BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan yang pesat di bidang teknologi informasi terutama internet, telah menimbulkan lonjakan informasi yang hebat. Hal ini terjadi karena internet memungkinkan banyak orang untuk memproduksi, memanipulasi, mengakses dan menyebarluaskan informasi dengan "mudah". Setiap hari jutaan orang di dunia menggunakan internet, baik untuk mencari dan memanipulasi informasi yang sudah ada, maupun untuk menciptakan dan menyebarkan informasi baru. Akibatnya, informasi tentang apapun, mulai dari yang sangat berguna bagi manusia atau yang sekedar "sampah", tersedia melimpah di internet.

Usaha untuk memperoleh informasi secara digital telah banyak dilakukan dan perkembangannya sangat pesat seiring dengan perkembangan teknologi komputer. Salah satu cara untuk memperoleh informasi yang seimbang seperti apa yang diinginkan adalah dengan membaca beberapa dokumen yang membahas topik yang sama. Akan tetapi cara ini menyulitkan pembaca untuk menangkap topik bahasan utama dari dokumen - dokumen tersebut karena harus mengingat – ingat isi dokumen yang telah dibaca sebelumnya. Pembaca harus mengintegrasikan dahulu dokumen – dokumen yang tetah dibaca didalam pikirannya sebelum dapat merangkum maksud dan topik utama dokumen – dokumen tersebut secara keseluruhan.

Dengan perkembangan teknologi informasi baik hardware maupun software kesulitan untuk mendapatkan sebuah informasi diharapkan tidak lagi menjadi sebuah kendala. Adanya internet membuat orang mudah untuk mendapatkan informasi.

Dalam proses penelusuran informasi melalui internet sering diperoleh informasi yang sangat banyak, tetapi sebagian besar diantaranya adalah informasi yang tidak dibutuhkan. Oleh karena itu, dari sudut pandang temu kembali informasi (*information retrieval*), semakin banyaknya informasi yang tersedia di internet justru semakin mempersulit untuk menemukan kembali informasi yang

relevan, yaitu informasi yang sesuai dengan kebutuhan. Dalam suatu sistem temu kembali informasi, kemampuan untuk menemukan informasi yang tersedia diukur dengan *recall* dan kemampuan untuk menemukan informasi yang relevan diukur dengan ketelitian, maka proses penelusuran dalam situasi seperti tersebut di atas akan menghasilkan *recall* yang tinggi tetapi ketelitian rendah.

Sistem yang tepat untuk masalah tersebut adalah Sistem Temu Kembali Informasi yang dapat menghasilkan integrasi dari beberapa dokumen elektronik yang berbeda dengan topik bahasan yang sama secara otomatis. Proses integrasi akan menghasilkan dokumen baru yang mengandung semua bagian dari dokumen – dokumen awal, namun memiliki susunan antar kalimat serta antar paragraf yang berbeda. Perbedaan ini karena saat proses integrasi topik – topik bahasan yang serupa (*similar*) dari semua dokumen dikumpulkan menjadi satu paragraf dan disusun ulang kalimat per kalimat sesuai dengan besarnya kesamaan (*similiarity*) antar kata (*term*). Dengan membaca hasil integrasi diharapkan pembaca dapat terbantu dalam menyerap informasi penting yang ada dalam kumpulan dokumen yang berbeda dan tidak perlu lagi membaca sekumpulan dokumen satu per satu.

Berdasarkan latar belakang diatas, maka dalam penelitian ini akan dibangun aplikasi Sistem Temu Kembali Informasi Bahasa Indonesia dengan Algoritma *Clustering Single Pass*. Diharapkan aplikasi yang dibuat akan lebih menghemat waktu dan memudahkan pencarian dokumen Bahasa Indonesia oleh pemakai.

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang dapat dirumuskan adalah dalam Sistem Temu Kembali Informasi Bahasa Indonesia, Operator sistem adalah pengguna yang tidak mempunyai latar belakang pengetahuan *query* yang cukup. Sehingga kebutuhan pengguna adalah kemudahan pencarian dokumen yang sejenis.

Untuk hal tersebut, maka permasalahan yang diangkat dalam penelitian ini adalah:

1. Bagaimana mengekstrak konsep dari suatu koleksi dokumen dengan klastering dokumen.
2. Bagaimana memanfaatkan klastering untuk mengekspand *query*.
3. Bagaimana klastering dokumen dapat diaplikasikan untuk suatu Sistem Temu Kembali Informasi.

1.3. Batasan Masalah

Dalam penelitian ini ada beberapa pembatasan masalah yang dilakukan, yaitu :

1. Dokumen yang digunakan adalah dokumen teks yaitu halaman arsip berita di http://www.kompas.com/archive.
2. Dalam penelitian ini digunakan Algoritma *Clustering Single Pass* untuk klasifikasi dokumen.

BAB II
TUJUAN DAN MANFAAT PENELITIAN

2.1. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Mengembangkan teknik klastering dengan Algoritma *Clustering Single Pass.*
2. Mengimplementasikan metode mengekstrak dan mengindeks teks dari koleksi dokumen teks menggunakan teknik klastering.
4. Menguji kinerja klastering menggunakan *recall* dan *precision*.

2.2. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah:

1. Dapat digunakan sebagai alat bantu untuk pencarian informasi untuk mendapatkan dokumen yang sama dengan dokumen sebelumnya.
2. Menghemat waktu pencarian informasi untuk mendapatkan dokumen yang mirip dengan dokumen sebelumnya.

BAB III
TELAAH PUSTAKA

3.1. Tinjauan Pustaka

Perkembangan internet yang sangat pesat dan kapasistas dokumen online yang besar menjadikan banyak riset yang membahas tentang sistem temu kembali informasi (information retrieval). Yue W, dkk (2007) dalam penelitiannya mengusulkan algoritma sistem temu kembali informasi (*information retrieval*) berbasis *query expansion* dan klasifikasi. Algoritma tersebut diinduksi dari *query* yang pendek dan metode pencarian informasi tradisional (*traditional retrieval information method*) yang menghasilkan presisi yang rendah walaupun tingkat recall cukup tinggi. Penelitian ini berusaha untuk mendapatkan lebih banyak dokumen yang relevan dengan mengekspand *query* (*query expansion*) dan klasifikasi dokumen. Hasil dari penelitian dengan algoritma query expansion dan klasifikasi dokumen mengahasilkan ketepatan dan efisiensi pencarian dokumen dibandingakan dengan metode query expand tradisional.

Fakta penelitian yang selama 20 tahun menunjukkan bahwa sistem yang berbasis indeks teks dengan menggunakan pembobotan sebuah dokumen yang tepat mengahasilkan pencarian lebih tepat. Menurut Salton, dkk (1988) hasilnya sangat tergantung dari efektifitas sistem pembobotan dokumen. Dalam penelitian ini merangkum informasi yang berhubungan dengan otomatisasi pembobotan dokumen dan menyediakan dasar pembobotan dokumen secara tunggal (*single-term-weghting*) dibandingan dengan menggunakan prosedur lain yang lebih rumit.

Steinbach M, dkk (2007) melakukan penelitian dengan memberikan sebuah pendekatan baru yang lebih mudah dipahami dan dapat digunakan untuk membentuk klaster. Pendekatan tersebut dimotivasi dari sebuah penelitian bahwa jika ada sebuah pola yang besar dalam sebuah data, pola ini biasanya terbagi dalam klaster yang berbeda jika menggunakan pendekatan klaster yang sudah

ada. Hal ini dikarenakan alagoritma yang dikembangkan tidak memanfaatkan pengetahuan tentang pola dan sering kali tujuan bertolak belakang dengan pemeliharaan pola (preserving pattern). Misalnya dengan meminimalkan jarak terdekat dari sentral klaster. Pada penelitian ini difokuskan untuk memberikan ciri, yaitu dengan menggunakan pola klaster dan memberikan cara yang paling efektif dalam proses klastering. Penelitian dilakukan dengan mengevaluasi dua algoritma klastering yaitu metode *Hierarchical Clustering* dan metode *Bisecting K-Mean Clustering*. Dari penelitian dengan mengunakan metode *Hierarchical Clustering* dalam melakukan proses klastering memberikan hasil lebih baik dibandingkan dengan menggunakan metode K-Means.

Karypis, dkk (2004), melakukan analisis bahwa algoritma klastering yang cepat dan berkualitas tinggi sangat berperan dalam menyediakan pedoman dan mekanisme pencarian dalam mengelola informasi yang luas untuk ukuran klaster yang kecil yang sangat berarti. Secara khusus algoritma klastering membangun hierarki koleksi dokumen yang sangat besar dengan alat yang ideal untuk interaktif visualisasi dan pengembangan data-view yang konsisten. Secara khusus, algoritma klastering sangat berarti dalam membangun hasil hierarki dalam koleksi dokumen yang sangat besar. Algoritma tersebut adalah alat yang ideal untuk penyajian visualisasi dan eksplorasi ketika kita memberikan data-view yang sesuai dengan prediksi, dan pada tingkat ketelian yang berbeda. Penelitian ini fokus pada algoritma klastering dokumen yang benar-benar membangun solusi dengan hierarchical dan memberikan kajian yang lengkap tentang algoritma *Agglomerative* dan algoritma *Partitional* dengan menggunakan kriteria yang berbeda dan menggabungkan skema dan menyajikan kelas (class) baru dari algoritma klastering. Yaitu dengan cara menggabungkan fitur dari kedua algoritma yaitu partitional dan agglomeratif. Dengan pendekatan tersebut akan mengurangi kesalahan pada tahap awal dengan menggunakan metode agglomerative meningkatkan hasil klastering. Hasil penelitian menunjukkan bahwa algoritma *agglomerative* lebih baik dibandingkan dengan metode algoritma *partitional*. Dengan menggunakan *Clustering Single Pass* membuat ideal dalam mengklaster koleksi dokumen yang sangat besar dengan komputasi

yang relatif rendah, tetapi menghasilkan klaster dengan hasil kualitas tinggi. Selanjutnya metode *Agglomerative* secara konsisten dibatasi lebih baik dibandingkan dengan metode *Agglomerative* tunggal dan beberapa kasus metode tersebut lebih unggul dari metode partitional.

Zhang, J., dkk (2001) *Fusion* dan dua pendekatan klaster untuk meningkatkat efektifitas informasi. Dalam fusion daftar peringkat digabungkan bersama-sama dengan arti yang bermacam-macam. Yang mendasari bahawa sistem temu kembali informasi yang berbeda akan saling melengkapi satu sama lain karena biasanya menekankan fitur permintaan yang berbeda ketika menentukan relevansi dan mengambil set dokumen yang berbeda. Dalam klastering, dokumen diklaster dengan baik sebelum maupun sesudah temu kembali informasi. Yang mendukung adalah bahwa dokumen yang serupa cenderung lebih relevan dengan query yang sama sehingga pendekatan ini cenderung lebih relevan dengan mengidentifikasi klaster dokumen yang serupa. Pada penelitian ini disajikan sebuah teknik fusion (penggabungan) yang dapat dikombinasikan dengan klastering untuk mendapatkan perbaikan dalam pencapaian klaster dari pendekatan konvensional. Metode yang digunakan menggunakan tiga langkah yaitu : (1) Similaritas Klastering Dokumen (Clustering Similar Documents), (2) Perangking Hasil Temu Kembali (Re-rangking Retrieval result), (3) Menggabungkan hasil pencarian (c*ombining retrieval result*).

Setelah evalusasi beberapa kali dari effet suatu kinerja steming dalam sistem temu kembali dokumen dalam Bahasa Indonesia, menurut Tala (2003) dapat disimpulkan sbb:

1. Stemmer Porter untuk Bahasa Indonesia menghasilkan banyak kata-kata yang tidak dimengerti yang disebabkan karena ambiguitas dalam morfologi Bahasa Indonesia. Dalam beberapa kasus menjadikan kesalahan kinerja dalam stemming. Apabila dikembangkan dengan kamus digital hal ini tidak sesuai karena dengan kamus digital akan sangat mahal. Akhirnya dilakukan penelitian lebih lanjut dengan memperluas

stemmer dengan kata-kata kejadian akan memberikan hasil yang lebih baik.

2. Seperti dalam Bahasa Inggris, dimotivasi dari metode Stammer yang dikembangkan Nazief untuk Bahasa Indonesia, memiliki 2 kendala yaitu : kemampuan stammer tergantung dari ukuran kamus. Tidak terdapat kata dasar dalam kamus. Kendala yang kedua adalah deteksi kata dasar dalam tata bahasa tidak memberikan hasil yang optimal untuk tujuan sistem temu kambali informasi. Apabila digunakan untuk sistem temu kembali informasi, stammer ini harus ditingkatkan dengan menggunakan alat lain seperti analisis tata bahasa atau menambahkan kamus tertentu.
3. Kata dasar yang memiliki arti sama dalam bahasa Indonesia perlu diberikan perlakuan khusus untuk mendapatkan manfaat dari proses Stemming. Penelitian morfologi lebih lanjut perlu dilihat kata dasar dalam Bahasa Indonesia karena tidak sama dengan Bahasa Belanda (Dutch) sehingga stammer porter dapat digunakan jika dilakukan kombisasi. Dan jika digunakan untuk sistem temu kembali informasi lebih kompleks kata dasar yang dibutuhkan.
4. Dari analisis yang dilakukan dari banyak pertanyaan, ditemukan bahwa sangat kecil variasi yang ditemukan dalam kata dasar. Rata-rata jumlah ragam kata dasar tertentu (seperti nama diri) dari semua pertanyaan hanya sekitar 4:135. Hal ini sangat kecil dibandingkan dengan Bahasa Slovene. Dalam penelitian juga ditemukan bahwa dengan menggunakan pengembangan stammer porter dan stammer Belanda jumlah kata dasar dan affiks lebih sedikit dibandingkan jika digunakan Stammer Slovenia. Dari hasil kinerja stammer penelitian yang dilakukan tersebut yang membedakan dengan penelitian yang lain.

Kegagalan dari hasil penelitian yang dilakukan mendeteksi perbedaan tidak selalu signifikan dibadingkan dengan sistem. Disadari bahwa penelitian yang dilakukan jauh dari sempurna karena penelitian bahwa korporasi tersebut dilakukan oleh dua orang. Juga diketahui bahwa dalam rumus yang dibuat

mengandung banyak arti yang sama. Sehingga perlu dilakukan tes pada sejumlah penelitian yang berbeda untuk koleksi yang lain.

3.2. Klastering Dokumen

Clustering (klastering) didefinisikan sebagai upaya mengelompokkan data ke dalam klaster sedemikian sehingga data-data di dalam klaster yang sama memiliki lebih kesamaan dibandingkan dengan data-data pada klaster yang berbeda. Bisa juga diartikan sebagai proses untuk mendefinisikan pemetaan / m*apping* $f : D \longrightarrow C$ dari beberapa data $D = \{t_1, t_{2, \ldots\ldots} t_n\}$ kedalam beberapa cluster $C=\{ c_{1,} c_{2,\ldots,} c_n\}$ berdasarkan kesamaan antar t_i . Sebuah klaster adalah sekumpulan obyek yang digabung bersama karena persamaan atau kedekatannya.

Klastering biasa digunakan pada banyak bidang, seperti : *data mining*, *pattern recognition* (pengenalan pola), *image classification* (pengklasifikasian gambar), ilmu biologi, pemasaran, perencanaan kota, pencarian dokumen, dan lain sebagainya.

Tujuan dari klastering adalah untuk menentukan pengelompokan dari suatu set data. Akan tetapi tidak ada ”ukuran terbaik” untuk pengelompokan data. Untuk pengelompokkan data tergantung tujuan akhir dari klastering, maka diperlukan suatu kriteria sehingga hasil klastering seperti yang diinginkan.

Penelitian tentang *clustering document* (klastering dokumen) telah banyak dilakukan. Secara umum klastering dokumen adalah proses mengelompokkan dokumen berdasarkan kemiripan antara satu dengan yang lain dalam satu klaster (Gordon, 1991; Ellis, 1996).

Tujuan klastering dokumen adalah untuk memisahkan dokumen yang relevan dari dokumen yang tidak relevan (Jian Zhang, dkk., 2001). Atau dengan kata lain, dokumen-dokumen yang relevan dengan suatu *query* cenderung memiliki kemiripan satu sama lain dari pada dokumen yang tidak relevan, sehingga dapat dikelompokkan ke dalam suatu klaster.

Klastering dokumen dapat dilakukan sebelum atau sesudah proses temu kembali (Jian Zhang, dkk., 2001). Pada klastering dokumen yang dilakukan sebelum proses temu kembali informasi, koleksi dokumen dikelompokkan ke dalam klaster berdasarkan kemiripan (*similarity*) antar dokumen. Selanjutnya

dalam proses temu kembali informasi, apabila suatu dokumen ditemukan maka seluruh dokumen yang berada dalam klaster yang sama dengan dokumen tersebut juga dapat ditemukan.

Dalam Sistem Temu Kembali Informasi, klastering dokumen memberikan beberapa manfaat, antara lain:

1. Mempercepat pemrosesan *query* dengan menelusur hanya pada sejumlah kecil anggota atau wakil klaster, sehingga dapat mempercepat proses temu kembali informasi
2. Membantu melokalisasi dokumen yang relevan
3. Membentuk kelas-kelas dokumen sehingga mempermudah penjelajahan dan pemberian interpretasi terhadap hasil penelusuran
4. Meningkatkan efektivitas dan efisiensi temu kembali informasi dan memberikan alternatif metode penelusuran

Selain itu, penggabungan antara penelusuran secara menyeluruh (*full search*) dengan penelusuran berbasis klaster (*cluster-based retrieval*) dapat meningkatkan ketelitian sampai dengan 25%. Hal yang sama dikemukakan oleh (Jian Zhang, dkk., 2001) bahwa penggabungan antara metode pengklasteran dengan *fusion* (pemberian peringkat terdadap dokumen secara keseluruhan) akan meningkatkan efektivitas temu kembali informasi.

Pada algoritma klastering, dokumen akan dikelompokkan menjadi *klaster-klaster* berdasarkan kemiripan satu data dengan yang lain. Prinsip dari *klastering* adalah memaksimalkan kesamaan antar anggota satu klaster dan meminimumkan kesamaan antar anggota *klaster* yang berbeda .

3.2.1 Metode Clustering Single Pass

Metode *Clustering Single Pass* adalah metode yang menggunakan strategi disain *Bottom-Up* yang dimulai dengan meletakkan setiap obyek sebagai sebuah klaster tersendiri (atomic klaster) dan selanjutnya menggabungkan atomic klaster – atomic klaster tersebut menjadi klaster yang lebih besar dan lebih besar lagi sampai akhirnya semua obyek menyatu dalam sebuah klaster atau proses dapat pula berhenti jika telah mencapai batasan kondisi tertentu.

Metode ini menggunakan strategi disain *Bottom-Up* yang dimulai dengan meletakkan setiap obyek sebagai sebuah *klaster atomic* dan selanjutnya menggabungkan *atomic klaster – atomic klaster* tersebut menjadi *klaster* yang lebih besar dan lebih besar lagi sampai akhirnya semua obyek menyatu dalam sebuah *klaster* atau proses dapat pula berhenti jika telah mencapai batasan kondisi tertentu.

Langkah-langkah dalam algoritma *Clustering Single Pass* untuk mengelompokkan *N* objek (item/variabel):

1. Mulai dengan *N klaster*r, setiap klaster mengandung entiti tunggal dan sebuah matriks simetrik dari jarak (similarities) *D* = {*dik*} dengan tipe *NxN*.
2. Cari matriks jarak untuk pasangan klaster yang terdekat (paling mirip). Misalkan jarak antara klaster *U* dan *V* yang paling mirip adalah *duv.*
3. Gabungkan klaster *U* dan *V*. Label klaster yang baru dibentuk dengan (*UV*).

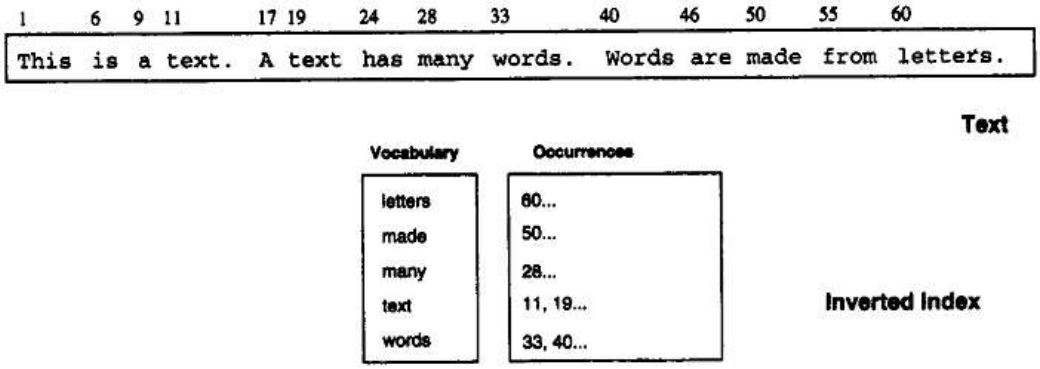
   Update entries pada matrik jarak dengan cara :

   a. Hapus baris dan kolom yang bersesuaian dengan klaster *U* dan *V*
   b. Tambahkan baris dan kolom yang memberikan jarak-jarak antara klaster (*UV*) dan klaster-klaster yang tersisa.
4. Ulangi langkah 2 dan 3 sebanyak (*N-1*) kali. (Semua objek akan berada dalam klaster tunggal setelah algoritma berahir). Catat identitas dari klaster yang digabungkan dan tingkat-tingkat (jarak atau similaritas) di mana penggabungan terjadi.

3.3. Index Inverted

Inverted file atau index inverted adalah mekanisme untuk pengindeksan kata dari koleksi teks yang digunakan untuk mempercepat proses pencarian. Struktur *inverted file* terdiri dari dua elemen, yaitu: kata (*vocabulary*) dan kemunculan (*occurences).* Kata-kata tersebut adalah himpunan dari kata-kata yang ada pada teks, atau merupakan ekstraksi dari kumpulan teks yang ada.

Dan tiap kata terdapat juga informasi mengenai semua posisi kemunculannya (*occurences*) secara rinci. Posisi dapat merefer kepada posisi kata ataupun karakter. Hal ini dapat dilihat dengan jelas dengan memperhatikan Gambar 3.1.



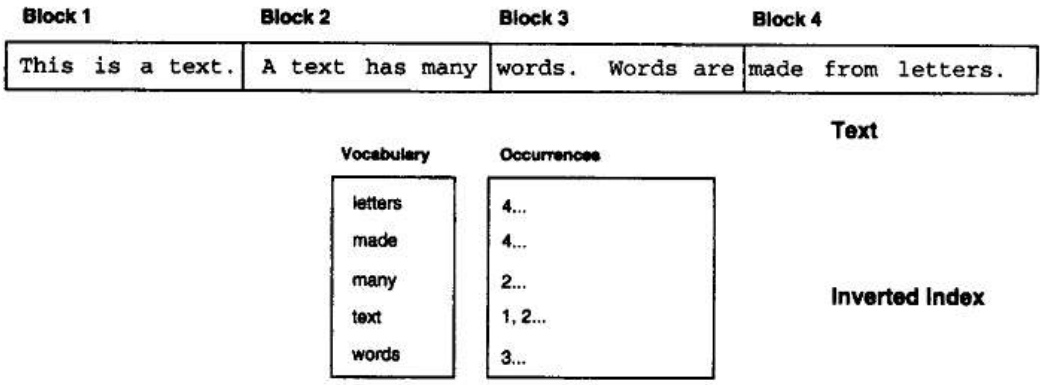
Gambar 3. 1 Contoh Teks dan I*nverted File*-nya

Pada gambar 3.1 kata-kata dikonversikan menjadi karakter huruf kecil (*lower-case).* Kolom vocabulary adalah kata-kata yang telah diekstraksi dari dari koleksi teks, sedangkan *occurences* adalah posisi kemunculan teks. Struktur inverted file seperti ini masihlah sangat sederhana sehingga berikutnya muncul beberapa masalah.

Nilai kemunculan dari kata-kata memerlukan ruangan (*space)* yang tidak sedikit, karena tiap kata muncul pada teks sekali pada struktur *occurences,* sehingga ada ruangan ekstra atau dilambangkan dengan *O(n*). Walaupun tidak semua kata diindekskan karena ada kata-kata *stopword* yang dibuang, overhead yang muncul akibat penambahan indeks ini sampai mencapai 30% sampai dengan 40% dari ukuran besar koleksi teks.

Sehingga kemudian untuk mengurangi kebutuhan ruangan yang besar tersebut, maka digunakanlah teknik yang disebut *block addressing.* Teknik pengindeksan ini sama seperti teknik klasik sebelumnya yang disebut *full inverted indices,* karena tetap sama elemennya yaitu *vocabulary* dan *occurences.* Namun perbedaan yang ada dan membuat teknik ini lebih unggul adalah pada pengalamatanya yang tidak satu-satu pada tiap kata seperti yang dilakukan oleh

teknik yang klasik, namun pengalamatannya berdasarkan blok-blok tertentu yang sudah didefiniskan. Hal ini akan lebih jelas dengan memperhatikan Gambar 3.2.



Gambar 3. 2 Contoh Teks dan I*nverted File*-nya

Dengan teknik ini kebutuhan ruangan untuk membuat tambahan pengindeksan akan lebih berkurang, karena dapat dipastikan bahwa jumlah blok akan lebih sedikit dibandingkan dengan jumlah keseluruhan kata. Secara eksperimental hanya diperlukan 5% dari koleksi teks untuk membuat tambahan pengindeksan dengan teknik ini. Sungguh efisien dalam penggunaan ruangan atau *space demand.* Namun *trade off* yang terjadi adalah pada tiap kali me-*retriev* kata maka yang akan di tunjuk adalah alamat blok kata tersebut. Sehingga harus dilakukan iterasi berikutnya pada blok tersebut untuk menemukan kata yang dimaksud. Tapi *trade off* ini tidak perlu dikhawatirkan karena tidak begitu banyak berpengaruh terhadap sistem karena hanya merupakan komputasi perbandingan sederhana jika dibandingkan efek positif yang sangat baik karena mampu mengefisienkan ruangan yang dibutuhkan untuk pengindeksan.

3.4. Stemming

Stemming merupakan suatu proses untuk menemukan kata dasar dari sebuah kata. Dengan menghilangkan semua imbuhan (*affixes)* baik yang terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*) dan *confixes* (kombinasi dari awalan dan akhiran) pada kata turunan. *Stemming* digunakan

untuk mengganti bentuk dari suatu kata menjadi kata dasar dari kata tersebut yang sesuai dengan struktur morfologi Bahasa Indonesia yang baik dan benar.

Imbuhan (*affixes*) pada Bahasa Indonesia lebih kompleks bila dibandingkan dengan imbuhan (*affixes*) pada Bahasa Inggris. Karena seperti yang telah disebutkan di atas bahwa imbuhan (*affixes*) pada Bahasa Indonesia terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*), bentuk perulangan (*repeated forms*) dan *confixes* (kombinasi dari awalan dan akhiran). Imbuhan-imbuhan yang melekat pada suatu kata harus dihilangkan untuk mengubah bentuk kata tersebut menjadi bentuk kata dasarnya. *Stemming* teks berbahasa Indonesia memiliki beberapa masalah yang sangat khusus terhadap bahasa. Salah satu masalah tersebut adalah perbedaan tipe dari imbuhan-imbuhan (*affixes*), yang lain adalah bahwa awalan (*prefixes*) dapat berubah tergantung dari huruf pertama pada kata dasar. Sebagai contoh ”me-” dapat berubah menjadi ”mem-” ketika huruf pertama dari kata dasar tersebut adalah ”b”, misalnya ”membuat” (*to make*), tetapi ”me-” juga dapat berubah menjadi ”meny-” ketika huruf pertama dari kata dasar melekat adalah ”s”, misalnya ”menyapu” (to sweep). Selanjutnya ketika ada lebih dari satu imbuhan (*affixes*) yang melekat pada suatu kata, maka urutan untuk menghilangkan imbuhan-imbuhan (*affixes*) pada kata tersebut menjadi sangat penting. Jika dalam proses meghilangkan imbuhan-imbuhan (*affixes*) tersebut kita tidak memperhatikan urutan penghilangan imbuhan-imbuhan (*affixes*) tersebut, maka kata dasar yang benar dari kata tersebut tidak akan ditemukan. Sebagai contoh pada kata ”*di*-beri-*kan*” (*to be given*) yang diturunkan dari kata dasar”beri” (to *give*). Jika kita menghilangkan akhiran (*suffixes*) ”kan” terlebih dahulu sebelum menghilangkan awalan (*prefix*) ”di-” maka pada proses *stemming* ini kita mendapatkan kata dasar yang benar yaitu ”beri” (*to give*), akan tetapi jika algoritma *stemming* mencoba untuk menghilangkan awalan (*prefixes*) terlebih dahulu sebelum akhiran (*suffixes*) maka hasil kata dasar yang dihasilkan dari proses *stemming* dengan menggunakan algoritma tersebut adalah ”ikan” (*fish*) (setelah menghilangkan awalan ”di” dan ”ber”) dimana ”ikan” merupakan kata dasaryang valid yang terdapat dalam

kamus tetapi ”ikan” bukan merupakan kata dasar yang benar untuk kata turunan ”diberikan”.

Penilitian terhadap *stemming* untuk text retrieval, machine translation, document summarization dan text classification sudah pernah dilakukan sebelumnya. Untuk *stemming* yang dilakukan pada Text Retrieval, *stemming* ini meningkatkan kesensitivan retrival dengan meningkatkan kemampuan untuk menemukan document yang relevan, tetapi hal itu terkait dengan pengurangan pada pemilihan dimana pengelompokkan menjadi kata dasar menyebabkan penghilangan makna kata. Pada Text Retrival *stemming* diharapkan dapat meningkatkan *recall*, tetapi memungkinkan untuk menurunkan *precision*.

3.4.1. Stemmer Tala

Stemmer Tala merupakan adopsi dari algoritma stemmer bahasa Inggris terkenal porter stemmer. Stemmer ini menggunakan rule base analisis untuk mencari root sebuah kata. Stemmer ini sama sekali tidak menggunakan kamus sebagai acuan, seperti halnya stemmer vega dan jelita.

Morphologi bahasa Indonesia dapat terdiri dari turunan dan imbuhan kata. Imbuhan yang sederhana digunakan akhiran dimana tidak akan merubah makna dari kata dasar. Turunan-turunan kata yang mungkin terjadi pada kata berbahasa Indonesia menurut Tala adalah :

1. Akhiran –lah, -kah, -pun, -tah adalah akhiran yang berfungsi untuk meyakinkan atau menekankan dan sama sekali tidak mempunyai makna.
   Contoh : dia + kah = diakah ,
   saya + lah = sayalah
2. Akhiran –ku, -mu, -nya adalah akhiran yang melekat pada kata dan membentuk kata ganti punya.
   Contoh : tas + ku = tasku,
   sepeda + mu = sepedamu

3. Turunan kata dalam bahasa Indonesia meliputi awalan + akhiran dan kombinasi dari keduanya. Awalan yang sering muncul adalah : ber-, di-, ke-, meng-, peng-, per-, ter-.

   Contoh : ber + lari = berlari
   di + makan = dimakan
   ke + kasih = kekasih
   meng + ambil = mengambil
   peng + atur = pengatur
   per + lebar = perlebar
   ter + baca = terbaca

4. Turunan akhiran -i , -kan dan -an.

   Contoh : gula + i = gulai
   makan + an = makanan
   beri + kan = berikan

5. Turunan akhiran dan awalan.

   Contoh : per + main + an = permainan
   ke + menang + an = kemenangan
   ber + jatuh + an = berjatuhan
   meng + ambil + i = mengambili

6. Pasangan awalan dan akhiran yang salah / tidak sah (illegal).

Tabel 3. 1 Awalan dan akhiran yang salah (Tala, 2003)

Prefix	Suffix
ber	i
di	an
ke	i|kan
meng	an
peng	i|kan
ter	an

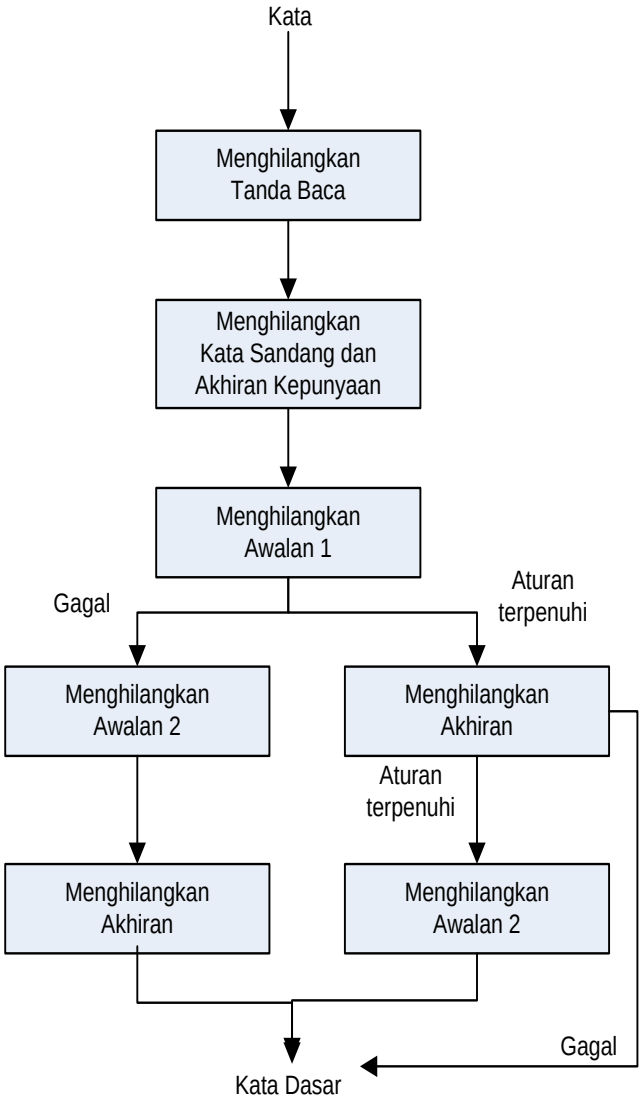
7. Awalan ganda yang mungkin terjadi.

Tabel 3. 2 Awalan ganda yang mungkin terjadi (Tala, 2003)

Prefix 1	Prefix 2
meng	per
di	ber
ter	
ke	

Dalam proses *stemming* bahasa Indonesia ini terdapat beberapa tahap. Sebuah kata akan dites dengan menggunakan *rule* yang dibuat pada setiap tahap. Pada setiap tahap, sebuah kata yang memenuhi kondisi untuk *rule* pada tahap itu maka kata tersebut akan diganti dengan kata baru yang dibentuk dengan *substitution rule* (aturan pengganti).

Arsitektur proses stemming untuk bahasa Indonesia dapat dilihat pada Gambar 3.3. Tahap pertama yang dilakukan adalah menghilangkan partikel kata (tanda baca) kemudian menghilangkan *possesive pronouns* (kata sandang dan akhiran kepunyaan). Baru setelah itu dilakukan proses untuk menghilangkan *prefiks* (awalan) pertama dari kata tersebut. Jika kata tersebut memiliki *prefiks* pertama maka proses selanjutnya yang dipilih adalah proses menghilangkan *sufiks* dan kemudian menghilangkan *prefiks* kedua. Namun jika kata tersebut tidak memiliki *prefiks* pertama maka proses selanjutnya adalah menghilangkan *prefiks* kedua berubah kemudian menghilangkan *sufiks*.



Gambar 3. 3 Proses Stemming Algoritma Tala (Tala, 2003)

1 Proses menghilangkan partikel

Pada proses ini dokumen melalui perlakuan untuk menghilangkan partikel / tanda baca. Selain tanda baca dalam proses ini juga dihilangkan semua angka serta kata-kata yang tidak bermakna (stopword).

2 Proses menghilangkan kata sandang dan kepunyaan

Pada proses ini dokumen melalui perlakuan untuk menghilangkan kata sandang dan kepunyaan. Proses ini dibagi dalam 2 cluster proses yang harus diproses secara urut.

Klaster 1 : Akhiran –lah, -kah, -pun, -tah diperlihatkan pada tabel 3.3, adalah akhiran yang berfungsi untuk meyakinkan atau menekankan dan sama sekali tidak mempunyai makna.

Tabel 3. 3 Tabel akhiran –lah, -kah, -pun

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kah	NULL	2	NULL	bukukah → buku
lah	NULL	2	NULL	adalah → ada
pun	NULL	2	NULL	bukupun → buku

Klaster 2 :Akhiran –ku, -mu, -nya diperlihatkan pada tabel 3.4, adalah akhiran yang melekat pada kata dan membentuk kata ganti punya.

Tabel 3. 4 Tabel akhiran kepunyaan

Suffix	Replacement	Measure Condition	Additional Condition	Examples
ku	NULL	2	NULL	bukuku → buku
mu	NULL	2	NULL	bukumu → buku
nya	NULL	2	NULL	bukunya → buku

3 Menghilangkan awalan 1

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi awalan 1 dalam 1 cluster proses yang harus diproses secara urut. Tabel 3.5 adalah daftar awalan 1 yang masuk ke cluster ke tiga pada proses stemming Tala

Tabel 3. 5 Tabel awalan 1

Prefix	Replacement	Measure Condition	Additional Condition	Examples
meng	NULL	2	NULL	mengukur → ukur
meny	s	2	V...*	menyapu → sapu
men	NULL	2	NULL	menduga → duga menuduh → uduh
mem	p	2	V...	memilah → pilah
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 penuduh → uduh
pem	p	2	V...	pemilah → pilah
pem	NULL	2	NULL	pembaca → baca
di	NULL	2	NULL	diukur → ukur
ter	NULL	2	NULL	tersapu → sapu
ke	NULL	2	NULL	kekasih → kasih

4 Menghilangkan awalan 2.

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi awalan 2 dalam 1 cluster proses yang harus diproses secara urut. Tabel 3.6 adalah daftar awalan 2 yang masuk ke cluster ke empat pada proses stemming Tala.

Tabel 3. 6 Tabel awalan 2

Prefix	Replacement	Measure Condition	Additional Condition	Examples
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

5 Menghilangkan akhiran.

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi akhiran dalam 1 cluster proses yang harus diproses secara urut. Tabel 3.7 adalah daftar akhiran yang masuk ke cluster ke lima pada proses stemming Tala.

Tabel 3. 7 Tabel akhiran

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kan	NULL	2	prefix $\notin$ {ke, peng}	tarikkan → tarik (meng)ambilkan → ambil
an	NULL	2	prefix $\notin$ {di, meng, ter}	makanan → makan (per)janjian → janji
i	NULL	2	V|K...c_1c_1, $c_1 \neq s, c_2 \neq i$ and prefix $\notin$ {ber, ke, peng}	tandai → tanda (men)dapati → dapat pantai → panta

Setelah 5 tahap dilalui maka kata sudah dianggap telah menjadi root atau kata dasar. Menurut Tala kata dasar pada bahasa Indonesia terdiri paling sedikit 2 kata, sehingga sebelum dilakukan penggantian / penghilangan awalan, akhiran ataupun partikel diperhatikan panjang huruf yang tersisa. Jumlah huruf yang akan diproses minimal 2 + (panjang imbuhan yang akan dihilangkan) + 2 (spasi, untuk depan dan belakang kata), Kemudian diperhatikan pula imbuhan yang tidak sah juga diperhatikan, agar mengurangi over stemming. Pasangan imbuhan yang tidak sah dapat dilihat pada tabel 3.1.

3.5. Cosine Simmilarity Distance

Metode cosine distance merupakan metode yang digunakan untuk menghitung similarity (tingkat kesamaan) antar dua buah obyek. Untuk tujuan klastering dokumen fungsi yang baik adalah fungsi Cosine Similaritas. Berikut adalah persamaan dari metode *Cosine Distance* :

Untuk notasi himpunan digunakan rumus :

$$Similarity(X,Y) = \frac{X \bigcap Y}{|X|^{\frac{1}{2}} . |Y|^{\frac{1}{2}}} \quad (3.1)$$

Dimana :

$X \cap Y$ adalah jumlah term yang ada di dokumen X dan yang ada di dokumenY

| X | adalah jumlah term yang ada di dokumen X

| Y | adalah jumlah term yang ada di dokumen Y

Dari notasi himpunan di atas dapat dibuat persamaan matematika sbb :

$$Similarity(x,y) = \frac{\sum_{i=1}^{t} x_i \, y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}} \qquad (3.2)$$

dimana :

x dan y adalah dukumen yang berbeda.

x_i = term i yang ada pada dokumen x

y_i = term i yang ada pada dokumen y

BAB IV
METODOLOGI PENELITIAN

Metode yang akan digunakan dalam penelitian ini terdiri dari langkah-langkah sebagai berikut:

1. Obyek Penelitian

   Obyek penelitian dari penelitian ini adalah dokumen teks berupa halaman web http://www.kompas.com.

2.. Data yang diperlukan

   Merupakan data yang mendukung dalam penelitian ini meliputi data primer dan data sekunder.

   a.. Data primer

      Data yang diperoleh dari arsip berita online Kompas http://www.kompas.com/archive dalam format HTML oleh penulis disimpan dalam format teks.

   b. Data Sekunder

      Data yang diperoleh dengan membaca dan mempelajari referensi mengenai *stemming*, *text mining*, *clustering*, *indexing*, *term weighting*, *simmilarity*, *query expand*.

3. Teknik Pengumpulan Data

   Pengumpulan data mempunyai tujuan mendapatkan materi – materi yang mempunyai keterkaitan dengan topik penelitian. Pengumpulan data dimaksudkan agar mendapatkan bahan-bahan yang relevan, akurat dan reliable. Maka teknik pengumpulan data yang dilakukan dalam penelitian ini adalah sebagai berikut :

   a. Observasi

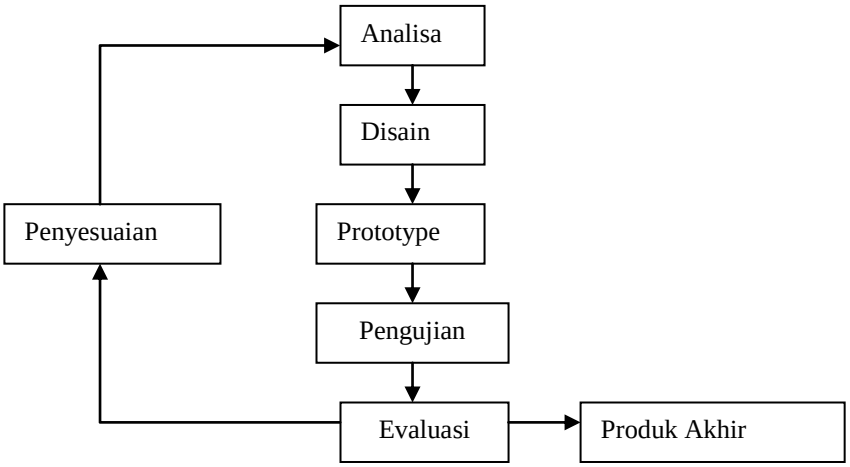
      Dengan melakukan pengamatan dan pencatatan secara sistematis tentang hal-hal yang berhubungan dengan pencarian kemiripan dokumen dengan mengunakan algoritma *Single Pas Clustering.*

b. Studi Pustaka

Teknik pengumpulan data dilakukan dengan studi pustaka yang berhubungan dengan analisis data, pemodelan sistem dan perancangan sistem aplikasi, dengan pengumpulan data dari bahan-bahan referensi, arsip, dan dokumen yang berhubungan dengan permasalahan dalam penelitian ini.

c. Metode Pengembangan

Penelitian ini menggunakan model *prototyping*. Di dalam model ini sistem dirancang dan dibangun secara bertahap dan untuk setiap tahap pengembangan dilakukan percobaan-percobaan untuk melihat apakah sistem sudah bekerja sesuai dengan yang diinginkan. Sistematika model *prototyping* terdapat pada Gambar 4.1. memperlihatkan tahapan pada prototyping.



Gambar 4. 1 Tahapan Prototyping (Pressman, 1997)

Berikut adalah tahapan yang dilakukan pada penelitian ini dengan metode pengembangan prototyping :

1. Analisa

   Pada tahap ini dilakukan analisa tentang masalah penelitian dan menentukan pemecahan masalah yang tepat untuk menyelesaikannya.

2. Disain

   Pada tahap ini dibangun rancangan sistem dengan menggunakan tools pengembangan sistem informasi yaitu DFD, ERD, Class Diagram dan flowchart.

3. Prototype

   Pada tahap ini dibangun aplikasi sistem temu kembali informasi bahasa Indonesia dengan metode *Clustering Single Pass* sesuai dengan disain dan kebutuhan sistem.

4. Pengujian

   Pada tahap ini dilakukan pengujian pada aplikasi yang sudah dibangun, pengujian dilakukan dengan validasi dengan menggunakan input query dalam bentuk teks dan kesesuaian *query* dengan hasil simmilaritas dan hasil klaster yang di dapat dari aplikasi.

5. Evaluasi

   Pada tahap ini dilakukan evaluasi apakah performa aplikasi sudah sesuai dengan yang diharapkan, apabila belum maka dilakukan penyesuaian – penyesuaian sesuai kebutuhan.

6. Penyesuaian

   Tahap ini dilakukan apabila pada evaluasi performa aplikasi kurang memadai dan dibutuhkan perbaikan, tahap ini melakukan penyesuaian dan perbaikan pada aplikasi sesuai dengan kebutuhan.

4.1. Analisis Sistem

Dalam membuat sebuah Sistem Klastering Dokumen dengan *Clustering Single Pass* diperlukan berbagai informasi yang berkenaan dengan rumusan masalah, ide pokok pemecahan masalah dan model Sistem Temu Kembali Informasi.

Oleh karena itu Sistem Klastering Dokumen dengan algoritma *Clustering Single Pass* yang dibuat diharapkan memiliki kemampuan :

1. Sistem ini harus mampu membuat database indexing dalam bentuk tabel indeks.

2. Sistem harus mampu membuat filter terhadap daftar kata umum (stop word) dan tidak menyimpannya dalam database.
3. Sistem harus mampu membuat klaster untuk dokumen yang akan disimpan dalam database.

4.1.1 Deskripsi sistem

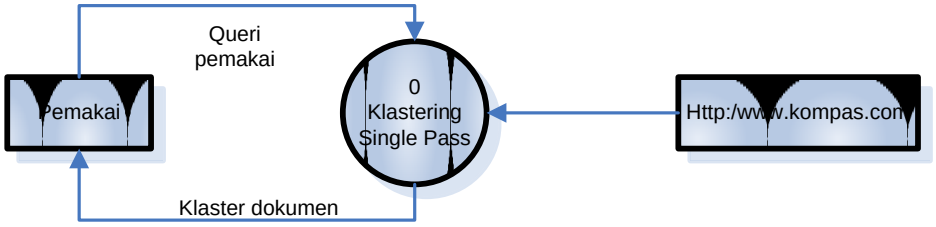
Program aplikasi adalah membangun sistem klastering dokumen dengan menggunakan algoritma *Clustering Single Pass*. Program aplikasi digunakan user untuk membantu mendapatkan dokumen yang berada dalam klaster yang sama.

Dengan menggunakan program aplikasi ini user akan dengan mudah mendapatkan informasi dokumen yang sejenis tanpa harus membaca beberapa dokumen. User tidak perlu menyimpulkan sendiri dari dokumen yang dibacanya untuk mendapatkan informasi yang diinginkan.

4.1.2 Diagram Alir Dokumen

Diagram alir data adalah representasi grafis dari aliran data yang melewati sistem. Dalam penelitian ini diagram alir data dirancang diagram konteks dan diagram level 1.

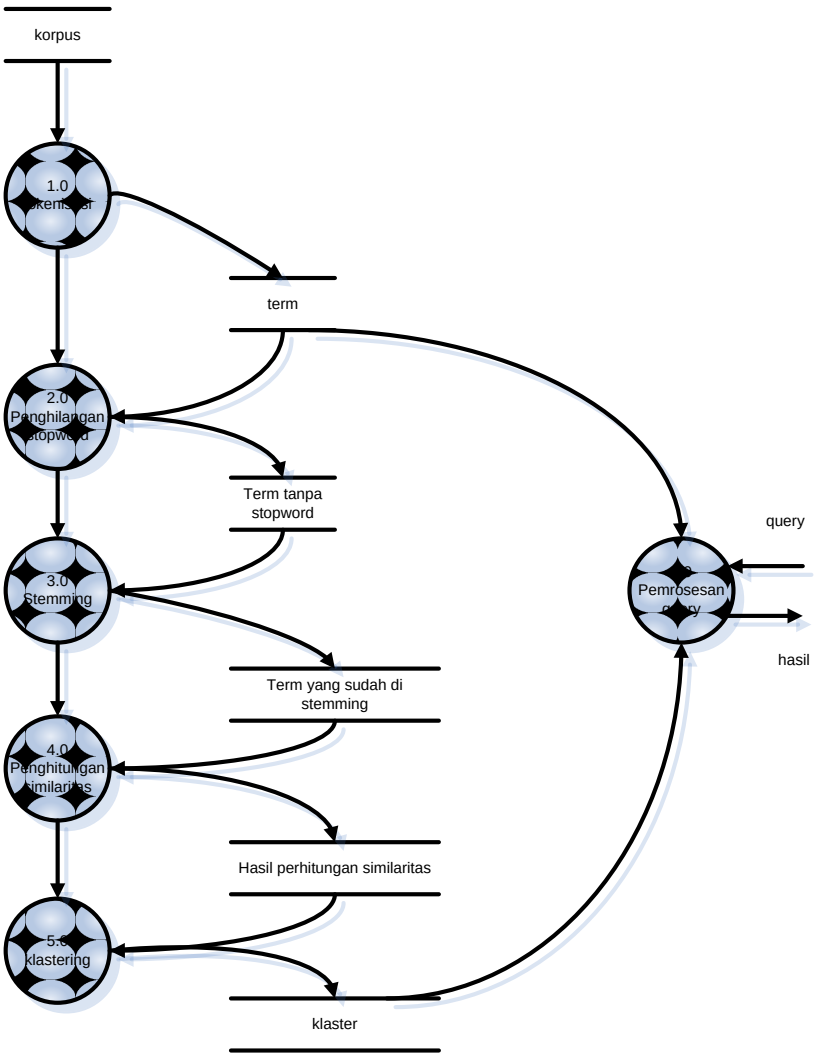
Diagram konteks sistem temu kembali informasi diperlihatkan pada Gambar 4.2. Sistem temu kembali informasi memiliki satu entitas luar yaitu user Entitas luar User memberikan input *query* (teks bahasa Indonesia). Dan sistem akan mengeluarkan output klaster dokumen, kelompok dokumen yang berada dalam klaster yang sama dengan *query* yang diinput user.



Gambar 4. 2 Diagram Konteks Sistem Temu Kembali Informasi

Dalam Gambar 4.3. diperlihatkan Gambar Diagram alir dokumen level 1 Sistem Temu Kembali Informasi terdiri dari proses baca file, proses preprosesing, proses indexing, proses hitung similaritas, proses pembentukan klaster dan pemrosesan *query*. Proses diawali dengan baca file abstrak dalam format teks. Hasil proses baca file akan disimpan dalam penyimpanan data corpus. Dari tabel corpus sistem akan melakukan proses tokenisasi, proses pembuangan stop word dan proses stemming. Kemudian sistem akan melakukan proses indexing. Hasil proses indexing adalah term yang telah diindeks dan disimpan dalam penyimpanan data koleksi. Penyimpanan data koleksi akan digunakan sistem untuk proses hitung similaritas, dari proses akan dihasilkan dokumen similar yang selanjutnya akan digunakan untuk proses pembentukkan klaster dokumen.

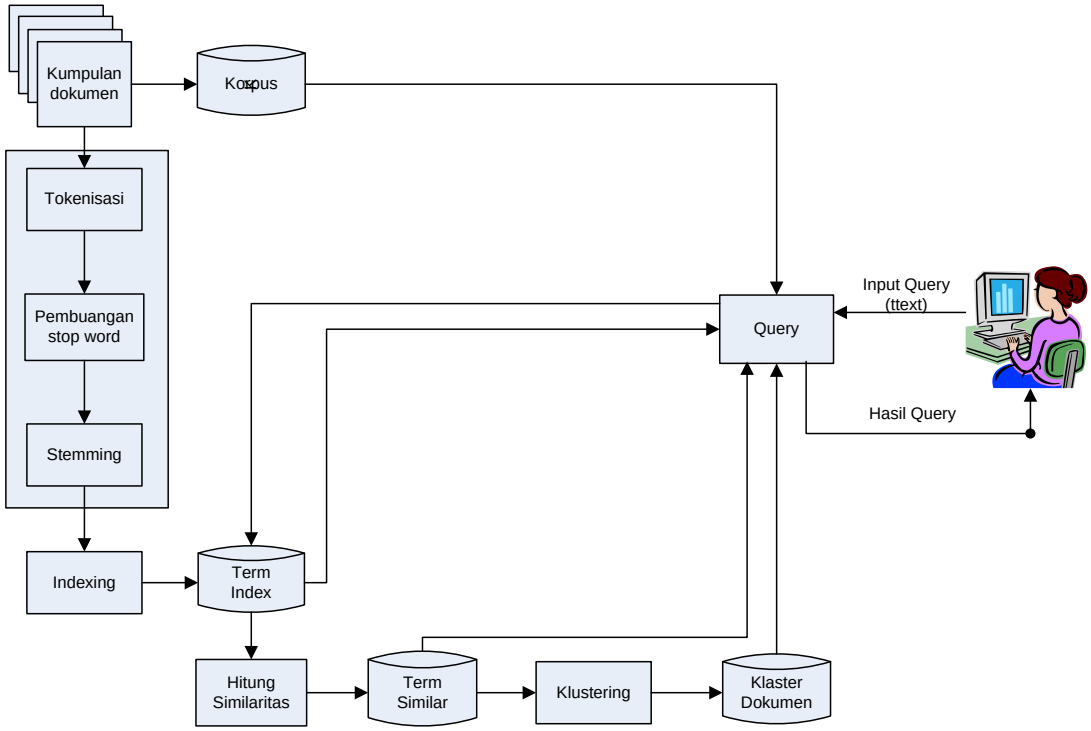
User akan melakukan input *query* yang akan diproses oleh sistem, sistem akan menghasilkan output klaster dokumen dan dokumen similar.



Gambar 4. 3 Diagram Alir Data Level 1 Sistem Temu Kembali Informasi

4.1.3 Arsitektur sistem temu kembali informasi

Sistem Temu Kembali Informasi dengan algoritma *Clustering Single Pass* sebagai suatu sistem memiliki beberapa proses (modul) yang membangun sistem secara keseluruhan. Modul Sistem Temu Kembali Informasi terdiri dari : modul tokenizations (tokenisasi), modul stop word removal (pembuangan stop word), modul stemming (pengubahan kata dasar), modul term indexing (pengindeksan kata), term similarity (kesamaan kata) dan modul clustering (pengelompokan). Secara lengkap arsitektur dari modul Sistem Temu Kembali Informasi dapat dilihat pada Gambar 4.4.



Gambar 4. 4 Arsitektur Sistem Temu Kembali Informasi

Masing-masing modul Sistem Temu Kembali Informasi dapat dijelaskan sebagai berikut :

1. Modul tokenisasi

Sebelum kata dipisahkan dari kalimatnya, terlebih dahulu dibersihkan dari tanda baca, tag html dan angka. Pada penelitian ini untuk membersihkan tanda baca dapat digunakan perintah yang disediakan oleh Java. Pembersihan dilakukan sebelum proses tokenisasi (*tokenizations)* dimaksudkan untuk memperkecil hasil dari tokenisasi. Pada proses tokenisasi akan dibaca dokumen abstrak dalam format teks akan dilakukan proses pemotongan string input berdasarkan tiap kata yang menyusunnya. Pada umumnya setiap kata terindentifikasi atau terpisahkan dengan kata yang lain oleh karakter spasi, sehingga proses tokenisasi mengandalkan karakter spasi pada dokumen untuk melakukan pemisahan kata.

2. Modul pembuangan stop word

Proses pembuangan stop word dimaksudkan untuk mengetahui suatu kata masuk ke dalam stop word atau tidak. Pembuangan stopword adalah proses pembuangan term yang tidak memiliki arti atau tidak relevan. Term yang diperoleh dari tahap tokenisasi dicek dalam suatu daftar *stopword*, apabila sebuah kata masuk di dalam daftar stopword maka kata tersebut tidak akan diproses lebih lanjut. Sebaliknya apabila sebuah kata tidak termasuk di dalam daftar stopword maka kata tersebut akan masuk keproses berikutnya. Daftar stop word tersimpan dalam suatu tabel, dalam penelitian ini menggunakan daftar stop word yang digunakan oleh Tala (2003), yang merupakan stop word Bahasa Indonesia yang berisi kata-kata seperti ; ini, itu, yang, ke, di, dalam, kepada, dan seterusnya sebanyak 780 kata.

3. Modul stemming

Proses *stemming* adalah proses pembentukan kata dasar. Term yang diperoleh dari tahap pembuangan stop word akan dilakukan proses stemming. Algoritma stemming yang digunakan adalah modifikasi Porter stemmer dari (Tala, 2003). *Stemming* digunakan untuk mereduksi bentuk term untuk menghindari ketidak cocokan yang dapat mengurangi *recall*, di mana term-term

yang berbeda namun memiliki makna dasar yang sama direduksi menjadi satu bentuk.

Proses stemming adalah bagian dari proses filtering, tujuan utama dari proses stemming adalah mengembalikan kata dalam bentuk dasarnya. Dengan kata dasar dapat mereduksi bentuk term untuk menghindari ketidak cocokan yang dapat mengurangi recall, di mana term-term yang berbeda namun memiliki makna dasar yang sama direduksi menjadi satu bentuk.

Struktur pembentukan kata dalam Bahasa Indonesia adalah sebagai berikut:

[awalan-1] + [awalan-2] + dasar + [akhiran] + [kepunyaan] + [sandang]

Masing-masing bagian tersebut (yang dalam kotak bisa ada atau tidak), digabungkan dengan kata dasar membentuk kata berimbuhan.

Penggunaan algoritma stemming Tala bertujuan untuk mempercepat waktu implementasi dan diharapkan performa yang stabil walaupun data dokumen bertambah terus. Algoritma Tala menggunakan algoritma rule based stemming seperti halnya dengan algoritma porter pada stemming bahasa Inggris.

Pada stemmer Tala terdapat 5 langkah utama dengan 3 langkah awal dan 2 langkah pilihan, langkah-langkah tersebut sbb:

a) Menghilangkan partikel
b) Menghilangkan kata sandang dan kepunyaan.
c) Menghilangkan awalan 1
d) Jika suatu aturan terpenuhi jalankan sbb :
    - Hilangkan Akhiran
    - Jika suatu aturan terpenuhi, hilangkan awalan 2. Jika tidak proses stemming selesai
e) Jika tidak ada aturan yang terpenuhi jalankan sbb :
    - Hilangkan awalan 2.
    - Hilangkan Akhiran
    - Proses stemming selesai.

Selain itu tala juga membagi imbuhan menjadi 5 cluster yang nantinya digunakan untuk menghilangkan imbuhan pada setiap tahapnya.

4. Modul indexing

Proses *indexing* merupakan tahapan preprocessing yang sangat penting dalam sistem temu kembali informasi sebelum pemrosesan query. Pada proses ini seluruh dokumen dalam koleksi disimpan dalam suatu file dengan format sedemikian sehingga dokumen satu dengan dokumen yang lain dapat dibedakan. Setelah kata telah dikembalikan dalam bentuk asal (kata dasar), kata-kata tersebut disimpan kedalam tabel basis data. Penelitian ini menggunakan metode *Inverted Index,* dengan struktur terdiri dari: kata (*term*) dan kemunculan. Kata-kata tersebut adalah himpunan dari kata-kata yang ada pada dokumen, merupakan ekstraksi dari kumpulan dokumen yang ada. Setiap term akan ditunjukkan informasi mengenai semua posisi kemunculannya secara rinci.

5. Modul hitung similaritas

Relevansi sebuah dokumen ke sebuah *query* didasarkan pada *similarity* (similaritas) diantara vektor dokumen dan vektor *query*. Koordinat dari bobot istilah secara dasarnya diturunkan dari frekuensi kemunculan dari istilah. Pada modul ini akan dihitung presentase kemunculan tiap kata (term) dan presentase kesamaan antar dua term. Metode yang digunakan untuk menghitung adalah metode *cosine simmilarity* dengan menggunakan rumus seperti diuraikan pada persamaan (3.2).

Masing-masing dokumen akan dihitung cacah term yang sama antara dokumen yang satu dengan dokumen yang lain. Hasil dari hitung cacah akan dihasilkan dokumen dengan nilai similaritas dokumen. Nilai similaritas dokumen yang tertinggi dapat dianggap bahwa dokumen tersebut paling simmilar, yaitu memiliki banyak kesamaan.

6. Modul klastering

Pada penelitian ini dokumen akan dibuat klaster dengan menggunakan metode *Clustering Single Pass*. Metode ini berawal dari objek-objek individual. Jadi pada awalnya banyaknya klaster sama dengan banyaknya objek. Pertama-tama objek-objek yang paling mirip dikelompokkan, dan kelompok-kelompok awal ini

digabungkan sesuai dengan kemiripannya (similaritas). Akhirnya, sewaktu kemiripan berkurang, semua subkelompok digabungkan menjadi satu klaster tunggal. Begitu seterusnya dari hasil similaritas yang tertinggi akan dibandingkan dengan dokumen yang satu dengan dokumen yang lain, sehingga didapat similaritas terendah. Hasil similaritas terendah menyatakan bahwa dokumen tersebut merupakan klaster yang berbeda.

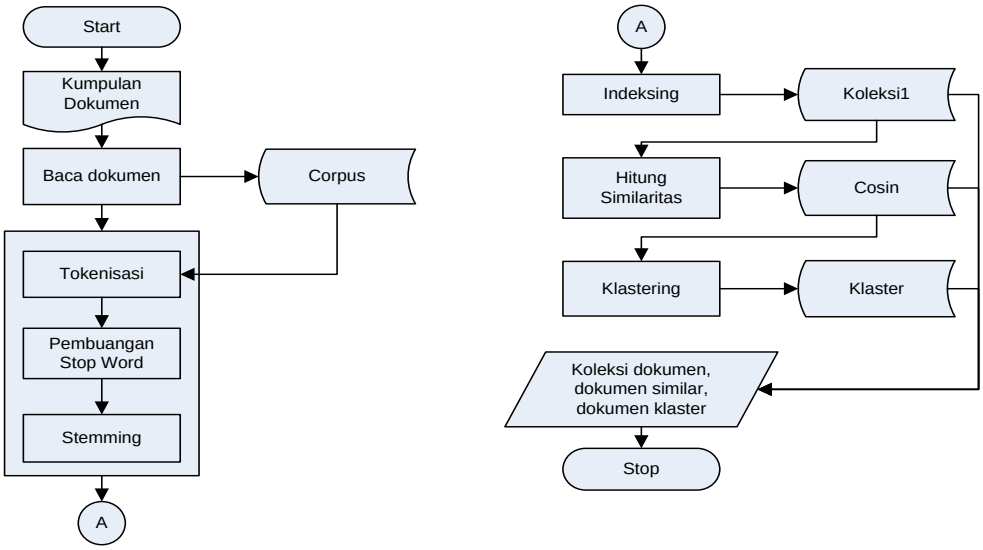
4.2 Flowchart Sistem Temu Kembali Informasi

Penelitian ini akan dibuat diagram alir untuk Sistem Temu Kembali Informasi dengan menggunakan diagram flowchart. Flowchart terdiri dari flowchart sistem dan flowchart program.

Flowchart sistem merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. Dengan kata lain, flowchart ini merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.

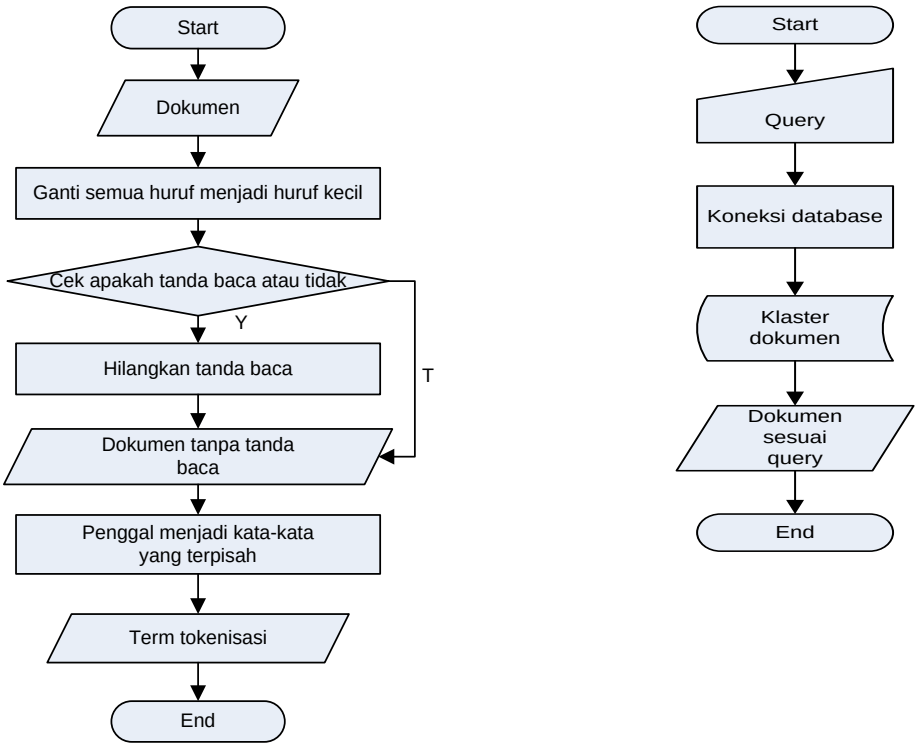
Flowchart sistem untuk Sistem Temu Kembali Informasi pada penelitian ini dibuat menjadi 2 flowchart. Yaitu flowchart sistem untuk pembentukan database klaster dokumen. Kedua adalah flowchart sistem untuk Sistem Temu Kembali Iinformasi.

Gambar 4.5 menjelaskan flowchart sistem untuk pembentukan database klaster dokumen. Proses untuk pembentukan database klaster dokumen terdiri dari: 1) proses preprosesing terdiri dari 3 proses : proses *tokenization* (tokenisasi), proses *stop word removal* (pembuangan stopword) dan proses *stemming*, 2) proses indexing, 3) proses *similarity calculations* (hitung similaritas), 4) proses *clustering* (klastering).



Gambar 4. 5 Flowchart Pembentukan Klaster Dokumen

Flowchart sistem untuk Sistem Temu Kembali Informasi dapat dilihat pada Gambar 4.6. Sistem Temu Kembali Informasi dimulai dari input berupa *query* (kata kunci). *Query* akan dilakukan proses koneksi ke database klaster dokumen. Database klaster dokumen merupakan database hasil dari flowchart sistem pembentukan database klaster dokumen. Hasil dari proses koneksi ke database klaster dokumen adalah dokumen dari database klaster dokumen yang relevan (sesuai) dengan *query* yang dimasukkan.



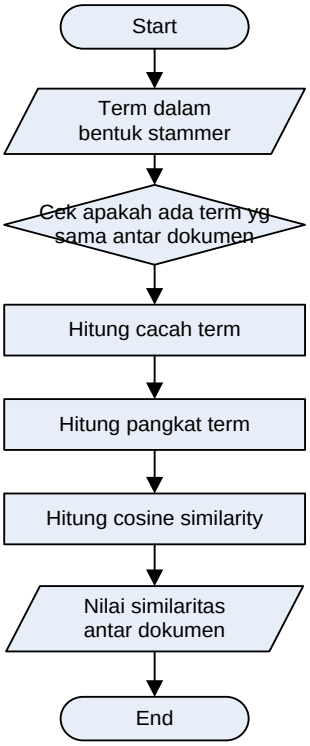
Gambar 4. 6 Flowchart Sistem Temu Kembali Informasi

Flowchart program merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur sesungguhnya dilaksanakan. Flowchart ini menunjukkan setiap langkah program atau prosedur dalam urutan yang tepat saat terjadi.

Penelitian ini dibuat flowchart program untuk proses pembentukan database klaster dokumen yang terdiri dari flowchart tokenisasi, flowchart pembuangan stop word, flowchart stemming, flowchart hitung similaritas dan flowchart klastering.

4.2.1 Flowchart tokenisasi

Seperti yang terlihat pada Gambar 4.7 pada proses preprosesing untuk tokenisasi, semua term dalam dokumen yang dibaca diganti dengan huruf kecil. Setelah itu tiap term akan dicek apakah tanda baca atau tidak. Jika tanda baca maka akan dihapus/dibuang. Proses akan dilanjutkan untuk membuat term menjadi token-token yang terpisah.

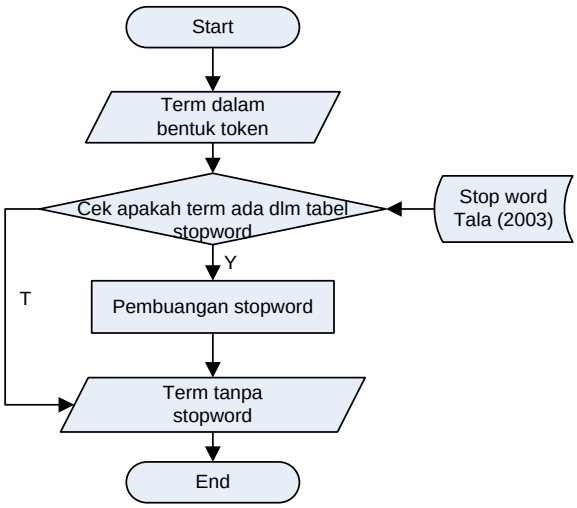


Gambar 4. 7 Flowchart Proses Tokenisasi

4.2.2 Flowchart pembuangan stop word

Proses pembuangan stop word dimaksudkan untuk mengetahui suatu term (kata) masuk ke dalam stop word atau tidak. Apabila sebuah kata masuk di dalam daftar stopword, maka kata tersebut tidak akan diproses lebih lanjut. Sebaliknya apabila sebuah kata tidak termasuk di dalam daftar stopword maka kata tersebut akan masuk keproses berikutnya.

Seperti terlihat pada Gambar 4.8 pembuangan stop word dilakukan dengan mengecek pada tabel stop word. Bila term cocok dengan salah satu isi tabel stop word, maka term tersebut dianggap sebagai stop word akan dibuang dan tidak akan diikutkan pada proses *stemming*. Dari proses pembuangan stop word akan menghasilkan term tanpa stop word.



Gambar 4. 8 Flowchart Proses Pembuangan Stop Word

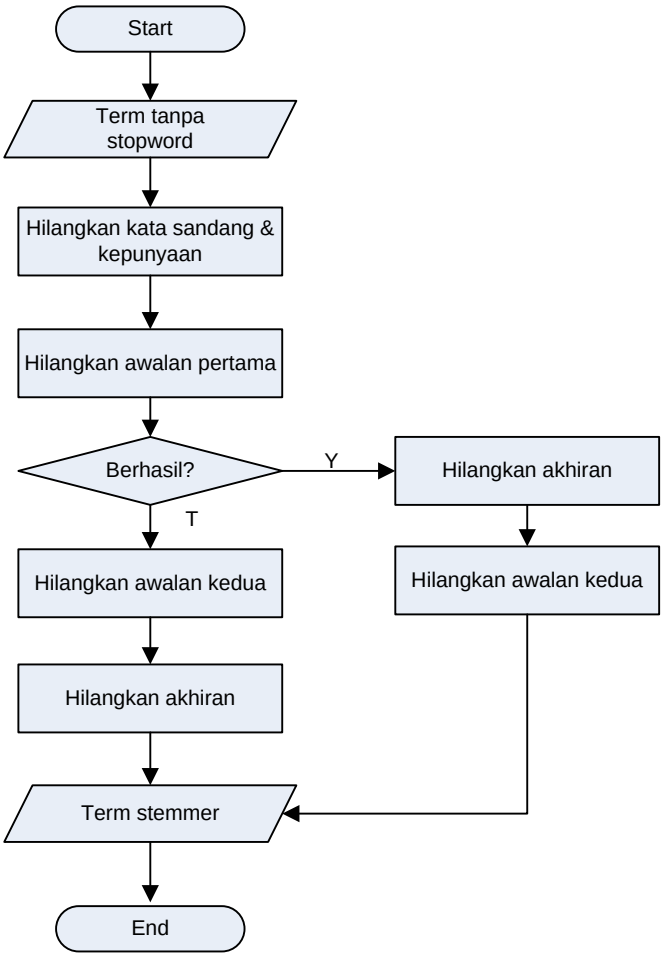
4.2.3 Flowchart stemming

Seperti terlihat pada Gambar 4.9 proses stemming dimulai dari input term tanpa stop word. Term yang dibaca akan dilakukan proses stemmer, pada penelitian ini digunakan Stemmer Tala (2003) yang mengadopsi stemmer Porter.

Tahap pertama proses stemming adalah menghilangkan partikel kata. Proses dilanjutkan dengan menghilangkan kata sandang dan kepunyaan, kemudian dilanjutkan dengan menghilangkan awalan kata pertama. Jika berhasil

maka proses selanjutnya adalah menghilangkan akhiran dan awalan kedua. Jika tidak berhasil maka dilakukan adalah menghilangkan awalan kedua dan akhiran.

Output dari proses stemmer adalah term dalam bentuk kata dasar. Hasil proses akan dilakukan perhitungan cacah term, jika ketemu term yang sama maka cacah term akan bertambah jumlahnya. Kata dasar ini akan diindeks oleh sistem dan akan disimpan dalam tabel koleksi. Tabel koleksi akan menyimpan term-term dari masing-masing dokumen.

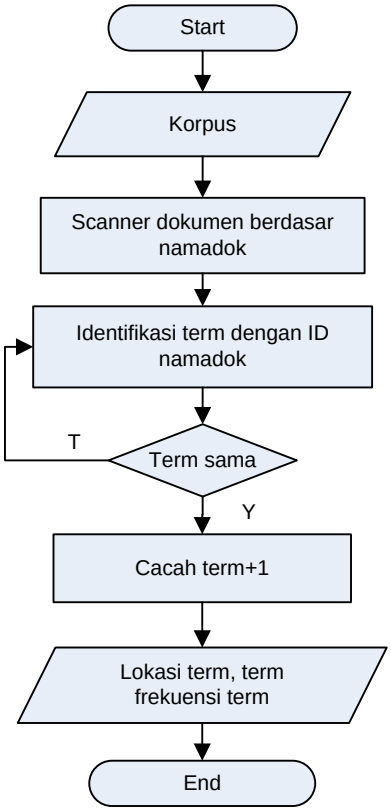


Gambar 4. 9 Flowchart Proses Stemming

4.2.4 Flowchart Indexing

Setelah kata telah dikembalikan dalam bentuk asal (kata dasar), kata-kata tersebut disimpan kedalam tabel basis data.. Pada proses ini seluruh dokumen

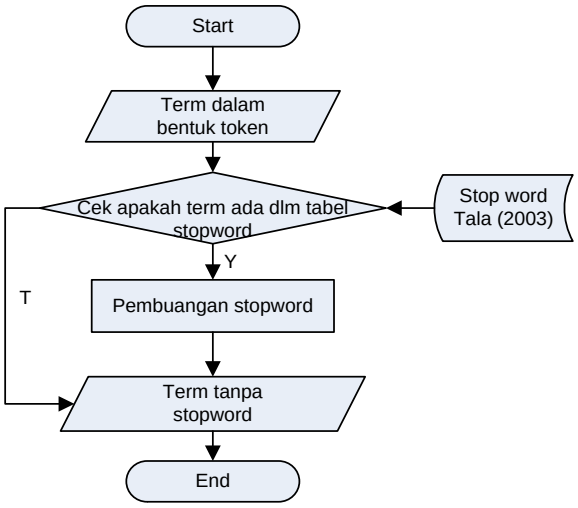
akan disimpan dalam suatu file dengan format sedemikian sehingga dokumen satu dengan dokumen yang lain dapat dibedakan. Proses indek menggunakan metode inverted indexing. Yaitu metode indeks dengan membedakan letak term (kata) tiap-tiap term dalam dokumen. Untuk membedakan letak term digunakan nama dokumen sebagai indeks. Tiap term yang terbaca akan disimpan dengan indeks nama dokumen. Flowchart untuk proses indexing dapat dilihat pada Gambar 4.10



Gambar 4. 10 Flowchart Proses Indexing

4.2.5 Flowchart hitung similaritas

Seperti terlihat pada Gambar 4.11. proses hitung similaritas dokumen beberapa proses telah dilakukan pada proses indexing. Proses indexing menghasilkan dokumen yang telah teridentifikasi lokasi dan jumlah cacah term, dan pangkat frekuensi term. Cacah term dan pangkat term akan digunakan untuk hitung *cosine similarity* dengan menggunakan persamaan (3.2). Output adalah nilai similaritas dokumen yang satu dengan dokumen yang lain.

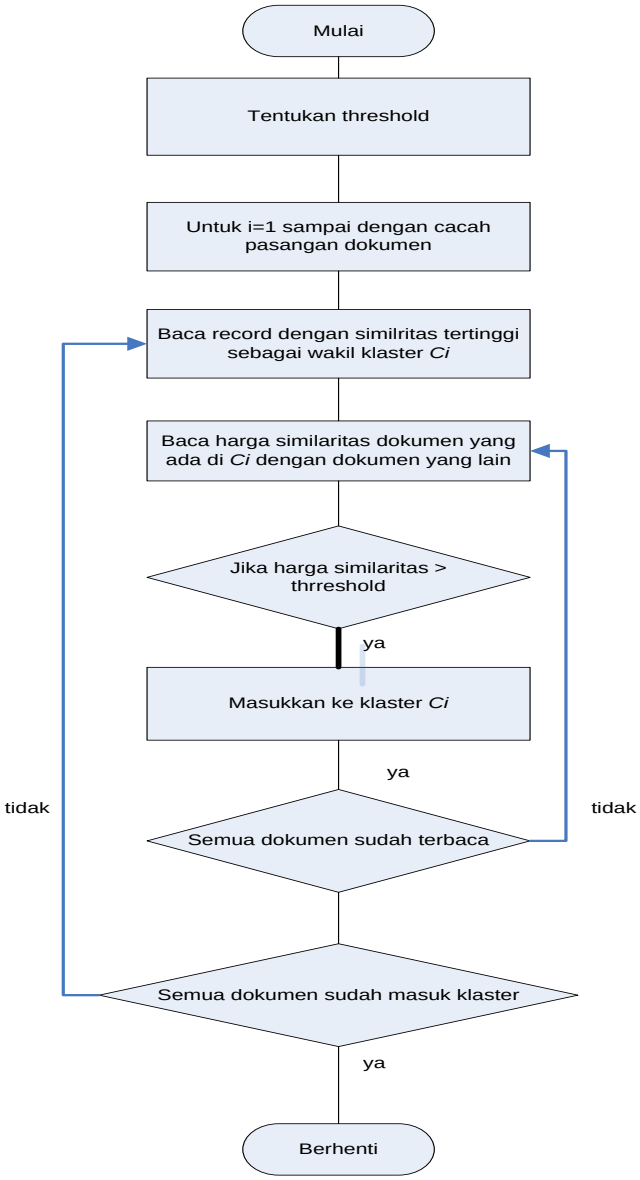


Gambar 4. 11 Flowchart Proses Hitung Similaritas

4.2.6 Flowchart klastering

Proses klastering pada penelitian ini digunakan algoritma Clustering Single Pass. Seperti terlihat pada Gambar 4.12. proses klastering akan dilakukan dari hasil output proses hitung similaritas, yaitu nilai similaritas antar dokumen. Proses pertama adalah mencari similaritas tertinggi (maksimal). Dokumen dengan similaritas tertinggi akan menjadi klaster *C1*. Selanjutnya dicari dokumen yang memiliki similaritas diatas threshold.

Proses akan berulang dengan mencari dokumen yang lain untuk dimasukkan ke klaster yang berbeda.



Gambar 4. 12 Flowchart Proses Klastering

4.3 Perancangan Database dan Class Diagram

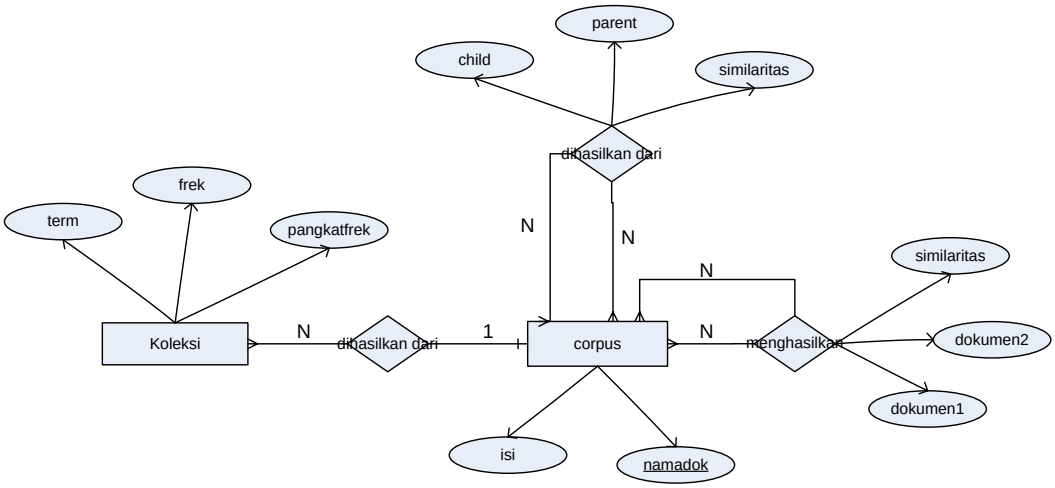
4.3.1 Perancangan database

Untuk menjawab pertanyaan tentang pemrosesan data, metode pemodelan data menggunakan ERD (*Entity Relationshyp Diagram*) atau Diagram Hubungan Entitas yang memungkinkan perekayasaan perangkat lunak untuk mengidentifikasi objek data dan hubungannya dengan menggunakan notasi grafis.

Entity Relationshyp Diagram digunakan untuk memudahkan struktur data dan hubungan antar data, karena hal ini relatif kompleks. Dengan *Entity Relationshyp Diagram* dapat melakukan pengujian model dengan mengabaikan proses yang harus dilakukan.

Dalam rancangan sistem basis data untuk Sistem Temu Kembali Informasi Bahasa Indonesia, digunakan *Entity Relationshyp Diagram* atau Diagram Hubungan Entitas dan desain tabel untuk menggambarkan atribut-atributnya yang ditunjukkan pada Gambar 4.13.

Terlihat pada Gambar 4.13 bahwa entity corpus memiliki hubungan *one to many* (satu ke banyak) dengan entity koleksi, karena satu corpus dapat menghasilkan banyak koleksi dengan atribut namadok sebagai *primary key* untuk relasi. Hal ini juga sama bahwa entity corpus berelasi *recursive,* dengan relasi *many to many* (banyak ke banyak) yang akan menghasilkan nilai similaritas sebagai hasil relasi. Nilai similaritas akan disimpan dalam tabel cosin. Entity corpus juga berelasi *recursive* dengan relasi *many to many* (banyak ke banyak) dengan nilai similaritas akan menghasilkan klaster dokumen. Klaster dokumen akan disimpan dalam tabel klaster.



Gambar 4. 13 ER-D Sistem Temu Kembali Informasi

Nilai similaritas yang dihasilkan dari relasi recursive entity corpus akan menghasilkan 2 (dua) tabel yaitu tabel cosin dan tabel klaster. Tabel cosin terdiri dari *field* dokumen1, dokumen2 dan similaritas.

Tabel klaster menyimpan *field* parent, child dan nilai similaritas. Parent adalah dokumen yang memiliki nilai similaritas lebih tinggi dibandingkan dengan dokumen yang lain. Child adalah dokumen yang memiliki nilai similaritas lebih rendah dibandingkan dengan dokumen yang lain.

Berikut adalah transformasi Entity Relationshyp Diagram ke tabel yang digunakan beserta tipe datanya:

1. Tabel stopword yang dibuat tidak ada dalam rancangan *Entity Relationshyp Diagram.* Tabel stop word berisi daftar stop word yang digunakan untuk pengecekan stopword yang ada di korpus. Tabel stop word digunakan untuk pengecekan term hasil proses tokenisasi. Term sebagai stop word dalam tabel disimpan dengan nama *field* term.

Tabel 4. 1 Tabel stopword.dbo

Nama Field	Type Field	Keterangan
Term	varchar(200)	Term stop word

2. Tabel corpus akan menyimpan dokumen berupa data abstrak skripsi yang diambil dari dokumen dalam format file teks. Dokumen abstrak akan tercatat di *field* isi dan nama dokumen tercatat dengan *field* namadok.

Tabel 4. 2 Tabel corpus.dbo

Nama Field	Type Field	Keterangan
situs	varchar(200)	nama file dokumen
Isi	Text	Isi dari dokumen

3. Tabel koleksi akan menyimpan term dari masing-masing dokumen, frekuensi tiap term tercatat di *field* frek dan pangkat frekuensi term dicatat dalam *field* pangkatfrek.

Tabel 4. 3 Tabel koleksi.dbo

Nama Field	Type Field	Keterangan
Namadok	varchar(50)	Nama file dokumen
Term	varchar(100)	term-term dari dokumen
Frek	Integer	Cacah frekuensi tiap-tiap term
Pangkatfrek	Integer	pangkat tiap-tiap term

4. Tabel cosin berisi daftar term dari setiap dokumen disimpan dalam *filed* term dan similaritas antar dokumen akan disimpan di *field* similaritas.

Tabel 4. 4 Tabel cosin.dbo

Nama Field	Type Field	Keterangan
Dokumen1	varchar(150)	nama dokumen pertama yang akan dihitung similaritas dengan dokumen2
Dokumen2	varchar(150)	nama dokumen kedua yang akan dihitung similaritas dengan dokumen1
Similaritas	Float	Nilai similaritas hasil dari hitung similaritas

5. Tabel klaster berisi daftar klaster dari tiap dokumen, nama dokumen dalam tabel klaster tercatat dengan *filed* child dan parent, nilai similaritas antara dokumen dengan klaster sejenis disimpan dengan nama *field* similaritas.

Tabel 4. 5 Tabel klaster.dbo

Nama Field	Type Field	Keterangan
Situs	Text	Dokumen anak
klaster	Text	No klaster

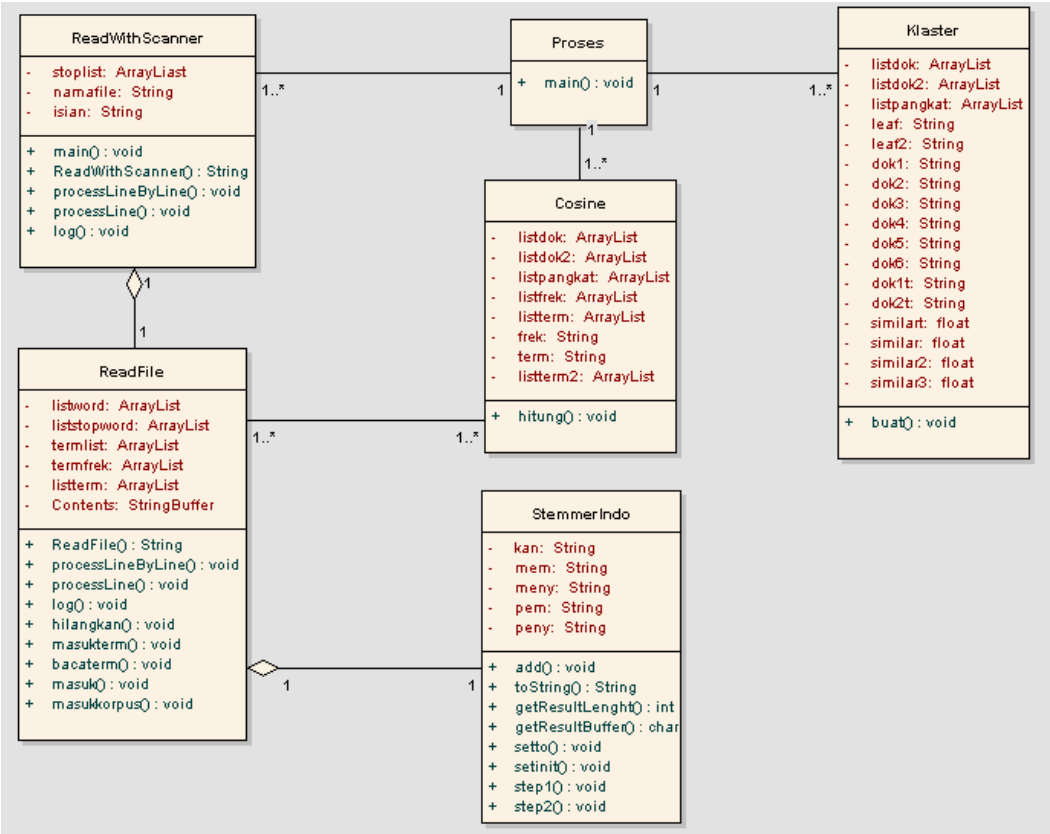
4.3.2 Class diagram

Seperti terlihat pada Gambar 4.14 Class Diagram Sistem Temu Kembali Informasi terdiri dari class ReadWithScanner, class ReadFile, class Cosine, class Klaster dan class proses.

Class ReadWithScanner terstruktur dari class ReadFile dengan asosiasi *one to one*. Class ReadFile terstruktur dari class StemmerIndo dengan asosiasi *one to one*. Class Cosine menggunakan class proses untuk menjalankan proses

hitung cosine. Asosiasi yang terjadi bahwa *one* (satu) class proses berasosiasi *many* (banyak) dengan class Cosine. Class Klaster juga menggunakan class proses untuk pembentukan klaster dokumen. Asosiasi yang terjadi adalah *one* (satu) class proses berasosiasi *many* (banyak) dengan class Klaster, yaitu bahwa dalam proses klaster saatu dokumen bisa memiliki lebih dari satu klaster.

Implementasi untuk tiap-tiap class dan method yang dimiliki class dalam Sistem Temu Kembali Informasi untuk lebih lengkapnya dijelaskan pada BAB V Implementasi untuk masing-masing class dan method.

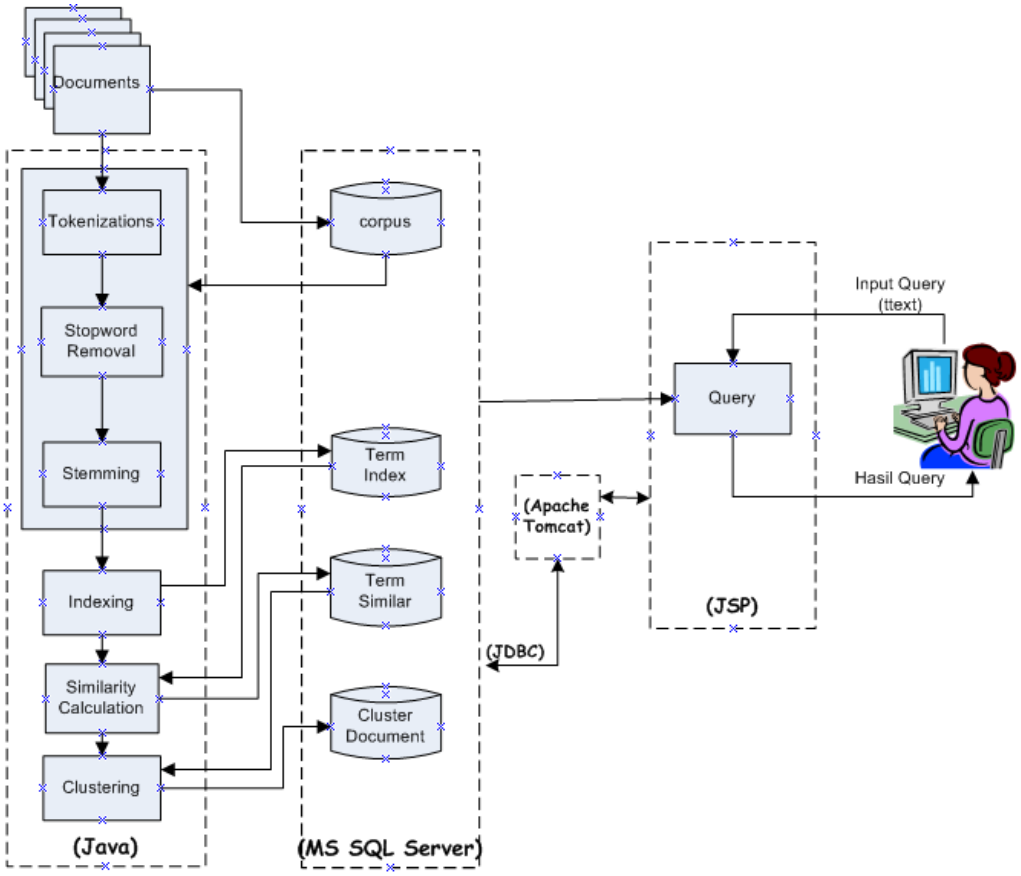


Gambar 4. 14 Class Diagram Sistem Temu Kembali Informasi

4.4 Arsitektur Implementasi Sistem

Perangkat lunak pada penelitian ini, dibuat dengan menggunakan bahasa pemrograman Java pada platform Java Development Kit (JDK) 6.0. Pemrograman Java digunakan untuk implementasi proses-proses dalam Sistem Temu Kembali Informasi. Database yang digunakan untuk menyimpan data

adalah MS SQL Server 2008. User dapat memasukkan *query* melalui interface yang dibangun dengan aplikasi JSP (Java Server Page) dengan Apache Tomcat 6.0 sebagai web server. Untuk mengkoneksikan web server dengan database MS SQL server digunakan aplikasi JDBC. Implementasi untuk perangkat lunak masing-masing proses diperlihatkan pada Gambar 4.15.



Gambar 4. 15 Implementasi Arsitektur Sistem Temu Kembali Informasi

Perangkat lunak untuk membangun Sistem Temu Kembali Informasi ini akan dijalankan di komputer dengan menggunakan sistem operasi Microsoft Windows XP dengan spesifikasi proccessor Intel Core 2 Duo, 2,4 GHz, RAM 1 GB, Hardisk 250 GB.

4.4.1 Implementasi Proses Preprocessing

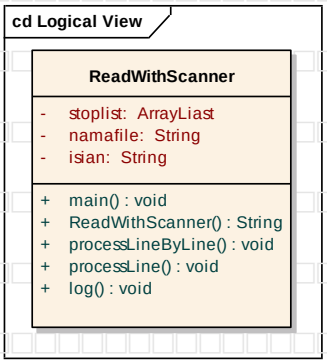
Proses preprosesing ini digunakan sebagai proses untuk persiapan data. Pada proses preprosesing digunakan table-tabel yang digunakan untuk menyimpan output hasil proses dengan menggunakan database SQL Server.

Tabel yang digunakan pada proses preprosesing terdiri dari tabel corpus, table koleksi dan table stopword yang digunakan untuk cek term hasil dari proses baca file.

Implementasi proses preprocessing dilakukan dengan proses tokenization (tokenisasi), proses *stop word removal* (pembuangan stop word) dan proses stemming. Proses preprosesing dimulai dari membaca dokumen abstrak dalam format teks. Selanjutnya akan dilakukan proses *stop word removal*, pada penelitian ini proses stop word removal dilakukan dengan cara *term* akan dicocokan dengan *term* dalam daftar stopword yang disimpan dalam table stopword. Pada implementasi proses preproprosesing digunakan class ReadWithScanner dan class ReadFile.

4.4.2 Implementasi proses baca file

Proses baca file dokumen teks yaitu halaman arsip berita di http://www.kompas.com/archive dilakukan melalui perantara class ReadWithScanner dan class ReadFile. Seperti terlihat pada class ReadWithScanner Gambar 4.16, class ReadWithScanner merupakan class perantara untuk melakukan proses baca file. *Method* ReadWithScanner() digunakan untuk melakukan proses baca file dokuman abstrak dengan cara baris per baris, untuk tiap-tiap file naskah abstrak. File naskah abstrak telah tersimpan dengan nama file yang teridentifikasi sesuai dengan bidang kelompoknya masing-masing.



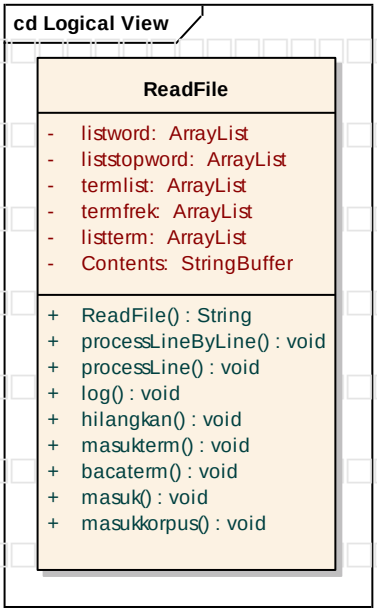
Gambar 4.16 Class ReadWithScanner

Pada Gambar 4.17 diperlihatkan implementasi untuk *method* ReadWithScanner(). *Method* ReadWithScanner akan mengkonstruksi class ReadWithScanner yang akan menjalankan perintah untuk membaca file di alamat http://www.kompas.com/archive adalah file yang menampung koleksi dari korpus. Sedangkan untuk melakukan proses baca isi file abstrak digunakan *method* processLine() seperti yang terlihat pada *method* ReadFile() pada class ReadWithScanner.

```
public  class ReadWithScanner {
static ArrayList stoplist = new ArrayList();

public static void main(String... aArgs) throws
FileNotFoundException {
String namafile="";

ReadWithScanner parser = new
ReadWithScanner("C:\\tomcat\\webapps\\
http://www.kompas.com/archive ");
  parser.processLineByLine();
    {
       ReadFile parser2 = new
ReadFile(stoplist.get(i).toString());
        parser2.processLineByLine();
        }   }
public ReadWithScanner(String aFileName){
    fFile = new File(aFileName);
  }
 public final void processLineByLine() throws
FileNotFoundException {
    Scanner scanner = new Scanner(fFile);
 try {
```

Gambar 4.17 Implementasi *Method* ReadWithScanner

Method ReadFile akan menggunakan perantara class ReadFile seperti terlihat pada Gambar 4.18 Class ReadFile dengan *method* ReadFile() yang akan melakukan baca isi file abstrak yang digunakan untuk membaca isi file dari dokumen abstrak. Proses baca isi file ini dilakukan baris perbaris untuk setiap file yang tersimpan dalam file hasil4.txt.



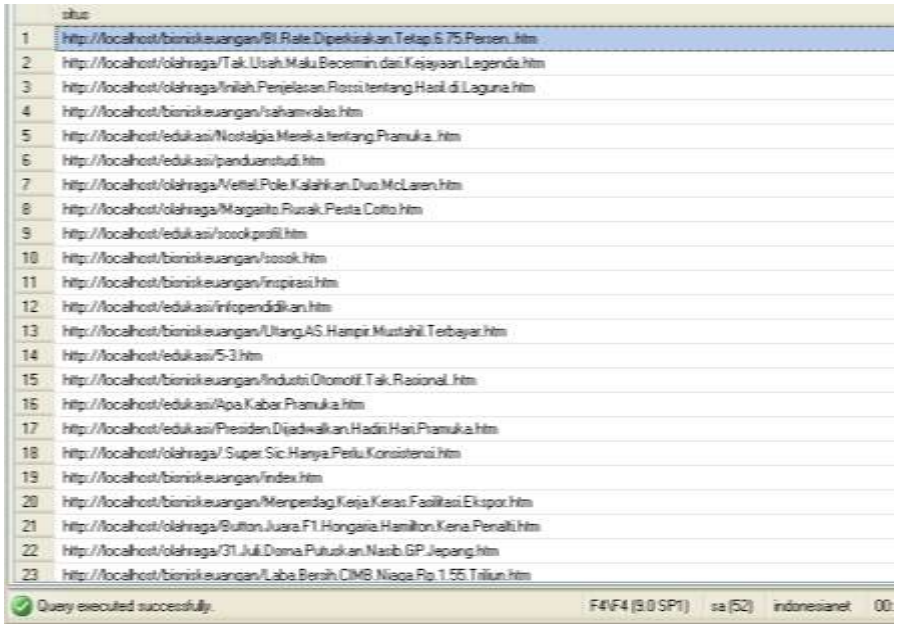
Gambar 4.18 Class ReadFile

Implementasi proses baca isi http://www.kompas.com/archive diperlihatkan pada Gambar 4.19 Pada *method* ReadFile dilakukan baca file abstrak dengan nama file disimpan dengan *attribute* namadok. Untuk proses baca isi file abstrak ini dilakukan pada *method* processLineByLine. Proses baca isi file abstrak dilakukan dengan cara baris per baris

```
public static void main(String... aArgs) throws
FileNotFoundException {
     ReadFile parser = new ReadFile(namafile);
     parser.processLineByLine();
       log("Done.");
  }
public ReadFile(String aFileName){
    fFile = new File(aFileName);
     namadok=aFileName;
  }
public final void processLineByLine() throws
FileNotFoundException {
      StringBuffer termbuf = new StringBuffer();
      Scanner scanner = new Scanner(fFile);
     try {
      while ( scanner.hasNextLine() ){
     processLine( scanner.nextLine() ); }
}
     finally {
      scanner.close();
  }}
```

Gambar 4.19 *Method* ReadFile

Hasil dari *method* ReadFile adalah file dokumen teks yaitu halaman arsip berita di http://www.kompas.com/archive yang akan disimpan dalam tabel corpus. Hasil *method* ReadFile akan disimpan dalam tabel ke SQL Server dengan nama *field* namadok dan isi dokumen dengan *field* isi. Gambar 4.20 memperlihatkan implementasi penyimpanan isi dokumen dari *method* ReadFile ke SQL Server.



	situs
1	http://localhost/bisnisk euangan/BI Rate Dipertahankan Tetap 6.75 Persen.htm
2	http://localhost/olahraga/Tak Usah Malu Becermin dari Kejayaan Legenda.htm
3	http://localhost/olahraga/Inilah Penjelasan Rossi tentang Hasil di Laguna.htm
4	http://localhost/bisniskeuangan/saham+alas.htm
5	http://localhost/edukasi/Nostalgia Mereka tentang Pramuka_.htm
6	http://localhost/edukasi/panduanstudi.htm
7	http://localhost/olahraga/Vettel Pole Kalahkan Duo McLaren.htm
8	http://localhost/olahraga/Margarito Rusak Pesta Cotto.htm
9	http://localhost/edukasi/scookprofil.htm
10	http://localhost/bisniskeuangan/sosok.htm
11	http://localhost/bisniskeuangan/inspirasi.htm
12	http://localhost/edukasi/infopendidikan.htm
13	http://localhost/bisniskeuangan/Utang AS Hampir Mustahil Terbayar.htm
14	http://localhost/edukasi/5-3.htm
15	http://localhost/bisniskeuangan/Industri Otomotif Tak Rasional_.htm
16	http://localhost/edukasi/Apa Kabar Pramuka.htm
17	http://localhost/edukasi/Presiden Dijadwalkan Hadiri Hari Pramuka.htm
18	http://localhost/olahraga/ Super Sic Hanya Perlu Konsistensi.htm
19	http://localhost/bisniskeuangan/index.htm
20	http://localhost/bisniskeuangan/Memperdag Kerja Keras Fasilitasi Ekspor.htm
21	http://localhost/olahraga/Button Juara F1 Hongaria Hamilton Kena Penalti.htm
22	http://localhost/olahraga/31 Juli Doma Putuskan Nasib GP Jepang.htm
23	http://localhost/bisniskeuangan/Laba Bersih CIMB Niaga Rp 1 55 Triliun.htm

Query executed successfully. F4\F4 (9.0 SP1) sa (52) indonesianet 00:

Gambar 4.20 Tabel Corpus

4.4.3 Implementasi proses tokenisasi

Proses tokenisasi (tokenizations) dokumen diimplementasikan dengan menggunakan class ReadFile. Pada class ReadFile melalui *method* processLineByLine() pertamakali proses akan melakukan pengubahan semua huruf menjadi huruf kecil. Proses pengubahan huruf digunakan perintah :

```
termlist.add(s.toString());
    masukterm(s.toString().toLowerCase());
```

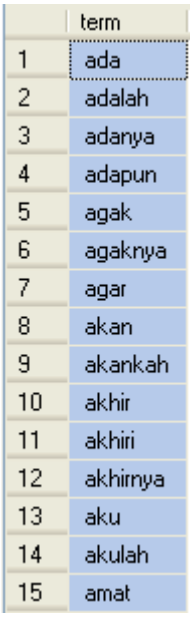
Kemudian proses dilanjutkan dengan *method* processLine() yang akan melakukan proses menghilangkan partikel/tanda baca dan karakter ilegal dalam dokumen. Proses penghilangan partikel/tanda baca ini dilakukan dengan fungsi yang telah disediakan oleh Java yaitu perintah *split(delimiters).* Implementasi selengkapnya untuk proses tokenizations dapat dilihat pada Gambar 4.21.

```
public final void processLineByLine() throws
   FileNotFoundException
{
   StringBuffer termbuf = new StringBuffer();
   Scanner scanner = new Scanner(fFile);

   try
   {
     while ( scanner.hasNextLine() )
     {
       processLine( scanner.nextLine() );
     }
       masukkorpus(namadok,contents.toString());
       for (int y=0;y<listword.size();y++)
     {
      masukterm(listword.get(y).toString().toLowerCase(),list
      word.get(y).toString().toLowerCase());
     }
           hilangkan();
           bacaterm();
           deleteterm();
   }
     finally
     {
       scanner.close();
     }
 }

public final void processLine(String aLine)
{
           Connection con=null;
           String isian=null;
           String delimiters = "[ ]";
           String[] tokens = { };
           isian=aLine.toString();
           isian=isian.replaceAll("\\W"," ");
           contents.append(isian);
     tokens = isian.split(delimiters);

     for (int i=0;i<tokens.length;i++)
     {
        if (tokens[i].toString().length()>=5)
        {
         listword.add(tokens[i]);
        }
     }
 }
```

Gambar 4.21 Proses Tokenisasi

4.4.4 Implementasi proses pemmbuangan stop word

Implementasi proses stop word removal (pembuangan stop word) dilakukan dengan mencocokan term hasil *token* dengan tabel stop word yang ada di database. Term yang digunakan untuk stop word disimpan dengan nama tabel stopword.dbo dengan menggunakan stop word Tala (2003). Gambar 4.22. memperlihatkan beberapa contoh stop word yang digunakan yang tersimpan dalam tabel stopword.dbo.



	term
1	ada
2	adalah
3	adanya
4	adapun
5	agak
6	agaknya
7	agar
8	akan
9	akankah
10	akhir
11	akhiri
12	akhirnya
13	aku
14	akulah
15	amat

Gambar 4.22 Tabel Stop Word

Pada *method* hilangkan() implementasi proses meghilangkan stop word mnggunakan perintah :

```
String sqlcommand2="delete FROM [korpus].[dbo].[term] where (term in (select term from stopword)) or (term like ' ')";
```

adalah *query* dalam SQL Server yang digunakan untuk mencocokan term dari hasil tokenization dengan daftar stop word yang disimpan dalam tabel SQL Server. *Term* hasil proses tokenizations jika sesuai/cocok dengan tabel stopword maka *term* tersebut akan dibuang dan tidak akan diikutkan pada proses

stemming. Implementasi selengkapnya untuk proses pengecekan stop word removal dapat dilhat pada Gambar 4.23.

```
public static void hilangkan()
{
     Connection con=null;
   try
   {
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     con = DriverManager.getConnection("jdbc:odbc:penelitian",
     "sa", "hernyfeb");

     Statement stmt = con.createStatement();
     Statement stmt2 = con.createStatement();

     String sqlcommand2="delete  FROM [korpus].[dbo].[term3]
     where (term  in (select term from stopword2)) or (term
     like '')";

     System.out.println(sqlcommand2);
     stmt2.executeUpdate(sqlcommand2);
     stmt2.close();
   }
}
```

Gambar 4.23 Proses Pembuangan Stop Word

Term hasil seleksi dari proses stop word removal akan disimpan dalam tabel koleksi.dbo. *Method* masukterm() digunakan untuk proses menyimpan *term* yang sudah dihilangkan stop word dengan menggunakan perintah *query* :

```
String   sqlcommand2="INSERT   INTO   [korpus].[dbo].[term]
([term]) "+ "VALUES  ('"+term+"')";
```

Implementasi selengkapnya untuk proses menyimpan term hasil proses penghilangan stop word dapat dilihat pada Gambar 4.24 :

```
public static void masukterm(String term) {
Connection con=null;
try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:penelitian",
"sa", "hernyfeb");

Statement stmt = con.createStatement();
Statement stmt2 = con.createStatement();

String sqlcommand2="INSERT INTO [korpus].[dbo].[term]
([term]) "+ "VALUES  ('"+term+"')";

System.out.println(sqlcommand2);
stmt.executeUpdate(sqlcommand2);
stmt.close();
 }
    catch( Exception e ) {
return;
}
  finally {
     if( con != null ) {
try { con.close( ); }
catch( Exception e ) { e.printStackTrace( ); }
}}
```

Gambar 4.24 Proses Simpan Term

Term yang telah disimpan dalam tabel koleksi akan dihitung cacah masing-masing term dan dihitung pangkat term dari masing-masing dokumen. Proses hitung cacah *term* dan pangkat *term* dilakukan pada *method* bacaterm() pada class ReadFile(). Proses dilakukan dengan membaca *term* berdasarkan nama file dokumen dan *term* yang tersimpan pada tabel koleksi. Perintah yang digunakan untuk membaca cacah *term* dengan menggunakan *query* :

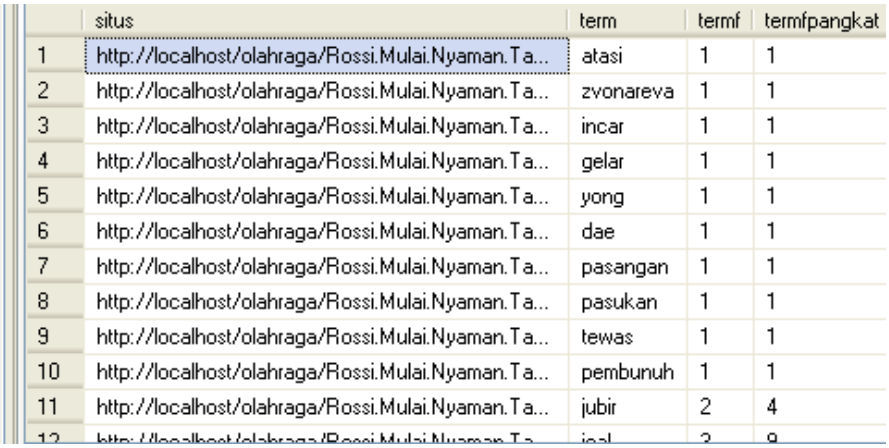
```
String sqlcommand2="select term,COUNT(*) as jumlah    FROM
[korpus].[dbo].[term]  where  term  not  like  ''  group  by
term";
```

Implementasi hitung cacah *term* dan pangkat *term* hasil dari proses baca file selengkapnya dapat dilihat pada Gambar 4.25.

```
 public  void bacaterm() {
   Connection con=null;

 try {
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   con =
DriverManager.getConnection("jdbc:odbc:penelitian", "sa",
"hernyfeb");

Statement stmt = con.createStatement();

String sqlcommand2="select term,COUNT(*) as jumlah   FROM
[korpus].[dbo].[term] where term not like '' group by
term";

ResultSet result2 =  stmt.executeQuery(sqlcommand2);
     while(result2.next()) {

String termin =result2.getString("term").toString();
     int cacah
=Integer.parseInt(result2.getString("jumlah").toString())
;
     int pangkat = cacah * cacah;
masuk(namadok,termin,cacah,pangkat);
stmt.close();
}
```

Gambar 4.25 Proses Hitung Cacah *Term*

Gambar 4.26. memperlihatkan tabel koleksi.dbo yaitu table untuk menyimpan hasil proses preprosesing. Term yang telah menjadi bentuk kata dasar akan disimpan dalam tabel dan diidentifikasi dengan menggunakan namadok sebagai lokasi indeks. Hasil proses hitung cacah *term* dan pangkat *term* juga akan disimpan dalam tabel koleksi. Nama dokumen akan disimpan dengan *field* namadok, term akan disimpan dengan nama *field* term, cacah term disimpan dengan nama *field* frek dan pangkat term akan disimpan dengan nama *field* pangkatfrek.



	situs	term	termf	termfpangkat
1	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	atasi	1	1
2	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	zvonareva	1	1
3	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	incar	1	1
4	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	gelar	1	1
5	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	yong	1	1
6	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	dae	1	1
7	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	pasangan	1	1
8	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	pasukan	1	1
9	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	tewas	1	1
10	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	pembunuh	1	1
11	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	jubir	2	4
12	http://localhost/olahraga/Rossi.Mulai.Nyaman.Ta...	jual	2	8

Gambar 4.26 Tabel Koleksi

4.5 Implementasi proses stemming (Tala)

4.5.1 Proses menghilangkan partikel

Pada proses ini dokumen dibersihkan dari partikel/tanda baca. Selain tanda baca dalam proses ini juga dihilangkan semua angka serta *term* (kata) yang tidak bermakna (stop word). Stop word yang diketahui disimpan dalam tabel basis data stop word kemudian untuk semua *term* yang ada dalam tabel tersebut akan dihilangkan. Isi tabel basis data stopword diambil dari daftar stopword Tala (Tala, 2003).

Masukan untuk proses stemming adalah *term* hasil dari tokenization. Tanda partikel/baca sudah dihilangkan sebelum dilakukan tokenization. Proses penghilangan partikel/tanda baca diimplementasikan pada *method* processLine() pada implementasi proses tokenization seperti terlihat pada Gambar 4.21. Data stop word tersimpan dalam tabel basis data, proses menghilangkan stopword akan dilakukan dengan perintah *query* yang diimplementasikan pada *method* hilangkan() pada implementasi proses pembuangan stop word seperti terlihat pada Gambar 4.23.

4.5.2 Proses menghilangkan kata sandang dan kepunyaan

Pada proses ini dokumen melalui perlakuan untuk menghilangkan kata sandang dan kepunyaan seperti ”lah”, ”pun”, ”ku”, ”mu”, ”nya” dan *sufiks* (akhiran) ”kan” akan diimplementasikan pada *method* step1() dan *method* setto(). Pada *method* step1() jika ditemukan kata seperti ”lah”, ”pun”, ”ku”, ”mu”, ”nya”, ”kan” akan memanggil *method* setto(). *Method* setto() akan melakukan proses jika term memiliki akhiran ”kan” akan dicek apakah *term* tersebut lebih dari 5 karakter, jika lebih maka akan dilakukan penghilangan akhiran. Gambar 4.27 adalah implementasi *method* step1() dan *method* setto() yang digunakan untuk menghilangkan kata sandang dan kepunyaan.

```
 private final void setto(String s)
   {  String kan="kan";
   int l = s.length();
   int o = j+1;
     if (kan.equals(s))
      {
     if (k > 5)
   System.out.println("uji");
    }
      else
    {
      for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);
        k = j+l;
      } }

private final void step1()
   {    if (ends("kah")) setto(""); else
         if (ends("lah")) setto(""); else
         if (ends("pun")) setto(""); else
         if (ends("ku")) setto(""); else
         if (ends("mu")) setto(""); else
         if (ends("nya")) setto(""); else
         if (ends("kan")) setto("");
    }
```

Gambar 4.27 Implementasi Menghilangkan Kata Sandang dan Kepunyaan

4.5.3 Menghilangkan awalan

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi awalan menjadi 2 (dua) tahap yang harus diproses secara urut. Pada penelitian ini proses menghilangkan awalan dilakukan dalam satu proses dengan menggunakan *method* step2() dan *method* setinit() yang ada dalam class ReadFile. Gambar 4.28 adalah implementasi yang digunakan untuk menghilangkan awalan.

```
private final void setinit(String s)
   {  String meny = "meny";
      String mem = "mem";
      String peny = "peny";
      String pem ="pem";
 int l = s.length();
 char[] temp= new char[k];

    if (meny.equals(s)){
       b[0]='s';t=1;   }
    if (peny.equals(s)){
       b[0]='s';t=1;   }
    if (mem.equals(s)) {
       b[0]='p';t=1;   }
    if (pem.equals(s))  {
       b[0]='p';t=1;  }}

private final void step2()
{
   if (starts("meng"))     setinit("meng"); else
   if (starts("meny"))     setinit("meny"); else
   if (starts("peny"))     setinit("meny"); else
   if (starts("men"))     setinit("men"); else
   if (starts("mem"))     setinit("mem"); else
   if (starts("me"))     setinit("me"); else
   if (starts("peng"))     setinit("peng"); else
   if (starts("pen"))     setinit("pen"); else
   if (starts("pem"))     setinit("pem"); else
   if (starts("di"))     setinit("di"); else
   if (starts("ter"))     setinit("ter"); else
   if (starts("ke"))     setinit("ke"); else
   if (starts("ber"))     setinit("ber"); else
   if (starts("per"))     setinit("per"); else
   if (starts("pe"))     setinit("pe");
}
```

Gambar 4.28 Implementasi Menghilangkan Awalan

Pada Gambar 4.28 *method* step2() diimplementasikan jika dalam term yang dibaca ditemukan awalan seperti "meny", "peny" akan dipanggil ke method() setto() bahwa awalan tersebut akan diganti dengan string ”s”. Jika dalam term yang dibaca ditemukan awalan sepertin ”mem”, ”pem” akan diganti dengan string ”p”

4.6 Implementasi proses indexing

. Penelitian ini menggunakan metode *Inverted Index*, dengan struktur terdiri dari: kata (*term*) dan kemunculan (*accurences*). Kata-kata tersebut adalah himpunan dari kata-kata yang ada pada dokumen, merupakan ekstraksi dari kumpulan dokumen yang ada. Setiap term akan ditunjukkan informasi mengenai semua posisi kemunculannya secara rinci.

Dokumen terlebih dahulu akan dilakukan proses scanner untuk mengetahui letak dokumen. Proses scanner akan melakukan baca file dokumen baris perbaris untuk identifikasi tiap-tiap dokumen. Hasil proses scanner dokumen disimpan dalam tabel corpus seperti terlihat pada Gambar 4.20

Setelah dilakukan proses scanner file dokumen, akan dilakukan proses ReadFile yaitu proses membaca term baris per baris dari tiap-tiap dokumen. Proses ReadFile meliputi proses tokenisasi, pembuangan stop word dan stemming. Proses ReadFile dilakukan baris per baris untuk tiap-tiap dokumen yang teridentifikasi. Setelah sebuah kata melalui proses stemming maka kata tersebut akan disimpan kedalam tabel koleksi, yang berfungsi mencatat tiap-tiap term dari dokumen yang dibaca. Proses akan mencatat posisi kemunculan term dan frekuensi kemunculan term. Apabila terdapat term yang sama maka tambahkan kolom frek dengan jumlah 1, sedangkan apabila belum ada maka sisipkan record term baru pada tabel koleksi dengan kolom jumlah=1.

Untuk kebutuhan proses hitung similaritas pada proses menyimpan term ke tabel dilkakukan hitung pangakat frekuensi term. Implementasi proses indexing untuk identifikasi term dari tiap-tiap dokumen menggunakan proses scanner file dan read file yang disediakan Java. Proses indeks menggunakan *method-method* yang ada di dalam *class* ReadWithScanner dan *class* Read File .Frekuensi dari

tiap-tiap term yang ada dalam sebuah dokumen dihasilkan dari *method* bacaterm() seperti terlihat pada Gambar 4.26

4.7 Implementasi proses hitung simmilaritas

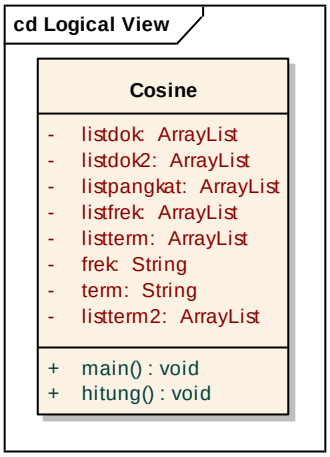
Gambar 4.29 adalah class cosine yang digunakan sebagai class perantara untuk proses hitung similaritas *Method* hitung() pertama kali akan melakukan proses koneksi ke database SQL untuk membaca data *term* yang tersimpan pada tabel koleksi. Proses akan melakukan hitung similaritas antara dokumen yang satu dengan dokumen yang lain samapi semua dokumen dihitung similaritasnya. Proses akan dimulai dari membaca dokumen yang akan dicari similaritasnya, dengan diasumsilkan dokumen pertama adalah dokx dan dokumen kedua adalah doky. Hitung similaritas akan diproses dengan membaca term yang telah tersimpan dalam table corpus hasil dari proses preprosesing. *Query* yang digunakan untuk mengambil data pada table adalah :

```
String SQLCommand = ("SELECT term,frek FROM
[korpus].[dbo].[koleksi3] where namadok like '"+dokx+"%'
order by namadok");
```

Method hitung() akan melakukan proses hitung cacah pangkat term. Cacah pangkat term ini akan digunakan untuk proses hitung similaritas dengan menggunakan rumus *cosine similarity*. Cacah term dan pangkat term diambil dari class ReadFile dengan *attribute* term dan frek yang dipanggil dari table koleksi. Proses hitung cacah pangkat term digunakan *query* sebagai berikut :

```
String SQLCommand2 = ("SELECT namadok ,sum(pangkatfrek) as
pangkat FROM [korpus].[dbo].[koleksi3] group by namadok
order by namadok");
```

Dengan menggunakan persamaan (3.2) pada *method* hitung() pada class Cosine digunakan sebagai perantara untuk menghitung nilai similaritas antar dokumen dari semua dokumen archive Kompas yang ada di dalam korpus. Class Cosine menggunakan class proses untuk melakakukan proses hitung. Dengan perantara class Proses, *method* hitung() pada class Cosine melakukan proses hitung cosine similaritas.



Gambar 4.29 Class Cosine

Implementasi hitung cosine dimulai dengan koneksi ke database SQL Server untuk memaba dokumen koleksi yang berisi cacah term dan pangkat term dokumen. Hitung cosine dilakukan antara dokumen satu dengan dokumen lainnya. Dalam implementasi dokumen satu diberikan atribute dokx sedangkan dokumen lainnya diberikan atribute doky. Kemudian akan dihitung similaritas antar dokumen dengan menggunakan persamaan 3.2.

Hasil dari hitung similaritas akan disimpan dalam tabel cosin. Dokumen yang dihitung nilai similaritasnya disimpan dengan *field* dokumen1 dan dokumen2. Nilai similaritas dari dokumen disimpan dengan *field* similaritas. Implementasi SQL untuk menyimpan hasil hitung similaritas adalah :

```
String SQLCommand4 ="INSERT INTO [korpus].[dbo].[cosin3]
([dokumen1] ,[dokumen2] ,[similaritas]) VALUES
('"+dokx+"','"+doky+"',"+hasil+")";
```

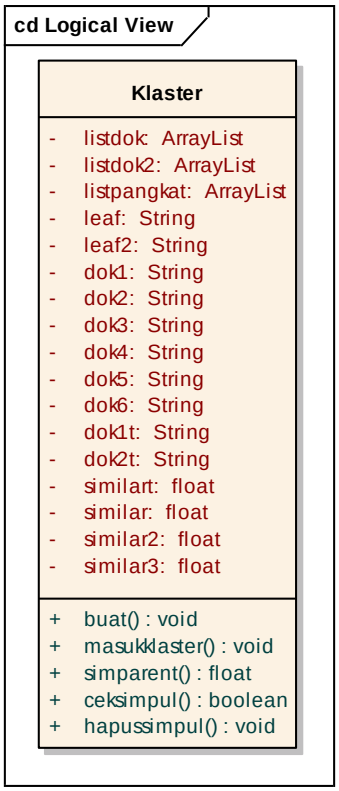
Tabel cosin untuk menyimpan nilai similaritas antara dokumen dapat dilihat pada Gambar 4.30 Tabel cosin menyimpan hasil implementasi dari *method* hitung() yang ada di class Cosine dengan *atribute* dokumen1, dokumen2 dan similaritas.

	situs	situs2	cosin
1	http://localhost/olahraga/Button.Juara.F1.Hongaria....	http://localhost/edukasi/5-3.htm	0.007684919
2	http://localhost/bisniskeuangan/Dulu.Kurir.Kini.Buka....	http://localhost/bisniskeuangan/perencanaankeuan...	0.029304033
3	http://localhost/bisniskeuangan/ekonomi.htm	http://localhost/olahraga/Empat.Pemain.Tunggal.Ber...	0.0114613185
4	http://localhost/bisniskeuangan/perencanaankeuan...	http://localhost/bisniskeuangan/sahamvalas.htm	0.074433655
5	http://localhost/bisniskeuangan/inspirasi.htm	http://localhost/olahraga/Margarito.Rusak.Pesta.Cott...	0.0018050541
6	http://localhost/bisniskeuangan/inspirasi.htm	http://localhost/olahraga/Remehkan.Cotto.Mayorga.I...	0.03068592
7	http://localhost/bisniskeuangan/inspirasi.htm	http://localhost/olahraga/Rossi.Bandingkan.Ducati....	0.036101084
8	http://localhost/bisniskeuangan/Sarinah.Ajak.Korea....	http://localhost/bisniskeuangan/fiskalmoneter.htm	0.4047619
9	http://localhost/bisniskeuangan/Sarinah.Ajak.Korea....	http://localhost/bisniskeuangan/perencanaankeuan...	0.04464286
10	http://localhost/bisniskeuangan/Sarinah.Ajak.Korea....	http://localhost/edukasi/3-3.htm	0.8214286
11	http://localhost/bisniskeuangan/Sarinah.Ajak.Korea...	http://localhost/edukasi/news.htm	0.89285713

Gambar 4.30 Tabel Cosin

4.8 Implementasi proses klastering

Klaster dokumen akan diproses dari nilai similaritas yang dihasilkan oleh *method* Cosine(). Seperti pada Gambar 4.31, class Klaster akan melakukan proses klaster dengan *method* buat().

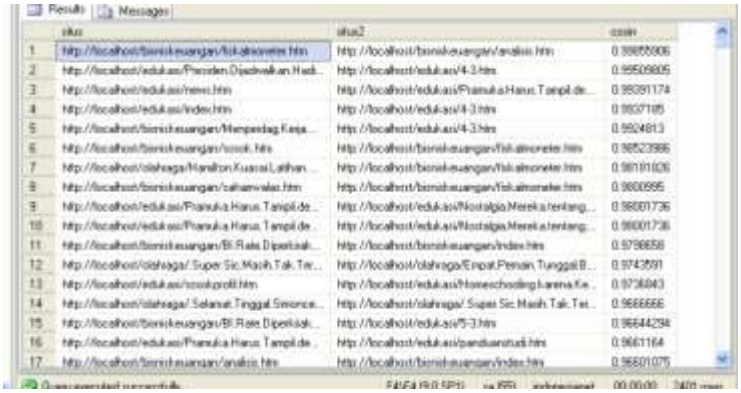


Gambar 4.31 Class Klaster

Nilai similaritas hasil dari proses hitung similaritas akan digunakan untuk melakukan proses klastering. Proses klastering dilakukan dengan mencari nilai similaritas yang maksimum dari table cosin. Implementasi dalam SQL adalah sbb:

```
SELECT [situs]
      ,[situs2]
      ,[cosin]
    FROM [indonesianet].[dbo].[hargatermcosin] order by cosin
desc
```

Hasil dari SQL adalah nilai similaritas yang telah dirangking dari nilai similaritas maksimum. Nilai similaritas maksimum dapat dilihat pada Gambar 4.32, menunjukkan bahwa dokumen http://localhost/bisniskeuangan/fiskalmoneter.htm dengan dokumen http://localhost/bisniskeuangan/analisis.htm memiliki nilai similaritas tertinggi yaitu 0.99855906. Maka dokumen http://localhost/bisniskeuangan/fiskalmoneter.htm dan http://localhost/bisniskeuangan/analisis.htm menjadi kandidat dari klaster pertama (*C1).*



Gambar 4.32 Tabel Nilai Similaritas Urut Maksimal

Tahap kedua proses klastering adalah mencari kadindat yang lain dalam klaster *C1* dengan mencari nilai similaritas maksimum dari dokumen http://localhost/bisniskeuangan/fiskalmoneter.htm dan

http://localhost/bisniskeuangan/analisis.htm. Implementasi dengan SQL dengan menggunakan query :

```
SELECT [situs]
      ,[situs2]
      ,[cosin]
    FROM [indonesianet].[dbo].[hargatermcosin]
where situs like
'http://localhost/bisniskeuangan/fiskalmoneter.htm'
order by cosin desc
go
SELECT [situs]
      ,[situs2]
      ,[cosin]
    FROM [indonesianet].[dbo].[hargatermcosin]
where situs like 'http://localhost/bisniskeuangan/analisis.htm'
order by cosin desc
go
```

	situs	situs2	cosin
1	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/bisniskeuangan/analisis.htm	0.99855906
2	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/bisniskeuangan/index.htm	0.86743516
3	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/bisniskeuangan/sosok.htm	0.76945245
4	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/bisniskeuangan/inspirasi.htm	0.56340057
5	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/bisniskeuangan/AS.Kambing.Hitam...	0.5057637
6	http://localhost/bisniskeuangan/fiskalmoneter.htm	http://localhost/edukasi/Jadi.Peneliti.Hukum.di.Bela	0.43659943

	situs	situs2	cosin
1	http://localhost/bisniskeuangan/analisis.htm	http://localhost/bisniskeuangan/index.htm	0.96601075
2	http://localhost/bisniskeuangan/analisis.htm	http://localhost/bisniskeuangan/ekonomi.htm	0.9499106
3	http://localhost/bisniskeuangan/analisis.htm	http://localhost/bisniskeuangan/sosok.htm	0.8908766
4	http://localhost/bisniskeuangan/analisis.htm	http://localhost/bisniskeuangan/inspirasi.htm	0.6135957
5	http://localhost/bisniskeuangan/analisis.htm	http://localhost/edukasi/Presiden.Dijadwal...	0.5509839
6	http://localhost/bisniskeuangan/analisis.htm	http://localhost/bisniskeuangan/AS.Kambin...	0.509839
7	http://localhost/bisniskeuangan/analisis.htm	http://localhost/edukasi/Jadi.Peneliti.Huku...	0.50268334

Gambar 4.33 Hasil Hitung Cosin Similaritas antar Dokumen

Hasil dari SQL adalah dokumen dengan nilai similaritas yang lebih besar dari nilai threshold akan masuk dalam satu klaster *C1*.

Proses akan dilanjutkan dengan mencari kandidat untuk klaster yang lain. Implementasi proses klaster akan dilakukan tahap yang sama mulai tahap kedua sampai dokumen dalam korpus habis dibuat klaster.

Gambar 4.34 Tampilan Klaster Hierarki dengan JSP

BAB V
HASIL DAN PEMBAHASAN

5.1. Data Penelitian

Data yang digunakan dalam penelitian diambil dari dokumen berita online http://www.kompas.com/archiveweb , data dalam bentuk format file teks. Untuk memvalidasi program aplikasi yang dibuat, koleksi data dikelompokkan menjadi bidang yaitu edukasi,olah raga, dan bisnis.

5.2 Alat Penelitian

Pada penelitian ini komputasi dilakukan dengan menggunakan komputer processor Intel Core 2 Duo, 2,4 GHz, RAM 1 GB, Hardisk 250 GB. Sistem Operasi yang digunakan adalah Windows XP. Bahasa pemrograman menggunakan Java dan menggunakan data base SQL Server 2008.

5.3 Testing dan Evaluasi

Setelah Sistem Informasi Temu Kembali Informasi Klastering Berita on Line dapat diimplementasikan sesuai dengan desain yang telah dibuat. Tahap selanjutnya adalah tahap melakukan percobaan atau testing dan evaluasi terhadap sistem yang dibuat. Pada tahap pengetesan ini penulis tidak menemukan kesalahan pada program baik secara logika maupun sintaks pada kode program.

Pengujian yang penulis lakukan dalam Sistem Informasi Temu Kembali Informasi Klastering Berita on Line yang berjumlah 60 file dalam format html, telah mampu untuk tidak melakukan indeks-indeks kata umum (stop word) dan telah membentuk kata dasar dari tiap *term* yang ada dalam dokumen abstrak tersebut. Selanjutnya setiap term telah dihitung frekuensinya dan diberikan pembobotan menggunakan cosine similaritas dan selanjutnya term tersebut disimpan pada database korpus.

Selanjutnya dalam pengujian terhadap sistem penulis melakukan pengujian input string *query* dan kemudian hasil pengujian input *query* dilakukan

pengukuran hasil retrieval (temu kembali informasi) hasil dengan pengujian *recall precision*.

5.4 Pengujian input query

Pada tahap pengujian input *query* dilakukan dengan cara memasukkan *query* "pendidikan", "jaringan", "bisnis", dan "manajemen bisnis". Terlihat pada Gambar 5.1 adalah salah satu contoh hasil tampilan dari input *query* "pendidikan". Hasil proses dari *query* akan ditampilkan dokumen-dokumen yang berada dalam klaster yang sama.

Gambar 5. 1 Hasil Dokumen Input Query "pendidikan"

5.5 Pengujian recall dan precision

Untuk mengevaluasi secara manual kesamaan diantara dokumen dalam cluster-cluster yang telah dikelompokkan digunakan standar sebagaimana Tabel 5.1. Tabel tersebut berisi berbagai kemungkinan hasil klasifikasi pada tiap *event* (*Per Event contengency table*).

Tabel 5.1. kategori hasil klasifikasi

	In Event	Not In event
In cluster	a	b
Not In Cluster	c	d

Tabel 5.1 menunjukkan bahwa hasil klasifikasi adakalanya memang termasuk event (a) yang dimaksud dan adakalanya tidak (b). Sedangkan dokumen yang tidak termasuk dalam hasil klasifikasi suatu *event*, adakalanya memang bukan anggota *event* itu (d) dan adakalanya ternyata seharusnya menjadi anggota event tersebut (c). Dalam hal ini, kempat parameter di atas digunakan untuk menghitung 2 parameter evaluasi, yakni :

1. *Recall*, yakni tingkat keberhasilan mengenali suatu event dari seluruh event yang seharusnya dikenali. Rumusnya adalah r = a/(a+c) untuk a+c > 0. Selain itu tidak didefinisikan
2. *Precision*, yakni tingkat ketepatan hasil klasifikasi terhadap suatu event.Artinya,dari seluruh dokumen hasil klasifikasi, berapa persenkah yang dinyatakan benar.
   Rumusnya adalah p = a/(a+b) jika a+b > 0.

Selain itu tidak didefinisikan Dari hasil evaluasi yang dilakukan terhadap data training yang diambil dari suara pembaruan *online m*ulai tanggal 1 Agustus 2001 sampai dengan 31 Agustus 2001 dengan perincian sebagai berikut :

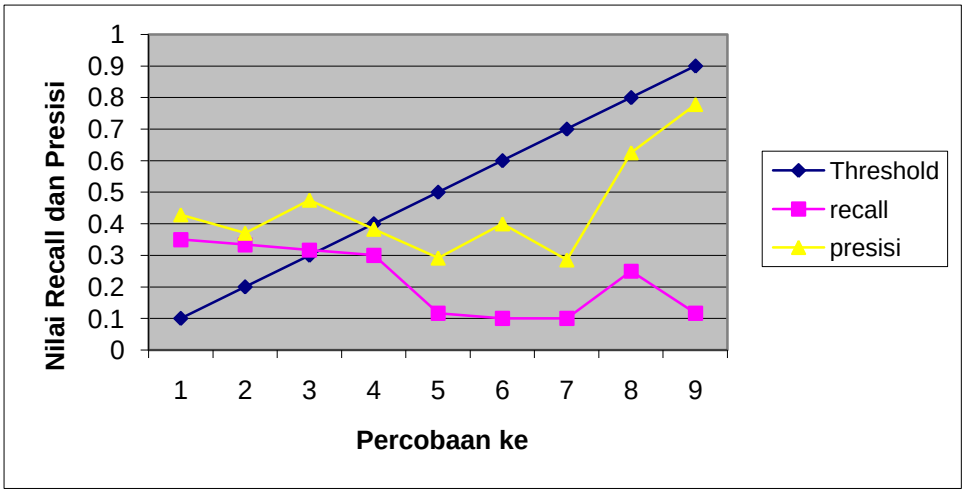
- Diambil 60 Dokumen sebagai Restrospective/training set yang dikalsifikasikan kedalam 23 event
- Setelah melalui proses *stemming* maka dapat dilakukan penghitungan frekuensi kata dalam dokumen dengan menggunakan kamus sejumlah 29.349 kata dasar bahasa Indonesia
- Dari matrik yang dibentuk dihasilkan sebanyak 13.000 *record* untuk kaitan dokumen dengan kata tf(t,d) dengan frekuensi di atas 0

Didapatkan hasil nilai Recall dan Precision sebagaimana Tabel 5.2.

Tabel 5.2 Tabel Hasil Uji Coba

Threshold	recall	presisi
0.1	0.35	0.428571429
0.2	0.333333333	0.37037037
0.3	0.316666667	0.475
0.4	0.3	0.382978723
0.5	0.116666667	0.291666667
0.6	0.1	0.4
0.7	0.1	0.285714286
0.8	0.25	0.625
0.9	0.116666667	0.777777778

Distribusi nilai kedua parameter dapat digambarkan dengan grafik pada Gambar 5.2



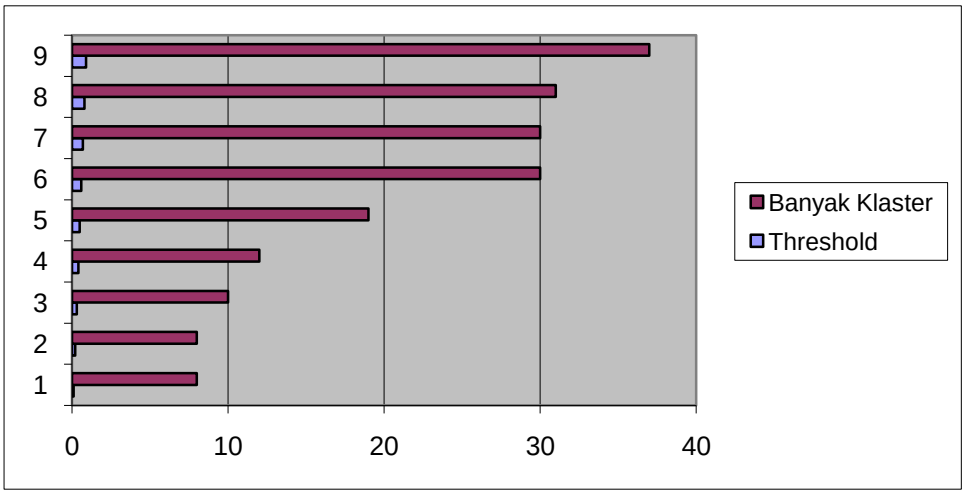
Gambar 5.2 Grafik Recall dan Precision untuk *Algoritma Single Pass*

Dari Gambar 5.2 terlihat bahwa nilai terbaik untuk recall didapat pada treshold pada angka 0.1. Sedanhkan nilai terbaik untuk presisi dihasilkan dari treshold pada angka 0.9. Treshold tersebut didapatkan dari percobaan secara linear terhadap 9 treshold yang berbeda. Percobaan dilakukan dengan menggunakan treshol mulai dari nilai treshold 0.1 sampai dengan mendapatkan jumlah kluster = jumlah dokumen atau nilai treshold di atas keseluruhan similarity maksimal. Untuk jumlah kluster yang dihasilkan dari 9 percobaan yang dilakukan dapat dilihat pada tabel 5.3

Tabel 5.3 Tabel Hasil Kluster

No	Threshold	Banyak Klaster
1	0.1	8
2	0.2	8
3	0.3	10
4	0.4	12
5	0.5	19
6	0.6	30
7	0.7	30
8	0.8	31
9	0.9	37

Sedangkan distribusi jumlah kluster yang dihasilkan dari percobaan yang dilakukan dapat dilihat pada Gambar 5.3



Gambar 5.3 Hasil Kluster

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Dari hasil penelitian yang telah dilakukan dapat disimpulkan hal-hal sebagai berikut:

1. Pembobotan term frekuensi dan cosine similaritas digunakan untuk menunjukkan kemiripan antar dokumen.
2. Sistem dapat menampilkan dokumen yang mempunyai kedekatan similaritas dari query yang diinputkan user.
3. Dokumen yang membahas topik yang sama cenderung untuk mengelompok menjadi satu klaster.
4. Klaster dapat membantu menemukan dokumen yang ada dalam satu klaster dengan query yang diinputkan user.
5. Klaster dapat membantu mendapatkan dokumen yang relevan.
6. Nilai treshold (nilai batas) yang paling bagus digunakan adalah 0.2 dengan nilai recall sebesar 0.33 dan precision 0.37

6.2. Saran

Dengan keterbatasan kemampuan dan waktu yang tersedia penulis menyadari bahwa masih banyak terdapat kekurangan dalam sistem ini terutama metode klastering yang digunakan. Kedepan nantinya diharapkan dalam pengembangan Sistem Informasi berbasis web, penulis menyarankan beberapa hal:

1. Sistem yang dibuat dapat dikembangkan lebih lanjut dengan menerapkan pada file teks Bahasa Indonesia dengan melakukan modifikasi stop word dan algoritma stemming agar hasil stemming lebih optimal.
2. Bagi peneliti lain yang berniat mengembangkan sistem Informasi Temu Kembali Bahasa Indonesia ini disarankan untuk menggunakan metode klastering yang diperluas sehingga hasil klaster dokumen akan lebih baik.
3. Untuk term yang bernilai 0 (nol) dalam setiao dokumen tidak perlu dilibatkan dalam perhitungan, karena hanya akan menambah waktu perhitungan.

DAFTAR PUSTAKA

Baeza-Yates, R. & Ribeiro-Neto, B**.,** 1999*, Modern Information Retrieval*, Addison-Wesley.

Fadillah Z Tala, 2003, *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands.

Ellis, David, 1996, *Progress and Problems in Information Retrieval*, 2nd ed. London: Library Association.

Gordon, Michael D., 1991, *User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm,*. Journal of American Society for Information Science, 311-322.

Karypis G., Zhao Y., 2004, *Hierarchical Clustering Algorithms for Document Datasets, University of Minnesota, Department of Computer Science and Engineering and Digital Technology Center and Army HPC Research Center, Minneapolis, MN 55455.*

Pressman R, 1997, *Software Engineering*, Mc Graw Hill, USA.

Rijsbergen, C. J.,1979, *Information Retrieval*, Information Retrieval Group, University of Glasgow.

Salton, G., 1989, *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer*, Addison – Wesly Publishing Company, Inc. All rights reserved.

Salton, G. and Buckley, 1988, *Term Weigting Approaches in Automatic Text Retrieval, Department of Computer Science*, Cornell University, Ithaca, NY 14853, USA.

Salton, G., 1971, *Cluster Search Strategies and the Optimization of Retrieval Efectiveness*, dalam G. Salton, ed. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Englewood Cliffs: Prentice-Hall, 223-242

Steinbach, M., Xiong H., Ruslim A., Kumar V., 2007, *Characterizing Pattern Preserving Clustering, Department of Management Science and Information Systems Rutgers*, the State University of New Jersey, USA.

Wen Yue, 2005, *Using Query Expansion and Classification for Information Retrieval, College of Computer and Communication*, Hunan University ChangSha, Hunan Province, 410082,China.

Zhang J., Jianfeng G., Ming Z., Jiaxing W., 2001, *Improving the Effectiveness of Information Retrieval with Clustering and Fusion*, Computational Linguistics and Chinese Language Processing, Vol. 6, No. 1, February 2001, pp. 109-125.

http://journal.uii.ac.id/index.php/Snati/issue/view/

DAFTAR ISI