SISTEM PERINGKAS DOKUMEN TEKS OTOMATIS BERBAHASA INDONESIA DENGAN METODE *LEXRANK : GRAPH-BASED SUMMARIZATION ALGORITHM*

HALAMAN JUDUL

SKRIPSI

Oleh:
WIDHY HAYUHARDHIKA NUGRAHA PUTRA
0510963052-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
2010


UNIVERSITAS BRAWIJAYA

SISTEM PERINGKAS DOKUMEN TEKS OTOMATIS BERBAHASA INDONESIA DENGAN METODE *LEXRANK : GRAPH-BASED SUMMARIZATION ALGORITHM*

HALAMAN JUDUL

SKRIPSI

Oleh:
WIDHY HAYUHARDHIKA NUGRAHA PUTRA
0510963052-96

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

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
2010


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

SISTEM PERINGKAS DOKUMEN TEKS OTOMATIS BERBAHASA INDONESIA DENGAN METODE LEXRANK : *GRAPH-BASED SUMMARIZATION ALGORITHM*

Oleh:
WIDHY HAYUHARDHIKA NUGRAHA PUTRA
0510963052-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 2 Februari 2010
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Bayu Rahayudi, ST., MT	**Achmad Ridok, Drs., M.Kom.**
NIP. 19740712 200604 1 001	**NIP. 19680807 199412 1 001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya
Ketua,

Dr. Agus Suryanto, MSc
NIP. 19690807 199412 1 001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama **:** Widhy Hayuhardhika Nugraha Putra
NIM **:** 0510963052
Jurusan **:** Matematika
Penulis skripsi berjudul **:** Sistem Peringkas Dokumen Teks Otomatis Berbahasa Indonesia Dengan Metode *Lexrank* : *Graph-Based Summarization Algorithm.*

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. **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, 2 Februari 2010
Yang menyatakan,

(Widhy Hayuhardhika Nugraha Putra)
NIM. 0510963052


UNIVERSITAS BRAWIJAYA

SISTEM PERINGKAS DOKUMEN TEKS OTOMATIS BERBAHASA INDONESIA DENGAN METODE *LEXRANK : GRAPH-BASED SUMMARIZATION ALGORITHM*

ABSTRAK

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan pada komputer. Terdapat dua pendekatan pada peringkasan teks, yaitu ekstraksi (*shallower approaches*) dan abstraksi (*deeper approaches*). Salah satu teknik peringkasan dengan pendekatan ekstraksi adalah LexRank. Metode ini mengekstrak kalimat topik dengan menghitung rangking matematis untuk tiap kalimat berdasarkan nilai *similarity* antarkalimat. Penelitian ini akan melakukan uji coba terhadap nilai *precision* dan *recall* berdasarkan beberapa nilai *treshold summarization* dan *treshold similarity* yang diberikan.

Pada penelitian ini dibuat implementasi metode LexRank dalam dua versi aplikasi, yang dibedakan oleh metode perangkingan kalimat topik, versi 1 melakukan perangkingan tiap kalimat terhadap seluruh kalimat dalam dokumen, dan versi 2 melakukan perangkingan hanya pada masing-masing paragraf. Setelah dilakukan beberapa kali uji coba terhadap dokumen yang telah dibuat ringkasannya oleh ahli, didapatkan hasil rata-rata *precision* dan *recall* pada aplikasi versi 1 sebesar 0,380 dan 0,500. Dan uji coba kedua menggunakan aplikasi versi 2 dihasilkan rata-rata *precision* sebesar 0,369 dan *recall* sebesar 0,549.

Penggunaan metode *stemming* yang akurat dan metode penilaian *similarity* sangat berpengaruh terhadap nilai *precision* dan *recall* sistem metode ini. Metode LexRank akan bekerja optimal pada dokumen yang kalimat topiknya mengandung banyak kata penting dari paragraf atau dokumen.

Kata Kunci : Text Mining, Peringkasan Dokumen, LexRank, *similarity*, *stemming*, TF-IDF, *vector space models*.

AUTOMATIC SUMMARIZATION SYSTEM FOR INDONESIAN TEXT DOCUMENT WITH LEXRANK METHOD : GRAPH-BASED SUMMARIZATION ALGORITHM

ABSTRACT

Automatic Text Summarization is making a shorter version of a text by using the application running on the computer. There are two approaches to the text Summarization, extraction (shallower approaches) and abstraction (deeper approaches). One of extraction approach Summarization technique is LexRank. This method extracts the topic sentence by calculating the mathematical ranking for each sentence based on similarity values beetwen sentences. This research will test the value of precision and recall based on some summarization treshold and similarity treshold.

In this research, implementation of LexRank method was performed on two versions of applications, which are distinguished by ranking method for topic sentences, version 1 ranks each sentence for all sentences in the document, and version 2 only ranks each sentence for its paragraph. After some trials on the document that was summarized by the experts, it obtains the results of average precision and recall for the application version 1 are 0,380 and 0,500. And the second test using the application version 2 obtains an average precision and recall for 0,369 and 0, 549.

The using of the accurate stemming and similarity assessment method is very influential on the value of precision and recall system. LexRank method will work optimally on the topic sentence document that contains many important words of the paragraph or document.

Keywords : Text Mining, peringkasan dokumen, LexRank, similarity, stemming, TF-IDF, vector space models.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, penulis masih dapat belajar dan mengerjakan skripsi yang berjudul "Sistem Peringkas Dokumen Teks Otomatis Berbahasa Indonesia Dengan Metode *Lexrank : Graph-Based Summarization Algorithm*".

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih kepada:

1. Bayu Rahayudi, ST., MT sebagai pembimbing I dan Achmad Ridok, Drs., M.Kom selaku pembimbing II. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
3. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya
4. Papa, Mama, Adik dan Eyang. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
5. Dinar Rani, Verly Ramadhani, Jayanti Utari, Mega Satya, Tresnaningtiyas, Eko Nugroho, Dharma Surya, M. Chandra Saputra, Arga Dinata,ST., Adhit, Siwi atas bantuan, dukungan, semangat dan doanya.
6. Sahabat-sahabat ilkomers angkatan 2005 dan seluruh warga Program Studi Ilmu Komputer Universitas Brawijaya.
7. Pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi skripsi ini untuk kelanjutan penelitian serupa di masa mendatang.

Penulis berharap semoga skripsi ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya, baik oleh Penulis selaku mahasiswa maupun pihak-pihak lain yang tertarik untuk menekuni pengembangan *Text Mining*.

Malang, 2 Februari 2010
Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam memahami dan mengerti isi yang terkandung dalam sebuah dokumen memerlukan beberapa metode untuk mengekstrak atau mengambil intisari dari dokumen tersebut. Salah satu caranya adalah dengan melakukan peringkasan terhadap dokumen tersebut. Peringkasan adalah sebuah cara yang dilakukan untuk mendapatkan informasi penting dari sebuah dokumen. Tujuan utama dari peringkasan adalah untuk mendapatkan poin utama dari versi asli, misalnya teks, film atau suatu kejadian untuk membantu pengguna untuk mendapatkan intisari dalam waktu singkat. Tentunya tidak mudah melakukan peringkasan dokumen jika jumlah atau ukuran dari dokumen tersebut tidak sedikit. Untuk menjawab tantangan peringkasan terhadap dokumen dalam jumlah besar, maka diperlukan sebuah metode peringkasan dokumen / teks secara otomatis (*automatic document summarization*) (Bondy & Murty, 1976).

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan pada komputer. Hasil peringkasan ini mengandung poin-poin penting dari teks asli (Gunes & Dragomir, 2004).

Riset peringkasan teks otomatis telah berlangsung dari tahun 1958 sampai sekarang. Berbagai metoda telah diterapkan dan masih terus dikembangkan oleh para peneliti di seluruh dunia. Berbagai penelitian yang telah dilakukan membuahkan langkah-langkah peringkasan teks otomatis, yaitu *topic identification* atau pengumpulan topik, *interpretation* atau penggabungan topik dan yang terakhir pembentukan ringkasan (Hovy, 2003).

Terdapat dua pendekatan pada peringkasan teks, yaitu ekstraksi (*shallower approaches*) dan abstraksi (*deeper approaches*). Pada teknik ekstraksi, sistem menyalin informasi yang dianggap paling penting dari teks asli menjadi ringkasan (sebagai contoh, klausa utama, kalimat utama, atau paragraf utama). Sedangkan teknik abstraksi melibatkan parafrase dari teks asli. Pada umumnya, abstraksi dapat meringkas teks lebih kuat daripada ekstraksi, tetapi sistemnya lebih sulit dikembangkan karena mengaplikasikan

teknologi *natural language generation* yang merupakan bahasan yang dikembangkan tersendiri.

Berdasarkan jumlah sumbernya, ringkasan teks dapat dihasilkan dari satu sumber (*single-document*) atau dari banyak sumber (*multi-document*). Suatu ringkasan dapat bersifat *general*, yaitu ringkasan yang berupaya mengambil sebanyak mungkin informasi umum yang mampu menggambarkan keseluruhan isi teks. Selain itu dapat juga informasi yang diambil untuk ringkasan berdasar pada *query* yang didefinisikan *user*.

Dalam skripsi dengan judul "Sistem Peringkas Dokumen Teks Otomatis Berbahasa Indonesia dengan Metoda *LexRank : graph-based summarization algorithm*" ini akan dibuat peringkasan dokumen menggunakan metoda *graph-based summarization algorithm.* Dalam *graph-based summarization algorithm*, sebuah teks direpresentasikan menjadi sebuah graf. *Vertex/node* pada graf tekstual dapat berupa kata-kata, kalimat-kalimat, atau paragraf-paragraf dalam teks. *Edge/link* dalam graf menunjukkan keterhubungan antar *vertex/node*. Keterhubungan dapat berupa *similarity* antarkalimat ataupun hubungan leksikal atau gramatikal antar kata/frasa. Pemilihan jenis unit teks untuk dijadikan *vertex* bergantung pada tujuan aplikasi yang akan dicapai. Misalnya untuk ekstraksi *keyphrase* biasanya frasa atau kata-kata menjadi *vertex*, sedangkan untuk ekstraksi ringkasan biasanya kalimat dipilih sebagai *vertex*.

Penelitian tentang metode peringkasan dokumen dengan menggunakan LexRank *graph-based summarization algorithm* ini pernah dilakukan dengan menggunakan algoritma *idf-modified-cosine* dalam menentukan *similarity* antarkalimatnya (Gunes & Dragomir, 2004). Penelitian yang lain juga pernah dilakukan oleh Indah Wayhuni (2007). Pada penelitiannya, Indah Wahyuni melakukan penelitian terhadap *Automated Text Summarization* dengan menggunakan metode *graph based.*

Pada penelitian ini digunakan metode perangkingan kalimat dengan memberikan skor berdasarkan *word overlapping* atau kata yang sama pada kalimat lainnya, *word overlapping* antara kalimat dengan judul, posisi kalimat pada paragraf, *cue phrase* atau ada tidaknya kalimat-kalimat kunci dan ada tidaknya *keyword* atau *query*. Dan untuk menentukan susunan hasil peringkasan, digunakan

metode pencarian jarak terpendek (*shortest path*) dengan algoritma Djikstra.

Sedangkan dalam skripsi ini, akan dilakukan perangkingan kalimat dengan metode *normalized* TF-IDF dan *vector space models* untuk mendapatkan nilai *similarity* antarkalimat. Tetapi, sebelum proses perangkingan, dilakukan proses *stemming* untuk mendapatkan frekuensi kata yang lebih akurat. Dan dalam menentukan bobot *verteks* untuk mendapatkan susunan hasil ringkasan, digunakan metode LexRank *graph-based summarization algorithm.*

1.2 Rumusan Masalah

Dari latar belakang yang telah disebutkan di atas, maka dapat ditarik rumusan masalah sebagai berikut.

1. Bagaimana membuat sebuah sistem yang dapat melakukan peringkasan dokumen secara otomatis dengan menggunakan algoritma LexRank : *graph based summarization algorithm*.
2. Bagaimanakah sistem yang dibuat membentuk ringkasan dokumen dari beberapa dokumen yang diujikan dengan tipe paragraf yang berbeda.

1.3 Tujuan

Tujuan dari penulisan skripsi ini adalah:

1. Pembuatan sebuah sistem yang dapat melakukan proses peringkasan dokumen teks secara otomatis dengan metode LexRank : *graph-based algorithm*.
2. Mengetahui apakah sistem yang dibuat dapat melakukan peringkasan dokumen terhadap beberapa dokumen berbahasa Indonesia yang diujikan dengan tipe paragraf yang berbeda.

1.4 Batasan Masalah

Pada penulisan skripsi ini, permasalahan hanya dibatasi pada pembuatan sistem peringkas dokumen teks otomatis dengan jenis dokumen tunggal (*single document*) dan berbahasa Indonesia.

Sistem akan dikembangkan dengan bahasa pemrograman Java dan diterapkan pada komputer personal (PC).

Pengujian dilakukan terhadap beberapa dokumen dengan tipe paragraf yang berbeda.

1.5 Manfaat

Manfaat yang dapat diambil dari penulisan skripsi ini adalah mendapatkan sistem yang dapat melakukan peringkasan dokumen teks secara otomatis untuk membantu ekstraksi informasi dalam dokumen.

1.6 Metode Penyelesaian Masalah

Metode penyelesaian masalah yang dilakukan pada penelitian ini, yaitu :

1. Studi Literatur

   Membaca dan mempelajari beberapa literatur (jurnal, buku dan artikel dari *website*) mengenai peringkasan teks otomatis, dan jenis-jenis paragraf dalam bahasa Indonesia.

2. Perancangan dan implementasi perangkat lunak

   Merancang dan membangun sebuah perangkat lunak berbasis Java yang mengimplementasikan proses peringkasan dokumen teks dengan menggunakan metode LexRank*:graph-based algorithm.*

3. Uji coba dan analisis hasil implementasi

   Menganalisis hasil implementasi, yaitu hasil-hasil ringkasan dari beberapa dokumen yang diujikan dengan tipe paragraf yang berbeda.

1.7 Sistematika penulisan

Skripsi ini akan menjabarkan keseluruhan penelitian yang dikelompokkan secara sistematis menjadi enam bab. Adapun pembagian bab-bab tersebut adalah:

BAB I PENDAHULUAN

Dalam bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metode penyelesaian masalah, manfaat, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Menjelaskan tentang dasar teori yang digunakan dalam menyusun skripsi. Hal tersebut, meliputi penjelasan Graf, kalimat dan paragraf, TF-IDF dan *vector space models similarity*, dan penjelasan mengenai algoritma LexRank.

BAB III METODOLIGI

Membahas Perancangan sistem, arsitektur sistem, diagram alur proses sistem, rancangan penelitian, dan contoh perhitungan manual.

BAB V IMPLEMENTASI DAN PEMBAHASAN

Bab ini membahas tentang lingkungan implementasi, implementasi sistem yang terbagi atas implementasi program dan implementasi antarmuka, serta analisa hasil uji coba.

BAB VI KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari pembahasan bab sebelumnya serta saran dari keseluruhan penelitian.


UNIVERSITAS BRAWIJAYA

BAB II

TINJAUAN PUSTAKA

2.1 Graf Matematis

Dalam matematika dan ilmu komputer, teori graf adalah struktur matematis yang digunakan untuk memodelkan hubungan antara obyek-obyek berpasangan dari koleksi tertentu. Sebuah "graf" dalam konteks ini mengacu pada koleksi *vertex* atau 'simpul' dan koleksi *edges* atau penghubung yang menghubungkan antara dua simpul atau lebih. Graf terdiri atas graf tidak terarah, yang berarti bahwa tidak ada perbedaan antara dua *vertex* yang terkait dengan setiap *edge*, atau graf terarah, yaitu graf yang dapat diarahkan dari satu *vertex* ke *vertex* yang lain.

Sumber lain mengatakan bahwa graf G mengandung tiga poin, yaitu V(G), E(G), dan μg dimana V(G) adalah set dari *vertex* atau simpul, E(G) adalah set dari *edge* atau penghubung yang memisahkan *vertex* V(G) dan fungsi μg yang menentukan masing-masing *edge* dari graf G dimana *edge-edge* tersebut menghubungkan *vertex* yang tidak beraturan (terdapat kemungkinan untuk sebuah *edge* menghubungkan *vertex* yang sama) (Bondy & Murty, 1976).

Definisi lain tentang graf juga disebutkan sebagai pasangan himpunan (V,E), ditulis dengan notasi G=(V,E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (*vertices* atau *nodes*) yang menghubungkan sepasang simpul (Munir, 2005). Jadi, sebuah graf dimungkinkan untuk tidak mempunyai sisi sama sekali tetapi simpulnya harus ada walaupun tanpa penghubung.

2.1.1 Jenis-jenis graf

Graf dapat dikelompokkan menjadi beberapa jenis bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang dari ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul atau berdasarkan orientasi arah pada sisi (Munir, 2005).

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis : (Munir, 2005)

1. **Graf sederhana** (*simple graph*)
   Yaitu graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana. Pada graf sederhana, sisi adalah pasangan tak-terurut (*unordered pairs*).
2. **Graf tak-sederhana** (*unsimple graph*)
   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu **graf ganda** (*multigraph*) yang mengandung sisi ganda dan **graf semu** (*pseudograph*) yang mengandung gelang (*loop*).

Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka graf dapat dibedakan menjadi dua jenis : (Munir, 2005)

1. **Graf tak-berarah** (*undirected graph*)
   Yaitu graf yang sisinya tidak mempunyai orientasi arah
2. **Graf berarah** (*directed graph*)
   Yaitu graf yang setiap sisinya diberikan orientasi arah.

Sebuah struktur graf bisa dikembangkan dengan memberi bobot pada tiap *edge*. Graf ini disebut dengan graf berbobot, yaitu graf yang setiap sisinya diberi sebuah harga (bobot) (Munir, 2005).

Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan jarak antara dua buah kota, biaya perjalanan antara dua kota, waktu tempuh pesan (*message*) dari sebuah simpul ke simpul yang lain (dalam jaringan komputer), ongkos produksi, dan sebagainya (Munir, 2005).

2.2 Komponen Pembentuk Dokumen Teks

2.2.1 Kalimat

Kalimat adalah satuan bahasa terkecil dalam wujud lisan atau tulisan yang sekurang-kurangnya terdiri atas subyek (S) dan predikat (P).

Subjek (S) adalah bagian kalimat yang menunjuk pelaku, yang menjadi pangkal / pokok pembicaraan. Subjek biasanya diisi oleh jenis kata/frasa benda (nomina), klausa, atau frasa verbal.

Predikat (P) adalah bagian kalimat yang memberi tahu melakukan (tindakan) apa atau dalam keadaan bagaimana S

(pelaku/tokoh atau benda dalam suatu kalimat). Selain memberi tahu tindakan atau perbuatan S, predikat dapat pula menyatakan sifat, situasi, status, ciri, atau jati diri S. termasuk juga sebagai P dalam kalimat adalah pernyataan tentang jumlah sesuatu yang dimiliki S. predikat dapat berupa kata atau frasa, sebagian besar berkelas verba atau adjektif. Tetapi dapat juga berupa numeralia, nomina, atau frasa nominal.

Obyek (O) adalah bagian kalimat yang melengkapi P. Obyek umumnya diisi oleh nomina, frasa nominal, atau klausa.
(Maimunah, 2007)

2.2.2 Paragraf

Paragraf disebut juga alinea. Kata paragraf diserap dari bahasa Inggris *paragraph*, sedangkan kata alinea dari bahasa Belanda dari kata latin *a linea* yang berarti "mulai dari garis baru". Paragraf adalah sebuah wacana mini atau satuan bentuk bahasa yang biasanya merupakan hasil penggabungan beberapa kalimat, artinya setiap unsur pada karangan panjang ada pada paragraf. Dalam paragraf kalimat-kalimat harus disusun dengan kohesi (kesatuan dalam paragraf), memiliki koherensi (keterpautan makna), dan memiliki isi yang memadai sebagai pendukung gagasan utama dalam paragraf.

Paragraf yang baik memiliki satu kalimat utama yang berisi tentang pokok pikiran paragraf atau gagasan dan beberapa kalimat penjelas yang merupakan uraian yang menjelaskan pokok pikiran.

Pikiran utama yaitu topik yang dikembangkan menjadi sebuah paragraf. Pikiran utama ini dinyatakan dalam kalimat topik. Dalam paragraf, pikiran utama berfungsi sebagai pengendali keseluruhan paragraf. Begitu menentukan pikiran utama dan mengekspresikannya dalam kalimat topik, penulis terikat oleh pikiran tersebut sampai akhir paragraf. Paragraf yang berisi analisis, klasifikasi, deduktif, induktif sebaiknya menggunakan kalimat topik. Namun harus disadari bahwa tidak semua paragraf harus menggunakan kalimat topik. Paragraf narasi dan deskripsi menggunakan kalimat yang sama kedudukannya, tidak ada yang lebih utama.

Ciri kalimat utama : pertama, mengandung permasalahan yang potensial untuk dirinci. Dan diuraikan lebih lanjut; kedua, merupakan kalimat yang dapat berdiri sendiri; ketiga, mempunyai

arti yang jelas tanpa harus disambungkan dengan kalimat lain; keempat, dapat dibentuk tanpa sambungan frase transisi (Maimunah, 2007).

2.2.3 Jenis Paragraf

Berdasarkan letak kalimat topiknya, paragraf dapat dibedakan menjadi :

a. **Paragraf Deduksi / Deduktif (kalimat topik di awal paragraf)**
   Kalimat topik pada awal paragraf pada umumnya berisi pikiran utama yang bersifat umum. Kalimat selanjutnya berisi pikiran penjelas yang bersifat khusus, disebut kalimat penjelas.
b. **Paragraf Induksi / Induktif (kalimat topik di akhir paragraf)**
   Paragraf diakhiri kalimat topik dan diawali dengan kalimat penjelas. Artinya paragraf ini menyajikan kasus khusus, contoh, penjelasan, keterangan, atau analisis terlebih dahulu, baru ditutup dengan kalimat topik.
c. **Paragraf Kombinasi (kalimat topik di awal dan akhir paragraf)**
   Kalimat topik dalam sebuah paragraf pada hakikatnya hanya satu. Penempatan kalimat topik yang kedua berfungsi untuk menegaskan kembali pikiran utama paragraf tersebut.
d. **Paragraf Penuh**
   Paragraf penuh maksudnya paragraf penuh dengan kalimat topik, seluruh kalimat sama pentingnya sehingga tidak satupun kalimat yang khusus menjadi kalimat topik.

(Maimunah, 2007)

2.3 *Text Mining*

Information Retrieval (Pengambilan Informasi) adalah kegiatan pencarian bahan (biasanya dokumen) dari lingkungan yang tak terstruktur (biasanya teks) untuk memenuhi informasi yang dibutuhkan dari sumber yang besar (biasanya komputer atau *server* atau di *internet*). *Information retrieval* banyak diterapkan pada beberapa kasus yang berhubungan dengan teks atau dokumen.
(Manning, Raghavan, & Schütze, 2007).

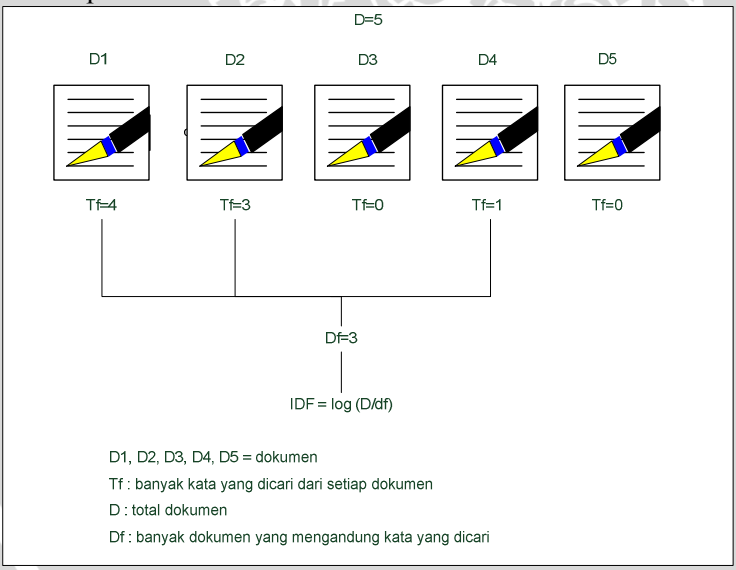
Text mining memiliki definisi menambang data yang berupa teks, dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar

dokumen. Tahapan secara umum dari proses *text mining* adalah *tokenizing, filtering, stemming, tegging,* dan *analyzing* (Manning, Raghavan, & Schütze, 2007)

Tokenizing adalah tahap pemotongan *string input* bersadarkan tiap kata yang menyusunnya. Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stop list* atau *stop word*. Tahap *stemming* adalah tahap mecari *root* (kata dasar) dari hasil filtering. Tahap *tagging* adalah tahap mencari bentuk awal dari tiap kata lampau hasil stemming. Tahap ini hanya digunakan untuk teks berbahasa Inggris, karena teks berbahasa Indonesia tidak memiliki bentuk lampau.

2.3.1 Algoritma TF/IDF

Salah satu algoritma *Text Minning* adalah algoritma TF/IDF. Algoritma TF/IDF digambarkan pada gambar 2.1 berikut. Algoritma ini dilakukan untuk mencari tingkat kemiripan sebuah dokumen terhadap sebuah kata kunci.



Gambar 2-1 Skema Algoritma TF/IDF

Dan formula yang digunakan untuk menghitung bobot (w) dari masing-masing dokumen terhadap kata / kalimat kunci adalah seperti pada persamaan 2.1.

$$w_{d,t} = tf_{d,t} * log \frac{D}{df_t} \quad \textbf{(2.1)}$$

Dimana :
d = dokumen ke – d
t = kata ke-t dari kata / kalimat kunci
$w_{d,t}$ = bobot dokumen ke – d terhadap kata ke – t
df_t = jumlah dokumen yang mengandung kata kunci ke-t

Setelah bobot (w) masing-masing dokumen diketahui, maka dilakukan proses *sorting*/pengurutan dimana semakin besar nilai w, semakin besar tingkat similiaritas dokumen tersebut terhadap kata yang dicari, demikian juga sebaliknya.

2.3.2 Vector Space Models

Hasil algoritma TF-IDF seringkali mendapatkan nilai bobot (w) yang sama untuk dua dokumen yang berbeda. Hal ini dikarenakan perhitungan bobot menggunakan TF-IDF berdasarkan perhitungan *query*. Sehingga *sorting* yang dilakukan untuk mendapatkan nilai *similarity* kurang akurat. Oleh karena itu diperlukan metode lain untuk mendapatkan nilai *similarity* antara kata/kalimat kunci terhadap sebuah dokumen, salah satunya dapat digunakan metode *vector space models*.

Dasar metode ini adalah menghitung nilai cosinus sudut dari dua vektor, yaitu bobot (w) dari setiap dokumen dan bobot (w) dari kata/kalimat kunci.

Dengan menggunakan *Vector space models* untuk menghitung nilai *similarity*, formula *similarity* yang dipakai adalah *similarity* tf-id ternormalisasi, yang dinyatakan dalam persamaan 2.2.

$$Sim(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}} \quad \textbf{(2.2)}$$

Dengan *Q* adalah *Query*, *D* adalah Dokumen, *i* adalah indeks dokumen, *w* adalah *weight* (bobot *term*) dan *j* adalah indeks *term*

Dan *weight (w)* untuk TF-IDF yang telah ternormalisasi dinyatakan dalam persamaan 2.3.

$$w_{Q,i} = \left\{\frac{tf_{Q,i}}{\max tf_{Q,i}}\right\} * log\left\{\frac{D}{df_i}\right\} \quad \textbf{(2.3)}$$

Dengan Q adalah *query*, *i* adalah indeks *term*, tf adalah *term frequency*, D adalah jumlah dokumen dan *dfi* adalah jumlah dokumen yang mengandung *term* ke-*i*.

Rumus *vector space models* ini nantinya akan digunakan sebagai rumus untuk mencari nilai *similarity* dari dua kalimat. Kalimat-kalimat akan direpresentasikan sebagai dokumen dan dihitung *similarity* dengan dokumen lain yang juga berupa kalimat.

2.3.3 Peringkasan Dokumen (*Summarization*)

Summarization atau ringkasan adalah sebuah teks yang dihasilkan dari kumpulan teks yang mengandung bagian penting dari informsi yang terdapat pada kumpulan teks asli, dan teks ini tidak lebih dari separuh teks aslinya (Hovy, 2003).

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan pada komputer. Hasil peringkasan ini mengandung poin-poin penting dari teks asli (Gunes & Dragomir, 2004).

Penelitian tentang peringkasan dokumen telah menghasilkan tiga metode peringkasan dokumen yang berbeda. Metode yang pertama adalah *topic identification* (Identifikasi Topik), merupakan tipe yang paling sederhana. Topik disini adalah persoalan khusus yang ditulis atau didiskusikan. Apapun tolok ukur kepentingan yang digunakan, apabila sistem telah menentukan unit-unit yang paling penting (kata-kata, kalimat, paragraf, dan sebagainya), maka kemudian dapat dengan mudah disusun (dengan membuat intisari) atau menampilkannya dalam bentuk diagram (dengan membuat ringkasan yang skematis). Ciri khas metode *topic identificatio*n ini memiliki beberapa teknik pendukung yang dapat diterapkan (Hovy, 2003).

Metode yang kedua adalah metode *interpretation* atau interpretasi yaitu perpaduan antara konsep, evaluasi dan proses lain

yang dilakukan oleh manusia dengan berbagai pemahaman masing-masing. Karena yang dihasilkan dari metode ini adalah sesuatu (hasil) yang baru, tidak dengan tegas terdapat pada input, maka sistem perlu memiliki pengetahuan khusus yang terpisah dari input yang diberikan (Hovy, 2003).

Hasil dari metode interpretasi biasanya berupa representasi abstrak yang tak terbaca, bahkan intisarinya jarang yang dapat dimengerti karena bergantung pada referensi yang dimiliki, menghilangkan ketersambungan pernyataan dan kadang menghilangkan atau mengulang beberapa materi. Oleh karena itu sistem memerlukan metode tambahan yang ketiga berupa *summary generation* (pembentukan ringkasan) untuk membentuk teks yang terbaca manusia (Hovy, 2003).

2.3.4 *Precision and Recall*

Performa dari identifikasi topik sebuah dokumen biasanya diukur menggunakan *Precision and Recall scores* (Hovy, 2003). Jika terdapat dua perbandingan hasil, hasil ringkasan sistem dan hasil ringkasan manusia, penilaian dilakukan pada seberapa dekat hasil ekstraksi topik yang dilakukan sistem dengan hasil ekstraksi yang dilakukan manusia. Pada skripsi ini akan dibandingkan hasil ekstraksi kalimat utama yang dilakukan sistem terhadap masing-masing tipe paragraf yang telah diketahui letak kalimat topiknya sesuai dengan tipe paragraf. *Precision* adalah nilai yang menandakan seberapa besar sistem berhasil mengekstrak kalimat topik yang sesuai, dan *recall* adalah nilai yang menandakan seberapa besar sistem gagal mengekstrak kalimat topik yang sesuai (Hovy, 2003).

Jika dimisalkan *correct* adalah jumlah kalimat topik yang berhasil diekstrak sistem dan sesuai dengan hasil ekstrak manusia, *wrong* adalah jumlah kalimat yang berhasil diekstrak sistem tetapi tidak diekstrak oleh manusia, dan *missed* adalah jumlah kalimat yang diekstrak manusia tetapi tidak diekstrak oleh sistem, maka rumus *Precision and Recall* ditunjukkan pada persamaan 2.4 dan 2.5 berikut (Hovy, 2003).

$$Precision = \frac{correct}{(correct + wrong)} \quad \textbf{(2.4)}$$

$$Recall = \frac{correct}{correct + missed} \quad \textbf{(2.5)}$$

2.4 LexRank

LexRank diadopsi dari PageRank, yang merupakan algoritma perankingan untuk halaman *web*. Metode PageRank menggunakan pengaplikasian graf matematis sebagai penyelesaiannnya. Formula PageRank adalah seperti yang dinyatakan dalam persamaan 2.6.

Saat diterapkan pada graf tekstual, biasanya terdapat bobot antara dua *vertex*, maka formula yang dipakai dinyatakan dalam persamaan 2.7.

$$S(V_i) = (1-d) + d * \sum_{V_j \in I_n(V_i)}^{n} \frac{1}{\left|\mathrm{Out}(\mathrm{V_j})\right|} \mathrm{S}(\mathrm{V_j}) \quad \textbf{(2.6)}$$

$$WS(V_i) = (1-d) + d * \sum_{V_j \in I_n(V_i)} \frac{\mathrm{w_{ji}}}{\sum_{\mathrm{V_k} \in \mathrm{Out}(\mathrm{V_j})} \mathrm{w_{jk}}} WS(V_j) \quad \textbf{(2.7)}$$

Formula 2.7 mengasumsikan graf berarah. Namun algortima ini dapat juga diterapkan pada graf tidak berarah dan berbobot (*undirected and weighted graph*), sehingga formulasinya seperti dalam persamaan 2.8.

$$WS(V_i) = d + (1-d) * \sum_{V_j \in Adj(V_i)} \frac{\mathrm{w_{ji}}}{\sum_{\mathrm{V_k} \in \mathrm{Adj}(\mathrm{V_j})} \mathrm{w_{jk}}} WS(V_j) \quad \textbf{(2.8)}$$

Dimana: V_i = vertex yang dihitung skor-nya; V_j = *vertex* yang bertetanggaan dengan V_i; V_k = *vertex* yang bertetanggaan dengan V_j; dan d = *damping factor* yang nilainya antara 0 dan 1, biasanya 0,85.

2.4.1 LexRank untuk Ekstraksi Dokumen

Unit teks yang dipilih sebagai *vertex* dalam ekstraksi kalimat tentunya keseluruhan kalimat dalam teks, satu *vertex* merepresentasikan sebuah kalimat. Sedangkan edge *vertex* merepresentasikan *similarity* antara dua kalimat yang saling berelasi. *Similarity* didefinisikan sebagai *content overlap* antara dua kalimat, yaitu jumlah kata yang sama antara keduanya. Selain *content overlap*, pengukuran *similarity* lain dapat juga dipakai, misalnya *cosine similarity*.

Secara garis besar, langkah-langkah perankingan pada graf tekstual adalah sebagai berikut:

- Mengidentifikasi unit teks terbaik untuk mendefinisikan tujuan perankingan, dan ditambahkan sebagai *vertex* dalam graf. Dalam hal ekstraksi kalimat, seluruh kalimat menjadi *vertex* dalam graf.
- Mengidentifikasi relasi yang menghubungkan unit teks tersebut, dan relasi ini digunakan untuk membuat *edge* antar *vertex*. Dalam hal ini diaplikasikan TF-IDF *similarity* antarkalimat.
- Setiap *vertex* di-*assign* skor awal. Skor awal tidak mempengaruhi skor akhir, hanya mempengaruhi jumlah iterasi.
- Mengiterasikan algoritma perankingan
- Mengurutkan *vertex* berdasarkan skor akhirnya. Nilai skor setiap *vertex* digunakan untuk melakukan penentuan ranking/pemilihan. Kalimat-kalimat *top-rank* akan dipilih menjadi ringkasan ekstraktif.

2.5 Penelitian Sebelumnya

Pada Program Studi Ilmu Komuter Fakultas MIPA Universitas Brawijaya pernah dilakukan penelitian terhadap pembuatan ringkasan dokumen otomatis. Dalam tugas akhir ini dibangun sebuah peringkas dokumen dengan menggunakan tipe ekstraktif dan menggunakan sebuah teks atau dokumen sebagi graf. Ringkasan yang dibentuk adalah *shortest path* atau jarak terpendek dari graf dokumen tersebut. Metode ini gabungan dari *sentences based summarization* dan *extraction based summarization* dengan algoritma *shortest path*. Pada *Sentences based summarization* dijelaskan bagaimana membentuk ringkasan dari pemilihan kalimat-kalimat yang penting. Dengan menggunakan lima fitur atau parameter yaitu : lokasi, kemiripan dengan judul, frekuensi

kemunculan kata, adanya kata kunci, dan *query*. Sedangkan pada *extraction based summarization* dijelaskan bagaimana membentuk ringkasan dengan pencarian jalur terpendek dari dokumen graph. Untuk evaluasi digunakan metode *precision and recall*. Selain itu, ringkasan hasil sistem akan dibandingkan dengan ringkasan yang dihasilkan oleh *autosummarized* pada Microsoft Word.

Perangkingan kalimat

Pembobotan kalimat atau pemilihan kalimat-kalimat penting yang digunakan dalam pembentukan graf dilakukan dengan memberikan skor pada masing-masing kalimat sesuai dengan lima fitur atau parameter, aturan pembobotannya adalah :

1. Fitur frekuensi

   Fitur frekuensi adalah menghitung kemunculan masing-masing kata pada sebuah kalimat, kemudian mengenolkan frekuensi kata yang berada dalam daftar *stop word* atau yang berada dibawah *threshold* tertentu.

2. Fitur lokasi

   Fitur ini bergantung pada posisi kalimat dalam paragraf, Jika kalimat terdapat pada paragraf pertama maka beri bobot lokasi1, jika tidak, maka beri bobot lokasi2. Jika kalimat terdapat pada awal paragraf maka beri bobot :

   bobot = bobot + lokasi3

   Jika tidak maka beri bobot :

   bobot = bobot lokasi + 4.

3. Fitur *query*

   User dapat dapat memberikan *Query* seperti pada pencarian di sebuah *search engine*. Jika *user* ingin memasukkan *query* dalam bentuk frase maka *user* tersbeut menggunakan tanda petik. Untuk setiap kalimat terdapat *query* maka diberi bobot=bobot. Jika tidak = 0

4. Fitur *cue*

   Kemunculan ungkapan kusus dalam bacaan dokumen juga merupakan petunjuk topik suatu bacaan, contohnya : “jadi”, “kesimpulannya”, “adalah”, ”hasilnya” , “contohnya” dll. Jika kalimat mengandung cue maka diberi bobot jika tidak tidak bobot= 0

5. Fitur judul

Memeriksa setiap kalimat dalam teks jika mengandung kata kata dalam judul maka diberi bobot jika tidak nol.

Pembentukan Graf

Dari bobot masing-masing kalimat yang telah ditentukan, dibuat sebuah graf keterhubungan antarkalimat. *Edge* antar *vertex* menggunakan rumus. Kemudian dari graf tersebut ditentukan susunan rigkasan dokumen dengan menentukan *shortest path* atau jalur terpendek menggunakan algoritma Djikstra.

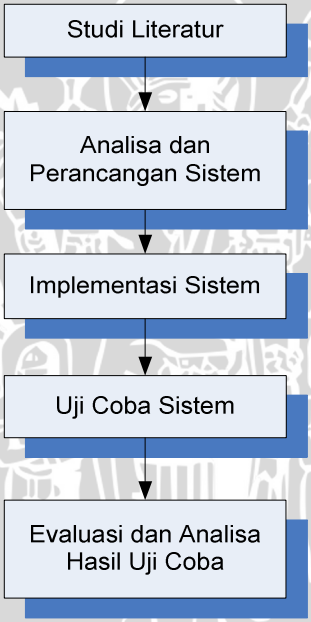
Kesimpulan

Peringkas dokumen dengan menggabungkan dua metode yaitu : pembobotan kalimat dan algoritma *shortest path* menghasilkan rata-rata *recall* sebesar 58%, rata-rata *precision* sebesar 52% dan rata-rata *F-measure* sebesar 54% sedangkan *autosummarize* menghasilkan rata-rata *recall* 49% *precision* sebesar 43% dan *F-Measure* sebesar 46%.

BAB III

METODOLOGI

Sistem ini diracang untuk menghasilkan aplikasi yang mampu menerapkan peringkasan dokumen dengan menggunakan metode *graph based summarization algorithm*. Sistem akan diterapkan pada mesin komputer personal (PC) dengan aplikasi berbasis Java. Pada sistem ini akan diujikan beberapa jenis dokumen dengan parameter tipe paragraf yang berbeda. Gambar 3.1 menggambarkan sistem yang akan dibangun.



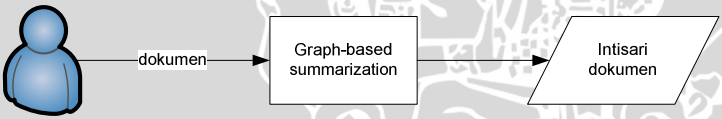
Gambar 3-1 Diagram Sistem

Adapun tahapan pembuatan sistem adalah sebagai berikut :

1. Melakukan studi literatur terhadap algoritma LexRank *graph based summarization*
2. Menganalisa dan merancang perangkat lunak untuk menerapkan algoritma *graph based summarization*.
3. Mengimplementasikan perangkat lunak berdasarkan analisa dan perancangan yang dilakukan.
4. Melakukan uji coba terhadap perangkat lunak dan mengumpulkan data hasil pengujian yang dilakukan.
5. Melakukan evaluasi terhadap perangkat lunak dan data hasil uji coba yang diperoleh.

3.1 Perancangan Sistem Secara Keseluruhan

Secara umum, sistem akan memiliki fungsi untuk menerima input dokumen dari *user* yang akan diringkas, kemudian dengan menggunakan metode *graph based algorithm* sistem akan melakukan peringkasan dokumen sehingga dihasilkan ekstrak atau intisari dari dokumen. Gambar 3.2 berikut akan menjelaskan skema aliran data dalam sistem.



Gambar 3-2 Aliran Data

Dalam skripsi ini akan dilakukan uji coba terhadap beberapa dokumen yang berbeda. Dokumen yang diujikan sebelumnya telah dilakukan peringkasan atau pengekstrakan kalimat topik oleh ahli (Mahasiswa Fakultas Ilmu Budaya Universitas Brawijaya). Dari dokumen-dokumen yang akan diujikan tersebut, akan diukur tingkat *precision and recall* hasil ringkasan dokumen.

Perangkat lunak untuk pengujian data dibuat dalam dua versi. Versi pertama melakukan pengekstrakan kalimat dan menghitung rangking tiap kalimat secara global, sehingga masing-masing kalimat akan dihitung nilai *similarity* dengan seluruh kalimat yang lainnya. Versi kedua melakukan pengekstrakan kalimat dan

menghitung rangking tiap kalimat dalam masing-masing paragraf, dan dipilih kalimat dengan rangking tertinggi sesuai dengan nilai *treshold* yang diberikan sebagai kalimat topik masing-masing paragraf.

Pada percobaan pertama akan dilakukan pada aplikasi versi 1 dengan metode pengambilan rangking secara keseluruhan dan percobaan kedua dilakukan terhadap aplikasi versi 2 dengan metode pengambilan rangking pada masing-masing paragraf.

Tabel 3.1 dan 3.2 adalah contoh pengambilan data dari uji coba *precision and recall.*

Tabel 3-1 Nilai Precision (P) dan Recall (R) Aplikasi versi 1

Kode Dokumen	*Treshold Similarity* (%)	*Treshold* Sailence (%)							
		0		25		50		75	
		P	R	P	R	P	R	P	R
	0								
	25								
	50								
	75								

Tabel 3-2 Nilai Precision (P) dan Recall (R) Aplikasi versi 2

Kode dokumen	*Treshold Similarity* (%)	*Treshold* Sailence (%)							
		0		25		50		75	
		P	R	P	R	P	R	P	R
	0								
	25								
	50								
	75								

3.2 Perancangan Proses

Perangkat lunak yang akan dibuat adalah aplikasi peringkas dokumen otomatis. Dokumen dalam hal ini merupakan *input* atau masukan dari *user* berupa *file* dalam bentuk teks berekstensi *.txt*. Kemudian setelah dokumen berhasil dimasukkan oleh *user*, maka sistem akan mulai melakukan proses peringkasan dokumen.

Proses peringkasan dokumen yang dilakukan oleh sistem diawali dengan membaca dan mengekstrak informasi yang terdapat dalam dokumen seperti jumlah kata, jumlah kalimat dan jumlah paragraf. Setelah proses pengambilan informasi dilakukan, sistem akan mulai membangun graf dari kalimat-kalimat dalam dokumen. Graf tersusun atas *vertex* yang merupakan representasi dari tiap kalimat, dan dihubungkan oleh *edge* atau sisi yang merupakan nilai kemiripan (*similarity*) dari dua kalimat yang dihubungkan.

Untuk mengetahui nilai *similarity* kalimat yang dihubungkan ini, digunakan metode *tf-idf* ternormalisasi. Proses *tf-idf similarity* melalui beberapa tahap *preprocessing*, yaitu proses *filtering* (penghilangan kata yang tidak penting) dan *stemming* (pemotongan kata atau *term* menjadi kata dasar). Secara umum, arsitektur sistem digambarkan pada gambar 3.3.

Proses *filtering* dalam penelitian ini menggunakan algoritma *stopword* dimana tiap kata (*term*) dicek apakah kata tersebut terdapat dalam daftar stopword. Jika terdapat dalam stopword, maka kata tersebut tidak dimasukkan dalam daftar kata unik.

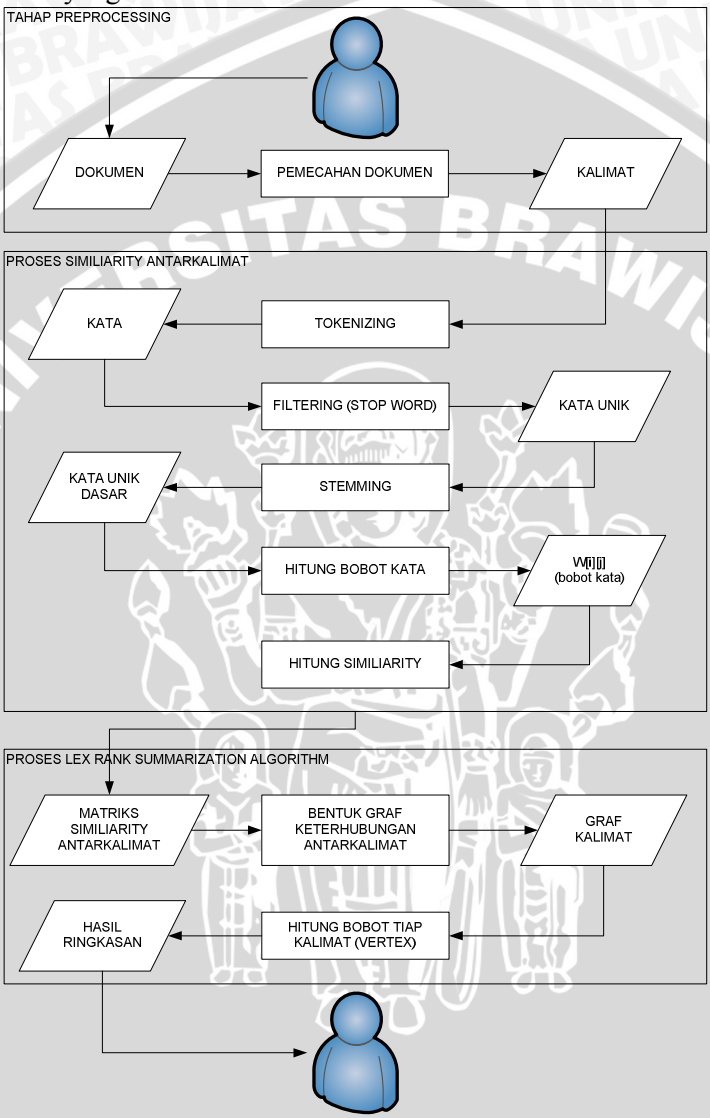
Selanjutnya proses *stemming*, yaitu proses pencarian kata dasar dari *term* unik hasil filtering. Untuk mendapatkan kata unik yang berupa kata dasar, dilakukan proses pemotongan kata dasar pada tiap *term.*

Langkah selanjutnya dalam proses *stemming* adalah memotong partikel seperti “-lah”, “-kah”, “-pun”. Kemudian memotong kata ganti kepemilikan seperti “-mu”, “ku-“, “-nya”.

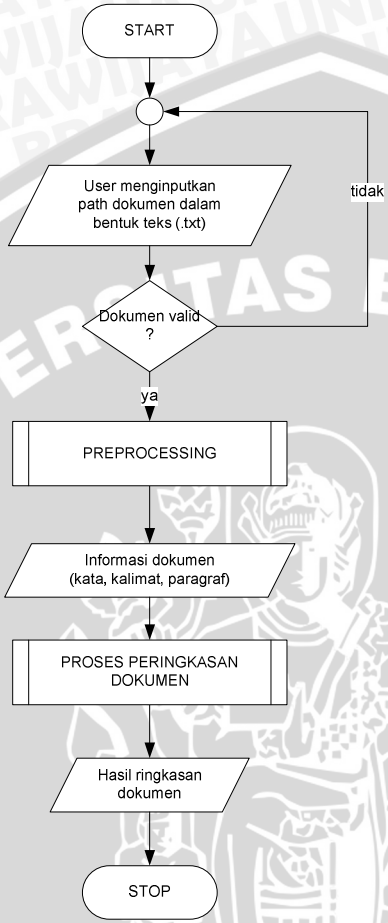
Langkah berikutnya yaitu melakukan pemotongan terhadap imbuhan, antara lain *prefiks* (awalan) dan *suffiks* (akhiran) dan *confix* (awalan dan akhiran).

Sistem akan bekerja setelah *user* menginputkan data berupa dokumen teks, kemudian jika sistem dapat membaca dokumen teks hasil *input user*, sistem akan melakukan proses peringkasan

dokumen. Pada gambar 3.4 digambarkan proses peringkasan dokumen yang dilakukan sistem

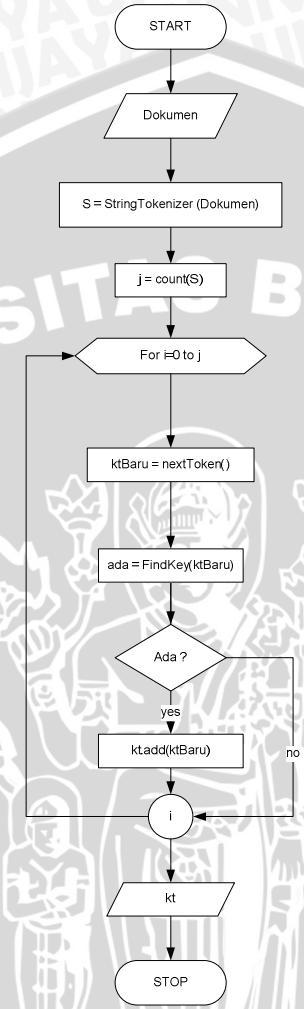


Gambar 3-3 Arsitektur Sistem



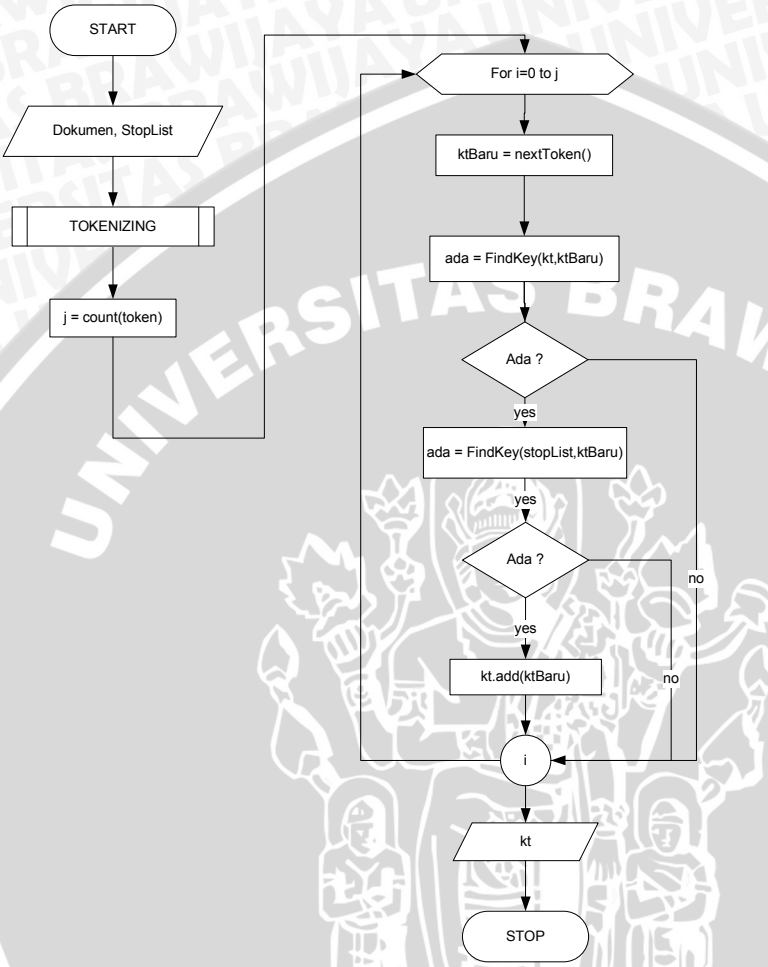
Gambar 3-4 Flowchart Proses Sistem

Pada tahap *preprocessing* terdapat proses *tokenizing* terhadap dokumen yang diinputkan. Proses *tokenizing* dokumen menjadi kata-kata digambarkan pada gambar 3.5 berikut.



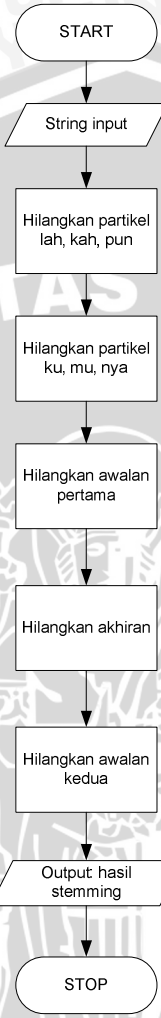
Gambar 3-5 Proses Tokenizing

Proses Filtering dilakukan untuk mendapatkan daftar kata yang unik atau kata yang penting. Proses filtering dilakukan dengan menggunakan algoritma stopList, yaitu melakukan pemfilteran kata yang diberikan terhadap daftar kata yang sering muncul (stopword). Tahap Filtering digambarkan pada gambar 3.6.



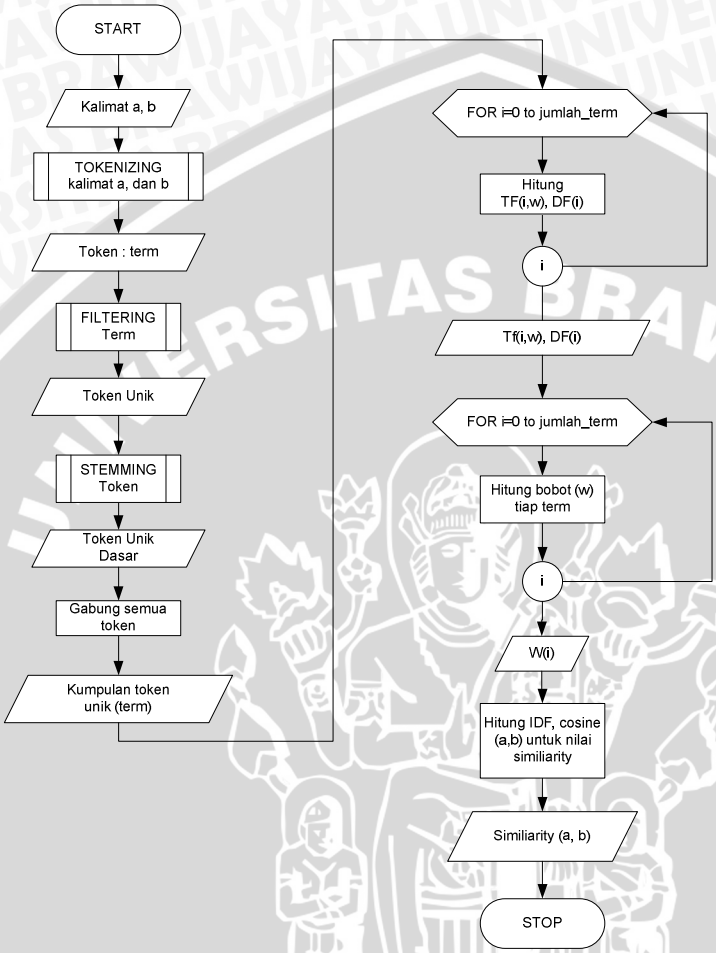
Gambar 3-6 Proses Filtering

Pada tahap *tokenizing* dan *filtering* terhadap dokumen disisipi proses *stemming* terhadap kata-kata yang dihasilkan agar didapatkan kata *root* atau kata dasar. Proses Stemming untuk mendapatkan kata dasar digambarkan pada gambar 3.7.



Gambar 3-7 Proses Stemming

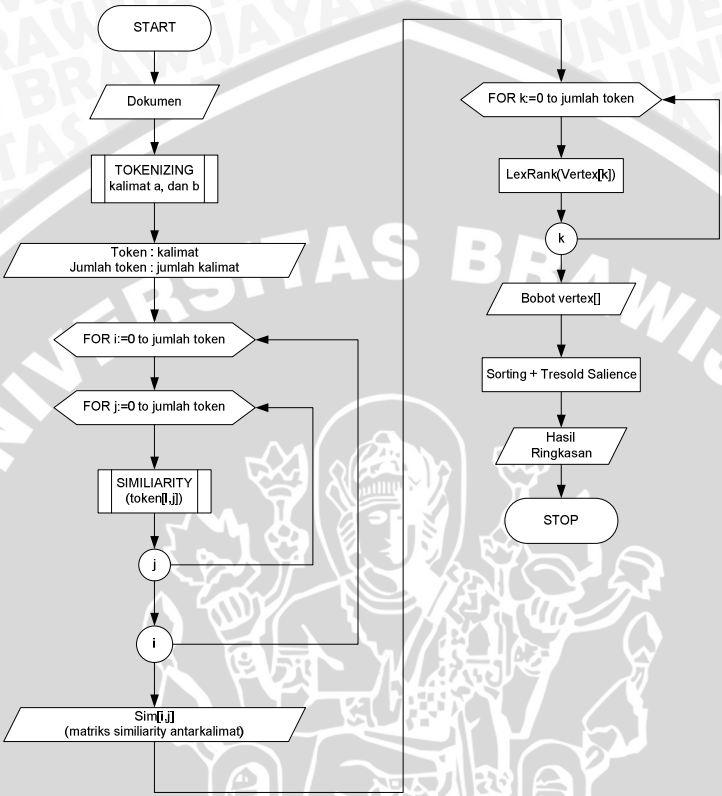
Untuk melakukan proses perhitungan LexRank, diperlukan data berupa matriks keterhubungan antarkalimat yang ditentukan dengan nilai *similarity* antarkalimat. Pada gambar 3.5 berikut digambarkan proses perhitungan nilai *similarity* antarkalimat menggunakan algoritma TF-IDF *vector space models*.



Gambar 3-8 Proses Perhitungan TF-IDF *similarity*

Proses TF-IDF *similarity* dilakukan terhadap seluruh kalimat terhadap kalimat yang lain, dan menggunakan asas kombinasi, dimana *similarity* kalimat A-B adalah sama dengan *similarity* kalimat B-A.

Dari proses penilaian *similarity* antarkalimat, didapatkan matriks *similarity* antarkalimat yang digunakan dalam perhitungan LexRank untuk menentukan bobot masing-masing kalimat dalam dokumen. Proses LexRank tampak pada gambar 3.6 berikut.



Gambar 3-9 Proses *LexRank Summarization* Algorithm

3.2.1 Parameter Input Perangkat Lunak

Pada aplikasi peringkas dokumen otomatis ini, ada beberapa parameter input yang harus dilengkapi oleh pengguna, yaitu :

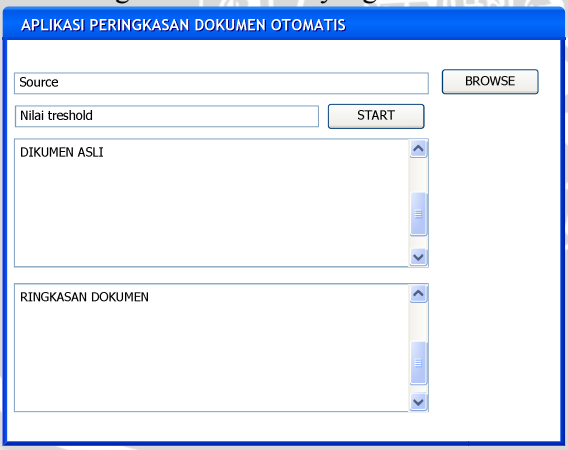
1. Sumber dokumen.
2. Nilai *treshold similarity* antarkalimat (%). Nilai ini akan berpengaruh pada keterhubungan antarkalimat.
3. Nilai *treshold summarization* (%). Nilai ini akan menentukan ukuran ringkasan dokumen.

3.2.2 Perancangan User Interface

Sistem ini akan diterapkan pada komputer personal (PC) dengan basis aplikasi Java *dekstop*. Secara garis besar, sistem akan berfungsi untuk membaca file sumber yang diberikan oleh *user*, kemudian mengekstrak informasi yang terkandung didalamnya (jumlah kata, kalimat, paragraf). Setelah didapatkan data generik dari dokumen yang diinputkan *user*, sistem akan melakukan pembobotan terhadap masing-masing kalimat sesuai dengan nilai *treshold* yang diberikan *user*.

Gambar 3.10 adalah contoh rancangan *user interface* dari sistem. Bagian-bagian dari antarmuka sistem adalah :

1. *Field* **SOURCE** untuk mengisikan path dari dokumen yang akan diproses.
2. Tombol **BROWSE** yang digunakan untuk membuka *open dialog*, yaitu bantuan bagi *user* dalam memilih dokumen.
3. *Field* **NILAI *TRESHOLD*** untuk mengisikan nilai batasan toleransi sistem.
4. Tombol **START** untuk memulai proses peringkasan dokumen.
5. *Field* **DOKUMEN ASLI** untuk menampilkan dokumen asli yang diinputkan *user*.
6. *Field* **RINGKASAN DOKUMEN** untuk menampilkan hasil ringkasan dokumen yang dilakukan sistem



Gambar 3-10 Form Utama

3.3 Contoh Perhitungan TF-IDF *algorithm* dan *Graph Based Summarization*

Misalkan diberikan sebuah dokumen teks yang terdiri atas beberapa paragraf sebagai berikut :

> Sebuah tiang beton milik perusahaan telekomunikasi setinggi 30 meter ambruk di Jalan Raya Raci, Pakal, Surabaya. Dua bangunan rumah rusak terkena ambrukan tiang itu. "Tiang itu ambruk saat ditarik dua pekerjanya yang akan merobohkan tiang itu" kata Heri, salah satu warga kepada detiksurabaya.com, Minggu (15/11/2009).
>
> Tiang sepanjang 30 meter itu juga ambruk memenuhi badan jalan sehingga arus lalu lintas dialihkan melewati jalan perkampungan. Forklift dan tukang las pun dikerahkan untuk mengangkat dan memotong tiang tersebut. Setelah terpotong, tiang itu diletakkan di pinggir jalan. Pemadaman listrik juga terjadi karena ambrukan tiang itu memutuskan kabel listrik.

Dari dokumen yang diberikan tersebut, langkah pertama yang dilakukan adalah melakukan ekstraksi terhadap informasi-informasi umum yang terdapat pada dokumen. Hasilnya tampak pada tabel 3.3 berikut.

Tabel 3-3 Informasi Dokumen

Informasi	Keterangan
Jumlah Paragraf	2
Jumlah Kalimat	7
Jumlah Kata (seluruh)	89

Kemudian dari dokumen tersebut dilakukan pemecahan menjadi kalimat-kalimat seperti pada tabel 3.4 berikut.

Tabel 3-4 Hasil Pemecahan Dokumen

No	Kode	Kalimat
A	s1p1	Sebuah tiang beton milik perusahaan telekomunikasi setinggi 30 meter ambruk di Jalan Raya Raci, Pakal, Surabaya
B	s2p1	Dua bangunan rumah rusak terkena ambrukan tiang itu

C	s3p1	Tiang itu ambruk saat ditarik dua pekerjanya yang akan merubuhkan tiang itu kata Heri, salah satu warga kepada detiksurabayacom, Minggu (15/11/2009)
D	s1p2	Tiang sepanjang 30 meter itu juga ambruk memenuhi badan jalan sehingga arus lalu lintas dialihkan melewati jalan perkampungan
E	s2p2	Forklift dan tukang las pun dikerahkan untuk mengangkat dan memotong tiang tersebut
F	s3p2	Setelah terpotong, tiang itu diletakkan di pinggir jalan
G	s4p2	Pemadaman listrik juga terjadi karena ambrukan tiang itu memutuskan kabel listrik

Kemudian dari daftar kalimat yang dibentuk, selanjutnya dibentuk matriks keterhubungan antarkalimat berdasarkan nilai *similarity* antarkalimat dan nilai *treshold* yang diberikan menggunakan algoritma TF-IDF ternormalisasi.

3.3.1 Tahap TF-IDF *normalized Similarity*

Dari daftar kalimat yang diberikan, langkah pertama yang dilakukan adalah *Tokenizing* yaitu mendaftar semua *term* (kata) dari masing-masing dokumen dan *query*, mengubahnya ke dalam bentuk dasar (*Stemming*) kemudian menghilangkan kata-kata yang dianggap sering muncul (misalnya : dan, ke di, yang, adalah, sehingga, itu, walaupun, maka, dll) atau proses *Filtering*. Kata-kata ini disebut dengan "*stop word"*. Hasil proses *Tokenizing, Stemming* dan *Filtering* tampak seperti pada tabel 3.5 berikut.

Tabel 3-5 Daftar Token

Token	**KALIMAT**						
	A	**B**	**C**	**D**	**E**	**F**	**G**
Tiang	√	√	√	√	√	√	√
Beton	√						

milik	√						
perusahaan	√						
telekomunikasi	√						
tinggi	√			√			
30	√			√			
ambruk	√	√	√				√
jalan	√						
Raya	√						
Raci	√						
Pakal	√						
Surabaya	√						
bangun		√					
rumah		√					
rusak		√					
kena		√					
tarik			√				
kerja			√				
rubuh			√				
Heri,			√				
warga			√				
detiksurabaya.com			√				
Minggu			√				
(15/11/2009)			√				
penuh				√			
badan				√			
jalan				√			
arus				√			
lalu				√			
lintas				√			
Alih				√			
Lewat				√			

kampung				√			
Forklift					√		
tukang					√		
Las					√		
Kerah					√		
angkat					√		
potong					√	√	
diletakkan						√	
pinggir						√	
Padam							√
Listrik							√
Putus							√
Kabel							√

Setelah didapatkan daftar *term* dan posisinya di tiap kalimat, dihitung nilai similiarty kalimat dengan kalimat yang lainnya menggunakan rumus TF-IDF *vector space models*.

Langkah pertama adalah menghitung TF(*Term Frequency*), DF(*Document Frequency*), dan dihitung *weight* (W) dari masing-masing term menggunakan persamaan 2.3. Kemudian dihitung nilai *similarity* antarkalimat menggunakan persamaan 2.2. Proses ini dilakukan pada semua kalimat terhadap kalimat yang lainnya. Hasilnya seperti yang ditampilkan pada tabel 3.6 – 3.26 berikut.

Tabel 3-6 *Similarity* Kalimat A-B

Query	**D0**	**D1**	**DF**	**W0**	**W1**
Tiang	1	1	2	0,301029996	0,301029996
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	0	1	0	0

ambruk	1	1	2	0,301029996	0,301029996
jalan	1	0	1	0	0
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0
surabaya	1	0	1	0	0
dua	0	1	1	0	0
bangun	0	1	1	0	0
rumah	0	1	1	0	0
rusak	0	1	1	0	0
kena	0	1	1	0	0
D	1			0,602059991	0,602059991
max tf	1			0,425720703	0,425720703
				Cosine (A-B)	0,362476233

Tabel 3-7 *Similarity* Kalimat A-C

Query	D0	D1	DF	W0	W1
tiang	1	2	2	0,150514998	0,301029996
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	1	2	0,150514998	0,150514998
jalan	1	0	1	0	0
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0
surabaya	1	0	1	0	0
saat	0	1	1	0	0

ditarik	0	1	1	0	0
dua	0	1	1	0	0
pekerjanya	0	1	1	0	0
merubuhkan	0	1	1	0	0
kata	0	1	1	0	0
heri,	0	1	1	0	0
salah	0	1	1	0	0
satu	0	1	1	0	0
warga	0	1	1	0	0
kepada	0	1	1	0	0
detiksurabaya.com	0	1	1	0	0
minggu	0	1	1	0	0
(15/11/2009)	0	1	1	0	0
D	1			0,301029996	0,451544993
max tf	2			0,212860351	0,336561767
				Cosine (A-C)	0,214921968

Tabel 3-8 *Similarity* Kalimat A-D

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,150514998	0,150514998
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	1	2	0,150514998	0,150514998
ambruk	1	1	2	0,150514998	0,150514998
jalan	1	2	2	0,150514998	0,301029996
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0

surabaya	1	0	1	0	0
panjang	0	1	1	0	0
penuh	0	1	1	0	0
badan	0	1	1	0	0
arus	0	1	1	0	0
lalu	0	1	1	0	0
lintas	0	1	1	0	0
alih	0	1	1	0	0
lewat	0	1	1	0	0
kampung	0	1	1	0	0
D	1			0,602059991	0,752574989
max tf	2			0,301029996	0,398225253
				Cosine (A-D)	0,599388731

Tabel 3-9 *Similarity* Kalimat A-E

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,301029996	0,301029996
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	0	1	0	0
jalan	1	0	1	0	0
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0
surabaya	1	0	1	0	0
Forklift	0	1	1	0	0
tukang	0	1	1	0	0

las	0	1	1	0	0
kerah	0	1	1	0	0
angkat	0	1	1	0	0
potong	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	1			0,301029996	0,301029996
				Cosine (A-E)	0,090619058

Tabel 3-10 *similarity* Kalimat A-F

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,301029996	0,301029996
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	0	1	0	0
jalan	1	1	2	0,301029996	0,301029996
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0
surabaya	1	0	1	0	0
setelah	0	1	1	0	0
potong	0	1	1	0	0
letak	0	1	1	0	0
pinggir	0	1	1	0	0
D	1			0,602059991	0,602059991
max tf	1			0,425720703	0,425720703
				Cosine (A-F) =	0,362476233

Tabel 3-11 *similarity* Kalimat A-G

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,301029996	0,301029996
beton	1	0	1	0	0
milik	1	0	1	0	0
perusahaan	1	0	1	0	0
telekomunikasi	1	0	1	0	0
setinggi	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	1	2	0,301029996	0,301029996
jalan	1	0	1	0	0
raya	1	0	1	0	0
raci	1	0	1	0	0
pakal	1	0	1	0	0
surabaya	1	0	1	0	0
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,602059991	0,602059991
max tf	1			0,425720703	0,425720703
				Cosine (A-G)	0,362476233

Tabel 3-12 *similarity* Kalimat B-D

Query	D0	D1	DF	W0	W1
dua	1	0	1	0	0
bangunan	1	0	1	0	0
rumah	1	0	1	0	0
rusak	1	0	1	0	0
kena	1	0	1	0	0

ambruk	1	1	2	0,150514998	0,150514998
tiang	1	1	2	0,150514998	0,150514998
panjang	0	1	1	0	0
30	0	1	1	0	0
penuh	0	1	1	0	0
badan	0	1	1	0	0
jalan	0	2	1	0	0
arus	0	1	1	0	0
lalu	0	1	1	0	0
lintas	0	1	1	0	0
alih	0	1	1	0	0
lewat	0	1	1	0	0
kampung	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	2			0,212860351	0,212860351
				Cosine (B-D)	0,090619058

Tabel 3-13 *similarity* kalimat B-C

Query	D0	D1	DF	W0	W1
dua	1	1	2	0,150514998	0,150514998
bangunan	1	0	1	0	0
rumah	1	0	1	0	0
rusak	1	0	1	0	0
kena	1	0	1	0	0
ambruk	1	1	2	0,150514998	0,150514998
tiang	1	2	2	0,150514998	0,301029996
saat	0	1	1	0	0
ditarik	0	1	1	0	0
kerja	0	1	1	0	0
rubuh	0	1	1	0	0
kata	0	1	1	0	0

Heri,	0	1	1	0	0
salah	0	1	1	0	0
satu	0	1	1	0	0
warga	0	1	1	0	0
kepada	0	1	1	0	0
detiksurabayacom	0	1	1	0	0
Minggu	0	1	1	0	0
(15/11/2009)	0	1	1	0	0
D	1			0,451544993	0,602059991
max tf	2			0,260699624	0,368684943
				Cosine (B-C)	0,384464104

Tabel 3-14 *similarity* Kalimat B-E

Query	D0	D1	DF	W0	W1
dua	1	0	1	0	0
bangunan	1	0	1	0	0
rumah	1	0	1	0	0
rusak	1	0	1	0	0
kena	1	0	1	0	0
ambruk	1	0	1	0	0
tiang	1	1	2	0,301029996	0,301029996
Forklift	0	1	1	0	0
tukang	0	1	1	0	0
las	0	1	1	0	0
kerah	0	1	1	0	0
angkat	0	1	1	0	0
potong	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	1			0,301029996	0,301029996
				Cosine (B-E)	0,090619058

Tabel 3-15 *similarity* Kalimat B-G

Query	D0	D1	DF	W0	W1
dua	1	0	1	0	0
bangunan	1	0	1	0	0
rumah	1	0	1	0	0
rusak	1	0	1	0	0
kena	1	0	1	0	0
ambruk	1	1	2	0,301029996	0,301029996
tiang	1	1	2	0,301029996	0,301029996
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,602059991	0,602059991
max tf	1			0,425720703	0,425720703
				Cosine (B-G)	0,362476233

Tabel 3-16 *similarity* Kalimat B-F

Query	D0	D1	DF	W0	W1
dua	1	0	1	0	0
bangunan	1	0	1	0	0
rumah	1	0	1	0	0
rusak	1	0	1	0	0
kena	1	0	1	0	0
ambruk	1	0	1	0	0
tiang	1	1	2	0,301029996	0,301029996
potong	0	1	1	0	0
letak	0	1	1	0	0
pinggir	0	1	1	0	0
jalan	0	1	1	0	0

D	1	0,301029996	0,301029996
max tf	1	0,301029996	0,301029996
		Cosine (B-F)	0,090619058

Tabel 3-17 *similarity* Kalimat C-G

Query	D0	D1	DF	W0	W1
tiang	2	1	2	0,301029996	0,150514998
ambruk	1	1	2	0,150514998	0,150514998
saat	1	0	1	0	0
ditarik	1	0	1	0	0
dua	1	0	1	0	0
kerja	1	0	1	0	0
rubuh	1	0	1	0	0
kata	1	0	1	0	0
Heri	1	0	1	0	0
salah	1	0	1	0	0
satu	1	0	1	0	0
warga	1	0	1	0	0
kepada	1	0	1	0	0
detiksurabayacom	1	0	1	0	0
Minggu	1	0	1	0	0
(15/11/2009)	1	0	1	0	0
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,451544993	0,301029996
max tf	2			0,336561767	0,212860351
				Cosine (C-G)	0,085968787

Tabel 3-18 *similarity* Kalimat C-D

Query	D0	D1	DF	W0	W1
tiang	2	1	2	0,301029996	0,150514998
ambruk	1	1	2	0,150514998	0,150514998
saat	1	0	1	0	0
ditarik	1	0	1	0	0
dua	1	0	1	0	0
kerja	1	0	1	0	0
rubuh	1	0	1	0	0
kata	1	0	1	0	0
Heri	1	0	1	0	0
salah	1	0	1	0	0
satu	1	0	1	0	0
warga	1	0	1	0	0
kepada	1	0	1	0	0
detiksurabayacom	1	0	1	0	0
Minggu	1	0	1	0	0
panjang	0	1	1	0	0
30	0	1	1	0	0
penuh	0	1	1	0	0
badan	0	1	1	0	0
jalan	0	2	1	0	0
arus	0	1	1	0	0
lalu	0	1	1	0	0
lintas	0	1	1	0	0
alih	0	1	1	0	0
lewat	0	1	1	0	0
kampung	0	1	1	0	0
D	1			0,451544993	0,301029996
max tf	2			0,336561767	0,212860351
				Cosine (C-D)	0,085968787

Tabel 3-19 *similarity* Kalimat E-F

Query	D0	D1	DF	W0	W1
Forklift	1	0	1	0	0
tukang	1	0	1	0	0
las	1	0	1	0	0
kerah	1	0	1	0	0
angkat	1	0	1	0	0
potong	1	1	2	0,301029996	0,301029996
tiang	1	1	2	0,301029996	0,301029996
setelah	0	1	1	0	0
letak	0	1	1	0	0
pinggir	0	1	1	0	0
jalan	0	1	1	0	0
D	1			0,602059991	0,602059991
max tf	1			0,425720703	0,425720703
				Cosine (E-F)	0,362476233

Tabel 3-20 *similarity* Kalimat C-E

Query	D0	D1	DF	W0	W1
tiang	2	1	2	0,301029996	0,150514998
ambruk	1	0	1	0	0
saat	1	0	1	0	0
ditarik	1	0	1	0	0
dua	1	0	1	0	0
kerja	1	0	1	0	0
rubuh	1	0	1	0	0
kata	1	0	1	0	0
Heri	1	0	1	0	0
salah	1	0	1	0	0
satu	1	0	1	0	0
warga	1	0	1	0	0

kepada	1	0	1	0	0
detiksurabayacom	1	0	1	0	0
Minggu	1	0	1	0	0
(15/11/2009)	1	0	1	0	0
Forklift	0	1	1	0	0
tukang	0	1	1	0	0
las	0	1	1	0	0
kerah	0	1	1	0	0
angkat	0	1	1	0	0
potong	0	1	1	0	0
D	1			0,301029996	0,150514998
max tf	2			0,301029996	0,150514998
				Cosine (C-E)	0,022654765

Tabel 3-21 *similarity* Kalimat D-F

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,150514998	0,150514998
panjang	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	0	1	0	0
penuh	1	0	1	0	0
badan	1	0	1	0	0
jalan	2	1	2	0,301029996	0,150514998
arus	1	0	1	0	0
lalu	1	0	1	0	0
lintas	1	0	1	0	0
alih	1	0	1	0	0
lewat	1	0	1	0	0
kampung	1	0	1	0	0
setelah	0	1	1	0	0
potong	0	1	1	0	0

letak	0	1	1	0	0
pinggir	0	1	1	0	0
D	1			0,451544993	0,301029996
max tf	2			0,336561767	0,212860351
				Cosine (D-F)	0,085968787

Tabel 3-22 *similarity* Kalimat E-G

Query	D0	D1	DF	W0	W1
Forklift	1	0	1	0	0
tukang	1	0	1	0	0
las	1	0	1	0	0
kerah	1	0	1	0	0
angkat	1	0	1	0	0
potong	1	0	1	0	0
tiang	1	1	2	0,301029996	0,301029996
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
ambruk	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	1			0,301029996	0,301029996
				Cosine (E-G)	0,090619058

Tabel 3-23 *similarity* Kalimat C-F

Query	D0	D1	DF	W0	W1
tiang	2	1	2	0,301029996	0,150514998
ambruk	1	0	1	0	0
saat	1	0	1	0	0
ditarik	1	0	1	0	0

dua	1	0	1	0	0
kerja	1	0	1	0	0
rubuh	1	0	1	0	0
kata	1	0	1	0	0
Heri	1	0	1	0	0
salah	1	0	1	0	0
satu	1	0	1	0	0
warga	1	0	1	0	0
kepada	1	0	1	0	0
detiksurabayacom	1	0	1	0	0
Minggu	1	0	1	0	0
(15/11/2009)	1	0	1	0	0
setelah	0	1	1	0	0
potong	0	1	1	0	0
letak	0	1	1	0	0
pinggir	0	1	1	0	0
jalan	0	1	1	0	0
D	1			0,301029996	0,150514998
max tf	2			0,301029996	0,150514998
				Cosine (C-F)	0,022654765

Tabel 3-24 *similarity* Kalimat D-G

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,150514998	0,150514998
panjang	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	1	2	0,150514998	0,150514998
penuh	1	0	1	0	0
badan	1	0	1	0	0
jalan	2	0	1	0	0
arus	1	0	1	0	0

lalu	1	0	1	0	0
lintas	1	0	1	0	0
alih	1	0	1	0	0
lewat	1	0	1	0	0
kampung	1	0	1	0	0
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	2			0,212860351	0,212860351
				Cosine (D-G)	0,090619058

Tabel 3-25 *similarity* Kalimat F-G

Query	D0	D1	DF	W0	W1
setelah	1	0	1	0	0
potong	1	0	1	0	0
tiang	1	1	2	0,301029996	0,301029996
letak	1	0	1	0	0
pinggir	1	0	1	0	0
jalan	1	0	1	0	0
padam	0	1	1	0	0
listrik	0	1	1	0	0
terjadi	0	1	1	0	0
ambruk	0	1	1	0	0
putus	0	1	1	0	0
kabel	0	1	1	0	0
D	1			0,301029996	0,301029996
max tf	1			0,301029996	0,301029996
				Cosine (F-G)	0,090619058

Tabel 3-26 *similarity* Kalimat D-E

Query	D0	D1	DF	W0	W1
tiang	1	1	2	0,150514998	0,150514998
panjang	1	0	1	0	0
30	1	0	1	0	0
ambruk	1	0	1	0	0
penuh	1	0	1	0	0
badan	1	0	1	0	0
jalan	2	0	1	0	0
arus	1	0	1	0	0
lalu	1	0	1	0	0
lintas	1	0	1	0	0
alih	1	0	1	0	0
lewat	1	0	1	0	0
kampung	1	0	1	0	0
Forklift	0	1	1	0	0
Tukang	0	1	1	0	0
Las	0	1	1	0	0
Kerah	0	1	1	0	0
Angkat	0	1	1	0	0
Potong	0	1	1	0	0
D	1			0,150514998	0,150514998
max tf	2			0,150514998	0,150514998
				Cosine (D-E)	0,022654765

Dari hasil perhitungan nilai *similarity* antarkalimat dapat dibentuk sebuah tabel yang berisi matrik *similarity* antarkalimat seperti pada tabel 3.27 berikut. Kemudian dilakukan proses normalisasi terhadap nilai matriks agar sebaran nilai *similarity* menjadi normal, seperti yang ditampilkan pada tabel 3.28.

Dari tabel matriks *similarity* ternormalisasi, dilakukan proses filter *treshold* terhadap nilai *similarity* untuk menentukan keterhubungan antarkalimat. Dua kalimat yang mempunyai nilai

similarity dibawah *treshold* berarti dianggap tidak ada keterhubungan diantaranya. Pada perhitungan ini diberikan nilai *treshold summarization* (ukuran ringkasan) 0,2 (20%). Hasilnya ditampilkan pada tabel 3.29. untuk nilai *similarity* yang dibawah *treshold* (ditandai dengan warna merah), maka artinya tidak ada keterhubungan antara dua kalimat tersebut.

Tabel 3-27 Matriks *similarity* antarkalimat

	A	B	C	D	E	F	G
A	1	0,362476233	0,214921968	0,599388731	0,090619058	0,362476233	0,362476233
B	0,362476233	1	0,384464104	0,090619058	0,090619058	0,090619058	0,362476233
C	0,214921968	0,384464104	1	0,085968787	0,022654765	0,022654765	0,085968787
D	0,599388731	0,090619058	0,085968787	1	0,022654765	0,085968787	0,090619058
E	0,090619058	0,090619058	0,022654765	0,022654765	1	0,362476233	0,090619058
F	0,362476233	0,090619058	0,022654765	0,085968787	0,362476233	1	0,090619058
G	0,362476233	0,362476233	0,085968787	0,090619058	0,090619058	0,090619058	1

Tabel 3-28 Matriks *Similarity* Ternormalisasi

	A	B	C	D	E	F	G
A	1	0,362476233	0,214921968	0,599388731	0,090619058	0,362476233	0,362476233
B	0,362476233	1	0,384464104	0,090619058	0,090619058	0,090619058	0,362476233
C	0,214921968	0,384464104	1	0,085968787	0,022654765	0,022654765	0,085968787
D	0,599388731	0,090619058	0,085968787	1	0,022654765	0,085968787	0,090619058
E	0,090619058	0,090619058	0,022654765	0,022654765	1	0,362476233	0,090619058
F	0,362476233	0,090619058	0,022654765	0,085968787	0,362476233	1	0,090619058
G	0,362476233	0,362476233	0,085968787	0,090619058	0,090619058	0,090619058	1

Tabel 3-29 Hasil Filter *Treshold*

	A	B	C	D	E	F	G
A	1	0,362476233	0,214921968	0,599388731	0,090619058	0,362476233	0,362476233
B	0,362476233	1	0,384464104	0,090619058	0,090619058	0,090619058	0,362476233
C	0,214921968	0,384464104	1	0,085968787	0,022654765	0,022654765	0,085968787
D	0,599388731	0,090619058	0,085968787	1	0,022654765	0,085968787	0,090619058
E	0,090619058	0,090619058	0,022654765	0,022654765	1	0,362476233	0,090619058
F	0,362476233	0,090619058	0,022654765	0,085968787	0,362476233	1	0,090619058
G	0,362476233	0,362476233	0,085968787	0,090619058	0,090619058	0,090619058	1

3.3.2 Lex Rank Graph Based *Summarization*

Hasil perhitungan TF-IDF yang berupa matriks keterhubungan dan nilai *similarity* antarkalimat digunakan sebagai acuan dalam perhitungan bobot untuk masing-masing kalimat menggunakan algoritma LexRank.

Langkah pertama dalam menentukan bobot tiap kalimat ini adalah dengan memberikan nilai awal 1 kepada masing-masing kalimat seperti pada tabel 3.30 sebagai nilai *vertex* untuk menghitung *error rate* nilai *vertex* menggunakan persamaan 2.9.

Tabel 3-30 Inisialisasi awal nilai p(u)

U	p(u)
A	1
B	1
C	1
D	1
E	1
F	1
G	1

Kemudian selanjutnya dilakukan iterasi menggunakan persamaan 2.8.Hasilnya ditampilkan pada tabel 3.31 berikut.

Tabel 3-31 LexRank

u	v	Z	p(v)	sim(u,v)	Σsim(z,v)	temp	p(u)
a	b	a,c,d,g	1	0,5291503	2,3357823	0,22654092	0,3266524
	c	a,b,d,g	1	0,501996	2,2543195	0,222681838	
	d	b,c,f,g	1	1	3,0622926	0,326552732	
	f	a,d,e	1	0,5291503	1,5602965	0,33913442	
	g	a,b,c,d	1	0,5291503	2,0894468	0,253248976	
b	a	b,c,d,f,g	1	0,5291503	3,0894468	0,171276703	0,26081992
	c	a,b,d,g	1	0,7483315	2,2543195	0,331954485	
	d	b,c,f,g	1	0,5291503	3,0622926	0,172795464	
	g	a,b,c,d	1	0,5291503	2,0894468	0,253248976	
c	a	b,c,d,f,g	1	0,501996	3,0894468	0,162487347	0,25448545
	b	a,c,d,g	1	0,7483315	2,3357823	0,320377241	
	d	b,c,f,g	1	0,501996	3,0622926	0,16392817	
	g	a,b,c,d	1	0,501996	2,0894468	0,240253073	
d	a	b,c,d,f,g	1	1	3,0894468	0,323682544	0,39861139
	b	a,c,d,g	1	0,5291503	2,3357823	0,22654092	
	c	a,b,d,g	1	0,501996	2,2543195	0,222681838	
	e	F	1	0,2645751	0,5291503	0,5	

	f	a,d,e	1	0,501996	1,5602965	0,32173116	
	g	a,b,c,d	1	0,5291503	2,0894468	0,253248976	
e	d	b,c,f,g	1	0,2645751	3,0622926	0,086397732	0,18525839
	f	a,d,e	1	0,5291503	1,5602965	0,33913442	
f	a	b,c,d,f,g	1	0,5291503	3,0894468	0,171276703	0,3217093
	d	b,c,f,g	1	0,501996	3,0622926	0,16392817	
	e	F	1	0,5291503	0,5291503	1	
g	a	b,c,d,f,g	1	0,5291503	3,0894468	0,171276703	0,24042281
	b	a,c,d,g	1	0,5291503	2,3357823	0,22654092	
	c	a,b,d,g	1	0,501996	2,2543195	0,222681838	
	d	b,c,f,g	1	0,5291503	3,0622926	0,172795464	

Dari hasil perhitungan LexRank, didapatkan nilai bobot masing-masing *vertex* atau kalimat. Hasil perhitungan LexRank ini kemudian dilakukan pengurutan (*sorting*) berdasarkan nilai LexRank. Nilai LexRank menandakan tingkat keutamaan suatu kalimat terhadap dokumen. Seperti yang ditampilkan pada tabel 3.32 berikut.

Tabel 3-32 Hasil LexRank

u	p(u)
a	0,179317648
b	0,15355745
f	0,149612908
d	0,141298726
g	0,140254041
c	0,137985775
e	0,133451077

Kemudian, proses selanjutnya adalah proses pemotongan hasil ringkasan sesuai dengan nilai ukuran ringkasan yang ditentukan. Dalam perhitungan ini diberikan nilai presentase ringkasan sebesar 30%, maka didapatkan hasil ringkasan seperti pada tabel 3.33 berikut.

Tabel 3-33 Hasil Pemotongan Ringkasan

u	p(u)
a	0,179317648
b	0,15355745
F	0,149612908
D	0,141298726
G	0,140254041
C	0,137985775
E	0,133451077

Sehingga dapat diambil kesimpulan bahwa urutan ringkasan berdasarkan algoritma LexRank *summarization algorithm* adalah sebagai berikut.

Sebuah tiang beton milik perusahaan telekomunikasi setinggi 30 meter ambruk di Jalan Raya Raci, Pakal, Surabaya.
Dua bangunan rumah rusak terkena ambrukan tiang itu
Setelah terpotong, tiang itu diletakkan di pinggir jalan

3.3.3 Precision and Recall

Setelah didapatkan hasil ringkasan dokumen dari sistem, kemudian dihitung nilai precision dan recall berdasarkan hasil ringkasan yang dilakukan oleh manusia.

Dalam percobaan diberikan hasil ringkasan yang dilakukan manusia sebagai berikut :

Sebuah tiang beton milik perusahaan telekomunikasi setinggi 30 meter ambruk di Jalan Raya Raci, Pakal, Surabaya.
Dua bangunan rumah rusak terkena ambrukan tiang itu
tiang sepanjang 30 meter itu juga ambruk memenuhi badan jalan sehingga arus lalu lintas dialihkan melewati jalan perkampungan

Dari hasil perbandingan dapat diketahui jumlah *correct* : 2, jumlah *wrong* : 1, jumlah *missed* : 1. Sehingga dapat dihitung nilai *precision* dan *recall* sebagai berikut :

$$Precision = \frac{2}{2+1} = 0{,}6667$$

$$Recall = \frac{2}{2+1} = 0{,}6667$$


UNIVERSITAS BRAWIJAYA

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

Sebelum melakukan implementasi sistem, beberapa syarat harus disiapkan untuk memenuhi kebutuhan dari program yang akan implementasikan baik dari segi perangkat keras (*hardware*) maupun perangkat lunak (*software*) komputer.

4.1 Lingkungan Implementasi

Lingkungan implementasi meliputi lingkungan perangkat keras serta lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem pembuatan perangkat lunak peringkas dokumen otomatis adalah *laptop* dengan spesifikasi :

1. Prosesor AMD Turion™ 64 X2 Mobile Technology TL-58 1.90 GHz
2. RAM 2 GB DDR2.
3. *Harddisk* dengan kapasitas 120 GB.

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pembuatan perangkat lunak peringkas dokumen otomatis adalah :

1. Sistem Operasi Windows 7 ™ version 6.1 64-bit
2. NetBeans IDE 6.7.1
3. Java SDK 1.6.0_16

4.2 Implementasi Program

Berdasarkan perancangan perangkat lunak pada subbab 3.2 maka pada subbab ini akan dibahas mengenai implementasi dari perancangan tersebut.

4.2.1 Struktur Data

Dalam penerapan sistem ini, dibentuk struktur data berupa kelas-kelas utama yang menerapkan tiap proses dalam sistem. Kelas-

kelas tersebut antara lain kelas Dokumen, Stemming, Similarity dan Graf.

1. Kelas Dokumen

Kelas Dokumen digunakan untuk merepresentasikan dokumen teks menjadi bentuk vektor kata, vektor kalimat atau dalam bentuk array string paragraf, dengan beberapa fungsi yang digunakan untuk memproses dokumen. Kelas versi 1.00 ini dibuat pada tanggal 16 Mei 2008 oleh Drs. A. Ridok, M.Kom, staff pengajar Program Studi Ilmu Komputer Jurusan Matematika Universitas Brawijaya.

Dalam kelas ini terdapat fungsi getKataAll() untuk melakukan proses *Tokenizing* memecah dokumen menjadi kata-kata yang digunakan dalam proses TF-IDF *similarity*. Juga terdapat fungsi getUnik() yang digunakan untuk mendapatkan daftar kata Unik setelah melalui proses *Filtering StopWord* atau penghilangan kata yang sering muncul atau kurang penting.

2. Kelas *Stemming*

Kelas ini digunakan untuk melakukan proses *Stemming* atau penrubahan kata menjadi kata dasar. Dalam kelas ini dilakukan pemrosesan terhadap kata, setiap kata dipecah menjadi karakter-karakter penyusunnya. Kelas ini dibuat oleh Arga Dinata,ST.

Fungsi-fungsi dalam kelas ini dibedakan berdasarkan langkah-langkah proses stemming. Yaitu penghilangan imbuhan berupa awalan, awalan bertingkat, atau akhiran, sehingga didapatkan kata berupa kata dasar.

3. Kelas Similarity

Kelas Similarity digunakan untuk merepresentasikan proses TF-IDF dan vector space models *similarity*. Input dari proses ini adalah dokumen a dan b yang diinisialisasi pada fungsi konstruktor. Untuk masing-masing dokumen, digunakan tipe data Dokumen dari kelas Dokumen yang telah dibuat sebelumnya, sehingga proses *tokenizing, filtering* dan *stemming* dapat dilakukan dengan mudah.

Dalam kelas similarity ini, fungsi-fungsi yang digunakan adalah fungsi gabung_dokumen() untuk menggabung *token* dari dokumen a dan b, fungsi set_tf() untuk menentukan nilai *term frequency* dari masing-masing *token* yang terdapat pada masing-masing dokumen dan disimpan dalam *array*, fungsi set_weight() untuk menghitung *weight* atau bobot tiap *term* pada kalimat a dan b, hasil dari perhitungan *weight* ini juga disimpan dalam bentuk *array*,

kemudian fungsi yang terakhir adalah sim_analysis(), yaitu fungsi yang digunakan untuk menghitung nilai *similarity* dari dua kalimat yang diinputkan berdasarkan nilai bobot yang terdapat dalam *array*.

4. Kelas Graf

Kelas Graf merupakan representasi dari perhitungan LexRank. Input dari proses LexRank adalah matriks *similarity* antarkalimat, sehingga digunakan tipe data *Similarity* dan Dokumen yang telah dibuat dari kelas sebelumnya.

Fungsi-fungsi yang digunakan dalam kelas ini adalah fungsi kostruktor, fungsi setTreshold() untuk menentukan *treshold* sistem berdasarkan nilai yang diberikan user, fungsi set_matriks() untuk menghitung matriks *similarity* antarkalimat, dan fungsi scoreThis() untuk menghitung rangking tiap kalimat dengan rumus LexRank.

Dalam konstruktor diinputkan dokumen dan digunakan tipe data Dokumen untuk mendapatkan token kalimat-kalimat dalam dokumen yang disimpan dalam bentuk *array*. Kemudian dengan menggunakan kelas *similarity*, dilakukan proses perhitungan *similarity* antarkalimat sesuai dengan kalimat yang digunakan sebagai sumber nilai *similarity*.

Dalam implementasi dibuat 2 versi sistem dengan perbedaan terletak pada pengambilan kalimat yang menjadi sumber perhitungan *similarity*. Versi pertama, kalimat yang diambil adalah seluruh kalimat dalam dokumen dan tiap kalimat dilakukan perhitungan *similarity* dengan seluruh kalimat lainnya. Versi yang kedua, jika versi yang pertama melakukan proses LexRank terhadap seluruh isi dokumen, maka pada versi ini dilakukan perhitungan LexRank pada masing masing paragraf, sehingga masing-masing paragraf memberikan masing-masing kalimat intinya, yaitu kalimat yang memiliki nilai LexRank paling tinggi.

4.2.2 Tahap Tokenizing, Filtering dan Stemming

Pada tahap *preprocessing* ini, yang menjadi input dari proses adalah set dokumen yang terdiri atas paragraf, kalimat dan kata dalam bentuk file teks berekstensi .txt.

Tahap Tokenizing dilakukan dalam kelas Dokumen yang memuat beberapa fungsi tokenizing seperti fungsi getParagraf() yang terdapat pada gambar 4.1 untuk memecah Dokumen menjadi *Array* Paragraf, fungsi getKalimat() yang terdapat pada gambar 4.3 untuk memecah dokumen menjadi kalimat dan fungsi getKataAll() yang

terdapat pada gambar 4.2 untuk mendapatkan kata-kata dari dokumen.

```
protected String[] getParagraf(){
        return this.dokumen.trim().split("[\\n\\r]{2,}");
}
```

Gambar 4-1 Sourcecode Fungsi getParagraf()

```
protected Vector getKataAll(){
Vector kt = new Vector();
StringTokenizer t = new StringTokenizer(this.dokumen);
int jumlahKata = t.countTokens();
for (int i=0; i<jumlahKata; i++){
        String ktBaru = t.nextToken();
           if (!FindKey(this.stopList,ktBaru))
           kt.add(ktBaru.toLowerCase());
        }
return kt;
}
```

Gambar 4-2 Sourcecode fungsi getKataAll()

```
Protected Vector getKalimat(){
Vector blok = new Vector();
String kalimat;
BreakIterator bi = BreakIterator.getSentenceInstance();
bi.setText(this.dokumen);
int index = 0;
while (bi.next() != BreakIterator.DONE) {
kalimat = this.dokumen.substring(index, bi.current());
String regexp = "[\\.\\?\\!]|<J>.*</J>";
String replace = "";
Pattern p = Pattern.compile(regexp);
Matcher m = p.matcher(kalimat);
kalimat = m.replaceAll(replace);
blok.add(kalimat);
index = bi.current();}
return blok;}
```

Gambar 4-3 Sourcecode Fungsi getKalimat()

Proses Filtering dilakukan dalam fungsi getUnik() dengan melakukan filter terhadap kata dalam StopWord. *Sourcecode* fungsi getUnik() ditampilkan dalam gambar 4.4 berikut.

```
protected Vector getUnik(){
Object kAcuan;
Vector VB = new Vector();
for (int i=0; i< this.kata_kata.size(); i++){
kAcuan = this.kata_kata.elementAt(i);
if (VB.isEmpty()) VB.add(kAcuan);
else if (!FindKey(VB,kAcuan)) VB.add(kAcuan);}
        return VB;}
```

Gambar 4-4 Sourcecode fungsi getUnik()

Proses stemming dilakukan dalam kelas stemming. Penggunaan kelas ini adalah dengan melakukan *instance* terhadap kelas stemming. Kemudian untuk mendapatkan kata *root* hasil *stemming* dilakukan pemanggilan fungsi stemm() seperti pada gambar 4.5 berikut.

```
public String stemm(String kata) {
        String    status, prefix, sec_prefix;
        String[]  feedback;
        kata = particleRem(kata);
        kata = possessiveRem(kata);
        feedback  = firstPrefixRem(kata).split(";;");
        kata      = feedback[0];
        prefix    = feedback[1];
        feedback  = suffixRem(kata, prefix).split(";;");
        kata      = feedback[0];
        status    = feedback[1];
        if(status.equals("1")) {
feedback  = secondPrefixRem(kata, prefix).split(";;");
                kata      = feedback[0];
                }
        else {
feedback       =    secondPrefixRem(kata,    prefix).split
(";;");
        kata           = feedback[0];
        sec_prefix  = feedback[1];
        feedback              =              suffixRem(kata,
sec_prefix).split(";;");
                }
                return feedback[0];
        }
```

Gambar 4-5 Sourcecode fungsi stemm()

Dalam fungs stemm tersebut terdapat beberapa fungsi stemming seperti particleRem(kata) untuk menghilangkan partikel (-lah, -kah, -pun, -tah), seperti gambar 4.6 berikut.

```
public String particleRem(String kata) {
String suku_kata;
if(kata.length() > 3) {
suku_kata = kata.substring(kata.length() - 3);
if(suku_kata.equals("lah") || suku_kata.equals("kah") ||
suku_kata.equals("pun") || suku_kata.equals("tah")) {
kata = kata.substring(0, kata.length() - 3);
                        }}
                return kata;
        }
```

Gambar 4-6 Sourcecode fungsi particleRem()

Selanjutnya adalah fungsi possesiveRem() yang berfungsi untuk menghilangkan kata ganti milik (-ku, -mu, -nya), seperti yang ditampilkan pada gambar 4.7 berikut.

```
public String possessiveRem(String kata) {
String suku_kata;
if(kata.length() > 2) {
       suku_kata = kata.substring(kata.length() - 2);
       if(suku_kata.equals("ku") ||
       suku_kata.equals("mu")) {
       kata = kata.substring(0, kata.length() - 2);
       }
else {
       suku_kata = kata.substring(kata.length() - 3);
       if(suku_kata.equals("nya")) {
       kata = kata.substring(0, kata.length() - 3);
              }
       }
}
return kata;
}
```

Gambar 4-7 Sourcecode fungsi possesiveRem()

Selanjutnya adalah fungsi firstPrefixRem() untuk menghilangkan awalan tahap 1 (ber,- di,- ke-, meng-, peng-, per-, ter-, meny-, mem-, men-, me-, peny-, pem-, pen-, pe-, bel-, be-, pel-, pe-, te-). Seperti yang ditampilkan pada gambar 4.9. dan fungsi secondPrefixRem() yang digunakan untuk menghapus awalan bertingkat (memper-, diper-) seperti yang ditunjukkan pada gambar 4.8.

Yang terakhir dari proses *stemming* yaitu fungsi suffixRem() digunakan untuk menghapus akhiran (-i, -kan, –an) ditunjukkan pada gambar 4.10.

```
public String secondPrefixRem(String kata, String prefix) {
       String suku_kata, sec_prefix = "null";
       if(prefix.equals("di") || prefix.equals("mem")) {
       if(kata.length() > 3) {
       suku_kata = kata.substring(0, 3);
       if((prefix.equals("di") || prefix.equals("ke") ||
   prefix.equals("mem") || prefix.equals("ter")) &&
    (suku_kata.equals("ber") || suku_kata.equals("per"))) {
kata = kata.substring(3, kata.length());
sec_prefix  = suku_kata;
       }}}
else if(!prefix.equals("null")){
return firstPrefixRem(kata);
}
              return kata.concat(";;" + sec_prefix);
       }
```

Gambar 4-8 sourcecode fungsi secondPrefixRem()

```
public String firstPrefixRem(String kata) {
int     done = 0;
String  suku_kata, prefix = "null";
if(kata.length() > 2) {
       suku_kata = kata.substring(0, 2);
       if(suku_kata.equals("di") ||
        suku_kata.equals("ke")) {
       prefix  = suku_kata;
       kata    = kata.substring(2, kata.length());
               done    = 1;    }
       else {
       suku_kata = kata.substring(0, 3);
       if(suku_kata.equals("ber") ||
        suku_kata.equals("per") ||
       suku_kata.equals("ter")) {
       prefix  = suku_kata;
       kata    = kata.substring(3, kata.length());
               done    = 1;    }
       else {
if(kata.length() > 4) {
       suku_kata = kata.substring(0, 4);
       if(suku_kata.equals("meng") ||
      suku_kata.equals("meny") ||
suku_kata.equals("peng") ||
suku_kata.equals("peny")) {
       prefix  = suku_kata;
       kata    = kata.substring(4, kata.length());
       done    = 1;    }
}}}
if(done == 0) {
suku_kata = kata.substring(0, 3);
if(suku_kata.equals("mem") || suku_kata.equals("men") ||
   suku_kata.equals("pem") || suku_kata.equals("pen") ||
   suku_kata.equals("bel") || suku_kata.equals("pel")) {
       prefix  = suku_kata;
       kata    = kata.substring(3, kata.length());   }
else {
       suku_kata = kata.substring(0, 2);
       if(suku_kata.equals("me") ||
suku_kata.equals("pe") ||
suku_kata.equals("be") ||
suku_kata.equals("te")) {
prefix  = suku_kata;
kata    = kata.substring(2, kata.length());
       }}}}
return kata.concat(";;" + prefix);
}
```

Gambar 4-9 Sourcecode fungsi firstPrefixRem()

```
public String suffixRem(String kata, String prefix) {
            int    hasSuffix = 0;
            String  suku_kata;
            if(kata.length() > 2) {
       if(kata.charAt(kata.length() - 1) == 'i') {
       if(!prefix.equals("ber") && !prefix.equals("ke")
  && !prefix.equals("peng")) {
       hasSuffix  = 1;
       kata       = kata.substring(0, kata.length() - 1);
       }}
       else {
       suku_kata = kata.substring(kata.length() - 3);
       if(suku_kata.equals("kan")) {
       if(!prefix.equals("ke") &&
      !prefix.equals("peng")) {
       hasSuffix  = 1;
       kata       = kata.substring(0, kata.length() - 3);
       }}
       else {
       suku_kata = kata.substring(kata.length() - 2);
       if(suku_kata.equals("an")) {
       if(!prefix.equals("di") && !prefix.equals("meng")
&&!prefix.equals("ter")) {
       hasSuffix  = 1;
       kata       = kata.substring(0, kata.length() - 2);
             }}}}}
return kata.concat(";;" + hasSuffix); }
```

Gambar 4-10 sourcecode fungsi suffixRem()

4.2.3 Tahap TF-IDF *Similarity*

Tahap selanjutnya adalah melakukan penghitungan nilai *similarity* antarkalimat menggunakan algoritma TF-IDF dan *vector space models*. Untuk melakukan proses perhitungan ini, digunakan kelas *similarity*. Penggunaannya adalah dengan melakukan instance terhadap kelas *similarity* dengan parameter input berupa String a yang merupakan String kalimat a dan String b yang merupakan String kalimat b. Fungsi Konstruktor kelas *similarity* digambarkan pada gambar 4.7 berikut.

```
public similarity(String a, String b){
        this.dokumen = new Dokumen[2];
        this.jumlah_dok = this.dokumen.length;
        this.dokumen[0] = new Dokumen(a, stop);
        this.dokumen[1] = new Dokumen(b, stop);
        this.gabung_dokumen();
        this.set_tf();
        this.set_weight();
        this.sim_analysis();    }
```

Gambar 4-11 Source Code Konstruktor kelas *similarity*

```
public void gabung_dokumen(){
String temp = "";
for (int i=0;i<jumlah_dok;i++){
 for (int j=0;j<this.dokumen[i].getKataAll().size();j++){
temp+=(this.dokumen[i].getKataAll().elementAt(j)+" ");}}
        this.all = new Dokumen(temp, stop);
        this.jumlah_term=this.all.getUnik().size();
        this.tf=new int[jumlah_dok][jumlah_term];
        this.w=new double[jumlah_dok][jumlah_term];
        this.D=new double[jumlah_dok];
        for (int y=0;y<jumlah_dok;y++){
            for (int z=0;z<jumlah_term;z++){
                this.tf[y][z]=0;
                this.w[y][z]=0;}
            this.D[y]=0;}
        this.df=new int[jumlah_term];
        this.QdotD=0;
        this.Q=0;
        this.fq=0;
        for (int j=0;j<jumlah_term;j++){
            this.df[j]=0;}}
```

Gambar 4-12 Sourcecode fungsi gabung_dokumen()

Kemudian seluruh token (kata) dari dua kalimat yang diinputkan digabung dan disimpan dalam tipe data dokumen yang baru. Proses penggabungan token dilakukan dalam fungsi gabung_dokumen seperti pada gambar 4.8.

Setelah token tergabung, dilakukan proses penghitungan nilai *term frequency* untuk masing-masing kata pada dokumen. Fungsi set_tf() digambarkan pada gambar 4.9 berikut.

```
public void set_tf(){
for (int i=0;i<jumlah_term;i++){
  for (int j=0;j<jumlah_dok;j++){
    for (int k=0;k<this.dokumen[j].getUnik().size();k++){
 if
(this.dokumen[j].getKataAll().elementAt(k).equals(this.all.getU
nik().elementAt(i))){
   this.tf[j][i]+=1;
   if (this.df[i]<this.jumlah_dok){
     this.df[i]+=1; }
   if (tf[j][i]>this.fq){
     this.fq=tf[j][i];
}}}}}}
```

Gambar 4-13 Sourcecode fungsi set_tf()

Dari hasil perhitungan term frequency, dilakukan perhitungan bobot untuk masing-masing kata terhadap masing-

masing dokumen dengan fungsi set_weight() seperti pada gambar 4.10 berikut.

```
public void set_weight(){
for (int h=0;h<this.jumlah_dok;h++){
for (int i=0;i<this.jumlah_term;i++){
if (fq!=0 && df[i]!=0){
w[h][i]=(this.tf[h][i]/this.fq)*Math.log10(jumlah_dok-
1/df[i]);}
else{
w[h][i]=0;}
}}}
```

Gambar 4-14 Sourcecode fungsi set_weight()

Langkah terakhir adalah menghitung nilai *similarity* antara kalimat a dan b dengan fungsi sim_analysis() yang terdapat pada gambar 4.11 berikut.

```
public void sim_analysis(){
        double temp;
        QstarD = 1;
        QdotD = 1;
        ////////////////////////////////////Di
        for (int j=0;j<jumlah_dok;j++){
            temp =0;
            for (int i=0;i<jumlah_term;i++){
                temp+=Math.pow(w[j][i],2);
            }
            if (temp!=0){
                this.D[j]=Math.sqrt(temp);
            }
            else{
                this.D[j]=1;
            }
            QstarD *= this.D[j];
        }

        ////////////////////////////////////Q.D
        for (int k=0;k<jumlah_dok;k++){
            temp=0;
            for (int l=0;l<jumlah_term;l++){
                temp+=w[k][l];
            }
            QdotD *= temp;
        }
        ////////////////////////////////////similarity

        this.sim = QdotD / QstarD;
        }
```

Gambar 4-15 Sourcecode fungsi sim_analysis()

4.2.4 Tahap Lex Rank *Summarization*

Tahap yang terakhir adalah tahap LexRank yang melakukan perangkingan terhadap masing-masing kalimat dengan input berupa matriks *similarity* dan keterhubungan antarkalimat. Untuk melakukan penghitungan LexRank digunakan kelas graf. Penggunaannya adalah dengan melakukan instance terhadap kelas graf dengan parameter input konstruktor berupa string dokumen yang akan diproses. Fungsi konstruktor ditunjukkan pada gambar 4.12 berikut.

```
public graf(String a){
this.logging = new log();
this.logging.clear();
this.source = new Dokumen(a, this.stop);
this.jum_kalimat = this.source.getKalimat().size();
this.jum_paragraf = this.source.getParagraf().length;
this.jum_kata = this.source.getKataAll().size();
this.kalimat= new Vector(jum_kalimat);
for (int i=0;i<jum_kalimat;i++){
this.kalimat.addElement(this.source.getKalimat().elementAt(i));
}
this.G = new double[jum_kalimat][jum_kalimat];
this.graf_weight = new double[jum_kalimat];
this.degree = new int[jum_kalimat];
this.logging.insert_log("Inisialisasi dokumen dan jumlah
kalimat...\nsukses...\n");
}
```

Gambar 4-16 Sourcecode Fungsi Konstruktor Kelas Graf

Kemudian digunakan fungsi set_matriks() untuk melakukan perhitungan matriks keterhubungan antarkalimat, seperti yang ditunjukkan pada gambar 4.13. Dalam fungsi set_matriks() ini digunakan *instance* dari kelas *similarity* untuk mencari nilai kemiripan antara dua kalimat.

```
public void set_matriks(){
similarity sim;
double treshold = 0.2;
double max = 0;
for (int i=0;i<jum_kalimat;i++){
for (int j=0;j<jum_kalimat;j++){
sim = new similarity(
this.kalimat.elementAt(i).toString(),
this.kalimat.elementAt(j).toString());
                if (i==j){
                    this.G[i][j]=1;
                }
                else{
                    this.G[i][j]=sim.sim;
                    if (sim.sim>max){
                        max = sim.sim;
                    }}}}
        for (int i=0;i<jum_kalimat;i++){
            for (int j=0;j<jum_kalimat;j++){
                this.G[i][j] =  this.G[i][j]/max;
            }}
        for (int k=0;k<jum_kalimat;k++){
            this.degree[k]=0;
            for (int l=0;l<jum_kalimat;l++){
                if (G[k][l]>0.1){
                    this.degree[k]++;
                }
            }
            this.graf_weight[k]=1;
        }
this.logging.insert_log("Inisialisasi  matriks  similarity
antarkalimat utama...\nsukses...\n");
    }
```

Gambar 4-17 Fungsi set_matriks()

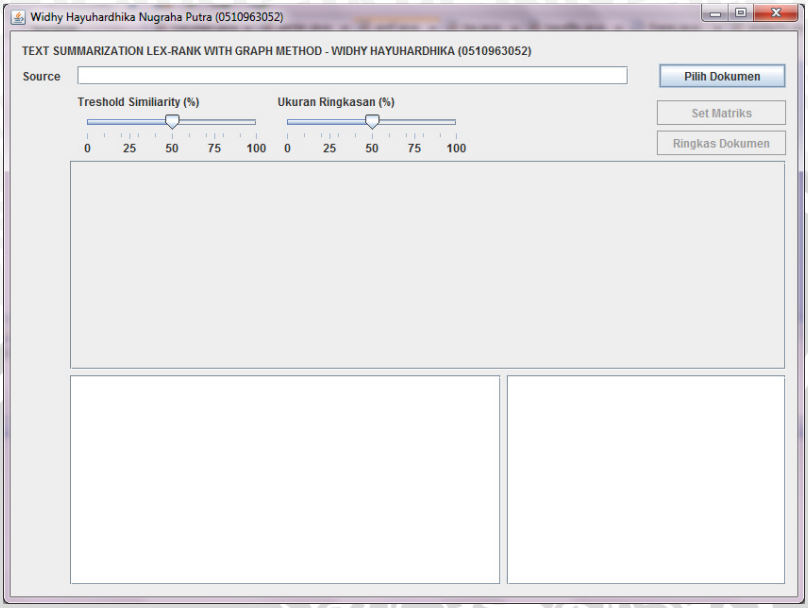
Kemudian untuk menghitung rangking tiap kalimat berdasarkan matriks *similarity* antarkalimat digunakan fungsi scoreThis() seperti yang ditunjukkan pada gambar 4.14.

```
public void scoreThis(){
double temp_1,temp_2,d;
d=0.85;
for (int j=0;j<this.jum_kalimat;j++){
temp_1=0;
   for (int k=0;k<this.jum_kalimat;k++){
     temp_2=0;
     for (int l=0;l<this.jum_kalimat;l++){
       if (G[k][l]>=this.treshold_sim){
          temp_2+=G[k][l]; }}
temp_1+=(G[j][k]/temp_2)*this.graf_weight[k];}
this.graf_weight[j]=d/this.jum_kalimat+(1-d)+d*temp_1;}
double t;
String a;
  for (int j=0;j<this.jum_kalimat;j++){
   for (int k=0;k<this.jum_kalimat;k++){
if (this.graf_weight[j]>this.graf_weight[k]){
t=this.graf_weight[j];
this.graf_weight[j]=this.graf_weight[k];
this.graf_weight[k]=t;
a=this.kalimat.elementAt(j).toString();
this.kalimat.setElementAt(this.kalimat.elementAt(k), j);
                  this.kalimat.setElementAt(a, k);
               }}}
this.logging.insert_log("skoring vertex..\nsukses...\n");
    }
```

Gambar 4-18 Sourcecode Fungsi scoreThis()

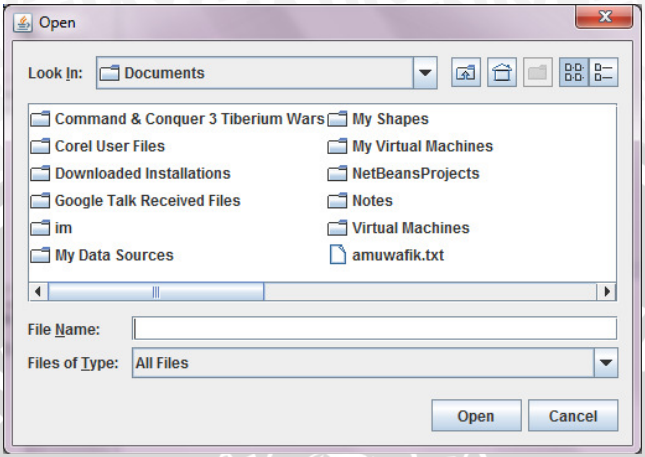
4.3 Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada sub bab 3.3 maka dihasilkan antarmuka yang ditunjukkan pada gambar 4.4. Pada form utama terdapat *field* ”source”, tombol ”Pilih Dokumen”, *slider* ”*treshold similarity*”, *slider* ”ukuran ringkasan”, tombol ”set matriks”, tombol ”ringkas dokumen”, tabel ”matriks *similarity*”, *text area* ”hasil” dan *text area* ”log”.

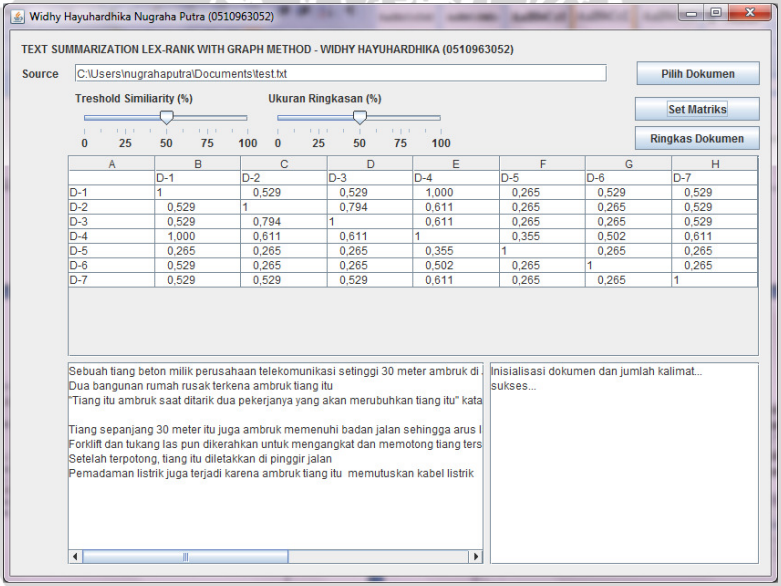


Gambar 4-19 Form Utama

field ”source”, tombol ”Pilih Dokumen” yang digunakan untuk menentukan dokumen mana yang akan dilakukan proses peringkasan dokumen. Jika tombol pilih dokumen ditekan, maka akan muncul *dialog box* ”open” yang ditampilkan pada gambar 4.5. *User* akan melakukan pemilihan dokumen yang akan dijadikan sumber peringkasan dokumen, setelah *user* menentukan pilihan dan menekan tombol ”Open” pada *dialog box*, maka *field* ”source” akan berisi *path* file teks yang dijadikan sumber ringkasan. Jika proses pemilihan dokumen berhasil dilakukan, maka tombol ”Set Matriks” akan muncul dan *text area* ”hasil” akan berisi dokumen yang dipilih *user*. Kemudian *user* harus menekan tombol ”Set Matriks” untuk membentuk tabel keterhubungan antarkalimat. Hasil pembentukan matriks ditampilkan pada gambar 4.6.



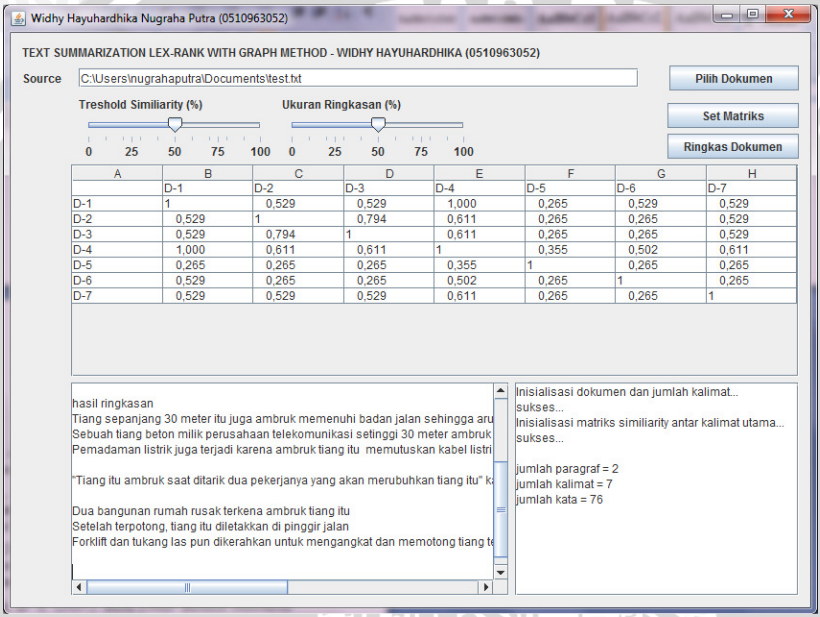
Gambar 4-20 Dialog Box "open"



Gambar 4-21 Hasil Set Matriks

Setelah matriks *similarity* antarkalimat terbentuk, maka tombol "Ringkas Dokumen" akan muncul dan tabel "matriks *similarity*" akan berisi nilai *similarity* antarkalimat. Langkah

selanjutnya adalah user menentukan nilai *treshold similarity* dan ukuran ringkasan dengan menggeserkan *slider* "*treshold similarity*" dan *slider* "ukuran ringkasan". Jika nilai *treshold similarity* dan ukuran ringkasan telah ditentukan, ditekan tombol "ringkasan dokumen" dan dihasilkan ringkasan dokumen yang ditampilkan pada *text area* "hasil" seperti yang tampak pada gambar 4.7.



Gambar 4-22 Hasil Ringkasan

4.4 Analisa Hasil

4.4.1 Hasil Uji Coba Versi 1

Hasil uji coba pada aplikasi versi 1 ditunjukkan pada tabel 4.1 hingga 4.4 berikut.

Tabel 4-1 Hasil Perhitungan *Precision and Recall* dengan *treshold* ringkasan 25% pada aplikasi versi 1

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,667	0,222	0,500	0,333	0,500	0,333	0,500	0,333	0,500	0,333
37	0,333	0,250	0,167	0,125	0,167	0,143	0,167	0,125	0,333	0,250
22	0,286	0,250	0,143	0,125	0,143	0,125	0,286	0,250	0,286	0,250
26	0,500	0,273	0,429	0,273	0,429	0,273	0,429	0,273	0,429	0,273
21	0,667	0,444	0,500	0,333	0,500	0,333	0,333	0,222	0,500	0,333
29	0,750	0,429	0,250	0,143	0,250	0,143	0,250	0,143	0,250	0,143
32	0,500	0,429	0,333	0,286	0,167	0,143	0,167	0,143	0,167	0,143
35	0,375	0,250	0,375	0,250	0,375	0,250	0,375	0,250	0,375	0,250
25	0,500	0,333	0,333	0,167	0,333	0,167	1,000	0,500	1,000	0,500
19	0,250	0,231	0,667	0,615	0,300	0,231	0,600	0,462	0,400	0,308
Rata-rata	**0,483**	**0,311**	**0,370**	**0,265**	**0,316**	**0,214**	**0,411**	**0,270**	**0,424**	**0,278**

Tabel 4-2 Perhitungan Precision and Recall dengan *treshold* ringkasan 50% pada aplikasi versi 1

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,462	0,667	0,615	0,833	0,615	0,889	0,615	0,889	0,615	0,889
37	0,250	0,375	0,250	0,375	0,333	0,500	0,333	0,500	0,333	0,500
22	0,200	0,375	0,133	0,250	0,200	0,375	0,200	0,375	0,200	0,375
26	0,538	0,636	0,385	0,455	0,385	0,455	0,385	0,455	0,385	0,455
21	0,385	0,556	0,462	0,667	0,385	0,556	0,308	0,444	0,385	0,556
29	0,625	0,714	0,500	0,571	0,500	0,571	0,500	0,571	0,500	0,571
32	0,400	0,571	0,250	0,429	0,250	0,429	0,167	0,286	0,083	0,143
35	0,250	0,333	0,500	0,667	0,533	0,667	0,563	0,750	0,563	0,750
25	0,625	0,833	0,571	0,667	0,571	0,667	0,429	0,429	0,429	0,500
19	0,316	0,462	0,316	0,462	0,474	0,692	0,316	0,462	0,474	0,692
Rata-rata	**0,405**	**0,552**	**0,398**	**0,537**	**0,425**	**0,580**	**0,381**	**0,516**	**0,397**	**0,543**

Tabel 4-3 Perhitungan Precision and Recall dengan *treshold* ringkasan 75% pada aplikasi versi 1

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,368	0,778	0,368	0,857	0,368	0,778	0,368	0,778	0,368	0,778
37	0,333	0,750	0,222	0,500	0,389	0,875	0,294	0,625	0,278	0,625
22	0,273	0,750	0,273	0,750	0,273	0,750	0,273	0,750	0,273	0,750
26	0,350	0,636	0,400	0,727	0,350	0,636	0,350	0,636	0,350	0,636
21	0,316	0,667	0,316	0,667	0,316	0,667	0,316	0,667	0,350	0,778
29	0,417	0,714	0,500	0,857	0,417	0,714	0,500	0,857	0,500	0,857
32	0,235	0,571	0,294	0,714	0,294	0,714	0,176	0,429	0,176	0,429
35	0,261	0,500	0,273	0,500	0,261	0,500	0,348	0,667	0,348	0,667
25	0,500	0,833	0,400	0,667	0,400	0,667	0,400	0,667	0,400	0,667
19	0,240	0,462	0,360	0,692	0,320	0,615	0,458	0,846	0,400	0,769
Rata-rata	**0,329**	**0,666**	**0,341**	**0,693**	**0,339**	**0,692**	**0,348**	**0,692**	**0,344**	**0,696**

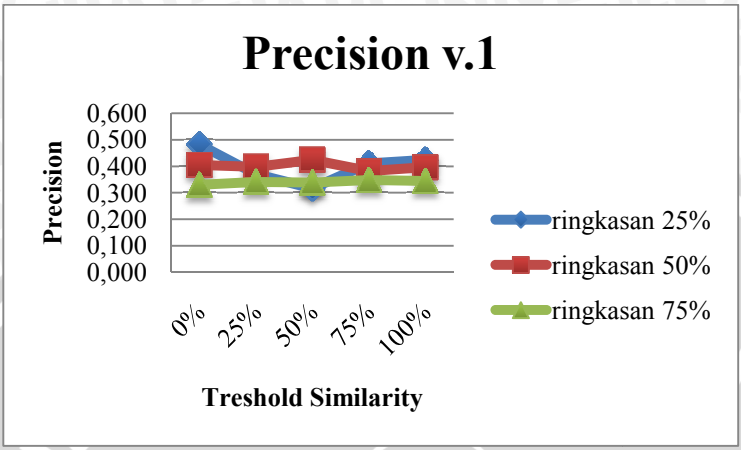
Hasil uji coba pertama (tabel 4.1) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi dengan versi 1. Sesuai dengan rancangan yang ada pada subbab 3.1, percobaan pertama digunakan *treshold summary* atau panjang ringkasan sebesar 25% dari dokumen asli.

Tabel kedua (tabel 4.2) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi versi 1 dengan *treshold summary* atau panjang ringkasan 50% dari dokumen asli.

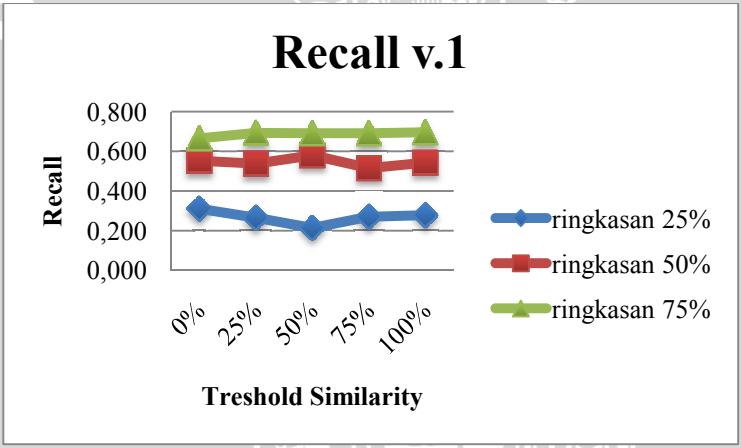
Sedangkan tabel ketiga (tabel 4.3) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi versi 1 dengan *treshold summary* atau panjang ringkasan 75% dari teks asli.

Dari hasil uji coba terhadap aplikasi versi 1 ini, diketahui rata-rata *precision* sebesar 0,3197 dan rata-rata *recall* sebesar 0,500. Rata-rata nilai *precision* dan *recall* untuk masing-masing *treshold*, *treshold similarity* maupun *treshold summary* dinyatakan dalam grafik akan tampak seperti gambar 4.25 dan 4.26 berikut, dimana sumbu x menyatakan besarnya *treshold similarity* dan sumbu y menyatakan besarnya nilai *precision/recall*.

Rata-rata nilai *precision* tertinggi 0,483 terjadi pada saat percobaan dengan nilai *treshold similarity* 0% dan ukuran ringkasan 25% dokumen awal. Sedangkan rata-rata nilai *recall* terendah 0,214 terjadi pada saat percobaan dengan nilai *treshold similarity* 50% dan ukuran ringkasan 25% dokumen awal.



Gambar 4-23 grafik precision pada aplikasi versi 1



Gambar 4-24 grafik recall pada aplikasi versi 1

4.4.2 Hasil uji coba versi 2

Sedangkan hasil uji coba pada aplikasi versi 2 ditunjukkan pada tabel 4.4 hingga 4.6 berikut.

Tabel 4-4 Perhitungan Precision and Recall dengan *treshold* ringkasan 25% pada aplikasi versi 2

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,375	0,333	0,375	0,333	0,500	0,444	0,500	0,444	0,500	0,444
37	0,375	0,375	0,375	0,375	0,625	0,625	0,500	0,500	0,500	0,500
22	0,111	0,125	0,111	0,125	0,222	0,250	0,111	0,125	0,000	0,000
26	0,333	0,273	0,444	0,364	0,444	0,364	0,667	0,545	0,444	0,364
21	0,500	0,800	0,500	0,444	0,444	0,444	0,500	0,444	0,500	0,444
29	0,800	0,571	0,600	0,429	0,400	0,286	0,400	0,286	0,400	0,286
32	0,333	0,286	0,333	0,286	0,333	0,286	0,333	0,286	0,333	0,286
35	0,222	0,167	0,333	0,250	0,375	0,250	0,444	0,333	0,333	0,250
25	0,167	0,200	0,167	0,167	0,333	0,333	0,500	0,500	0,500	0,500
19	0,375	0,231	0,375	0,231	0,375	0,231	0,375	0,231	0,375	0,231
Rata-rata	**0,359**	**0,336**	**0,361**	**0,300**	**0,405**	**0,351**	**0,433**	**0,369**	**0,389**	**0,330**

Tabel 4-5 Perhitungan Precision and Recall dengan *treshold* ringkasan 50% pada aplikasi versi 2

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,333	0,556	0,333	0,556	0,333	0,556	0,400	0,667	0,333	0,556
37	0,500	0,875	0,500	0,875	0,500	0,875	0,571	1,000	0,571	1,000
22	0,176	0,375	0,176	0,375	0,235	0,500	0,235	0,500	0,235	0,500
26	0,400	0,545	0,400	0,545	0,333	0,455	0,533	0,727	0,333	0,455
21	0,357	0,556	0,357	0,556	0,357	0,556	0,357	0,556	0,357	0,556
29	0,400	0,571	0,400	0,571	0,400	0,571	0,400	0,571	0,400	0,571
32	0,357	0,714	0,357	0,714	0,286	0,571	0,286	0,571	0,214	0,429
35	0,412	0,583	0,412	0,583	0,412	0,583	0,412	0,583	0,412	0,583
25	0,375	0,500	0,375	0,500	0,375	0,500	0,375	0,500	0,375	0,500
19	0,188	0,231	0,313	0,385	0,313	0,385	0,250	0,308	0,313	0,385
Rata-rata	**0,350**	**0,551**	**0,362**	**0,566**	**0,354**	**0,555**	**0,382**	**0,598**	**0,354**	**0,553**

Tabel 4-6 Perhitungan Precision and Recall dengan *treshold* ringkasan 75% pada aplikasi versi 2

KODE DOKUMEN	*TRESHOLD SIMILARITY* (%)									
	0		25		50		75		100	
	P	R	P	R	P	R	P	R	P	R
36	0,300	0,667	0,421	0,889	0,350	0,778	0,300	0,667	0,350	0,778
37	0,444	1,000	0,444	1,000	0,444	1,000	0,444	1,000	0,400	0,750
22	0,227	0,625	0,227	0,625	0,227	0,625	0,227	0,625	0,227	0,625
26	0,350	0,538	0,400	0,727	0,300	0,545	0,350	0,538	0,400	0,727
21	0,350	0,778	0,350	0,778	0,350	0,778	0,350	0,778	0,350	0,778
29	0,333	0,571	0,500	0,857	0,417	0,714	0,333	0,571	0,417	0,714
32	0,333	0,857	0,333	0,857	0,294	0,714	0,333	0,857	0,333	0,714
35	0,320	0,667	0,320	0,667	0,400	0,833	0,360	0,750	0,385	0,833
25	0,500	1,000	0,417	0,833	0,417	0,833	0,417	0,833	0,417	0,833
19	0,375	0,692	0,333	0,615	0,333	0,615	0,375	0,692	0,333	0,615
Rata-rata	**0,353**	**0,740**	**0,375**	**0,785**	**0,353**	**0,744**	**0,349**	**0,731**	**0,361**	**0,737**

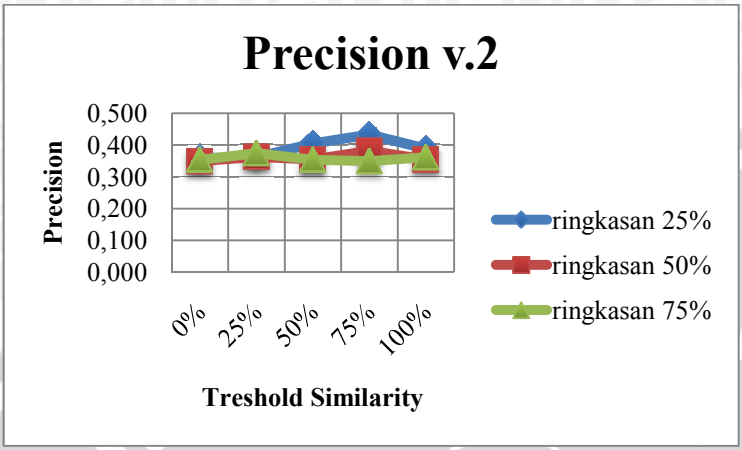
Hasil uji coba pertama (tabel 4.4) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi dengan versi 2. Sesuai dengan rancangan yang ada pada subbab 3.1, percobaan pertama digunakan *treshold summary* atau panjang ringkasan sebesar 25% dari dokumen asli.

Tabel kedua (tabel 4.5) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi versi 2 dengan *treshold summary* atau panjang ringkasan 50% dari dokumen asli.

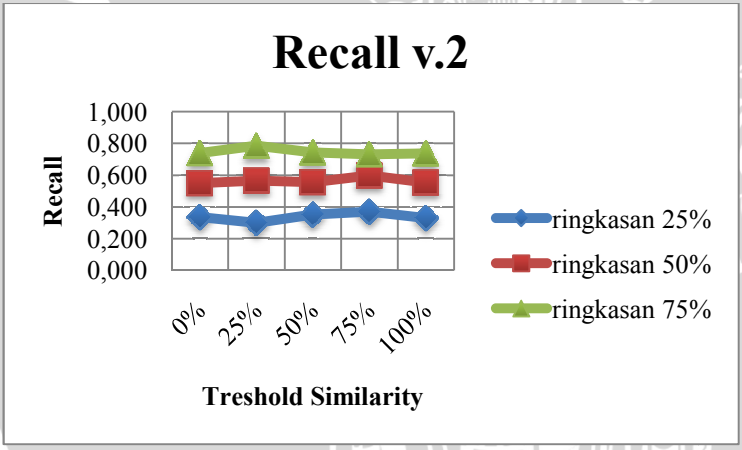
Sedangkan tabel ketiga (tabel 4.6) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi versi 2 dengan *treshold summary* atau panjang ringkasan 75% dari teks asli.

Dari hasil uji coba terhadap aplikasi versi 2 ini, diketahui rata-rata *precision* sebesar 0,369 dan rata-rata *recall* sebesar 0,549. Rata-rata nilai *precision* dan *recall* untuk masing-masing *treshold*, *treshold similarity* maupun *treshold summary* dinyatakan dalam grafik akan tampak seperti gambar 4.25 dan 4.26 berikut, dimana sumbu x menyatakan besarnya *treshold similarity* dan sumbu y menyatakan besarnya nilai *precision/recall*.

Rata-rata nilai *precision* tertinggi 0,433 terjadi pada saat percobaan dengan nilai *treshold similarity* 75% dan ukuran ringkasan 25% dokumen awal. Sedangkan rata-rata nilai *recall* terendah 0,300 terjadi pada saat percobaan dengan nilai *treshold similarity* 25% dan ukuran ringkasan 25% dokumen awal.



Gambar 4-25 grafik precision pada versi 2



Gambar 4-26 grafik recall pada versi 2

4.4.3 Analisa Hasil Secara Keseluruhan

Secara keseluruhan, metode LexRank menghasilkan rata-rata nilai precision sebesar 0,312 dan rata-rata nilai recall sebesar 0,437

Dari hasil uji coba yang dilakukan, diketahui beberapa hal yang berpengaruh terhadap data uji coba yang dihasilkan. Antara lain metode *stemming* yang dipakai, dan data uji coba yang digunakan.

Masih lemahnya metode *stemming* yang dilakukan dalam aplikasi menjadi hal yang sangat berpengaruh terhadap data hasil uji coba, dimana *stemming* yang dilakukan tidak menggunakan kamus kata dasar sebagai acuannya. *Stemming* yang dilakukan dalam aplikasi secara langsung memotong tiap awalan atau akhiran tanpa melakukan pengecekan apakah kata yang diproses adalah kata dasar. Hal ini mengakibatkan jika ada kata dasar yang mengandung suku kata yang sama dengan salah satu imbuhan, hasil *stemming* menjadi tidak valid. Seperti misalnya kata "ambrukan" yang tersusun atas kata dasar "ambruk" dan mendapat akhiran "-an" dan kata "mengambrukkan" yang tersusun atas kata dasar "ambruk" dan mendapat akhiran "-kan". Jika dilakukan *stemming* pada aplikasi ini, maka hasil yang didapat dari kata "ambrukan" adalah kata dasar "ambru" karena dianggap memiliki akhiran "-kan", beda hasilnya pada kata "mengambrukkan" akan dihasilkan kata dasar "ambruk".

Kekurangan pada metode *stemming* ini akan berpengaruh terhadap nilai *similarity* antarkalimat. Karena dua kata yang sebenarnya satu kata dasar tadi tidak terhitung sebagai kata atau *term* yang sama, sehingga akan mempengaruhi nilai *similarity* yang dihasilkan. Disamping itu, metode penentuan milai *similarity* antarkalimat yang digunakan juga berpengaruh besar terhadap rangking kalimat topik.

Hal lain yang berpengaruh adalah data uji coba yang digunakan. Dari beberapa percobaan yang dilakukan, sistem peringkas dokumen ini akan mengenali kalimat topik sebagai kalimat yang paling banyak mengandung kata-kata penting dari dokumen. Sehingga peringkas dokumen ini akan bekerja optimal pada dokumen-dokumen uji coba yang memiliki kalimat topik berupa deskripsi panjang dari sebuah paragraf atau dokumen. Sistem tidak akan berhasil mengekstrak kalimat topik yang berupa gambaran singkat isi dari paragraf atau dokumen.

Jika dinilai secara proses, aplikasi versi 2 lebih baik daripada versi 1. hal ini dikarenakan ringkasan yang dihasilkan dari aplikasi versi 2 adalah kalimat-kalimat yang memiliki rangking LexRank paling tinggi dan masing-masing paragraf akan menyumbangkan kalimat topiknya sehingga ringkasan yang dihasilkan lebih merata. Dari sisi waktu proses, versi 2 juga dinilai lebih baik karena proses perangkingan dilakukan pada masing-masing paragraf dan

perhitungan *similarity* antarkalimat hanya dilakukan dalam satu paragraf saja dimana jumlah kalimatnya relatif lebih sedikit jika dibandingkan dengan versi 1 yang melakukan perhitungan nilai *similarity* antara satu kalimat dengan seluruh kalimat dalam dokumen.

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang dihasilkan dalam skripsi ini adalah :

1. Telah dibuat sebuah sistem peringkas dokumen teks otomatis yang mengiplementasikan metode LexRank : *graph based summarization algorithm* untuk dokumen teks berbahasa indonesia. Dan telah dilakukan pengujian terhadap sistem yang dibuat dengan mengujikan *treshold similarity* dan *treshold summary* yang beragam.
2. Sistem berhasil melakukan peringkasan dokumen teks berbahasa Indonesia dengan uji coba pertama menggunakan aplikasi versi 1 dihasilkan rata-rata *precision* sebesar 0,380 dan *recall* sebesar 0,500. Dan uji coba kedua menggunakan aplikasi versi 2 dihasilkan rata-rata *precision* sebesar 0,369 dan *recall* sebesar 0,549.
3. Dalam sistem yang telah dibuat, metode *stemming* dan metode perhitungan *similarity* yang digunakan akan berpengaruh secara signifikan terhadap hasil perangkingan kalimat topik. Aplikasi yang dibuat akan menghasilkan ringkasan yang baik jika kalimat topik yang digunakan banyak mengandung kata penting dari paragraf atau dokumen.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan penggunaan metode *stemming* yang lebih akurat dengan menggunakan kamus kata dasar. Dan juga metode penghitungan *similarity* yang digunakan harus lebih baik. Disamping itu disarankan untuk menggunakan data uji yang benar-benar valid dan bervariatif untuk menghasilkan data hasil uji coba yang lebih akurat.


UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Bondy, J. A., & Murty, U. S. (1976). *Graph Theory With Applications*. New York: Elsevier Science Publishing Co.

Gunes, E., & Dragomir, R. R. (2004). LexRank : Graph-Based Centrality as Salience in Text *Summarization*. *Journal of Artificial Intelegence Research 22* , 1-23.

Hovy, E. (2003). Text *Summarization*. Dalam R. Mitkov, *The Oxford Handbook of Computational Linguistics* (hal. 583-589). Oxford: Oxford University Press.

Maimunah, D. S. (2007). *Buku Pintar Bahasa Indonesia.* Jakarta: Prestasi Pustaka.

Manning, C. D., Raghavan, P., & Schütze, H. (2007). *An Introduction to Information Retrieval.* Cambridge, England: Cambridge University Press.

Munir, R. (2005). *Matematika Diskrit Edisi Ketiga.* Bandung: Informatika.

Wahyuni, I. (2007). *Graph Based Summarization.* Malang: Program Studi Ilmu Komputer Fakultas MIPA Universitas Brawijaya.


UNIVERSITAS BRAWIJAYA

LAMPIRAN

Daftar Stop Word

yang	Sebagai	sekitar	pertama
di	Bahwa	secara	kedua
dan	Dapat	dilakukan	memang
itu	Para	sementara	pernah
dengan	Harus	tapi	apa
untuk	Namun	sangat	mulai
tidak	Kita	Hal	sama
ini	Dua	sehingga	tentang
dari	Satu	seorang	bukan
dalam	Masih	bagi	agar
akan	Hari	besar	semua
pada	Hanya	lagi	sedang
juga	mengatakan	selama	kali
saya	Kepada	antara	kemudian
ke	Kami	waktu	hasil
karena	Setelah	sebuah	sejumlah
tersebut	melakukan	jika	juta
bisa	Lalu	sampai	persen
ada	Belum	jadi	sendiri
mereka	Lain	terhadap	katanya
lebih	Dia	Tiga	demikian
kata	kalau	serta	masalah
tahun	terjadi	pun	mungkin
sudah	banyak	salah	umum
atau	menurut	merupakan	setiap
saat	jalan	atas	bulan
oleh	anda	sejak	bagian
menjadi	hingga	membuat	bila
orang	tak	baik	lainnya
ia	baru	memiliki	terus
telah	beberapa	kembali	luar

adalah	Ketika	selain	cukup
seperti	Saja	tetapi	termasuk
maka	sebelumnya	maka	maupun
masuk	Wib	masuk	mantan
mengalami	Tempat	mengalami	lama
sering	Perlu	sering	jenis
ujar	menggunakan	ujar	segera
kondisi	memberikan	kondisi	misalnya
akibat	Rabu	akibat	mendapat
hubungan	Sedangkan	hubungan	bawah
empat	Kamis	empat	jangan
paling	Langsung	paling	meski
mendapatkan	Apakah	mendapatkan	terlihat
selalu	Pihak	selalu	akhirnya
lima	Melalui	lima	jumat
meminta	Diri	meminta	punya
melihat	Mencapai	melihat	yakni
sekarang	Minggu	sekarang	terakhir
mengaku	Aku	mengaku	kecil
mau	Berada	mau	panjang
kerja	Tinggi	kerja	badan
acara	Ingin	acara	juni
menyatakan	Sebelum	menyatakan	of
masa	Tengah	masa	jelas
proses	Kini	proses	jauh
tanpa	The	tanpa	tentu
selatan	Tahu	selatan	semakin
sempat	Bersama	sempat	tinggal
adanya	depan	adanya	kurang
hidup	selasa	hidup	mampu
datang	Begitu	datang	posisi
senin	Merasa	senin	asal

rasa
sebesar
berat
dirinya
memberi
pagi
sabtu
ternyata
mencari
sumber
ruang
menunjukkan
biasanya
nama
sebanyak
utara
berlangsung
barat
kemungkinan
yaitu
berdasarkan
sebenarnya
cara
utama
pekan
terlalu
membawa
kebutuhan
suatu
menerima
penting
tanggal

berbagai
mengenai
tingkat
awal
sedikit
nanti
pasti
muncul
dekat
lanjut
ketiga
biasa
dulu
kesempatan
ribu
akhir
membantu
terkait
sebab
menyebabkan
khusus
bentuk
ditemukan
diduga
mana
ya
kegiatan
sebagian
tampil
hampir
bertemu
usai

rasa
digunakan
justru
padahal
menyebutkan
gedung
apalagi
program
milik
teman
menjalani
keputusan
upaya
mengetahui
mempunyai
berjalan
menjelaskan
mengambil
benar
lewat
belakang
ikut
barang
meningkatkan
kejadian
kehidupan
keterangan
penggunaan
masing-masing
menghadapi
pula
terutama

sekali
Sesuai
Bagaimana
Berarti
Keluar

LAMPIRAN

Kode dokumen : 36
Sikap pemerintah yang tetap mempertahankan ujian nasional amat disesalkan. Tidak sekadar mengabaikan aspirasi masyarakat, kenekatan ini juga menimbulkan kesan mereka tak siap menerima kekalahan. Nyatanya, Menteri Pendidikan Nasional Muhammad Nuh pun berencana mengajukan peninjauan kembali terhadap putusan kasasi yang memenangkan penggugat ujian nasional. Pak Menteri bahkan menggambarkan perkara itu mirip permainan sepak bola: masih mungkin menang pada injury time. Itu sebabnya, ia akan melawan putusan Mahkamah Agung, dan terkesan belum mau melaksanakan putusan yang menolak kasasi pemerintah. Harus diakui, Menteri Pendidikan, sebagai salah satu tergugat selain Presiden dan Wakil Presiden, secara hukum berhak mengajukan peninjauan kembali. Tapi langkah ini tak bisa digunakan sebagai alasan untuk menunda pelaksanaan putusan. Sebagai konsekuensi ditolaknya kasasi, pemerintah wajib menjalankan vonis Pengadilan Negeri Jakarta Pusat pada Mei 2007 yang dikuatkan oleh pengadilan tinggi tujuh bulan kemudian. Majelis hakim saat itu mengabulkan sebagian gugatan kelompok masyarakat yang dirugikan oleh ujian nasional. Keinginan utama penggugat, yakni penghapusan ujian nasional, memang tak ditolak, tapi pemerintah diwajibkan memperbaiki pelaksanaannya. Hakim juga menyatakan pemerintah terbukti lalai dalam memenuhi hak asasi manusia dalam bidang pendidikan. Selain diharuskan memperbaiki kualitas guru dan sarana pendidikan, pemerintah diwajibkan menolong korban ujian nasional yang mengalami gangguan psikologis. Pemerintah jelas belum melaksanakan perintah pengadilan. Padahal mengevaluasi dan memperbaiki ujian nasional amat penting agar kebijakan yang didasari peraturan pemerintah ini tak menimbulkan ketidakadilan. Begitu pula memperbaiki mutu pendidikan di seluruh pelosok Tanah Air.

Sebab, tanpa pemerataan kualitas pendidikan, ujian nasional hanya akan menciptakan tragedi bagi banyak siswa. Bahkan ada di antara mereka yang mengalami gangguan psikologis dan hingga sekarang belum disantuni oleh pemerintah.

Khalayak juga bertanya-tanya, bukti baru apa lagi yang disodorkan pemerintah untuk mengajukan peninjauan kembali. Jika Menteri Pendidikan ingin menggambarkan bahwa mutu pendidikan di semua sekolah di negeri ini sudah merata sehingga ujian nasional layak dipakai sebagai penentu kelulusan, jelas akan sulit mencari buktinya.

Mestinya pemerintah lebih cerdas menyikapi putusan pengadilan yang sebenarnya setuju ujian nasional digunakan sebagai tolok ukur pendidikan nasional. Itu sebabnya, ujian ini tak perlu dihapus, tapi berbagai akibat buruknya harus diatasi. Putusan ini sesungguhnya gampang dilaksanakan dengan menjadikan ujian nasional bukan lagi sebagai penentu utama kelulusan. Biarlah sekolah yang menentukan standar kelulusan. Toh, peringkat tiap sekolah, juga kualitas pendidikan di berbagi daerah, tetap bisa dilihat lewat hasil ujian nasional.

Ketimbang sibuk melawan keinginan masyarakat lewat peninjauan kembali, revisi fungsi ujian nasional itulah yang seharusnya digodok pemerintah.

LAMPIRAN

Kode Dokumen : 37

Meski terlambat, masih lebih baik daripada tidak sama sekali. Setelah tertunda selama enam tahun, akhirnya rancangan undang-undang pengendalian tembakau disiapkan pemerintah. Inilah aturan yang tidak hanya penting untuk meredam agresifnya industri rokok dalam memasarkan produknya, tapi juga terutama untuk mencegah makin banyaknya perokok pemula berusia muda.

Pembuatan undang-undang itu sebetulnya merupakan amanat sidang Badan Kesehatan Dunia pada Mei 2003. Dalam sidang ini Indonesia, bersama puluhan negara lain, menyepakati perlunya dibuat peraturan hukum yang bersifat mengikat untuk mengendalikan dampak tembakau, atau Framework Convention on Tobacco Control (FCTC).

Atas dasar kesepakatan itu, Badan Kesehatan Dunia mewajibkan setiap negara peserta konvensi segera membuat undang-undang pengawasan tembakau. Termasuk dalam cakupan undang-undang ini, antara lain, pengawasan produksi industri rokok, pengaturan jenis iklan, hingga pemberian sanksi keras bagi penyelundup rokok. Pendeknya, lewat undang-undang ini, produksi, promosi, dan konsumsi produk tembakau diatur ketat. Tujuannya, mencegah makin banyaknya perokok.

Anehnya, meski termasuk salah satu negara peserta konvensi, Indonesia tak kunjung meratifikasi ketentuan ini. Padahal, hingga 2008, sudah 160 negara meratifikasinya. Bahkan sampai sekarang pun Indonesia satu-satunya negara di Asia-Pasifik yang belum meratifikasi FCCT.

Itu sebabnya, jika undang-undang pengendalian tembakau segera dibuat, kita sudah melangkah maju. Adanya pasal dalam Undang-Undang No. 36 Tahun 2009 tentang Kesehatan yang mengatur zat adiktif--rokok salah satunya--memang bagus, tapi masih kurang kuat. Melalui ratifikasi dan pembuatan undang-undang pengendalian tembakau, kita tidak hanya memiliki sanksi yang lebih jelas bagi pelanggarnya, tapi juga terikat pada hukum internasional.

Adanya undang-undang yang kuat tentang pengendalian tembakau makin mendesak karena konsumsi rokok di negara kita sudah pada tahap "mengerikan". Data WHO menyebutkan, pada 2008 Indonesia adalah negara pengisap rokok terbesar ketiga dunia setelah Cina dan India. Usia perokok pun dari tahun ke tahun kian muda.

Menteri Kesehatan Endang Rahayu Sedyaningsih, selaku pihak yang harus mengawal rancangan undang-undang ini, pun tak perlu ragu berhadapan dengan kalangan pro-rokok. Argumen bahwa pendapatan negara akan terganggu akibat berkurangnya cukai dari industri rokok pun hanya mitos.

Hasil survei Lembaga Demografi UI menyebutkan, biaya kesehatan yang harus dikeluarkan pemerintah untuk menangani penyakit tembakau mencapai US$ 18,1 miliar. Angka ini setara dengan 5,1 kali lipat total hasil cukai yang diperoleh dari produsen rokok. Jelaslah, lebih menguntungkan membatasi rokok. Bukan hanya karena dengan pembatasan maka pemerintah bisa menekan biaya kesehatan, tapi yang lebih penting lagi adalah menyelamatkan generasi muda yang sangat rawan tergoda iklan rokok.

LAMPIRAN

Kode Dokumen : 22
Khalayak, yang telah lama menanti penyelesaian hingga tuntas kemelut kasus Bibit Samad Rianto dan Chandra M. Hamzah, kembali menelan rasa kecewa. Presiden Susilo Bambang Yudhoyono kemarin memang telah menegaskan: solusi terbaik kasus mereka adalah tidak membawanya ke pengadilan. Tapi realisasinya belum ada, dan sejumlah rekomendasi lain dari Tim 8 pun tak dihiraukan. Dalam pidatonya, Presiden mengatakan telah terjadi pro-kontra seputar kasus dua pejabat nonaktif Komisi Pemberantasan Korupsi itu. Inilah salah satu alasan ia mendorong kepolisian dan kejaksaan menghentikan kasus mereka. Kebijakan ini sesuai dengan butir pertama dari enam butir rekomendasi Tim 8, yang bertugas memverifikasi proses hukum kasus Bibit-Chandra. Masalahnya, publik harus menunggu lagi karena baik kepolisian maupun kejaksaan belum merealisasi perintah itu. Kemarin seorang pejabat Kejaksaan Agung mengatakan baru berencana menghentikan penuntutan perkara Chandra Hamzah. Adapun kasus Bibit sampai kini masih berada di kepolisian, dan hingga kini juga belum ada penjelasan dari Kepala Polri. Masyarakat merasa penyelesaian kasus ini amat bertele-tele. Karena Presiden sebelumnya telah memanggil Kapolri dan Jaksa Agung, kenapa kepolisian dan kejaksaan tidak mengumumkan penghentian kasus ini sekaligus pada hari yang sama? Lambannya penanganan kasus ini hanya menimbulkan kesan bahwa pemerintah setengah hati melaksanakan rekomendasi Tim 8. Cara penyelesaian berbelit-belit itu sudah terbayangkan. Sehari sebelumnya, Presiden Yudhoyono mengungkapkan bahwa kemelut kasus Bibit-Chandra akan diselesaikan lewat out-of-court settlement (penyelesaian di luar pengadilan). Banyak orang bertanya-tanya, lantaran istilah itu biasa dipakai dalam perkara perdata. Artinya, pihak yang bersengketa berunding lewat mediator untuk berdamai.

Pertanyaannya, dalam kasus Bibit-Chandra, perdamaian terjadi antara siapa dan siapa?

Sempat pula beredar rumor, pemerintah berusaha meminta Bibit-Chandra tidak aktif lagi di KPK bila kasusnya dihentikan. Mudah-mudahan kabar yang dibantah oleh banyak pihak ini memang sekadar isapan jempol. Sebaiknya, jika pemerintah ingin memperbaiki kesalahannya, jangan ada syarat apa pun. Sebab, Tim 8 jelas menyatakan bahwa kasus Bibit-Chandra dipaksakan. Tim ini juga menyerukan agar pejabat yang bertanggung jawab atas penanganan kasus ini diberi sanksi. Rekomendasi inilah yang terkesan diabaikan oleh Presiden karena tak ada instruksi apa pun mengenai masalah ini.

Rekomendasi Tim 8 lainnya, misalnya soal pengusutan tuntas praktek mafia hukum yang diduga melibatkan Anggodo Widjojo, juga tidak disinggung. Presiden malah mengulang soal program pemberantasan makelar kasus yang akan dilakukan oleh pemerintah. Masalahnya, tentu sulit bagi masyarakat untuk mempercayai bahwa program ini dijalankan secara serius jika kasus yang berada di depan mata tidak diusut.

Khalayak yang mengharapkan munculnya gebrakan besar jelas kecewa. Hanya secercah harapan yang diberikan Presiden, dan itu pun kita masih harus menunggu realisasinya.

LAMPIRAN

Kode Dokumen : 26
Indonesia adalah negara yang terletak diwilayah tropis. Tidak heran jika penyakit – penyakit tropis sering menjangkit penduduknya. Khususnya malaria. Penyakit yang juga disebabkan oleh nyamuk ini, tak jarang ada di Indonesia. Malaria adalah penyakit yang disebabkan oleh parasit protozoa yang paling biasa di dunia, yiaitu plasmodium yang menyumbang kepada lebih kurang 3 juta kes dan 1.5 hingga 2.7 juta kematian setiap tahun. Ia disebarkan oleh nyamuk betina genus Anopheles (nyamuk tiruk), terutamanya Anopheles sundaicus di Asia dan An. gambiae di Afrika. Ramai orang mendapat malaria semasa mengembara ke negara-negara tropika atau subtropika. Malaria berlaku di kebanyakan bahagian sub Sahara Arika, Asia Tenggara dan Selatan, Mexico, Haiti, Amerika Tengah dan Selatan, Papua New Guinea dan Kepulauan Solomon. Penyakit tropis satu ini, sangat jarang diobati dengan cara alami. Padahal tak jarang di daerah tropis itu sendiri, menyimpan berbagai macam tanaman obat untuk malaria. Indonesia dikenal sebagai negara megadiversity terbesar nomor dua di dunia setelah Brasil. Nusantara memang memiliki begitu banyak flora dan fauna. Kekayaan hayati yang sudah dimanfaatkan nenek moyang kita sejak ratusan tahun lalu, sampai kini masih potensial dikembangkan. Salah satunya adalah tanaman johar (Cassia siamea Lamk), yang telah digunakan secara empirik tradisional untuk mengobati malaria. Pengobatan malaria menjadi penting, karena saat ini berbagai upaya untuk mengatasi malaria masih belum memuaskan. Penggunaan johar untuk atasi malaria sudah dilakukan masyarakat Jawa. Sedang di Aceh johar dikenal sebagai obat tradisional untuk penyakit kuning atau hepatitis.Kebiasaan menggunakan johar kemudian diteliti, untuk menjawab cara kerjanya dalam mengatasi malaria. Mungkinkah dapat membunuh parasit malaria, menurunkan demam, atau meningkatkan daya tahan tubuh?

Maka dilakukanlah penelitian pengaruh johar terhadap Plasmodium berghei in vivo pada mencit dan Plasmodium falciparum in vitro. Dilakukan pula penelitian untuk melihat efek antipiretik johar pada tikus yang didemamkan. Untuk mengetahui peningkatan daya tahan tubuh, dilakukan penelitian imunomodulator menggunakan tikus.

Selain itu, ada berbagai penelitian pelengkap antara lain toksisitas akut sampai subkronik, penelitian mutagenik untuk mengetahui efek perubahan gen yang dapat mengarah pada timbulnya kanker dan penelitian fitokimia untuk mengetahui kandungan zat berkhasiat, serta penelitian formulasi untuk memperoleh formula terbaik dilihat dari sisi teknologi farmasi.

Dari hasil penelitian di atas dapat disimpulkan bahwa ekstrak etanol 70 persen daun johar mempunyai efek antimalaria yang cukup aman. Namun, untuk menguji efek sebenarnya memang harus melalui uji klinik, padahal uji klinik hingga saat ini merupakan salah satu kendala pengembangan tanaman obat sampai diperolehnya fitofarmaka.

Ironisnya lagi, hasil penelitian yang sudah cukup jauh ini tidak membuat produsen obat tradisional tertarik meneruskannya ke skala industri. Jadi, harapan adanya pemikiran kegiatan penelitian yang saling link and match dengan industri masih berupa cita-cita belaka.

LAMPIRAN

Kode Dokumen : 21

Upaya Departemen Komunikasi dan Informatika menguasai wewenang penyadapan amatlah berlebihan dan patut ditentang. Tak hanya menabrak undang-undang, langkah ini juga akan melumpuhkan Komisi Pemberantasan Korupsi karena jadi tak leluasa menyadap demi penyelidikan.

Keinginan itu diungkapkan oleh Menteri Komunikasi Tifatul Sembiring, yang sedang menyiapkan peraturan pemerintah mengenai prosedur penyadapan. Peraturan ini merujuk pada Undang-Undang No. 11/2008 tentang Informasi dan Transaksi Elektronik. Tifatul menegaskan, penyadapan harus diatur. Konsekuensinya, kelak lembaga penegak hukum, seperti KPK, kepolisian, dan kejaksaan, harus menghubungi departemen yang dipimpinnya untuk memperoleh informasi penyadapan setelah mendapat perintah pengadilan.

Tifatul beralasan, hal serupa juga terjadi di mancanegara, seperti Australia dan Korea Selatan. Di sana penyadapan dilakukan oleh departemen komunikasi. Indonesia, dalam gagasannya, perlu mengikuti model ini dalam menyusun peraturan pemerintah tentang penyadapan yang ditargetkan rampung enam bulan lagi.

Persoalannya, Tifatul mungkin lupa bahwa tak satu pun undang-undang yang memberi Departemen Komunikasi wewenang menyadap atau menjadi koordinator penyadapan. Undang-Undang Informasi dan Transaksi Elektronik pun menyebutkan: wewenang menyadap hanya dimiliki oleh institusi penegak hukum. Maka, keinginan Pak Menteri untuk menjadikan Departemen Komunikasi sebagai pengelola penyadapan sungguh mengada-ada.

Rencana itu juga mengundang kontroversi lantaran bertentangan dengan UU No.30/2002 tentang Komisi Pemberantasan Korupsi. Pada pasal 12 undang-undang ini ditegaskan, KPK berwenang melakukan penyadapan. Berbeda dengan institusi penegak hukum lain, lembaga ini bahkan tak perlu meminta izin pengadilan untuk menyadap sekaligus merekam perbincangan lewat telepon. Jika gagasan Tifatul

dituangkan dalam peraturan, kewenangan KPK melakukan penyadapan bakal terusik.

Itulah yang memercikkan kecurigaan adanya agenda terselubung di balik penyusunan aturan penyadapan. Adakah hal ini merupakan salah satu langkah dari serangkaian upaya sistematis untuk melemahkan KPK? Kecurigaan ini wajar lantaran hingga sekarang pun perseteruan "cicak-buaya" belum reda. Masyarakat juga belum lupa ketika parlemen berniat membatasi kewenangan penyadapan oleh KPK saat menyusun Undang-Undang Pengadilan Tindak Pidana Korupsi beberapa waktu yang lalu.

Sudah bukan rahasia lagi bahwa serangkaian keberhasilan KPK menjebloskan para koruptor ke penjara salah satunya berkat aktivitas penyadapan. Dan, kini senjata ampuh itulah yang hendak dilumpuhkan. Maka, gagasan Menteri Tifatul bisa dianggap senapas dengan segala gerakan upaya pelemahan KPK.

Belum terlambat bagi Tifatul untuk merevisi gagasannya. Soal penyadapan memang perlu diatur agar tidak melanggar privasi orang, namun jangan sampai menabrak undang-undang dan mengebiri KPK.

LAMPIRAN

Kode Dokumen : 29

Berbagai kasus pencemaran lingkungan dan memburuknya kesehatan masyarakat yang banyak terjadi dewasa ini diakibatkan oleh limbah cair dari berbagai kegiatan industri, rumah sakit, pasar, restoran hingga rumah tangga. Hal ini disebabkan karena penanganan dan pengolahan limbah tersebut belum mendapatkan perhatian yang serius. Sebenarnya, keberadaan limbah cair dapat memberikan nilai negatif bagi suatu kegiatan industri. Namun, penanganan dan pengolahannya membutuhkan biaya yang cukup tinggi sehingga kurang mendapatkan perhatian dari kalangan pelaku industri, terutama kalangan industri kecil dan menengah.

Industri pengolahan makanan dari kedelai baik dalam skala kecil maupun menengah banyak terdapat di wilayah Kabupaten Banyumas terutama industri tahu dan tempe. Kedelai dan produk makanan yang dihasilkannya merupakan sumber makanan yang dapat diperoleh dengan mudah dan murah serta memiliki kandungan gizi yang tinggi. Industri tempe dan tahu menghasilkan limbah organik baik dalam bentuk cair maupun padat, namun kebanyakan industri tersebut membuang limbahnya secara langsung ke lingkungan tanpa pengolahan terlebih dahulu sehingga mencemari lingkungan.

Teknologi pengolahan limbah baik cair maupun padat merupakan kunci dalam memelihara kelestarian lingkungan. Apapun macam teknologi pengolahan limbah cair dan limbah padat baik domestik maupun industri yang dibangun harus dapat dioperasikan dan dipelihara masyarakat setempat. Jadi teknologi yang dipilih harus sesuai dengan kemampuan teknologi masyarakat yang bersangkutan.

Berbagai teknik pengolahan limbah cair untuk menyisihkan bahan polutannya yang telah dicoba dan dikembangkan selama ini belum memberikan hasil yang optimal. Untuk mengatasi masalah tersebut, maka diperlukan suatu metode penanganan limbah yang tepat, terarah dan berkelanjutan. Salah satu metode yang dapat diaplikasikan

adalah dengan cara BIO-PROSES, yaitu mengolah limbah organik baik cair maupun organik secara biologis menjadi biogas dan produk alternatif lainnya seperti sumber etanol dan methanol. Dengan metode ini, pengelolaan limbah tidak hanya bersifat "penanganan" namun juga memiliki nilai guna/manfaat. Selain itu, dengan metode bio-proses, teknologi yang digunakan sederhana, mudah dipraktekkan dengan peralatan yang relatif murah dan mudah didapat sehingga para industri kecil dan menengah tidak lagi beranggapan bahwa pengolahan limbah cair merupakan beban yang sangat mahal.

Untuk mengetahui efektifitas teknologi bioproses dalam membentuk energi alternatif (biogas) maka dilakukan penelitian tentang pembentukan biogas melalui teknologi bioproses dengan media limbah cair industri tahu dan tempe.

LAMPIRAN

Kode Dokumen : 35
Jalan panjang masih harus dilalui Prita Mulyasari buat menggapai keadilan. Setelah dikalahkan lagi di pengadilan banding oleh Rumah Sakit Omni Serpong, Tangerang, ia berupaya mengajukan permohonan kasasi. Kami mendukung langkah ini karena ia seharusnya tidak dinyatakan bersalah. Ibu dua anak itu divonis membayar ganti rugi Rp 204 juta kepada Omni hanya karena menulis keluhan mengenai layanan rumah sakit ini lewat surat elektronik. Inilah putusan Pengadilan Tinggi Banten yang diumumkan baru-baru ini. Putusan ini tak jauh berbeda dengan vonis sebelumnya di pengadilan negeri. Ganti ruginya saja yang lebih kecil. Tapi Prita tetap dinyatakan bersalah mencemarkan nama baik penggugat. Serangan dari Rumah Sakit Omni tak berhenti di situ. Sebab, Prita juga diadili secara pidana dalam kasus yang sama. Ia dijerat dengan pasal pencemaran nama baik, yang diatur dalam Kitab Undang-Undang Hukum Pidana (KUHP) serta Undang-Undang Informasi dan Transaksi Elektronik. Kasus ini sudah sampai pada tahap penuntutan, dan Prita dituntut hukuman 6 bulan penjara. Khalayak jelas prihatin. Mengapa penegak hukum mengabaikan rasa keadilan masyarakat? Kasus ini sempat mengundang protes besar-besaran ketika Prita ditahan. Tersangka kemudian dibebaskan lewat putusan sela, tapi belakangan ia tetap diseret ke pengadilan karena putusan itu dianulir oleh pengadilan tinggi. Bukan hanya Prita yang dipingpong, masyarakat pun dipermainkan. Nasib Prita mirip Minah di Banyumas, Jawa Tengah, yang dihukum hanya gara-gara mencuri tiga buah kakao. Penegak hukum tampak begitu sigap ketika melayani pengaduan dari pihak yang kuat, dan amat tega menimpakan putusan bersalah terhadap pihak yang lemah. Prita bahkan seharusnya tidak divonis bersalah dan tak mesti membayar ganti rugi karena gugatan Omni amat lemah. Sebab, tindakan tergugat tidaklah termasuk pencemaran nama baik karena ia

menceritakan pengalamannya sendiri. Prita tidak menyebarkan kebohongan, apalagi fitnah. Keluhan seperti ini justru merupakan hak pasien atau konsumen yang dilindungi undang-undang.

Delik pencemaran nama baik itu sendiri, baik dalam dalam KUHP maupun UU Transaksi Elektronik, juga sering dipersoalkan dan seharusnya tidak digunakan oleh penegak hukum. Aturan ini bertentangan dengan kebebasan berpendapat dan cenderung disalahgunakan demi membela kepentingan pihak yang berpengaruh secara politik atau ekonomi.

Pihak Omni memang menawarkan perdamaian dengan syarat tergugat meminta maaf secara terbuka. Sikap Prita yang tetap mengajukan kasasi sekaligus menolak tegas tawaran ini patut dipuji. Sebab, meminta maaf sama saja dengan mengakui kesalahan. Khalayak perlu mendukung sikap ini karena perjuangannya harus dilihat bukan sekadar untuk membebaskan diri dari jeratan hukum, melainkan juga mencegah orang lain kelak mendapat perlakuan yang sama.

Kegigihan Prita mestinya pula membuka hati hakim kasasi yang menangani gugatan ini, juga hakim Pengadilan Negeri Tangerang yang segera memutus kasus pidananya. Berilah ia keadilan.

LAMPIRAN

Kode Dokumen : 25

Akhir-akhir ini Indonesia menghadapi kejadian luar biasa, yaitu demam berdarah yang sangat meresahkan masyarakat. Kejadian tersebut berdampak pada kepanikan petugas kesehatan di rumah sakit serta sarana pelayanan kesehatan lain, karena terjadi lonjakan pasien yang dirawat di sarana-sarana pelayanan kesehatan

Penyakit Demam Berdarah atau Dengue Hemorrhagic Fever (DHF) ialah penyakit yang disebabkan oleh virus dengue yang ditularkan melalui gigitan nyamuk Aedes aegypti dan Aedes albopictus. Kedua jenis nyamuk ini terdapat hampir di seluruh pelosok Indonesia, kecuali di tempat-tempat ketinggian lebih dari 1000 meter di atas permukaan air laut.

Penyakit DBD sering salah didiagnosis dengan penyakit lain seperti flu atau tipus. Hal ini disebabkan karena infeksi virus dengue yang menyebabkan DBD bisa bersifat asimtomatik atau tidak jelas gejalanya. Data di bagian anak RSCM menunjukkan pasien DBD sering menunjukkan gejala batuk, pilek, muntah, mual, maupun diare. Masalah bisa bertambah karena virus tersebut dapat masuk bersamaan dengan infeksi penyakit lain seperti flu atau tipus. Oleh karena itu diperlukan kejelian pemahaman tentang perjalanan penyakit infeksi virus dengue, patofisiologi, dan ketajaman pengamatan klinis. Dengan pemeriksaan klinis yang baik dan lengkap, diagnosis DBD serta pemeriksaan penunjang (laboratorium) dapat membantu terutama bila gejala klinis kurang memadai.

Meningkatnya jumlah kasus serta bertambahnya wilayah yang terjangkit, disebabkan karena semakin baiknya sarana transportasi penduduk, adanya pemukiman baru, kurangnya perilaku masyarakat terhadap pembersihan sarang nyamuk, terdapatnya vektor nyamuk hampir di seluruh pelosok tanah air serta adanya empat sel tipe virus yang bersirkulasi sepanjang tahun.

Departemen kesehatan telah mengupayakan berbagai

strategi dalam mengatasi kasus ini. Pada awalnya strategi yang digunakan adalah memberantas nyamuk dewasa melalui pengasapan, kemudian strategi diperluas dengan menggunakan larvasida yang ditaburkan ke tempat penampungan air yang sulit dibersihkan. Akan tetapi kedua metode tersebut sampai sekarang belum memperlihatkan hasil yang memuaskan.

LAMPIRAN

Kode Dokumen : 19

Janji pemerintah untuk memberikan layanan ibadah haji yang lebih baik rupanya tidak terbukti. Seperti tahun-tahun sebelumnya, pengelolaan jemaah haji tahun ini tetap semrawut. Persoalan ini semakin kronis karena Departemen Agama lamban mereformasi penyelenggaraan ibadah haji.

Lihatlah perlakuan buruk yang dialami ribuan anggota jemaah haji. Begitu tiba di Arab Saudi, mereka menghadapi urusan yang melelahkan. Proses keimigrasian berlangsung berjam-jam. Karena ada renovasi, Bandara King Abdul Azis hanya menyediakan satu-dua counter pemeriksaan untuk jemaah asal Indonesia. Mengapa pemerintah tidak bisa mengusahakan agar jemaah haji kita dilayani beberapa counter?

Mereka pun dibawa ke pemondokan yang jauh. Sebagian besar penginapan di Mekah dan Madinah ternyata masih seperti dulu, jauh dari Masjidil Haram maupun Masjid Nabawi. Padahal sebelumnya digembar-gemborkan bahwa tahun ini lokasi pemondokan bisa ditempuh dengan berjalan kaki. Ternyata mayoritas jemaah haji masih menempati lokasi yang jaraknya 7 kilometer.

Banyak pemondokan juga kurang bersih. Ada sebuah penginapan yang kamar mandinya harus digunakan untuk 30 kamar. Bahkan ada jemaah haji yang mendapat kamar di basement, yang minim ventilasi udara dan fasilitas air. Padahal Departemen Agama sudah memutuskan memilih pemondokan yang cukup mahal: 3.000 riyal. Adapun tahun lalu hanya 2.000 riyal.

Urusan transportasi pun bermasalah. Di Mekah, bus-bus yang menjemput ke Masjidil Haram sering terlambat. Bahkan ada yang menurunkan jemaah di 5 kilometer dari Masjidil Haram, yang membuat banyak anggota jemaah kita memilih sembahyang di masjid dekat pemondokan daripada di Masjidil Haram.

Jika jemaah haji asal Malaysia bisa mendapat layanan yang jauh lebih baik, mengapa jemaah haji kita selalu

telantar? Inilah akibat diabaikannya kewajiban pemerintah: membenahi penyelenggaraan haji secara komprehensif, mulai urusan keimigrasian, pemondokan, transportasi, hingga akomodasi.

Pemerintah telah memiliki acuan, yakni Undang-Undang Nomor 13 Tahun 2008 tentang Penyelenggaraan Haji. Undang-undang ini diterbitkan untuk menjawab kritik tentang dominasi pemerintah, baik sebagai regulator maupun operator penyelenggaraan haji, serta lemahnya pengawasan. Tapi sebagian ketentuan dalam undang-undang ini belum bisa dilaksanakan lantaran pemerintah belum membuat aturan pelaksananya.

Hingga sekarang pemerintah juga belum menyiapkan Komisi Pengawas Haji, yang seharusnya didirikan pada April lalu, setahun setelah undang-undang itu berlaku. Belum adanya Komisi ini membuat pemerintah tak pernah mendapat masukan yang kritis demi memperbaiki penyelenggaraan haji.

Menteri Agama Suryadharma Ali hanya bisa memberikan janji lagi, hal yang sering pula disampaikan Maftuh Basyuni, Menteri Agama sebelumnya. Suryadharma, misalnya, berjanji bahwa pada tahun depan lokasi pemondokan haji kita paling jauh 4 kilometer dari Masjid Nabawi maupun Masjidil Haram.

Janji seperti itu percuma saja jika tak ada pembenahan secara menyeluruh dalam penyelenggaraan haji. Bisa saja soal pemondokan akan beres tahun depan. Tapi sanggupkah pemerintah menjamin layanan yang lain, seperti katering, fasilitas di penginapan, dan transportasi, akan lebih baik?

LAMPIRAN

Kode Dokumen : 32
Malaria pada manusia berkembang melalui dua fasa: fasa exoerythrokytik (hepatik) dan erythrokytik. Apabila nyamuk yang dijangkiti menusuk kulit manusia bagi menghisap darah, sporozoite dalam kelenjar liur nyamuk memasuki saluran darah dan bergerak ke hati. Dalam tempoh 30 minit ia memasuki hos manusia, ia menjangkiti hepatokyte, membiak secara aseksua dan asymptomatikal bagi tempoh6–15 hari. Apabila berada dalam hati organisma ini berubah bagi menghasilkan beribu merozoite yang, selepas memecahkan dinding sel hos mereka, lepas kedalam salur darah dan menjangkiti sel darah merah, dengan itu memulakan fasa erythrokytik dalam kitaran hayatnya. Parasit ini lepas dari hati tanpa dikesan dengan membalut dirinya dalam sel membren sel hati hoa yang dijangkiti. Dalam sel darah merah, parasit membiak lebih lanjut, sekali lagi secara aseksual, kadang-kala keluar daripada hosnya bagi menjajah sel darah merah baru. Kitaran peningkatan ini berlaku beberapa kali. Dengan itu, gambaran klasik gelombang demam yang timbul akibat gelombang serentak merozoites lepas dan menakluk sel darah merah yang baru. Setengah sporozoite P. vivax dan P. ovale tidak membentuk merozoites fasa exoerythrocytic serta merta, sebaliknya menghasilkan hypnozoite yang kekal pendam bagi tempoh beberapa bulan (6–12 bulan biasanya) sehingga selama tiga tahun. Selepas tempoh pendam, ia kembali aktif dan menghasilkan merozoite. Hypnozoite bertanggungjawab bagi tempoh pengeraman yang panjang dan pengulangan lama bagi dua spesies malaria ini. Parasit ini agak terlindung dari serangan sistem imunisasi badan manusia disebabkan bagi kebanyakan kitaran hayat manusia, ia berada dalam hati an sel darah merah dan agak terlindung tidak kelihatan kepada pemantauan sistem imunisasi. Bagaimanapun, sel darah merah yang dijangkiti dimusnahkan dalam limpa. Untuk mengelakkan nasib ini,

parasit P. falciparum menghasilkan protin pelekat pad permukaan sel darah yang dijangkiti, menyebabkan sel darah melekat pada dinding saluran darah kecil, dengan itu mengasingkan parasit dari melalui kitaran umum dan limpa. Daya lekat ini yang menimbulkan kerumitan hemorage pada malaria. venules endothelial tinggi (High endothelial venules) (cabang terkecil bagi sistem kitaran) boleh tersumbat akibat lekatan secara besar-besaran sel darah yang dijangkiti ini. Sekatan dalam salur ini menyebabkan simptom seperti pada malaria plasental dan serebral. pada malaria serebral sel darah merah yang terasing mampu membolosi halangan otak-darah, kemungkinannya menyebabkan koma.

Sungguhpun protin pelekat permukaan sel darah merah (dikenali sebagai PFEMP1, bagi Plasmodium falciparum erythrocyte membrane protein 1) terdedah kepada sistem imunisasi, ia bukanlah merupakan sasaran imunisasi yang baik disebabkan kepelbagaiannya yang tinggi; terdapat sekurang-kurangnya 60 variasi protin bagi setiap parasit dan kemungkinannya pelbagai versi tanpa had dalam keseluruhan populasi parasit. Seperti pencuri menukar penyamaran mereka atau perisik dengan pelbagai pasport, parasit menukar antara pelbagai himpunan protin permukaan PfEMP1 yang luas, dengan itu kekal di hadapan sistem imunisasi yang mengejarnya.

Setengah merozoites bertukar menjadi gametocyte jantan atau betina. Sekiranya nyamuk menggigit seseorang yang dijangkiti, ia berpotensi untuk turut menghisap gametocytes dalam darah. Kesuburan dan gabungan seksual parasit berlaku dalam perut nyamik, dengan itu mengtakrifkan nyamuk sebagai hos sah (definitive host) bagi malaria. Sporozoites baru terbentuk dan bergerak ke kelenjar liur nyamuk, melengkapi kitaran. Nyamuk anat tertarik kepada wanita mengandung, dan malaria bagi wanita mengandung adalan punca utama kelahiran mati (stillbirth), kematian anak kecil (infant mortality) dan kelahiran kurang berat badan (low birth weight).