SISTEM PERINGKAS DOKUMEN OTOMATIS PADA TEKS BERBAHASA INDONESIA DENGAN ALGORITMA *CLUSTERRANK*

JUDUL

SKRIPSI

Oleh:
FEBRYAN
0710963007-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
2011


UNIVERSITAS BRAWIJAYA

SISTEM PERINGKAS DOKUMEN OTOMATIS PADA TEKS BERBAHASA INDONESIA DENGAN ALGORITMA *CLUSTERRANK*

HALAMAN JUDUL

SKRIPSI

Oleh:
FEBRYAN
0710963007-96

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
2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

SISTEM PERINGKAS DOKUMEN OTOMATIS PADA TEKS BERBAHASA INDONESIA DENGAN ALGORITMA *CLUSTERRANK*

Oleh**:**
FEBRYAN
0710963007-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 22 Desember 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Achmad Ridok, M.Kom	**Reza Andria S, ST**
NIP. 196808251994031002	**NIP. 197906212006041003**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, MSc
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	Febryan
NIM	**:**	0710963007
Jurusan	**:**	Matematika
Penulisan skripsi berjudul	**:**	Sistem Peringkas Dokumen Otomatis Pada Teks Berbahasa Indonesia Dengan Algoritma *ClusterRank*

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, 22 Desember 2011
Yang menyatakan,

(Febryan)
NIM. 0710963007

SISTEM PERINGKAS DOKUMEN OTOMATIS PADA TEKS BERBAHASA INDONESIA DENGAN ALGORITMA *CLUSTERRANK*

ABSTRAK

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi pada komputer. Proses sistem peringkasan dokumen teks otomatis berbahasa indonesia menggunakan algoritma *ClusterRank* yaitu penggabungan antara metode *clustering* dengan *group average HAC,* dan metode perangkingan graf dengan *Lexrank.*

Pada penelitian ini digunakan sebanyak 10 dokumen teks berbahasa indonesia dengan sebagai dokumen uji. Pengujian dokumen dilakukan dengan memberikan dua nilai *threshold* yaitu *threshold* ukuran ringkasan dan *threshold* jumlah *cluster*. *Threshold* ukuran ringkasan diantaranya 25%, 50%, dan 75% dari ukuran asli dokumen. *Threshold* jumlah *cluster* diantaranya 4, 6, 8, 10, dan 12 *cluster*.

Akurasi yang dihasilkan sistem ditunjukkan dengan nilai rata – rata keseluruhan dari *precision* dan *recall* mencapai 0,412 dan 0,520. Nilai *precision* tertinggi dihasilkan pada *threshold* ringkasan = 25% dan *threshold cluster* = 6 yaitu dengan *precision* = 0,472. Untuk nilai *recall* tertinggi dihasilkan pada *threshold* ringkasan = 75% dan *threshold cluster* = 12 yaitu dengan *precision* = 0,554. Nilai *precision* dan *recall* bergantung pada kesesuain hasil ringkasan sistem dengan hasil ringkasan manusia. Pada penelitian ini, dokumen dengan kode dokumen 4 menghasilkan nilai *precision* paling tinggi yaitu dengan *precision* = 0,833 pada *threshold* ringkasan 25% dan *threshold cluster* 6 dan 8. Hal ini menunjukkan bahwa kalimat penting yang diekstrak sistem pada dokumen 4 hampir sesuai dengan hasil ringkasan manusia.

Kata Kunci : Text Mining, Peringkasan Dokumen, *ClusterRank, Group Average HAC, Lexrank.*

AUTOMATIC SUMMARIZATION SYSTEM FOR INDONESIAN TEXT DOCUMENT USING CLUSTERRANK ALGORITHM

ABSTRACT

Text summarization is a process of creating a shortening version of a given text automatically that provides useful information for the user. This system uses Clusterrank algorithm that combines two methods, group average HAC (Hierarchical Agglomerative Clustering) algorithm for clustering and Lexrank algorithm for graph ranking .

In this research, 10 files of Indonesian text were used as testing documents. Documents testing was done by providing two threshold values, those were summary threshold and cluster threshold. Summary thresholds consist of 25%, 50%, and 75% from the size of the documents. Cluster thresholds consist of 4, 6, 8, 10, and 12 clusters.

Accuracy of the system was shown by the results of mean precision and recall that reach 0,412 and 0,520. The highest precission is achieved at summary threshold = 25% and cluster threshold = 6 clusters, which had precission = 0,472. The highest recall is achieved at summary threshold = 75% and cluster threshold= 12 clusters, which had recall = 0,554. Precission and recall depend on the matching of system summary with human summary. In this research, the document with code 4 generated the highest precission, which had precission = 0,833 at summary threshold = 25% and cluster threshold = 6 and 8 clusters. It showed that the important sentences on document 4, which was extracted by the system, almost suited with human summary.

KeyWords : Text Mining, Text Summarization, ClusterRank, Group Average HAC, similarity, Lexrank.

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 Otomatis Pada Teks Berbahasa Indonesia Dengan Algoritma Clusterrank"*. 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. Drs. Achmad Ridok, M.Kom sebagai pembimbing I dan Reza Andria S.,ST sebagai 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, Arief, Zahra, Rifa, dan nenekku tercinta. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
5. Mila Febry, Dinar Rani, Ahmad Azwar Anas, Martin, Widy Hayuhardika, Yuita Arumsari, Sigit, Sindy Yudi, teman – teman kost kesumba 19 dan kumis kucing 42A atas bantuan, dukungan, semangat dan doanya.
6. Sahabat-sahabat ilkomers angkatan 2007 dan seluruh warga Program Studi Ilmu Komputer Universitas Brawijaya.
7. Pihak lain yang telah terlibat baik secara langsung maupun tidak langsung 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, 22 Desember 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemahaman dari isi yang terkandung dalam sebuah dokumen memerlukan beberapa metode untuk mengekstrak atau mengambil intisari dari dokumen tersebut. Salah satu metode adalah dengan melakukan peringkasan terhadap dokumen tersebut. Peringkasan adalah sebuah metode yang dilakukan untuk mendapatkan informasi penting dari sebuah dokumen. Tujuan utama dari peringkasan adalah untuk mendapatkan poin utama dari versi asli. Tidak mudah melakukan peringkasan dokumen jika jumlah atau ukuran dari dokumen tersebut banyak. Salah satu metode dalam peringkasan dokumen dengan jumlah besar, adalah sebuah metode peringkasan dokumen atau 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 pada komputer. Hasil peringkasan ini mengandung poin-poin penting dari teks asli (Gunes & Dragomir, 2004).

Penelitian tentang peringkasan teks ini telah diteliti oleh komunitas NLP (*Natural Language Processing*) hampir setengah abad terakhir. Hasil peringkasan teks dapat dihasilkan dari satu atau beberapa dokumen teks, yang menyampaikan informasi penting yang terkandung dalam teks aslinya, dan panjang dari ringkasannya tidak lebih panjang dari setengah teks asli. Secara sederhana, peringkasan teks dapat mencakup tiga aspek yaitu, pertama, ringkasan dapat dihasilkan dari dokumen tunggal atau beberapa dokumen, kedua, ringkasan harus menjaga informasi penting, dan ketiga, ringkasan harus pendek. Berbagai metode telah diterapkan dan masih terus dikembangkan oleh para peneliti di seluruh dunia. Berbagai penelitian yang telah dilakukan menghasilkan langkah-langkah peringkasan teks otomatis, yaitu *topic identification* atau pengumpulan topik, *interpretation* atau penggabungan topik dan yang terakhir pembentukan ringkasan. (Dipanjan & Martins, 2007)

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*.

Penelitian tentang peringkasan dokumen pernah dilakukan oleh Indah Wahyuni (2007). Pada penelitiannya, Indah Wahyuni melakukan penelitian *Automated Text Summarization* dengan menggunakan metode *graph based.* Pada penelitian ini digunakan metode perangkingan kalimat dengan memberikan nilai 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*. Untuk menentukan susunan hasil peringkasan, digunakan metode pencarian jarak terpendek (*shortest path*) dengan algoritma Djikstra. Penelitian yang lain juga pernah dilakukan oleh Widhy Hayuhardika (2010). Pada penelitiannya, dilakukan perangkingan kalimat dengan metode *normalized* TF-IDF dan *vector space models* untuk mendapatkan nilai *cosine 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.*

Pada penelitian skripsi ini digunakan algoritma *ClusterRank*, pemilihan algoritma ini berdasarkan pada hasil penelitian yang dilakukan oleh Garg N (2009) yang mendapatkan tingkat akurasi yang lebih tinggi dibandingkan dengan algoritma lainnya pada kasus

peringkasan dokumen rapat. Tahapan pada algoritma *Clusterrank* pertama adalah dilakukan proses *clustering* yaitu proses pengelompokkan *section – section* dokumen yang berupa kalimat yang kemudian akan dikomputasikan untuk mendapatkan nilai *similarity* antar *cluster*. Setiap *cluster* akan direpresentasikan sebagai sebuah *node* atau *vertex* pada graf. Kemudian *vertex* tersebut akan dirangking dengan perhitungan *LexRank*. Selanjutnya untuk menentukan kalimat mana yang dapat disebut sebagai ringkasan maka dilakukan pemberian nilai pada setiap kalimat berdasarkan *cluster*-nya.

1.2 Rumusan Masalah

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

1. Bagaimana implementasi Algoritma *Clusterrank* pada peringkasan dokumen teks bahasa indonesia
2. Bagaimanakah akurasi dari algoritma *ClusterRank* dalam meringkas dokumen diukur dengan *precision* dan *recall*.

1.3 Tujuan

Tujuan dari penulisan skripsi ini adalah:

1. Mengetahui hasil implementasi peringkasan dokumen teks berbahasa Indonesia menggunakan algoritma *ClusterRank.*
2. Mengetahui hasil akurasi algoritma *ClusterRank* dalam 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:

1. Jenis dokumen yang diuji adalah dokumen teks bahasa Indonesia tunggal (*single document*)
2. Dokumen yang diuji harus satu topik.
3. Sistem akan dikembangkan dengan bahasa pemrograman C# dan diterapkan pada komputer personal (PC)
4. Pengujian dilakukan terhadap beberapa dokumen dengan tipe paragraf yang berbeda.
5. Tidak dilakukannya perbandingan dengan metode lain

1.5 Manfaat

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

1.6 Metode Penulisan

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 dengan bahasa pemrograman C# yang mengimplementasikan proses peringkasan dokumen teks dengan menggunakan metode *ClusterRank*.

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

Dalam penulisan skripsi ini , akan menjabarkan keseluruhan penelitian yang kelompokkan secara sistematis menjadi lima bab. Pembagian bab – bab tersebut adalah ;

BAB I. PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat penelitian, metode penulisan dan penyelesaian masalah, dan sistematika penulisan.

BAB II. TINJAUAN PUSTAKA

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

BAB III. METODE PENELITIAN

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

BAB IV. HASIL DAN PEMBAHASAN

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

BAB V. PENUTUP

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

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

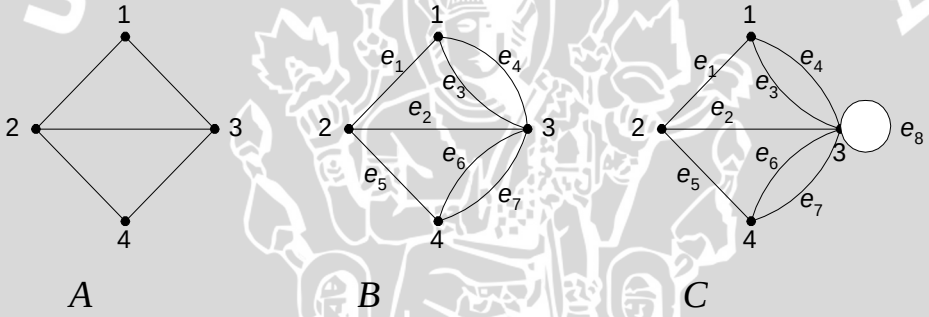
Graf dapat dikelompokkan menjadi beberapa jenis bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang dari ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul atau berdasarkan orientasi arah pada sisi (Munir, 2005).

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*).

   Contoh gambar graf sederhana dan tak sederhana seperti pada gambar 2.1.

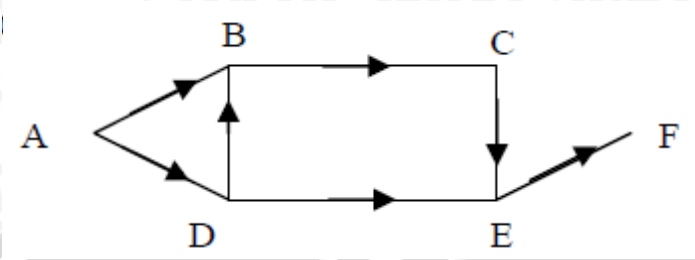


Gambar 2-1 (a) graf sederhana, (b) graf ganda, dan (c) graf semu

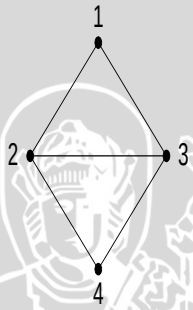
Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka graf dapat dibedakan menjadi dua jenis : (Munir, 2005)

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.

   Contoh gambar graf berarah dan tak berarah terlihat seperti pada gambar 2.2 dan 2.3.



Gambar 2-2 Graf berarah



Gambar 2-3 Graf tak berarah

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 Predikat. 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 frasa 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 analisis 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* yang mana *stoplist* berisi daftar kata – kata yang tidak penting yang nantinya akan dibandingkan dengan kata – kata yang ada pada dokumen. Pada penelitian skripsi ini tahap *tokenizing* dan *filtering* mengambil rancangan dari penelitian yang dilakukan oleh Hayuhardika W (2010). Tahap *stemming* adalah tahap mecari *root* (kata dasar) dari hasil *filtering*. Pada penelitian skripsi ini tahap *stemming* adalah *stemming porter* yang mengambil rancangan dari penelitian Agusta L yang dipublikasikan pada Konferensi Nasional Sistem dan Informatika (2009). 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 *Porter Stemming*

Proses *stemming* terhadap kata-kata dilakukan agar didapatkan kata *root* atau kata dasar. Rancangan dari proses ini diambil dari rancangan yang dilakukan oleh Agusta L (2009) pada penelitiannya. Proses *stemming* ini dilakukan dengan menggunkan algoritma *porter stemming.* Langkah – langkah algoritma *porter stemming* adalah :

1. Hapus *Particle*,
2. Hapus Possesive Pronoun.
3. Hapus awalan pertama. Jika tidak ada lanjutkan ke langkah 4a, jika ada cari maka lanjutkan ke langkah 4b.
4. a . Hapus awalan kedua, lanjutkan ke langkah 5a.

b. Hapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai *root word*. Jika ditemukan maka lanjutkan ke langkah 5b.

5. a. Hapus akhiran. Kemudian kata akhir diasumsikan sebagai *root word*

   b. Hapus awalan kedua. Kemudian kata akhir diasumsikan sebagai *root word*.

Terdapat 5 kelompok aturan pada Algoritma Porter untuk Bahasa Indonesia ini. Aturan tersebut dapat dilihat pada Tabel 2.1 sampai dengan tabel 2.5.

Tabel 2-1 Aturan untuk *Inflectional Particle*

Akhiran	Replament	Measure Condition	Additional Condition	Contoh
-kah	NULL	2	NULL	bukukah
-lah	NULL	2	NULL	pergilah
-pun	NULL	2	NULL	bukupun

Tabel 2-2 Aturan untuk *Inflectional Possesive Pronoun*

Akhiran	Replament	Measure Condition	Additional Condition	Contoh
-ku	NULL	2	NULL	bukuku
-mu	NULL	2	NULL	bukumu
-nya	NULL	2	NULL	bukunya

Tabel 2-3 Aturan untuk *First Order Derivational Prefix*

Akhiran	Replament	Measure Condition	Additional Condition	Contoh
meng-	NULL	2	NULL	mengukur -> ukur
meny-	S	2	V….*	menyapu -> sapu
men-	NULL	2	NULL	menduga -> duga
mem-	P	2	V….	memaksa -> paksa
mem-	NULL	2	NULL	membaca -> baca
me-	NULL	2	NULL	merusak -> rusak
peng-	NULL	2	NULL	pengukur -> ukur
peny-	S	2	V….	penyapu -> sapu

pen-	NULL	2	NULL	penduga -> duga
pem-	P	2	V….	pemaksa -> paksa
pem-	NULL	2	NULL	pembaca -> baca
di-	NULL	2	NULL	diukur -> ukur
ter-	NULL	2	NULL	tersapu -> sapu
ke-	NULL	2	NULL	kekasih -> kasih

Tabel 2-4 Aturan untuk Second *Order Derivational Prefix*

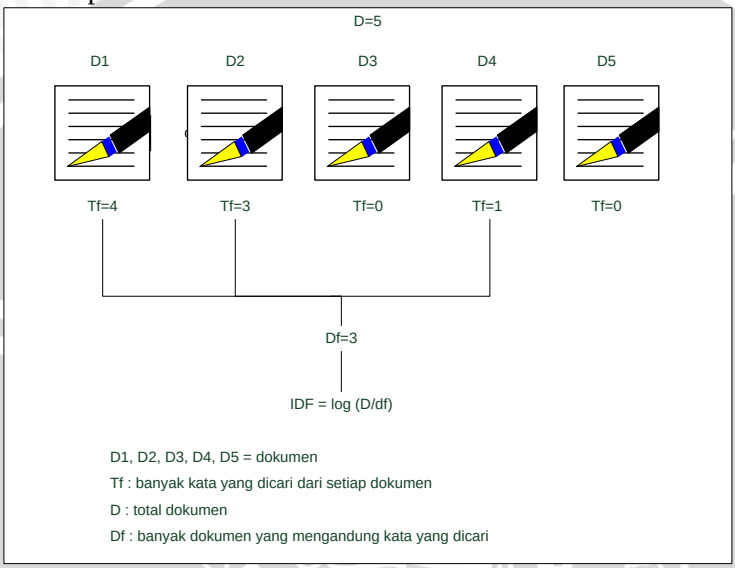
Akhiran	Replament	Measure Condition	Additional Condition	Contoh
ber-	NULL	2	NULL	berlari -> lari
bel-	NULL	2	*ajar*	belajar -> ajar
be-	NULL	2	*k*er*	bekerja -> kerja
per-	NULL	2	NULL	perjelas -> jelas
pel-	NULL	2	ajar	pelajar -> ajar
pe-	NULL	2	NULL	pekerja -> kerja

Tabel 2-5 Aturan untuk *Derivational Suffix*

Akhiran	Replament	Measure Condition	Additional Condition	Contoh
-kan	NULL	2	prefix bukan anggota {ke, peng}	tarikan -> tarik, Mengambilkan -> ambil
-an	NULL	2	prefix bukan anggota {di, meng, ter}	makanan -> makan, perjanjian -> janji
-i	NULL	2	*k*er*	tandai -> tanda, mendapati -> dapat

2.3.2 Pembobotan TF-IDF (*Word Frequency – Inverse Document Frequency*) dengan TF Ternormalisasi

Salah satu metode *Text Minning* adalah metode TF - IDF. Metode TF - IDF digambarkan pada gambar 2.4 berikut. Metode ini dilakukan untuk mencari tingkat kemiripan sebuah dokumen terhadap sebuah kata kunci.



Gambar 2-4 Skema Algoritma TF/IDF

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

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

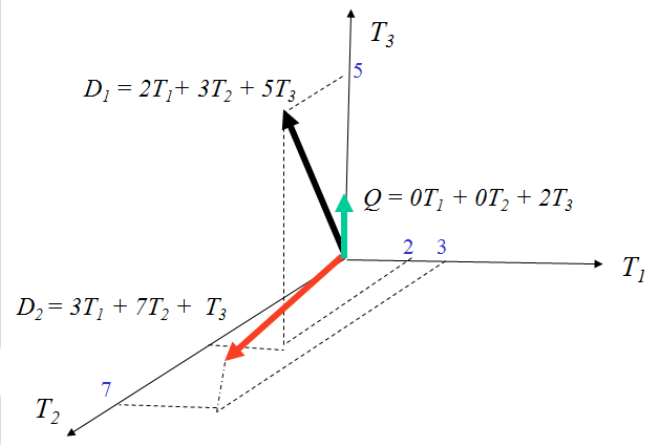
Dimana :
d = dokumen ke – d
t = kata ke-t dari kata / kalimat kunci
$w_{d,t}$ = bobot dokumen ke – d terhadap kata ke – t
df_t = jumlah dokumen yang mengandung kata kunci ke-t
$tf_{d,t}$ = frekuensi kata t pada dokumen ke-d

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

2.3.3 *Vector Space Model*

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 pengurutan 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 model*.

Dasar metode ini adalah menghitung nilai cosinus sudut dari dua vektor, yaitu bobot (w) dari setiap dokumen dan bobot (w) dari kata/kalimat kunci. Ilustrasi dari *Vector Space Model* (VSM) ditunjukkan pada gambar 2.5 (Garcia, 2006)



Gambar 2-5 Ilustrasi *Vector Space Model*

Dari gambar tersebut dapat dijelaskan bahwa T merupakan kata, D adalah dokumen, dan Q adalah *query* atau kata kunci. Dengan menggunakan *Vector space models* untuk menghitung nilai *cosine similarity*, formula yang dipakai adalah *cosine similarity* TF - IDF ternormalisasi, yang dinyatakan dalam persamaan 2.2.

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

Dimana *Q* adalah dokumen *query*, *D* adalah dokumen, *i* adalah indeks dokumen, $W_{Q,j}$ adalah bobot *term* ke-j pada dokumen *query*, $W_{i,j}$ adalah bobot *term* ke-j pada dokumen ke- i ,dan *j* adalah indeks *term*. *Weight (w)* untuk TF-IDF yang telah ternormalisasi dinyatakan dalam persamaan 2.3.

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

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

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

2.3.4 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).

2.3.5 *Precision* dan *Recall*

Performa dari identifikasi topik sebuah dokumen biasanya diukur menggunakan nilai *precision* dan *recall* (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* dan *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 *ClusterRank* Untuk Ekstraksi Dokumen

ClusterRank adalah metode ekstraksi dokumen untuk meringkas dengan berbasis graf. Metode ini menggabungkan antara metode pengelompokkan atau *clustering* dengan metode pengurutan atau *ranking.* Metode ini pernah diterapkan pada penelitian Nikhil Garg dan Benoit Favre di Universitas Berkeley Amerika untuk meringkas dokumen *meeting*. Pada penelitian skripsi ini, metode ini digunakan untuk meringkas dokumen bahasa indonesia tunggal. Algoritma *clusterrank* terdiri dari empat langkah dalam penerapannya yaitu *clustering, graph construction, LexRank, dan Sentence Scoring.*

2.4.1 *Clustering*

Proses *clustering* adalah proses pengelompokkan atau pengklasifikasian dokumen dengan memperhitungkan kedekatan setiap dokumen. Pada kasus pengklasifikasian dokumen, kedekatan setiap dokumen dihitung berdasarkan *similarity*-nya. Metode *clustering* yang digunakan pada penellitian skripsi ini adalah *HAC* (*hierarchical agglomerative clustering*). Penghitungan jarak menggunakan *group-average*. Menurut Selim Mimaroglu, penghitungan jarak antar dua *cluster* menggunakan persamaan 2.6 :

$$\frac{1}{|Ci||Cj|}\sum_{a\epsilon Ci}\sum_{b\epsilon Cj} d(a,b) \quad \textbf{(2.6)}$$

Dimana *d* adalah fungsi jarak. Pada permasalahan ini, fungsi jarak yang digunakan adalah *cosine similarity*. *Ci* adalah Cluster ke-i, *Cj* adalah Cluster ke-j, *a* adalah Anggota *Ci*, dan *b* adalah anggota *Cj*. (Mimaroglu, 2006). Hasil dari pembentukkan *cluster* dengan

metode HAC akan selalu membentuk *root cluster* sejumlah 1 pada *dendogram*. Sehingga dibutuhkan metode untuk memilih beberapa *cluster* untuk diproses selanjutnya pada tahap *graph construction*. Metode pemilihan *cluster* yang dipakai pada penelitian skripsi ini adalah *simple* HAC, dimana disesuaikan seperti proses *flat clustering*, yaitu *user* dapat memberikan nilai *k* sebagai jumlah *cluster* yang diinginkan. Pemilihan *cluster* dengan metode ini memanfaatkan hasil pembentukkan terakhir dari *cluster*, sebagai contoh jika *user* memilih k = 4, maka pembentukkan *cluster* akan berhenti pada hasil pembentukkan 4 *cluster* terakhir(Christopher D. Manning, 2009).

2.4.2 *Graph Construction*

Setiap *cluster* yang telah terbentuk pada tahap *clustering* yang berisikan kalimat – kalimat, kemudian setiap *cluster* itu dianggap sebagai satuan *vertex*, dengan kata lain satu *cluter* mewakili satu *vertex*. Sedangkan *edge vertex* merepresentasikan *similarity* antara dua *cluster* yang saling berelasi. *Similarity* didefinisikan sebagai *content overlap* antara dua *cluster*, yaitu jumlah kata yang sama antara keduanya. Selain *content overlap*, pengukuran *similarity* lain. Pada penelitian skripsi ini pengukuran *similarity* menggunakan *cosine similarity*, karena merupakan fungsi yang paling baik yang memiliki keuntungan untuk efisiensi komputasi apabila vektor dokumen ditransformasikan ke dalam bentuk vektor satuan, dibandingkan dengan menggunakan metode yang lain seperti *dice, jaccard, euclidean distance* , dan *pearson correlation* (Strehl, 2000). Untuk menghitung *edge* dari *cluster X* ke *cluster Y* menggunakan persamaan 2.7.

$$edge(X,Y) = \frac{sim\,(X,Y)}{\sum_z sim\,(X,Z)} \qquad \textbf{(2.7)}$$

Edges Cluster yang diperoleh direpresentasikan sebagai matriks *G* yang mana menotasikan *edge* dari setiap *node* atau *vertex*. Matriks G dihitung lagi untuk membentuk matriks M yang dibentuk dengan persamaan 2.8.

$$M = d * G + (1 - d) * \frac{1}{|nodes\,|} \qquad \textbf{(2.8)}$$

Dimana d = *damping factor* yang nilainya antara 0 dan 1, biasanya 0,85. *Nodes* yang berarti jumlah *cluster* yang terbentuk.

2.4.3 *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 2.9.

Saat diterapkan pada graf tekstual, biasanya terdapat bobot antara dua *vertex*, maka formula yang dipakai dinyatakan dalam persamaan 2.10.

$$S(V_i) = (1-d) + d * \sum_{V_j \in I_n(V_i)} \frac{1}{\left|\mathrm{Out}(\mathrm{V_j})\right|} \mathrm{S(V_j)} \quad \textbf{(2.9)}$$

$$WS(V_i) = (1-d) + d * \sum_{V_j \in I_n(V_i)} \frac{\mathrm{w_{ji}}}{\sum_{\mathrm{V_k} \in \mathrm{Out}\,(\mathrm{V_j})} \mathrm{w_{jk}}} WS(V_j) \quad \textbf{(2.10)}$$

Formula 2.10 mengasumsikan graf berarah. Namun algoritma ini dapat juga diterapkan pada graf tidak berarah dan berbobot (*undirected and weighted graph*), sehingga formulasinya seperti dalam persamaan 2.11.

$$WS(V_i) = d + (1-d) * \sum_{V_j \in Adj\,(V_i)} \frac{\mathrm{w_{ji}}}{\sum_{\mathrm{V_k} \in \mathrm{Adj}\,(\mathrm{V_j})} \mathrm{w_{jk}}} WS(V_j) \quad \textbf{(2.11)}$$

Dimana: $\mathrm{V_i}$ = *vertex* yang dihitung nilai-nya; $\mathrm{V_j}$ = *vertex* yang bertetanggaan dengan $\mathrm{V_i}$; $\mathrm{V_k}$ = *vertex* yang bertetanggaan dengan $\mathrm{V_j}$; dan d = *damping factor* yang nilainya antara 0 dan 1, biasanya 0,85.

2.4.4 *Sentence Scoring*

Setelah perhitungan nilai setiap cluster, untuk melanjutkannya pada perhitungan nilai kalimat digunakan

pendekatan dengan menghitung *centroid* dari kalimat dalam *cluster* dan mengurutkan kalimat – kalimat berdasarkan nilai *similarity* – nya dengan *centroid*–nya. Bagaimanapun, kalimat – kalimat dalam *cluster* memungkinkan mengandung kata – kata yang tidak berelasi dengan kalimat lainnya. Untuk menghindari pengaruhnya itu, maka *centroid* dibuat dengan hanya menghitung kata – kata yang sama dengan *cluster* lainnya. Karena score *LexRank* setiap *cluster* tergantung dengan *word overlap* dengan *cluster* lainnya, maka perhitungan ini lebih diandalkan untuk mengestimasikan seberapa pentingnya sebuah kalimat dalam *cluster*. Untuk sebuah *cluster X*, bobot dari sebuah kata *w* dalam *centroid Cx,* dihitung dengan persamaan 2.12.

$$Cx(w) = \sum_{Y:w\epsilon Y, Y\neq X} LR(Y) * M(X,Y) * IDF(w) \quad \textbf{(2.12)}$$

Dimana *LR* dinotasikan sebagai nilai *LexRank* sebuah *cluster*, *Cx* adalah *centroid* dari *cluster X*, *M* adalah nilai *edge* yang diambil dari matriks pembentukkan graf. Kemudian untuk menghitung nilai dari kalimat menggunakan persamaan 2.13.

$$score(s) = sim(s, Cx) * LR(X) \quad \textbf{(2.13)}$$

Dimana *sim(s, Cx)* adalah nilai *cosine similarity* yang dihitung dengan persamaan 2.14.

$$sim(s, Cx) = \frac{\sum_{\{w:ws \wedge w\epsilon Cx\}} Ws(w) * Cx(w)}{\sqrt{\sum_{ws} Ws^2(w)}\sqrt{\sum_{w\epsilon Cx} Cx^2(w)}} \quad \textbf{(2.14)}$$

Dimana *Ws(w)* diambil dari *IDF* dari *w*. Hasil perhitungan nilai dari setiap kalimat ini yang nantinya akan diurutkan mulai dari kalimat dengan nilai tertinggi hingga terendah, dan nilai tersebut yang akan menentukan ringkasan.

2.5 Peringkasan Dokumen Teks Bahasa Indonesia dengan Metode *LexRank*

Pada Program Studi Ilmu Komuter Fakultas MIPA Universitas Brawijaya pernah dilakukan penelitian terhadap pembuatan ringkasan dokumen otomatis dengan menggunakan algoritma *Lexrank*. Secara garis besar, langkah-langkah perankingan pada graf tekstual adalah sebagai berikut:

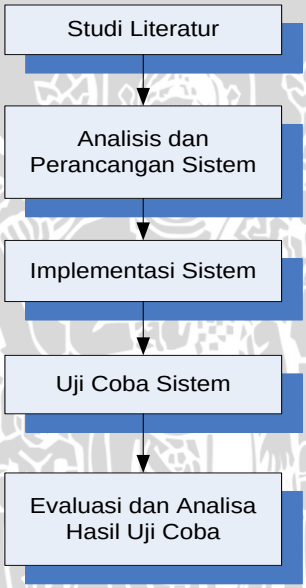
1. Mengidentifikasi unit teks terbaik untuk mendefinisikan tujuan perankingan, dan ditambahkan sebagai *vertex* dalam graf. Dalam hal ekstraksi kalimat, seluruh kalimat menjadi *vertex* dalam graf.
2. Mengidentifikasi relasi yang menghubungkan unit teks tersebut, dan relasi ini digunakan untuk membuat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

Dalam penelitian tugas akhir ini, 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.

BAB III

METODOLOGI

Sistem ini diracang untuk menghasilkan aplikasi yang mampu menerapkan peringkasan dokumen dengan menggunakan metode *ClusterRank graph based summarization algorithm*. Sistem akan dibuat pada mesin komputer personal (PC) dengan bahasa pemrograman C# . 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 *ClusterRank graph based summarization*

2. Menganalisa dan merancang perangkat lunak untuk menerapkan algoritma *ClusterRank 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 Analisis Data

Kumpulan data (*dataset*) yang digunakan dalam penelitian skripsi ini adalah dokumen tungal (*single document*) yang berbentuk *plaintext*, berbahasa Indonesia, dan bersifat satu topik. Data uji yang digunakan diambil dari sekumpulan dokumen dari penelitian sebelumnya tentang peringkasan dokumen yang pernah dilakukan oleh Widdhy Hayuhardika, mahasiswa Ilmu Komputer Universitas Brawijaya pada tahun 2010. Jumlah dari data uji ini berjumlah 10 dokumen dimana pada penelitiannya data uji sudah diringkas secara manual (oleh manusia) untuk dibandingkan dengan hasil ringkasan sistem agar mendapatkan nilai *precision* dan *recall* sistem. Daftar mengenai dokumen uji terdapat pada lampiran 2.

3.2 Perancangan Sistem

Dalam subbab ini, akan dijelaskan mengenai deskripsi sistem secara umum serta batasan sistem yang akan digunakan untuk membuat sistem peringkas dokumen teks otomatis bahasa indonesia dengan algoritma *clusterrank*.

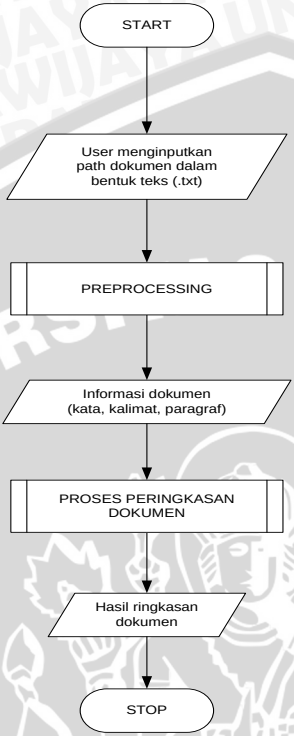
3.2.1 Deskripsi Umum Sistem

Secara umum, sistem akan memiliki fungsi untuk menerima *input* dokumen yang akan diringkas, jumlah *cluster*, dan ukuran ringkasan yang diinginkan dari *user*, kemudian dengan menggunakan metode *ClusterRank* sistem akan melakukan peringkasan dokumen sehingga dihasilkan ekstrak atau intisari dari dokumen. Dalam skripsi ini, sistem yang akan dibuat adalah sistem yang memproses dokumen berbahasa indonesia dalam bentuk *plain* teks. Kemudian dokumen yang diinputkan adalah *singledocument* bukan *multidocument*.

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, akan dibentuk *cluster* yang berisi kalimat – kalimat pada dokumen yang dibentuk dari perhitungan nilai kemiripan (*cosine similarity*) dari seluruh kalimat menggunakan *group average HAC*.

Untuk mengetahui nilai *cosine similarity* kalimat yang dihubungkan ini, digunakan metode *vector space model*. Proses *vector space model* melalui beberapa tahap *preprocessing*, yaitu proses pemecahan dokumen teks menjadi kalimat – kaliamat, proses *casefolding* (proses mengubah kata – kata menjadi huruf kecil semua), proses *filtering* (penghilangan kata yang tidak penting), proses *tokenizing*(pemecahan teks menjadi unit terkecil yaitu kata atau *term*), dan *stemming* (pemotongan kata atau *term* menjadi kata dasar). Proses *filtering* dalam penelitian ini menggunakan algoritma *stopword* dimana tiap kata (*term*) diperiksa apakah kata tersebut terdapat dalam daftar *stopword*, jika terdapat dalam *stopword*, maka kata tersebut tidak dimasukkan dalam daftar kata unik, selanjutnya menuju ke tahap *stemming*.

Proses *stemming* adalah 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.* Untuk lebih jelasnya proses secara umum digambarkan pada gambar 3.2.



Gambar 3-2 Flowchart Proses Sistem

3.2.2 Batasan Sistem

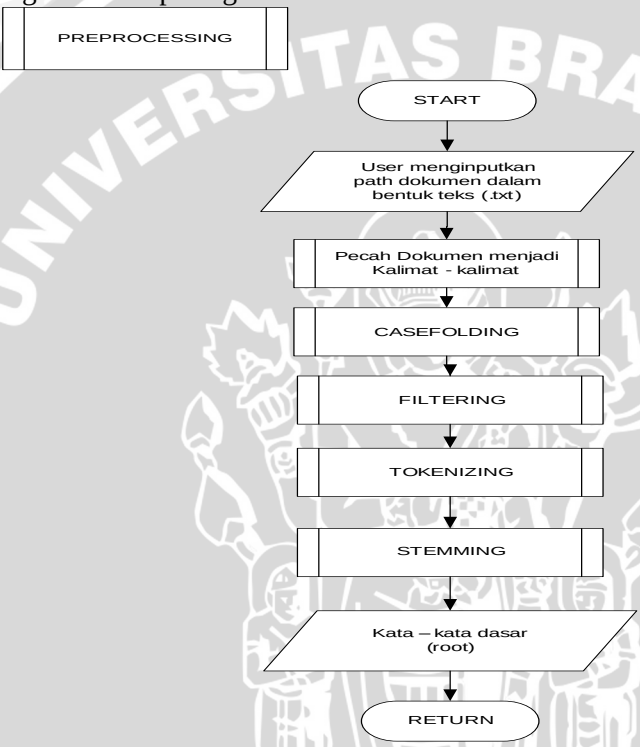
Batasan dari sistem yang dibuat pada penelitian ini diantaranya adalah :

1. Inputan dokumen pada sistem yang digunakan adalah dokumen teks berbahasa Indonesia *single document* yang berformat *plain* teks.
2. Inputan dokumen harus satu topik.
3. Sistem berupa aplikasi *desktop* menggunakan bahasa pemrograman C#.
4. Perhitungan *precision* dan *recall* akan dilakukan secara manual karena sistem *precision* dan *recall* membandingkan hasil ringkasan system dengan ringkasan manusia.

3.3 Perancangan Proses

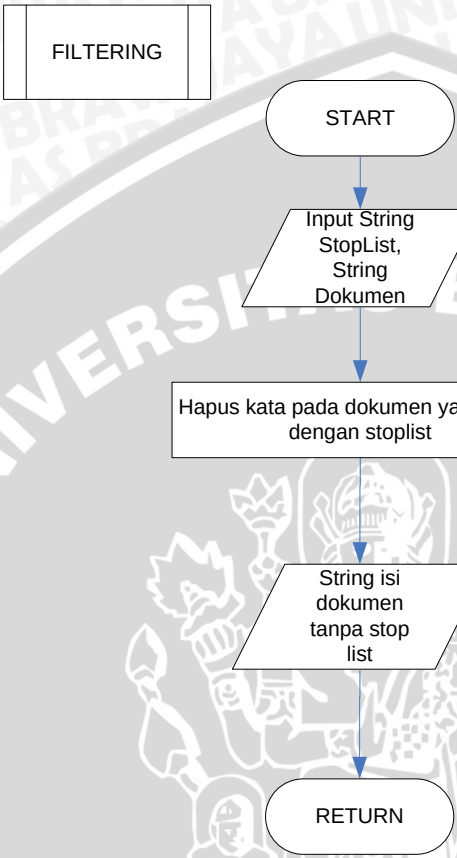
3.3.1 *Preprocessing*

Pada tahap *preprocessing* terdapat proses pemecahan dokumen teks menjadi kalimat – kalimat, proses *casefolding*, *filtering*, *tokenizing*, dan *stemming* terhadap dokumen yang diinputkan. Proses *preprocessing* dokumen menjadi kata-kata digambarkan pada gambar 3.3 berikut.



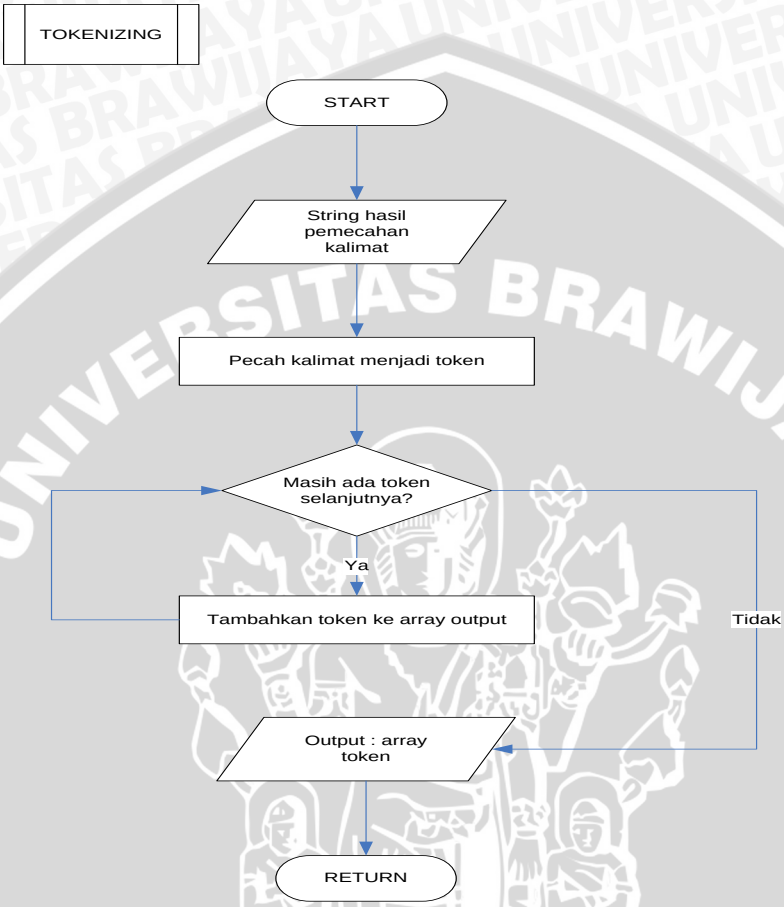
Gambar 3-3 Proses *Preprocessing*

Proses *filtering* dilakukan untuk mendapatkan daftar kata yang unik atau kata yang penting. Proses *filtering* adalah melakukan penghilangan kata yang diberikan terhadap daftar kata yang sering muncul (*stopword*). Tahap *filtering* digambarkan pada gambar 3.4.



Gambar 3-4 Proses *filtering*

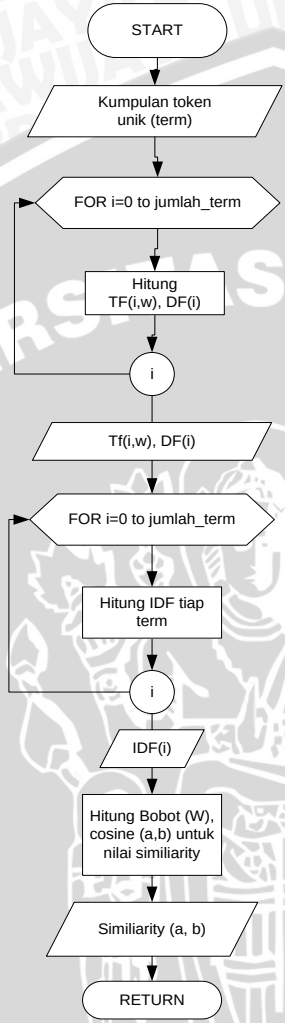
Proses *tokenizing* dilakukan untuk memecah kata dari dokumen dan mendapatkan daftar kata. Proses *tokenizing* digambarkan pada gambar 3.5. Setelah proses *tokenizing* selesai, selanjutnya adalah proses *stemming*. Algoritma *stemming* yang digunakan adalah *porter stemming* yang dirancang oleh Agusta L (2009) seperti yang tercantum pada bab 2.



Gambar 3-5 Proses *Tokenizing*

3.3.2 Proses TF-IDF dan *Vector Space Model*

Untuk melakukan proses Clusterrank, diperlukan data berupa matriks keterhubungan antarkalimat yang ditentukan dengan nilai *similarity* antarkalimat. Pada gambar 3.6 berikut digambarkan proses perhitungan nilai *similarity* antarkalimat menggunakan algoritma TF-IDF dan *vector space model*.

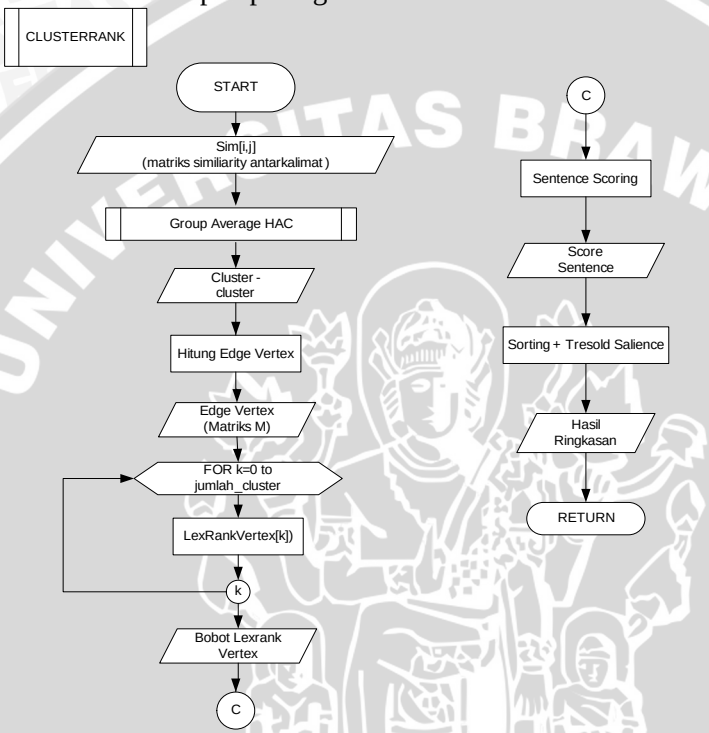


Gambar 3-6 Proses *Similarity* antar kalimat

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.

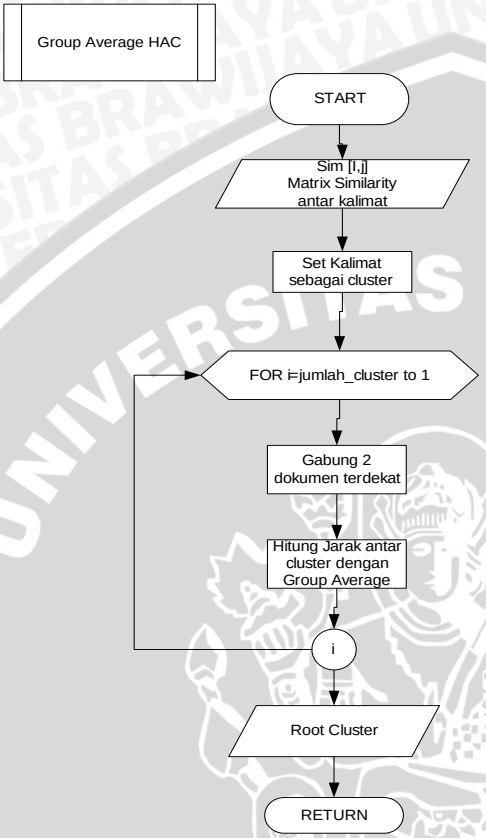
3.3.3 Proses *ClusterRank*

Dari proses penilaian *similarity* antarkalimat, didapatkan matriks *similarity* antarkalimat yang digunakan dalam perhitungan *ClusterRank* untuk melakukan proses peringkasan dokumen. Proses *ClusterRank* tampak pada gambar 3.7 berikut.



Gambar 3-7 Proses *ClusterRank*

Pada proses *ClusterRank* inilah terdapat proses *clustering* yang menggunakan algoritma *Group Average Hierarchical Agglomerative Clustering* seperti yang tampak pada gambar 3.8 berikut.



Gambar 3-8 Proses *group average HAC*

3.4 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 summarization* (%). Nilai ini akan menentukan ukuran ringkasan dokumen.
3. Nilai *treshold clustering*. Nilai ini akan menentukan pembentukkan *cluster* yang diinginkan. Nilai ini tidak boleh melebihi jumlah kalimat yang terinisialisasi.

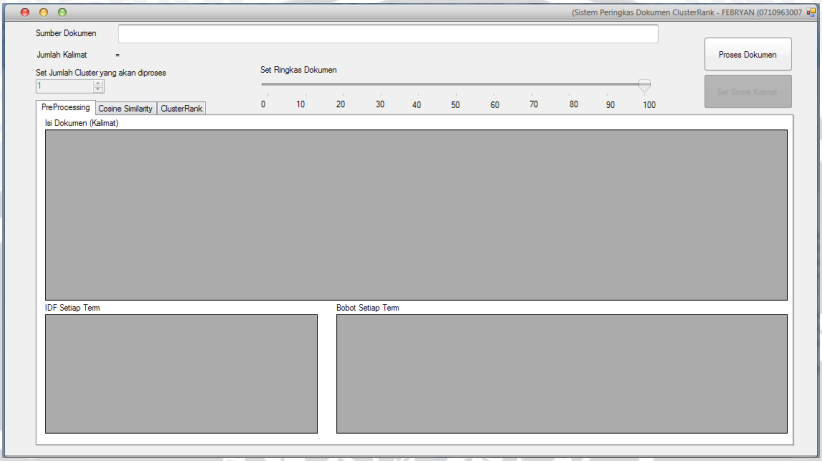
3.5 Perancangan User Interface

Sistem ini akan diterapkan pada komputer personal (PC) dengan bahasa Pemrograman C#. Secara garis besar, sistem akan berfungsi untuk membaca file sumber yang diberikan oleh *user*, kemudian mengekstrak informasi yang terkandung didalamnya (jumlah kata, kalimat). Setelah didapatkan data (kalimat, jumlah kata) dari dokumen yang diinputkan *user*, sistem akan melakukan pembobotan terhadap masing-masing kalimat, pembentukkan *cluster* dan peringkasan dokumen dengan ukuran ringkasan sesuai dengan nilai *treshold* ringkasan yang diberikan *user*.

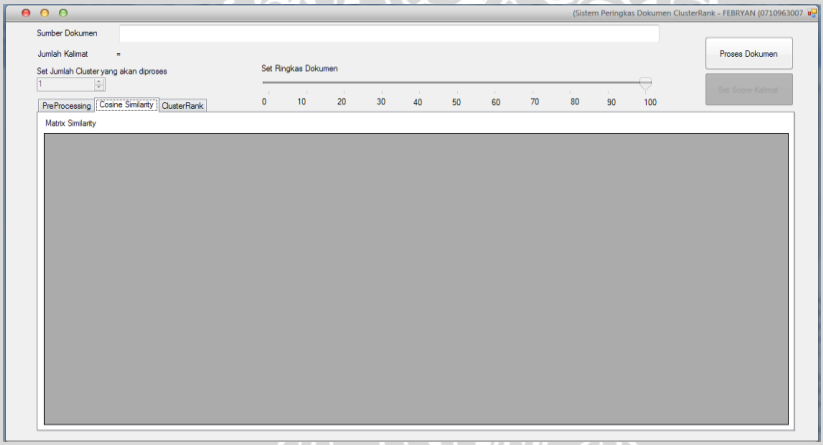
Gambar 3.9 – 3.11 adalah contoh rancangan *user interface* dari sistem. Bagian-bagian dari antarmuka sistem adalah :

1. *Field* **Sumber Dokumen** untuk menampilkan informasi *path* dari dokumen yang akan diproses.
2. Tombol **Proses Dokumen** yang digunakan untuk memproses dokumen mulai dari awal pemilihan dokumen yang dipilih lewat *open dialog*, kemudian langsung menampilkan informasi isi teks dokumen, perhitungan bobot dokumen.
3. Tombol **Set Score Kalimat** untuk menampilkan nilai *Lexrank* dari setiap *cluster* yang terbentuk dan menampilkan hasil nilai kalimat.
4. Pada Rancangan *User Interface* ini, digunakan *tab control* yang membagi halaman form menjadi 3, yaitu *form "*isi kalimat"yang menampilkan isi dari dokumen dan menampilkan hasil perhitungan bobot setiap kata dari dokumen, kemudian *form Matrix Similarity* yang menampilkan hasil perhitungan *cosine similarity* antar kalimat, dan *form ClusterRank* yang menampilkan hasil pembentukkan *cluster*, nilai *LexRank* setiap *cluster*, nilai setiap kalimat, dan hasil pembentukkan ringkasan.
5. *NumericUpDown* **Set Jumlah Cluster** untuk menentukan jumlah cluster yang akan dibentuk.

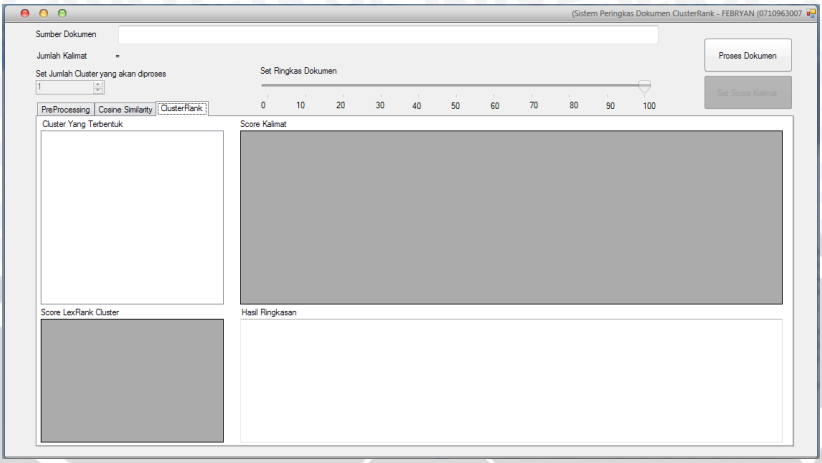
6. *TrackBar* **Set Ringkas Dokumen** untuk menentukan jumlah kalimat yang diinginkan sebagai hasil ringkasan dokumen.



Gambar 3-9 *Form* Utama



Gambar 3-10 *Form* Similarity Antar Kalimat



Gambar 3-11 *Form* Hasil Ringkasan

3.7 Perancangan Uji Coba

Dalam skripsi ini akan dilakukan uji coba terhadap beberapa dokumen yang berbeda. Dokumen yang diujikan sebelumnya telah dilakukan peringkasan atau pengekstrakan kalimat topik pada penelitian Hayuhardika, W (2010) Fakultas MIPA Universitas Brawijaya. Dari dokumen-dokumen yang akan diujikan tersebut, akan diukur tingkat *precision and recall* hasil ringkasan dokumen.

Pada percobaan perangkat lunak, aka dilakukan pengukuran nilai *precision* dan *recall* suatu ringkasan dokumen terhadap nilai *treshold cluster* yaitu jumlah pembentukan *cluster* yang diinginkan dan *treshold ringkasan* yaitu ukuran ringkasan yang diinginkan. Tabel 3.1 adalah contoh pengambilan data dari uji coba *precision and recall*.

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

Kode Dokumen	*Treshold Cluster* (int)	*Treshold* Ringkasan (%)					
		25		50		75	
		P	R	P	R	P	R
	4						
	6						
	8						

	10						
	12						

3.8 Contoh Perhitungan *Vector Space Model* dan *ClusterRank*

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 3.2 berikut.

Tabel 3-2 Informasi Dokumen

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

Kemudian dari dokumen tersebut dilakukan pemecahan menjadi kalimat-kalimat dan diberi identitas setiap kalimat (A, B, C, dan seterusnya) seperti yang tampak pada tabel 3.3 berikut.

Tabel 3-3 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 menggunakan algoritma *vector space model* dengan pembobotan TF-IDF ternormalisasi.

3.8.1 Tahap *Vector Space Model* dengan Pembobotan TF-IDF Ternormalisasi

Dari daftar kalimat yang diberikan, langkah pertama yang dilakukan adalah menghilangkan kata-kata yang dianggap sering muncul sebagaimana terlampir dalam lampiran 1. Proses ini dinamakan *filtering*. Keudian *tokenizing* yaitu mendaftar semua *term* (kata) dari masing-masing dokumen, mengubahnya ke dalam bentuk dasar (*Stemming*) kemudian Hasil proses *filtering*, *tokenizing, stemming,* dan perhitungannya tampak seperti pada tabel 3.4 berikut.

Tabel 3-4 Daftar Token

Token	**KALIMAT**						
	A	**B**	**C**	**D**	**E**	**F**	**G**
tiang	√	√	√	√	√	√	√
beton	√						
milik	√						

perusahaan	√						
telekomunikasi	√						
tinggi	√			√			
ambruk	√	√	√				√
jalan	√					√	
raya	√						
raci	√						
pakal	√						
surabaya	√						
satu		√					
dua		√	√				
bangun		√					
rumah		√					
rusak		√					
kena		√					
tarik			√				
kerja			√				
rubuh			√				
heri			√				
warga			√				
detiksurabaya.com			√				
minggu			√				
penuh				√			
badan				√			
jalan				√			
arus				√			
lalu				√			
lintas				√			
alih				√			
lewat				√			
kampung				√			

forklift					√		
tukang					√		
las					√		
kerah					√		
angkat					√		
potong					√	√	
letak						√	
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 menghitung TF(*Term Frequency*), dan DF(*Document Frequency*). Kemudian dihitung *weight* (W) dari masing-masing *term* menggunakan persamaan 2.3. Hasil perhitungan TF dan DF seperti yang ditampilkan pada tabel 3.5 dan hasil perhitungan bobot setiap term ditampilkan pada tabel 3.6 dan tabel 3.7.

Tabel 3-5 Hasil Perhitungan TF, DF, dan IDF

Token	**TF**							DF	IDF
	A	**B**	**C**	**D**	**E**	**F**	**G**		
tiang	1	1	2	1	1	1	1	7	0
beton	1							1	2,807354922
milik	1							1	2,807354922
perusahaan	1							1	2,807354922
telekomunikasi	1							1	2,807354922
tinggi	1			1				2	1,807354922
ambruk	1	1	1				1	4	0,807354922
jalan	1			2		1		3	1,222392421

raya	1							1	2,807354922
raci	1							1	2,807354922
pakal	1							1	2,807354922
surabaya	1							1	2,807354922
satu			1					1	2,807354922
dua		1	1					2	1,807354922
bangun		1						1	2,807354922
rumah		1						1	2,807354922
rusak		1						1	2,807354922
kena		1						1	2,807354922
tarik			1					1	2,807354922
kerja			1					1	2,807354922
rubuh			1					1	2,807354922
heri			1					1	2,807354922
warga			1					1	2,807354922
detiksurabaya			1					1	2,807354922
minggu			1					1	2,807354922
penuh				1				1	2,807354922
badan				1				1	2,807354922
arus				1				1	2,807354922
lalu				1				1	2,807354922
lintas				1				1	2,807354922
alih				1				1	2,807354922
lewat				1				1	2,807354922
kampung				1				1	2,807354922
forklift					1			1	2,807354922
tukang					1			1	2,807354922
las					1			1	2,807354922
kerah					1			1	2,807354922
angkat					1			1	2,807354922
potong					1	1		2	1,807354922

letak						1		1	2,807354922
pinggir						1		1	2,807354922
padam							1	1	2,807354922
listrik							1	1	2,807354922
putus							1	1	2,807354922
kabel							1	1	2,807354922

Tabel 3-6 Hasil Perhitungan Bobot Term Kalimat A – B – C

Token	W (BOBOT)		
	A	B	C
tiang	0	0	0
beton	2,807354922	0	0
milik	2,807354922	0	0
perusahaan	2,807354922	0	0
telekomunikasi	2,807354922	0	0
tinggi	1,807354922	0	0
ambruk	0,807354922	0,807354922	0,403677461
jalan	1,222392421	0	0
raya	2,807354922	0	0
raci	2,807354922	0	0
pakal	2,807354922	0	0
surabaya	2,807354922	0	0
satu	0	0	1,403677461
dua	0	1,807354922	0,903677461
bangun	0	2,807354922	0
rumah	0	2,807354922	0
rusak	0	2,807354922	0
kena	0	2,807354922	0
tarik	0	0	1,403677461
kerja	0	0	1,403677461
rubuh	0	0	1,403677461

heri	0	0	1,403677461
warga	0	0	1,403677461
detiksurabaya	0	0	1,403677461
minggu	0	0	1,403677461
penuh	0	0	0
badan	0	0	0
arus	0	0	0
lalu	0	0	0
lintas	0	0	0
alih	0	0	0
lewat	0	0	0
kampung	0	0	0
forklift	0	0	0
tukang	0	0	0
las	0	0	0
kerah	0	0	0
angkat	0	0	0
potong	0	0	0
letak	0	0	0
pinggir	0	0	0
padam	0	0	0
listrik	0	0	0
putus	0	0	0
kabel	0	0	0

Tabel 3-7 Hasil Perhitungan Bobot Kalimat D – E – F – G

Token	W (BOBOT)			
	D	**E**	**F**	**G**
tiang	0	0	0	0
beton	0	0	0	0
milik	0	0	0	0

perusahaan	0	0	0	0
telekomunikasi	0	0	0	0
tinggi	0,9036775	0	0	0
ambruk	0	0	0	0,80735492
jalan	1,2223924	0	1,2223924	0
raya	0	0	0	0
raci	0	0	0	0
pakal	0	0	0	0
surabaya	0	0	0	0
satu	0	0	0	0
dua	0	0	0	0
bangun	0	0	0	0
rumah	0	0	0	0
rusak	0	0	0	0
kena	0	0	0	0
tarik	0	0	0	0
kerja	0	0	0	0
rubuh	0	0	0	0
heri	0	0	0	0
warga	0	0	0	0
detiksurabaya	0	0	0	0
minggu	0	0	0	0
penuh	1,4036775	0	0	0
badan	1,4036775	0	0	0
arus	1,4036775	0	0	0
lalu	1,4036775	0	0	0
lintas	1,4036775	0	0	0
alih	1,4036775	0	0	0
lewat	1,4036775	0	0	0
kampung	1,4036775	0	0	0
forklift	0	2,8073549	0	0

tukang	0	2,8073549	0	0
las	0	2,8073549	0	0
kerah	0	2,8073549	0	0
angkat	0	2,8073549	0	0
potong	0	1,8073549	1,8073549	0
letak	0	0	2,8073549	0
pinggir	0	0	2,8073549	0
padam	0	0	0	2,80735492
listrik	0	0	0	2,80735492
putus	0	0	0	2,80735492
kabel	0	0	0	2,80735492

Kemudian setelah nilai bobot setiap term pada semua kalimat telah didapat, maka selanjutnya menghitung nilai *similarity* antar kalimat dengan menggunakan rumus *similarity* antarkalimat seperti persamaan 2.2. Proses ini dilakukan pada seluruh kalimat terhadap kalimat lainnya. Hasil perhitungan *similarity* ini ditampilkan pada tabel 3.8 – 3.19 berikut.

Tabel 3-8 *Similarity* A – B, A – C, dan A – D

Token	Similarity		
	A - B	**A - C**	**A - D**
tiang	0	0	0
beton	0	0	0
milik	0	0	0
perusahaan	0	0	0
telekomunikasi	0	0	0
tinggi	0	0	1,633266
ambruk	0,651822	0,325911	0
jalan	0	0	1,494243
Cosine	**0,013232**	**0,00963**	**0,08891**

Tabel 3-9 *Similarity* A – E, A – F, dan A – G

Token	Similarity		
	A - E	**A - F**	**A - G**
ambruk	0	0	0,651822
jalan	0	1,494243	0
Cosine	**0**	**0,03986**	**0,01389**

Tabel 3-10 *Similarity* B-A, B-C, B-D, dan B-E

Token	Similarity			
	B - A	**B-C**	**B-D**	**B-E**
ambruk	0,651822	0,325911	0	0
dua	0	1,633266	0	0
Cosine	**0,01323**	**0,08043**	**0**	**0**

Tabel 3-11 *Similarity* B-F, dan B-G

Token	Similarity	
	B-F	**B-G**
ambruk	0	0,651822
Cosine	**0**	**0,0193**

Tabel 3-12 *Similarity* C-A, C-B, dan C-D

Token	Similarity		
	C-A	**C-B**	**C-D**
ambruk	0,325911	0,325911	0
dua	0	1,633266	0
Cosine	**0,00963**	**0,08043**	**0**

Tabel 3-13 *Similarity* C-E, C-F, dan C-G

Token	Similarity		
	C-E	**C-F**	**C-G**
ambruk	0	0	0,325911
Cosine	**0**	**0**	**0,0193**

Tabel 3-14 *Similarity* D-A, D-B, dan D-C

Token	Similarity		
	D-A	**D-B**	**D-C**
tinggi	1,633266	0	0
jalan	1,494243	0	0
Cosine	**0,08891**	**0**	**0**

Tabel 3-15 *Similarity* D-E, D-F, dan D-G

Token	Similarity		
	D-E	**D-F**	**D-G**
jalan	0	1,494243	0
Cosine	**0**	**0,07759**	**0**

Tabel 3-16 *Similarity* E-D, E-F, dan E-G

Token	Similarity		
	E-D	**E-F**	**E-G**
potong	0	3,266532	0
Cosine	**0**	**0,11038**	**0**

Tabel 3-17 *Similarity* F-A, F-B, dan F-C

Token	Similarity		
	F.A	**F.B**	**F.C**
jalan	1,494243	0	0
Cosine	**0,03986**	**0**	**0**

Tabel 3-18 *Similarity* F-D, F-E, dan F-G

Token	Similarity		
	F.D	**F.E**	**F.G**
jalan	1,494243	0	0
potong	0	3,266532	0
Cosine	**0,07759**	**0,11038**	**0**

Tabel 3-19 *Similarity* G-A, G-B, dan G-C

Token	Similarity		
	G.A	**G.B**	**G.C**
ambruk	0,651822	0,651822	0,325911
Cosine	**0,01389**	**0,0193**	**0,01404**

Dari hasil perhitungan nilai *similarity* antarkalimat dapat dibentuk sebuah tabel yang berisi matrik *similarity* antarkalimat seperti pada tabel 3.20 berikut.

Tabel 3-20 Matriks *similarity* antarkalimat

	A	B	C	D	E	F	G
A	**1**	**0,013232299**	**0,009626489**	**0,088910422**	**0**	**0,039863155**	**0,013887727**
B	**0,013232299**	**1**	**0,080426996**	**0**	**0**	**0**	**0,019301473**
C	**0,009626489**	**0,080426996**	**1**	**0**	**0**	**0**	**0,014041809**
D	**0,088910422**	**0**	**0**	**1**	**0**	**0,077585167**	**0**
E	**0**	**0**	**0**	**0**	**1**	**0,110379512**	**0**
F	**0,039863155**	**0**	**0**	**0,077585167**	**0,110379512**	**1**	**0**
G	**0,013887727**	**0,019301473**	**0,014041809**	**0**	**0**	**0**	**1**

3.8.2 *Group Average HAC*

Hasil perhitungan TF-IDF yang berupa matriks keterhubungan dan nilai *similarity* antarkalimat digunakan sebagai acuan dalam penentuan *cluster*.

Langkah pertama dalam perhitungan *clustering* ini adalah menentukan nilai similaritas maksimal dari matriks *similarity* yang kemudian akan digabung menjadi satu *cluster*. Nilai similaritas *cluster* itu kemudian ditentukan dengan persamaan 2.6. Perhitungan ini terus dilakukan hingga seluruh dokumen membentuk cluster. Hasil perhitungan *clustering* ditampikan pada tabel 3.21 – 3.27. Pada tabel pembentukkan *cluster* terdapat perbedaan warna yang diberikan, bahwa kolom dengan warna merah berarti nilai itu tidak diikut sertakan dalam perhitungan, dan kolom yang berwarna kuning adalah yang berarti berisikan nilai maksimal yang akan diproses dengan perhitungan *group average HAC(Hierarchichal Agglomerative Clustering)* untuk menentukan nilai similaritas antar *cluster* yang terbentuk.

Tabel 3-21 Matrix Iterasi 1 pada pembentukan *cluster*

	A	B	C	D	E	F	G
A	1	0,0132	0,0096	0,0889	0	0,0399	0,0139
B	0,01323	1	0,0804	0	0	0	0,0193
C	0,00963	0,0804	1	0	0	0	0,014
D	0,08891	0	0	1	0	0,0776	0
E	0	0	0	0	1	0,1104	0
F	0,03986	0	0	0,0776	0,1104	1	0
G	0,01389	0,0193	0,014	0	0	0	1

Max **0,1104**

C1 E, F

Tabel 3-22 Matriks iterasi 2 pada pembentukan *cluster*

	A	B	C	D	C1	G
A	1	0,0132	0,0096	0,08891	0,0199	0,0399
B	0,01323	1	0,0804	0	0	0
C	0,00963	0,0804	1	0	0	0,014
D	0,08891	0	0	1	0,0388	0

C1	0,01993	0	0	0,0388	1	0
G	0,03986	0	0,014	0	0	1

Max **0,0889**

C2 A, D

Tabel 3-23 Matriks iterasi 3 pada pembentukan *cluster*

	C2	**B**	**C**	**C1**	**G**
C2	1	0,00661615	0,004813	0,029362	0,0140418
B	0,0066161	1	0,080427	0	0,0587242
C	0,0048132	0,080427	1	0	0,0140418
C1	0,0293621	0	0	1	0
G	0,0140418	0,05872416	0,014042	0	1

Max **0,080427**

C3 B, C

Tabel 3-24 Matriks iterasi 4 pada pembentukan *cluster*

	C2	**C3**	**C1**	**G**
C2	1	0,0057147	0,029362	0,014042
C3	0,0057147	1	0	0,008336
C1	0,0293621	0	1	0
G	0,0140418	0,00833582	0	1

Max **0,029362**

C4 C1, C2

Tabel 3-25 Matriks iterasi 5 pada pembentukan *cluster*

	C3	**C4**	**G**
C3	1	0,00285735	0,008336
C4	0,0028573	1	0,003472
G	0,0083358	0,00347193	1

Max **0,008336**

C5 G, C3

Tabel 3-26 Matriks iterasi 6 pada pembentukan *cluster*

	C4	C5
C4	1	0,00306221
C5	0,0030622	1

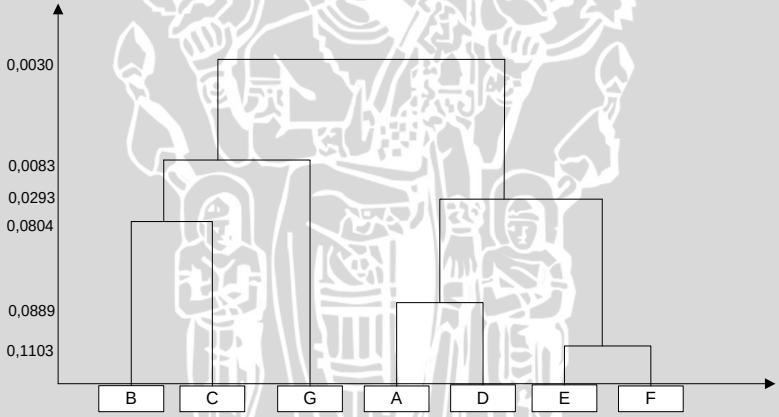
Max 0,003062

C6 C4, C5

Tabel 3-27 Hasil Pembentukkan *cluster*

C1	C2	C3	C4	C5	C6
E	A	B	C1	C3	C4
F	D	C	C2	G	C5

Hasil perhitungan *cluster* menghasilkan pembentukkan 6 cluster dengan beranggotakan kalimat – kalimat pada dokumen. Hasil *clustering* ini diperjelas dengan *dendogram* pada gambar 3.12.



Gambar 3-12 *Dendogram* Hasil *Cluster*

Dari hasil proses *clustering*, maka dipilih k *cluster* sesuai dengan keiinginan user, hal ini merujuk pada cara pengambilan jumlah *cluster* dalam algoritma HAC, pemilihan *cluster* menggunakan cara yang paling sederhana yaitu pemilihan seperti pada proses *flat clustering*. Sebagai contoh dipilih 3 yang terakhir

terbentuk dari data yang diinputkan. *Cluster* tersebut adalah *cluster* C3, C4 , dan kalimat G. Hasil pembentukkan *clustering* ini yang selanjutnya akan direpresentasikan sebagai *vertex*, setiap *cluster* akan mewakili setiap *vertex*. Hasil clustering dapat direpresentasikan dalam bentuk tabel 3.28.

Tabel 3-28 Hasil Pemilihan 3 *cluster* terbesar

Cluster	Anggota
C1= C3	B,C
C2= C4	A,D,E,F
C3= G	G

3.8.3 *Graph Contruction*

Setiap *cluster* yang telah terbentuk pada tahap *clustering* yang berisikan kalimat – kalimat, dianggap sebagai satuan *vertex*, dengan kata lain satu *cluter* mewakili satu *vertex*. Setiap *cluster* yang jarak antar *cluster* telah didapatka menggunakan persamaan *group Average*. Kemudian untuk penentuan nilai *edges* dari setiap *cluster* itu digunakan perhitungan dengan persamaan 2.7. Dari hasil pembentukkan *cluster* dapat dibentuk sebuah tabel yang berisi matrik jarak antar*cluster* seperti pada tabel 3.29 berikut.

Tabel 3-29 Matrix *similarity* antar*cluster*

	C1	**C2**	**C3**
C1	**1**	**0.00285735**	**0.00833582**
C2	**0.002857349**	**1**	**0.003471932**
C3	**0.00833582**	**0.00347193**	**1**

Dari hasil Matrix *similarity* antar*cluster* tersebut, kemudian dilakukan perhitungan *edges* untuk setiap *vertex* dengan menggunakan persamaan 2.7 dan 2.8, hasil perhitungan edges ini ditampilkan berupa matrix M. Hasil pembentukkan dari matrix inilah yang nantinya akan digunakan untuk perhitungan *lexrank*. Hasil perhitungan Matrix M ditampilkan pada tabel 3.30.

Tabel 3-30 Matrix M sebagai Hasil Perhitungan *edges*

	C1	C2	C3
C1	**1**	**0.21561**	**0.53315**
C2	**0.21561**	**1**	**0.25124**
C3	**0.53315**	**0.25124**	**1**

3.8.4 Perhitungan *Lexrank*

Hasil pembentukkan matrix M digunakan sebagai acuan dalam perhitungan bobot untuk masing-masing *cluster* menggunakan algoritma *LexRank*.

Langkah pertama dalam menentukan bobot tiap *cluster* ini adalah dengan memberikan nilai awal 1 kepada masing-masing *cluster* seperti pada tabel 3.31 sebagai nilai *vertex* untuk menghitung *error rate* nilai *vertex* menggunakan persamaan 2.11.

Tabel 3-31 Inisialisasi awal nila p(u)

u	p(u)
C1	**1**
C2	**1**
C3	**1**

Kemudian selanjutnya dilakukan iterasi menggunakan persamaan 2.11. Hasilnya ditampilkan pada tabel 3.32, dimana u adalah *cluster* yang dihitung bobot *vertex*nya dan p(u) adalah nilai bobot *vertex* akhir setiap *cluster*.

Tabel 3-32 ***Lexrank***

u	v	z	p(v)	sim(u,v)	Σsim(z,v)	temp	p(u)	error
c1	c2	c1,c3	0.342433845	0.472853481	0.73642674	0.219873922	0.340217758	0.0003502
	c3	c1,c2	0.318711147	0.26357326	0.527146519	0.159355573		
c2	c1	c2,c3	0.339867508	0.472853481	0.73642674	0.218226098	0.339970584	0.0024633
	c3	c1,c2	0.318711147	0.26357326	0.527146519	0.159355573		
c3	c1	c2,c3	0.339867508	0.26357326	0.73642674	0.12164141	0.319963533	0.0012524
	c2	c1,c3	0.342433845	0.26357326	0.73642674	0.122559923		

Dari hasil perhitungan LexRank, didapatkan nilai bobot masing-masing *vertex* atau *cluster*. Hasil perhitungan LexRank ini kemudian dilakukan pengurutan (*sorting*) berdasarkan nilai LexRank. Seperti yang ditampilkan pada tabel 3.33 berikut.

Tabel 3-33 Hasil Lexrank

cluster	p(u)
C1	0.34021776
C2	0.33997058
C3	0.31996353

3.8.5 *Sentences Scoring*

Setelah perhitungan nilai setiap cluster, untuk melanjutkannya pada perhitungan score kalimat digunakan pendekatan dengan menghitung *centroid* dari kalimat dalam *cluster* dan mengurutkan kalimat – kalimat berdasarkan *similarity* – nya dengan *centroid*–nya. *Centroid* dibuat dengan hanya menghitung kata – kata yang sama dengan *cluster* lainnya. Perhitungan *centroid* ini menggunakan persamaan 2.12. Hasil dari perhitungan *centroid* ditampilkan pada tabel 3.34 berikut.

Tabel 3-34Hasil Centroid

Centroid	Value
Cc1(ambruk) =	0.196906654
Cc2(ambruk) =	0.124124349
Cc1(tiang) =	0
Cc2(tiang) =	0
Cc3(ambruk) =	0.146443839
Cc3(tiang) =	0

Dari hasil perhitungan *centroid* tersebut, maka perhitungan dilanjutkan untuk menghitung *similaritas* antara kalimat dengan *centroidnya*. Perhintungan *similaritas* ini hanya menghitung antara

kalimat dengan *centroid* pada *cluster*nya, jadi kalimat tidak akan dihitung *similaritasnya* terhadap *centroid* pada *cluster* lainnya. Perhitungan ini menggunakan persamaan 2.14, dan hasil perhitungan ditampilkan pada tabel 3.35 berikut.

Tabel 3-35 *Similaritas* kalimat dengan *centroid*

Similarity	Value
sim(A, Cc2)	1
sim(B, Cc3)	1
sim(C, Cc3)	1
sim(D, Cc2)	1
sim(E, Cc1)	0
sim(F, Cc1)	0
sim(G, Cc3)	1

Kemudian dihitung nilai setiap kalimat dengan menggunakan persamaan 2.13. Nilai kalimat menandakan tingkat keutamaan suatu kalimat terhadap dokumen. Hasil perhitungan nilai kalimat ini kemudian dilakukan pengurutan (*sorting*) berdasarkan nilai kalimatnya. Hasil nilai setiap kalimat ditampilkan pada tabel 3.36 berikut.

Tabel 3-36 Nilai Kalimat

Kalimat	Score
A	0.340217758
D	0.340217758
B	0.339970584
C	0.339970584
G	0.319963533
E	0
F	0

Kemudian, proses selanjutnya adalah proses pemotongan hasil ringkasan sesuai dengan nilai ukuran ringkasan yang

ditentukan. Dalam perhitungan ini diberikan nilai presentase ringkasan sebesar 50%, maka didapatkan hasil ringkasan seperti pada tabel 3.37 berikut.

Tabel 3-37 Hasil Pemotongan Ringkasan

Kalimat	Score
A	0.340217758
D	0.340217758
B	0.339970584
C	0.339970584
G	0.319963533
E	0
F	0

Dapat dilihat pada tabel di atas dinyatakan nilai kalimat B, C, dan G adalah sama. Namun karena ringkasan yang diinginkan adalah 50%, maka hanya kalimat B dan C yang diikutsertakan pada ringkasan. Hal ini dikarenakan posisi kalimat dalam dokumen menentukan nilai pentingnya sebuah kalimat. Sebagai contoh, kalimat pada posisi pertama dalam dokumen akan dinyatakan dengan posisi maksimum dengan bernilai 1, dan kalimat pada posisi terakhir dalam dokumen akan dinyatakan dengan nilai 0. (Gunes, E., & Dragomir, R. R, 2004)

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

Sebuah tiang beton milik perusahaan telekomunikasi setinggi 30 meter ambruk di Jalan Raya Raci, Pakal, Surabaya.
Tiang sepanjang 30 meter itu juga ambruk memenuhi badan jalan sehingga arus lalu lintas dialihkan melewati jalan perkampungan.
Dua bangunan rumah rusak terkena ambrukan tiang itu
Tiang itu ambruk saat ditarik dua pekerjanya yang akan merubuhkan tiang itu kata Heri, salah satu warga kepada detiksurabayacom, Minggu (15/11/2009)

3.8.6 *Precision dan 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* : 3, jumlah *wrong* : 1, jumlah *missed* : 0. Sehingga dapat dihitung nilai *precision* dan *recall* sebagai berikut :

$$Precision = \frac{3}{3+1} = 0{,}75$$

$$Recall = \frac{3}{3+0} = 1$$

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 Intel Core™ i5 CPU M480 2.67GHz
2. RAM 4 GB DDR3.
3. *Harddisk* dengan kapasitas 500 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 ™ 32-bit
2. Microsoft Visual Studio 2010 sebagai editor C#

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 adalah kelas `Kalimat`, `Kata`, `Stemming`, `ClusterKalimat`, dan `ProsesClusterRank`.

1. Kelas `Kalimat`

Kelas Kalimat digunakan untuk merepresentasikan dokumen teks menjadi bentuk *string* kalimat dalam bentuk *array*, dengan beberapa fungsi yang digunakan untuk memproses dokumen. Dalam kelas ini terdapat fungsi memecah dokumen menjadi kalimat – kaliamat yang diberikan nilai *index* posisi kalimatnya. Terdapat ekstensi *method* kalimat untuk memproses *filtering stopword* yaitu penghilangan kata yang kurang penting.

2. Kelas `Kata`

Kelas kata digunakan untuk proses *tokenizing* memecah kalimat menjadi kata-kata yang digunakan dalam proses TF-IDF *similarity*. Dalam kelas kata terdapat ekstensi *method* kata yang didalamnya terdapat fungsi – fungsi untuk menghitung TF (*term frequency*), IDF, hingga *Cosine Similarity* antar kalimat. Nilai dari hasil perhitungan *Cosine Similarity* antar kalimat ini disimpan dalam bentuk struktur data `dictionary<Tuple <T1, T2>,double>` yang merupakan sebuah struktur data yang mempunyai dua tipe data dimana ada `Tuple<T1,T2>` yang merupakan pasangan dari kalimat – kalimat yang dihitung *cosine similarity*nya dan tipe data `double` yang menyimpan nilai *cosine similarity* antar kalimat tersebut.

3. Kelas `Stemming`

Kelas ini digunakan untuk melakukan proses *Stemming* atau perubahan 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 pada tanggal 1 Februari 2010.

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.

4. Kelas `ClusterKalimat`

Kelas `ClusterKalimat` digunakan untuk menghitung nilai *cosine similarity* antar *cluster*. Nilai ini disimpan dalam bentuk *arraylist*. Dalam kelas ini nilai

similarity antar *cluster* dihitung dengan perhitungan *group average HAC*.

5. Kelas `ProsesClusterRank`

   Kelas `ProsesClusterRank` digunakan untuk proses algoritma *ClusterRank*. Dalam kelas ini terdapat beberapa fungsi yaitu diantaranya fungsi pembentukkan *cluster* dengan menggabungkan *cluster* yang mana mempunyai jarak yang dekat dalam hal ini berarti mempunyai nilai *cosine similarity* yang maksimum. Hasil dari penggabungan *cluster* dari fungsi ini yang akan diproses di kelas `Tree` untuk selanjutnya ditampilkan pembentukkan *cluster* dalam bentuk *tree.* Selanjutnya terdapat fungsi `HitungMatrixM()`, yaitu fungsi untuk menghitung nilai *edges* antar *nodes* dari graf yang dibentuk dari hasil representasi *cluster* yang dipilih untuk diproses. Input dari proses ini adalah nilai dari *cosine similarity* antar *cluster* yang dipilih yang kemudian hasil perhitungannya disimpan dalam *arrayList.* Kemudian fungsi selanjutnya adalah fungsi `HitungScoreLex()` yang menghitung rangking tiap *cluster* dengan rumus *LexRank*. Fungsi `HitungCentroid()` yang berfungsi untuk menghitung *centroid* kata dari setiap *cluster*, yanf selanjutnya dilanjutkan pada proses `HitungSimilaritydanScore()` yang mana disini dihitung *similaritas* antara kalimat dengan *cluster*nya yang kemudian dihitung nilai setiap kalimat.

4.2.2 Tahap Preprocessing

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

Tahap *tokenizing* dilakukan dalam kelas `Kalimat` yang memuat *extension method* `ekstensikalimat` yang berfungsi untuk memecah dokumen menjadi *array* kalimat dan sekaligus memproses *filtering* stopword, fungsi tersebut terdapat pada gambar 4.1. Pada kelas `Kalimat` ini pun terdapat fungsi `Kalimat()` yang berfungsi untuk mendapatkan kata – kata dari dokumen yang sudah dipecah menjadi kalimat – kalimat yang telah diberikan *index* posisi kalimat, seperti yang terlihat pada gambar 4.2.

```
public static class EkstensiKalimat
    {

        static string[] kataBuangan =
(……).Split(new[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
  static string[] PemisahKalimat = new[] { ". ",
"?","! ","\r","\n"};

        static string[] KataBenarBenarBuang =
"detiksurabaya.com google.com
facebook.com".Split(new[] {" "},
StringSplitOptions.RemoveEmptyEntries);

        public static Kalimat[] PisahKalimat(this
string teks)
        {
            foreach (var kata in
KataBenarBenarBuang)
            {
                teks = teks.Replace(kata, "");
            }
            var kecil = teks.ToLower();
            foreach (var kata in kataBuangan)
            {
                kecil = kecil.Replace(kata, "");
            }
            var pisahanAsli =
teks.Split(PemisahKalimat,
StringSplitOptions.RemoveEmptyEntries).Where(x=>x!="
").ToArray();
            return
kecil.ToLower().Split(PemisahKalimat,
StringSplitOptions.RemoveEmptyEntries).Where(x =>
x.Replace(" ", "").Length > 0).Select((x, i) => new
Kalimat(x, i, pisahanAsli[i])).ToArray();
        }
    }
```

Gambar 4-1 ***Sourcecode extension method*** **`EkstensiKalimat`**

```
public Kalimat(string teks,int posisi,string teksAsli)
        {
            TeksAsli = teksAsli;
            Teks = teks;
            this.Posisi = posisi;
            var pisahan = teks.Split(PemisahKata,
StringSplitOptions.RemoveEmptyEntries).Select(x=>x.Stem(
)).ToArray();
```

```
            DaftarKata =
pisahan.Distinct().ToDictionary(x => x, x =>
pisahan.Count(y => y == x));
        }
```

Gambar 4-2 *Sourcecode* `fungsi Kalimat()`

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 4.3 berikut.

```
public String stemm(String kata)
        {
            String status, prefix, sec_prefix;
            String[] feedback;
            kata = particleRem(kata);
            kata = possessiveRem(kata);
            feedback =
firstPrefixRem(kata).Split(new[] { ";;" },
StringSplitOptions.None);
            kata = feedback[0];
            prefix = feedback[1];
            feedback = suffixRem(kata,
prefix).Split(new[] { ";;" },
StringSplitOptions.None);
            kata = feedback[0];
            status = feedback[1];
            if (status.Equals("1"))
            {
                feedback = secondPrefixRem(kata,
prefix).Split(new[] { ";;" },
StringSplitOptions.None);
                kata = feedback[0];
            }
            else
            {
                feedback = secondPrefixRem(kata,
prefix).Split(new[] { ";;" },
StringSplitOptions.None);
                kata = feedback[0];
                sec_prefix = feedback[1];
                feedback = suffixRem(kata,
sec_prefix).Split(new[] { ";;" },
StringSplitOptions.None);
```

```
            }
            return feedback[0];
        }
```

Gambar 4-3 *Sourcecode* fungsi `stemm()`

Dalam fungsi stemm tersebut terdapat beberapa fungsi stemming seperti particleRem(kata) untuk menghilangkan partikel (-lah, -kah, -pun, -tah), seperti gambar 4.4 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 4-4 *Sourcecode* fungsi `particleRem()`

Selanjutnya adalah fungsi `possesiveRem()` yang berfungsi untuk menghilangkan kata ganti milik (-ku, -mu, -nya), seperti yang ditampilkan pada gambar 4.5 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 4-5 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 4.7. dan fungsi `secondPrefixRem()` yang digunakan untuk menghapus awalan bertingkat (memper-, diper-) seperti yang ditunjukkan pada gambar 4.6. Langkah terakhir dari proses *stemming* yaitu `fungsisuffixRem()` digunakan untuk menghapus akhiran (-i, -kan, –an) ditunjukkan pada gambar 4.8.

```
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-3);
        sec_prefix = suku_kata;
      }
     }
   }
   else if (!prefix.Equals("null"))
   {
     return firstPrefixRem(kata);
   }


     kata = kata + ";;" + sec_prefix;
```

```
      return kata;
    }
```

Gambar 4-6 *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-2);
      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-3);
        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-4);
         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-3);
}
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-2);
}
}
}
}
kata = kata + ";;" + prefix;
return kata;
}
```

Gambar 4-7 *Sourcecode* fungsi firstPrefixRem()

```
public String suffixRem(String kata, String prefix)
{
int hasSuffix = 0;
String suku_kata;
if (kata.Length > 2)
{
if (kata[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);
}
}
}
}
}
kata = kata+";;" + hasSuffix;
return kata;
}
```

Gambar 4-8 *sourcecode* `fungsi suffixRem()`

4.2.3 Tahap Pembobotan TF – IDF dan *Cosine Similarity*

Tahap selanjutnya adalah melakukan penghitungan nilai *cosine similarity* antarkalimat menggunakan algoritma TF-IDF dan *vector space models*. Untuk melakukan proses perhitungan ini, digunakan kelas Kata. Parameter input untuk proses ini adalah kalimat – kalimat yang telah mengalami proses *Preprocessing* sebelumnya. Pada kelas Kata ini terdapat di dalamnya yaitu *extension method* `EkstensiKata` yang mana di dalam *method* ekstensi ini seluruh proses TF – IDF dan *cosine similarity* dilakukan. Seperti yang terlihat pada gambar 4.9 berikut yang berisikan proses perhitungan *term frequency* untuk masing – masing kata pada dokumen, kemudian perhitungan bobot untuk masing – masing kata terhadap masing – masing dokumen. Selanjutnya dilakukannya proses perhitungan nilai *cosine similarity* antar kalimat yang satu dengan lainnya, dan hasil dari nilai *cosine similarity* ini disimpan dalam tipe data dokumen yang baru.

```
public static class EkstensiKata
```

```
{
public static Dictionary<string,Kata> HitungKata(this
Kalimat[] daftarKalimat)
{
Kalimat.DaftarKalimat = daftarKalimat;
var daftarKataUnik =
daftarKalimat.First().DaftarKata.Select(x => x.Key);
foreach (var kalimat in daftarKalimat)
{
daftarKataUnik =
daftarKataUnik.Union(kalimat.DaftarKata.Keys);
}
var daftarKata = daftarKataUnik.ToDictionary(x=>x,x=>
new Kata(x));
//Untuk menghitung tf - idf
foreach (var kata in daftarKata.Keys.ToArray())
{
var teksYangDicari = kata;
var tf = daftarKalimat.ToDictionary(k => k,
k => k.DaftarKata.ContainsKey(teksYangDicari) ?
k.DaftarKata[teksYangDicari] : 0);
var df =
daftarKalimat.Count(k=>k.DaftarKata.ContainsKey(teksYang
Dicari));
var idf = Math.Log(daftarKalimat.Count() /
(double)df,2);
daftarKata[teksYangDicari] = new Kata(teksYangDicari,
tf, df, idf);
}
Kata.DaftarKata = daftarKata.Values.ToArray();
//Untuk menghitung bobot kata
foreach (var kalimat in daftarKalimat)
{
var maxTF = kalimat.MaxTF;
foreach (var kata in daftarKata.Values.ToArray())
{
kata.TFNormalisasi[kalimat] = kata.TF[kalimat] /
(double)maxTF;
kata.Bobot[kalimat] = kata.TFNormalisasi[kalimat] *
kata.IDF;
}
}
//Untuk menghitung Cosine Similarity
foreach (var kalimat in daftarKalimat)
{
var total = 0.0;
```

```
foreach (var kata in daftarKata.Values)
{
total += kata.Bobot[kalimat];
}
kalimat.TotalKuadratBobot = total;
}
Matrix.Cosine = new Dictionary<Tuple<Kalimat, Kalimat>,
double>();
foreach (var kalimatA in daftarKalimat)
{
foreach (var kalimatB in daftarKalimat)
{
if (kalimatA==kalimatB)
{
continue;
}
var total = 0.0;
foreach (var kata in daftarKata.Values)
{
total += kata.Bobot[kalimatA] * kata.Bobot[kalimatB];
}
Matrix.Cosine.Add(new Tuple<Kalimat, Kalimat>(kalimatA,
kalimatB), total/
Math.Sqrt(kalimatA.TotalKuadratBobot*kalimatB.TotalKuadr
atBobot)  );
}
}
return daftarKata;
}
}
```

Gambar 4-9 *SourceCode* `kelas Kata`

4.2.4 Tahap *ClusterRank*

Tahap selanjutnya adalah tahap *ClusterRank*. Pada tahap ini terdiri dari beberapa proses, yaitu diantaranya proses *clustering*, pembentukkan Matrik M yang merupakan representasi dari nilai *edges* setiap *nodes*, perhitungan *Lexrank*, perhitungan *centroid* kata , dan perhitungan nilai kalimat. Perhitungan dari proses *clustering* menggunakan metode *group average HAC*. Proses pembentukkan *cluster* membutuhkan parameter input yang berupa nilai – nilai *cosine similarity* antar kalimat. Proses perhitungan *clustering* terdapat pada kelas `ClusterKalimat`, seperti yang terlihat pada gambar 4.10. Proses *clustering* ini membutuhkan kelas tambahan yaitu kelas `Tree` dan kelas `ProsesClusterRank` yang berisikan

fungsi konstruktor `gabungcluster()` yang digunakan untuk merepresentasikan bentuk *cluster* dalam bentuk *tree* seperti yang terlihat pada gambar 4.11 dan 4.12.

```
public class ClusterKalimat
    {
        public double SkorLex=1;

        public static List<ClusterKalimat>
DaftarCluster;

        public Cabang cabang;

        public List<Kalimat> Anggota;

public bool Contains(string kata)
        {
 return Anggota.Any(x =>
x.DaftarKata.ContainsKey(kata));
        }
        public double Cosine(ClusterKalimat kedua)
        {
            var pengali = 1 /
(double)(this.Anggota.Count * kedua.Anggota.Count);
            var totalSim = 0.0;
            foreach (var kalimatA in this.Anggota)
            {
                foreach (var kalimatB in
kedua.Anggota)
                {
                    totalSim += Matrix.Cosine[new
Tuple<Kalimat, Kalimat>(kalimatA, kalimatB)];
                }
            }
            return pengali * totalSim;
        }
    }
}
```

Gambar 4-10 *Sourcecode* Kelas `ClusterKalimat`

```
public static class Tree
    {
        public static List<Cabang> anggota;
    }
    public class Cabang
    {
```

```
        public Cabang cabang1;
        public Cabang cabang2;
        public Kalimat nilai;

        public TreeNode TN
        {
            get
            {
                var tn = new TreeNode();
                if (nilai!=null)
                {
                    tn.Text = "Kalimat" +
(nilai.Posisi+1).ToString();
                }
                if (cabang1!=null)
                {
                    tn.Nodes.Add(cabang1.TN);
                }
                if (cabang2!=null)
                {
                    tn.Nodes.Add(cabang2.TN);
                }
                return tn;
            }
        }
    }
```

Gambar 4-11 *Sourcecode* Kelas `Tree`

```
public static List<ClusterKalimat> Proses(int
targetCluster)
        {
            Tree.anggota =
Kalimat.DaftarKalimat.Select(x => new Cabang() {
nilai = x }).ToList();
            var i = 0;
            ClusterKalimat.DaftarCluster =
Kalimat.DaftarKalimat.Select(kal => new
ClusterKalimat() { cabang = Tree.anggota[i++],
Anggota = new List<Kalimat>(new[] { kal })
}).ToList();
            Kalimat kalA = null;
            Kalimat kalB = null;
            var max = double.MinValue;
            if (ClusterKalimat.DaftarCluster.Count
> targetCluster)
            {
                foreach (var kalimatA in
Kalimat.DaftarKalimat)
```

```
                {
                    foreach (var kalimatB in
Kalimat.DaftarKalimat)
                    {
                        if (kalimatA == kalimatB)
                        {
                            continue;
                        }
                        var nilai =
Matrix.Cosine[new Tuple<Kalimat, Kalimat>(kalimatA,
kalimatB)];
                        if (nilai > max)
                        {
                            kalA = kalimatA;
                            kalB = kalimatB;
                            max = nilai;
                        }
                    }
                }
                var clusterA =
ClusterKalimat.DaftarCluster.First(x =>
x.Anggota.Contains(kalA));
                var clusterB =
ClusterKalimat.DaftarCluster.First(x =>
x.Anggota.Contains(kalB));
                gabungkanCluster(clusterA,
clusterB);
            }

            while
(ClusterKalimat.DaftarCluster.Count >
targetCluster)
            {
                ClusterKalimat clusterA = null;
                ClusterKalimat clusterB = null;
                max = double.MinValue;
                foreach (var clA in
ClusterKalimat.DaftarCluster)
                {
                    foreach (var clB in
ClusterKalimat.DaftarCluster)
                    {
                        if (clA == clB)
                        {
                            continue;
                        }
```

```
                        var cosine =
clA.Cosine(clB);
                        if (cosine > max)
                        {
                            max = cosine;
                            clusterA = clA;
                            clusterB = clB;
                        }
                    }
                }
                gabungkanCluster(clusterA,
clusterB);
            }
            return ClusterKalimat.DaftarCluster;
        }
private static void gabungkanCluster(ClusterKalimat
clusterA, ClusterKalimat clusterB)
        {
            var cabangBaru = new Cabang() { cabang1
= clusterA.cabang, cabang2 = clusterB.cabang };

clusterA.Anggota.AddRange(clusterB.Anggota);
            Tree.anggota.Remove(clusterA.cabang);
            Tree.anggota.Remove(clusterB.cabang);
            Tree.anggota.Add(cabangBaru);
            clusterA.cabang = cabangBaru;

ClusterKalimat.DaftarCluster.Remove(clusterB);
        }
```

Gambar 4-12 *Sourcecode* fungsi `gabungcluster()`

Selanjutnya adalah proses pembentukan matrix M yaitu menghitung nilai *edges* antar *nodes* yang mana *nodes* ini adalaha representasi graf dari *cluster*. Proses ini dilakukan pada kelas `ProsesClusterRank` pada fungsi `HitungMatrixM()` seperti yang ditunjukkan pada gambar 4.13.

```
public static void HitungMatrixM(this
List<ClusterKalimat> daftar)
        {
            Matrix.M = new
Dictionary<Tuple<ClusterKalimat, ClusterKalimat>,
double>();
            var totalJarak = 0.0;
            for (int indexA = 0; indexA <
daftar.Count() - 1; indexA++)
            {
                for (int indexB = indexA + 1;
```

```
indexB < daftar.Count(); indexB++)
                {
                    totalJarak +=
daftar[indexA].Cosine(daftar[indexB]);
                }
            }
            foreach (var clA in daftar)
            {
                foreach (var clB in daftar)
                {
                    if (clA == clB)
                    {
                        continue;
                    }
                    var edge =
clA.Cosine(clB)/totalJarak;
                    var m = edge * 0.85 + 0.15 /
daftar.Count;

                    Matrix.M.Add(new
Tuple<ClusterKalimat,ClusterKalimat>(clA,clB),
                        m);
                }
            }
        }
```

Gambar 4-13 *Sourcecode* fungsi HitungMatrixM()

Kemudian proses dilanjutkan untuk menghitung nilai *LexRank* dari setiap *cluster* seperti yang ditunjukkan pada gambar 4.14. Setelah perhitungan *LexRank*, proses dilanjutkan pada perhitungan *centroid* kata – kata di setiap *cluster* dan dihitung pula *similaritas* kalimat dengan *clusternya*, hingga tahap terakhir adalah perhitungan nilai dari masing – masing kalimat. Kalimat – kalimat ini akan diurutkan mulai dari kalimat yang mempunyai nilai tertinggi hingga terendah, seperti yang ditunjukkan pada gambar 4.15 – 4.16.

```
private static void
HitungScorLex(List<ClusterKalimat> daftar)
        {
            var berubah = false;
            do
            {
                foreach (var clusterU in daftar)
                {
                    var totalTemp = 0.0;
                    foreach (var clusterV in
```

```
daftar.Where(x => x != clusterU))
                    {
                        var clusterZ =
daftar.Where(x => x != clusterU && x !=
clusterV).ToArray();
                        totalTemp +=
clusterV.SkorLex * Matrix.M[new
Tuple<ClusterKalimat, ClusterKalimat>(clusterU,
clusterV)] /
                                clusterZ.Sum(z =>
Matrix.M[new Tuple<ClusterKalimat,
ClusterKalimat>(z, clusterV)]);
                    }
                    var skorBaru = 0.85 /
daftar.Count + 0.15 * totalTemp;
                    if (skorBaru !=
clusterU.SkorLex)
                    {
                        berubah = true;
                        clusterU.SkorLex =
skorBaru;
                    }
                }
            }
            while (!berubah);
        }
```

Gambar 4-14 *Sourcecode* fungsi `HitungScoreLex()`

```
 private static void
HitungCentroid(List<ClusterKalimat> daftar)
        {
            foreach (var kata in
Kata.DaftarKata.Where(k => daftar.Count(cl =>
cl.Contains(k.Teks)) > 1))
            {
                foreach (var cluster in
daftar.Where(cl => cl.Contains(kata.Teks)))
                {
                    var nilai = 0.0;
                    foreach (var clB in
daftar.Where(cl => cl.Contains(kata.Teks) && cl !=
cluster))
                    {
                        nilai += clB.SkorLex *
Matrix.M[new Tuple<ClusterKalimat,
ClusterKalimat>(cluster, clB)] * kata.IDF;
                    }
                    kata.Centroid.Add(cluster,
```

```
nilai);
                }
            }
        }
```

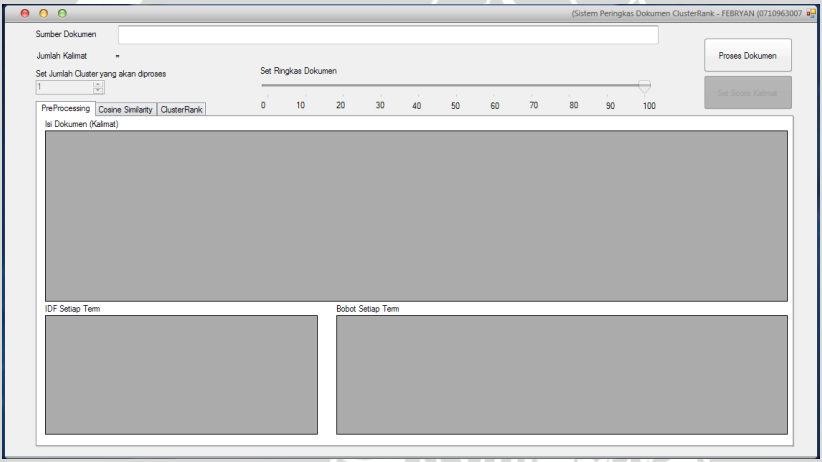
Gambar 4-15 *Sourcecode* fungsi `HitungCentroid()`

```
private static void
HitungSimilarityDanScore(List<ClusterKalimat>
daftar)
        {
            foreach (var cluster in daftar)
            {
                foreach (var kalimat in
cluster.Anggota)
                {
                    var pengali = 0.0;
                    var pembagi1 = 0.0;
                    var pembagi2 = 0.0;
                    foreach (var kata in
Kata.DaftarKata.Where(k =>
k.Centroid.ContainsKey(cluster) &&
kalimat.DaftarKata.ContainsKey(k.Teks)))
                    {
                        pengali += kata.IDF *
kata.Centroid[cluster];
                        pembagi1 += kata.IDF *
kata.IDF;
                        pembagi2 +=
kata.Centroid[cluster] * kata.Centroid[cluster];
                    }
                    kalimat.Similarity = pengali /
(Math.Sqrt(pembagi1) * Math.Sqrt(pembagi2));
                    kalimat.Similarity =
double.IsNaN(kalimat.Similarity) ? 0 :
kalimat.Similarity;
                    kalimat.Score =
kalimat.Similarity * cluster.SkorLex;
                    kalimat.Score =
double.IsNaN(kalimat.Score) ? 0 : kalimat.Score;
                }
            }
        }
```

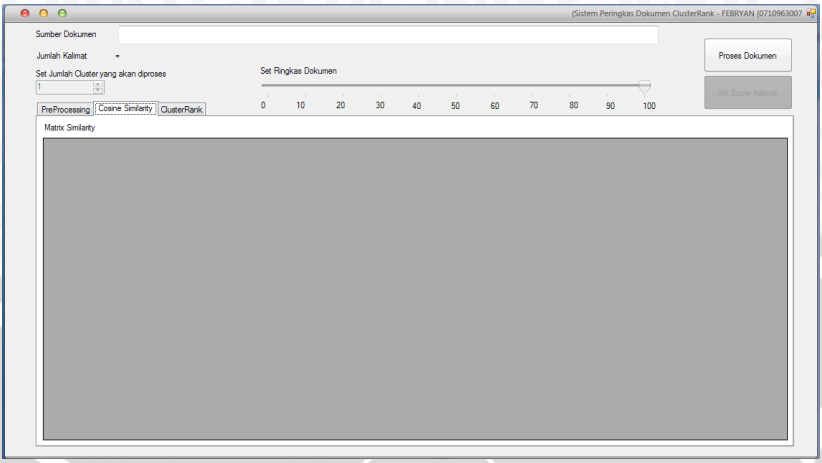
Gambar 4-16 *Sourcecode* fungsi `HitungSimilaritydanScore()`

4.3 Implementasi Antarmuka

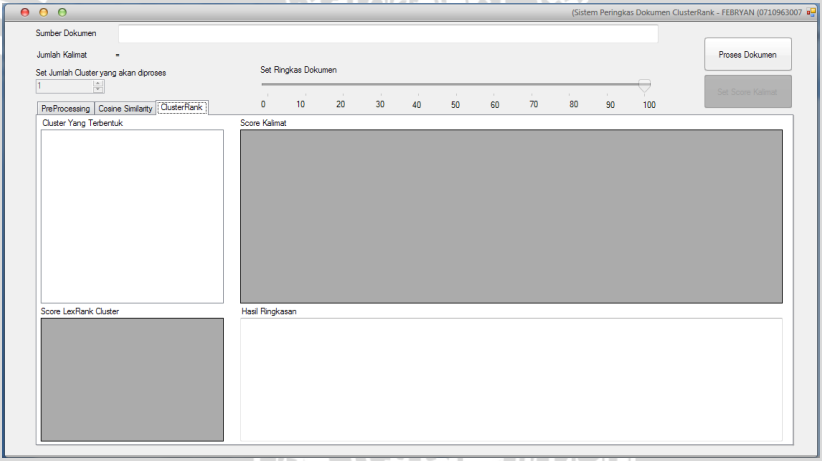
Berdasarkan rancangan antarmuka pada sub bab 3.2 maka dihasilkan antarmuka yang ditunjukkan pada gambar 4.17 - 4.19. Pada form utama terdapat *field* ”sumber dokumen”, tombol ”Proses Dokumen”, tombol ”*Set Score* kalimat” , *slider* ”ukuran ringkasan” dan *numericUpDown* ”Set Jumlah *Cluster*”. Dalam form utama ini terdapat *tabcontrol* yang berisikan tiga halaman tab, yang mana halaman *tab* 1 berisikan tabel ”bobot setiap kata”, tabel ”isi kalimat dari dokumen”, dan tabel ”hasil IDF kata”. Kemudian pada halaman *tab* 2 terdapat tampilan untuk *matrix simiularity* antar kalimat. Kemudian pada halaman *tab* ke-3 terdapat *treeview* yang menampilkan *tree* hasil clustering, tabel “*ScoreLexRank*”, tabel “*Score* Kalimat”, dan *textbox* “Hasil Ringkasan”.



Gambar 4-17 Form Utama *Tab* 1



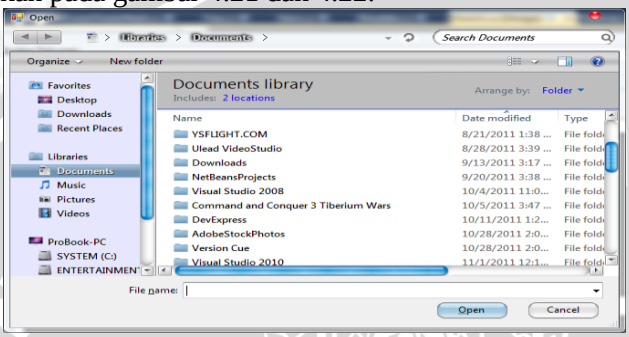
Gambar 4-18 Form Utama *Tab* 2



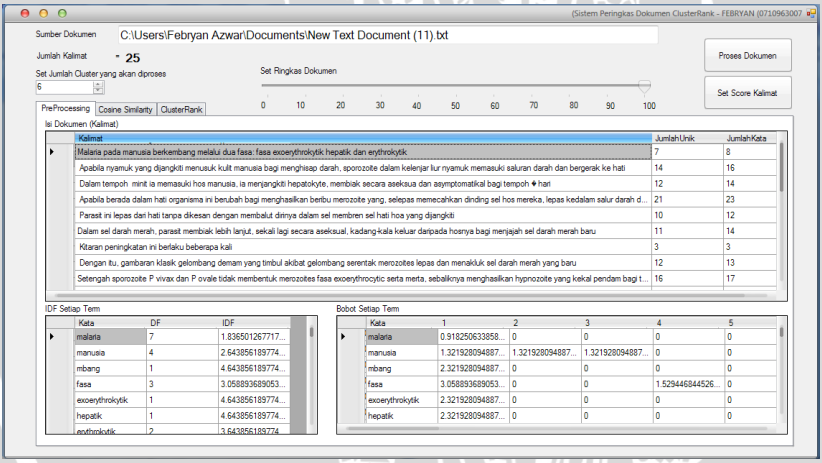
Gambar 4-19 Form Utama *Tab* 3

Tombol ”Proses Dokumen” digunakan untuk menentukan dokumen mana yang akan dilakukan proses peringkasan dokumen. Jika tombol proses dokumen ditekan, maka pertama akan muncul *dialog box* ”open” yang ditampilkan pada gambar 4.20. *User* akan melakukan pemilihan dokumen yang akan dijadikan sumber peringkasan dokumen, setelah *user* menentukan pilihan dan menekan tombol ”Open” pada *dialog box*, maka *field* ”sumber dokumen” akan berisi *path* file teks yang dijadikan sumber ringkasan. Jika proses

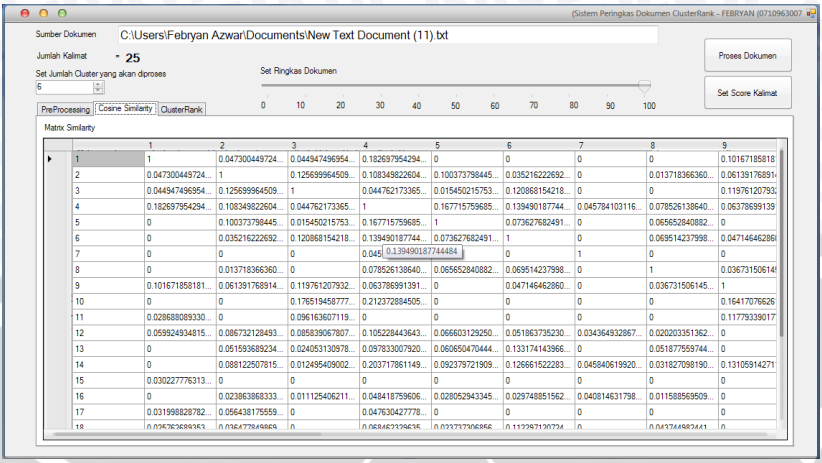
pemilihan dokumen berhasil dilakukan, maka selanjutnya proses selanjutnya adalah langsung pada tabel hubungan (*similarity*) antarkalimat, dan isi kalimat – kalimat dari dokumen yang dipilih. Hasil informasi dokumen yang diproses dan pembentukan matriks ditampilkan pada gambar 4.21 dan 4.22.



Gambar 4-20 Diaolog Box "Open"

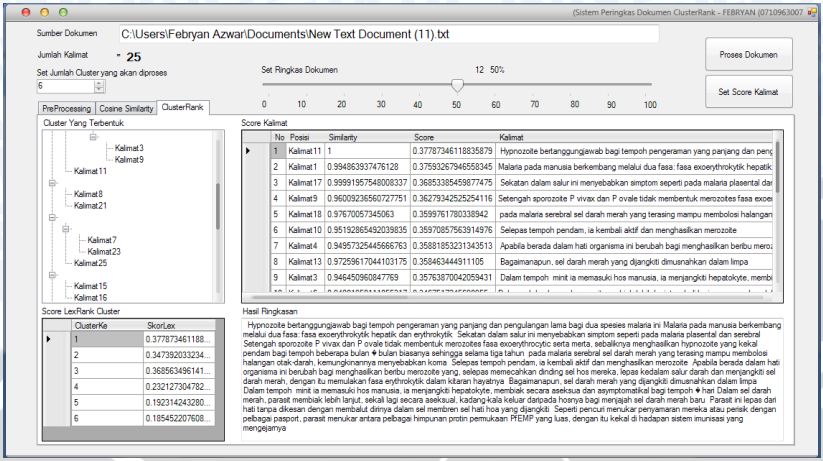


Gambar 4-21 Isi Dokumen & perhitungan Bobot kata



Gambar 4-22 Hasil *Matrix Similarity* Antar Kalimat

Setelah matriks *cosine similarity* antarkalimat terbentuk, maka tombol "*set score* kalimat", *numericUpDown* untuk set *cluster* dan *slider* "ukuran Ringkasan" akan berfungsi. Langkah selanjutnya adalah user menentukan jumlah *cluster* yang akan dipilih untuk diproses selanjutnya, manampilkan nilai kalimat dengan menekan tombol "*set score* kalimat" dan ukuran ringkasan dengan menggeserkan *slider* "ukuran ringkasan". Nilai kalimat akan muncul pada tabel dan dihasilkan ringkasan dokumen yang ditampilkan pada textbox "hasil ringkasan" seperti yang tampak pada gambar 4.23.



Gambar 4-23 Hasil Ringkasan

4.4 Analisa Hasil dan Pembahasan

4.4.1 Hasil Uji Coba

Hasil uji coba pada aplikasi peringkasan dokumen tunjukkan pada tabel 4.1 hingga 4.4. Percobaan dilakukan dengan memberikan nilai pembentukkan *cluster* yang diinginkan mulai dari 4 *cluster*, 6 *cluster*, 8 *cluster*, 10 *cluster*, dan 12 *cluster*. Kemudian ada nilai *threshold* ringkasan yang terdiri dari 25%, 50%, dan 75% dari jumlah dokumen awal.

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

Kode Dokumen	4 Cluster		6 Cluster		8 Cluster		10 Cluster		12 Cluster	
	P	R	P	R	P	R	P	R	P	R
2	0.500	0.273	0.500	0.273	0.333	0.182	0.500	0.273	0.667	0.364
3	0.000	0.000	0.500	0.286	0.250	0.143	0.250	0.143	0.000	0.000
4	0.333	0.200	0.833	0.500	0.833	0.500	0.667	0.400	0.667	0.400
5	0.500	0.300	0.333	0.200	0.667	0.400	0.500	0.300	0.500	0.300
6	0.286	0.222	0.429	0.333	0.143	0.111	0.143	0.111	0.429	0.333
7	0.500	0.300	0.333	0.200	0.500	0.300	0.333	0.200	0.500	0.300
8	0.250	0.200	0.625	0.500	0.500	0.400	0.250	0.200	0.375	0.300
9	0.750	0.429	0.500	0.286	0.250	0.143	0.500	0.286	0.750	0.429
10	0.375	0.231	0.500	0.308	0.250	0.154	0.125	0.077	0.375	0.231
11	0.167	0.125	0.167	0.125	0.167	0.125	0.500	0.375	0.000	0.000
rata-rata	**0.366**	**0.228**	**0.472**	**0.301**	**0.389**	**0.246**	**0.377**	**0.236**	**0.426**	**0.266**

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

Kode Dokumen	4 Cluster		6 Cluster		8 Cluster		10 Cluster		12 Cluster	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
2	0.538	0.636	0.462	0.545	0.615	0.727	0.385	0.455	0.538	0.636
3	0.500	0.571	0.500	0.571	0.500	0.571	0.375	0.429	0.375	0.429
4	0.417	0.500	0.417	0.500	0.583	0.700	0.583	0.700	0.583	0.700
5	0.333	0.400	0.333	0.400	0.417	0.500	0.417	0.500	0.417	0.500
6	0.429	0.667	0.357	0.556	0.357	0.556	0.357	0.556	0.429	0.667
7	0.333	0.400	0.500	0.600	0.333	0.400	0.333	0.400	0.417	0.500
8	0.375	0.600	0.375	0.600	0.375	0.600	0.375	0.600	0.438	0.700
9	0.571	0.571	0.571	0.571	0.286	0.286	0.571	0.571	0.429	0.429
10	0.438	0.538	0.313	0.385	0.375	0.462	0.438	0.538	0.438	0.538
11	0.250	0.333	0.250	0.333	0.333	0.444	0.333	0.444	0.333	0.444
rata-rata	**0.418**	**0.522**	**0.408**	**0.506**	**0.417**	**0.525**	**0.417**	**0.519**	**0.440**	**0.554**

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

Kode Dokumen	4 Cluster		6 Cluster		8 Cluster		10 Cluster		12 Cluster	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
2	0.538	0.636	0.462	0.545	0.615	0.727	0.385	0.455	0.538	0.636
3	0.500	0.571	0.500	0.571	0.500	0.571	0.375	0.429	0.375	0.429
4	0.417	0.500	0.417	0.500	0.583	0.700	0.583	0.700	0.583	0.700
5	0.333	0.400	0.333	0.400	0.417	0.500	0.417	0.500	0.417	0.500
6	0.429	0.667	0.357	0.556	0.357	0.556	0.357	0.556	0.429	0.667
7	0.333	0.400	0.500	0.600	0.333	0.400	0.333	0.400	0.417	0.500
8	0.375	0.600	0.375	0.600	0.375	0.600	0.375	0.600	0.438	0.700
9	0.571	0.571	0.571	0.571	0.286	0.286	0.571	0.571	0.429	0.429
10	0.438	0.538	0.313	0.385	0.375	0.462	0.438	0.538	0.438	0.538
11	0.250	0.333	0.250	0.333	0.333	0.444	0.333	0.444	0.333	0.444
rata-rata	**0.418**	**0.522**	**0.408**	**0.506**	**0.417**	**0.525**	**0.417**	**0.519**	**0.440**	**0.554**

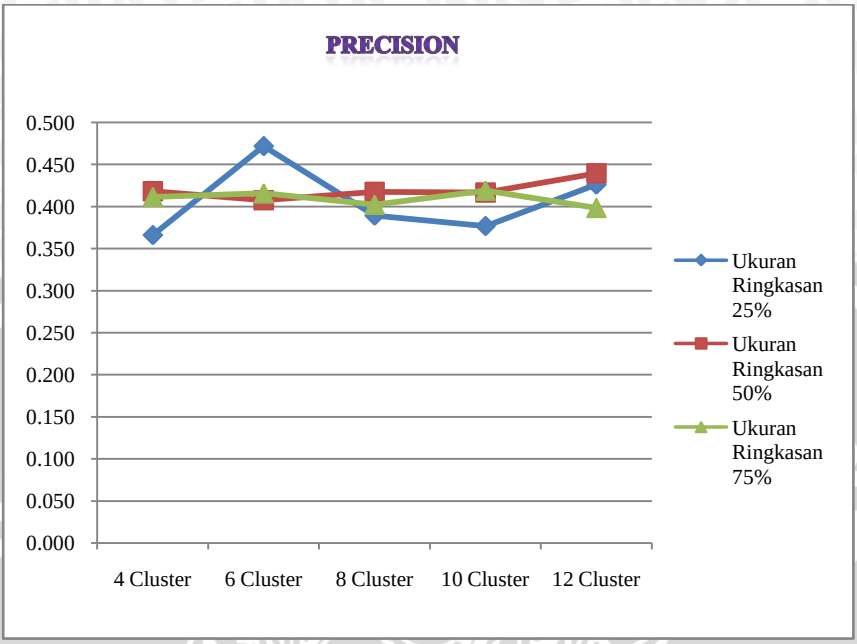
Hasil uji coba (tabel 4.1) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi. Sesuai dengan rancangan yang ada pada subbab 3.1, percobaan pertama digunakan *treshold summary* atau panjang ringkasan sebesar 25% dari dokumen asli.

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

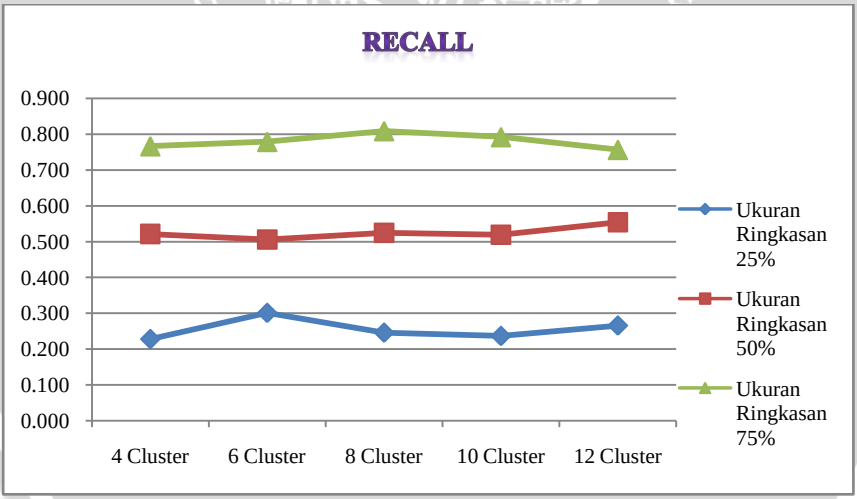
Tabel ketiga (tabel 4.3) menunjukkan perbandingan *precision* dan *recall* pada percobaan peringkasan dokumen menggunakan aplikasi dengan *treshold summary* atau panjang ringkasan 75% dari teks asli.

Dari hasil uji coba terhadap aplikasi ini, diketahui rata-rata *precision* sebesar 0.412 dan rata-rata *recall* sebesar 0.520. Rata-rata nilai *precision* dan *recall* untuk masing-masing *treshold*, *treshold similarity* maupun *treshold summary* dinyatakan dalam grafik akan tampak seperti gambar 4.24 dan 4.25 berikut, dimana sumbu x menyatakan besarnya *treshold* jumlah *cluster* dan sumbu y menyatakan besarnya nilai *precision* dan *recall*.

Rata-rata nilai *precision* tertinggi 0,472 terjadi pada saat percobaan dengan nilai *treshold* pembentukkan 6 *cluster* dan ukuran ringkasan 25% dokumen awal. Sedangkan rata-rata nilai *recall* terendah 0,228 terjadi pada saat percobaan dengan nilai *treshold* pembentukkan 4 *cluster* dan ukuran ringkasan 25% dokumen awal.



Gambar 4-24 grafik *precision* pada aplikasi



Gambar 4-25 grafik *recall* pada aplikasi

4.4.2 Analisa Hasil Secara Keseluruhan

Secara keseluruhan, metode *ClusterRank* menghasilkan rata-rata nilai *precision* sebesar 0,412 dan rata-rata nilai *recall* sebesar 0,520. Dari hasil uji coba yang dilakukan, diketahui beberapa hal yang berpengaruh terhadap data uji coba yang dihasilkan. Antara lain metode *stemming* yang dipakai, dan metode pemilihan *cluster*.

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 "pelayanan" pada dokumen 9 yang tersusun atas kata dasar "layan" dan mendapat akhiran "-an" dan awalan "pe-", dan kata "layanan" yang tersusun atas kata dasar "layan" dan mendapat akhiran "-an". Jika dilakukan *stemming* pada aplikasi ini, maka hasil yang didapat dari kata "pelayanan" adalah kata dasar "ayan" karena dianggap memiliki awalan "pel-", beda hasilnya pada kata "layanan" akan dihasilkan kata dasar "layan". Contoh lainnya adalah apabila bertemu dengan kata dengan mempunya huruf akhir "i", dimana pada metode *stemming* diberlakukan aturan untuk menghilangkan imbuhan "i", tetapi masalahnya proses ini akan menghapus semua kata yang mengandung huruf terakhir "i" karena tidak adanya kamus kata dasar yang digunakan sebagai pembanding, seperti kata "tinggi" akan diproses dan menghasilkan kata "tingg".

Kekurangan pada metode *stemming* ini akan berpengaruh terhadap nilai *cosine similarity* antarkalimat. Karena dua kata yang sebenarnya satu kata dasar tadi tidak terhitung sebagai kata atau *term* yang sama, sehingga akan mempengaruhi nilai *cosine similarity* yang dihasilkan. Disamping itu, Metode pemilihan *cluster* sangat berpengaruh terhadap ringkasan, hal ini dikarenakan memang metode pemilihan *cluster* menggunakan metode *simple HAC* dan tidak terlalu efisien dalam pembentukkan *cluster*, sehingga *cluster* yang terbentuk tidak terlalu baik, terbukti pada beberapa dokumen saat dibentuk *cluster* seperti yang terjadi pada dokumen 9, pada saat

dibentuk 6 *cluster*, ada dua *cluster* yang hanya mempunyai masing – masing satu anggota, dikarenakan pemilihan *cluster* terpilih sesuai dengan hasil *record* perhitungan *group average*.

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 seperti contoh yang terjadi pada percobaan terhadap dokumen 9.

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang dihasilkan dalam skripsi ini adalah :

1. Telah diimplementasi sistem peringkasan dokumen otomatis pada teks berbahasa Indonesia yang menggunakan algoritma *ClusterRank* dengan memanfaatkan penggabungan algoritma *group average* HAC (*Hierarchical Agglomerative Clustering*) sebagai metode *clustering* kalimat – kalimat pada dokumen, dan metode *Lexrank* sebagai metode perangkingan graf.
2. Akurasi yang dihasilkan sistem ditunjukkan dengan nilai rata – rata keseluruhan dari *precision* dan *recall* sebesar 0,412 dan 0,520. Nilai *precision* tertinggi dihasilkan pada *threshold* ringkasan 25% dan *threshold cluster* 6 yaitu 0,472. Untuk nilai *recall* tertinggi dihasilkan pada *threshold* ringkasan 75% dan *threshold cluster* 12 yaitu 0,554. Nilai *precision* dan *recall* bergantung pada kesesuain hasil ringkasan sistem dengan hasil ringkasan manusia. Pada penelitian ini dokumen dengan kode dokumen 4 menghasilkan nilai *precision* paling tinggi yaitu 0,833 pada *threshold* ringkasan 25% dan *threshold cluster* 6 dan 8. Hal ini menunjukkan bahwa kalimat penting yang diekstrak sistem pada dokumen 4 hampir sesuai dengan hasil ringkasan manusia.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan :

1. Pengembangan sistem dapat dikembangkan kembali dengan penggunaan metode *stemming* yang berbeda, seperti Nazief – Andriani.
2. Pengembangan sistem dapat dikembangkan untuk data uji berupa dokumen yang banyak atau *multi document*.
3. Pembentukkan *cluster* dan pemilihan *cluster* dapat dikembangkan dengan beberapa metode lainnya.

DAFTAR PUSTAKA

Agusta, L. 2009. Perbandingan Algoritma *Stemming Porter* dengan Algoritma *Nazief & Andriani* untuk *Stemming* Dokumen teks Bahasa Indonesia. Konferensi Nasional Sistem dan Informatika, 196-201

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

Christopher D. Manning, et al.2009. *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Feng, Ao. 2007. *Document Clustering - an Optimization Problem*. Amherst: Computer Science Department University of Massachusetts

Garg, N., B. Favre., K. Reidhammer., & D. Hakkani. 2009. *ClusterRank: A Graph Based Method for Meeting Summarization.* International Computer Science Institute. Berkeley: USA

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

Hayuhardika, W. 2010. *Sistem Peringkas Dokumen Otomatis Berbahasa Indonesia dengan Metode Lexrank:Graph Based Summarization Algorithm.* Malang: Program Studi Ilmu Komputer Fakultas MIPA Universitas Brawijaya.

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.

Mimaroglu, S. 2006. *Agglomerative Clustering*. Bahcesehir University: Computer Engineering Department

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

Strehl,A,et al.2000.*Impact of Similarity Measures on Web-Page Clustering*. Proceeding of the Workshop of Artificial Intelligent for Web Search, 17th National Conference on Artificial Intelligence,2000.

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

LAMPIRAN

LAMPIRAN 1

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	com	sehingga	tentang
dari	Masih	seorang	bukan
dalam	Hari	bagi	agar
akan	Hanya	besar	semua
pada	mengatakan	lagi	sedang
juga	Kepada	selama	kali
saya	Kami	antara	kemudian
ke	Setelah	waktu	hasil
karena	melakukan	sebuah	sejumlah
tersebut	Lalu	jika	juta
bisa	Belum	sampai	persen
ada	Lain	jadi	sendiri
mereka	Dia	terhadap	katanya
lebih	kalau	Tiga	demikian
kata	terjadi	serta	masalah
tahun	banyak	pun	mungkin
sudah	menurut	salah	umum
atau	jalan	merupakan	setiap
saat	anda	atas	bulan
oleh	hingga	sejak	bagian
menjadi	tak	membuat	bila
orang	baru	baik	lainnya

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

datang	Merasa	datang	posisi
senin	berbagai	senin	asal
rasa	mengenai	rasa	sekali
sebesar	tingkat	digunakan	Sesuai
berat	awal	justru	Bagaimana
dirinya	sedikit	padahal	Berarti
memberi	nanti	menyebutkan	Keluar
pagi	pasti	gedung	tanggal
sabtu	muncul	apalagi	terutama
ternyata	dekat	program	menerima
mencari	lanjut	milik	penting
sumber	ketiga	teman	bertemu
ruang	biasa	menjalani	usai
menunjukkan	dulu	keputusan	menghadapi
biasanya	kesempatan	upaya	pula
nama	ribu	mengetahui	
sebanyak	akhir	mempunyai	
utara	membantu	berjalan	
berlangsung	terkait	menjelaskan	
barat	sebab	mengambil	
kemungkinan	menyebabkan	benar	
yaitu	khusus	lewat	
berdasarkan	bentuk	belakang	
sebenarnya	ditemukan	ikut	
cara	diduga	barang	
utama	mana	meningkatkan	
pekan	ya	kejadian	
terlalu	kegiatan	kehidupan	
membawa	sebagian	keterangan	
kebutuhan	tampil	penggunaan	
suatu	hampir	masing-masing	

LAMPIRAN 2
DOKUMEN UJI

Kode dokumen : 04
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.

Kode Dokumen : 05

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.

Kode Dokumen : 06

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.

Kode Dokumen : 02

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.

Kode Dokumen : 07

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.

Kode Dokumen : 03

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.

Kode Dokumen : 08

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.

Kode Dokumen : 09

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.

Kode Dokumen : 10

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?

Kode Dokumen : 11

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).