena bahasa Indonesia tidak memiliki rumus bentuk baku yang permanen. Contoh proses *stemming* terdapat dalam Gambar 2.12.



Gambar 2.12. Contoh *Stemming*

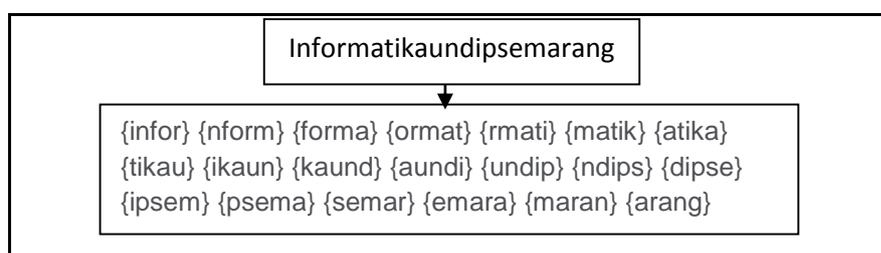
Dalam tugas akhir ini, tahap-tahap preprocessing tersebut tidak semua digunakan. Tahap-tahap yang digunakan adalah *filtering*, *whitespace insensitivity* dan *parsing*. *Stemming* tidak digunakan karena pada penelitian Salmuasih dapat diketahui bahwa tanpa menggunakan *stemming* akan mempercepat waktu proses, tetapi akurasi yang nilai kemiripannya rendah (Salmuasih et al., 2013).

## 2.5. Parsing K-Gram

*Parsing* yang digunakan merupakan rangkaian *terms* dengan panjang yang ditentukan K. Kebanyakan yang digunakan sebagai *terms* adalah kata. *Parsing K-gram* merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau karakter. Metode *k-grams* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah k dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen (Salmuasih et al., 2013). Pada tahap ini digunakan sebagai masukkan *string* untuk tahap *Hashing*.

Dalam *Markov Model* nilai *K-Gram* yang sering digunakan yaitu, *2-gram* (*bigram*), *3-gram* (*trigram*), *4-gram*, *5-gram* dan seterusnya). Dalam *natural language processing*, penggunaan *K-Gram*, proses *parsing token* (*tokenisasi*) lebih sering menggunakan *3-gram* dan *4-gram*, sedangkan *2-gram* digunakan dalam *parsing sentence*, misal dalam *part-of-speech (POS)*. Penggunaan *2-gram* dalam *tokenisasi* akan menyebabkan tingkat perbandingan antar karakter akan semakin besar. Contohnya pada kata „makan” dan „mana” yang merupakan dua kata yang sama sekali berbeda. Dengan menggunakan metode *bigram* dalam mencari kemiripan, hasil dari *bigram* tersebut yaitu kata “makan” akan menghasilkan *bigram* *ma, ak, ka, an* serta kata “mana” akan menghasilkan *bigram* *ma, an, na*. Dengan demikian, akan terdapat banyak kesamaan kata dalam pemrosesan nilai kemiripan. Namun jika menggunakan *3-gram* (“makan” = *mak, aka, kan* dan “mana” = *man, ana*) atau *4-gram* (“makan” = *maka, akan, dan* “mana” = *mana*) akan mengecilkan kemungkinan terjadinya kesamaan pada kata yang strukturnya berbeda (Surahman, 2013).

Contoh pada Gambar 2.13 menggunakan nilai *5-gram* yang mana semakin mengecilkan kemungkinan terjadinya kesamaan kata karena semakin besar k-gram semakin kecil adanya kesamaan kata. Hasil teks dari Gambar 2.13 tersebut menghasilkan 20 *gram*.



Gambar 2.13. *Parsing 5-gram*

## 2.6. Persentase Kemiripan

Dalam penelitian Mutiara, *range* persentase nilai kemiripan yang digunakan adalah (Mutiara, 2011) :

1. 0% : Hasil uji 0% berarti kedua dokumen tersebut benar-benar berbeda baik dari segi isi dan kata secara keseluruhan.
2. < 15% : Hasil uji 15% berarti kedua dokumen tersebut hanya mempunyai sedikit kesamaan.
3. 15 - 50% : Hasil uji 15-50% berarti menandakan dokumen tersebut termasuk kemiripan tingkat sedang.
4. > 50% : Hasil uji lebih dari 50% berarti dapat dikatakan bahwa dokumen tersebut mendekati kemiripan.
5. 100% : Hasil uji 100% menandakan bahwa dokumen tersebut adalah memiliki kemiripan yang sama karena dari awal sampai akhir mempunyai isi yang sama persis.

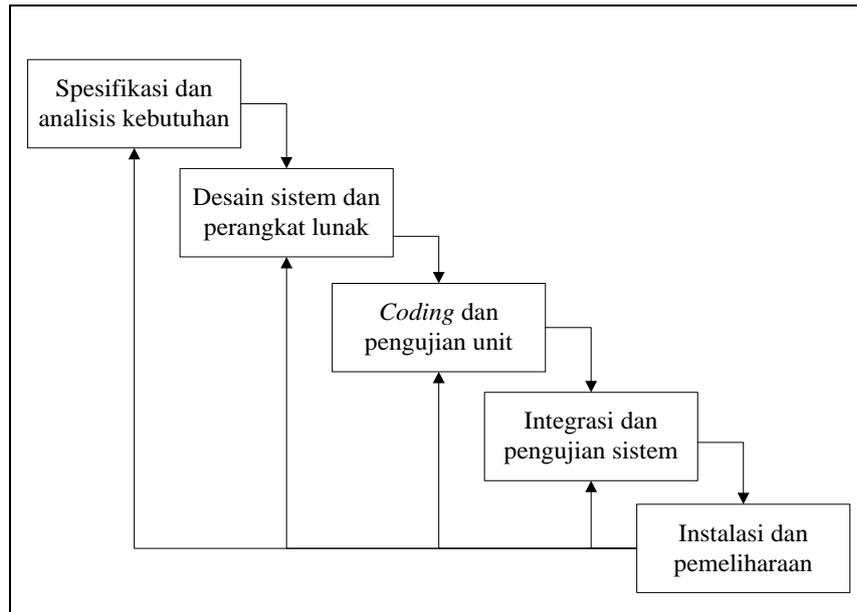
## 2.7. Tahap Pengembangan Sistem

Dalam pengembangan suatu perangkat lunak, digunakan beberapa model, salah satunya adalah model *Waterfall* atau *Classic Life Cycle*. Model ini merupakan model klasik yang bersifat sistematis, berurutan dalam membangun perangkat lunak. Ada lima tahap dalam model *waterfall*, yaitu spesifikasi dan analisis kebutuhan, desain sistem dan perangkat lunak, *coding* dan pengujian unit, integrasi dan pengujian sistem, serta instalasi dan pemeliharaan.

Model *waterfall* mempunyai keunggulan dalam membangun dalam mengembangkan sistem, antara lain (Sommerville, 2007) :

1. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.
2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya. Jadi setiap fase atau tahapan akan mempunyai dokumen tertentu.

Sesuai dengan namanya *waterfall* (air terjun) maka tahapan dalam model ini disusun bertingkat dan setiap tahap dilakukan berurutan. Model *waterfall* dapat dilihat pada Gambar 2.14. (Sommerville, 2007).



Gambar 2.14. Model *Waterfall* (Sommerville, 2007)

Tahapan model *waterfall* dapat dijelaskan sebagai berikut :

1. Spesifikasi dan Analisis kebutuhan

Mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Meningkatkan usaha proses pengumpulan persyaratan dan berfokus pada perangkat lunak. Untuk mengetahui program yang akan dibuat, maka harus diketahui cakupan informasi perangkat lunak, seperti fungsi, tingkah laku, kinerja dan antarmuka.

Setelah menganalisis data dan mengelompokkannya berdasarkan jenis datanya maka tahap selanjutnya adalah melakukan analisis kebutuhan sistem. Analisis kebutuhan sistem meliputi :

- a. *Entity Relationship Diagram* (ERD)
- b. *Data Context Diagram* (DCD)
- c. *Data Flow Diagram* (DFD)

2. Desain sistem dan perangkat lunak

Desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya. Desain menspesifikasikan abstraksi sistem dan perangkat lunak serta mendefinisikan arsitektur sistem secara keseluruhan. Proses desain menterjemahkan persyaratan ke dalam representasi perangkat lunak yang

bisa diperkirakan demi kualitasnya sebelum memulai pembuatan kode. Tahapan perancangan sistemnya adalah :

a. *Flowchart*

b. Perancangan Antarmuka

3. Coding dan pengujian unit

Peneliti melakukan implementasi dengan menggunakan PHP MySQL. Pada tahap ini, semua algoritma dan proses pada perancangan sistem akan diimplementasikan dalam sebuah aplikasi sebagai wujud dari sistem. Sistem ini akan dikembangkan dalam bahasa pemrograman PHP.

4. Integrasi dan pengujian sistem

Tahap pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya. Pengujian yang baik adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya. Ada dua macam metode pengujian, yaitu *black-box* dan *white-box*. Metode *black-box* yaitu menguji fungsionalitas dari perangkat lunak saja tanpa harus mengetahui struktur internal program (*source code*), sedangkan metode *white-box* yaitu menguji fungsionalitas dari perangkat lunak dengan menguji struktur internal program (*source code*). Tahap pengujian ini, Penulis melakukan pengujian sistem dengan metode *black-box*.

5. Instalasi dan pemeliharaan

Umumnya tahap ini merupakan tahap yang paling lama. Saat mengoperasikan program di lingkungannya dan melakukan pemeliharaan, akan melakukan penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru. Namun, pada tahap ini penulis tidak menjelaskan mengenai instalasi dan pemeliharaan sistem tersebut.

Dalam pelaksanaan metode *waterfall* tersebut, dapat terjadi tumpang tindih antar tahap dan pertukaran informasi antara tahap yang satu dengan tahap yang lainnya. Misalnya pada saat desain ditemukan masalah di tahap spesifikasi kebutuhan, pada saat

*coding* ditemukan masalah di tahap desain, dan sebagainya. Proses pengembangan perangkat lunak ini tidak sepenuhnya *linear*, tetapi memperbolehkan serangkaian iterasi dalam pelaksanaannya (Sommerville, 2007).

## 2.8. Data Flow Diagram (DFD)

*Data Flow Diagram* (DFD) adalah sebuah teknik grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari *input* menjadi *output*. DFD juga dikenali sebagai grafik aliran data atau *bubblechart* (Pressman, 2001). Simbol-simbol *DFD* disajikan pada Tabel 2.1.

Tabel 2.1. Simbol-simbol pada DFD (Pressman, 2001)

Notasi	Nama	Keterangan
	Entitas eksternal	Sebuah elemen sistem atau sistem yang lain yang menghasilkan informasi bagi transformasi oleh perangkat lunak atau menerima informasi yang dihasilkan oleh perangkat lunak.
	Proses	Menunjukkan transformasi yang diaplikasikan ke data (kontrol) dan mengubahnya dengan berbagai macam cara.
	<i>Data store</i>	Tempat penyimpanan data dalam sistem.
	<i>Data flow</i>	Menggambarkan aliran data antara entitas eksternal dan proses, proses dan proses, serta proses dan penyimpanan data.

## 2.9. Entity Relation Diagram (ERD)

Pemodelan data berfungsi untuk menjelaskan objek data utama yang akan diproses oleh sistem, bagaimana komposisi dari masing-masing objek data termasuk atributnya, hubungan antara masing-masing objek data dan objek yang lainnya dan bagaimana hubungan antara objek dengan proses yang mentransformasikannya (Pressman, 2001).

Untuk menjawab berbagai hal tersebut, metode pemodelan data menggunakan diagram-ER atau *Entity Relationship Diagram*. ERD hanya berfokus pada data dan melihat data secara independen dari pemrosesan yang mentransformasikan data tersebut. ERD terdiri dari sekumpulan objek-objek, yang disebut dengan entitas dan hubungan yang terjadi diantara objek-objek tersebut. ERD terdiri dari tiga informasi yang saling tergantung, yaitu objek data atau entitas, atribut yang menggambarkan objek data tersebut atau atribut, dan hubungan objek data satu dan lain atau relasi.

1. Objek data atau entitas

Suatu entitas merupakan suatu objek dasar atau individu yang mewakili sesuatu yang nyata eksistensinya dan dapat dibedakan dari objek-objek yang lain. Suatu entitas mempunyai sekumpulan sifat, dan nilai dari beberapa sifat tersebut adalah unik, sehingga dapat mengidentifikasi entitas tersebut. Sekumpulan entitas yang mempunyai tipe sama (sejenis) dan berada dalam ruang lingkup yang sama membentuk suatu himpunan entitas.

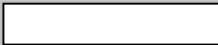
2. Atribut

Atribut merupakan sifat-sifat atau *property* yang dimiliki oleh entitas. Berdasarkan sifat keunikannya, atribut dibagi menjadi dua, yaitu atribut *key* (*identifier*) dan atribut *non-key* (*descriptor*). Atribut *key* digunakan untuk menentukan suatu entitas secara unik (*primary key*), sedangkan, atribut *non-key* digunakan untuk menspesifikasikan karakteristik dari suatu entitas yang tidak unik.

3. Relasi dan himpunan relasi

Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari sejumlah himpunan entitas yang berbeda. Kumpulan semua relasi di antara entitas-entitas yang terdapat pada himpunan entitas membentuk suatu himpunan relasi. Notasi-notasi yang digunakan dalam ERD dapat dilihat pada Tabel 2.2 berikut :

Tabel 2.2. Tabel Notasi *Entity Relation Diagram* (ERD)

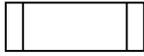
Notasi	Keterangan
	Entity
	Atribut
	Hubungan
	Garis

**2.10. Flowchart**

*Flowchart* adalah penyajian yang sistematis tentang proses dan logika dari kegiatan penanganan informasi atau penggambaran secara grafik dari langkah-langkah dan urutan prosedur dari suatu program (Silberschatz et al., 2001). *Flowchart* menolong analis dan *programmer* untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain

dalam pengoperasian (Silberschatz et al., 2001). Simbol-simbol *flowchart* disajikan pada Tabel 2.3.

Tabel 2.3. Simbol-simbol pada *Flowchart* (Silberschatz et al., 2001)

Simbol	Nama	Fungsi
	<i>Terminator</i>	Permulaan/ akhir program
	Garis Alir ( <i>Flowline</i> )	Arah aliran program
	Nilai Awal ( <i>Preparation</i> )	Proses inisialisasi/ Pemberian harga awal
	Proses	Proses perhitungan/ Proses pengolahan data
	<i>Input/ Output Data</i>	Proses <i>input/ output</i> data, parameter dan informasi
	<i>Predefined Process</i> (sub program)	Permulaan sub program/ proses menjalankan sub program
	<i>Decision</i>	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya.
	<i>On Page Connector</i>	Penghubung bagian-bagian <i>flowchart</i> yang berada pada satu halaman
	<i>Database</i>	Informasi penyimpanan di dalam data store
	<i>Internal Storage</i>	Penyimpanan storage untuk akses langsung.

## 2.11. MySQL

MySQL adalah sebuah *Database Open Source* populer di dunia. Penggunaannya sebagai database bahasa pemrograman seperti PHP dan Java. Untuk memudahkan penggunaan MySQL, terdapat *software open source* berbasis GUI yakni PHP MyAdmin. PHP MyAdmin ini juga terdapat secara default pada Xampp yaitu *software* yang membundel apache, PHP, MySQL serta Perl, ditambah modul-modul tambahan. MySQL sebenarnya produk yang berjalan pada *platform* Linux. Karena sifatnya open source, MySQL dapat dijalankan pada semua *platform* baik Windows maupun Linux. Selain itu, MySQL juga merupakan program pengakses *database* yang bersifat jaringan sehingga dapat digunakan untuk aplikasi *multi user*. Kelebihan lain dari MySQL adalah MySQL menggunakan bahasa *query* standar yang dimiliki *Structure Query Language (SQL)*. SQL adalah suatu bahasa permintaan terstruktur yang telah

distandarkan untuk semua program pengakses *database* seperti Oracle, Posgres SQL, SQL Server, dan lain-lain (Anonim, 2008).

## 2.12. PHP

*Hypertext Preprocessor* (PHP) merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting*. PHP memungkinkan anda untuk membuat halaman *web* yang bersifat dinamis. PHP dapat dijalankan pada berbagai macam sistem operasi misalkan : Windows, LINUX, dan Mac OS. Selain Apache, PHP juga mendukung beberapa *server web* lain, misalkan Microsoft IIS, Caudium, PWS dan lain-lain. Sistem manajemen *database* yang sering digunakan bersama PHP adalah MySQL. Namun, PHP juga mendukung sistem manajemen *database Oracle, Microsoft Access, Interbase, dBase, PostgreSQL*, dan lain-lain. Hingga kini, PHP sudah berkembang hingga versi 5. PHP bersifat *open source* sehingga setiap orang dapat menggunakannya secara gratis (Anonim, 2008).

## 2.13. Pengukuran dan Kesalahan

Pengukuran adalah proses perbandingan antara suatu besaran yang tidak diketahui dengan suatu besaran standar yang diperoleh, yang meliputi hubungan suatu alat ukur di dalam *system* dengan pertimbangan dan pengamatan dari hasil respon pada instrument. Pengukuran yang diperoleh adalah pengukuran besaran yang disebut dengan harga sebenarnya (*true value*) akan tetapi sangat sukar untuk memberi definisi harga yang sebenarnya (Tsuneo, 2011).

Tingkatan dimana suatu pengukuran sesuai dengan harga yang diharapkan ditunjukkan dalam syarat-syarat kesalahan dari pengukuran. Kesalahan mungkin ditunjukkan lain yaitu dengan kesalahan absolut atau prosentase kesalahan. Kesalahan absolut dapat didefinisikan sebagai perbedaan antara variable nilai yang diharapkan dengan variable nilai pengukuran. Jika kita mengharapakan untuk menunjukkan kesalahan sebagai prosentase, maka dapat kita rumuskan sebagai hasil bagi antara kesalahan absolute dengan harga yang diharapkan dikali dengan 100 % (Tsuneo, 2011). Persentase Kesalahan dapat dilihat pada persamaan 2.3. Persentase tersebut dibutuhkan untuk menyatakan pengukuran untuk tingkat keakuratan, seperti pada Persamaan 2.4

Rumus Persentase Kesalahan seperti dalam persamaan 2.3.

$$\% \text{ Kesalahan} = \left| \frac{Y_n - X_n}{Y_n} \right| \times 100\% \dots\dots\dots(2.3)$$

Keterangan

$Y_n$  : Nilai yang diharapkan

$X_n$  : Nilai pengukuran

Rumus Tingkat Keakuratan seperti dalam persamaan 2.4.

$$a = 100\% - \% \text{ Kesalahan} \dots\dots\dots (2.4)$$

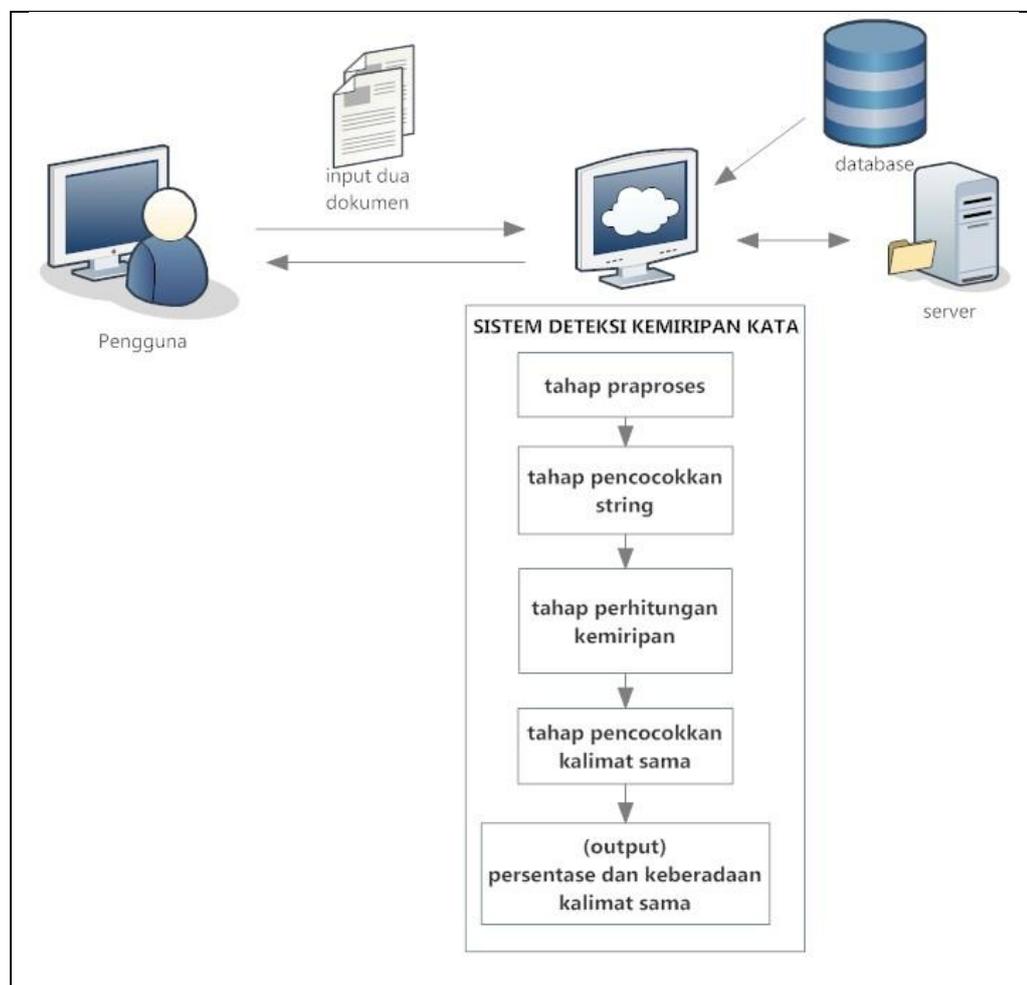
## BAB III

### SPESIFIKASI, ANALISIS, DAN PERANCANGAN

Bab ini menyajikan tahapan proses pembangunan perangkat lunak, yaitu gambaran umum sistem, spesifikasi analisis kebutuhan, dan perancangan. Ketiga tahapan tersebut merupakan langkah-langkah dari pengembangan perangkat lunak menggunakan model proses *waterfall*.

#### 3.1. Gambaran Umum Sistem

Gambaran umum sistem yang dikembangkan mempunyai 4 komponen yaitu pengguna, sistem, *server* dan *database*. Secara garis besar gambaran umum sistem deteksi kemiripan kata pada dua dokumen menggunakan Algoritma *Rabin-Karp* yang dikembangkan digambarkan seperti pada Gambar 3.1.



Gambar 3.1 Gambaran Umum Sistem Deteksi Kemiripan Kata

Tahapan proses sistem deteksi kemiripan kata yaitu sebagai berikut :

1. *Input* dua dokumen

Memasukkan dua dokumen, yaitu dokumen asli dan dokumen uji. Dokumen asli sebagai pola (*pattern*) atau dokumen dasar yang digunakan sebagai acuan kemiripan dokumen uji. Dokumen uji sebagai teks atau dokumen yang akan diuji kemiripannya berdasarkan dokumen asli. Dokumen yang dimasukkan berekstensi (\*.docx). Pembacaan isi dokumen yang diproses terdiri dari huruf dan angka.

2. Tahap Praproses (*Text Preprocessing*)

Setelah dokumen asli dan dokumen uji dimasukkan, maka akan dilakukan tahap praproses (*text preprocessing*). *Text preprocessing* merupakan tahap awal memproses isi dokumen. Tahap ini terdiri dari proses *filtering*, *whitespace insensitivity*, dan *parsing k gram* :

a. *Filtering*

Proses ini menghilangkan kata-kata yang tidak penting (*stopwords*) pada isi dokumen. Daftar *stopwords* terlampir pada Lampiran 1.

b. *Whitespace insensitivity*

Proses ini menghapus tanda baca, *spasi* dan mengubah menjadi huruf kecil (*lowercase*) pada isi dokumen. Daftar tanda baca ada pada Lampiran 2.

c. *Parsing k-gram*

Proses ini memecah isi dokumen yang sudah melalui proses *filtering* dan *whitespace insensitivity*. Nilai k yang ditentukan *5-gram*, penjelasan dapat dilihat pada Gambar 2.13. Dokumen hasil *Whitespace insensitivity* dipecah menjadi 5 ukuran kata atau *parse*.

3. Tahap Pencocokkan *String*

Setelah tahap praproses selesai, kemudian dilanjutkan dengan tahap pencocokkan *string* yaitu *rolling hash* dan pencocokkan *string* dengan algoritma *Rabin-Karp*. Adapun detail mengenai proses tersebut adalah :

a. *Rolling Hash*

Proses ini mengubah hasil praproses (kata atau *parse*) menjadi *hash value*. Nilai *hash value* tersebut sebagai masukkan algoritma *Rabin-Karp*. Persamaan

*Rolling Hash* dapat dilihat di Persamaan 2.1 dan 2.2.

b. Pencocokan *String* dengan Algoritma *Rabin-Karp*

Hasil dari *rolling hash* menghasilkan *hash value* kedua dokumen tersebut. Kemudian, nilai *hash* dua dokumen akan dicocokkan kesamaannya. Proses algoritma ini dapat dilihat pada *pseudocode* Kode 2.1.

4. Tahap Perhitungan Kemiripan

Tahapan ini menghitung hasil kemiripan kata antara dokumen uji dengan dokumen asli. Nilai kemiripan dihitung berdasarkan pada banyaknya jumlah *hash* yang telah diproses dengan Algoritma *Rabin-Karp* dan jumlah *hash* pada dokumen uji dapat dilihat pada Persamaan 3.1. Hasil perhitungan kemiripan ini berasal dari hasil dokumen uji yang menghasilkan nilai persentase sebagai batas yang menentukan tingkat plagiarisme.

$$s = \frac{n_f}{n_{(uji)}} \dots\dots\dots(3.1)$$

Keterangan :

$s$  : Hasil perhitungan kemiripan kata

$n_{(uji)}$  : Jumlah *hash* Dokumen Uji

$n_f$  : Jumlah *hash* yang sama antara dokumen uji dan dokumen asli.

5. Tahap Pencocokkan Kalimat Sama

Pada tahapan ini dilakukan pengambilan data dari dokumen asli dan dokumen uji. Dengan memotong paragraf menjadi kalimat-kalimat. Kemudian, kalimat ini akan dicocokkan dengan kalimat pada dokumen lain. Tahapan ini, akan menampilkan kalimat yang sama pada setiap dokumen.

6. *Output* Persentase dan keberadaan kalimat sama

Hasil *output* sistem kemiripan kata dihasilkan dari tahap perhitungan kemiripan dan tahap pencocokkan kalimat sama. Tahap perhitungan kemiripan menghasilkan nilai persentase dokumen uji, sedangkan tahap pencocokkan kalimat sama menghasilkan keberadaan kalimat sama setiap dokumen. Setiap kalimat sama ditandai dengan warna berbeda.

Berikut ini adalah contoh penerapan tahapan proses Sistem Deteksi Kemiripan Kata pada Gambar 3.1 adalah :

Contoh 3.1.

1. *Input* dua dokumen

Terdapat dua dokumen yang dimasukkan, yaitu sebagai dokumen asli dan dokumen uji. Isi kedua dokumen tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1. Contoh Dokumen

Dokumen	Isi Dokumen
Dokumen asli	Algoritma yang digunakan adalah Rabin-Karp. Hasil persentase dan keberadaan kalimat.
Dokumen uji	Hasil persentase dan keberadaan kalimat.

2. Tahap Praproses (*Text Preprocessing*)

a. *Filtering*

Pada Tabel 3.1. kata-kata yang ada pada *stopwords* dihilangkan. *Stopwords* yang ada pada dokumen asli kata “yang”, “digunakan”, “adalah”, sedangkan pada dokumen uji kata “dan”. Hasil *filtering* dapat dilihat di Tabel 3.2

Tabel 3.2. Dokumen hasil *filtering*

Dokumen	Isi Dokumen
Dokumen asli	Algoritma yang digunakan adalah Rabin-Karp. Hasil persentase dan keberadaan kalimat.
Dokumen uji	Hasil persentase dan keberadaan kalimat.

b. *Whitespace Insensitivity*

Proses selanjutnya, hasil *filtering* dilakukan pembuangan tanda baca (“.”) dan (“,”), spasi dan mengubah isi dokumen menjadi huruf kecil (*lowercase*). Hasil *Whitespace Insensitivity* dapat dilihat pada Tabel 3.3.

Tabel 3.3 Dokumen hasil *whitespace insensitivity*

Dokumen	Isi Dokumen
Dokumen asli	algoritmarabinkarhasilpersentasekeberadaankalimat
Dokumen uji	hasilpersentasekeberadaankalimat

c. *Parsing 5-gram*

Kemudian hasil *whitespace insensitivity* dipecah menjadi *5-gram*/ ukuran kata. Hasil *parsing* dokumen asli terdapat pada Tabel 3.4, sedangkan hasil

*parsing* dokumen uji terdapat pada Tabel 3.5. Hasil *parsing* kedua dokumen digunakan sebagai masukan untuk langkah selanjutnya.

Tabel 3.4 *Term* hasil *Parsing 5-gram* Dokumen Asli

No	Term	No	Term	No	Term
1	algor	17	rphas	33	ekebe
2	lgori	18	phasi	34	keber
3	gorit	19	hasil	35	ebara
4	oritm	20	asilp	36	berad
5	ritma	21	silpe	37	erada
6	itmar	22	ilper	38	radaa
7	tmara	23	lpers	39	adaan
8	marab	24	perse	40	daank
9	arabi	25	ersen	41	aanka
10	rabin	26	rsent	42	ankal
11	abink	27	senta	43	nkali
12	binka	28	entas	44	kalim
13	inkar	29	ntase	45	alima
14	nkarp	30	tasek	46	limat
15	kaph	31	aseke		
16	arpha	32	sekeb		

Tabel 3.5 *Term* hasil *Parsing 5-gram* Dokumen Uji

No	Term	No	Term	No	Term	No	Term
1	hasil	10	paper	19	aseke	27	adaan
2	asilb	11	apers	20	sekeb	28	daank
3	silbe	12	perse	21	ekebe	29	aanka
4	ilber	13	ersen	22	keber	30	ankal
5	lberu	14	rsent	23	ebara	31	nkali
6	berup	15	senta	24	berad	32	kalim
7	erupa	16	entas	25	erada	33	alima
8	rupap	17	ntase	26	radaa	34	limat
9	upape	18	tasek	27	adaan		

### 3. Tahap Pencocokan String

#### a. *Rolling Hash*

Sebagai contoh, misal kata/ parse yang diambil dari term dokumen asli yaitu “algor” dan “lgori”. Hasil desimal nilai ASCII dapat dilihat pada Tabel 3.6. Nilai basis yang ditentukan adalah 10.

Tabel 3.6. Nilai ASCII untuk kata “algor” dan “lgori”

Char	Dec	Char	Dec
a	97	l	108
l	108	g	103
g	103	o	111

Char	Dec	Char	Dec
o	111	r	114
r	114	i	105

Perhitungan *Hashing* dibawah ini menggunakan persamaan 2.1.

$$\begin{aligned}
 H_{(algor)} &= \text{ascii}(a) * 10^{(4)} + \text{ascii}(l) * 10^{(3)} + \text{ascii}(g) * 10^{(2)} \\
 &\quad + \text{ascii}(o) * 10^{(1)} + \text{ascii}(r) * 10^{(0)} \\
 H_{(algor)} &= 97 * 10000 + 108 * 1000 + 103 * 100 + 111 * 10 + 114 * 1 \\
 &= 1089524
 \end{aligned}$$

Nilai hashing pada substring “algor” bernilai 1089524. Untuk substring selanjutnya menggunakan persamaan 2.2.

$$\begin{aligned}
 H_{(lgori)} &= (H_{(algor)} - \text{ascii}(a) * 10^{(4)}) * 10 + \text{ascii}(i) \\
 H_{(lgori)} &= (1089524 - 97 * 10000) * 10 + 105 = 1195345
 \end{aligned}$$

Nilai *rolling hashing* pada substring “lgori” bernilai 1195345. Nilai *rolling hash* dokumen asli dapat dilihat pada Tabel 3.7, sedangkan untuk dokumen uji dapat dilihat pada Tabel 3.8.

b. Pencocokkan *String* dengan Algoritma *Rabin-Karp*

Proses ini mencocokkan *hash* dokumen asli dan hash dokumen uji yang sama. Pada Tabel 3.7 dan 3.8 menampilkan *hash* yang sama antara dua dokumen dengan memberikan warna biru.

Tabel 3.7 *Term* hasil *Rolling Hash* Dokumen Asli

No	Term	Hash	No	Term	Hash	No	Term	Hash
1	algor	1089524	17	rphas	1263485	33	ekebe	1128181
2	lgori	1195345	18	phasi	1234955	34	keber	1181924
3	gorit	1153566	19	hasil	1149658	35	ebara	1119337
4	oritm	1235769	20	asilp	1096692	36	berad	1093470
5	ritma	1257787	21	silpe	1267021	37	erada	1134797
6	itmar	1177984	22	ilper	1170324	38	radaa	1248067
7	tmara	1279937	23	lpers	1203355	39	adaan	1080780
8	marab	1199468	24	perse	1233651	40	daank	1107907
9	arabi	1094785	25	ersen	1136620	41	aanka	1079167
10	rabin	1247960	26	rsent	1266316	42	ankal	1091778
11	abink	1079707	27	senta	1263257	43	nkali	1217885
12	binka	1097167	28	entas	1132685	44	kalim	1178959
13	inkar	1171784	29	ntase	1226951	45	alima	1089687
14	nkarp	1217952	30	tasek	1269617	46	limat	1196986

No	Term	Hash	No	Term	Hash	No	Term	Hash
15	karpb	1179624	31	aseke	1096271			
16	arpha	1096337	32	sekeb	1262808			

Keterangan :

warna biru : *hash* yang sama

warna hitam : *hash* yang berbeda

Tabel 3.8. *Term* hasil *Rolling Hash* Dokumen Uji

No	Term	Hash	No	Term	Hash	No	Term	Hash
1	hasil	1149658	13	ersen	1136620	25	erada	1134797
2	asilb	1096678	14	rsent	1266316	26	radaa	1248067
3	silbe	1266881	15	senta	1263257	27	adaan	1080780
4	ilber	1168924	16	entas	1132685	28	daank	1107907
5	lberu	1189357	17	ntase	1226951	29	aanka	1099167
6	berup	1093682	18	tasek	1269617	30	ankal	1091778
7	erupa	1136917	19	aseke	1096271	31	nkali	1217885
8	rupap	1269282	20	sekeb	1262808	32	kalim	1178959
9	upape	1292921	21	ekebe	1128181	33	alima	1089687
10	paper	1229324	22	keber	1181924	34	limat	1196986
11	apers	1093355	23	ebara	1119337			
12	perse	1233651	24	berad	1093470			

Keterangan :

warna biru : *hash* yang sama

warna hitam : *hash* yang berbeda

Dari tabel tersebut dapat dilihat bahwa jumlah *hash* yang sama sebanyak 24.

#### 4. Menghitung Kemiripan

Jumlah *hash* Dokumen Uji ( $n_{(uji)}$ ) = 34

Jumlah *hash* yang sama ( $n_f$ ) = 24

$$s = \frac{n_f}{n_{(uji)}} \times 100\% = \frac{24}{34} \times 100\% = 70,58\%$$

Jadi, hasil perhitungan kemiripan dokumen uji dengan dokumen asli adalah 70,58%.

#### 5. Proses Pencocokkan Kalimat Sama

Proses ini mengambil data dari dokumen asli dan dokumen uji. Dengan memotong paragraf menjadi kalimat-kalimat. Setelah itu, mencocokkan kalimat-kalimat

tersebut. Hasil proses ini menampilkan kalimat yang sama dalam pencocokkan dua dokumen. Hasil pencocokkan kalimat sama dapat dilihat pada Tabel 3.9.

Tabel 3.9 Hasil Pencocokkan Kalimat Sama

Dokumen	Isi Dokumen
Dokumen asli	Algoritma yang digunakan adalah Rabin-Karp. Hasil persentase dan keberadaan kalimat.
Dokumen uji	Hasil persentase dan keberadaan kalimat.

### 3.2. Spesifikasi dan Analisis Kebutuhan Sistem

Dalam proses analisis kebutuhan ini akan digambarkan kebutuhan sistem dalam bentuk spesifikasi sistem menggunakan SRS (*Software Requirement Specification*), pemodelan data menggunakan ERD (*Entity Relationship Diagram*) dan pemodelan fungsional menggunakan DFD (*Data Flow Diagram*).

#### 3.2.1. Spesifikasi Sistem

Dalam membangun sistem deteksi kemiripan kata diperlukan spesifikasi kebutuhan yang jelas sebagai tujuan utamanya agar tidak keluar dari rencana yang telah ditetapkan. Beberapa kemampuan sistem yang didefinisikan diantaranya adalah :

Tabel 3.10. Spesifikasi Kebutuhan Fungsional

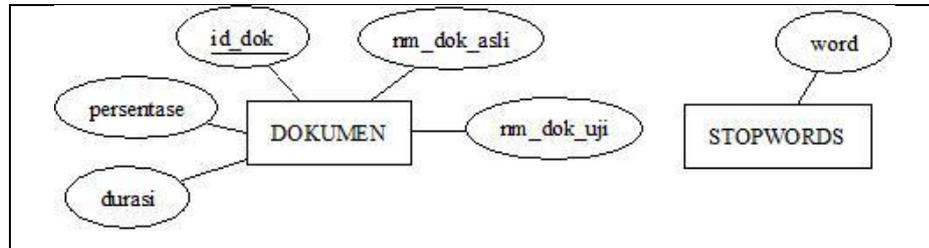
SRS ID	Deskripsi
SRS F-01	Memasukkan dua file dokumen dengan ekstensi *.docx sebagai dokumen query
SRS F-02	Melakukan pemrosesan teks terhadap dokumen query yaitu dengan <i>filtering</i> , <i>white insensitivity</i> , dan <i>parsing</i>
SRS F-03	Melakukan proses <i>rolling hash</i> terhadap dokumen <i>query</i> yang telah melakukan pemrosesan teks
SRS F-04	Melakukan proses pencocokkan dua dokumen <i>query</i> yang telah melakukan proses <i>rolling hash</i> , dengan menggunakan algoritma <i>Rabin-Karp</i>
SRS F-05	Melakukan proses perhitungan kemiripan dari hasil algoritma <i>Rabin-Karp</i> .
SRS F-06	Melakukan proses <i>highlight</i> pada kalimat yang sama antar dua dokumen <i>query</i> .
SRS F-07	Menampilkan <i>runtime</i> atau lama eksekusi sistem

#### 3.2.2. Permodelan Data

Data yang digunakan dalam Sistem Deteksi Kemiripan Kata ini diambil dari proses suatu sistem. Proses tersebut seperti *upload* dokumen, *filtering*, dan perhitungan kemiripan. Permodelan data pada Sistem Deteksi Kemiripan Kata dapat digambarkan dengan *Entity Relationship Diagram* (ERD).

ERD menunjukkan hubungan antar himpunan entitas dari sistem ini. ERD yang dapat dilihat pada Gambar 3.2 menunjukkan ada dua himpunan entitas yaitu

DOKUMEN dan STOPWORDS. Himpunan entitas DOKUMEN dan entitas STOPWORDS tidak memiliki kardinalitas yang berarti tiap entitas tidak memiliki hubungan.



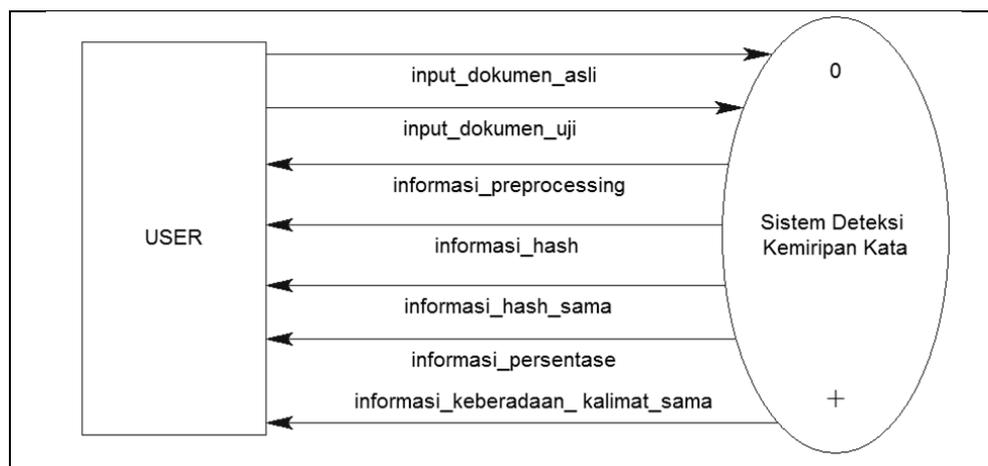
Gambar 3.2 ERD Sistem Deteksi Kemiripan Kata

### 3.2.2. Permodelan Fungsional

Permodelan fungsional dapat digambarkan dengan *Data Flow Diagram* (DFD). DFD pada sistem deteksi kemiripan kata hanya melibatkan satu entitas yaitu *user*.

#### 3.2.2.1. Data Context Diagram

*Data Context Diagram* (DCD) atau DFD level 0 pada Sistem Deteksi Kemiripan Kata dapat dilihat pada Gambar 3.3.



Gambar 3.3 *Data Context Diagram*

*Data Context Diagram* diatas menunjukkan aliran data yang masuk ke dalam sistem dari entitas luar dan aliran data yang keluar dari sistem menuju ke entitas luar. Penjelasan dari *Data Context Diagram* sistem deteksi kemiripan kata pada dua dokumen menggunakan algoritma *Rabin-Karp* adalah sebagai berikut :

1. Proses 0

Nama Proses : Sistem Deteksi Kemiripan Kata

Keterangan : Proses utama untuk deteksi kemiripan kata pada dua dokumen menggunakan *rabin-karp*

2. Entitas Luar

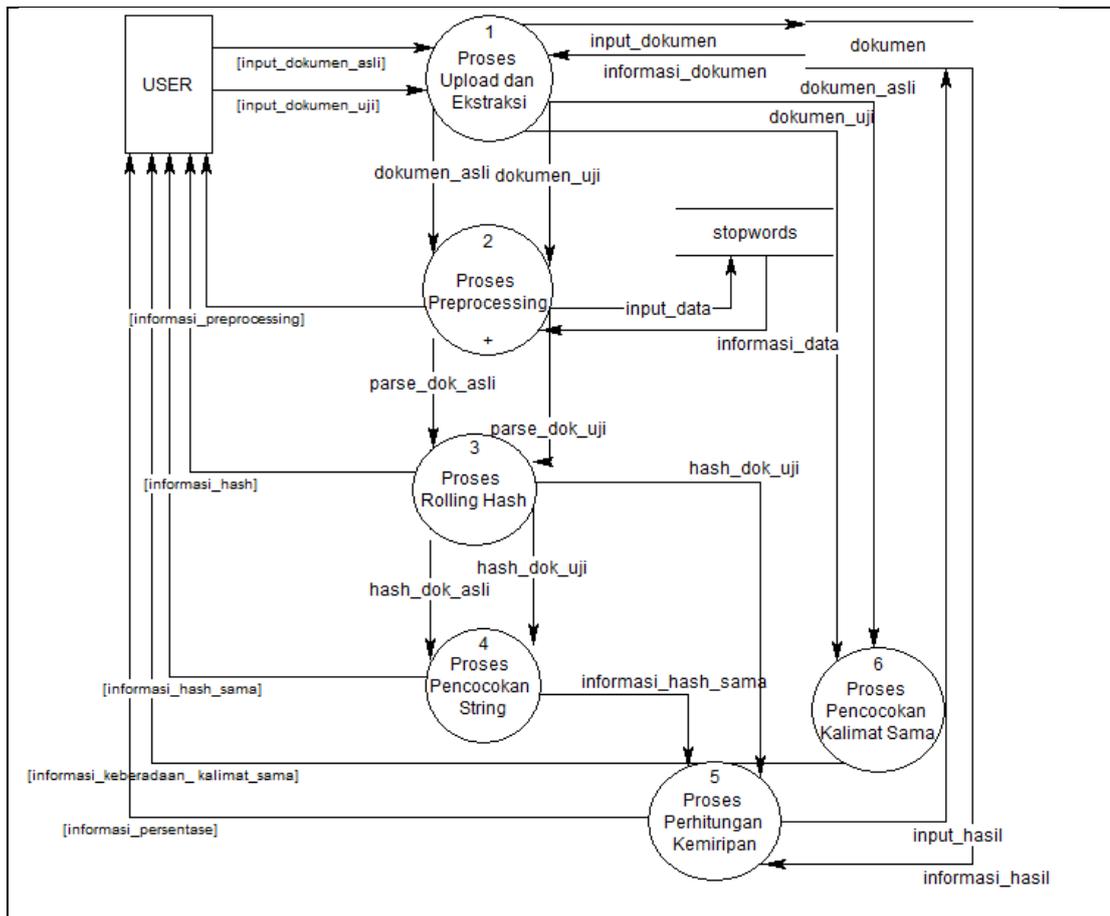
Nama Entitas : *User*

Keterangan : Pengguna sistem deteksi kemiripan kata yang memiliki peran terhadap sistem sebagai penginput dokumen dan melihat informasi hasil pengujian sistem.

Masukkan dari *user* : a. *input\_dokumen\_asli*  
 b. *input\_dokumen\_uji*

Keluaran ke *user* : a. *informasi\_preprocessing*  
 b. *informasi\_hash*  
 c. *informasi\_hash\_sama*  
 d. *informasi\_persentase*  
 e. *informasi\_keberadaan\_kalimat\_sama*

3.2.2.2. DFD level 1



Gambar 3.4. Data Flow Diagram level 1

*Data Context Diagram* pada Gambar 3.3, dapat dipecah menjadi *Data Flow Diagram* yang lebih rinci lagi dalam DFD level 1, seperti yang ditunjukkan pada Gambar 3.4.

Pada Gambar 3.4. DFD level 1 memiliki enam proses. Penjelasan proses DFD level 1 sistem deteksi kemiripan kata adalah sebagai berikut :

1. Proses 1

Nama Proses : Proses *Upload* dan Ekstraksi

Masukkan dari user : a. *input\_dokumen\_asli*  
b. *input\_dokumen\_uji*

Keluaran ke user : -

Keterangan : Proses awal yang akan menerima masukkan dua dokumen berekstensi \*.docx. Penyimpanan dokumen menggunakan direktori dan *data store* DOKUMEN. Selanjutnya, teks dalam dokumen akan diekstraksi ke bentuk xml agar dapat dibaca oleh sistem.

2. Proses 2

Nama Proses : Proses *Preprocessing*

Masukkan dari user : -

Keluaran ke user : *informasi\_preprocessing*

Keterangan : Memproses teks dengan menghapus tanda baca dan membuang kata tidak penting. Hasil ekstraksi dokumen akan diproses dengan *stopwords*, *white insensitivity*, dan *parsing*.

3. Proses 3

Nama Proses : Proses *Rolling Hash*

Masukkan dari user : -

Keluaran ke user : *informasi\_hash*

Keterangan : Proses mentransformasi *string* yang sudah di *parsing* menjadi nilai *ascii*. Komputasi perhitungan nilai *hash* menggunakan *rolling hash* lebih cepat karena fungsi *hash* yang input-nya dikelompokkan ke dalam suatu blok akan digerakkan melewati *input* secara keseluruhan.

4. Proses 4

Nama Proses : Proses Pencocokkan String

Masukkan dari user : -

Keluaran ke user : informasi\_hash\_sama

Keterangan : Proses utama yang akan membandingkan hasil *rolling hash* dua dokumen. Perbandingannya dengan metode pencocokkan string, yaitu algoritma *Rabin-Karp*. Sehingga menghasilkan nilai *hash\_sama*.

5. Proses 5

Nama Proses : Proses Perhitungan Kemiripan

Masukkan dari user : -

Keluaran ke user : informasi\_persentase

Keterangan : Proses perhitungan nilai *hash\_sama* dan nilai *hashing*. Hasil perhitungan akan disimpan pada *data store* DOKUMEN.

6. Proses 6

Nama Proses : Proses Pencocokkan Kalimat Sama

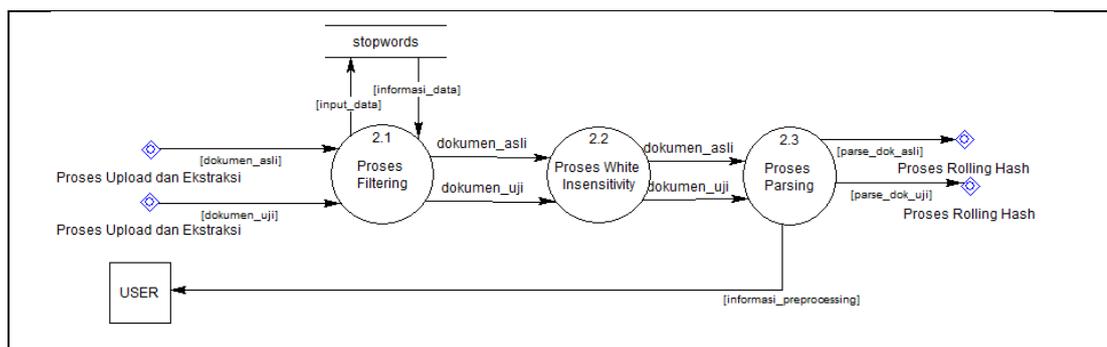
Masukkan dari user : -

Keluaran ke user : informasi\_keberadaan\_kalimat\_sama

Keterangan : Proses akhir yang dapat menampilkan kemiripan kalimat pada kedua dokumen. Data diambil dari hasil *parsing*, isi dokumen dan hasil pencocokkan *string*.

### 3.2.2.3. DFD level 2 Sub Proses *Preprocessing*

Pada DFD level 1 Proses 2 dapat dipecah menjadi proses yang lebih rinci lagi dalam DFD level 2 Sub Proses *Preprocessing* dapat dilihat pada Gambar 3.5.



Gambar 3.5. DFD level 2 Proses *Preprocessing*

DFD level 2 Sub Proses *Preprocessing* memiliki tiga proses yang melibatkan *data store* STOPWORDS. Berikut adalah penjelasan proses DFD level 2 Sub Proses *Preprocessing* :

1. Proses 2.1

Nama Proses : Proses *Filtering*

Masukkan dari user : -

Keluaran ke user : -

Keterangan : Menghapus kata-kata tidak penting. Dengan mencocokkan data pada *data store* STOPWORDS. Apabila terdapat kata pada *data store* tersebut maka kata tersebut dihapus.

2. Proses 2.2

Nama Proses : Proses *White Insensitivity*

Masukkan dari user : -

Keluaran ke user : -

Keterangan : Menghapus tanda baca, *spasi* dan mengubah menjadi huruf kecil.

3. Proses 2.3

Nama Proses : Proses *Parsing*

Masukkan dari user : -

Keluaran ke user : *informasi\_preprocessing*

Keterangan : Proses pemecahan kata menjadi 5 ukuran kata/ *parse*.

### 3.3. Desain Sistem

Tahap desain memiliki tujuan untuk menentukan kondisi akhir yang diharapkan dari sistem dan merumuskan cara yang harus dilakukan untuk memperoleh hasil tersebut. Tahap desain untuk sistem deteksi kemiripan kata meliputi perancangan proses sistem deteksi kemiripan kata dan perancangan antarmuka sistem.

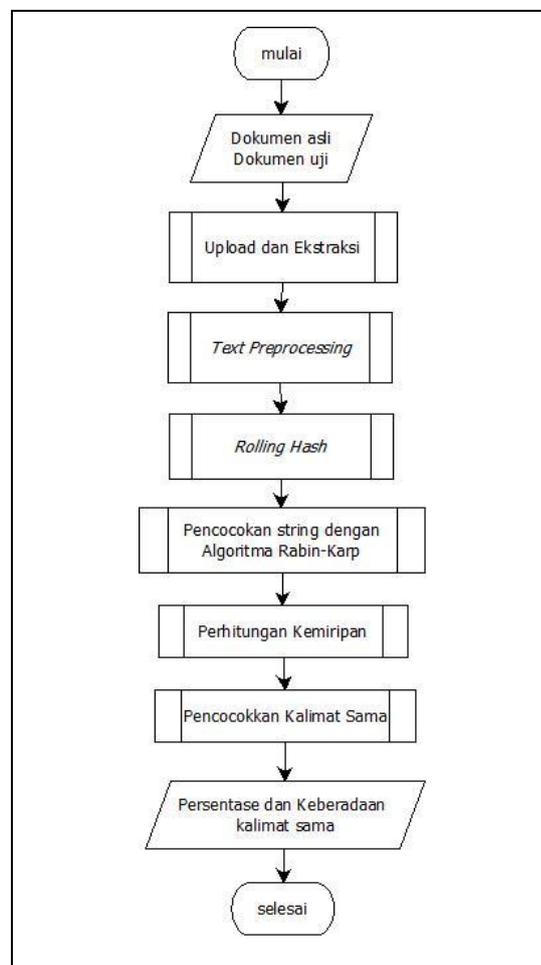
#### 3.3.1. Perancangan Proses Sistem Deteksi Kemiripan Kata

Perancangan proses sistem deteksi kemiripan kata pada dua dokumen menggunakan Algoritma *Rabin-Karp* yang merupakan gambaran dari kebutuhan dalam perangkat lunak sebelum memulai pembuatan kode. Perancangan tersebut

meliputi diagram alur (*flowchart*) dari sistem deteksi kemiripan kata. Terdapat 7 proses utama yang ada dalam sistem deteksi kemiripan kata, yaitu :

1. Proses *Upload* dan Ekstraksi
2. Proses *Text Preprocessing*
3. Proses *Rolling Hash*
4. Proses Pencocokkan *String* dengan Algoritma *Rabin-Karp*
5. Proses Perhitungan Kemiripan
6. Proses Pencocokkan Kalimat Sama

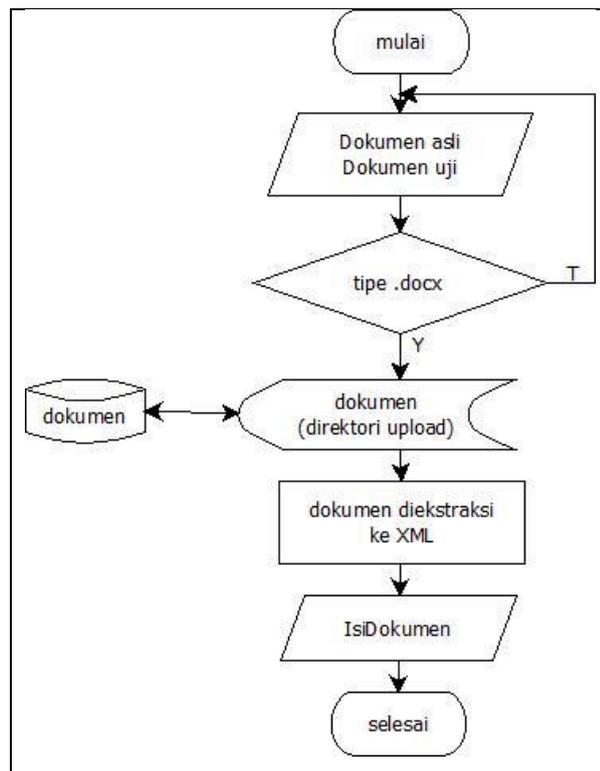
*Flowchart* sistem deteksi kemiripan kata pada dua dokumen menggunakan algoritma *Rabin-Karp*, dapat dilihat pada Gambar 3.6. *Flowchart* sistem deteksi kemiripan kata pada dua dokumen menggunakan algoritma *Rabin-Karp* yang digambarkan secara keseluruhan pada Lampiran 3.



Gambar 3.6. *Flowchart* Sistem Deteksi Kemiripan pada Dua Dokumen

### 3.3.1.1. Proses *Upload* dan Ekstraksi

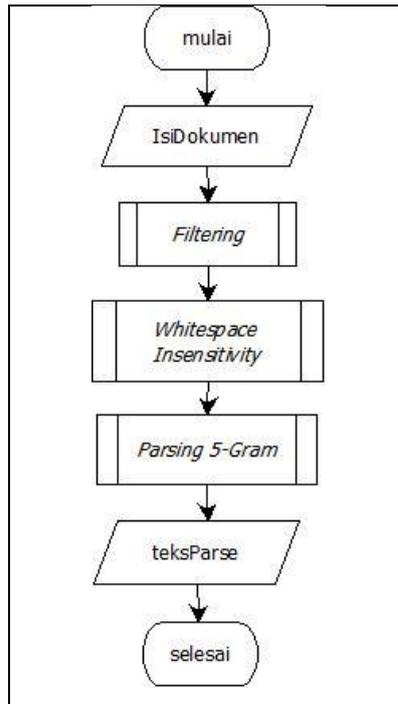
Proses *Upload* merupakan proses awal sistem, yang mana proses ini menerima masukkan file. Pada penelitian ini, dokumen yang dimasukkan berekstensi \*.docx. Penyimpanan dokumen terdapat pada direktori (*upload/*) dan *data store* "DOKUMEN". Setelah itu, dilakukan proses ekstraksi dokumen dengan mengambil data *file* dokumen pada direktori *upload*. Proses ini bertujuan untuk mengubah data dokumen (\*.docx) menjadi XML, sehingga dapat terbaca oleh sistem. *Output* dari proses ini adalah isiDokumen. Detail proses *upload* dan ekstraksi dapat dilihat pada Gambar 3.7.



Gambar 3.7 *Flowchart* Proses *Upload* dan Ekstraksi

### 3.3.1.2. Proses *Text Preprocessing*

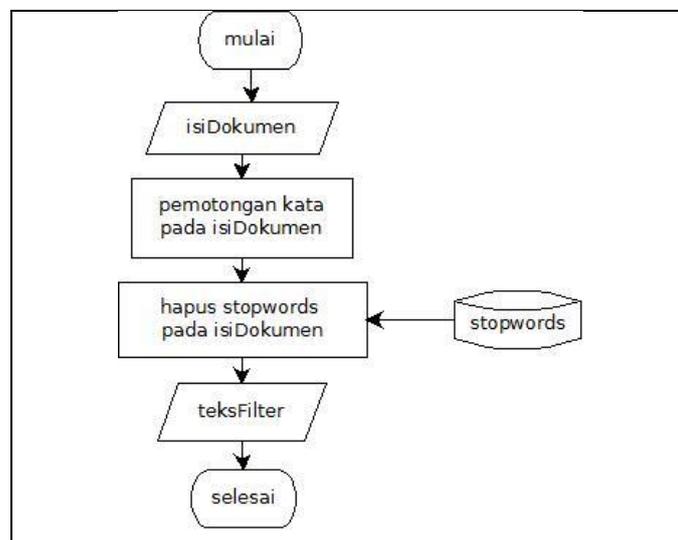
Dokumen yang telah melalui tahap *upload* dan ekstraksi kemudian di *preprocessing*. *Output* pada Gambar 3.7. tersebut digunakan sebagai masukan dari proses ini. Selanjutnya, terdapat tiga sub proses *text preprocessing* yaitu *filtering*, *whitespace Insensitivity*, dan *parsing 5-gram* yang dapat dilihat pada Gambar 3.8. *Output* proses *text preprocessing* dari hasil proses *Parsing 5-gram*.



Gambar 3.8. *Flowchart* Proses *Preprocessing*

### 3.3.1.2.1. Proses *Filtering*

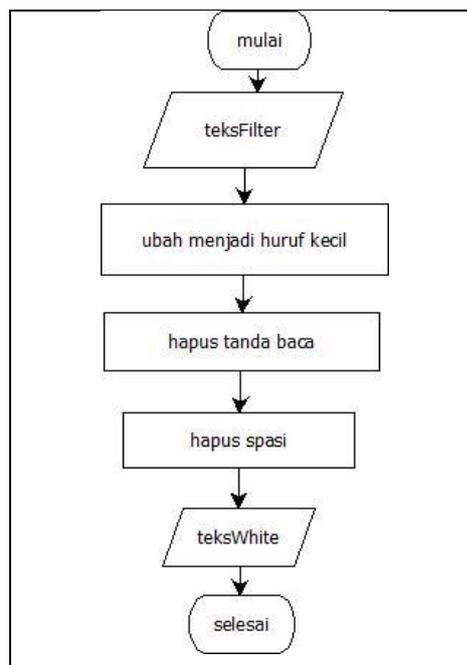
Proses awal dari proses *preprocessing* yaitu proses *filtering*. Pada Gambar 3.8. masukkan berupa isiDokumen. Proses *filtering* yaitu dengan memotong kata-kata pada isiDokumen dan dilanjutkan dengan proses menghapus *stopwords* yang berasal dari *data store* “STOPWORDS” pada isiDokumen yang telah melalui proses pemotongan. Keluaran berupa hasil isiDokumen yang telah melalui proses penghapusan. Detail proses *filtering* dapat dilihat pada Gambar 3.9.



Gambar 3.9. *Flowchart* Proses *Filtering*

### 3.3.1.2.2. Proses *Whitespace Insensitivity*

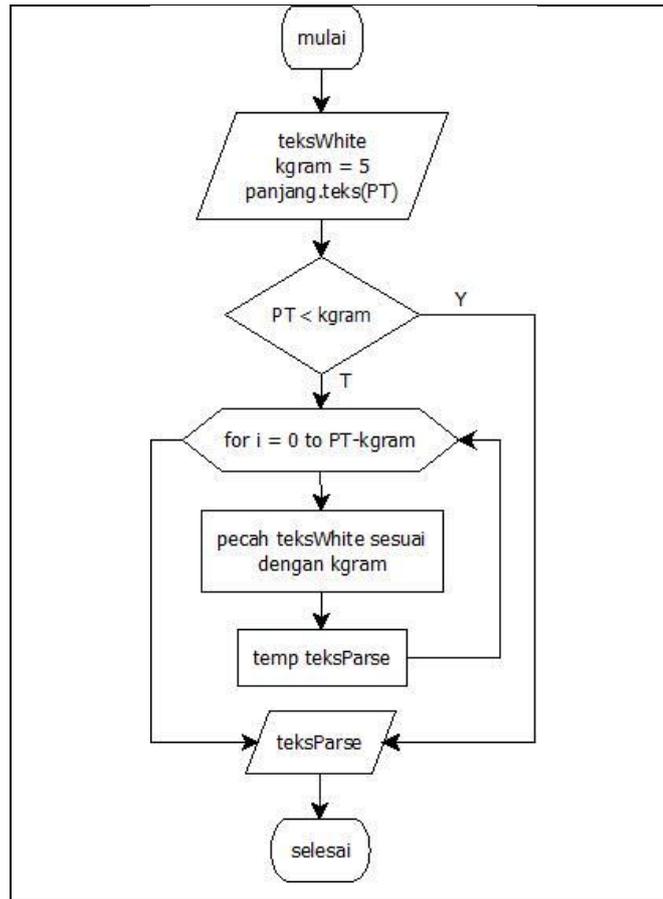
Pada Gambar 3.9. Proses *text preprocessing* yang telah melalui proses *filtering* mempunyai keluaran teksFilter. Keluaran tersebut sebagai masukan proses *whitespace insensitivity*. Masukan teksFilter diubah menjadi huruf kecil (*lower case*) kemudian tanda baca dan *spasi* yang terdapat pada teksFilter akan dihapus. Keluaran berupa teks yang tidak memiliki tanda baca. Detail proses *whitespace Insensitivity* dapat dilihat pada Gambar 3.10.



Gambar 3.10. Flowchart Proses *Whitespace Insensitivity*

### 3.3.1.2.3. Proses *Parsing 5-gram*

Proses akhir *text preprocessing* adalah proses *parsing*. Hasil keluaran dari proses *white insensitivity* pada Gambar 3.10. digunakan sebagai masukan *Parsing 5-gram*. Masukan berupa teksWhite/ teks dan nilai *kgram* yang digunakan sebanyak *5-gram*. Proses awal *parsing 5-gram* terdapat struktur pemilihan, jika panjang teks kurang dari *kgram* maka hasil teks disimpan sebagai *output* teksParse, jika tidak maka dilakukan pengulangan untuk memecah teks sesuai dengan panjang *gram* sebanyak 5 ukuran kata, sampai batas nilai *i*. Hasil pemecahan kata dimasukkan ke dalam *output* teksParse. Detail proses *parsing 5-gram* dapat dilihat pada Gambar 3.11.

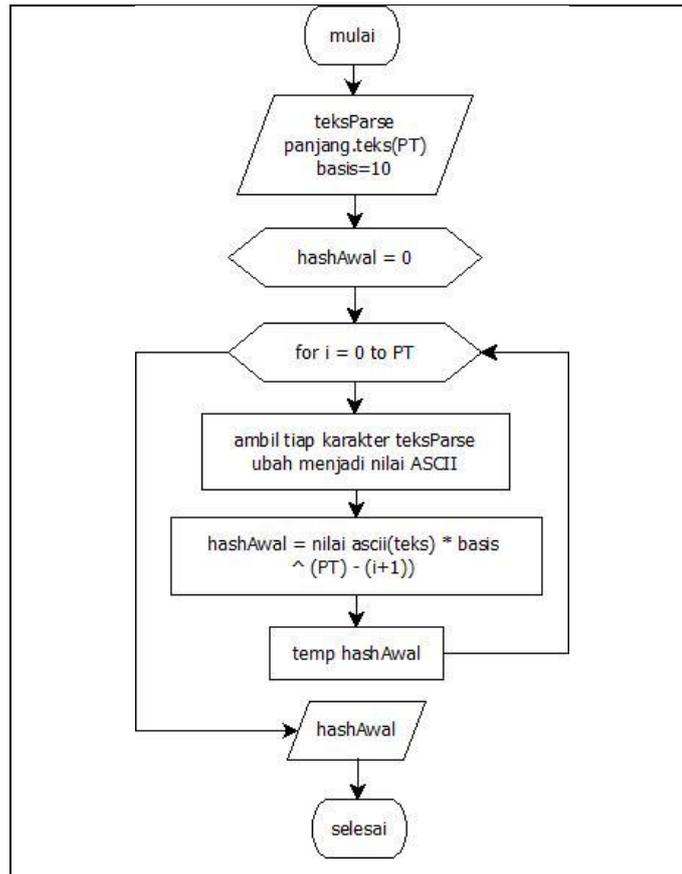


Gambar 3.11. Flowchart Proses Parsing 5-gram

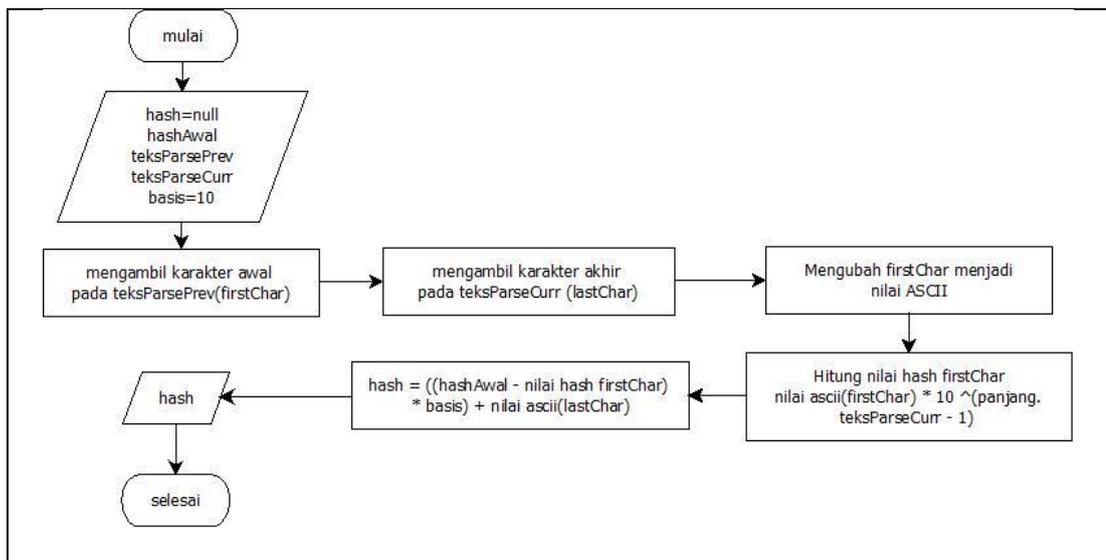
### 3.3.1.3. Proses Rolling Hash

Dokumen yang telah mengalami tahap *text preprocessing* akan langsung diproses menggunakan langkah Algoritma *Rabin-Karp*. Langkah yang diambil dalam perbandingan dokumen adalah melakukan proses *rolling hash*. *Rolling Hash* terdapat tiga proses, yaitu proses *hashing*, proses *rolling hash*, dan proses *pergeseran hash*. Penentuan *hash* awal menggunakan proses *hashing* dapat dilihat pada Gambar 3.12. dan untuk hash selanjutnya menggunakan proses *rolling hash* pada Gambar 3.13.

Pada proses *hashing*, masukkan teksParse berasal dari keluaran proses *parsing k-gram* pada Gambar 3.11. Inisialisasi awal hashAwal bernilai 0 dan nilai basis yang digunakan bernilai 10. Dilakukan perulangan menghitung setiap karakter teksParse untuk diubah menjadi nilai ASCII, rumus *hashing* dapat dilihat pada Persamaan 2.1. Hasil proses *hashing* menjadi hashAwal untuk proses selanjutnya. Detail proses *hashing* dapat dilihat pada Gambar 3.12.



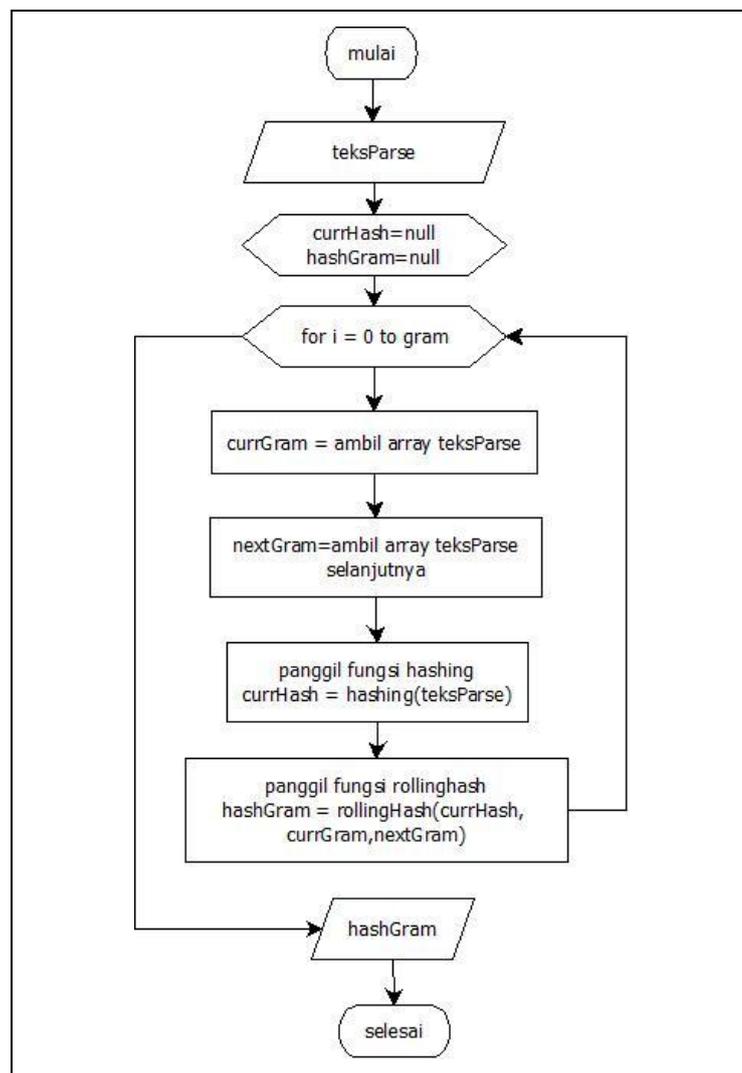
Gambar 3.12. Flowchart Proses Hashing



Gambar 3.13. Flowchart Proses Rolling Hash

Proses selanjutnya yaitu proses *rolling hash*. Rumus *rolling hash* dapat dilihat pada Persamaan 2.1. Masukkan proses ini adalah inisialisasi *hash* bernilai *null*/kosong, *hashAwal* dari *output hashing*, *teksParsePrev*, *teksParseCurr*, dan *basis* bernilai 10.

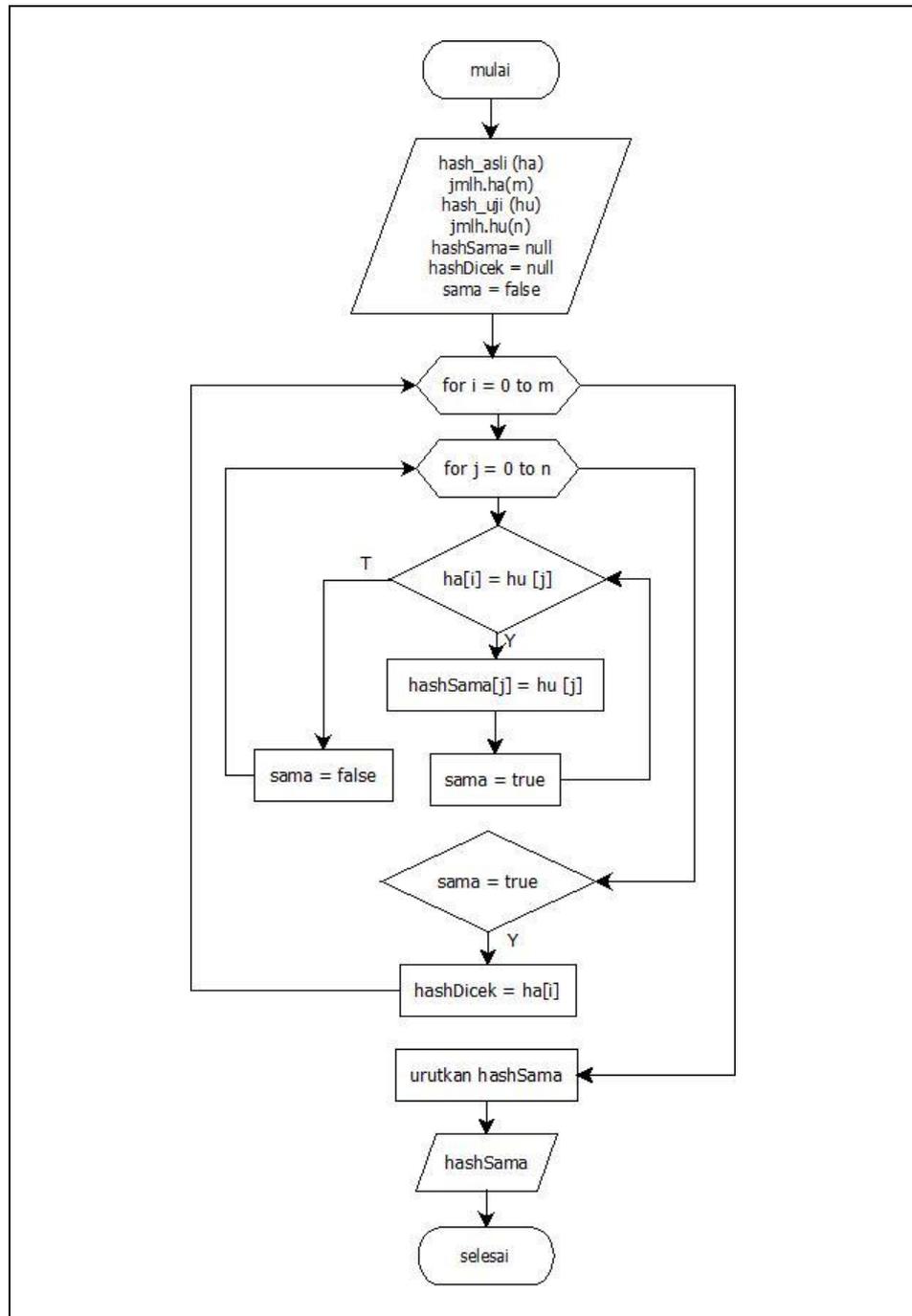
teksParsePrev adalah mengambil nilai parse sebelumnya dan teksParseCurr adalah mengambil nilai *parse* yang diproses. Proses ini dilanjutkan dengan mengambil huruf awal pada teksParsePrev dan mengambil huruf terakhir pada teksParseCurr. Kemudian, proses ini akan menghitung nilai *hash* huruf awal firstChar. Perhitungan tersebut kemudian dikurangkan dengan nilai hashAwal dan ditambah dengan nilai *ascii* dari lastChar. Proses pergeseran ini terus dilakukan sampai *string* selesai. Detail proses *rolling hash* terdapat pada Gambar 3.13.



Gambar 3.14. *Flowchart* Proses Pergeseran Hash

Proses utama untuk menghasilkan nilai hash yaitu dengan memanggil proses hashing dan rolling hash pada proses pergeseran hash. Masukkan teksParse sebagai variabel masukkan proses *hashing*. Dengan melakukan perulangan untuk memproses hash dari proses *hashing* ke proses *rolling hash*. Menghasilkan rangkaian hash yang dapat dilihat pada Gambar 3.14

### 3.3.1.4. Proses Pencocokkan String dengan Algoritma Rabin-Karp

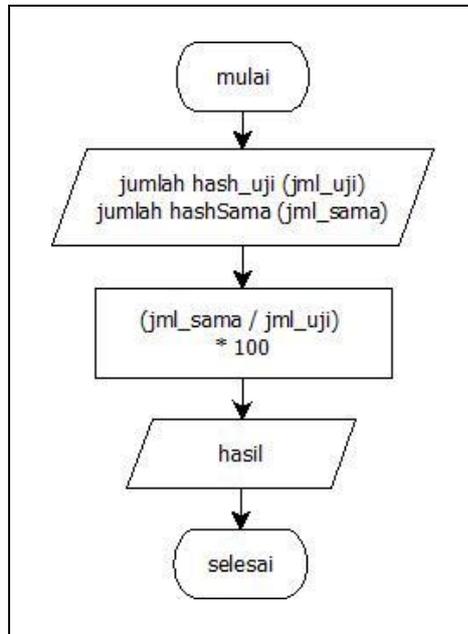


Gambar 3.15 Flowchart Perancangan Proses Pencocokkan String Algoritma Rabin-Karp

Proses ini sesuai dengan *pseudocode* algoritma Rabin-Karp pada Kode 2.1. Masukkan *hash* telah diketahui dari *output* proses *rolling hash*. Masukkan proses pencocokkan adalah string nilai *hash* dokumen asli, nilai *hash* dokumen uji, inisialisasi awal *hashSama* bernilai *null*, *hashDicek* bernilai *null* dan variabel *sama* bernilai *false*.

### 3.3.1.5. Proses Perhitungan Kemiripan

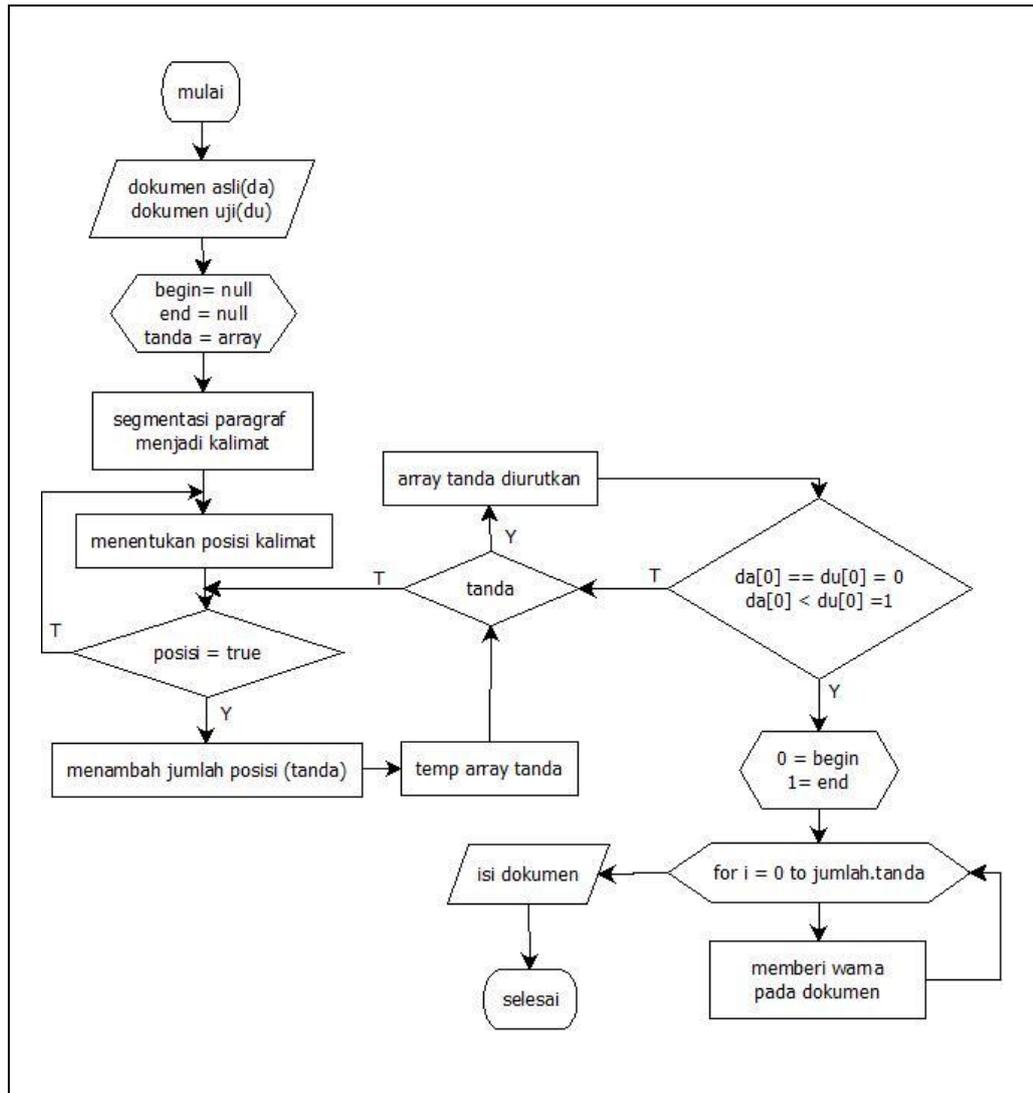
Nilai kemiripan dihitung berdasarkan pada banyaknya jumlah hash yang telah diproses oleh algoritma Rabin-Karp dan jumlah hash pada dokumen uji. Dengan membagi jumlah hash sama dan jumlah hash dokumen uji, lalu dikalikan 100%, dapat dilihat pada Persamaan 3.1. Proses ini menghasilkan persentase kemiripan kata. Detail proses perhitungan kemiripan dapat dilihat pada Gambar 3.16.



Gambar 3.16. *Flowchart* Proses Hitung Kemiripan

### 3.3.1.6. Proses Pencocokkan Kalimat Sama

Masukkan dari proses pencocokkan kalimat sama berupa isiDokumen pada dokumen asli dan dokumen uji. Pada proses pertama memotong paragraf menjadi kalimat pada salah satu dokumen. Kemudian menentukan posisi kata awal sampai kata akhir dari kalimat. Jika terdapat posisi maka menjumlahkan posisi selanjutnya. Posisi disimpan pada variabel tanda. Setelah itu, array tanda diurutkan, kemudian kedua dokumen array dicocokkan. Apabila dokumen array sama akan diketahui nilai 0, namun jika dokumen berbeda bernilai 1. Inisialisasi nilai 0 berupa variabel begin, nilai 1 berupa variabel end. Hasil tanda berupa isi dokumen yang diberi warna sesuai keberadaan kalimat sama. Detail proses pencocokkan kalimat sama dapat dilihat pada Gambar 3.17.



Gambar 3.17. *Flowchart* Proses Pencocokkan Kalimat Sama

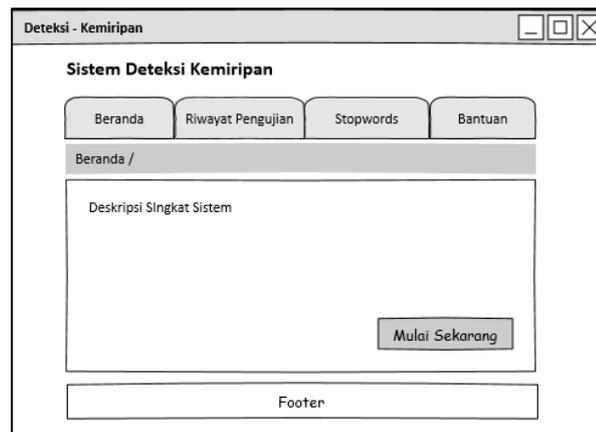
### 3.3.2. Perancangan Antarmuka Sistem

Rancangan antarmuka dibuat sederhana untuk mempermudah pengguna dalam penggunaannya. Antarmuka untuk sistem deteksi kemiripan kata adalah sebagai berikut :

1. Antarmuka Halaman Depan
2. Antarmuka *Upload* Dokumen
3. Antarmuka Pemrosesan Dokumen
4. Antarmuka Hasil Pengujian
5. Antarmuka Riwayat Pengujian
6. Antarmuka *Stopwords*
7. Antarmuka Bantuan

### 3.3.2.1. Antarmuka Halaman Depan

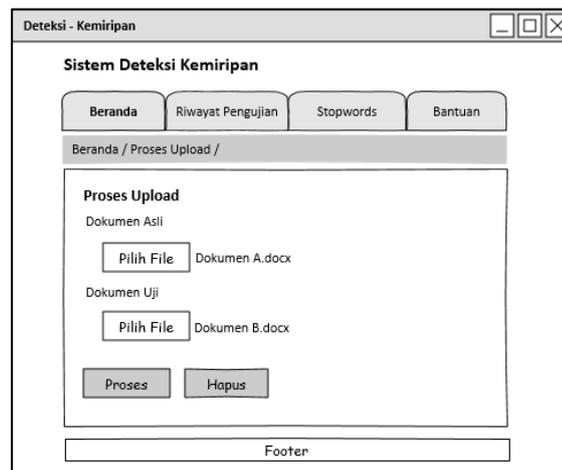
Jendela utama Sistem Deteksi Kemiripan Kata terdiri dari 4 menu yaitu Menu Beranda, Riwayat Pengujian, Stopwords dan Bantuan. Halaman Depan Sistem terdapat pada Menu Beranda. Menu Beranda menampilkan deskripsi singkat sistem, terdapat *button* Mulai Sekarang untuk memulai sistem ke tahap selanjutnya. Menu Beranda mempunyai 3 halaman selanjutnya, yaitu Proses *Upload*, Pemrosesan Dokumen, dan Hasil Pengujian. Desain Antarmuka Halaman Depan dapat dilihat pada Gambar 3.18.



Gambar 3.18. Desain Antarmuka Halaman Depan

### 3.3.2.2. Antarmuka *Upload* Dokumen

Halaman *Upload* Dokumen terdapat pilihan untuk memasukkan dua *file* yaitu dokumen yang asli dan dokumen yang diuji. *Button* Proses untuk memproses kedua dokumen ke halaman Pemrosesan Dokumen, sedangkan *button* Hapus untuk membatalkan *file*. Desain Antarmuka *Upload* Dokumen dapat dilihat pada Gambar 3.19.



Gambar 3.19. Desain Antarmuka *Upload* Dokumen

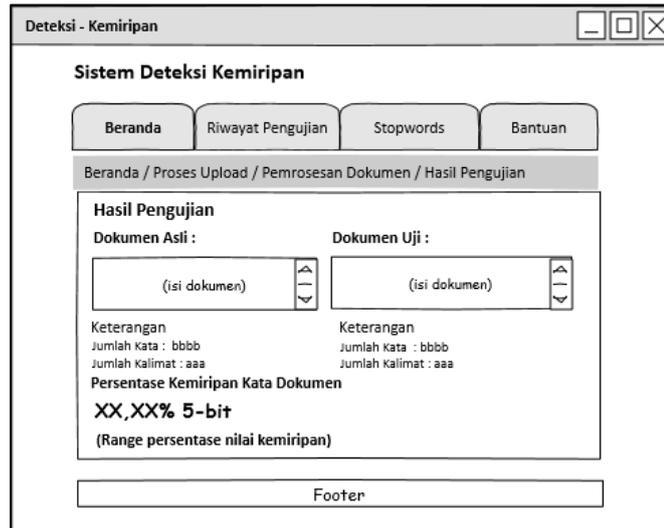
### 3.3.2.3. Antarmuka Pemrosesan Dokumen

Halaman Pemrosesan Dokumen menampilkan proses dokumen yang telah diproses sebagaimana diperlihatkan pada Gambar 3.20. Prosesnya terdiri dari tahap praproses, *parse* kata *5-gram* dan *hash*, dan hasil pencocokkan *string* (Algoritma *Rabin-Karp*). *Form* praproses berisi dokumen yang telah melalui proses *filtering* dan *whitespace insensitivity*. Tabel *parse* kata *5-gram* dan *hash* menampilkan kata-kata pada dokumen asli dan dokumen uji yang telah diproses *rolling hash* sehingga menghasilkan nilai *hash*. *Form* Hasil Pencocokkan *String* menampilkan *hash* yang sama antara nilai *hash* kedua dokumen. Untuk melanjutkan ke hasil selanjutnya pengguna dapat menekan *button* Selanjutnya.

Gambar 3.20. Desain Antarmuka Pemrosesan Dokumen

### 3.3.2.4. Antarmuka Hasil Pengujian

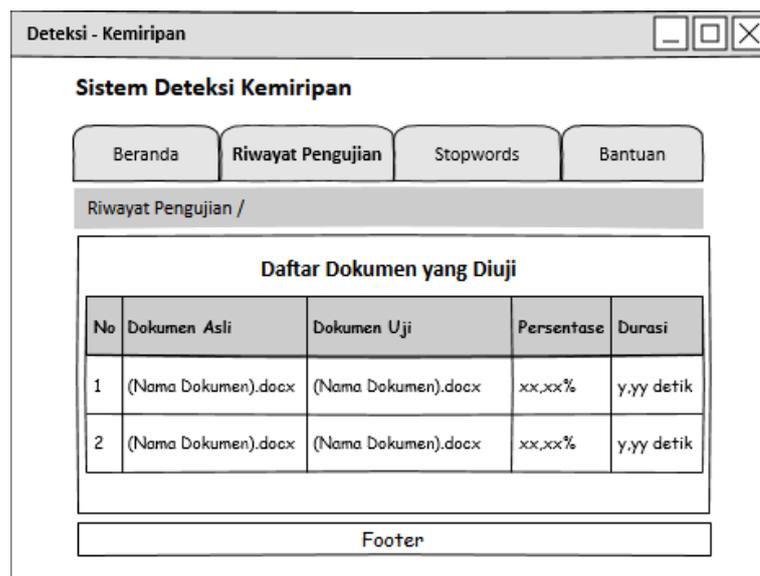
Halaman Hasil Pengujian menampilkan hasil kemiripan kalimat antara kedua dokumen dan persentase kemiripan. Terdapat dua *form* berisi hasil kemiripan dokumen asli dan dokumen uji beserta keterangan jumlah kalimat dan kata. Pada persentase kemiripan kata terdapat hasil persentase kemiripan dokumen dan keterangan hasil kemiripan. Desain Antarmuka Hasil Pengujian dapat dilihat pada Gambar 3.21.



Gambar 3.21. Desain Antarmuka Hasil Pengujian

### 3.3.2.5. Antarmuka Daftar Dokumen yang Diuji

Menu Riwayat Pengujian menampilkan daftar dokumen yang telah diuji dalam bentuk tabel. Tabel tersebut terdiri dari dokumen asli, dokumen uji, persentase dan durasi. Desain Antarmuka Daftar Dokumen yang Diuji dapat dilihat pada Gambar 3.22.

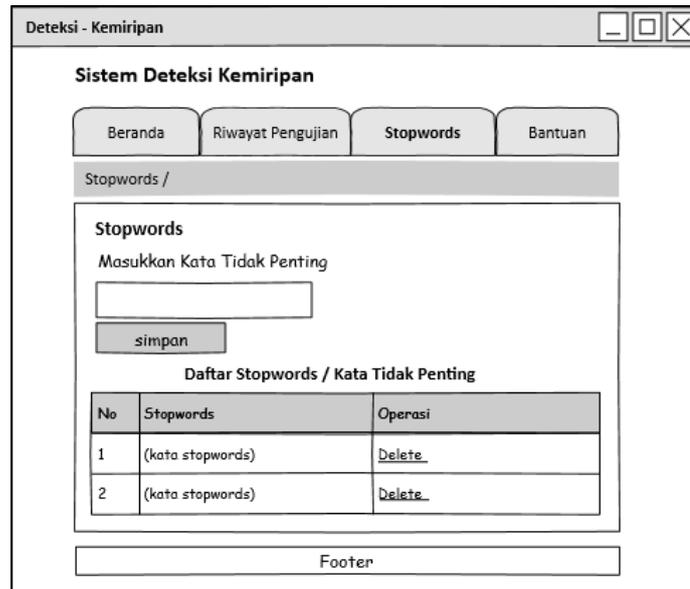


Gambar 3.22. Desain Antarmuka Daftar Dokumen yang Diuji

### 3.3.2.6. Antarmuka *Stopwords*

Menu *Stopwords* menampilkan *form* untuk memasukkan kata tidak penting dan daftar *stopwords/* kata tidak penting. Tabel Daftar *Stopwords* terdiri dari kolom *stopwords* dan operasi. Kolom *Stopwords* berisi kata-kata yang dimasukkan,

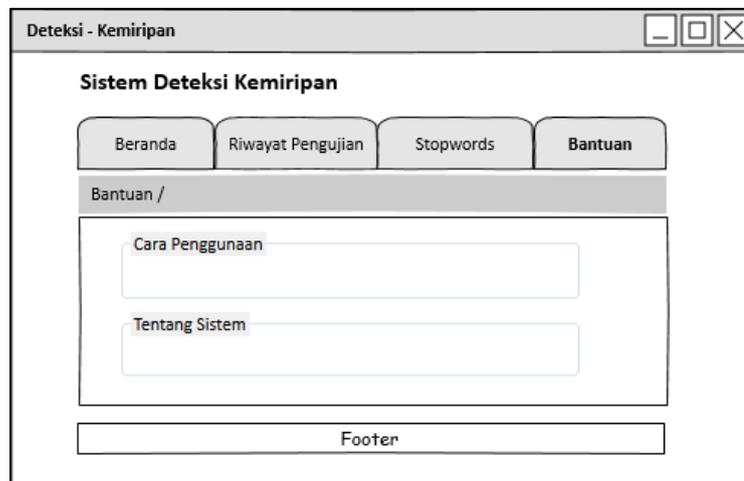
sedangkan kolom operasi untuk menghapus kata tidak penting. Desain Antarmuka Daftar *Stopwords* dapat dilihat pada Gambar 3.23.



Gambar 3.23. Desain Antarmuka *Stopwords*

### 3.3.2.7. Antarmuka Bantuan

Menu Bantuan berfungsi menjelaskan penggunaan sistem deteksi kemiripan kata bagi pengguna (*User*). Pada halaman ini, telah dijelaskan proses sistem berjalan serta ketentuan parameter *input*. Desain Antarmuka Bantuan dapat dilihat pada Gambar 3.24.



Gambar 3.24. Desain Antarmuka Bantuan

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

Bab ini menjelaskan mengenai implementasi dari perancangan data, perancangan fungsi dan perancangan antarmuka dari Sistem Deteksi Kemiripan Kata pada dua dokumen menggunakan Algoritma *Rabin-Karp*, serta pengujian terhadap sistem dengan metode *blackbox*.

#### **4.1. Implementasi**

Sub bab ini menyajikan implementasi sistem, meliputi spesifikasi perangkat yang digunakan untuk mengembangkan Sistem Deteksi Kemiripan Kata, implementasi basis data, implementasi fungsional dan implementasi antarmuka.

##### **4.1.1. Spesifikasi Perangkat**

Adapun spesifikasi perangkat keras dan perangkat lunak yang digunakan untuk membangun Sistem Deteksi Kemiripan Kata adalah sebagai berikut :

1. CPU : Intel(R) Core(TM) i5 2410M @ 2,3 GHz, RAM 4,00 GB
2. Sistem Operasi : Windows 7 Professional
3. Bahasa Pemrograman : PHP 5.4.7, HTML, CSS, Java Script.
4. Alat bantu pemrograman : Notepad++ 5.9.3, Mozilla Firefox 30, Google Chrome 35.0, Xampp (Apache Web Server, MySQL)

##### **4.1.2. Implementasi Basis Data**

Pada implementasi rancangan data menggunakan permodelan data, yang telah dibuat sebelumnya. Sebagaimana dijelaskan pada Gambar 3.2. bahwa dalam Sistem Deteksi Kemiripan Kata ini digunakan dua entitas data yaitu dokumen dan *stopwords*. Setiap entitas data tersebut dibuat menjadi tabel-tabel dalam basis data sistem. Tabel 4.1 menunjukkan tabel yang digunakan dalam basis data Sistem Deteksi Kemiripan Kata.

Tabel 4.1. Daftar Tabel Sistem Deteksi Kemiripan Kata

Nama Tabel	Field	Primary Key	Deskripsi
<b>dokumen</b>	id_dok	id_dok	Berisi informasi tentang nama dokumen dan hasil yang telah diproses sistem
	nm_dok_asli		
	nm_dok_uji		
	persentase		
	durasi		
<b>stopwords</b>	word		Berisi daftar kata tidak penting

Struktur Tabel Sistem Deteksi Kemiripan Kata dapat dilihat pada Tabel 4.2 dan Tabel 4.3. Tabel 4.2 menjelaskan tentang *field* yang terdapat pada tabel DOKUMEN beserta deskripsi dan tipe *field* tersebut. Tabel 4.3 menjelaskan tentang *field* yang terdapat pada tabel STOPWORDS beserta deskripsi dan tipe *field* tersebut.

Tabel 4.2. Struktur Tabel DOKUMEN

Field	Deskripsi	Tipe	Null	Keterangan
<b>id_dok</b>	id untuk dokumen	int(10)	No	<i>Primary key, auto-increment</i>
<b>nm_dok_asli</b>	Nama untuk setiap dokumen asli	varchar (50)	No	-
<b>nm_dok_uji</b>	Nama untuk setiap dokumen uji	varchar(50)	No	-
<b>persentase</b>	Hasil perhitungan kemiripan	char (6)	No	-
<b>durasi</b>	waktu proses sistem	char(10)	No	-

Tabel 4.3. Struktur Tabel STOPWORDS

Field	Deskripsi	Tipe	Null	Keterangan
<b>word</b>	Daftar kata <i>stopwords</i>	varchar (100)	No	-

### 4.1.3. Implementasi Fungsi

Implementasi fungsional merupakan hasil transformasi dari perancangan fungsional yang telah dijelaskan pada sub bab 3.3.1. Pada Sistem Deteksi Kemiripan Kata ini terdapat beberapa fungsi seperti fungsi *upload* dan ekstraksi, fungsi *text preprocessing*, fungsi *rolling hash*, fungsi pencocokan *string*, fungsi perhitungan kemiripan dan fungsi pencocokan kalimat sama.

#### 4.1.3.1. Fungsi Upload dan Ekstraksi

Fungsi *Upload* akan memasukkan dua *file query* ke direktori dan *database*. Dua *file* ini terdiri dari dokumen asli dan dokumen uji. Proses *input* masing-masing dokumen hanya dapat meng*upload* satu *file* pada setiap dokumen. Lalu sistem menyimpan ke dalam direktori sistem pada folder “upload/ ” dan nama dokumen ke dalam database. Kemudian ke dua dokumen ini akan melakukan proses ekstraksi

dengan mengubah *file* berekstensi \*.docx menjadi *file* xml. Fungsi *upload* dan ekstraksi dapat dilihat pada Kode 4.1.

```

$file = $_FILES['docfile1'];
$nama_file1 = $file['name']; //dokumen asli
$file2 = $_FILES['docfile2'];
$nama_file2 = $file2['name']; //dokumen uji
$nama_tmp = $file['tmp_name'];
$nama_tmp2 = $file2['tmp_name'];
$upload_dir = "upload/";
move_uploaded_file($nama_tmp,$upload_dir.$nama_file1);
move_uploaded_file($nama_tmp2,$upload_dir.$nama_file2);
$query="INSERT INTO dokumen (nm_dok_asli,nm_dok_uji) VALUES
('$nama_file1','$nama_file2')";
$result=mysql_query($query,$koneksi) or die ('error
kenapa?'.mysql_error());
if($result){
    $dir=$_SERVER['DOCUMENT_ROOT'].'/skripsi/d-plag v 2.0/upload/';
    $files=scandir($dir,1);
    $jumlah_file = count($files) - 2;
    //nama file
    $document1=$nama_file1;
    $document2=$nama_file2;
    //panggil fungsi extract text
    $doc1 = extracttext("upload/".$document1);
    $doc2 = extracttext("upload/".$document2);
}
//fungsi ekstrak text word ke XML
function extracttext($filename){
    //check file
    $ext=end(explode('.', $filename));
    if($ext=='docx'|'doc'){
        $dataFile="word/document.xml";
    }else{
        $dataFile="content.xml";
    }
    //buat zip
    $zip=new ZipArchive;
    //buka file cek folder
    if(true === $zip->open($filename)){
        if(($index=$zip->locateName($dataFile)) !==false){
            //index
            $text=$zip->getFromIndex($index);
            $xml= DOMDocument::loadXML($text,LIBXML_NOENT |
LIBXML_XINCLUDE | LIBXML_NOERROR | LIBXML_NOWARNING);
            $xml_data=$xml->saveXML();
            return strip_tags($xml_data);
        }
        $zip->close();
    }
    return "File not found";
}

```

Kode 4.1. Fungsi *Upload* dan Ekstraksi

#### 4.1.3.2. Fungsi *Text Preprocessing*

Pada Fungsi *Text Preprocessing* ini akan memproses teks dengan membersihkan tanda baca, kata-kata tidak penting dan pemecahan kata. Fungsi yang digunakan dalam tahapan ini terdiri dari tiga yaitu fungsi *white insensitivity*, fungsi *filtering*, fungsi *parsing 5-gram*.

##### 1. Fungsi *Filtering*

Fungsi dari *filtering* adalah dengan memotong isi dokumen dan mencocokkannya dalam daftar stopwords. Apabila terdapat kesamaan stopwords dengan isi stopwords pada dokumen, maka kata pada dokumen akan dihapus. Fungsi dari *filtering* dapat dilihat pada Kode 4.2.

```
function filtering($string){
    //mengambil daftar kata di tabel stopwords
    $koneksi=koneksiDB();
    $query = "SELECT * FROM stopwords";
    $hasil=mysql_query($query,$koneksi) or die ('error
kenapa?'.mysql_error());
    if($hasil){
        while ($row = mysql_fetch_array($hasil)){
            $stopword[] = $row['word'];
        }
    }
    //segmentasi isi dokumen ke kata
    $array = explode(" ", $string);
    //hapus stopwords
    $arrays=array_diff($array,$stopword);
    $result=implode(" ",$arrays); //dibuat spasi
    return $result;
}
```

Kode 4.2. Fungsi *Filtering*

##### 2. Fungsi *White Insensitivity*

Fungsi dari *white insensitivity* yaitu akan mengubah semua isi dokumen menjadi huruf kecil dan menghapus tanda baca. Fungsi dari *white insensitivity* dapat dilihat pada Kode 4.3.

```
function whiteInsensitive($teks) {
    $string      = strtolower($teks);
    $delimiter   = array(' ','.',',','!','"',"'",'-',
    '/','/','{','}','+', '_','!','@','#','$','%','^','&','*','(',')',' ',
    '?','>','<','[',']','|','\','~',';':','=','\\','\n','\r');
    $w          = str_replace($delimiter,"",$string);
    return $w;
}
```

Kode 4.3. Fungsi *White Insensitivity*

### 3. Fungsi *Parsing 5-gram*

Fungsi dari *parsing 5-gram* yaitu akan melakukan pemecahan kata sebanyak 5 huruf. Fungsi dari *parsing 5-gram* dapat dilihat pada Kode 4.4.

```
function kGram($teks) {
    $i = 0;
    $gram = 5;
    $length = strlen($teks);
    $teksSplit;
    if (strlen($teks) < $gram) {
        $teksSplit[] = $teks;
    } else {
        for ($i; $i <= $length - $gram; $i++) {
            $teksSplit[] = substr($teks, $i, $gram);
        }
    }
    return $teksSplit;
}
```

Kode 4.4. Fungsi *Parsing 5-gram*

#### 4.1.3.3. Fungsi *Rolling Hash*

Fungsi dari *Rolling Hash* yaitu akan menghitung hasil dari *parsing*, dari setiap *parse* yang dihitung akan menghasilkan nilai *hash*. Fungsi dari *Rolling Hash* dapat dilihat pada Kode 4.5.

```
function hashing($string) {
    $basis = 10;
    $pjpgKarakter = strlen($string);
    $hash = 0;
    for ($i = 0; $i < $pjpgKarakter; $i++) {
        $ascii = ord($string[$i]);
        $hash += $ascii * pow($basis, ($pjpgKarakter - ($i + 1)));
    }
    return $hash;
}

function rollHash($hashAwal, $stringPrev, $stringCurr){
    $hashe=null;
    $panjangCurr = strlen($stringCurr);
    //m dari menge ambil plg dpn untuk dikurangi
    //l dari enyal ambil paling belakang untuk ditambahi
    $firstChar = $stringPrev[0];
    $lastChar = $stringCurr[$panjangCurr - 1];
    $ascii=ord($firstChar);
    $basis = 10;
    $firstCharHash = $ascii * pow($basis, ($panjangCurr -
1));
    $hashe = (($hashAwal - $firstCharHash) * $basis) +
ord($lastChar);
    return $hashe;
}

function hasho($gram) {
```

```

$hashGram = null;
//$prevHash=null;
foreach($gram as $a => $teks){
    //$prevGram=current($gram);
    //$currGram=next($gram);

    $hashGram[] = hashing($teks);
}
//print_r($hashGram);
return $hashGram;
}

```

Kode 4.5 Fungsi *Rolling Hash*

#### 4.1.3.4. Fungsi Pencocokkan String

Fungsi dari Pencocokkan *String* yaitu melakukan pencocokan nilai *hash* antara dua *file*. Hasilnya berupa nilai *hash* yang sama. Fungsi dari Pencocokkan *String* dapat dilihat pada Kode 4.6.

```

function fingerprint($hash1, $hash2) {
    $fingerprint = null;
    $hashUdahDicek = null;
    $sama = false;
    $countHash1 = count($hash2);
    for ($i = 0; $i < count($hash1); $i++) {
        for ($j = 0; $j < $countHash1; $j++) {
            if ($hash1[$i] == $hash2[$j]) {
                $fingerprint[$j] = $hash2[$j];

                $sama = true;
            } else {
                $sama = false;
            }
        }
        if ($sama == true) {
            $hashUdahDicek[] = $hash1[$i];
        }
    }
    ksort($fingerprint);
    //print_r($fingerprint);
    return $fingerprint;
}

```

Kode 4.6. Fungsi Pencocokkan *String*

#### 4.1.3.5. Fungsi Perhitungan Kemiripan

Fungsi dari Perhitungan Kemiripan yaitu menghitung kemiripan dokumen uji dengan dokumen asli. Dengan membagi jumlah hash sama dengan jumlah hash dokumen uji untuk mendapatkan hasil persentase kemiripan kata. Fungsi dari Perhitungan Kemiripan dapat dilihat pada Kode 4.7.

```
function similarityCoefficient($fingerprint, $hash2) {
    return number_format(((count($fingerprint))/(count($hash2))
    )) * 100, 2, '.', '');
}
```

Kode 4.7. Fungsi Perhitungan Kemiripan

#### 4.1.3.6. Fungsi Pencocokan Kalimat Sama

Fungsi dari Pencocokan Kalimat Sama yaitu akan melakukan pemotongan kalimat kedua dokumen lalu mencocokkannya. Apabila kesamaan kalimat akan diberi tanda warna berbeda. Fungsi dari Pencocokkan kalimat sama dapat dilihat pada Kode 4.8.

```
function highlightKeywords($data, $key) {
    $keywords=explode(".", $key);
    $find = array();
    $replace = array();
    $begin = "<span class=\"keywordHighlight\">";
    $end = "</span>";
    $hits = array();
    foreach ($keywords as $kw) {
        $offset = 0;
        while (($pos = strpos($data, $kw, $offset)) !==
false) {
            $hits[] = array($pos, $pos + strlen($kw));
            $offset = $pos + 1;
        }
    }
    if ($hits) {
        usort($hits, function($a, $b) {
            if ($a[0] == $b[0]) {
                return 0;
            }
            return ($a[0] < $b[0]) ? -1 : 1;
        });
        $thisthat = array(0 => $begin, 1 => $end);
        for ($i = 0; $i < count($hits); $i++) {
            foreach ($thisthat as $key => $val) {
                $pos = $hits[$i][$key];
                $data = substr($data, 0, $pos) . $val
. substr($data, $pos);
                for ($j = 0; $j < count($hits); $j++) {
                    if ($hits[$j][0] >= $pos) {
                        $hits[$j][0] += strlen($val);
                    }
                    if ($hits[$j][1] >= $pos) {
                        $hits[$j][1] += strlen($val);
                    }
                }
            }
        }
    }
    return $data;
}
```

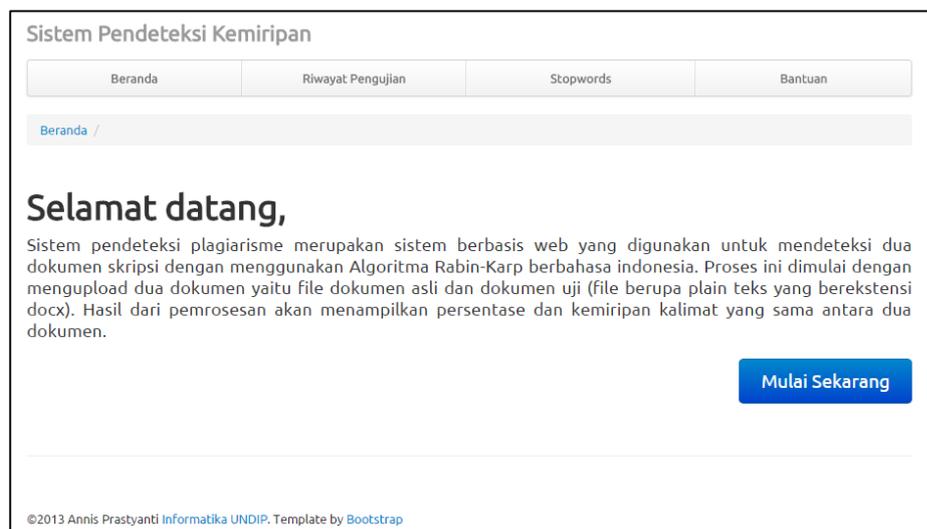
Kode 4.8. Fungsi Pencocokkan Kalimat Sama

#### 4.1.4. Implementasi Antarmuka

Implementasi Antarmuka merupakan implementasi dari perancangan antarmuka yang telah dijelaskan pada subbab 3.3.2. Implementasi antarmuka pada Sistem Deteksi Kemiripan Kata terdiri dari enam antarmuka yaitu halaman depan, *upload* dokumen, pemrosesan dokumen, hasil pengujian, riwayat pengujian, dan daftar *stopwords*.

##### 4.1.4.1. Implementasi Antarmuka Halaman Depan

Antarmuka awal sistem ditunjukkan dengan halaman *Index* yang memiliki menu yaitu Menu Beranda, Menu Riwayat Pengujian, Menu *Stopwords* dan Menu Bantuan. Penjelasan masing-masing menu sebagai berikut. Menu Bantuan untuk mengetahui cara penggunaan sistem dan deskripsi sistem. Implementasi halaman *Index* dapat dilihat pada Gambar 4.1. Halaman ini menampilkan penjelasan singkat mengenai tujuan sistem dan bagaimana cara menggunakannya. Pengguna dapat menekan *button* Mulai Sekarang untuk memulai sistem.



Gambar 4.1. Implementasi Antarmuka Halaman Depan

##### 4.1.4.2. Implementasi Antarmuka Upload Dokumen

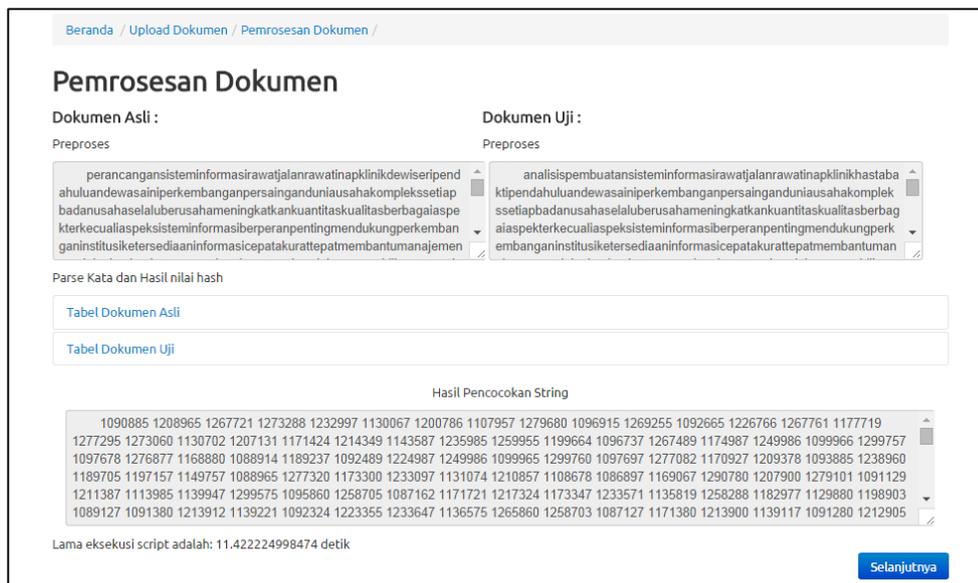
Implementasi antarmuka *upload* dokumen dapat dilihat pada Gambar 4.2. Pengguna akan meng*upload* dokumen yang akan di deteksi, yaitu dokumen asli dan dokumen uji. Dokumen yang akan di *upload* bertipe \*.docx.



Gambar 4.2. Implementasi Antarmuka *Upload* Dokumen

#### 4.1.4.3. Implementasi Antarmuka Pemrosesan Dokumen

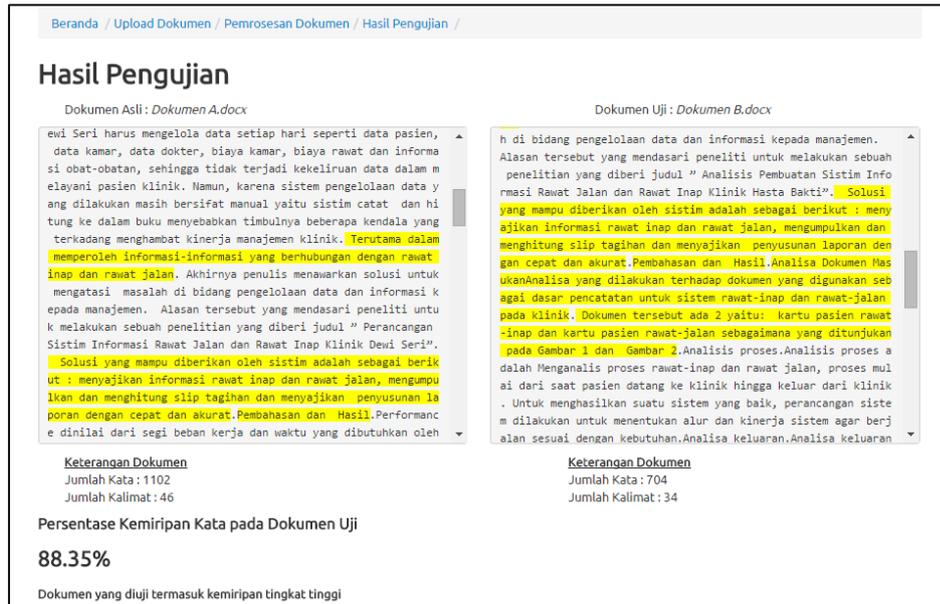
Implementasi antarmuka pemrosesan dokumen dapat dilihat pada Gambar 4.3. Pengguna akan disajikan tampilan halaman mengenai pemrosesan dokumen yang terdiri dari pemrosesan dokumen asli dan pemrosesan dokumen uji. Pada halaman ini juga ditampilkan *form* hasil dari proses *preprocessing*, *rolling hash*, dan pencocokan *string* dengan algoritma *Rabin-Karp*, serta lama eksekusi pemrosesan dokumen.



Gambar 4.3. Implementasi Antarmuka Pemrosesan Dokumen

#### 4.1.4.4. Implementasi Antarmuka Hasil Pengujian

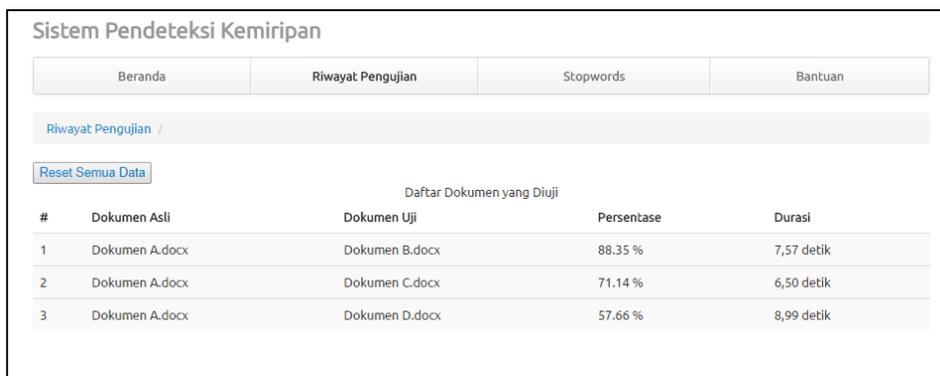
Implementasi antarmuka hasil pengujian yang dapat dilihat pada Gambar 4.4. Pada tampilan ini pengguna akan disajikan tampilan halaman yang memuat hasil pengujian deteksi plagiarisme yang berupa presentase dokumen yang dideteksi, serta menampilkan keberadaan kalimat yang terdeteksi dengan ditandai warna yang berbeda beserta keterangan jumlah kalimat dan kata pada dokumen tersebut.



Gambar 4.4. Implementasi Antarmuka Hasil Pengujian

#### 4.1.4.5. Implementasi Antarmuka Riwayat Pengujian

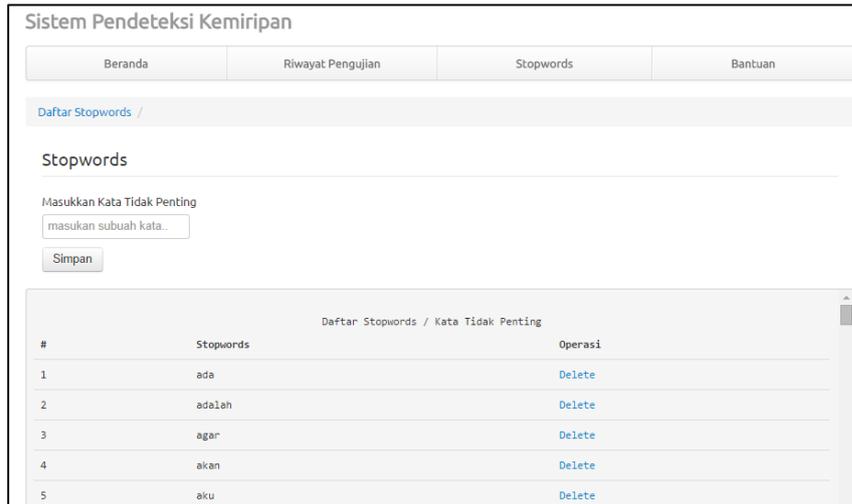
Implementasi antarmuka riwayat pengujian yang dapat dilihat pada Gambar 4.5. Pada tampilan ini pengguna disajikan tampilan halaman daftar pengujian dokumen yang terdiri dari dokumen asli, dokumen uji, persentase, dan durasi. Terdapat *button* Reset Semua Data untuk menghapus semua riwayat pengujian.



Gambar 4.5. Implementasi Antarmuka Riwayat Pengujian

#### 4.1.4.6. Implementasi Antarmuka Stopwords

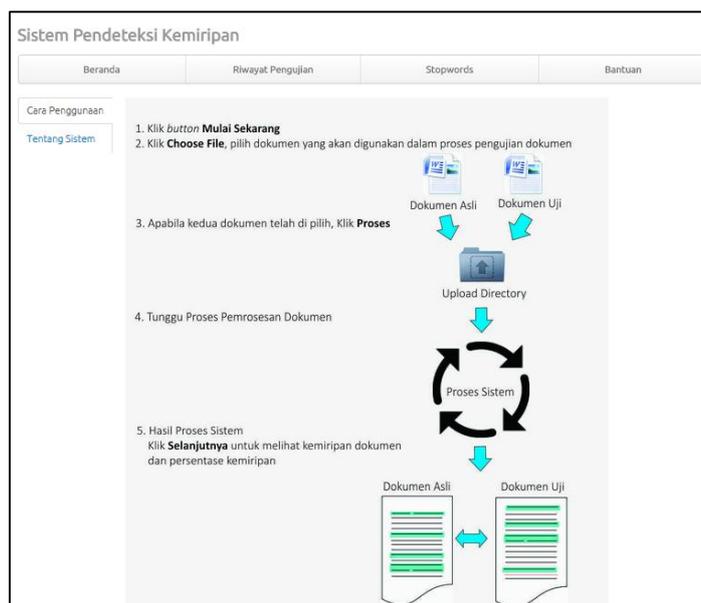
Implementasi antarmuka *stopwords* yang dapat dilihat pada Gambar 4.6. Pada tampilan ini pengguna akan disajikan tampilan halaman daftar *stopwords* dan form masukkan kata *stopwords*.



Gambar 4.6. Implementasi Antarmuka *Stopwords*

#### 4.1.4.7. Implementasi Antarmuka Bantuan

Seperti yang dijelaskan sebelumnya, Implementasi antarmuka bantuan berfungsi menjelaskan cara penggunaan sistem dan informasi tentang sistem bagi pengguna. Implementasi Antarmuka Bantuan dapat dilihat pada Gambar 4.7.



Gambar 4.7. Implementasi Antarmuka Bantuan

## 4.2. Pengujian

Pengujian sistem deteksi kemiripan kata dilakukan dengan menguji fungsi utama dari sistem, yaitu untuk melakukan pencocokan dua dokumen. Pengujian dikatakan berhasil jika sistem dapat mendeteksi keberadaan kalimat yang sama pada dokumen serta dapat menampilkannya dalam bentuk persentase.

### 4.2.1. Lingkungan Pengujian

Lingkungan meliputi perangkat keras dan perangkat lunak yang digunakan dalam proses pengujian Sistem Deteksi Kemiripan Kata adalah sebagai berikut :

1. CPU : Intel(R) Core(TM) i5 2410M @ 2,3 GHz, RAM 4,00 GB
2. Sistem Operasi : Windows 7 Professional
3. Alat bantu pemrograman : Notepad++ 5.9.3, Mozilla Firefox 30, Google Chrome 35.0, Xampp (Apache Web Server, MySQL, php MyAdmin 5.5.27)

### 4.2.2. Rencana Pengujian

Rencana pengujian Sistem Deteksi Kemiripan Kata merunut pada *Software Requirement Spesification* (SRS) yang telah dibuat dengan menggunakan butir-butir rencana pengujian dapat dilihat pada Tabel 4.4. Semua pengujian direncanakan diuji dengan metode *black box*.

Pengujian validitas sistem akan diujicobakan pada 10 dokumen skripsi yang memiliki keterkaitan judul yang berisi pendahuluan, pembahasan, dan penutup. Setiap dokumen yang memiliki topik judul yang sama akan dibandingkan. Memastikan sistem dapat mengidentifikasi kalimat sama secara sistem dan manual. Hasil tersebut akan memberikan nilai keakuratan sistem dengan menggunakan persamaan 2.3. Data Pengujian Validitas dapat dilihat pada Tabel 4.5.

Tabel 4.4. Rencana Pengujian Fungsional Sistem Deteksi Kemiripan Kata

No.	Fungsional	Butir Uji	Jenis Pengujian	Identifikasi Pengujian
1	SRS F-01	Memasukkan file yang tidak berekstensi *.docx	Black Box	U-1-01
		Dapat memasukkan dua file berekstensi *.docx		U-1-02
2	SRS F-02	Memastikan isi dokumen yang diproses filtering akan dihapus kata-kata tidak penting	Black Box	U-2-01
		Memastikan isi dokumen yang diproses <i>white insensitivity</i> akan melakukan penghilangan penomeran dan tanda baca		U-2-02
		Memastikan isi dokumen yang diproses <i>parsing</i> akan menghasilkan parse kata		U-2-03
3	SRS F-03	Memastikan setiap <i>parse</i> kata yang diproses akan menghasilkan nilai <i>hash</i>	Black Box	U-3-01
4	SRS F-04	Memastikan setiap <i>hash</i> yang dicocokkan merupakan nilai <i>hash</i> yang sama	Black Box	U-4-01
5	SRS F-05	Melakukan proses perhitungan kemiripan dokumen yang diuji	Black Box	U-5-01
6	SRS F-06	Menampilkan indikasi kalimat sama dengan memberi warna berbeda	Black Box	U-6-01
7	SRS F-07	Menampilkan hasil <i>running time</i>	Black Box	U-7-01

Tabel 4.5 Data Pengujian Validitas Sistem Deteksi Kemiripan Kata

No.	Nama Dokumen	Judul Dokumen	∑ Halaman	Ukuran (kilobyte)	Sumber
1	Dokumen A.docx	Perancangan Sistem Informasi Rawat Jalan dan Rawat Inap Klinik Dewi Seri	4 halaman	1.134	<a href="http://research.lppm-stmik.ibbi.ac.id/xxx.pdf">research.lppm-stmik.ibbi.ac.id/xxx.pdf</a>
2	Dokumen B.docx	Analisis Pembuatan Sistem Informasi Rawat Jalan dan Rawat Inap Klinik Hasta Bakti	3 halaman	29	<a href="http://research.lppm-stmik.ibbi.ac.id/xxx.pdf">research.lppm-stmik.ibbi.ac.id/xxx.pdf</a> (edit)
3	Dokumen C.docx	Analisis Kebutuhan Pengembangan Sistem Informasi Akuntansi di Rumah Sakit Universitas Hasanuddin	3 halaman	21	<a href="http://repository.unhas.ac.id/bitstream/handle/xxx">http://repository.unhas.ac.id/bitstream/handle/xxx</a>

No.	Nama Dokumen	Judul Dokumen	∑ Halaman	Ukuran (kilobyte)	Sumber
4	Dokumen D.docx	Perancangan Dan Pembuatan Sistem Informasi Rawat Inap Pasien Askes Pada Rumah Sakit Tgk. Fakinah Banda Aceh	10 halaman	3.567	<a href="http://180.241.122.205/docjurnal/FARIDA_HANDAYANI-09111xxx.pdf">http://180.241.122.205/docjurnal/FARIDA_HANDAYANI-09111xxx.pdf</a>
5	Dokumen E.docx	Pembangunan Sistem Informasi Data Balita Pada Posyandu Desa Ploso Kecamatan Punung Kabupaten Pacitan	8 halaman	101	<a href="http://download.portalgaruda.org/article.php?article=132291xxx">http://download.portalgaruda.org/article.php?article=132291xxx</a>
6	Dokumen F.doc	Perancangan Sistem Informasi Geografis Penyebaran DBD di Wilayah Kota Depok Dengan Menggunakan Arcview	10 Halaman	793	<a href="http://www.gunadarma.ac.id/library/articles/graduate/comp-uter-science/2009/Artikel_1010xxx.pdf">http://www.gunadarma.ac.id/library/articles/graduate/comp-uter-science/2009/Artikel_1010xxx.pdf</a>
7	Dokumen G.docx	Sistem Informasi Peringatan Dini Dan Peramalan Penderita Demam Berdarah Di Surakarta	10 halaman	700	<a href="http://eprints.uns.ac.id/5343/1/1351509082010xxx.pdf">http://eprints.uns.ac.id/5343/1/1351509082010xxx.pdf</a>
8	Dokumen H.docx	Analisis Spasial Penyebaran Penyakit Demam Berdarah <i>Dengue</i> Dengan Indeks Moran Dan Geary's C (Studi Kasus Di Kota Semarang Tahun 2011)	6 halaman	990	<a href="http://ejournal-s1.undip.ac.id/index.php/gaussian/article/viewFile/2745/2xxx">http://ejournal-s1.undip.ac.id/index.php/gaussian/article/viewFile/2745/2xxx</a>
9	Dokumen I.doc	Analisis Spasial Penyebaran Penyakit Demam Berdarah <i>Dengue</i> Dengan Indeks Moran Dan Geary's C (Studi Kasus Di Kota Semarang Tahun 2011) (Ringkasan)	3 Halaman	18	<a href="http://ejournal-s1.undip.ac.id/index.php/gaussian/article/viewFile/2745/2xxx">http://ejournal-s1.undip.ac.id/index.php/gaussian/article/viewFile/2745/2xxx</a>
10	Dokumen J.docx	Sistem Informasi Peringatan Dini Dan Peramalan Penderita Demam Berdarah Di Surakarta (Pendahuluan)	1 Halaman	13	<a href="http://eprints.uns.ac.id/5343/1/1351509082010xxx.pdf">http://eprints.uns.ac.id/5343/1/1351509082010xxx.pdf</a>

#### 4.2.3. Pelaksanaan Pengujian

Pelaksanaan pengujian dilakukan dengan cara mengimplementasikan rencana pengujian fungsional yang telah disusun. Bentuk pengujian yang dilakukan disesuaikan dengan rencana pengujian yang telah dijelaskan pada sub bab 4.2.2 pada Tabel 4.4. Tabel Hasil dan Evaluasi Pengujian Sistem Deteksi Kemiripan Kata dapat dilihat pada Tabel 4.6.

Setelah dilakukan pengujian fungsional, dilakukan pengujian validitas sistem. Data pengujian disesuaikan dengan rencana pengujian yang telah dijelaskan pada sub bab 4.2.2. Pengujian yang telah dilakukan oleh Sistem Deteksi Kemiripan Kata ini diperoleh dari hasil pencocokkan dua dokumen dengan perhitungan manual yang dapat dilihat pada Contoh 3.1 pada subbab 3.1.1. Hasil identifikasi manual dilakukan dengan cara menghitung kalimat sama antara dokumen satu dengan lainnya. Rumus perhitungan tingkat keakuratan sistem menggunakan persamaan 2.4 pada sub bab 2.13. Hasil pengujian validitas Sistem Deteksi Kemiripan Kata dapat dilihat pada Tabel 4.7.

Tabel 4.6. Hasil dan Evaluasi Pengujian Sistem Deteksi Kemiripan Kata

Identifikasi Pengujian	Deskripsi	Prosedur Pengujian	Hasil	Evaluasi	Kesimpulan
U-1-01	Memasukkan file yang tidak berekstensi *.docx	Memasukkan file yang tidak berekstensi *.docx	Muncul pesan peringatan bahwa user harus memasukkan file yang berekstensi *.docx	Pesan kesalahan sudah sesuai. Pesan kesalahan muncul ketika memasukkan <i>file</i> yang tidak sesuai	Diterima
U-1-02	Memasukkan dua file berekstensi *.docx	Memasukkan dua file yang berekstensi *.docx	<ol style="list-style-type: none"> <li>1. Sistem dapat melakukan proses pembacaan isi dokumen</li> <li>2. Semakin banyak halaman pada dokumen, waktu prosesnya semakin lama</li> </ol>	Proses <i>input</i> melalui unggah <i>file</i> sesuai dengan perencanaan	Diterima
U-2-01	Memastikan isi dokumen yang diproses <i>filtering</i> akan dihapus kata-kata tidak penting	Daftar kata-kata tidak penting yang terdaftar dalam database akan dipanggil saat proses <i>filtering</i>	Sistem dapat melakukan proses penghapusan kata-kata yang tidak penting.	Proses <i>filtering</i> telah sesuai dengan analisis dan perencanaan	Diterima
U-2-02	Memastikan isi dokumen yang diproses <i>white insensitivity</i> akan melakukan penghilangan penomeran dan tanda baca	Hasil pemrosesan <i>filtering</i> akan dilanjutkan dengan penghapusan tanda baca dan penomeran	Isi dokumen yang mengandung penomoran dan tanda baca melakukan proses penghilangan	Proses <i>white insensitivity</i> telah sesuai dengan analisis dan perencanaan	Diterima

Identifikasi Pengujian	Deskripsi	Prosedur Pengujian	Hasil	Evaluasi	Kesimpulan
U-2-03	Memastikan isi dokumen yang diproses <i>parsing</i> akan menghasilkan <i>parse</i> kata	Hasil pemrosesan <i>white insensitivity</i> akan dilanjutkan dengan melakukan <i>parsing/</i> pemecahan kata sebanyak 5 huruf	Sistem dapat melakukan pemecahan kata-kata sebanyak 5 huruf	Proses <i>parsing</i> telah sesuai dengan analisis dan perencanaan	Diterima
U-3-01	Memastikan setiap <i>parse</i> kata yang diproses akan menghasilkan nilai <i>hash</i>	Hasil dari pemrosesan teks akan diproses dengan <i>rolling hash</i> menghasilkan nilai <i>hash</i>	Sistem melakukan perhitungan <i>hashing</i> menggunakan <i>rolling hash</i> , memastikan agar proses <i>hashing</i> selanjutnya dapat menggunakan <i>rolling hash</i>	<ol style="list-style-type: none"> <li>1. Hasil perhitungan <i>hashing</i> sangat bergantung pada hasil pemrosesan teks</li> <li>2. Hasil perhitungan <i>rolling hash</i> sesuai dengan perhitungan secara manual</li> </ol>	Diterima
U-4-01	Memastikan setiap hash yang dicocokkan merupakan nilai <i>hash</i> yang sama	Melakukan proses pencocokan nilai <i>hash</i> menggunakan Algoritma <i>Rabin-Karp</i>	Sistem melakukan proses pencocokan hash menggunakan Algoritma <i>Rabin-Karp</i> dan menghasilkan nilai hash yang sama	Hasil pencocokan sudah sesuai dengan pencocokan secara manual	Diterima
U-5-01	Melakukan proses perhitungan kemiripan dokumen yang diuji	Proses perhitungan kemiripan diambil dari pencocokan dokumen asli dan dokumen yang diuji	Perhitungan kemiripan menggunakan rumus dengan perbandingan dokumen yang diuji	Hasil perhitungan sudah sesuai dengan perhitungan secara manual	Diterima
U-6-01	Menampilkan indikasi kalimat sama dengan memberi warna berbeda	Memastikan kemiripan kalimat sama antara dokumen uji dan dokumen asli sesuai	Menampilkan kemiripan kalimat antara dokumen uji dan dokumen asli dengan memberi <i>highlight</i> berwarna kuning	Hasil sesuai dengan indikasi secara manual	Diterima
U-7-01	Menampilkan hasil <i>running time</i>	Memastikan proses sistem dapat diketahui	Menampilkan lama proses sistem berupa satuan detik	Menampilkan waktu proses sistem	Diterima

Tabel 4.7. Hasil Pengujian Validitas Sistem Deteksi Kemiripan Kata

No	Dokumen Asli		Dokumen Uji		Persentase Kemiripan Kata	Durasi (detik)	$\Sigma$ Kalimat Sama		Tingkat Keakuratan Sistem
	Nama Dokumen	$\Sigma$ kalimat	Nama Dokumen	$\Sigma$ kalimat			Identifikasi Sistem	Identifikasi Manual	
1	Dokumen A.docx	47	Dokumen B.docx	35	88,35 %	6,68	18 kalimat	17 kalimat	94,1 %
2	Dokumen A.docx	47	Dokumen C.docx	31	69,84%	8,75	12 kalimat	12 kalimat	100 %
3	Dokumen A.docx	47	Dokumen D. docx	97	55,84 %	13,11	3 kalimat	3 kalimat	100 %
4	Dokumen A.docx	47	Dokumen E. docx	79	57,34 %	14,04	6 kalimat	6 kalimat	100 %
5	Dokumen B.docx	35	Dokumen C.docx	31	46,83 %	7,13	2 kalimat	2 kalimat	100 %
6	Dokumen B.docx	35	Dokumen D.docx	97	51,16 %	10,9	3 kalimat	4 kalimat	75 %
7	Dokumen B.docx	35	Dokumen E.docx	79	43,99 %	7,76	3 kalimat	3 kalimat	100 %
8	Dokumen C.docx	31	Dokumen D.docx	97	45,37 %	8,53	7 kalimat	7 kalimat	100 %
9	Dokumen C.docx	31	Dokumen E.docx	79	55,23 %	10,45	8 kalimat	8 kalimat	100 %
10	Dokumen D.docx	97	Dokumen E.docx	79	51,57 %	15,31	6 kalimat	6 kalimat	100 %
11	Dokumen F.doc	155	Dokumen G.docx	84	62,04 %	35,77	5 kalimat	5 kalimat	100 %
12	Dokumen F.doc	155	Dokumen H.docx	86	51,87 %	35,04	11 kalimat	11 kalimat	100 %
13	Dokumen F.doc	155	Dokumen I.doc	55	56,24 %	17,51	7 kalimat	7 kalimat	100 %
14	Dokumen F.doc	155	Dokumen J.docx	16	72,84 %	7,22	5 kalimat	3 kalimat	33 %
15	Dokumen G.docx	84	Dokumen H.docx	86	39,68 %	16,48	1 kalimat	1 kalimat	100 %
16	Dokumen G.docx	84	Dokumen I.doc	55	42,38 %	11,02	1 kalimat	1 kalimat	100 %
17	Dokumen G.docx	84	Dokumen J.docx	16	100 %	6,46	16 kalimat	16 kalimat	100 %
18	Dokumen H.docx	86	Dokumen I.doc	55	99,82 %	16,23	54 kalimat	53 kalimat	98 %
19	Dokumen H.docx	86	Dokumen J.docx	16	57,26 %	4,97	2 kalimat	2 kalimat	100 %
20	Dokumen I.docx	55	Dokumen J.docx	16	45,68 %	3,73	2 kalimat	2 kalimat	100%

#### **4.2.4. Analisis Hasil Pengujian**

Berdasarkan rencana fungsional yang disajikan pada Tabel 4.5 serta hasil pengujian yang telah dilakukan berdasarkan Lampiran 3, dapat dilihat bahwa semua pengujian telah diterima. Dengan demikian, Sistem Deteksi Kemiripan Kata telah memenuhi semua spesifikasi kebutuhan fungsional yang telah didefinisikan sebelumnya.

Pada rencana pengujian validasi terdapat 10 dokumen yang diuji, dapat dilihat pada Tabel 4.6. Hasil pengujian validasi telah dilakukan berdasarkan Tabel 4.7. Pada Tabel 4.7 dapat dilihat bahwa sistem deteksi kemiripan kata menghasilkan persentase kemiripan kata dari 20 dokumen pengujian diperoleh 13 dokumen termasuk kemiripan tingkat tinggi, 1 dokumen memiliki kemiripan sama, dan 6 dokumen termasuk kemiripan tingkat sedang. Rata-rata waktu proses eksekusi sistem 11,6 detik dari 20 dokumen. Dilihat dari tingkat lama eksekusi sistem, dokumen yang memiliki halaman dan konten teks lebih banyak akan semakin lama proses eksekusinya.

Hasil identifikasi jumlah kalimat sama dengan cara manual memiliki sedikit perbedaan dengan identifikasi jumlah kalimat sama pada Sistem Deteksi Kemiripan Kata. Selain dapat dilihat dari jumlah kalimat pada sistem dan manual perbedaan juga dapat dilihat dari tingkat keakuratan sistem. Berdasarkan hasil tingkat keakuratan sistem persentase kesesuaian sebesar 95% yang diperoleh dari rata-rata nilai keakuratan sistem. Pada pengujian kalimat sama dari 20 dokumen, 4 diantaranya berbeda/ tidaksesuai. Hal ini dikarenakan pada dokumen berisi nomer, sub bab, tabel dan gelar singkatan, sehingga berpengaruh pada pencocokkan kalimat sama. Dapat dicontohkan misalnya kata yang seharusnya bukan kalimat justru teridentifikasi, sebaliknya kalimat yang seharusnya kalimat justru tidak teridentifikasi itu disebabkan karena sebelum kalimat ada subbab.

## **BAB V**

### **PENUTUP**

Penutup berisi kesimpulan dari pengerjaan penelitian tugas akhir ini dan saran-saran penulis untuk pengembangan lebih lanjut dari penelitian serupa.

#### **5.1. Kesimpulan**

Kesimpulan yang dapat diambil dari pembuatan tugas akhir ini adalah :

1. Pembuatan sistem deteksi kemiripan kata pada dua dokumen menggunakan Algoritma Rabin-Karp berhasil dibangun dengan berbasis *web*. Sistem ini dapat menghasilkan persentase kemiripan kata dan mengindikasikan kalimat sama pada dua dokumen.
2. Hasil perhitungan tingkat keakuratan sistem memiliki nilai keakuratan 95% diperoleh dari 20 dokumen pengujian identifikasi kalimat sama .
3. Rata-rata waktu proses eksekusi sistem sebesar 11,6 detik dari 20 dokumen pengujian sistem deteksi kemiripan kata. Semakin banyak kata yang diproses, maka akan semakin lama proses eksekusi.

#### **5.2. Saran**

Penelitian ini masih dapat dikembangkan lebih lanjut. Adapun saran untuk mengembangkan penelitian ini adalah sebagai berikut :

1. Sistem dapat dikembangkan menjadi sistem yang dapat mendeteksi banyak dokumen.
2. Perlu adanya pengembangan metode untuk memecah kalimat yang dapat mendeteksi adanya gelar/ singkatan dan struktur dokumen yang kompleks seperti nomer, tabel, bab, sub bab, dan lain lain.
3. Sistem dapat mengekstrak dokumen dengan ekstensi lain seperti Portable Document Format(\*.pdf), dan Open Office Writer(\*.odt).

## DAFTAR PUSTAKA

- Anonim, *Pattern Search*. The George Washington University. [Online] Available at: <http://www.seas.gwu.edu/~simhaweb/alg/lectures/module5/module5.html> [Accessed 11 Juli 2013].
- Anonim, 2008. *Diklat Pemrograman PHP dan MYSQL*. Jakarta: STMIK Muhammadiyah. [Online] Available at : [http://e-nixsoft.com/johari/file\\_kuliah/Pemrograman5\\_PHP.pdf](http://e-nixsoft.com/johari/file_kuliah/Pemrograman5_PHP.pdf) [Accessed 17 Oktober 2013].
- Adam, D.A. & Baedlowi, N., 2009. *String Matching dengan Menggunakan Algoritma Rabin-Karp*. Bandung: Institute Technology Bandung.
- Astuti, B., et al., 2012. *Identifikasi Perilaku Plagiat pada Mahasiswa Fakultas Ilmu Pendidikan. Universitas Negeri Yogyakarta*. Yogyakarta: Universitas Negeri Yogyakarta.
- Atmopawiro, A., 2007. *Pengkajian Dan Analisis Tiga Algoritma Efisien Rabin-Karp, Knuth-Morris-Pratt, Dan Boyer-Moore Dalam Pencarian Pola Dalam Suatu Teks*. Bandung: Institut Teknologi Bandung.
- Cormen, T.H., et al., 2009. *Introduction to Algorithms*. MIT Press: USA. pp. 985-995
- Djafar, F. B. 2014. *Penerapan Algoritma Smith-Waterman Dalam Sistem Pendeteksi Kesamaan Dokumen*. Gorontalo: Universitas Negeri Gorontalo.
- Gipp, B. & Meuschke, N., 2011. *Citation Pattern Matching Algorithms for Citationbased Plagiarism Detection: Greedy Citation Tiling, Citation Chunking and Longest Common Citation Sequence. Proceeding of the 11th ACM Symposium on Document Engineering*. USA:. Mountain View CA.
- Mooney, R.J., 2006 *Mechine Learning Text Categorization*. Austin. University of Texas
- Mulyana, 2010. *Pencegahan Tindak Plagiarisme Dalam Penulisan Skripsi Upaya Memperkuat Pembentukan Karakter Di Dunia Akademik*. Yogyakarta: FBS Universitas Negeri Yogyakarta.
- Mutiara, B. & Agustina, S., 2008. *Anti Plagiarsm Application with Algorithm Karp-Rabin at Thesis in Gunadarma University*. Depok: Gunadarma University.
- Muzgovoy, M. 2007. *Enhancing Computer-Aided Plagiarism Detection*. Finland: University of Joensuu.
- Nugroho, E., 2011. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp*. Malang: Universitas Brawijaya.

- Pressman, R. S., 2001, *Software Engineering : A Practitioner's Approach Fifth Edition*, New York, McGraw – Hill. pp. 302-317
- Purwitasari, D., Kusmawan, P.Y. & Yuhana, U.L., 2010. *Deteksi Keberadaan Kalimat Sama sebagai Indikasi Penjiplakan dengan Algoritma Hashing Berbasis N-Gram*. Surabaya: Institut Teknologi Sepuluh Nopember.
- Riyanti, Y., 2009. *Penerapan Algoritma Rabin-Karp dalam Mencari Persentase Kemiripan (Similarity) Isi Dua Dokumen Teks Berbahasa Indonesia*. Yogyakarta: Universitas Gajah Mada.
- Salmuasih & Sunyoto, A., 2013. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Konsep Similarity Menggunakan Algoritma Rabin-Karp*. Yogyakarta: STMIK AMIKOM.
- Silberschatz, K., & Sudarshan. 2001. *Database System Concepts, Fourth Edition*. The McGraw-Hill Companies
- Sommerville, I., 2007. *Software Engineering 9th edition*. Addison Wesley: Publisher. pp. 29-32
- Surahman, A.M., 2013. *Perancangan Sistem Penentuan Similarity Kode Program Pada Bahasa C Dan Pascal Dengan Menggunakan Algoritma Rabin-Karp*. Pontianak: Universitas Tanjungpura.
- Tsuneo, F., et al., 2011. *Pengukuran dan Kesalahan*. [Online] Available at : <http://www.scribd.com/doc/60303652/Bab-1-Pengukuran-Dan-Kesalahan> [Accessed 20 Agustus 2014].

## **LAMPIRAN - LAMPIRAN**

## Lampiran 1. Tabel Daftar *Stopwords*

No	Stopwords	No	Stopwords	No	Stopwords	No	Stopwords
1	yang	34	sebagai	67	secara	100	tentang
2	di	35	bahwa	68	dilakukan	101	bukan
3	dan	36	dapat	69	sementara	102	agar
4	itu	37	para	70	tapi	103	semua
5	dengan	38	harus	71	sangat	104	sedang
6	untuk	39	namun	72	hal	105	kali
7	tidak	40	kita	73	sehingga	106	kemudian
8	ini	41	dua	74	seorang	107	hasil
9	dari	42	satu	75	bagi	108	sejumlah
10	dalam	43	masih	76	besar	109	juta
11	akan	44	hari	77	lagi	110	sendiri
12	pada	45	hanya	78	selama	111	katanya
13	juga	46	mengatakan	79	antara	112	demikian
14	saya	47	kepada	80	waktu	113	masalah
15	ke	48	kami	81	sebuah	114	mungkin
16	karena	49	setelah	82	jika	115	umum
17	tersebut	50	melakukan	83	sampai	116	setiap
18	bisa	51	lalu	84	jadi	117	bulan
19	ada	52	belum	85	terhadap	118	bagian
20	mereka	53	lain	86	serta	119	bila
21	lebih	54	dia	87	pun	120	lainnya
22	kata	55	kalau	88	atas	121	terus
23	tahun	56	terjadi	89	sejak	122	luar
24	sudah	57	banyak	90	membuat	123	cukup
25	atau	58	menurut	91	baik	124	termasuk
26	saat	59	anda	92	memiliki	125	sebelumnya
27	oleh	60	hingga	93	kembali	126	bahkan
28	menjadi	61	tak	94	selain	127	perlu
29	orang	62	baru	95	tetapi	128	sedangkan
30	ia	63	beberapa	96	pertama	129	aku
31	telah	64	ketika	97	memang	130	begitu
32	adalah	65	saja	98	pernah	131	maka
33	seperti	66	sekitar	99	apa		