AGGLOMERATIVE HIERARCHICAL CLUSTERING UNTUK PENGGABUNGAN DOKUMEN TEKS BERBAHASA INDONESIA DENGAN METODE *COMPLETE LINKAGE*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang Ilmu Komputer

Oleh**:**

BERNADETTE SHANTI C.
0310960017

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2009

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

AGGLOMERATIVE HIERARCHICAL CLUSTERING* UNTUK PENGGABUNGAN DOKUMEN TEKS BERBAHASA INDONESIA DENGAN METODE *COMPLETE LINKAGE

Oleh :
BERNADETTE SHANTI C.
0310960017

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 14 Agustus 2009
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Edy Santoso, SSi., M.Kom.	**Dian Eka Ratnawati, S.Si., M.Kom.**
NIP. 132 304 307	**NIP. 132 300 224**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc.
NIP. 131 126 049

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Bernadette Shanti C.**
NIM	**: 0310960017**
Jurusan	**: Matematika**
Penulis Skripsi berjudul	**:** ***Agglomerative Hierarchical Clustering*** **untuk Penggabungan Dokumen Teks Berbahasa Indonesia dengan Metode** ***Complete Linkage.***

Dengan ini menyatakan bahwa :

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Kode program yang termaktub dalam Skripsi ini dibuat oleh Aldo Septyan Praherda (NIM. 0310960005)**
3. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, Agustus 2009
Yang menyatakan,

(Bernadette Shanti C.)
NIM. 0310960017

UNIVERSITAS BRAWIJAYA

AGGLOMERATIVE HIERARCHICAL CLUSTERING UNTUK PENGGABUNGAN DOKUMEN TEKS BERBAHASA INDONESIA DENGAN METODE *COMPLETE LINKAGE*

ABSTRAK

Sistem penggabungan dokumen merupakan sistem yang memberi kemudahan kepada pengguna untuk mendapatkan informasi dari beberapa dokumen sekaligus. Dengan sistem ini, pengguna tidak perlu lagi membaca masing-masing dokumen satu persatu untuk mendapatkan informasi yang dibutuhkan. Dengan demikian, informasi yang didapat akan lebih mudah dicerna dan dimengerti.

Permasalahan yang timbul pada saat menggabungkan dokumen adalah bagaimana menggabungkan kalimat-kalimat dari dokumen yang berbeda supaya tetap memiliki keterkaitan satu sama lain. Penggabungan kalimat dilakukan dengan cara mengelompokkan kalimat-kalimat menjadi suatu *cluster* dengan algoritma *hierarchical clustering* melalui pendekatan *agglomerative*, dimana anggota *cluster* awal adalah masing-masing kalimat yang ada, dan kemudian bergabung menjadi *cluster* yang lebih besar dengan kalimat yang lain. Aturan penggabungan kalimat ditentukan berdasarkan bobot yang dimiliki masing-masing kalimat.

Untuk mengevaluasi sistem penggabungan dokumen ini dilakukan pembandingan dengan dokumen ideal. Ukuran yang digunakan adalah rasio dari jumlah kalimat yang bersesuaian dengan kalimat ideal yang dibandingkan dengan jumlah total kalimat yang ada. Efektifitas sistem meningkat sejalan dengan bertambahnya jumlah dokumen yang digabungkan.

UNIVERSITAS BRAWIJAYA

AGGLOMERATIVE HIERARCHICAL CLUSTERING FOR INDONESIAN TEXT DOCUMENT MERGING BY USING *COMPLETE LINKAGE* METHOD

ABSTRACT

Document merging system is a system which simplifies user in getting information from several documents at once. By this system, user does not have to read each document one by one to get information they need anymore. Therefore, the information will be easier to absorb and understand.

The problem which occurred when we merge documents is how to merge with paying attention to the connections among sentences. A sentence clustering is done by hierarchical clustering algorithm with agglomerative approach where initial cluster members are each sentence itself, and then they integrate with other sentence become larger cluster. The rule of integrating is defined by sentence weight.

A comparison with an ideal document is used to evaluate this merging document system. The measure that used is the ratio of total equivalent sentences and total numbers of the sentences. The effectiveness of system increases along with the increasing of the number of source document that merged.

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji Syukur kepada Tuhan karena atas berkat dan rahmat-Nya, akhirnya Skripsi sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer ini dapat terselesaikan.

Skripsi ini mengambil tema pemrosesan teks untuk dapat diterapkan sebagai solusi pengintegrasian dokumen sebagai pengembangan dari penelitian sebelumnya (Budhi, 2008).

Dalam kesempatan ini, penulis menyampaikan ucapan terima kasih kepada:

1. Edy Santoso, SSi., M.Kom., selaku dosen pembimbing utama.
2. Dian Eka Ratnawati, S.Si., M.Kom., selaku pembimbing kedua selama penulisan skripsi.
3. Wayan Firdaus Mahmudy, S.Si., MT., selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya, sekaligus penasehat akademik.
4. Keluarga, yang mendukung baik secara material maupun spiritual.
5. Segenap bapak dan ibu dosen, yang telah mendidik dan mengajarkan ilmunya selama penulis menempuh pendidikan Program Studi Ilmu Komputer di Universitas Brawijaya.
6. Segenap staf dan karyawan Jurusan Matematika Universitas Brawijaya, yang telah membantu kelancaran administrasi selama persiapan dan pengerjaan skripsi.
7. Aldo Septyan Praherda, yang telah membantu dalam pembuatan kode program.
8. Teman-teman yang selalu mendoakan dan semua pihak yang telah membantu yang tidak dapat disebutkan satu-persatu.

Menyadari akan segala keterbatasan dan adanya kekurangan dalam skripsi ini, maka diharapkan saran dan kritik yang sifatnya membangun demi perbaikan di masa yang akan datang.

Malang, Januari 2009

Penulis

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

Halaman
LEMBAR PENGESAHAN SKRIPSI iii
LEMBAR PERNYATAAN v
ABSTRAK vii
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
DAFTAR LAMPIRAN xix
BAB I 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi Pemecahan Masalah 3
1.7 Sistematika Penulisan 4
BAB II 5
2.1 Clustering 5
2.2 Hierarchical Clustering 6
2.2.1 Divisive 7
2.2.2 Agglomerative 7
2.3 Metode Complete-Linkage 8
2.4 Struktur Morfologi Bahasa Indonesia 11
2.5 Text Pre-processing 12
2.5.1 Case folding 12
2.5.2 Parsing 12
2.5.3 Penghilangan *Stopword* 13
2.5.4 Stemming 13
2.6 Pembobotan 14
2.7 Evaluasi 16
BAB III 19
3.1 Metodologi 19
3.2 Analisis Sistem 20
3.2.1 Deskripsi Umum Sistem 20
3.2.2 Batasan Sistem 21
3.3 Perancangan Proses 21

3.3.1 Text Pre-processing 23
3.3.2 Pembobotan 28
3.3.3 Hierarchical Clustering 32
3.4 Perancangan Tabel 37
3.5 Perancangan Antarmuka 38
3.6 Metode Pengujian 41
BAB IV 43
4.1 Lingkungan Implementasi 43
4.1.1 Lingkungan perangkat keras 43
4.1.2 Lingkungan perangkat lunak 43
4.2 Implementasi Program 43
4.2.1 Implementasi text pre-processing 43
a. Implementasi parsing 44
b. Implementasi *stemming* 45
4.2.2 Implementasi pembobotan 46
a. Penghitungan bobot I 46
b. Penghitungan Bobot II 47
c. Penghitungan Bobot III 48
d. Penghitungan Bobot IV 49
e. Penghitungan Bobot V 50
f. Penghitungan Bobot Relasi 51
4.2.3 Implementasi Hierarchical Clustering 54
4.3 Implementasi Tabel 55
4.4 Implementasi Antarmuka 56
4.5 Uji Coba 58
4.5.1 Skenario Evaluasi 58
4.5.2 Hasil Evaluasi 59
4.5.3 Analisis Evaluasi 60
BAB V 63
LAMPIRAN 67

DAFTAR GAMBAR

Halaman

Gambar 2.1 *Clustering* 5
Gambar 2.2 Diagram teknik *clustering* 6
Gambar 2.3 Dendogram 6
Gambar 2.4 Langkah ke-0 8
Gambar 2.5 Langkah ke-1 9
Gambar 2.6 Langkah ke-2 9
Gambar 2.7 Langkah ke-3 10
Gambar 2.8 Langkah ke-4 10
Gambar 2.9 Pohon hirarki *clustering* beberapa kota di Italia.....10
Gambar 2.10 Ilustrasi dari *recall* dan *precision* 17
Gambar 3.1 Tahapan penelitian.....19
Gambar 3.2 Diagram alir proses penggabungan dokumen.....20
Gambar 3.3 Contoh dokumen1 23
Gambar 3.4 Contoh dokumen2 23
Gambar 3.5 Dokumen1 dalam bentuk kalimat ber-*index* 24
Gambar 3.6 Dokumen2 dalam bentuk kalimat ber-*index* 24
Gambar 3.7 Dokumen1 setelah *text pre-processing* 26
Gambar 3.8 Dokumen2 setelah *text pre-processing*.....26
Gambar 3.9 Diagram alir proses *stemming* 27
Gambar 3.10 Graf relasi kalimat dokumen1 30
Gambar 3.11 Graf relasi kalimat dokumen2 31
Gambar 3.12 Pohon hirarki L_0.....33
Gambar 3.13 Pohon hirarki L_1.....33
Gambar 3.14 Pohon hirarki L_2.....34
Gambar 3.15 Pohon hirarki L_3.....34
Gambar 3.16 Pohon hirarki L_4.....35
Gambar 3.17 Pohon hirarki L_5.....35
Gambar 3.18 Pohon hirarki L_6.....35
Gambar 3.19 Pohon hirarki L_7.....36
Gambar 3.20 Pohon hirarki L_8.....36
Gambar 3.21 Hasil penggabungan dokumen.....37
Gambar 3.22 Rancangan antarmuka tahapan proses 39
Gambar 3.23 Rancangan antarmuka tahapan analisis 40
Gambar 4.1 *Source code* fungsi pencarian delimiter.....43
Gambar 4.2 *Source code* fungsi *parsing* 44

Gambar 4.3 *Source code* fungsi penghilangan *stopword* 45
Gambar 4.4 *Source code* implementasi *stemming* 45
Gambar 4.5 *Source code* penghitungan bobot kalimat W1 46
Gambar 4.6 *Source code* fungsi `GetTotalKata()` 46
Gambar 4.7 *Source code* penghitungan bobot kalimat W2 47
Gambar 4.8 *Source code* fungsi `isKalimatUtama()` 48
Gambar 4.9 *Source code* penghitungan bobot kalimat W3 48
Gambar 4.10 *Source code* penghitungan bobot kalimat W4 49
Gambar 4.11 *Source code* penghitungan bobot kalimat W5 50
Gambar 4.12 *Source code* pendeklarasian kelas `BobotRelasi`.....50
Gambar 4.13 *Source code* fungsi `GetBobotRelasi()` 51
Gambar 4.14 *Source code* fungsi penghitungan bobot relasi 52
Gambar 4.15 Implementasi deklarasi kelas `Node` 53
Gambar 4.16 Implementasi metode *complete linkage*.....54
Gambar 4.17 Implementasi antarmuka tahap proses.....56
Gambar 4.18 Implementasi antarmuka tahap analisis 58
Gambar 4.19 Grafik evaluasi recall, precision, dan f-measure.....61

DAFTAR TABEL

Halaman

Tabel 3.1 Frekuensi kata pada dokumen1 28
Tabel 3.2 Frekuensi kata pada dokumen2 29
Tabel 3.3 Bobot kalimat 31
Tabel 3.4 Bobot kalimat setelah dinormalisasi.....31
Tabel 3.5 Bobot relasi.....32
Tabel 3.6 Langkah ke-0 32
Tabel 3.7 Langkah ke-1 33
Tabel 3.8 Langkah ke-2 34
Tabel 3.9 Langkah ke-3 34
Tabel 3.10 Langkah ke-4 34
Tabel 3.11 Langkah ke-5 35
Tabel 3.12 Langkah ke-6 35
Tabel 3.13 Langkah ke-7 35
Tabel 3.14 Langkah ke-8 36
Tabel 3.15 Rancangan tabel frekuensi kata 37
Tabel 3.16 Rancangan tabel posisi kata.....38
Tabel 3.17 Rancangan tabel kalimat utama.....38
Tabel 4.1 Implementasi tabel GlobalKata 55
Tabel 4.2 Implementasi tabel KalimatAsli 55
Tabel 4.3 Implementasi tabel KalimatUtama 56
Tabel 4.4 Implementasi tabel Kata 56
Tabel 4.5 Implementasi tabel Teks.....56
Tabel 4.6 Hasil evaluasi pada 1 dokumen hasil.....59
Tabel 4.7 Hasil evaluasi dengan 1 dokumen sumber.....59
Tabel 4.8 Hasil evaluasi dengan 2 dokumen sumber.....60
Tabel 4.9 Hasil evaluasi dengan 3 dokumen sumber.....60
Tabel 4.10 Hasil statistik deskriptif sistem.....60

UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

Halaman

Lampiran 1 Daftar stopword 65
Lampiran 2 Aturan derivasional untuk prefiks 71
Lampiran 3 State diagram *stemming* 74

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi dan modernisasi, kebutuhan akan informasi pun terus meningkat. Banyak cara yang dapat dilakukan untuk memperoleh informasi yang dibutuhkan, salah satunya melalui media elektronik, dalam hal ini komputer dalam bentuk dokumen elektronik. Metode pencarian yang banyak dikembangkan dewasa ini memudahkan didapatkannya segala informasi sesuai topik yang dibutuhkan. Dengan demikian, maka akan semakin mudah untuk mendapatkan informasi selengkap-lengkapnya.

Informasi yang lengkap dapat diperoleh dengan cara menggali informasi dari berbagai sumber. Dengan membaca beberapa referensi dokumen sesuai topik yang dibutuhkan akan membuat informasi yang didapat menjadi lebih lengkap dan seimbang. Hal ini dikarenakan beberapa dokumen akan menyajikan dengan cara yang berbeda walaupun membahas topik yang sama. Dengan demikian, memungkinkan suatu dokumen memiliki informasi tambahan yang tidak ada pada dokumen lainnya sehingga masing-masing dokumen dapat saling melengkapi satu sama lain.

Di sisi lain, membaca beberapa dokumen memiliki kesulitan tersendiri. Dengan topik yang sama, sangat memungkinkan adanya kalimat-kalimat yang mirip dan berulang-ulang di setiap dokumen. Hal ini dapat mengakibatkan pembaca kesulitan menangkap pokok bahasan dan informasi-informasi penunjang yang penting di dalam setiap dokumen karena harus mengingat – ingat isi dokumen yang telah dibaca sebelumnya dan menggabungkannya di dalam pikiran sebelum dapat merangkum maksud dan topik utama dokumen – dokumen tersebut secara keseluruhan. Dengan adanya permasalahan tersebut, maka akan lebih memudahkan bila dokumen-dokumen yang membahas topik yang sama tersebut digabungkan terlebih dahulu baru kemudian dibaca.

Dalam penelitian ini akan dicoba untuk membuat aplikasi untuk memenuhi kebutuhan di atas. Dengan menggunakan algoritma *clustering*, aplikasi penggabungan dokumen ini akan menghasilkan satu dokumen baru hasil penggabungan beberapa dokumen yang memiliki topik yang sama tersebut secara otomatis. Dokumen baru

tersebut mencakup bagian dari masing-masing dokumen awal, namun memiliki susunan kalimat dan paragraf yang berbeda dari dokumen awal. Perbedaan ini dikarenakan kalimat-kalimat yang ada disusun ulang sesuai dengan besarnya kesamaan (*similarity*) antarkalimatnya pada proses penggabungan dokumen tersebut. Dengan demikian, diharapkan pembaca lebih mudah untuk menyerap informasi penting dan mendapat informasi yang lengkap tanpa harus membaca beberapa dokumen satu per satu.

Pada penelitian sebelumnya (Budhi, 2008), digunakan algoritma *hierarchical clustering* dengan pendekatan *agglomerative* karena *clustering* yang diterapkan pada penelitian ini tidak bersifat memecah sekumpulan data menjadi kelompok-kelompok, namun sebaliknya menggabungkan data-data individu (*atomic cluster*) menjadi kelompok-kelompok data. Sedangkan metode yang digunakan yaitu *single-linkage*, yang menggunakan jarak terpendek sebagai jarak antar-*cluster*-nya.

Penelitian kali ini menggunakan algoritma yang sama dengan penelitian sebelumnya, namun menggunakan metode yang berbeda, yakni *complete linkage*. Metode ini menggunakan jarak terjauh sebagai jarak antar-*cluster*-nya. Dipilihnya metode ini karena sangat berguna untuk memperkecil varian dalam cluster dan cocok untuk data yang mengandung *outlier* (Barakbah, 2006). Diharapkan dengan metode ini hasil penghitungan *cluster* , dalam hal ini pembentukan kalimat dalam paragraf, dapat lebih optimal.

Berdasarkan pada latar belakang yang telah dipaparkan, maka judul pada skripsi ini adalah "***AGGLOMERATIVE HIERARCHICAL CLUSTERING* UNTUK PENGGABUNGAN DOKUMEN TEKS BERBAHASA INDONESIA DENGAN METODE *COMPLETE LINKAGE*"**.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam skripsi ini dapat dirumuskan permasalahan, yaitu:

1. Bagaimana menggabungkan beberapa dokumen yang memiliki kesamaan topik.
2. Bagaimana tingkat akurasi yang dihasilkan oleh sistem.

1.3 Batasan Masalah

Dari permasalahan yang telah dipaparkan dalam subbab 1.2, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Dokumen yang digabungkan hanya dokumen teks yang memiliki format *.txt
2. Pemilihan dokumen yang akan digabungkan ditentukan oleh *user*.
3. Urutan paragraf tidak dipertimbangkan benar salahnya.
4. Tidak dilakukan penghitungan tingkat kesamaan antar dokumen.
5. Tidak dilakukan perbandingan dengan metode lain.

1.4 Tujuan

Tujuan yang ingin dicapai dalam pelaksanaan skripsi ini adalah:

1. Mengetahui kinerja algoritma *agglomerative hierarchical clustering* dan menghasilkan sebuah perangkat lunak yang mengimplementasikan algoritma *agglomerative hierarchical clustering* untuk penggabungan dokumen yang memiliki kesamaan topik.
2. Mengetahui tingkat akurasi yang dihasilkan oleh sistem.

1.5 Manfaat

Manfaat yang dapat diambil dari penulisan skripsi ini adalah :

1. Memahami penerapan algoritma *agglomerative hierarchical clustering* untuk menggabungkan beberapa dokumen yang memiliki kesamaan topik.
2. Membantu proses perolehan informasi yang lengkap dan seimbang dari berbagai sumber tanpa harus membaca beberapa dokumen satu per satu.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan Skripsi ini adalah:

1. Studi literatur
   Mempelajari teori-teori yang berhubungan dengan konsep *agglomerative hierarchical clustering* dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan algoritma *agglomerative hierarchical clustering.*
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

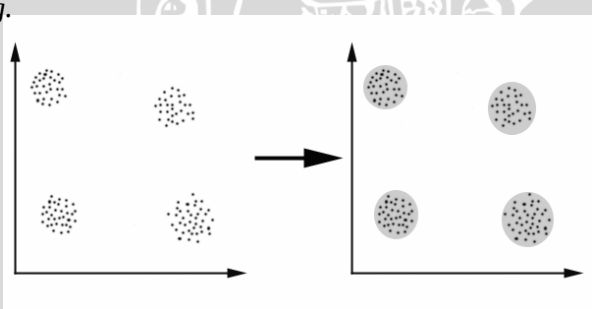
1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan *agglomerative hierarchical clustering* dan pemrosesan teks.
3. BAB III METODOLOGI DAN PERANCANGAN SISTEM
   Pada bab ini akan dijelaskan mengenai langkah-langkah yang digunakan dalam pemrosesan dokumen teks dengan teknik *clustering* data menggunakan algoritma *agglomerative hierarchical clustering.*
4. BAB IV IMPLEMENTASI DAN UJI COBA SISTEM
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa sistem perangkat lunak yang dibangun.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1 Clustering

Clustering dapat dianggap sebagai persoalan paling penting mengenai *unsupervised learning* (pembelajaran tak terawasi). Persoalan ini berhadapan dengan penemuan sebuah struktur atau pola dalam sebuah koleksi data tanpa label. Definisi lepas dari *clustering* adalah proses pengorganisasian obyek ke dalam kelompok-kelompok yang anggotanya serupa dalam beberapa cara (http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/).

Dengan demikian, *clustering* merupakan sebuah metode di mana data-data (*record)* yang serupa dikelompokkan menjadi satu. Kadang-kadang, *clustering* memiliki arti seperti segmentasi , yakni pemisahan. Data-data yang ada dipisah-pisahkan menurut kemiripannya satu sama lain. Contoh sederhana dari *clustering* adalah ketika seseorang mencuci pakaian. Pakaian-pakaian yang akan dicuci dipisahkan menurut warnanya. Pakaian putih dipisahkan dari pakaian yang berwarna. Oleh karena itu, *cluster* adalah sebuah koleksi obyek yang anggota-anggotanya saling serupa satu sama lain dan tidak serupa dengan obyek lain di luar *cluster* tersebut (Barakbah,2006). Gambar 2.1 mendeskripsikan pengertian *clustering*.

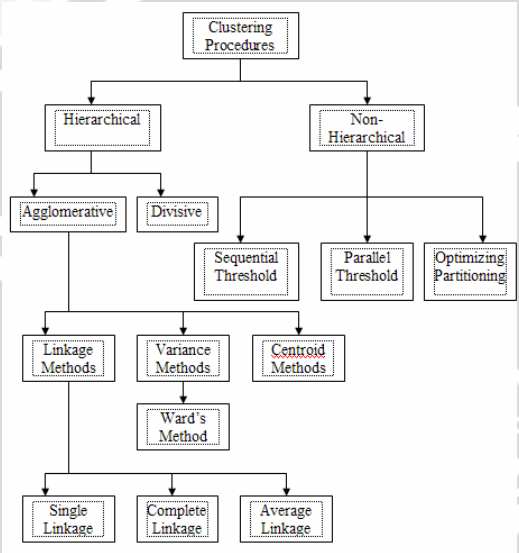


Gambar 2.1 *Clustering*

Clustering biasa digunakan dalam data mining. Teknik ini dapat diaplikasikan secara luas dalam berbagai bidang, beberapa di antaranya adalah untuk memprediksi segmentasi pasar, perilaku konsumen, juga pengelompokan dokumen yang memiliki

karakteristik sejenis. Selanjutnya, kelompok obyek hasil *clustering* tersebut dapat dianalisa untuk kepentingan penelitian lebih lanjut.

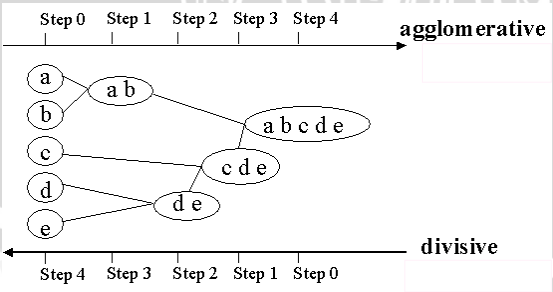
Clustering dapat dilakukan melalui pendekatan hirarkis dan non-hirarkis. Masing-masing metode dalam *clustering* dijabarkan dalam gambar 2.2.



Gambar 2.2 Diagram teknik *clustering*

2.2 Hierarchical Clustering

Hierarchical clustering adalah suatu algoritma *clustering* yang dijalankan dengan pendekatan hirarkis, yakni hasil *clustering* yang berupa tingkatan-tingkatan yang direpresentasikan dalam sebuah *tree* yang disebut *dendogram*. Gambar 2.3 adalah contoh *dendogram*.



Gambar 2.3 *Dendogram*

Seperti yang ditampilkan dalam gambar 2.2, *hierarchical clustering* memiliki dua pendekatan, yaitu:

a. Divisive
b. Agglomerative

2.2.1 Divisive

Metode divisive diawali dengan kondisi seluruh obyek data yang ada terletak dalam satu *cluster*. Kemudian *cluster* yang besar tersebut dibagi lagi menjadi *cluster-cluster* yang lebih kecil, hingga terbentuk *cluster* individu. Namun metode ini jarang sekali digunakan.

2.2.2 Agglomerative

Cara kerja agglomerative berkebalikan dengan divisive. Metode ini menggabungkan *cluster* yang ada secara iteratif. Pada kondisi awal, masing-masing obyek data dianggap sebagai *cluster* tersendiri. Dengan demikian, jika terdapat n data, maka *cluster* yang terbentuk adalah sejumlah n *cluster*. Kemudian, setelah jarak tiap-tiap *cluster* dengan *cluster* lainnya dihitung, dua *cluster* yang memiliki jarak terdekat satu sama lain digabungkan menjadi satu, sehingga jumlah *cluster* menjadi n-1. Langkah tersebut diulang hingga terbentuk *cluster* tunggal berukuran n.

Penghitungan jarak antara *cluster* dapat dilakukan dengan beberapa metode:

a. *Single linkage*: metode ini menggunakan jarak minimum antar anggota *cluster*, sehingga sering disebut juga *nearest neighbor method.* Metode ini sangat cocok untuk dipakai pada kasus *shape independent clustering*, karena kemampuannya untuk membentuk pola tertentu dari cluster. Namun metode ini tidak bagus untuk metode *condensed clustering* (Barakbah, 2006).
b. *Complete linkage*: yang digunakan adalah jarak maksimum antar anggota *cluster*. Maka dari itu, metode ini disebut *farthest neighbor method.* Metode ini sangat ampuh untuk memperkecil varian di dalam cluster karena melibatkan centroid pada penggabungan antar *cluster*, juga baik untuk data yang mengandung outlier. Pemaparan metode ini lebih jauh pada subbab selanjutnya.
c. *Average linkage*: menggunakan jarak rata-rata dari seluruh anggota *cluster*. Metode ini relatif yang terbaik dari metode-

metode hierarchical yang ada, namun ini harus dibayar dengan waktu komputasi yang paling tinggi dibandingkan dengan metode-metode lainnya.

2.3 Metode Complete-Linkage

Metode ini disebut juga metode tetangga terjauh (*the farthest neighbor method*). Penggunaan metode ini adalah untuk penghitungan jarak antar *cluster* yang dihitung berdasarkan jarak maksimal masing-masing obyek yang ada di dalam suatu *cluster* terhadap obyek-obyek dalam *cluster* lainnya.

Jarak antar *cluster* dihitung sebagai $D(x_i,y_j)$ di mana x_i adalah anggota *cluster* pertama, dan y_j adalah anggota *cluster* kedua. Dengan kata lain, jarak antara dua buah *cluster* dihitung sebagai jarak antara dua obyek terjauh dalam dua *cluster* tersebut.

Secara matematis, jarak kedua *cluster* dapat dinyatakan dalam persamaan berikut:

$$D(X,Y) = \max_{x \in X, y \in Y} d(x,y) \quad .....(2.1)$$

Berikut ini diberikan contoh penggunaan metode *complete-linkage* pada pengelompokan beberapa kota di Italia berdasarkan jarak dalam kilometer. Jarak kota-kota tersebut direpresentasikan dalam sebuah gambar dan matriks di bawah ini:

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0

Gambar 2.4 Langkah ke-0

Berdasarkan data pada mariks di atas, jarak terdekat antara dua kota adalah MI dan TO. Dengan demikian, pada iterasi ke-0, keduanya digabungkan menjadi satu *cluster*.

Iterasi ke-1:

Setelah MI-TO membentuk *cluster*, jarak masing-masing *cluster* dihitung ulang dengan rumus penghitungan jarak d(i,j) = max d(i,j). Berikut disajikan matriks baru berisi data jarak setelah penggabungan MI-TO:

	BA	FI	MI/TO	NA	RM
BA	0	662	996	255	412
FI	662	0	400	468	268
MI/TO	996	400	0	869	669
NA	255	468	869	0	219
RM	412	268	669	219	

Gambar 2.5 Langkah ke-1

Max d(i,j) = d(NA,RM) = 219, sehingga NA,RM bergabung menjadi satu *cluster* bernama NA/RM.

Iterasi ke-2:
Setelah NA,RM membentuk *cluster*, berikut matriks data jarak yang telah diperbaharui:

	BA	FI	MI/TO	NA/RM
BA	0	662	877	412
FI	662	0	295	468
MI/TO	877	295	0	869
NA/RM	412	468	869	0

Gambar 2.6 Langkah ke-2

Max d(i,j) = d(MI/TO,FI) = 295, sehingga MI/TO,FI bergabung menjadi satu *cluster* bernama MI/TO/FI.

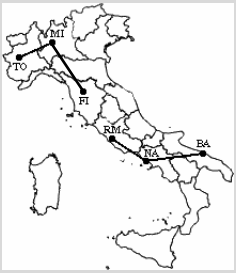
Iterasi ke-3:
Setelah MI/TO,FI membentuk *cluster*, berikut matriks data jarak yang telah diperbaharui:

	NA/RM	BA	MI/TO/FI
NA/RM	0	412	869
BA	412	0	877
MI/TO/FI	869	877	0

Gambar 2.7 Langkah ke-3

Max d(i,j) = d(NA/RM,BA) = 412, sehingga BA,NA/RM bergabung menjadi satu *cluster* bernama NA/RM/BA.

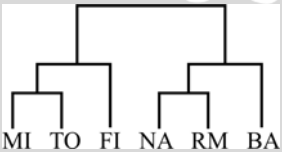
Iterasi ke-4:
Setelah BA,NA/RM membentuk *cluster*, berikut matriks data jarak yang telah diperbaharui:



	NA/RM/BA	MI/TO/FI
NA/RM/BA	0	877
MI/TO/FI	877	0

Gambar 2.8 Langkah ke-4

Terakhir, kedua *cluster* yang terbentuk digabung menjadi satu *cluster* tunggal. Dengan kata lain, *cluster* terakhir ini mencakup seluruh data yang ada. Langkah ke-0 sampai dengan ke-4 membentuk pohon hirarki pada gambar 2.9.



Gambar 2.9 Pohon hirarki *clutering* beberapa kota di Italia

2.4 Struktur Morfologi Bahasa Indonesia

Struktur kata dalam Bahasa Indonesia terdiri dari kata dasar dan imbuhan, yakni awalan, akhiran, dan pasangan kombinasi dari keduanya (Departemen Pendidikan dan Kebudayaan, 1987). Morfologi kata-kata dalam Bahasa Indonesia terdiri dari struktur infleksional dan derivatif. Infleksional adalah struktur paling sederhana yang dinyatakan dengan akhiran yang tidak mempengaruhi arti utama dari kata dasarnya. Akhiran infleksional dapat dibagi menjadi dua kelompok:

1. Akhiran *-lah, -kah, -pun, -tah*

Akhiran-akhiran tersebut sesungguhnya merupakan partikel atau kata fungsi yang tidak memiliki arti. Kemunculannya dalam suatu kata adalah untuk penekanan, contoh:

dia + kah → *diakah*

memiliki arti ”dia”, dengan penekanan yang mengandung pertanyaan

saya + lah → *sayalah*

memiliki arti “saya”, dengan penekanan yang mengandung penegasan

2. Akhiran *-ku, -mu, -nya*

Akhiran-akhiran tersebut, yang ditambahkan pada kata, membentuk kata ganti milik, contoh:

tas + mu → *tasmu* (tas milik kamu)

sepeda + ku → *sepedaku* (sepeda milik aku)

Masing-masing akhiran dalam kelompok 1 maupun 2 dapat muncul dalam kata yang sama. Ketika keduanya muncul, harus mengikuti aturan bahwa akhiran dari kelompok kedua selalu mendahului kelompok pertama. Penambahan akhiran-akhiran ini tidak akan mengubah pengejaan kata dasar. Dengan kata lain, tidak ada huruf dalam kata dasar yang diluluhkan pada kata jadiannya.

Sedangkan akhiran derivasional adalah *-i, -kan, -an*. Contoh-contoh kata yang mengandung akhiran tersebut yaitu:

gula + i → gulai (memberi gula pada)

makan + an → makanan (sesuatu yang dapat dimakan)

beri + kan → berikan (beri kepada)

Awalan yang sering muncul adalah *ber-, di-, ke-, meng-, peng-, per-, ter-* (Tarigan, 1995). Berikut adalah contoh penggunaan masing-masing awalan:

ber + lari → berlari

di + makan → dimakan

ke + kasih → kekasih
meng + ambil → mengambil
peng + atur → pengatur
per + lebar → perlebar
ter + baca → terbaca

Beberapa awalan seperti *ber-*, *meng-*, *peng-*, *per-*, *ter-* kadang muncul dalam beberapa bentuk lain. Bentuknya tergantung dari huruf pertama dari kata yang ditambahkan. Tidak seperti struktur infleksional, pengejaan katanya dapat berubah ketika awalan ditambahkan.

Sebagai contoh, kata *menyapu* terbentuk dari awalan *meng-* dan kata dasar *sapu*. Dalam kasus ini, awalan *meng-* berubah menjadi *meny-* dan huruf pertama dari kata dasar menjadi luluh. Aturan untuk bentuk-bentuk penambahan awalan yang lainnya dapat dilihat pada lampiran.

2.5 Text Pre-processing

Untuk dapat memproses suatu dokumen teks diperlukan beberapa tahapan, yaitu *case folding*, *parsing*, penghilangan *stopword*, dan *stemming*.

2.5.1 Case folding

Dalam tahapan ini dilakukan semua huruf dalam dokumen diubah menjadi huruf kecil. Hanya huruf 'a' sampai 'z' yang diterima, sedangkan karakter selain huruf dihilangkan dan dianggap sebagai token, yaitu karakter dasar yang sudah tidak dapat diturunkan lagi.

2.5.2 Parsing

Parsing merupakan proses penguraian kalimat menjadi kata-kata dengan menggunakan analisis sintaks atau tata bahasa. Bahasa Indonesia memiliki aturan dalam pembentukan frase maupun kalimat. Aturan inilah yang menjadi acuan dalam proses *parsing*.

2.5.3 Penghilangan *Stopword*

Stopword adalah kata-kata yang seringkali muncul dalam dokumen namun artinya tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu (Tala,2003). Kata-kata yang termasuk dalam *stopword* diantaranya artikel, preposisi, dan kata hubung. Misalnya 'dan', 'sebenarnya', 'sebuah', dan seterusnya. Jumlah *stopword* sangat banyak hingga mencapai ratusan, dapat dilihat pada lampiran.

2.5.4 Stemming

Stemming adalah proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama, dalam hal ini menguraikan kata berimbuhan menjadi bentuk kata dasarnya. *Stemming* yang digunakan adalah algoritma IndoStemmer, yang merupakan pengembangan dari algoritma Porter Stemmer yang disesuaikan dengan aturan bahasa Indonesia. IndoStemmer merupakan sistem yang dihasilkan dari penelitian yang telah ada sebelumnya (Daqiqil ID, 2007).

Stemmer ini dibagi menjadi dua komponen, yaitu *ruleset* dan *stemmer*. *Ruleset* adalah kumpulan aturan-aturan yang akan digunakan oleh *stemmer* untuk mengubah kata jadian menjadi kata dasar, sedangkan *stemmer* sendiri adalah sebuah *parser* yang digunakan untuk membaca *ruleset* dan melakukan proses *stemming*. *Ruleset* dibuat dalam bentuk file XML dengan tujuan agar *ruleset* tersebut mudah dikustomisasi.

Langkah untuk melakukan proses stemming kata dapat dibagi menjadi tiga tahap yaitu:

a. Pembersihan kata
   Menghilangkan karakter-karakter yang bukan bagian dari kata, seperti tanda titik (.), koma (,), tanda seru (!), dan lain-lain.
b. Pembacaan *ruleset*
   Membaca dokumen XML yang berisi *ruleset*. Sebelum dibaca, dokumen tersebut harus dicek terlebih dahulu apakah sudah sesuai dengan dokumen pendefinisinya.
c. Pemisahan awalan dan akhiran
   Pemisahan awalan dan akhiran dilakukan dengan membaca *ruleset* yang telah disimpan di memori. Proses pemisahan akan dilakukan sampai *state* stop atau jumlah *state* telah mencapai batas maksimal *state*. Jika kata memenuhi syarat dan kondisi

sebuah transisi, maka akan dilakukan proses pemisahan awalan atau akhiran sesuai dengan nama *action* pada transisi tersebut.

Penghilangan awalan (*prefix removal*) dibagi menjadi dua tahap yang disusun berdasarkan pola yang ada pada aturan bahasa Indonesia, yaitu jumlah awalan sebuah kata dasar maksimal adalah dua. *State* pertama berisi awalan me-, di-, ter-, ke-, karena awalan-awalan ini selalu mendapat posisi paling depan jika sebuah kata jadian memiliki awalan bertingkat. *State* kedua berisi ber- dan per-. Awalan ini dapat digabung dengan awalan *layer/ state* pertama, tetapi posisinya selalu setelah awalan *layer/ state* pertama.

Penghilangan akhiran (*suffix removal*) dibagi menjadi empat *state*. *State* pertama berisi akhirn –an, -wan, -wati -is, -isme, dan –isasi. *State* kedua berisi akhiran –i, dan -kan. *State* ketiga berisi akhiran –ku, -mu, -nya. Dan *State* keempat berisi akhiran –lah, dan –kah. Tidak semua *state* akhiran dapat digabungkan, tetapi hanya *state*- *state* tertentu saja yang bisa digabung dan sesuai urutannya.

Pembagian *state* berdasar fungsi dan urutan dari masing-masing akhiran adalah sebagai berikut:

a. Akhiran S1 (*State* pertama) : membentuk kata benda
b. Akhiran S2 : membuat kata kerja transitif
c. Akhiran S3 : mempunyai arti ”memiliki”
d. Akhiran S4 : merupakan partikel dalam sebuah kata

Penggabungan yang dapat dilakukan adalah akhiran S1 dengan S3 dan atau S4. Sedangkan S4 yang merupakan partikel dapat bergabung dengan akhiran apa saja asal posisi partikel harus berada di posisi akhir. Pembagian *state* ini digambarkan melalui *state diagram* oleh (Daqiqil ID, 2007) seperti yang tertera pada lembar lampiran.

2.6 Pembobotan

a. Bobot Kalimat

Bobot kalimat adalah nilai sebuah kalimat yang mengindikasikan seberapa penting arti kalimat tersebut pada sebuah paragraf. Semakin tinggi nilai kalimatnya semakin penting pula artinya dalam paragraf (Budhi, 2008). Proses *parsing, stemming,* dan penghilangan *stopword* harus dikerjakan terlebih dahulu sebelum proses penghitungan bobot kalimat dilakukan.

Pada penelitian ini digunakan metode penghitungan bobot kalimat seperti pada penelitian sebelumnya (Budhi, 2008). Ada lima macam bobot yang digunakan:

*W*1 : Banyaknya kata yang sama antara kalimat yang dihitung dengan daftar kata kunci (*keyword*) pada dokumen tempat kalimat tersebut berada.

*W*2 : Nilai yang ditentukan dari kemunculan kata-kata di dalam kalimat terhadap pemakaian kata-kata tersebut pada dokumen tempat kalimat berada.

$$\frac{\sum_{i=1}^{n} kp_i \times fkp_i}{\sum k} \quad ....(2.2)$$

kp_i = kata penting ke-i
fkp_i = frekuensi kata penting ke-i pada dokumen
$\sum k$ = jumlah total kata yang ada dalam dokumen

*W*3 : Nilai ini ditentukan oleh posisi di mana kalimat tersebut berada terhadap paragrafnya. Berdasarkan kaidah Deduktif-Induktif bahasa Indonesia ada dua macam nilai yang dipakai, yaitu: bila kalimat tersebut berada pada awal/akhir paragraf memiliki bobot 2, sementara bila tidak memiliki bobot 1.

*W*4 : Menghitung banyaknya relasi sebuah kalimat dengan kalimat-kalimat lain pada dokumen yang sama. Relasi masing-masing kalimat disajikan dalam bentuk graf. Masing-masing kalimat menjadi *node* pada graf. Kalimat-kalimat yang memiliki kemiripan dihubungkan oleh *edge*. Menurut (Sjobergh, 2005) kemiripan berarti kata-katanya saling cocok. Dalam hal ini, bila dua kalimat memiliki paling tidak satu buah kata yang sama, maka kedua kalimat tersebut akan terhubung oleh *edge*.

*W*5 : Jumlah banyaknya kata kunci yang sama antara kalimat yang dihitung dengan daftar kata kunci pada semua dokumen yang akan diintegrasikan.

Dengan demikian, bobot kalimat total (W) dapat dirumuskan dengan persamaan 2.3.

$W(S_j) = W1(S_j) + W2(S_j) + W3(S_j) + W4(S_j) + W5(S_j)$ .....(2.3)

S_j adalah kalimat ke-j, dengan j adalah nomor kalimat pada dokumen.

b. Bobot Relasi Antar Kalimat

Bobot relasi antar dua kalimat adalah sama dengan jarak antara kedua kalimat tersebut. Konsekuensinya adalah bila bobot relasi antara dua kalimat tertentu lebih kecil dari yang lain, maka jarak keduanya juga lebih dekat (Budhi, 2008). Secara formal, misal terdapat n kalimat P= $\{S_1, S_2, ..., S_n\}$, maka bobot relasi antara dua kalimat S_i dan S_j dinyatakan dalam persamaan 2.4.

$$R(Si,Sj) = \left\{ \begin{array}{l} \dfrac{(j-i)^2}{\alpha(Si,Sj) \times W(Sj)}, i < j \\ \infty, i \geq j \end{array} \right\} \text{..... (2.4)}$$

di mana i, j adalah berturut-turut letak kalimat ke i dan ke j; $\alpha(Si,Sj)$ adalah jumlah kata yang sama antara S_i dan S_j setelah *stopword* yang ada dihilangkan; dan $W(S_j)$ adalah bobot kalimat ke-j.

2.7 Evaluasi

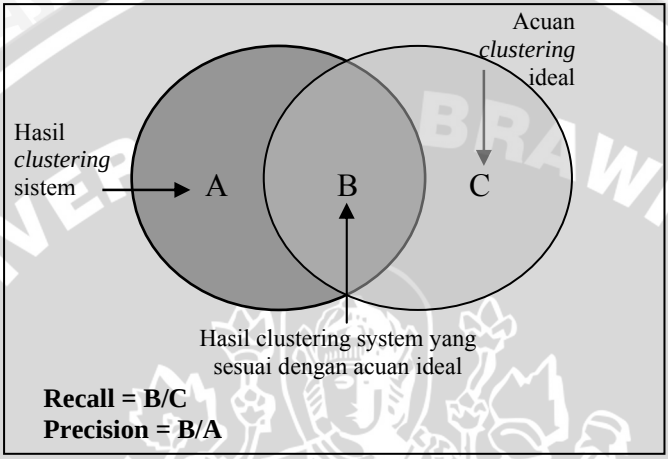
Metode untuk evaluasi hasil penggabungan, sama halnya dengan sebuah ringkasan merupakan topik yang cukup sulit, baik evaluasi terhadap dokumen yang dihasilkan dari mesin pemroses otomatis ataupun hasil manual oleh abstraktor atau ahli bahasa profesional. Hal ini dikarenakan tidak terdapat definisi dokumen ideal. Terdapat dua klasifikasi metode evaluasi (Sparck dan Galliers, 1996) yaitu:

1. Extrinsik, kualitas dokumen yang diukur berdasarkan bagaimana membantu penyelesaian tugas *user*.
2. Intrinsik, hanya diukur dari kualitas hasil (*output*) dokumen yang dihasilkan.

Metode yang ada saat ini adalah intrinsik. Pengevaluasi menciptakan sekumpulan dokumen ideal, masing-masing satu untuk menguji dokumen dan membandingkannya dengan hasil sistem. Dokumen ideal adalah dokumen yang dibuat secara manual oleh orang yang berkompeten di bidang tata bahasa. Hasil evaluasi ini merupakan ukuran efektifitas suatu sistem. Sedangkan efisiensi sistem diukur dari rata-rata waktu yang digunakan untuk memproses sebuah dokumen.

Dalam penelitian ini digunakan standar pengukuran *recall, precision dan F-measure. Recall* adalah jumlah dari kalimat anggota

cluster sesuai yang dihasilkan oleh sistem. *Precision* adalah ukuran dari jumlah kalimat anggota *cluster* yang dihasilkan sistem dan kalimat tersebut sesuai. Hubungan antara *precision dan recall* ditunjukkan oleh gambar 2.10.



Gambar 2.10 Ilustrasi dari *recall* dan *precision*

Sebagai tambahan untuk ukuran *recall* dan *precision*, terdapat satu lagi kunci ukuran untuk menentukan performansi sistem, yaitu yang disebut dengan *F-measure*. F_1 *measure* adalah ukuran rata-rata keseimbangan dari *recall* dan *precision*. Rata-rata keseimbangan akan membantu meminimalkan pengaruh dari suatu variabel, sementara pada waktu yang sama akan mempenalti kontribusi yang ganjil (pencilan). Nilai *F-measure* merupakan gabungan antara *recall* dan *precision y*ang didefinisikan dengan persamaan berikut ini (Yang dan Liu, 1999) :

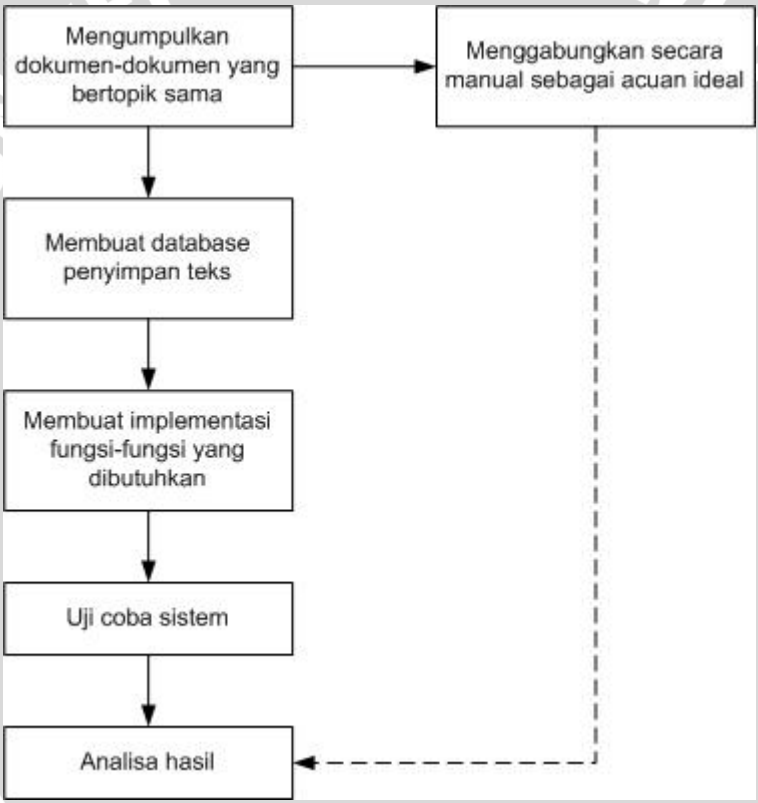
$$F_1 = \frac{2 \times recall \times precision}{recall + precision} \quad \text{.....(2.5)}$$

UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi

Penelitian ini dilakukan melalui beberapa tahapan. Diawali dengan penentuan topik yang akan akan diteliti dan dilanjutkan dengan studi literatur. Studi literatur mencakup pencarian jurnal dan referensi pendukung penelitian. Gambar 3.1 menyajikan tahapan-tahapan penelitian yang dilakukan.



Gambar 3.1 Tahapan penelitian

3.2 Analisis Sistem

Pada subbab analisis sistem ini dijelaskan mengenai deskripsi umum sistem dan batasan-batasan dari sistem.

3.2.1 Deskripsi Umum Sistem

Sistem yang akan dibuat ini mendukung penggabungan beberapa dokumen yang membahas topik materi yang sama secara otomatis. Sebelum proses dilakukan, terlebih dahulu pengguna harus menentukan dokumen mana yang akan digabungkan, sehingga dokumen-dokumen yang telah dipilih dianggap membahas materi yang sama dan telah memiliki kesamaan topik. Kemudian, dilakukan proses *text pre-processing*, yaitu meliputi *parsing, stemming* dan penghilangan *stopword*. Dalam proses parsing, masing-masing kalimat dipecah menjadi kata. Setelah *parsing*, dilakukan penghilangan *stopword*, baru kemudian dilakukan proses *stemming*. Setelah *text pre-processing* dilakukan, kata-kata yang merupakan *keyword* ditandai untuk proses selanjutnya, yaitu penghitungan bobot.

Proses selanjutnya adalah penghitungan bobot. Bobot dihitung sesuai kriteria yang telah dijelaskan pada bab 2. Setelah dihitung bobot masing-masing kalimat, dilakukan penghitungan bobot relasi antar kalimat. Bobot relasi ini kemudian dipakai sebagai masukan untuk proses *clustering* untuk penggabungan dokumen dengan metode *complete linkage agglomerative hierarchical clustering*. Sebelum digabungkan, kalimat yang memiliki jarak sangat dekat dengan kalimat lain, dengan kata lain memiliki bobot relasi sangat kecil, akan dihilangkan. Asumsinya, kedua kalimat tersebut sangat mirip sehingga tidak perlu ditampilkan keduanya, cukup salah satu saja.

Pada proses penggabungan, awalnya semua kalimat pada semua dokumen dianggap sebagai *atomic cluster*. Selanjutnya, secara bertahap *cluster-cluster* tersebut disatukan menggunakan aturan *maximum distance* pada persamaan 2.1. Setelah semua kalimat tergabung menjadi sebuah *cluster*, dilakukan proses untuk memecah *cluster* menjadi paragraf-paragraf. Cara yang dipakai adalah, kalimat yang terlebih dahulu bergabung menjadi *cluster-cluster* besar dianggap sebagai sebuah paragraf tersendiri. Asumsinya, bila kalimat-kalimat tersebut bergabung secara alami, dapat dianggap kalimat-kalimat tersebut memiliki tingkat kesamaan yang tinggi dan

membahas sub-topik bahasan yang sama. Sementara untuk kalimat-kalimat tersisa yang memiliki bobot relasi tak hingga dan tidak dapat bergabung ke dalam *cluster-cluster* besar akan dihilangkan karena dianggap tidak memiliki keterkaitan antarkalimat. Dengan demikian, dokumen hasil *clustering* akan menjadi lebih ringkas karena hanya memuat kalimat-kalimat yang dianggap penting dan saling memiliki keterkaitan dengan topik yang bersangkutan.

Setelah proses penggabungan selesai, dokumen hasil penggabungan disajikan dalam bentuk teks yang dapat disimpan ke dalam bentuk file oleh pengguna.

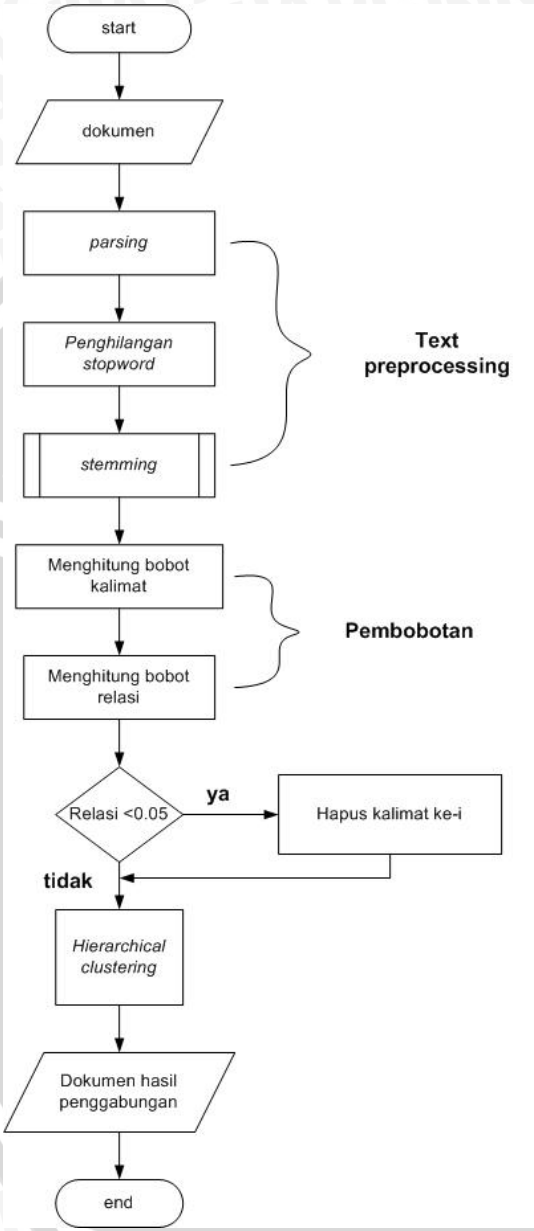
3.2.2 Batasan Sistem

Berikut ini adalah beberapa batasan dari sistem yang akan dibuat:

1. *Stopword* yang digunakan adalah *stopword* berbahasa Indonesia
2. Kesalahan akibat proses *stemming* tidak diperhitungkan karena menggunakan sistem *stemming* Indostemmer yang telah ada sebelumnya.
3. Evaluasi dilihat dari kebenaran pengelompokan kalimat dalam paragraf dibandingkan dengan dokumen ideal tanpa memperhitungkan urutan kalimat dalam paragraf.

3.3 Perancangan Proses

Proses penggabungan dokumen melalui beberapa tahapan, diantaranya *text preprocessing*, pembobotan kalimat, dan *clustering*. Pembobotan kalimat dilakukan untuk menentukan nilai masing-masing kalimat yang nantinya digunakan untuk proses selanjutnya, yaitu *clustering.* Proses *clustering* digunakan untuk menyusun ulang kalimat-kalimat dalam dokumen hasil penggabungan sesuai bobot relasi yang menunjukkan hubungan antar kalimatnya. Secara jelas proses penggabungan dokumen ditunjukkan dalam gambar 3.2



Gambar 3.2 Diagram alir proses penggabungan dokumen

3.3.1 Text Pre-processing

Berdasarkan gambar 3.2, masukan berupa dokumen. Dalam hal ini, dokumen ditentukan oleh pengguna. Sehingga, dokumen yang dipilih dianggap telah memiliki kesamaan topik. Sebagai contoh, berikut disajikan dokumen yang akan digabungkan berjumlah dua buah. Gambar 3.3 dan gambar 3.4 adalah contoh kedua dokumen yang akan digabungkan tersebut.

"Samsung Pecahkan Rekor Layar *LCD* Tertipis"

Inovasi tiada henti terus dipertunjukkan Samsung Electronics. Kali ini perusahaan elektronik asal Korea Selatan tersebut memperkenalkan layar LCD yang diklaim tertipis di dunia. Ukuran ketebalan LCD 0,74 mm mematahkan rekor layar LCD tertipis yang sebelumnya juga dibuat Samsung, setebal 0,82 mm.

Layar LCD super tipis bikinan perusahaan dari Negeri Ginseng itu tampaknya akan diaplikasikan pada ponsel Ultra Edition generasi mendatang. Sebagai perbandingan, ponsel ramping Ultra Edition saat ini memiliki ketebalan layar 1,9 mm. Apabila layar LCD tersebut nantinya telah diproduksi massal maka ponsel-ponsel Samsung akan meninggalkan para rivalnya untuk urusan desain ponsel teramping.

Gambar 3.3 Contoh dokumen1

"*LCD* Tertipis di Dunia"

Tren peranti *mobile* mendatang rupanya mengarah kepada bentuk desain yang kompak, tipis, dan tahan goncangan. LCD yang ada di pasaran saat ini selain rentan akan goncangan, juga cenderung masih tebal sehingga dari segi desain kurang terlihat kompak. Samsung melalui inovasinya dengan menciptakan LCD super tipis dengan ketebalan hanya 0,74 mm (setara ketebalan kartu kredit), menjadikan peranti *mobile* mendatang lebih kompak, tipis, dan tahan goncangan. LCD hasil inovasi Samsung ini berhasil mengalahkan LCD keluaran LG yang terpaut hanya 0,07 mm lebih tipis, menjadikannya sebagai LCD tertipis di dunia. LCD dari Samsung ini akan mulai diproduksi secara massal sekitar pertengahan 2007.

Gambar 3.4 Contoh dokumen2

Pada proses selanjutnya, kedua dokumen tersebut diuraikan menjadi kalimat-kalimat (dalam hal ini, judul diabaikan), dan masing-masing kalimat diberi nomor sesuai urutannya pada masing-masing dokumen tempat kalimat tersebut berada, sehingga menjadi kalimat yang disertai *index* seperti yang ditampilkan pada gambar 3.5 dan 3.6. Kalimat-kalimat ber-*index* tersebut kemudian diduplikat dan disimpan dalam basis data, sehingga dokumen asli tetap tersimpan sementara duplikatnya digunakan selama proses penggabungan dokumen. Nantinya, setelah proses penggabungan selesai dan menghasilkan urutan *index* yang baru, kalimat yang tersimpan dalam basis data digunakan untuk menggabungkan kembali kalimat yang telah dipisah dan dihilangkan bagian-bagian *stopword*-nya selama *text preprocessing*.

[1]Inovasi tiada henti terus dipertunjukkan Samsung Electronics.
[2]Kali ini perusahaan elektronik asal Korea Selatan tersebut memperkenalkan layar LCD yang diklaim tertipis di dunia.
[3]Ukuran ketebalan LCD 0,74 mm mematahkan rekor layar LCD tertipis yang sebelumnya juga dibuat Samsung, setebal 0,82 mm.
[4]Layar LCD super tipis bikinan perusahaan dari Negeri Ginseng itu tampaknya akan diaplikasikan pada ponsel Ultra Edition generasi mendatang.
[5]Sebagai perbandingan, ponsel ramping Ultra Edition saat ini memiliki ketebalan layar 1,9 mm.
[6]Apabila layar LCD tersebut nantinya telah diproduksi massal maka ponsel-ponsel Samsung akan meninggalkan para rivalnya untuk urusan desain ponsel teramping.

Gambar 3.5 Dokumen1 dalam bentuk kalimat ber-*index*

[1]Tren peranti *mobile* mendatang rupanya mengarah kepada bentuk desain yang kompak, tipis, dan tahan goncangan.
[2]LCD yang ada di pasaran saat ini selain rentan akan goncangan, juga cenderung masih tebal sehingga dari segi desain kurang terlihat kompak.
[3]Samsung melalui inovasinya dengan menciptakan LCD super tipis dengan ketebalan hanya 0,74 mm (setara ketebalan kartu kredit), menjadikan peranti *mobile* mendatang lebih kompak, tipis, dan tahan goncangan.
[4]LCD hasil inovasi Samsung ini berhasil mengalahkan LCD keluaran LG yang terpaut hanya 0,07 mm lebih tipis, menjadikannya sebagai LCD tertipis di dunia.
[5]LCD dari Samsung ini akan mulai diproduksi secara massal sekitar pertengahan 2007.

Gambar 3.6 Dokumen2 dalam bentuk kalimat ber-*index*

Proses selanjutnya adalah *parsing*. *Parsing* yang digunakan adalah *parsing* sederhana yaitu memecah sebuah teks menjadi kumpulan kata-kata tanpa memperhatikan keterkaitan antar kata dan peran atau kedudukannya dalam kalimat. Karakter yang diterima dalam pembentukan kata adalah karakter huruf saja. Dengan demikian, seperti kata ulang yang ada dalam kaidah bahasa Indonesia akan diurai menjadi dua kata bukan satu kesatuan kata. Karena karakter yang diterima hanya berupa huruf, maka angka dan tanda baca lain dianggap sebagai delimiter dan tidak dibaca sebagai bagian dari dokumen.

Pada gambar 3.5 dan 3.6, kata yang digarisbawahi merupakan *stopword* berdasarkan (Tala, 2003). Dalam proses penghilangan stopword, kata-kata tersebut dihilangkan. Kata-kata yang tersisa akan melalui proses *stemming*. Setelah melalui proses *stemming*, semua kata-kata yang berimbuhan diubah menjadi kata dasarnya.

Langkah untuk melakukan proses stemming kata dapat dibagi menjadi tiga tahap yaitu:

a. Pembersihan kata
   Menghilangkan karakter-karakter yang bukan bagian dari kata, seperti tanda titik (.), koma (,), tanda seru (!), dan lain-lain.
b. Pembacaan *ruleset*
   Membaca dokumen XML (Extensible Markup Language) yang berisi *ruleset*. Sebelum dibaca, dokumen tersebut harus dicek terlebih dahulu apakah sudah sesuai dengan dokumen pendefinisinya.

c. Pemisahan awalan dan akhiran

Pemisahan awalan dan akhiran dilakukan dengan membaca *ruleset* yang telah disimpan di memori. Proses pemisahan akan dilakukan sampai *state* stop atau jumlah *state* telah mencapai batas maksimal *state.* Jika kata memenuhi syarat dan kondisi sebuah transisi, maka akan dilakukan proses pemisahan awalan atau akhiran sesuai dengan nama *action* pada transisi tersebut. Gambar 3.9 menggambarkan prosedur *stemming* yang dilakukan pada tiap-tiap dokumen.

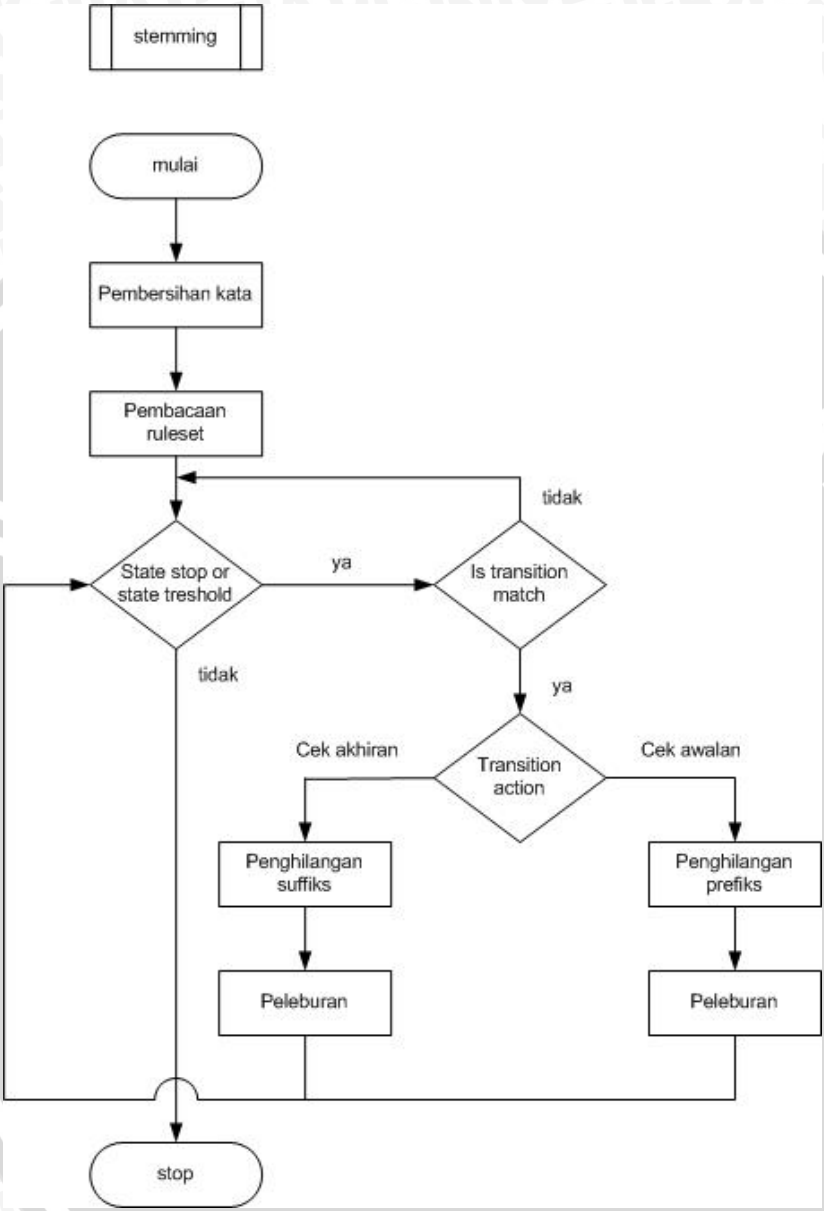
Gambar 3.7 dan 3.8 adalah perubahan dokumen setelah melalui *text pre-processing*. Angka yang terdapat pada dokumen merupakan nomor kalimat

[1]inovasi tiada henti tunjuk samsung electronics
[2]kali usaha elektronik asal korea selatan kenal layar lcd klaim tipis dunia
[3]ukur tebal lcd mm patah rekor layar lcd tipis samsung tebal mm
[4]layar lcd super tipis bikin usaha negeri ginseng tampak aplikasi ponsel ultra edition generasi datang
[5]banding ponsel ramping ultra edition milik tebal layar mm
[6]layar lcd produksi massal ponsel ponsel samsung tinggal rival urusan desain ponsel ramping

Gambar 3.7 Dokumen1 setelah *text pre-processing*

[1]tren peranti mobile datang arah bentuk desain kompak tipis tahan goncang
[2]lcd pasar rentan goncang cenderung tebal segi desain kurang lihat kompak
[3]samsung inovasi cipta lcd super tipis tebal mm setara tebal kartu kredit jadi peranti mobile datang kompak tipis tahan goncang
[4]lcd hasil inovasi samsung hasil kalah lcd keluar lg paut mm tipis jadi lcd tipis dunia
[5]lcd samsung mulai produksi secara massal tengah

Gambar 3.8 Dokumen2 setelah *text pre-processing*



Gambar 3.9 Diagram alir proses *stemming*

3.3.2 Pembobotan

a. Bobot Kalimat

Berdasarkan persamaan 2.3, yang pertama kali dihitung adalah bobot masing-masing kalimat yang terdiri dari lima macam. Untuk menghitung W2, perlu diketahui terlebih dahulu frekuensi tiap-tiap kata pada masing-masing dokumen. Kemudian, ditentukan kata-kata mana yang termasuk kata penting dalam dokumen. Dalam hal ini, suatu kata dianggap sebagai kata penting apabila memiliki frekuensi lebih dari tiga (>3).

Berdasarkan tabel 3.1, kata-kata penting dalam dokumen1 yaitu: samsung, layar, lcd, tipis, dan ponsel. Sedangkan jumlah kata yang terkandung dalam dokumen1 adalah sebanyak 44 buah kata.

Untuk dokumen2, penghitungan frekuensi disajikan dalam tabel 3.2. Berdasarkan tabel 3.2, kata-kata penting dalam dokumen2 adalah: lcd dan tipis. Sedangkan jumlah kata yang terkandung dalam dokumen2 adalah sebanyak 39 buah kata. Setelah diketahui kata-kata yang dianggap kata penting dalam tiap-tiap dokumen, dilakukan penghitungan bobot W2.

Tabel 3.1 Frekuensi kata pada dokumen1

Kata	Frek	Kata	Frek	Kata	Frek
samsung	4	asal	1	tampak	1
pecah	1	korea	1	aplikasi	1
rekor	2	selatan	1	ponsel	5
layar	6	kenal	1	ultra	2
lcd	6	klaim	1	edition	2
tipis	4	dunia	1	generasi	1
inovasi	1	ukur	1	datang	1
tiada	1	tebal	3	banding	1
henti	1	mm	3	ramping	2
tunjuk	1	patah	1	milik	1
electronics	1	super	1	produksi	1
kali	1	bikin	1	massal	1

usaha	2	negeri	1	tinggal	1
elektronik	1	ginseng	1	rival	1
urusan	1	desain	1		

Tabel 3.2 Frekuensi kata pada dokumen2

Kata	Frek	Kata	Frek	Kata	Frek
lcd	7	rentan	1	jadi	2
tipis	6	cenderung	1	hasil	2
dunia	2	tebal	3	kalah	1
tren	1	segi	1	keluar	1
peranti	2	kurang	1	lg	1
mobile	2	lihat	1	paut	1
datang	2	samsung	3	mulai	1
arah	1	inovasi	2	produksi	1
bentuk	1	cipta	1	secara	1
desain	2	super	1	massal	1
kompak	3	mm	2	tengah	1
tahan	2	setara	1	pasar	1
goncang	3	kartu	1	kredit	1

Penghitungan bobot W2 dilakukan sesuai dengan persaman 2.2. S_j adalah kalimat pada kedua dokumen dengan nomor kalimat seperti pada gambar 3.5 dan 3.6. Dengan demikian $W2(S_j)$ merupakan bobot dari masing-masing kalimat tersebut.

$W2(S_1)$: $\frac{1}{67} \times 4 = 0.0597$

$W2(S_2)$: $\frac{1}{67} \times 6 + \frac{1}{67} \times 6 + \frac{1}{67} \times 4 = 0.2388$

$W2(S_3)$: $\frac{2}{67} \times 6 + \frac{1}{67} \times 6 + \frac{1}{67} \times 4 + \frac{1}{67} \times 4 = 0.3880$

$W2(S_4)$: $\frac{1}{67} \times 6 + \frac{1}{67} \times 6 + \frac{1}{67} \times 4 + \frac{1}{67} \times 5 = 0.3134$

$W2(S_5)$: $\frac{1}{67} \times 5 + \frac{1}{67} \times 6 = 0.1641$

$W2(S_6)$: $\frac{1}{67} \times 6 + \frac{1}{67} \times 6 + \frac{3}{67} \times 5 + \frac{1}{67} \times 4 = 0.4626$

$W2(S_7)$: $\frac{1}{64} \times 6 = 0.0937$

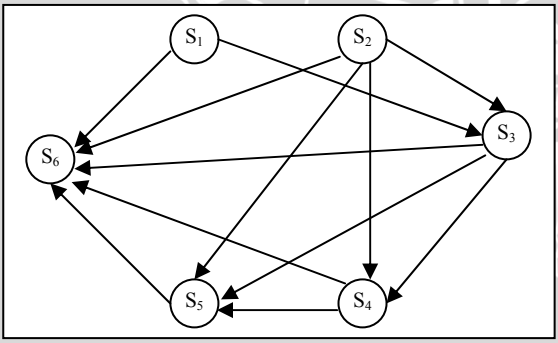
$W2(S_8)$: $\frac{1}{64} \times 7 = 0.1093$

$W2(S_9)$: $\frac{1}{64} \times 7 + \frac{2}{64} \times 6 = 0.2968$

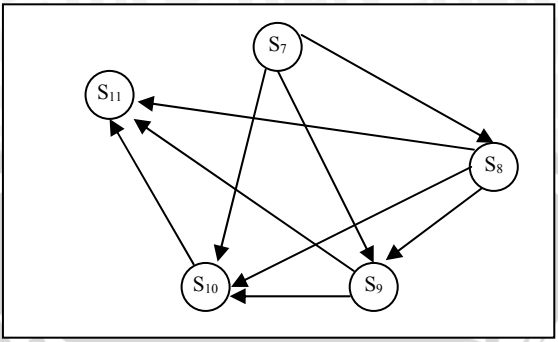
$W2(S_{10})$: $\frac{3}{64} \times 7 + \frac{2}{64} \times 6 = 0.5156$

$W2(S_{11})$: $\frac{1}{64} \times 7 = 0.1093$

Untuk dapat mengetahui berapa banyak relasi yang dimiliki masing-masing kalimat, dibuat dua buah graf. Graf pertama pada gambar 3.10 untuk kalimat-kalimat pada dokumen1, sedangkan graf kedua untuk kalimat-kalimat pada dokumen2 disajikan pada gambar 3.11.



Gambar 3.10 Graf Relasi Kalimat Dokumen1



Gambar 3.11 Graf Relasi Kalimat Dokumen2

Berdasarkan gambar 3.10 dan 3.11, relasi antar kalimat dihitung dengan banyaknya *edge* yang dimiliki tiap-tiap kalimat. Dengan demikian, bobot masing-masing kalimat pada contoh dokumen1 dan dokumen2 disajikan pada tabel 3.3. Karena interval nilai yang dimiliki tiap bobot berbeda-beda, maka agar tiap-tiap bobot memiliki pengaruh yang sama besar terhadap penghitungan bobot total, maka dilakukan proses normalisasi seperti pada tabel 3.4.

Tabel 3.3 Bobot kalimat

Kalimat ke	1	2	3	4	5	6	7	8	9	10	11
W1	1	3	5	4	2	6	1	1	3	5	1
W2	0.06	0.24	0.39	0.31	0.16	0.46	0.09	0.11	0.30	0.52	0.11
W3	2	1	2	2	1	2	2	1	1	1	2
W4	2	4	5	4	4	5	3	4	4	4	3
W5	1	3	5	4	2	6	1	1	4	6	2

Tabel 3.4 Bobot kalimat setelah dinormalisasi

Kalimat ke	1	2	3	4	5	6	7	8	9	10	11
W1	0.17	0.5	0.83	0.67	0.33	1	0.17	0.17	0.5	0.83	0.17
W2	0.11	0.46	0.75	0.60	0.31	0.88	0.17	0.21	0.58	1	0.21
W3	1	0.5	1	1	0.5	1	1	0.5	0.5	0.5	1
W4	0.4	0.8	1	0.8	0.8	1	0.6	0.8	0.8	0.8	0.6
W5	0.17	0.5	0.83	0.67	0.33	1	0.17	0.17	0.67	1	0.33
$W(S_i)$	1.85	2.76	4.41	3.74	2.27	4.88	2.11	1.85	3.05	4.13	2.31

b. Bobot Relasi

Setelah didapatkan bobot dari masing-masing kalimat, kemudian dihitung bobot relasi dari masing-masing kalimat tersebut sesuai dengan persamaan 2.4. Hasil penghitungan bobot relasi disajikan dalam bentuk matrik pada tabel 3.5.

Tabel 3.5 Bobot relasi

i→j	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
S_1	∝	∝	0.93	∝	∝	6.72	∝	∝	9.41	9.42	41.49
S_2	∝	∝	0.08	0.14	3.67	2.15	8.71	14.57	7.20	4.96	33.61
S_3	∝	∝	∝	0.09	0.54	0.81	5.57	5.06	2.12	2.85	13.28
S_4	∝	∝	∝	∝	0.10	0.36	1.57	6.48	1.84	4.19	20.33
S_5	∝	∝	∝	∝	∝	0.09	∝	3.64	2.35	5.81	∝
S_6	∝	∝	∝	∝	∝	∝	0.35	0.81	1.32	1.86	2.59
S_7	∝	∝	∝	∝	∝	∝	∝	0.20	0.17	2.09	∝
S_8	∝	∝	∝	∝	∝	∝	∝	∝	0.07	0.93	3.73
S_9	∝	∝	∝	∝	∝	∝	∝	∝	∝	0.05	0.83
S_{10}	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝	0.21
S_{11}	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝

S_1 sampai dengan S_{11} merupakan kalimat ke-1 sampai ke-11. Nilai ∝ yang didapatkan pada beberapa kolom dalam tabel dikarenakan nilai $i \geq j$ seperti pada persamaan 2.4 atau tidak adanya kata-kata yang sama antara S_i dan S_j.

3.3.3 Hierarchical Clustering

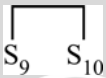
Proses selanjutnya adalah *hierarchical clustering*. Pada proses ini, yang di-*cluster* adalah bobot relasi dari masing-masing kalimat yang telah dihitung pada tabel 3.5. Langkah-langkah yang dilakukan seperti yang telah dipaparkan pada subbab 2.3 digambarkan dalam bentuk tabel.

Tabel 3.6 Langkah ke-0

i→j	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
S_1	∝	∝	0.93	∝	∝	6.72	∝	∝	9.41	9.42	41.49
S_2	∝	∝	0.08	0.14	3.67	2.15	8.71	14.57	7.20	4.96	33.61
S_3	∝	∝	∝	0.09	0.54	0.81	5.57	5.06	2.12	2.85	13.28
S_4	∝	∝	∝	∝	0.10	0.36	1.57	6.48	1.84	4.19	20.33
S_5	∝	∝	∝	∝	∝	0.09	∝	3.64	2.35	5.81	∝
S_6	∝	∝	∝	∝	∝	∝	0.35	0.81	1.32	1.86	2.59
S_7	∝	∝	∝	∝	∝	∝	∝	0.20	0.17	2.09	∝
S_8	∝	∝	∝	∝	∝	∝	∝	∝	0.07	0.93	3.73

S_9	∝	∝	∝	∝	∝	∝	∝	∝	∝	0.05	0.83
S_{10}	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝	0.21
S_{11}	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝

Pada langkah ke-0 (L_0) masing-masing kalimat dalam dokumen masih berdiri sendiri sehingga jumlah *cluster* =11. Kemudian dicari nilai yang paling kecil dalam tabel, yaitu 0.05 yang merupakan bobot relasi antara S_9 dan S_{10} ($R(S_9,S_{10})$). Sehingga pada langkah selanjutnya *cluster* yang terbentuk =10, karena S_9 dan S_{10} bergabung menjadi satu *cluster*. Gambar 3.12 merupakan pohon hirarki yang terbentuk pada L_0.



Gambar 3.12 Pohon hirarki L_0

Langkah selanjutnya, dilakukan iterasi berulang-ulang hingga terbentuk *cluster* tunggal atau nilai yang tersisa adalah ∝ sehingga penghitungan tidak dapat dilanjutkan lagi. Proses iterasi berturut-turut disajikan dalam tabel 3.7 – 3.14.

Tabel 3.7 Langkah ke-1

i→j	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_{910}	S_{11}
S_1	∝	∝	0.93	∝	∝	6.72	∝	∝	9.42	41.49
S_2	∝	∝	0.08	0.14	3.67	2.15	8.71	14.57	7.20	33.61
S_3	∝	∝	∝	0.09	0.54	0.81	5.57	5.06	2.85	13.28
S_4	∝	∝	∝	∝	0.10	0.36	1.57	6.48	4.19	20.33
S_5	∝	∝	∝	∝	∝	0.09	∝	3.64	5.81	∝
S_6	∝	∝	∝	∝	∝	∝	0.35	0.81	1.86	2.59
S_7	∝	∝	∝	∝	∝	∝	∝	0.20	2.09	∝
S_8	∝	∝	∝	∝	∝	∝	∝	∝	0.93	3.73
S_{910}	∝	∝	∝	∝	∝	∝	∝	∝	∝	0.83
S_{11}	∝	∝	∝	∝	∝	∝	∝	∝	∝	∝

Gambar 3.13 Pohon hirarki L_1

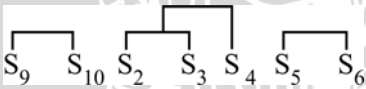
Tabel 3.8 Langkah ke-2

i→j	S_1	S_{23}	S_4	S_5	S_6	S_7	S_8	S_{910}	S_{11}
S_1	∝	∝	∝	∝	6.72	∝	∝	9.42	41.49
S_{23}	∝	∝	0.14	3.67	2.15	8.71	14.57	7.20	33.61
S_4	∝	∝	∝	0.10	0.36	1.57	6.48	4.19	20.33
S_5	∝	∝	∝	∝	0.09	∝	3.64	5.81	∝
S_6	∝	∝	∝	∝	∝	0.35	0.81	1.86	2.59
S_7	∝	∝	∝	∝	∝	∝	0.20	2.09	∝
S_8	∝	∝	∝	∝	∝	∝	∝	0.93	3.73
S_{910}	∝	∝	∝	∝	∝	∝	∝	∝	0.83
S_{11}	∝	∝	∝	∝	∝	∝	∝	∝	∝

Gambar 3.14 Pohon hirarki L_2

Tabel 3.9 Langkah ke-3

i→j	S_1	S_{23}	S_4	S_{56}	S_7	S_8	S_{910}	S_{11}
S_1	∝	∝	∝	∝	∝	∝	9.42	41.49
S_{23}	∝	∝	0.14	3.67	8.71	14.57	7.20	33.61
S_4	∝	∝	∝	0.36	1.57	6.48	4.19	20.33
S_{56}	∝	∝	∝	∝	∝	3.64	5.81	∝
S_7	∝	∝	∝	∝	∝	0.20	2.09	∝
S_8	∝	∝	∝	∝	∝	∝	0.93	3.73
S_{910}	∝	∝	∝	∝	∝	∝	∝	0.83
S_{11}	∝	∝	∝	∝	∝	∝	∝	∝



Gambar 3.15 Pohon hirarki L_3

Tabel 3.10 Langkah ke-4

i→j	S_1	S_{234}	S_{56}	S_7	S_8	S_{910}	S_{11}
S_1	∝	∝	∝	∝	∝	9.42	41.49
S_{234}	∝	∝	3.67	8.71	14.57	7.20	33.61
S_{56}	∝	∝	∝	∝	3.64	5.81	∝
S_7	∝	∝	∝	∝	0.20	2.09	∝
S_8	∝	∝	∝	∝	∝	0.93	3.73
S_{910}	∝	∝	∝	∝	∝	∝	0.83
S_{11}	∝	∝	∝	∝	∝	∝	∝

Gambar 3.16 Pohon hirarki L_4

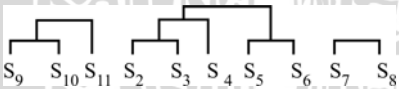
Tabel 3.11 Langkah ke-5

i→j	S_1	S_{234}	S_{56}	S_{78}	S_{910}	S_{11}
S_1	∝	∝	∝	∝	9.42	41.49
S_{234}	∝	∝	3.67	14.57	7.20	33.61
S_{56}	∝	∝	∝	∝	5.81	∝
S_{78}	∝	∝	∝	∝	2.09	∝
S_{910}	∝	∝	∝	∝	∝	0.83
S_{11}	∝	∝	∝	∝	∝	∝

Gambar 3.17 Pohon hirarki L_5

Tabel 3.12 Langkah ke-6

i→j	S_1	S_{234}	S_{56}	S_{78}	S_{91011}
S_1	∝	∝	∝	∝	41.49
S_{234}	∝	∝	3.67	14.57	33.61
S_{56}	∝	∝	∝	∝	∝
S_{78}	∝	∝	∝	∝	∝
S_{91011}	∝	∝	∝	∝	∝



Gambar 3.18 Pohon hirarki L_6

Tabel 3.13 Langkah ke-7

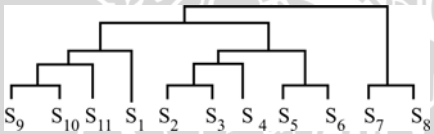
i→j	S_1	S_{23456}	S_{78}	S_{91011}
S_1	∝	∝	∝	41.49
S_{23456}	∝	∝	∝	∝
S_{78}	∝	∝	∝	∝
S_{91011}	∝	∝	∝	∝

Gambar 3.19 Pohon hirarki L_7

Tabel 3.14 Langkah ke-8

i→j	S_{23456}	S_{78}	S_{191011}
S_{23456}	∝	∝	∝
S_{78}	∝	∝	∝
S_{191011}	∝	∝	∝

Pada L_8 iterasi berhenti karena semua kolom yang tersisa bernilai ∝. Masing-masing pohon yang terbentuk merupakan satu paragraf dalam dokumen baru hasil penggabungan. Penyusunan paragraf disesuaikan dengan urutan terbentuknya pohon hirarki dari $L_0 - L_8$. Sehingga bila ketiga *cluster* yang terbentuk pada gambar 3.19 dijadikan *cluster* tunggal akan tampak seperti yang ditampilkan pada gambar 3.20.



Gambar 3.20 Pohon hirarki L_8

Dokumen baru hasil penggabungan ditampilkan pada gambar 3.21. Kalimat yang dicetak miring merupakan kalimat-kalimat yang berasal dari dokumen2 sedangkan yang dicetak biasa adalah kalimat-kalimat yang berasal dari dokumen1.

Samsung melalui inovasinya dengan menciptakan LCD super tipis dengan ketebalan hanya 0,74 mm (setara ketebalan kartu kredit), menjadikan peranti mobile mendatang lebih kompak, tipis, dan tahan goncangan. LCD hasil inovasi Samsung ini berhasil mengalahkan LCD keluaran LG yang terpaut hanya 0,07 mm lebih tipis, menjadikannya sebagai LCD tertipis di dunia. LCD dari Samsung ini akan mulai diproduksi secara massal sekitar pertengahan 2007. Inovasi tiada henti terus dipertunjukkan Samsung Electronics.

Kali ini perusahaan elektronik asal Korea Selatan tersebut memperkenalkan layar LCD yang diklaim tertipis di dunia. Ukuran ketebalan LCD 0,74 mm mematahkan rekor layar LCD tertipis yang sebelumnya juga dibuat Samsung, setebal 0,82 mm. Layar LCD super tipis bikinan perusahaan dari Negeri Ginseng itu tampaknya akan diaplikasikan pada ponsel Ultra Edition generasi mendatang. Sebagai perbandingan, ponsel ramping Ultra Edition saat ini memiliki ketebalan layar 1,9 mm. Apabila layar LCD tersebut nantinya telah diproduksi massal maka ponsel-ponsel Samsung akan meninggalkan para rivalnya untuk urusan desain ponsel teramping.

Tren peranti mobile mendatang rupanya mengarah kepada bentuk desain yang kompak, tipis, dan tahan goncangan. LCD yang ada di pasaran saat ini selain rentan akan goncangan, juga cenderung masih tebal sehingga dari segi desain kurang terlihat kompak.

Gambar 3.21 Hasil penggabungan dokumen

3.4 Perancangan Tabel

Dari perancangan proses yang telah dipaparkan, maka diperlukan beberapa file eksternal untuk menyimpan sejumlah data yang dibutuhkan. Dalam hal ini, file eksternal akan dibangun dalam bentuk tabel database. Adapun tabel-tabel tersebut ditunjukkan dalam Tabel 3.15, 3.16, dan 3.17

Tabel 3.15 Rancangan tabel frekuensi kata

Field	Type	Keterangan
id_kata	integer	Primary Key
id_dokumen	integer	dokumen tempat kata berada
kata	varchar(50)	kata yang muncul
frek	integer	frekuensi kemunculan kata

Tabel 3.15 menyimpan semua kata yang pernah muncul dalam sebuah dokumen setelah dilakukan *text pre-processing*. Frekuensi ini nantinya digunakan dalam proses pembobotan kalimat.

Tabel 3.16 Rancangan tabel posisi kata

Field	Type	Keterangan
id_kata	integer	
id_dokumen	integer	dokumen tempat kata berada
id_kalimat	integer	kalimat tempat kata berada
posisiKata	integer	letak kata terhadap kalimat

Tabel 3.16 menyimpan posisi letak kata terhadap suatu kalimat. Posisi kata akan digunakan dalam penghitungan bobot kalimat dan bobot relasi.

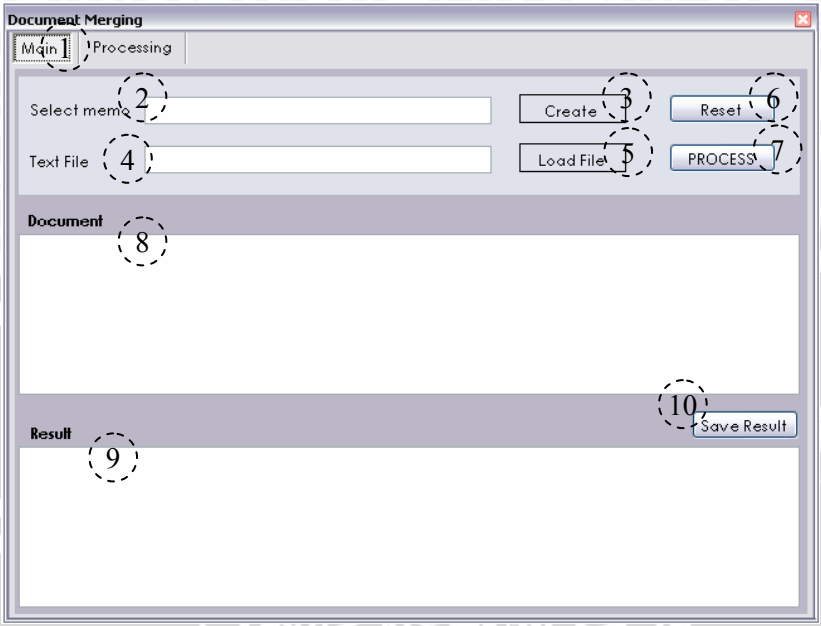
Tabel 3.17 Rancangan tabel kalimat utama

Field	Type	Keterangan
id_kalimat	integer	primary key
id_dokumen	integer	dokumen tempat kalimat berada

Tabel 3.17 menyimpan kalimat-kalimat yang merupakan kalimat utama.

3.5 Perancangan Antarmuka

Antarmuka yang akan dibangun dibagi menjadi dua bagian utama. Yaitu antarmuka untuk menampilkan masukan keluaran (input output) dan antarmuka untuk menampilkan proses dan analisis. Gambar 3.22 merupakan antarmuka input output.

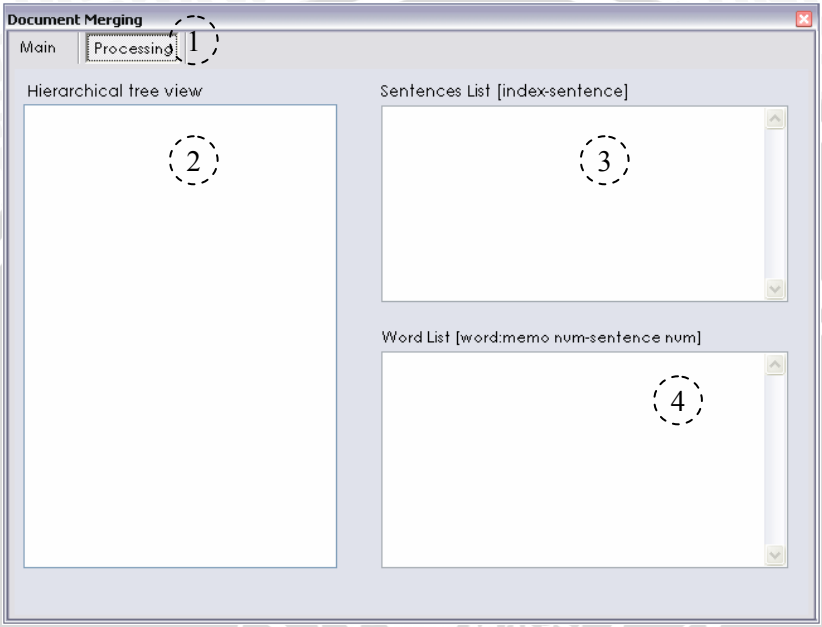


Gambar 3.22 Rancangan antarmuka input output

Adapun penjelasan dari bagian-bagian antarmuka input output adalah:

1. Main, halaman utama yang menampilkan masukan dan keluaran hasil dari penggabungan dokumen.
2. Select memo, digunakan untuk memilih dokumen mana yang akan ditampilkan pada memo document (8).
3. Create new, tombol yang berfungsi untuk menambahkan memo baru setiap kali akan menambah file dokumen.
4. Text file, menampilkan lokasi dan nama file yang digunakan sebagai dokumen sumber.
5. Load file, tombol yang berfungsi untuk mengambil file dari direktori penyimpanan.
6. Reset, tombol yang berfungsi untuk membersihkan antarmuka seperti pada saat pertama kali dijalankan.
7. Process, tombol yang berfungsi untuk memulai proses penggabungan dokumen.
8. Memo untuk menampilkan dokumen sumber disertai label yang menunjukkan index dokumen yang bersangkutan.
9. Memo untuk menampilkan dokumen hasil penggabungan.

10. Save result, tombol yang digunakan untuk menyimpan dokumen hasil.



Gambar 3.23 Rancangan antarmuka proses

Adapun penjelasan dari bagian-bagian antarmuka proses dan analisis adalah:

1. Processing, halaman yang menampilkan proses clustering.
2. Hierarchical tree view, menampilkan pohon hirarki hasil clustering.
3. Sentences list, menampilkan daftar kalimat-kalimat yang ada beserta index dan lokasi pada dokumen mana kalimat tersebut berada.
4. Word list, menampilkan daftar kata-kata yang muncul pada setiap dokumen disertai lokasi keberadaan kata terhadap dokumen dan kalimat.

3.6 Metode Pengujian

Pengujian yang dilakukan dalam penelitian ini dimaksudkan untuk mengetahui kualitas dokumen hasil penggabungan yang dihasilkan oleh sistem. Kualitas dokumen ditentukan dari nilai *recall*, *precision*, dan *F-measure*. Dalam penelitian ini, nilai-nilai tersebut diperoleh dari perbandingan jumlah kalimat yang dihitung melalui persamaan berikut ini:

$$Recall = \frac{\textit{jumlah kalimat hasil clustering yang relevan}}{\textit{jumlah total kalimat dokumen ideal}}$$

$$Precision = \frac{\textit{jumlah kalimat hasil clustering yang relevan}}{\textit{jumlah total kalimat hasil clustering}}$$

Dalam pengujian ini, masing-masing dokumen hasil dibandingkan dengan 1 dokumen ideal.

UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN UJI COBA

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengembangan sistem penggabungan dokumen ini adalah sebagai berikut :

1. Prosesor AMD Athlon 64 2.06 Ghz
2. Memori 1024 MB
3. Harddisk dengan kapasitas 80 GB
4. Monitor 15”
5. Keyboard
6. Mouse

4.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem penggabungan dokumen ini adalah :

1. Sistem Operasi Windows XP
2. Microsoft Visual Studio 2008
3. Microsoft SQL Server Compact 3.5 SP1
4. Text editor Notepad 2

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang terdapat pada subbab 3.2, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1 Implementasi text pre-processing

Tahap awal dari sistem clustering ini adalah penguraian kalimat. Penguraian kalimat dilakukan berdasarkan ada tidaknya delimiter. Delimiter merupakan karakter yang dikenali sebagai pembatas atau pemisah. Untuk itu, dalam proses penguraian kalimat terdapat fungsi untuk mengenali delimiter, baik itu delimiter kata, kalimat, maupun paragraf. Perbedaan masing-masing delimiter

tersebut diketahui dari nilai yang dikembalikan dari tiap-tiap iterasi yang dijalankan dalam fungsi `isDelimiter`, sehingga karakter yang ditemukan setelah delimiter akan dianggap sebagai kata, kalimat, atau paragraf baru.

```
private int isDelimiter(char c)
{
  //delimiter
  char[] arrDelimiterKata = { ' ', '-', '(', ')', '[', ']',
  '{', '}', '&', ':', '<', '>', '/', '+', ',', '\'', '\"',
  '$' };
  char[] arrDelimiterKalimat = { ';', '?', '!', '.', '\t' };
  char[] arrDelimiterParagraf = { '\n', '\0', '\r' };

  for (int i = 0; i < arrDelimiterKata.Length; i++)
     {
       if (c == arrDelimiterKata[i])
         return 1;
     }
  for (int i = 0; i < arrDelimiterKalimat.Length; i++)
    {
      if (c == arrDelimiterKalimat[i])
        return 2;
    }
  for (int i = 0; i < arrDelimiterParagraf.Length; i++)
    {
      if (c == arrDelimiterParagraf[i])
        return 3;
    }
      return 0;
```

Gambar 4.1 *source code* fungsi pencarian delimiter

a. Implementasi parsing

Proses parsing seperti yang telah dijelaskan pada subbab 3.2.1 adalah proses parsing sederhana yang memecah teks menjadi kumpulan kata. Karena pemecahan kata bertujuan salah satunya untuk mendapatkan frekuensi masing-masing kata, maka fungsi parsing ini diletakkan dalam prosedur untuk memasukkan kata beserta frekuensinya ke dalam Dictionary `frekuensiKata`, yaitu prosedur `Add`.

```
for (; i < n; i++)
{
  char c = teks[i];
  int delimiter = isDelimiter(c);
  if (delimiter > 0)
  {
      if (!string.IsNullOrEmpty(katatemp))
      {
          kalimattemp += katatemp;
          /*letak proses stemming*/
          //ganti kalimat
          if (delimiter >= 2)
          {
              DBTeks.TambahKalimatAsli(idDokumen,    idkalimat,
              kalimattemp);
          idkalimat++;
           //reset posisi kata
           posisikata = 0;
           //reset kalimattemp
           kalimattemp = "";
          }
          //ganti kata
          if (delimiter == 1)
          {
              posisikata++;
          }
      }
      katatemp = "";
  }
  else
  {
      katatemp += c;
  }
}
```

Gambar 4.2 *source code* fungsi parsing

b. Implementasi *stemming*

Proses *stemming* yang menggunakan `class IndoStemmer` dari penelitian (Daqiqil ID,2007) dijalankan di dalam fungsi parsing seperti yang digambarkan pada gambar 4.2. Sebelum dijalankan, terlebih dahulu dicek apakah suatu kata tersebut merupakan *stopword* melalui fungsi `isStopWord` yang terdapat dalam implementasi *stemming*. Jika fungsi ini mengembalikan nilai `true`, maka suatu kata dikenali sebagai *stopword* sehingga diabaikan dan tidak dikenai proses *stemming* karena stemming hanya dilakukan pada kata-kata yang bukan merupakan *stopword*.

```
private bool isStopWord(string kata)
{
     for (int i = 0; i < arrStopWord.Length; i++)
    {
         if (kata == arrStopWord[i])
       return true;
    }
    return false;
}
```

Gambar 4.3 *source code* fungsi pengecekan *stopword*

```
if (!isStopWord(katatemp))
  {
   string kata = ibnoestemmer.stem(katatemp.Trim().ToLower());
   DBTeks.TambahKata(idDokumen, idkalimat, kata, posisikata);
  }
```

Gambar 4.4 *source code* implementasi *stemming*

4.2.2 Implementasi pembobotan

Pembobotan dilakukan sebanyak lima kali. Kriteria masing-masing bobot, seperti yang dipaparkan pada sub bab 2.6, dalam sub bab ini diimplementasikan ke dalam bentuk coding dan dijelaskan pada poin a sampai dengan f.

a. Penghitungan bobot I

Bobot I (W1) dihitung berdasarkan kata kunci yang terdapat dalam suatu kalimat. Proses pertama adalah pencarian kalimat yang memiliki kata kunci, beserta jumlah kata kunci yang ada dalam kalimat tersebut. Banyaknya kata kunci dalam suatu kalimat merupakan bobot dari kalimat tersebut. Gambar 4.5 menunjukkan implementasi dari penghitungan bobot kalimat I.

```
private static void GetBobotKalimatI(ref
double[]arrayBobotS1,int iddokumen)
  {
    //cari idkalimat yang punya keyword, beserta jumlah
      keyword di kalimat itu
    string sql = "SELECT idkalimat,COUNT(*) AS jum FROM Teks
    inner JOIN kata ON Teks.idKata = kata.idKata
    WHERE(Teks.IDDokumen = @iddokumen) AND (kata.Frekuensi >
    @minkataforkeyword) GROUP BY Teks.IDKalimat";
```

```
            DataAccessParameter[] paramsx =
        {
            new DataAccessParameter("@iddokumen",iddokumen),
            new DataAccessParameter
            ("@minkataforkeyword",minkataforkeyword)
        };
        DataTable dt = new DataTable();
        DataAccess.FillDataTable(dt, sql, paramsx);
        if (dt.Rows != null && dt.Rows.Count != 0)
        {
            foreach (DataRow row in dt.Rows)
            {
                int idkalimat =
                Convert.ToInt32(row["idkalimat"].ToString());
                int jum = Convert.ToInt32(row["jum"].ToString());
                arrayBobotS1[idkalimat] = jum;
                //cari nilai bobot tertinggi, untuk normalisasi
                if (jum > maxS1)
                {
                    maxS1 = jum;
                }
            }
        }
    }
```

Gambar 4.5 source code penghitungan bobot kalimat W1

Pada prosedur `GetBobotKalimatI()` dihasilkan jumlah keyword pada masing-masing kalimat dan kemudian dicari nilai bobot tertinggi untuk penghitungan normalisasi.

b. Penghitungan Bobot II

```
private static int GetTotalKata(int iddokumen)
{
   string sql = "select count(*) from teks where iddokumen =
   @iddokumen";
   DataAccessParameter[] paramsx = {
         new DataAccessParameter("@iddokumen",iddokumen)
                                    };
   object o = DataAccess.ExecuteScalar(sql, paramsx);
   if (o == null)
      return 0;
   int c = 0;
   if (int.TryParse(o.ToString(), out c))
      return c;
   return 0;
}
```

Gambar 4.6 source code fungsi `GetTotalKata()`

Fungsi `GetTotalKata()`mengembalikan jumlah total kata dalam suatu dokumen berupa nilai integer. Fungsi ini diperlukan karena nilai bobot II (W2) ditentukan dari rasio kemunculan kata kunci di dalam kalimat terhadap pemakaian kata-kata tersebut pada dokumen tempat kalimat berada.

```
private  static  void GetBobotKalimatII(ref double[]
arrayBobotS2,int iddokumen, int start,int end)
  {
    int totalkata = GetTotalKata(iddokumen);
    List<Keyword> keywords = GetKeywords(iddokumen);
    //untuk setiap kalimat
    for (int kal = start; kal <= end; kal++)
    {
      double res = 0.000000;
      //foreach (Keyword key in keywords)
      for (int i = 0; i < keywords.Count;i++)
      {
        res = res + (double)(GetJumKataXDlmKalimat(kal,
        keywords[i].idkata) * keywords[i].frekuensi) /
        totalkata;
      }
      arrayBobotS2[kal] = res;
      //cari nilai bobot tertinggi
      if (res > maxS2)
      {
        maxS2 = res;
      }
    }
  }
```

Gambar 4.7 source code penghitungan bobot kalimat W2

c. Penghitungan Bobot III

Dalam prosedur penghitungan bobot III (W3), dipanggil fungsi `isKalimatUtama()`. Fungsi ini mengembalikan nilai boolean. Parameter `idkalimat` diambil dari tabel `kalimatutama`. Sehingga dari tabel tersebut dapat diketahui apakah suatu kalimat merupakan kalimat utama atau bukan.

```
private static bool IsKalimatUtama(int idkalimat)
{
   string sql = "select count(*) from kalimatutama where
   idkalimat=@idkalimat";
   DataAccessParameter[] paramsx = {
      new DataAccessParameter("@idkalimat",idkalimat)
                                        };
   object o = DataAccess.ExecuteScalar(sql, paramsx);
   if (o == null)
      return false;
   int c = 0;
   if (int.TryParse(o.ToString(), out c))
   {
      if (c == 1)
         return true;
   }
   return false;
}
```

Gambar 4.8 source code fungsi `isKalimatUtama()`

Jika suatu kalimat diketahui merupakan kalimat utama maka bobotnya sama dengan 2, sementara bila bukan kalimat utama,bobotnya sama dengan 1.

```
private static void GetBobotKalimatIII(ref double[]
arrayBobotS3, int start, int end)
{
  for (int kal = start; kal <= end; kal++)
  {
    if (IsKalimatUtama(kal))
    }
      arrayBobotS3[kal] = 2;
      maxS3 = 2;
    }
    else
    {
      arrayBobotS3[kal] = 1;
    }
  }
}
```

Gambar 4.9 source code penghitungan bobot kalimat W3

d. Penghitungan Bobot IV

Dalam penghitungan bobot IV (W4), dilihat ada atau tidaknya relasi antarkalimat berdasarkan ada atau tidaknya kata yang sama antara suatu kalimat dengan kalimat lainnya.

```
private static void GetBobotKalimatIV(ref double[]
arrayBobotS4, int start, int end)
{
    string sql = "SELECT distinct IDKalimat FROM Teks WHERE
    (IDKata IN (SELECT IDKata FROM Teks AS Teks_1 WHERE
    (IDKalimat = @idkalimat))) AND (IDKalimat <> @idkalimat)";
    for (int kal = start; kal <= end; kal++)
    {
      DataAccessParameter[] paramsx = {
              new DataAccessParameter("@idkalimat",kal)
                                    };
      DataTable tabletemp = new DataTable();
      DataAccess.FillDataTable(tabletemp, sql, paramsx);
      int count = tabletemp.Rows.Count;
      arrayBobotS4[kal] = count;
      if (count > maxS4)
      {
        maxS4 = count;
      }
    }
}
```

Gambar 4.10 source code penghitungan bobot kalimat W4

Bobot yang dimiliki masing-masing kalimat adalah jumlah relasi yang dimiliki suatu kalimat dengan kalimat yang lainnya.

e. Penghitungan Bobot V

Proses penghitungan bobot V (W5) hampir sama dengan penghitungan W1, yaitu ditentukan dari banyaknya kata kunci yang ada. Namun perbedaannya terletak pada kata kunci yang menjadi acuan. Dalam penghitungan W5, kata kunci bukan hanya dari satu dokumen yang bersangkutan melainkan dari semua dokumen yang ada.

```
private static void GetBobotKalimatV(ref double[]
arrayBobotS5, int start, int end)
{
  string sql = "SELECT COUNT(*) FROM Teks WHERE (IDKalimat =
  @idkalimat) AND (IDKata IN (SELECT IDKata FROM Kata WHERE
  (Frekuensi > @minkataforkeyword)))";
  for (int kal = start; kal <= end; kal++)
  {
    DataAccessParameter[] paramsx = {
            new
            DataAccessParameter("@minkataforkeyword",minkatafo
            rkeyword),
            new DataAccessParameter("@idkalimat",kal),
                                    };
    object o = DataAccess.ExecuteScalar(sql, paramsx);
    int bobot = 0;
    if (o == null)
      bobot = 0;
    int.TryParse(o.ToString(), out bobot);
    arrayBobotS5[kal] = bobot;
    if (bobot > maxS5)
      {
        maxS5 = bobot;
      }
  }
}
```

Gambar 4.11 source code penghitungan bobot kalimat W5

f. Penghitungan Bobot Relasi

Bobot relasi yang merupakan jarak antara dua buah kalimat diimplementasikan ke dalam *linked list*. Tahap pertama, terlebih dahulu dibuat kelas `BobotRelasi` yang mendeklarasikan variabel `nodes` dan `minedge`. Variabel `nodes` menyimpan kalimat sedangkan `minedge` menyimpan jarak terdekat yaitu bobot relasi yang terkecil.

```
public class BobotRelasi
    {
        public List<Node> nodes;
        public Node.Edge minedge;
    }
```

Gambar 4.12 source code pendeklarasian kelas `BobotRelasi`

Bobot relasi dihitung dalam fungsi `GetBobotRelasi()`. Langkah pertama adalah membuat array bobot sebanyak enam buah. Array pertama sampai kelima berturut-turut memuat bobot W1 sampai dengan W5. Sedangkan array keenam memuat bobot total kalimat, yaitu jumlah W1+W2+W3+W4+W5. Implementasinya disajikan dalam gambar 4.13 berikut.

```
public static BobotRelasi GetBobotRelasi()
{
   //index mulai dari 1
   int jumsemuakalimat = GetLastIDKalimat();

   //buat arraybobot sepanjang jumlah kalimat
   double[] arrayBobotS1 = new double[jumsemuakalimat +1];
   double[] arrayBobotS2 = new double[jumsemuakalimat + 1];
   double[] arrayBobotS3 = new double[jumsemuakalimat + 1];
   double[] arrayBobotS4 = new double[jumsemuakalimat + 1];
   double[] arrayBobotS5 = new double[jumsemuakalimat + 1];
   double[] arrayBobotSJ = new double[jumsemuakalimat + 1];

   //hitung bobot, dilakukan perdokumen
   string sql1 = "SELECT IDDokumen, MIN(IDKalimat) AS
   startindex, MAX(IDKalimat) AS endindex FROM Teks GROUP BY
   IDDokumen";
   DataTable dt = new DataTable();
   DataAccess.FillDataTable(dt, sql1, null);
   if (dt.Rows == null || dt.Rows.Count == 0)
   {
      MessageBox.Show("Tabel Teks kosong");
      return null;
   }
   foreach (DataRow row in dt.Rows)
   {
      int iddokumen =
          Convert.ToInt32(row["iddokumen"].ToString());
      int start =
          Convert.ToInt32(row["startindex"].ToString());
      int end = Convert.ToInt32(row["endindex"].ToString());
      GetBobotKalimatI(ref arrayBobotS1, iddokumen);
      GetBobotKalimatII(ref arrayBobotS2,iddokumen, start,
      end);
      GetBobotKalimatIII(ref arrayBobotS3,start,end);
      GetBobotKalimatIV(ref arrayBobotS4,start,end);
      GetBobotKalimatV(ref arrayBobotS5, start,end);
   }
```

Gambar 4.13 source code fungsi `GetBobotRelasi()`.

Langkah selanjutnya adalah menormalisasi semua bobot yang ada sebelum dijumlahkan ke dalam bobot total (Wj). Normalisasi bobot dilakukan dengan membagi bobot tiap-tiap

kalimat dengan nilai bobot terbesar. Bobot hasil normalisasi digunakan dalam penghitungan bobot relasi sesuai dengan persamaan 2.4 yang diimplementasikan dalam gambar 4.14. Fungsi ini mengembalikan nilai bertipe data BobotRelasi.

```
/*bobot relasi = node dihubungkan oleh edge. misal node i
dihubungkan edge dengan node j dengan bobot x*/
//i kombinasi j
BobotRelasi bobotrelasi = new BobotRelasi();
bobotrelasi.nodes = new List<Node>();
double minjarak = 999999;
Node[] nodes = new Node[jumsemuakalimat+1];

//set nama tiap node
for (int i = 1; i <=jumsemuakalimat; i++)
{
   nodes[i] = new Node(i.ToString());
   nodes[i].Name = i.ToString();
}
for (int i = 1; i <= jumsemuakalimat; i++)
{
   for (int j = i + 1; j <= jumsemuakalimat; j++)
   {
      //prevent devide by zero
      double devider = (GetJumKataSamaDi(i, j) *
      arrayBobotSJ[j]);
      if (devider > 0)
      {
         double bobot = (double)Math.Pow(j - 1, 2) / devider;
         Node.Edge edge = new Node.Edge(nodes[i], nodes[j],
         bobot);
         System.Console.WriteLine(nodes[i].Name + "->" +
         nodes[j].Name + " : " + bobot.ToString());
         nodes[i].Tetangga.Add(edge);
         //cari bobot terkecil
         if (bobot < minjarak)
            {
               minjarak = bobot;
               bobotrelasi.minedge = edge;
            }
       }
    }
    bobotrelasi.nodes.Add(nodes[i]);
 }
 return bobotrelasi;
}
```

Gambar 4.14 source code fungsi `GetBobotRelasi()`.

4.2.3 Implementasi Hierarchical Clustering

Proses *clustering* terdapat pada kelas `HClustering` dan diimplementasikan dalam fungsi `Process()`. Di dalam fungsi ini terdapat kelas `Node` yang merepresentasikan graf. Kelas `Node` memiliki variabel `name` yang berisi index kalimat misalnya s1,s2,s3, dan seterusnya. Sedangkan variabel `listTetangga` berupa array yang berisi index kalimat yang saling berhubungan. Dalam kelas ini ditambahkan kelas `Edge` yang merupakan jarak antara node, yaitu jarak suatu node dengan tetangganya. Berikut diberikan implementasi kelas `Node`.

```
public class Node
{
   public class Edge
   {
      public Node node1;
      public Node node2;
      public double jarak;
      public Edge(Node node1, Node node2, double jarak)
      {
         this.node1 = node1;
         this.node2 = node2;
         this.jarak = jarak;
      }
   }
   private string name;
   public string Name
   {
      get{return this.name;}
      set{this.name=value;}
   }
   private List<Edge> listTetangga;
   public List<Edge> Tetangga
   {
      get { return this.listTetangga; }
   }
   public Node(string name)
   {
      this.name = name;
      this.listTetangga = new List<Edge>();
   }
   public void AddTetangga(Edge e)
   {
      this.listTetangga.Add(e);
   }
}
```

Gambar 4.15 implementasi deklarasi kelas `Node`

Metode *complete linkage* yang digunakan dalam penelitian ini diimplementasikan pula dalam fungsi `Process()`seperti ditampilkan pada gambar 4.16.

```
//cari jarak max( complete linkage
  double jarak = Math.Max(minedge.node1.Tetangga[i].jarak,
                 minedge.node2.Tetangga[j].jarak);
  newnode.Tetangga.Add(new Node.Edge(newnode,
                 minedge.node1.Tetangga[i].node2, jarak));
  System.Console.WriteLine("add edge" + newnode.Name + "->" +
                 minedge.node1.Tetangga[i].node2.Name+ " " +
                 jarak.ToString());
```

Gambar 4.16 implementasi *complete linkage*

Implementasi metode *complete linkage* tersebut mencari jarak baru antara gabungan dua buah node dengan node lainnya. Fungsi yang digunakan adalah fungsi internal C# yaitu `Math.max` untuk mencari nilai maksimum.

Langkah selanjutnya yaitu mencari *node* baru yang terhubung dengan *node* terpendek. Pencarian dilakukan di semua *node* yang saling terhubung atau *node* yang menjadi tetangga .

4.3 Implementasi Tabel

Tabel yang digunakan dalam implementasi program ini sebanyak lima tabel, yaitu tabel GlobalKata, tabel KalimatAsli, tabel KalimatUtama, tabel Kata dan tabel Teks.

Tabel 4.1 Tabel GlobalKata

idGlobalKata	Kata	Frekuensi

Tabel GlobalKata menyimpan semua kata yang ada beserta frekuensinya yang muncul pada keseluruhan dokumen setelah dikenai proses stemming. `idkata` merupakan nilai integer yang di-random. Tabel ini digunakan pada proses penghitungan bobot kelima yang memerlukan kata kunci yang mengacu pada keseluruhan dokumen.

Tabel 4.2 Tabel KalimatAsli

idDokumen	idKalimat	TeksKalimat

Tabel KalimatAsli menyimpan kalimat ber-*index* pada gambar 3.5 dan 3.6, dengan `idKalimat` sebagai *index*-nya. Tabel ini digunakan untuk menampilkan kembali kalimat-kalimat yang utuh sesuai urutan dari hasil proses *clustering* yang telah selesai dijalankan.

Tabel 4.3 Tabel KalimatUtama

idKalimat

Tabel KalimatUtama menyimpan idkalimat yang menjadi kalimat utama, yaitu kalimat-kalimat yang terletak di awal dan akhir paragraf. Tabel ini digunakan untuk penghitungan bobot ketiga, sehingga kalimat yang tersimpan dalam tabel KalimatUtama akan mempunyai bobot (W3) = 2.

Tabel 4.4 Tabel Kata

idDokumen	Frekuensi	idGlobalKata

Tabel Kata menyimpan frekuensi kata pada masing-masing dokumen. Frekuensi yang ada pada tabel Kata digunakan untuk menentukan kata kunci pada masing-masing dokumen untuk penghitungan bobot pertama.

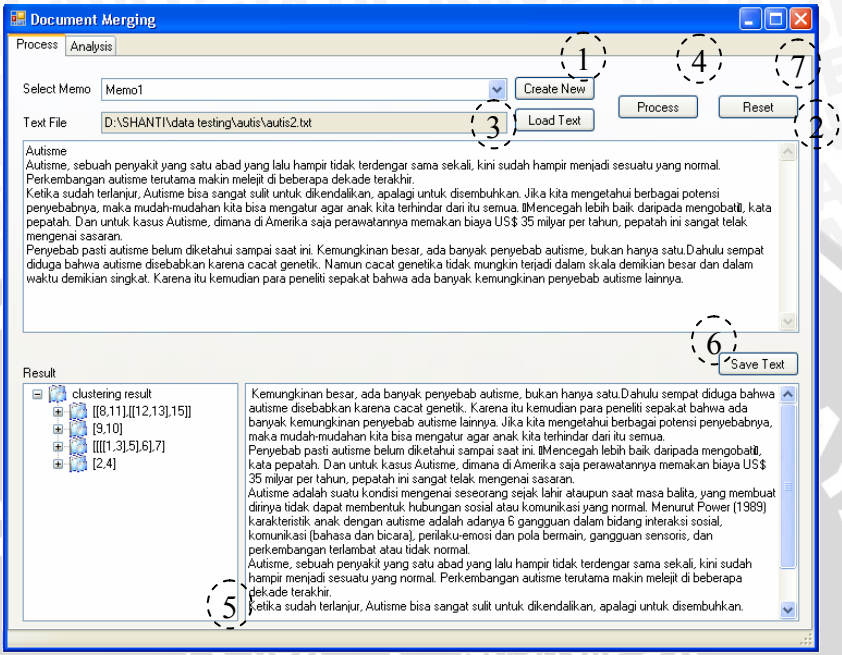
Tabel 4.5 Tabel Teks

idDokumen	idKalimat	idGlobalKata

Tabel Teks menyimpan letak kata pada suatu kalimat dan dokumen. Informasi tentang letak kata ini digunakan pada saat penghitungan bobot relasi yang memerlukan penghitungan jumlah kata yang sama pada antara suatu kalimat dengan kalimat lainnya

4.4 Implementasi Antarmuka

Dalam sistem ini terdapat dua halaman antarmuka yang diimplementasikan dalam satu buah form. Masing-masing halaman diimplementasikan dalam bentuk tabcontrol. Halaman pertama merupakan tahapan proses untuk pemrosesan penggabungan dokumen seperti tampak pada gambar 4.17.



Gambar 4.17 Implementasi antarmuka tahap proses

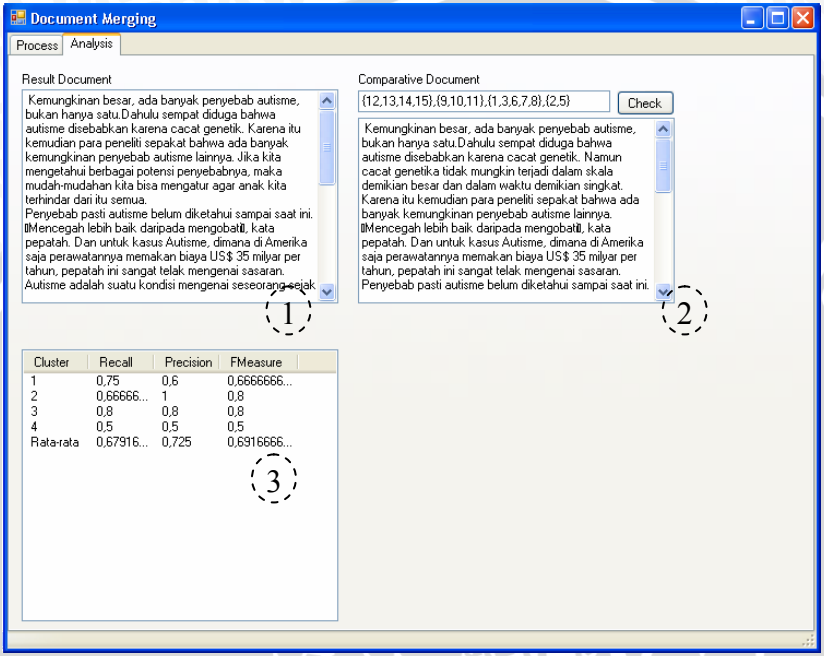
Pada antarmuka halaman pertama ini terdapat 7 bagian yang semuanya telah dijelaskan pada subbab 3.5.

File dokumen yang akan digabungkan diambil dari memori dengan menekan tombol ke-1 dan ke-2. Langkah ini diulang setiap kali akan mengeluarkan dokumen baru dari memori. File dokumen beserta lokasinya akan ditampilkan pada bagian ke-3. Setelah itu, untuk memulai proses penggabungan, ditekan tombol ke-4, maka hasil penggabungan ditampilkan pada bagian ke-5. Pada bagian ke-5 ini, hasil proses ditampilkan sebagai dokumen dan treeView.

Jangka waktu dimulainya proses sampai munculnya hasil bergantung pada panjang pendek dan banyaknya dokumen yang digabungkan dalam satu kali proses. Semakin banyak dokumen dan semakin panjang teks yang terdapat pada masing-masing dokumen, maka waktu yang dibutuhkan untuk proses akan semakin lama.

Bagian ke-6 adalah tombol yang berfungsi untuk menyimpan dokumen hasil yang telah ditampilkan dalam bentuk file berekstensi .txt. Bila pengguna ingin mengulang kembali proses penggabungan dengan contoh dokumen yang lain, maka pengguna dapat menekan

tombol Reset pada bagian ke-7 untuk mengembalikan tampilan form seperti semula dan mengulang proses dari langkah awal.



Gambar 4.18 Implementasi antarmuka tahap analisis

Pada antarmuka tahap analisis yang tampak pada gambar 4.18 terdapat 3 bagian. Bagian pertama menampilkan dokumen hasil sama seperti pada halaman proses. Sedangkan pada bagian ke-2 ditampilkan dokumen pembanding sebagai bahan pengevaluasi tingkat akurasi sistem. Tingkat akurasi sistem, yang dihitung sesuai dengan penghitungan yang telah dijelaskan pada subbab 2.6, ditampilkan pada bagian ke-3. Bagian ini menampilkan nilai *recall*, *precisiom*, dan *F-measure* dari masing-masing paragraf.

4.5 Uji Coba

4.5.1 Skenario Evaluasi

Pengevaluasian dilakukan dengan cara menghitung relevansi hasil *clustering* menggunakan standar ukuran evaluasi *recall*, *precision*, dan *f-measure*. Sebagai bahan pembanding, digunakan 1 dokumen ideal. *Cluster* yang diukur dalam penelitian ini adalah

paragraf dengan kalimat-kalimat sebagai anggota *cluster*-nya. Dengan demikian nilai *recall, precision*, dan *f-measure* untuk 1 dokumen hasil adalah rata-rata dari nilai keseluruhan *cluster-cluster*-nya.

Evaluasi dilakukan sebanyak 15 kali, di mana setiap kali proses penggabungan, jumlah dokumen sumber ditambah. Pada 5 percobaan pertama digunakan 2 dokumen sumber, pada 5 percobaan selanjutnya digunakan 3 dokumen sumber, dan pada 5 percobaan terakhir menggunakan 4 dokumen sumber.

4.5.2 Hasil Evaluasi

Berikut ini adalah tabel-tabel nilai *recall*, *precision*, dan *f-measure* dari uji coba yang telah dilakukan. Pada tabel 4.6 diberikan contoh penghitungan untuk 1 percobaan, yang dilakukan dengan 2 dokumen sumber pada dokumen yang memiliki topic tentang "Autisme".

Tabel 4.6 Tabel hasil evaluasi 1 dokumen hasil

Cluster	Recall	Precision	F-Measure
ke-1	0.75	0.6	0.667
ke-2	0.667	1	0.8
ke-3	0.8	0.8	0.8
ke-4	0.5	0.5	0.5
Rata-rata	0.679	0.725	0.692

Hasil uji coba penggabungan 2 dokumen ditunjukkan pada tabel 4.7. Hasil uji coba penggabungan 3 dokumen ditunjukkan pada tabel 4.8. Hasil uji coba penggabungan 4 dokumen ditunjukkan pada tabel 4.9. Hasil statistik berupa rata-rata, nilai minimum dan maksimum dari *recall*, *precision*, dan *f-measure* ditunjukkan pada tabel 4.10.

Tabel 4.7 Tabel hasil evaluasi dengan 2 dokumen sumber

Topik	Recall	Precision	F-Measure
Autisme	0.646	0.692	0.668
Penyebab autisme	0. 674	0. 721	0. 698
Musik Renaissance	0.626	0.662	0.633
Gigi Sensitif	0.612	0.685	0.642
Pengaruh Musik Klasik	0.616	0.673	0.658
Rata-rata	0.656	0.624	0.639

Tabel 4.8 Tabel hasil evaluasi dengan 3 dokumen sumber

Topik	Recall	Precision	F-Measure
Autisme	0.742	0.723	0.732
Penyebab autisme	0.734	0.742	0.736
Musik Renaissance	0.787	0.766	0.776
Gigi Sensitif	0.719	0.728	0.721
Pengaruh Musik Klasik	0.742	0.723	0.732
Rata-rata	0.752	0.750	0.749

Tabel 4.9 Tabel hasil evaluasi dengan 4 dokumen sumber

Topik	Recall	Precision	F-Measure
Autisme	0.841	0.751	0.793
Penyebab autisme	0.786	0.761	0.773
Musik Renaissance	0.792	0.783	0.787
Gigi Sensitif	0.841	0.796	0.820
Pengaruh Musik Klasik	0.765	0.792	0.764
Rata-rata	0.812	0.791	0.801

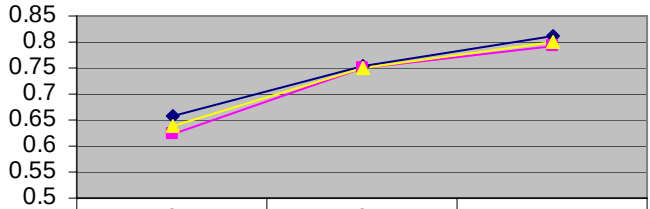
Tabel 4.10 Tabel hasil statistik deskriptif sistem

	Recall	Precision	F-Measure
Nilai minimum	0.656	0.624	0.639
Nilai maksimum	0.812	0.791	0.801
Rata-rata	0.740	0.722	0.729

4.5.3 Analisis Evaluasi

Dari hasil evaluasi relevansi di atas, didapatkan nilai rata-rata *recall*, *precision*, dan *f-measure* yang berbeda untuk jumlah dokumen yang berbeda. Hubungan antara banyak dokumen dengan keakuratan hasil *clustering* adalah berbanding lurus dimana semakin banyak dokumen maka keakuratan hasil *clustering* akan semakin meningkat (ditunjukkan pada gambar 4.19).

Hasil ini dipengaruhi pula oleh keakuratan yang dihasilkan pada saat proses *stemming.* Kebenaran kata-kata yang dihasilkan dari proses *stemming* tersebut dapat meningkatkan keakuratan hasil *clustering.*



	2	3	4
recall	0.656	0.752	0.812
precision	0.624	0.75	0.791
f-measure	0.639	0.749	0.801

Banyaknya dokumen

recall precision f-measure

Gambar 4.19 Grafik rata-rata evaluasi recall, precision, dan f-measure

UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapat setelah pengerjaan Skripsi ini antara lain:

1. Hasil *clustering* menggunakan metode *complete linkage* menghasilkan rata-rata *recall* sebesar 0.74 , rata-rata *precision* sebesar 0.722, dan rata-rata *f-measure* sebesar 0.729.
2. Efektifitas sistem semakin meningkat dengan bertambahnya dokumen sumber yang digabungkan. Terjadi peningkatan rata-rata terhadap nilai *recall* 14%, peningkatan nilai *precision* 4%, dan peningkatan nilai *F*-measure sebesar 9,5%.

5.2 Saran

Saran yang dapat diberikan setelah pengerjaan skripsi ini antara lain:

1. Perbaikan sistem *stemming* dapat lebih meningkatkan efektifitas sistem penggabungan dokumen.
2. Dapat dikembangkan dengan menambahkan algoritma untuk mencari hubungan keterkaitan antar kalimat.

UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Barakbah, A.R. 2006. *Clustering*. Workshop Data Mining Jurusan Teknologi Informasi Politeknik Elektronika Negeri Surabaya, 18-20 Juli 2006.

Borgatti, S.P. 1994. *How to Explain Hierarchical Clustering*. University of South Carolina.

Budhi, G.S., Arlinah I. R. dan Taufik, H. 2008. *Hierarchical Clustering untuk Aplikasi Automated Text Integration.* Seminar Nasional Aplikasi Teknologi Informasi 2008.

Budhi, G.S., Intan, R., Silvia R. dan Stevanus R.R. 2007. *Indonesian Automated Text Summarization.* Proceeding ICSIIT, 26-27 Juli 2007

Daqiqil ID. 2007. *Perancangan dan Penerapan Algoritma Stemming Kata Berbahasa Indonesia.* Universitas Brawijaya, Malang.

Departemen Pendidikan dan Kebudayaan Republik Indonesia. 1987. *Pedoman Umum Ejaan Bahasa Indonesia yang Disempurnakan*. Pustaka Setia.

Jones, K.S dan Galliers, JR. 1996. *Evaluating Natural Language Processing System: An Analysis and Review*. New York: Springer.

Sjobergh, J. dan Araki K. 2005. *Extraction Based Summarization Using a Shortest Path Algorithm.* KTH Nada. Swedia.

Steinbach, M., Karypis, G., Kumar, V. *A Comparison of Document Clustering Techniques*. 2000. Department of Computer Science and Egineering, University of Minnesota.

Supranto, J.M.A. 1992. *Teknik Sampling untuk Survey dan Eksperimen.* Rineka Cipta. Jakarta.

Tala, F. Z. 2003. *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Master of Logic Project, Institute for

Logic, Language and Computation. Universiteit van Amsterdam. The Netherlands.

Tarigan, H. G. 1995. *Pengajaran Morfologi*. Angkasa. Bandung.

Yang, Y. dan Liu, X. 1999. *A re-examinationof text categorization methods.* In Proceedings of ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR), pp 42-49. 1999. http://www.cs.cmu.edu/~yiming/publications.html

LAMPIRAN

Lampiran 1: Daftar Stopword

1	ada	36	awalnya	71	benarlah
2	adalah	37	bagai	72	berada
3	adanya	38	bagaikan	73	berakhir
4	adapun	39	bagaimana	74	berakhirlah
5	agak	40	bagaimanakah	75	berakhirnya
6	agaknya	41	bagaimanapun	76	berapa
7	agar	42	bagi	77	berapakah
8	akan	43	bagian	78	berapalah
9	akankah	44	bahkan	79	berapapun
10	akhir	45	bahwa	80	berarti
11	akhiri	46	bahwasanya	81	berawal
12	akhirnya	47	baik	82	berbagai
13	aku	48	bakal	83	berdatangan
14	akulah	49	bakalan	84	beri
15	amat	50	balik	85	berikan
16	amatlah	51	banyak	86	berikut
17	anda	52	bapak	87	berikutnya
18	andalah	53	baru	88	berjumlah
19	antar	54	bawah	89	berkali
20	antara	55	beberapa	90	berkata
21	antaranya	56	begini	91	berkehendak
22	apa	57	beginian	92	berkeinginan
23	apaan	58	beginikah	93	berkenaan
24	apabila	59	beginilah	94	berlainan
25	apakah	60	begitu	95	berlalu
26	apalagi	61	begitukah	96	berlangsung
27	apatah	62	begitulah	97	berlebihan
28	artinya	63	begitupun	98	bermaksud
29	asal	64	bekerja	99	bermula
30	asalkan	65	belakang	100	bersama
31	atas	66	belakangan	101	bersiap
32	atau	67	belum	102	bertanya
33	ataukah	68	belumlah	103	berturut
34	ataupun	69	benar	104	bertutur
35	awal	70	benarkah	105	berujar

106	berupa	141	demikian	176	dilalui
107	besar	142	demikianlah	177	dilihat
108	betul	143	dengan	178	dimaksud
109	betulkah	144	depan	179	dimaksudkan
110	biasa	145	di	180	dimaksudkannya
111	biasanya	146	dia	181	dimaksudnya
112	bila	147	diakhiri	182	diminta
113	bilakah	148	diakhirinya	183	dimintai
114	bisa	149	dialah	184	dimisalkan
115	bisakah	150	diantara	185	dimulai
116	boleh	151	diantaranya	186	dimulailah
117	bolehkah	152	diberi	187	dimulainya
118	bolehlah	153	diberikan	188	dimungkinkan
119	buat	154	diberikannya	189	dini
120	bukan	155	dibuat	190	dipastikan
121	bukankah	156	dibuatnya	191	diperbuat
122	bukanlah	157	didapat	192	diperbuatnya
123	bukannya	158	didatangkan	193	dipergunakan
124	bulan	159	digunakan	194	diperkirakan
125	bung	160	diibaratkan	195	diperlihatkan
126	cara	161	diibaratkannya	196	diperlukan
127	caranya	162	diingat	197	diperlukannya
128	cukup	163	diingatkan	198	dipersoalkan
129	cukupkah	164	diinginkan	199	dipertanyakan
130	cukuplah	165	dijawab	200	dipunyai
131	cuma	166	dijelaskan	201	diri
132	dahulu	167	dijelaskannya	202	dirinya
133	dalam	168	dikarenakan	203	disampaikan
134	dan	169	dikatakan	204	disebut
135	dapat	170	dikatakannya	205	disebutkan
136	dari	171	dikerjakan	206	disebutkannya
137	daripada	172	diketahui	207	disini
138	datang	173	diketahuinya	208	disinilah
139	dekat	174	dikira	209	ditambahkan
140	demi	175	dilakukan	210	ditandaskan

211	ditanya	246	hendaknya	281	juga
212	ditanyai	247	hingga	282	jumlah
213	ditanyakan	248	ia	283	jumlahnya
214	ditegaskan	249	ialah	284	justru
215	ditujukan	250	ibarat	285	kala
216	ditunjuk	251	ibaratkan	286	kalau
217	ditunjuki	252	ibaratnya	287	kalaulah
218	ditunjukkan	253	ibu	288	kalaupun
219	ditunjukkannya	254	ikut	289	kalian
220	ditunjuknya	255	ingat	290	kami
221	dituturkan	256	ingin	291	kamilah
222	dituturkannya	257	inginkah	292	kamu
223	diucapkan	258	inginkan	293	kamulah
224	diucapkannya	259	ini	294	kan
225	diungkapkan	260	inikah	295	kapan
226	dong	261	inilah	296	kapankah
227	dua	262	itu	297	kapanpun
228	dulu	263	itukah	298	karena
229	empat	264	itulah	299	karenanya
230	enggak	265	jadi	300	kasus
231	enggaknya	266	jadilah	301	kata
232	entah	267	jadinya	302	katakan
233	entahlah	268	jangan	303	katakanlah
234	guna	269	jangankan	304	katanya
235	gunakan	270	janganlah	305	ke
236	hal	271	jauh	306	keadaan
237	hampir	272	jawab	307	kebetulan
238	hanya	273	jawaban	308	kecil
239	hanyalah	274	jawabnya	309	kedua
240	hari	275	jelas	310	keduanya
241	harus	276	jelaskan	311	keinginan
242	haruslah	277	jelaslah	312	kelamaan
243	harusnya	278	jelasnya	313	kelihatan
244	hendak	279	jika	314	kelihatannya
245	hendaklah	280	jikalau	315	kelima

316	keluar	351	luar	386	memperbuat
317	kembali	352	macam	387	mempergunakan
318	kemudian	353	maka	388	memperkirakan
319	kemungkinan	354	makanya	389	memperlihatkan
320	kemungkinannya	355	makin	390	mempersiapkan
321	kenapa	356	malah	391	mempersoalkan
322	kepada	357	malahan	392	mempertanyakan
323	kepadanya	358	mampu	393	mempunyai
324	kesampaian	359	mampukah	394	memulai
325	keseluruhan	360	mana	395	memungkinkan
326	keseluruhannya	361	manakala	396	menaiki
327	keterlaluan	362	manalagi	397	menambahkan
328	ketika	363	masa	398	menandaskan
329	khususnya	364	masalah	399	menanti
330	kini	365	masalahnya	400	menantikan
331	kinilah	366	masih	401	menanya
332	kira	367	masihkah	402	menanyai
333	kiranya	368	masing	403	menanyakan
334	kita	369	mau	404	mendapat
335	kitalah	370	maupun	405	mendapatkan
336	kok	371	melainkan	406	mendatang
337	kurang	372	melakukan	407	mendatangi
338	lagi	373	melalui	408	mendatangkan
339	lagian	374	melihat	409	menegaskan
340	lah	375	melihatnya	410	mengakhiri
341	lain	376	memang	411	mengapa
342	lainnya	377	memastikan	412	mengatakan
343	lalu	378	memberi	413	mengatakannya
344	lama	379	memberikan	414	mengenai
345	lamanya	380	membuat	415	mengerjakan
346	lanjut	381	memerlukan	416	mengetahui
347	lanjutnya	382	memihak	417	menggunakan
348	lebih	383	meminta	418	menghendaki
349	lewat	384	memintakan	419	mengibaratkan
350	lima	385	memisalkan	420	mengibaratkannya

421	mengingat	456	misalnya	491	pertama
422	mengingatkan	457	mula	492	pertanyaan
423	menginginkan	458	mulai	493	pertanyakan
424	mengira	459	mulailah	494	pihak
425	mengucapkan	460	mulanya	495	pihaknya
426	mengucapkannya	461	mungkin	496	pukul
427	mengungkapkan	462	mungkinkah	497	pula
428	menjadi	463	nah	498	pun
429	menjawab	464	naik	499	punya
430	menjelaskan	465	namun	500	rasa
431	menuju	466	nanti	501	rasanya
432	menunjuk	467	nantinya	502	rata
433	menunjuki	468	nyaris	503	rupanya
434	menunjukkan	469	nyatanya	504	saat
435	menunjuknya	470	oleh	505	saatnya
436	menurut	471	olehnya	506	saja
437	menuturkan	472	pada	507	sajalah
438	menyampaikan	473	padahal	508	saling
439	menyangkut	474	padanya	509	sama
440	menyatakan	475	pak	510	sambil
441	menyebutkan	476	paling	511	sampai
442	menyeluruh	477	panjang	512	sampaikan
443	menyiapkan	478	pantas	513	sana
444	merasa	479	para	514	sangat
445	mereka	480	pasti	515	sangatlah
446	merekalah	481	pastilah	516	satu
447	merupakan	482	penting	517	saya
448	meski	483	pentingnya	518	sayalah
449	meskipun	484	per	519	se
450	meyakini	485	percuma	520	sebab
451	meyakinkan	486	perlu	521	sebabnya
452	minta	487	perlukah	522	sebagai
453	mirip	488	perlunya	523	sebagaimana
454	misal	489	pernah	524	sebagainya
455	misalkan	490	persoalan	525	sebagian

526	sebaik	561	sekadar	596	semua
527	sebaiknya	562	sekadarnya	597	semuanya
528	sebaliknya	563	sekali	598	semula
529	sebanyak	564	sekalian	599	sendiri
530	sebegini	565	sekaligus	600	sendirian
531	sebegitu	566	sekalipun	601	sendirinya
532	sebelum	567	sekarang	602	seolah
533	sebelumnya	568	sekecil	603	seolah-olah
534	sebenarnya	569	seketika	604	seorang
535	seberapa	570	sekiranya	605	sepanjang
536	sebesar	571	sekitar	606	sepantasnya
537	sebetulnya	572	sekitarnya	607	sepantasnyalah
538	sebisanya	573	sekurangnya	608	seperlunya
539	sebuah	574	sela	609	seperti
540	sebut	575	selain	610	sepertinya
541	sebutlah	576	selaku	611	sepihak
542	sebutnya	577	selalu	612	sering
543	secara	578	selama	613	seringnya
544	secukupnya	579	selamanya	614	serta
545	sedang	580	selanjutnya	615	serupa
546	sedangkan	581	seluruh	616	sesaat
547	sedemikian	582	seluruhnya	617	sesama
548	sedikit	583	semacam	618	sesampai
549	sedikitnya	584	semakin	619	sesegera
550	seenaknya	585	semampu	620	sesekali
551	segala	586	semampunya	621	seseorang
552	segalanya	587	semasa	622	sesuatu
553	segera	588	semasih	623	sesuatunya
554	seharusnya	589	semata	624	sesudah
555	sehingga	590	semata-mata	625	sesudahnya
556	seingat	591	semaunya	626	setelah
557	sejak	592	sementara	627	setempat
558	sejauh	593	semisal	628	setengah
559	sejenak	594	semisalnya	629	seterusnya
560	sejumlah	595	sempat	630	setiap

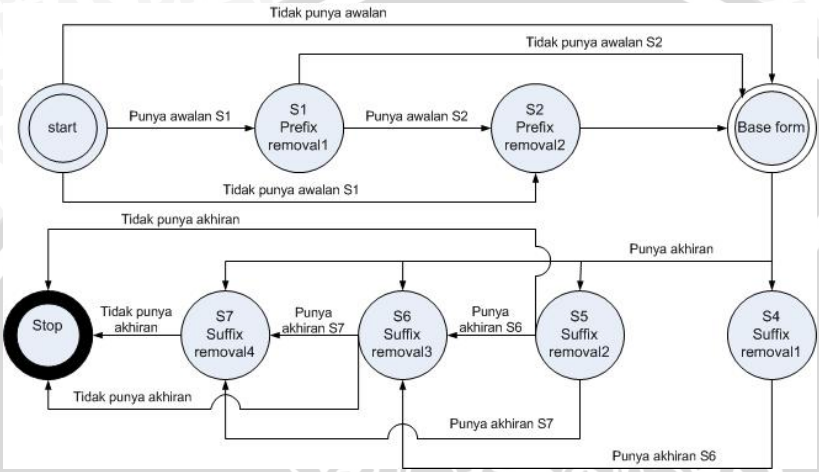
631	setiba	667	tegasnya	703	tiap
632	setibanya	668	telah	704	tiba
633	setidaknya	669	tempat	705	tidak
634	setinggi	670	tengah	706	tidakkah
635	seusai	671	tentang	707	tidaklah
636	sewaktu	672	tentu	708	tiga
637	siap	673	tentulah	709	tinggi
638	siapa	674	tentunya	710	toh
639	siapakah	675	tepat	711	tunjuk
640	siapapun	676	terakhir	712	turut
641	sini	677	terasa	713	tutur
642	sinilah	678	terbanyak	714	tuturnya
643	soal	679	terdahulu	715	ucap
644	soalnya	680	terdapat	716	ucapnya
645	suatu	681	terdiri	717	ujar
646	sudah	682	terhadap	718	ujarnya
647	sudahkah	683	terhadapnya	719	umum
648	sudahlah	684	teringat	720	umumnya
649	supaya	685	terjadi	721	ungkap
650	tadi	686	terjadilah	722	ungkapnya
651	tadinya	687	terjadinya	723	untuk
652	tahu	688	terkira	724	usah
653	tahun	689	terlalu	725	usai
654	tak	690	terlebih	726	waduh
655	tambah	691	terlihat	727	wah
656	tambahnya	692	termasuk	728	wahai
657	tampak	693	ternyata	729	waktu
658	tampaknya	694	tersampaikan	730	waktunya
659	tandas	695	tersebut	731	walau
660	tandasnya	696	tersebutlah	732	walaupun
661	tanpa	697	tertentu	733	wong
662	tanya	698	tertuju	734	yaitu
663	tanyakan	699	terus	735	yakin
664	tanyanya	700	terutama	736	yakni
665	tapi	701	tetap	737	yang
666	tegas	702	tetapi		

Lampiran 2: Aturan derivasional untuk prefiks

Prefiks (alomorf)	Variasi (morf)	Aturan
meng	meng	+ vokal | k | g | h ... , contoh : ambil → mengambil ikat → mengikat hilang → menghilang
	meny	+ s... , contoh : sapu → menyapu sisir → menyisir
	mem	+ b | f | p... , contoh : beku → membeku fitnah → memfitnah pukul → memukul
	men	+ c | d | j | t… , contoh : cuci → mencuci darat → mendarat jual → menjual tukar → menukar
	me	+ l | m | n | r | y |w… , contoh : lintas → melintas makan → memakan nikah → menikah rusak → merusak wabah → mewabah yakin → meyakin(kan)
peng	peng	+ vokal | k | g | h... , contoh : ikat → pengikat urus → pengurus ganggu → penggangu halus → penghalus
	peny	+ s... , contoh : saring → penyaring
	pem	+ b | f | p… , contoh : baca → pembaca fitnah → pemfitnah pukul → pemukul

	pen	+ c | d | j | t… , contoh : cuci → pencuci datang → pendatang jual → penjual tukar → penukar
	pe	+ l | m | n | r | y |w… , contoh : lintas → pelintas makan → pemakan rusak → perusak warna → pewarna
ber	bel	+ ajar , contoh : ajar → belajar
	be	+ r | KVr... , contoh : rencana → berencana kerja → bekerja
	ber	+ seluruh huruf selain morf bel dan ber , contoh : tamu → bertamu
per	pel	+ ajar , contoh : ajar → pelajar
	pe	+ r | KVr... , contoh : ramal → peramal
	per	+ seluruh huruf selain morf pel dan per , contoh : kaya → perkaya
ter	te	+ r... , contoh : rasa → terasa
	ter	+ K | V… , dimana K≠r , contoh : atur → teratur

Lampiran 3: State Diagram Stemming



Keterangan:
S1 : awalan me-, di-, ter-, ke-
S2 : awalan ber-, per-
S4 : akhiran -an, -wan, -is, -isme, -wati, -at, -in, -asi, -isasi
S5 : akhiran -i, -kan
S6 : akhiran -mu, -ku, -nya
S7 : akhiran -lah, -kah

UNIVERSITAS BRAWIJAYA