BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada saat ini informasi sangatlah mudah didapat salah satunya adalah dari internet kita dapat mendapatkan informasi yang sangat luas. Dengan banyaknya informasi yang didapat maka akan menyulitkan dalam menemukan dokumen seperti yang diinginkan. Dengan semakin bertambahnya dokumen yang didapat, pendayagunaan sistem temu kembali dokumen menjadi penting agar dapat menghemat waktu dan kerja untuk mendapatkan dokumen yang mirip (*similar*) dengan kata kunci (*query*) yang diipuntkan oleh pengguna. Pada prinsipnya, penyimpanan dokumen teks dan proses pencarian kembali dokumen tersebut sifatnya sederhana, selama ada kumpulan dokumen yang disimpan dan pengguna yang memberikan pertanyaan ataupun kebutuhan. Maka sistem temu kembali dokumen dapat mengembalikan kumpulan dokumen yang mirip dengan menghitung *similarity* atau tingkat kesamaan antara dokumen dengan *query* yang diinputkan oleh pengguna.

Hasil pencarian *query* atau kombinasi kata yang diberikan pengguna dikembalikan oleh sistem temu kembali dokumen ketika kata-kata tersebut ditemukan pada kumpulan dokumen. Sehingga jumlah kemunculan dari *query* pada tiap dokumen dan posisi rinci kata (*term*) tersebut juga diperlukan. Oleh karena itu, lalu digunakanlah algoritma pencarian kata secara *sequensial* dan *pattern matching* karena sederhana dan mudah diimplementasikan. Namun pada implementasinya diharapkan algoritma yang digunakan untuk pencarian dokumen diharapkan dapat digunakan dapat menampung koleksi dokumen dengan ukuran besar atau banyak. Sehingga kemudian dipertimbangkanlah untuk membangun struktur data pada koleksi dokumen yang disebut indeks untuk mempercepat proses pencarian.

Perubahan ini sangatlah memuaskan dan mampu meningkatkan performansi pencarian sehingga lebih cepat untuk koleksi dokumen yang besar dan banyak jumlahnya. Implementasi dari teknik pengindeksan yang digunakan salah satunya adalah *indeks inverted* yang terdiri dari daftar kata-kata yang telah diekstraksi, posisi kemunculan kata secara rinci.

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang dapat dirumuskan adalah bagaimana membuat aplikasi dengan algoritma yang dapat menghemat waktu pencarian dokumen dan mampu mengindek kata pada dokumen teks berbahasa Indonesia yang dapat memberikan informasi dokumen teks dan posisi teks secara tepat.

1.3. Batasan Masalah

Dalam penelitian ini ada beberapa pembatasan masalah yang dilakukan, yaitu: hanya melakukan implementasi pembuatan struktur data dokumen teks dengan metode indeks inverted.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah :

a. Merancang dan membuat aplikasi yang dapat membuat struktur data untuk melakukan indeks teks dokumen bahasa Indonesia.
b. Merancang dan mebuat aplikasi yang dapat melakukan pencarian kemiripan dokumen teks bahasa Indonesia dengan teks yang diinputkan pengguna.

1.5. Metodologi Penelitian

1.5.1. Obyek Penelitian

Obyek penelitian dari penelitian ini adalah dokumen teks abstrak skripsi Fakultas Teknologi Informasi Unisbank Semarang.

Data Yang diperlukan

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

a. Data primer

Data yang diperoleh langsung dari situs basis data abstrak skripsi Fakultas Teknologi Informasi Unisbank Semarang.

b. Data Sekunder

Data yang diperoleh dengan membaca dan mempelajari referensi mengenai stemming kata, indeks, stopword dan pembobotan *term* (kata).

1.5.2. Teknik Pengumpulan Data

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

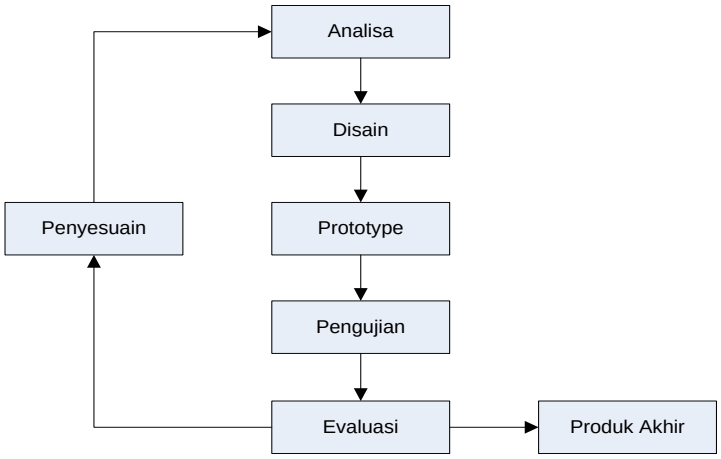
Dengan melakukan pengamatan dan pencatatan secara sistematis tentang hal-hal yang berhubungan dengan basis data dokumen teks dan kemampuan pencarian kemiripan dokumen.

b. Studi Pustaka

Dengan pengumpulan data dari bahan-bahan referensi, arsip, dan dokumen yang berhubungan dengan permasalan dalam penelitian ini.

1.5.3. 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 1.1 memperlihatkan tahapan pada prototyping.



Gambar 1.1 Tahapan Prototyping (Pressman, 2001)

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

a. **Analisa**

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

b. **Desain**

   Pada tahap ini dibangun rancangan sistem dengan beberapa diagram bantu seperti Data Flow Diagram, flowchart, class diagram, ERD

c. **Prototype**

   Pada tahap ini dibangun aplikasi berbasis web yang sesuai dengan disain dan kebutuhan sistem.

d. **Pengujian**

   Pada tahap ini dilakukan pengujian pada pustaka fungsi yang sudah dibangun.

e. **Evaluasi**

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

f. **Penyesuaian**

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

1.6. Sistematika Penulisan

Sistematika penulisan terdiri dari empat bab yang masing-masing bab menguraikan hal-hal yang berbeda.

Bab I Pendahuluan

Pada bab ini diuraikan mengenai permasalahan yang dibahas secara umum yang meliputi : latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian.

Bab II Landasan Teori

Dalam Bagian ini memuat hal-hal teoritis yang ada hubunganya dengan penyelesaian masalah dalam penelitian ini. Pada bab ini di uraikan antara lain tentang stemming bahasa Indonesia, Indeks, similaritas, *query*

Bab III Perancangan Sistem

Dalam bagian ini dibahas tentang rancang bangun aplikasi. Rancangan meliputi Arsitektur sistem, DFD, E-R Diagram, Class Diagram dan Flowchart.

Bab IV Implementasi

Dalam bagian ini dibahas tentang langkah-langkah implementasi untuk perangkat lunak aplikasi yang telah selesai di rancang pada bab III. Disini disertakan juga kode sumber dari fungsi-fungsi utama.

Bab V Kesimpulan Dan Saran

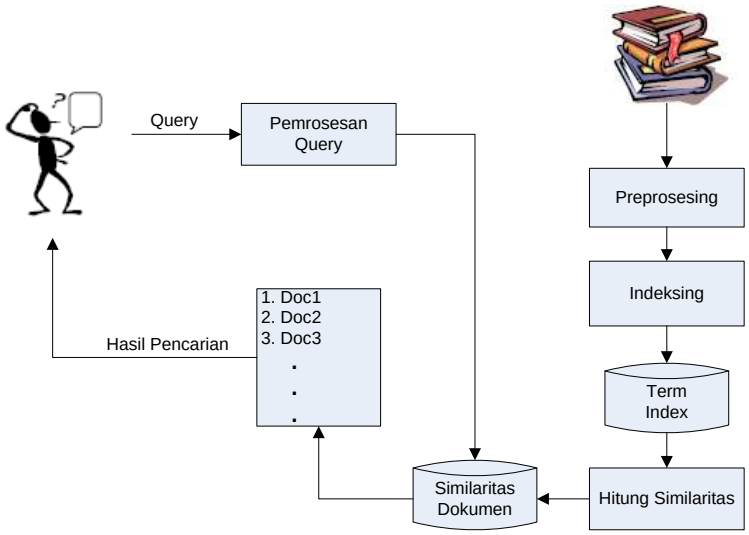
Pada bab ini berisi kesimpulan dan saran dari penelitian ini. Selain hasil penelitian berupa perangkat lunak aplikasi, juga dihasilkan juga saran-saran untuk penelitian lebih lanjut pada bidang pencarian dokumen teks.

BAB II
LANDASAN TEORI

2.1. Temu Kembali Informasi

Teknik pencarian informasi pada sistem Information Retrieval berbeda dengan sistem pencarian pada sistem manajemen basisdata (DBMS) . Dalam sistem temu kembali terdapat dua bagian utama yaitu bagian pengindeksan (*indexing*) dan pencarian (*searching*). Kedua bagian tersebut memiliki peran penting dalam proses temu kembali informasi.

Sebagai suatu sistem, sistem temu kembali informasi memiliki beberapa bagian yang membangun sistem secara keseluruhan. Gambaran bagian-bagian yang terdapat pada suatu sistem temu kembali informasi digambarkan pada gambar 2.1. Terlihat pada gambar bahwa kumpulan dokumen teks akan dilakukan proses preprosesing kemudian dilakukan proses indeks *term* (kata). Hasil proses indeks akan disimpan dalam database selanjutnya akan dilakukan proses hitung similaritas antar dokumen. Hasil hitung similaritas antar dokumen akan disimpan dalam database. Nilai kemiripan (similaritas) dokumen yang disimpan dalam database akan digunakan untuk proses *query*. Hasil *query* adalah dokumen-dokumen yang mirip (similar) dengan kata kunci yang diinputkan pengguna.



Gambar 2. 1 Proses Pencarian Dokumen Teks

Modul untuk sistem pencari dokumen teks meliputi 1) modul preprosesing : tokenization (tokenisasi), filtering, stopword removal (pembuangan stopword/kata umum), stemming (pembentukkan kata dasar), 2) modul indeksing, 3) hitung similaritas, 4) pemrosesan *query*

Salah satu aplikasi dari sistem temu kembali informasi adalah *search engine* atau mesin pencarian yang terdapat pada jaringan internet. Pengguna dapat mencari halaman-halaman web yang dibutuhkannya melalui *search engine.* Contoh lain penerapan dari sistem temu kembali informasi adalah sistem informasi perpustakaan, data/*text mining, knowledgeacquisition*.

Sistem temu kembali informasi terutama berhubungan dengan pencarian informasi yang isinya tidak memiliki struktur. Demikian pula ekspresi kebutuhan pengguna yang disebut *query*, juga tidak memiliki struktur. Hal ini yang membedakan sistem temu kembali informasi dengan sistem basis data. Dokumen adalah contoh informasi yang tidak terstruktur. Isi dari suatu dokumen sangat tergantung pada pembuat dokumen tersebut.

2.2. Stemming Bahasa Indonesia

Penelitian pencarian tentang efek *stemming* bahasa Indonesia dalam proses temu kembali dilakukan oleh Tala (2003). Algoritma *stemmer* untuk bahasa Indonesia yang dikembangkan adalah algoritma *purely ruled-based stemmer*. Algoritma ini adalah mengadopsi dari algoritma *English Porter Stemmer* yang dikembangkan oleh Frakes (1992). Dipilihnya algoritma Porter untuk dikembangkan sebagai algoritma stemmer untuk bahasa Indonesia karena pemikiran dasar dari algoritma stemmer Porter cocok dengan struktur morfologi kata-kata di dalam bahasa Indonesia. Perbedaan algoritma ini dengan algoritma yang telah dikembangkan oleh Nazief dan Adriani yaitu tidak adanya *dictionary* sehingga algoritma dapat dikatakan murni berbasis *rule* (*purely rule-based stemmer*).

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 2. 1 Awalan dan akhiran yang salah (Tala, 2003)

Prefix	**Suffix**
Ber	i
Di	an
Ke	i|kan
Men	an
Peng	i|kan
Ter	an

7. Awalan ganda yang mungkin terjadi.

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

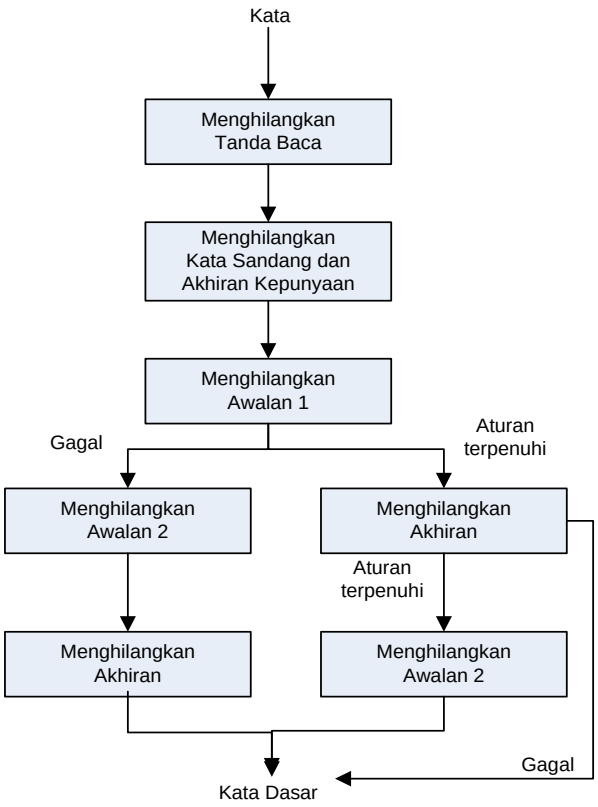
Prefix 1	Prefix 2
Meng	Per
Di	Ber
Ter	
Ker	

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 2.2. 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*.

2.2.1. 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).



Gambar 2. 2 Proses Stemming Algoritma Tala (Tala, 2003)

2.2.2 Proses menghilangkan kata sandang dan kepunyaan

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

Tahap 1 : Akhiran –lah, -kah, -pun, -tah diperlihatkan pada tabel 2.3, adalah akhiran yang berfungsi untuk meyakinkan atau menekankan dan sama sekali tidak mempunyai makna.

Tabel 2. 3 Tabel akhiran –lah, -kah, -pun

Suffix	Replacement	Measure Condition	Additional Condition	Example
kah	NULL	2	NULL	bukukah – buku
Lah	NULL	2	NULL	adalah – ada
pun	NULL	2	NULL	bukupun – buku

Tahap 2 :Akhiran –ku, -mu, -nya diperlihatkan pada tabel 2.4, adalah akhiran yang melekat pada kata dan membentuk kata ganti punya.

Tabel 2. 4 Tabel akhiran kepunyaan

Suffix	Replacement	Measure Condition	Additional Condition	Example
Ku	NULL	2	NULL	bukuku - buku
Mu	NULL	2	NULL	bukumu- buku
nya	NULL	2	NULL	Bukunya - buku

2.2.3 Menghilangkan awalan 1

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala membagi awalan 1 dalam tahap 1 yang harus diproses secara urut. Tabel 2.5 adalah daftar awalan 1 yang masuk ke tahap ketiga pada proses stemming Tala

Tabel 2. 5 Tabel awalan 1

Suffix	Replacement	Measure Condition	Additional Condition	Example
meng	NULL	2	NULL	mengukur - ukur
meny	s	2	V….	menyapu – sapu
men	NULL	2	NULL	menduga – duga menuduh - tuduh
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

2.2.4 Menghilangkan awalan 2.

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi awalan 2 dalam 1 tahap proses yang harus diproses secara urut. Tabel 2.6 adalah daftar awalan 2 yang masuk ke tahap keempat pada proses stemming Tala.

Tabel 2. 6 Tabel awalan 2

Suffix	Replacement	Measure Condition	Additional Condition	Example
ber	NULL	2	NULL	berlari – lari
bel	NULL	2	Ajar	belajar – ajar
Be	NULL	2	Ker….	bekerja – kerja
per	NULL	2	NULL	perjelas – jelas
pel	NULL	2	Ajar	pelajar – ajar
Pe	NULL	2	NULL	pekerja – kerja

2.2.5 Menghilangkan akhiran.

Pada proses ini dokumen melalui perlakuan untuk menghilangkan awalan, stemmer Tala melokalisasi akhiran dalam proses tahap 1 yang harus diproses secara urut. Tabel 2.7 adalah daftar akhiran yang masuk ke tahap kelima pada proses stemming Tala.

Tabel 2. 7 Tabel akhiran

Suffix	Replacement	Measure Condition	Additional Condition	Example
Kan	NULL	2	prefix ɛ́ {ke, peng}	tarikkan → tarik mengambilkan → ambil
An	NULL	2	prefix ɛ́ {di, meng, ter}	makanan → makan (per)janjian → janji
I	NULL	2	V|K…c1c1, c1≠s, c2 ≠ I and prefik ɛ́ {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 2.1.

2.3. Pembobotan Istilah

Masalah dalam *term weighting* (pembobotan istilah) menurut Salton (1989) adalah bagaimana memutuskan bobot tiap istilah yang muncul dalam *query* atau koleksi dokumen memperlihatkan bahwa pembobotan term dapat meningkatkan kinerja perolehan informasi yang didapatkan dengan term tanpa bobot. Jadi, skema pembobotan istilah yang baik akan terlihat dalam hasil saat membedakan istilah yang diinginkan dan tidak diinginkan.

Pembobotan istilah dalam index rangking pada umumnya dibagi menjadi tiga katagori sebagai berikut :

1. Frekuensi istilah (*term frequency)* $tf_{i,j}$: banyaknya kemunculan istilah t_i dalam dokumen d_j. Istilah-istilah yang mempunyai frekuensi lebih tinggi dalam sebuah dokumen dipertimbangkan sebagai descriptor yang lebih baik dari isi sebuah dokumen.
2. Frekuensi dokumen (*document frequency)* df_i : banyaknya dokumen dalam koleksi dimana istilah t_i muncul di dalamnya. Sebuah istilah yang relevan sering muncul beberapa kali dalam sebuah dokumen. Di sisi lain, istilah yang tidak relevan sering muncul secara homogen dalam semua dokumen.
3. Frekuensi koleksi (collection frequency) cf_i : banyaknya kemunculan istilah t_i dalam koleksi.

Masalah perangkingan/pembobotan dokumen adalah bagaimana memanfaatkan informasi bobot istilah yang telah dihasilkan untuk memperoleh dokumen yang relevan ke *query*. Perangkingan dokumen seharusnya mencakup dan menyatukan informasi bobot istilah (yakni,

bobot istilah *query* dan bobot istilah dokumen) menjadi model pembobotan istilah. Beberapa model pembobotan istilah yaitu (Salton, 1989):

a. Model Boolean

   Model Boolean retrieval merupakan model untuk sistem temu kembali informasi yang bertujuan untuk membentuk *query* dalam format ekspresi Boolean dari *term-term*, seperti AND, OR dan NOT. Seperti namanya Boolean Model , maka setiap *term* dalam setiap dokumen merupakan vektor 0 (nol) atau 1 (satu). Bernilai 1 jika *term* tersebut terdapat dalam dokumen, dan benilai 0 jika *term* tersebut tidak terdapat dalam dokumen.

b. Model Vector Space Model

   Vector Space Model (VSM) merepresentasikan dokumen dan *query* dengan vektor-vektor bobot istilah dalam sebuah ruang multidimensi. Dalam VSM, sebuah istilah direpresentasikan dengan sebuah dimensi dari ruang vektor. Jadi, relevansi sebuah dokumen ke sebuah *query* didasarkan pada similaritas diantara vektor dokumen dan vektor *query*. Koordinat dari bobot istilah secara dasarnya diturunkan dari frekuensi kemunculan dari istilah dengan mempertimbangkan berbagai variasi sebagai contoh, frekuensi istilah, frekeunsi dokumen dan frekuensi koleksi.

c. Model Probabilistic

   Ide dasar dari model probabilistic adalah bahwa jika diketahui beberapa dokumen relevan ke sebuah *query*, maka bobot yang lebih tinggi diberikan ke isitilah-isitilah yang mana muncul dalam dokumen-dokumen tersebut untuk mencari dokumen-dokumen lain yang relevan. Dalam model probabilistik, probabilitas munculnya setiap istilah dilatih dengan sebuah himpunan dokumen, himpunan *query* dan himpunan penentuan similaritas diantara tiap dokumen dan tiap *query*. Teorema Bayes sering digunakan dalam model ini untuk memberitahu bagaimana memperbaharui atau merevisi kepercayaan berkaitan dengan *query* yang baru dan dokumen baru.

   Dokumen yang relevan ke *query* yang baru dapat diperoleh didasarkan pada probabilitas kemunculan istilah *query* dalam himpunan dokumen training.

2.4. Cosine Similaritas

Kesamaan antar dokumen dapat diukur dengan fungsi similaritas (mengukur kesamaan) atau fungsi jarak (mengukur ketidaksamaan). Beberapa fungsi similaritas atau fungsi jarak yang dapat dijumpai adalah *Disk, Jaccard, Overlap, Asimmetric, Minowski distance, Euclidean distance, Pearson Correlation, Cosine.*

Untuk tujuan klastering dokumen fungsi yang baik adalah fungsi *Cosine Similaritas*. Berikut adalah persamaan dari metode *Cosine Similaritas* (Salton, 1989): Untuk notasi himpunan digunakan rumus :

$$Similarity(X,Y) = \frac{X \bigcap Y}{|X|^{\frac{1}{2}} . |Y|^{\frac{1}{2}}} \quad (2.1)$$

Dimana :

$X \bigcap Y$ 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 :

$$Similarity(x,y) \quad \frac{\sum_{i=1}^{t} x_i \, y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}} = \quad (2.2)$$

dimana :

x dan y adalah dukumen yang berbeda.

x_i = term i yang ada pada dokumen x

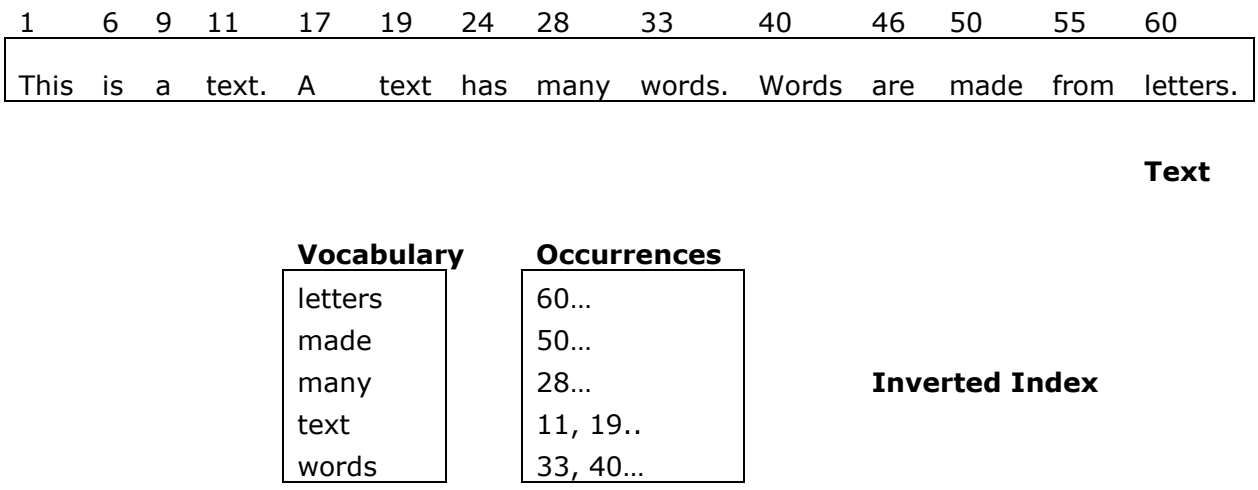
y_i = term i yang ada pada dokumen y

2.5. Inverted Indeks

Inverted file atau index inverted adalah mekanisme untuk pengindeksan kata (*term*) 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 2.3.



Gambar 2. 3 Contoh Teks dan Inverted File-nya

Pada gambar 2.3 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.

Operasi Kata dan Pengindekan Otomatis

Berikut ini proses-proses yang harus dilakukan dalam operasi kata kunci

1. Identifikasi kata dalam dokumen
2. Buang kata tidak penting dari daftar kata setelah dicocokkan dengan kamus khusus atau daftar kata tidak penting
3. Identifikasi sinonim dengan mencocokkan setiap kata pada daftar kata sinonim
4. Ambil akar kata dengan menggunakan suatu algorithm yang menghilangkan kata depan dan belakang
5. Hitung frequency kata dalam tiap dokumen

6. Hitung bobot kata
7. Buat basis data berbasis invert untuk term-term tersebut beserta bobotnya

2.6. Teknik Boolean Temu Kembali Informasi

Model Boolean dalam sistem temu kembali merupakan model yang paling sederhana. Model ini berdasarkan teori himpunan dan aljabar Boolean. Dokumen adalah himpunan dari istilah (*term*) dan *query* adalah pernyataan Boolean yang ditulis pada term. Dokumen diprediksi apakah relevan atau tidak. Model ini menggunakan operator boolean. Istilah dalam sebuah *query* dihubungkan dengan menggunakan operator AND, OR atau NOT. Metode ini merupakan metode yang paling sering digunakan pada`mesin penelusur (search engine) karena kecepatannya.

Keuntungan menggunakan model Boolean (Baeza, 1999) :

1. Model Boolean merupakan model sederhana yang menggunakan teori dasar himpunan sehingga mudah diimplementasikan.
2. *Query* sederhana dan mudah dimengerti.
3. Operator Boolean bisa mendekati bahasa alami. Operator AND dapat menemukan hubungan antar konsep, OR dapat menemukan terminologi alternatif, NOT dapat menemukan arti alternatif.

Teknik *boolean* merupakan suatu cara dalam mengekspresikan keinginan pemakai ke sebuah query dengan memakai operator-operator *boolean* (Salton,1989) yaitu : "and", "or", dan "not". Adapun maksud dari operator "and" adalah untuk menggabungkan istilah-istilah kedalam sebuah ungkapan, dan operator "or" adalah untuk memperlakukan istilah-istilah sebagai sinonim, sedangkan operator "not" merupakan sebuah pembatasan. Pada teknik *boolean* sederhana, *query* diproses sesuai dengan operator yang digunakan dan menampilkan dokumen berdasarkan urutan dokumen ditemukan. Sedangkan pada teknik *boolean* berperingkat, dokumen diperingkat berdasarkan bobot dari dokumen.

Pencocokkan dengan metode *boolean* semata-mata memilih dokumen yang mengandung string yang *match* dengan *query* dan menerapkan beberapa hubungan logika dengan koleksi dokumen yang mengandung string yang *match* tersebut. Koleksi dokumen cukup disimpan dalam struktur data

terindeks berupa *hashtable* atau daftar *inverted file*. Karena metode ini hanya dapat menyatakan dokumen sebagai *match* atau tidak dengan *query*, maka ia tidak dapat merangking dokumen berdasarkan urutan relevansinya terhadap *query*. Metode ini juga tidak dapat memperhitungkan makna kata, bentuk kata, kemungkinan sinonim atau polisemy. Kelemahan metode ini untuk ukuran basis data yang besar dengan metode *boolean match* dapat menghasilkan daftar yang sangat besar dan terlalu banyak mengandung dokumen yang tidak relevan.

BAB III
PERANCANGAN SISTEM

3.1. Data Penelitian

Data yang digunakan dalam penelitian diambil dari dokumen teks abstrak skripsi Fakultas Teknologi Informasi Unisbank Semarang, data dalam bentuk format file teks sejumlah 101 dokumen abstrak. Untuk memvalidasi program aplikasi yang dibuat, koleksi data dikelompokkan menjadi beberapa kelompok topik yaitu sistem informasi, sistem penunjang keputusan, sistem pakar, sistem geografis.

3.2. Ruang lingkup produk

Fungsi proyek pengembangan perangkat lunak ini adalah untuk membuat sebuah aplikasi baru yang disebut : Prototipe sistem pencari dokumen teks Bahasa Indonesia dengan format teks. Pengguna aplikasi ini adalah user yang menginginkan informasi abstrak naskah dibidang Teknologi Informasi. Aplikasi ini memungkinkan user untuk mendapatkan data abstrak skripsi Bahasa Indonesia. Aplikasi ini akan memberikan beberapa fasilitas, yaitu:

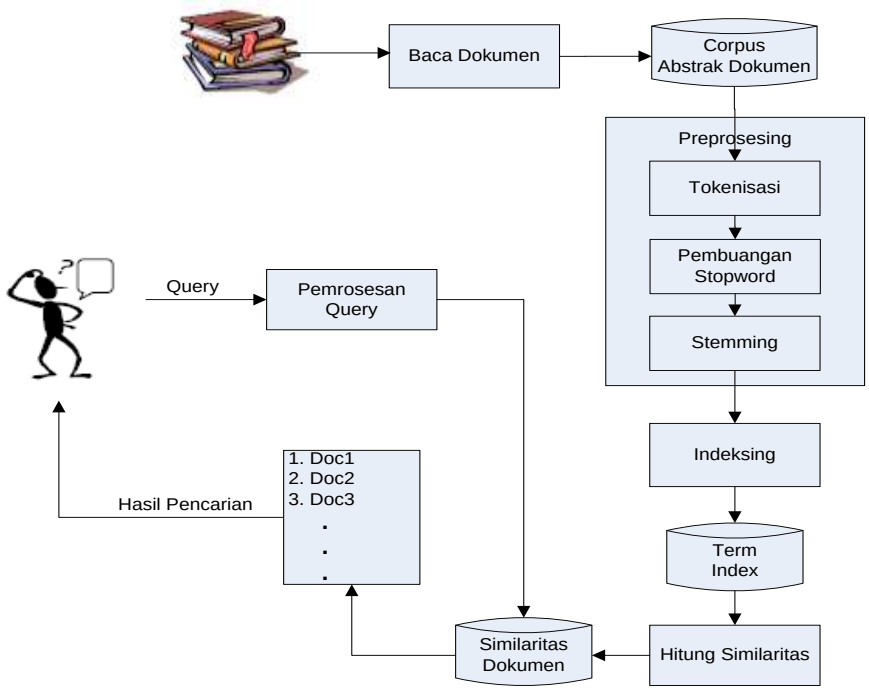
1. Program aplikasi dapat diakses oleh user melalui web.
2. User dapat menggunakan program aplikasi ini untuk melakukan pencarian dokumen teks Bahasa Indonesia.

Dalam pengembangan aplikasi prototype mesin pencari dokumen teks untuk data abstrak skripsi dengan format data teks ini diharapkan dapat memberikan beberapa keuntungan bagi user, yaitu :

1. Melakukan pencarian dokumen teks bahasa Indonesia.
2. Mendapatkan dokumen teks yang *similar* (mirip) dengan *query* (kata) kunci yang diinputkan user.

3.3. Arsitektur Sistem Pencari Dokumen Teks

Pada gambar 3.1 terlihat arsitektur system pencari dokumen teks. Modul dari system terdiri dari 4 modul yaitu : 1) modul preprosesing, yang terdiri dari modul tokenisasi, modul pembuangan stopword, modul stemming, 2) modul indeksing, 3) modul similaritas dan 4) modul *query*



Gambar 3. 1 Arsitektur Sistem Pencari Dokumen Teks

Masing-masing modul Sistem Pencari Dokumen Teks dapat dijelaskan sebagai berikut :

Modul Baca Dokumen

Mrupakan modul yang melakukan proses baca file abstrak dokumen dalam bentuk teks. Proses baca file dilakukan secara baris perbaris. Kemudian masing-masing file akan dilakukan proses baca isi file secara baris perbaris. Pada proses baca file dan proses baca isi file dilakukan pembersihan dokumen. Yaitu pembersihan partikel-partikel kata seperti tanda baca, tag-tag html. Hasil proses baca file dan isi file akan disimpan dalam tabel corpus.

Modul Tokenisasi

Isi abstrak skripsi yang disimpan dalam tabel corpus akan dilanjutkan dengan proses tokenisasi. Proses tokenisasi adalah 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.

Modul Pembuangan Stopword

Proses pembuangan stopword dimaksudkan untuk mengetahui suatu kata masuk ke dalam stopword 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 stopword tersimpan dalam suatu tabel, dalam penelitian ini menggunakan daftar stop word yang digunakan oleh Tala (2003), yang merupakan stopword Bahasa Indonesia yang berisi kata-kata seperti ; ini, itu, yang, ke, di, dalam, kepada, dan seterusnya sebanyak 780 kata. Kata ini dapat dilihat di lampiran 1.

Modul Stemming

Proses *stemming* adalah bagian dari proses preprosesing, tujuan utama dari proses stemming adalah mengembalikan kata dalam bentuk dasarnya. Term yang diperoleh dari tahap pembuangan stopword 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.

Modul Indeksing

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 yang dapat menyimpan dan mengidentifikasi tiap-tiap term yang masuk ke dalam koleksi dokumen. Sehingga term yang tersimpan dalam tabel koleksi dapat teridentifikasi keberadaannya berdasarkan nama dokumen. Setelah kata telah dikembalikan dalam bentuk asal (kata dasar), kata-kata tersebut disimpan kedalam tabel basis data. Penelitian ini menggunakan metode *Index Inverted*, 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

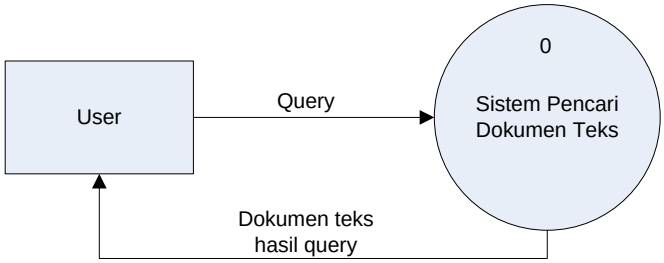
Modul Hitung Similaritas

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 (*term*). Pada modul ini akan dihitung presentase kemunculan tiap istilah dan presentase kesamaan antar dua term. Metode yang digunakan untuk menghitung adalah metode *cosine simmilarity* dengan menggunakan rumus seperti diuraikan pada persamaan (2.1).

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.

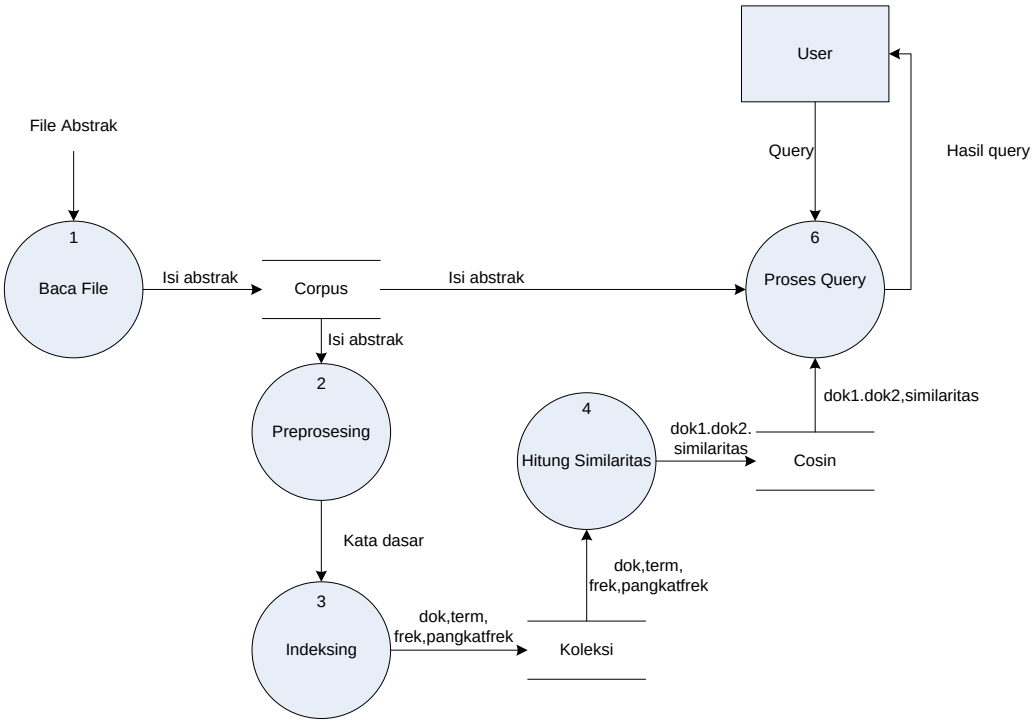
3.4. Perancangan Sistem Menggunakan Data Flow Diagram

Perancangan dengan Data Flow Diagram (DFD) dimulai dengan perancangan *context diagram*. Dengan *context diagram* akan terlihat siapa yang berinteraksi dengan sistem dan apa respon sistem terhadap interaksi tersebut. Seperti terlihat pada gambar 3.2 terlihat bahwa *user* dapat memasukkan *query* ke dalam sistem. *Query* inilah yang akan diolah dan hasilnya dikirim kembali kepada *user* dalam bentuk abstrak dokumen yang sudah terurut berdasarkan similaritas dokumen.



Gambar 3. 2 Context Diagram Sistem Pencari Dokumen Teks

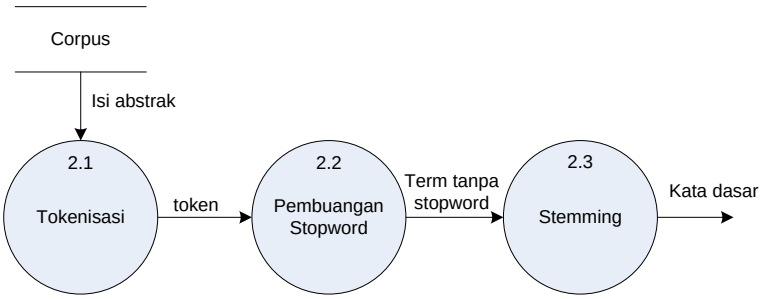
Dari context diagram lebih lanjut akan dijabarkan DFD level 1 sistem pencari dokumen teks seperti terlihat pada gambar 3.3.



Gambar 3. 3 DFD Level 1 Sistem Pencari Dokumen Teks

Terlihat pada gambar 3.3 proses pembentukan dokumen dimulai dari proses baca file. Pada proses baca file, file dokumen abstrak dan isi abstrak akan dibaca secara baris perbaris. Selain itu pada proses baca file akan dilakukan juga penghilangan partikel kata. Hasil proses baca file akan disimpan dalam tabel corpus. Dari tabel corpus ini akan dilanjutkan ke proses preprosesing, yaitu proses pembersihan dokumen teks sebelum diindeks. Output dari proses preprosesing adalah kata dasar, selanjutnya kata dasar yang tersebut akan dilanjutkan ke proses indeksing. Hasil dari proses indeksing akan disimpan ke tabel koleksi. Tabel koleksi memuat: term, nama file (dokumen), frekuensi term, pangkat frekuansi. Kemudian proses akan dilanjutkan dengan proses hitung simmilaritas. Proses hitung similaritas akan disimpan dalam tabel cosin. Dengan tabel cosin ini akan digunakan untuk proses *query* yang diinputkan user. Hasil *query* adalah dokumen yang memiliki kemiripan dengan dirangking dari yang maksimum.

Selanjutnya pada gambar 3.4 adalah DFD level 2 yang menjabarkan lebih lanjut dari proses 2 preprosesing dari gambar 3.3. Proses preprosesing akan dijabarkan menjadi 3 proses yaitu proses tokenisasi, proses pembuangan stopword dan proses stemming. Proses tokenisasi adalah proses membentuk dokumen menjadi token-token. Antara *term* yang satu dengan yang lain dipisahkan dengan spasi. Proses dilanjutkan dengan pembuangan *stopword*, yaitu kata-kata umum yang tidak memiliki makna akan dibuang dan tidak diikutkan pada proses berikutnnya. Setelah pembuangan *stopword* selesai dilakukan, proses akan dilanjutkan dengan proses *stemming* yaitu proses pembentukkan kata dasar. 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. Output dari proses *stemming* adalah kata dasar yang akan dilanjutkan pada proses indeksing untuk disimpan dalam koleksi dokumen.



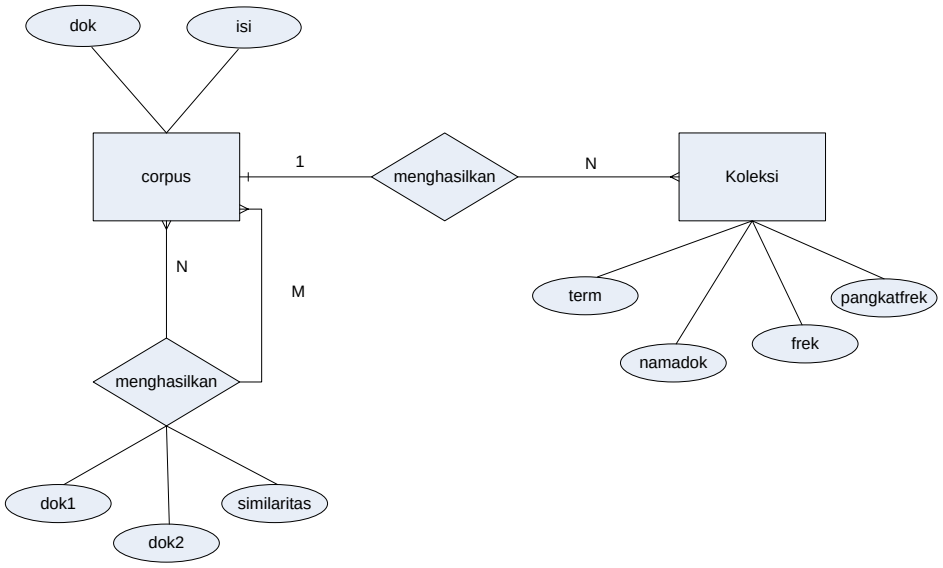
Gambar 3. 4 DFD Level 2 Proses Tokenisasi

3.5. Perancangan dengan menggunakan Entity Relationshyp Diagram

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

Dalam rancangan sistem basis data untuk sistem pencari dokumen teks ini digunakan ERD atau Diagram Hubungan Entitas dan desain tabel

untuk menggambarkan atribut-atributnya yang ditunjukkan pada gambar 3.5.



Gambar 3. 5 ERD Sistem Pencari Dokumen Teks

Terlihat pada gambar 3.5 bahwa *entity* corpus berelasi satu ke banyak (*one to many*) dengan *entity* koleksi. Yaitu satu dok dalam entity corpus dapat menghasilkan banyak dok dalam entity koleksi. Entity corpus berelasi recursive dengan relasi banyak ke banyak (many to many). Hasil relasi disimpan sebagai tabel cosin. Yaitu tabel yang menyimpan nilai simmilaitas antara dokumen yang satu dengan dokumen yang lainnya.

Berikut adalah transformasi ERD ke tabel yang digunakan beserta tipe datanya:

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

Tabel 3. 1 Tabel stopword.dbo

Nama Field	Type Field	Keterangan
term	varchar(200)	term stopword

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 3. 2 Tabel corpus.dbo

Nama Field	Type Field	Keterangan
dok	varchar(200)	nama file dokumen
isi	text	isi dari dokumen

3. Tabel koleksi akan menyimpan *term* output dari proses *stemming*, frekuensi tiap *term* tercatat di *field* frek dan pangkat frekuensi *term* dicatat dalam *field* pangkatfrek.

Tabel 3. 3 Tabel koleksi.dbo

Nama Field	Type Field	Keterangan
dok	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 menyimpan nilai similaritas antara dokumen yang satu dengan dokumen lainnya. Nilai similaritas similaritas antar dokumen akan disimpan di *field* similaritas. Nama dokumen yang satu dengan yang lain dibedakan dengan nama *field* dok1 dan dok2

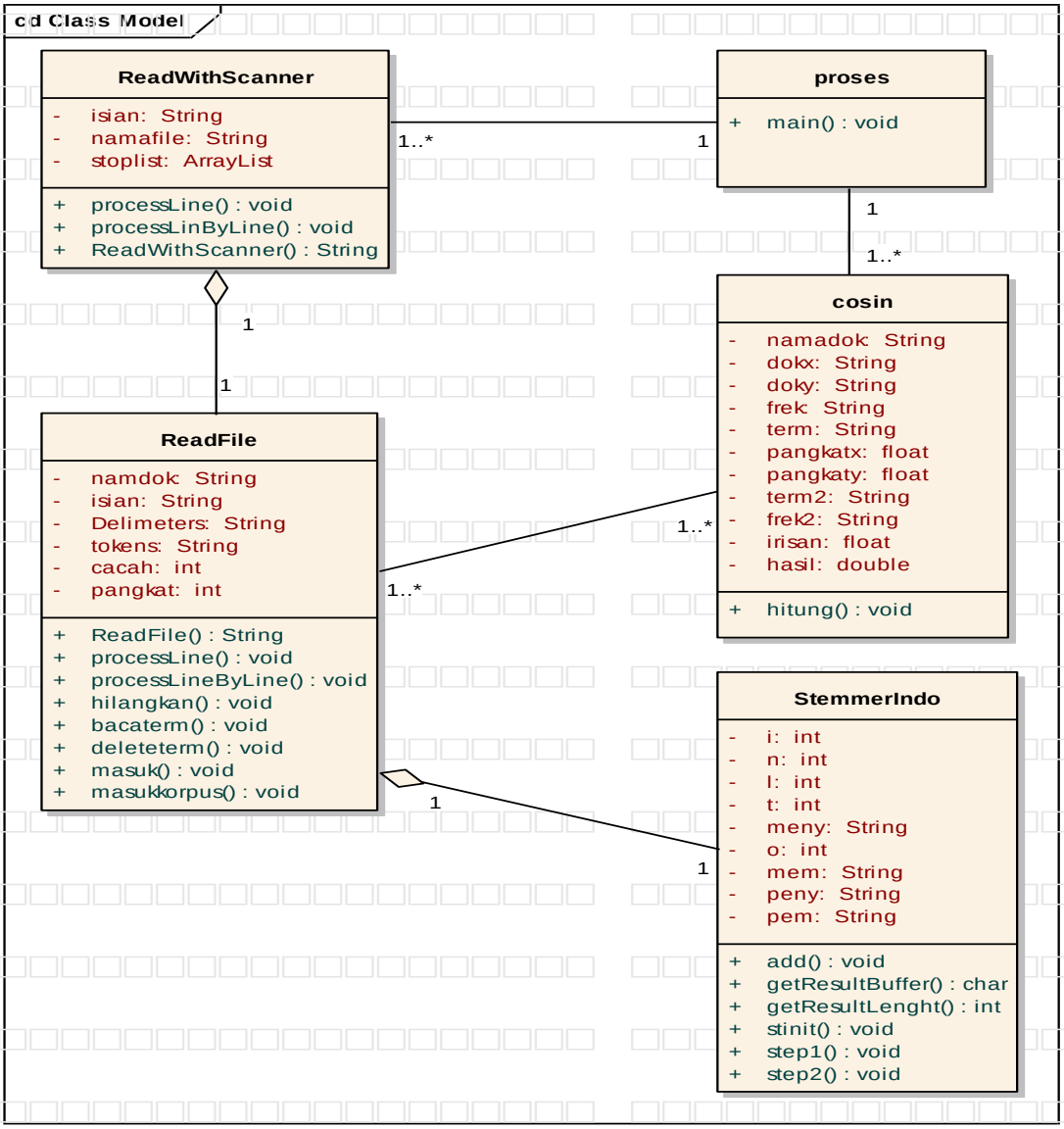
Tabel 3. 4 Tabel cosin.dbo

Nama Field	Type Field	Keterangan
dok1	varchar(150)	nama dokumen pertama yang akan dihitung similaritas dengan dokumen2
dok2	varchar(150)	nama dokumen kedua yang akan dihitung similaritas dengan dokumen1
similaritas	float	nilai similaritas hasil dari hitung similaritas

3.6. Rancangan Class Diagram Sistem Pencari Dokumen Teks

Rancangan class diagram untuk Sistem Pencari Dokumen teks terlihat pada gambar 3.6. Class Diagram dari class ReadWithScanner, class ReadFile, class cosin 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 cosin menggunakan class proses untuk menjalankan proses hitung cosin similaritas. Asosiasi yang terjadi bahwa *one* (satu) class proses berasosiasi *many* (banyak) dengan class cosin.

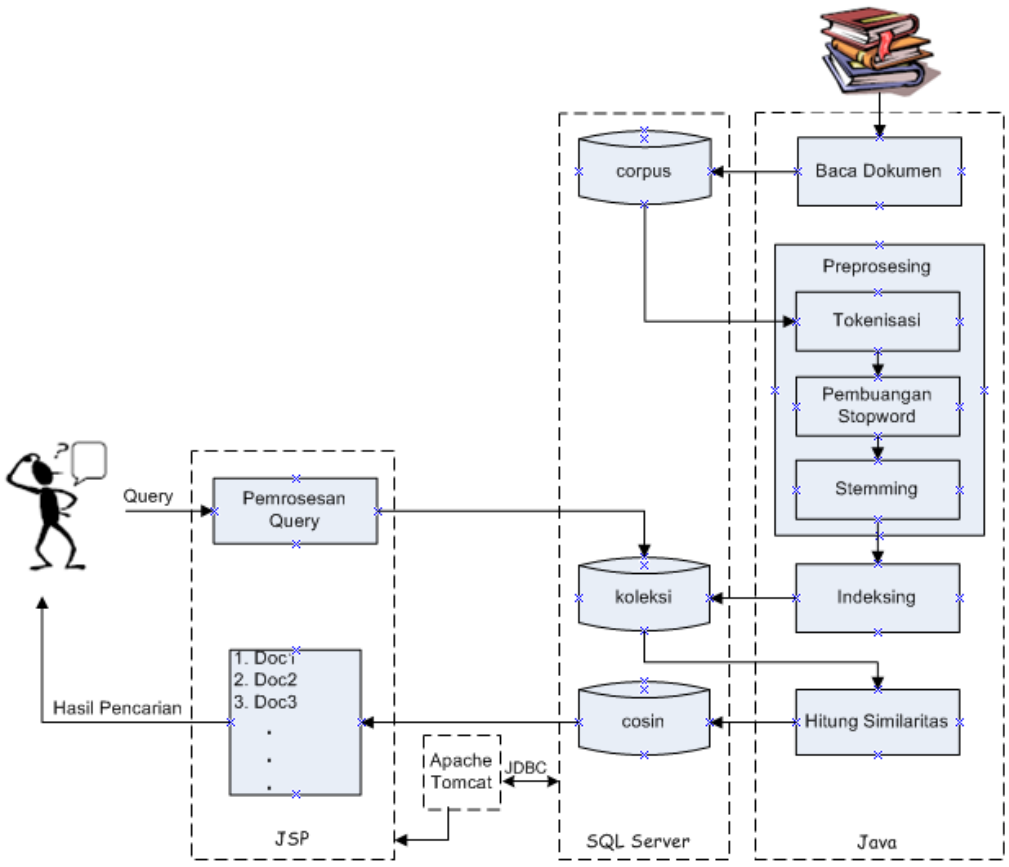


Gambar 3.6 Class Diagram Sistem Pencari Dokumen Teks

BAB IV
IMPLEMENTASI

4.1. 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 Prototipe Sistem Pencari Dokumen Teks. 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 4.1.



Gambar 4. 1 Implementasi Sistem Pencari Dokumen Teks

4.2. Implementasi Proses Baca Dokumen Teks

Proses persiapan data dimulai dari proses membaca dokumen abstrak skripsi dalam format teks. Proses baca dokumen teks abstrak skripsi dilakukan dengan cara dibaca file per file. Proses baca file ini dilakukan dengan perantara *method* ReadWithScanner(). Selanjutnya masing-masing file dokumen abstrak skripsi akan dilakukan baca teks dokumen dengan cara baris per baris. Proses baca dokumen teks ini dilakukan dengan menggunakan *method* ReadFile(). Dokumen abstrak skripsi yang telah dibaca akan disimpan dalam tabel corpus. Dengan format nama file disimpan dalam kolom *namadok* dan dokumen abstrak skripsi dalam format teks disimpan dalam kolom *isi*. Hasil proses baca dokumen teks skripsi disimpan dalam tabel corpus seperti terlihat pada gambar 4.2 :

	nomor	namadok	isi
1	8	D:\DataSkripsi\gis1.txt	PENGEMBANGAN SISTEM INFORMASI GEOGRAFIS DENGAN PE...
2	9	D:\DataSkripsi\gis2.txt	SISTEM INFORMASI GEOGRAFIS ANALISA KEPADATAN LALULIN...
3	10	D:\DataSkripsi\gis3.txt	SISTEM INFORMASI GEOGRAFIWILAYAH SUMUR PENGEBORAN...
4	11	D:\DataSkripsi\gis4.txt	SISTEM INFORMASI GEOGRAFIS AREA JANGKAUAN PEMANCAR...
5	12	D:\DataSkripsi\gis5.txt	RANCANG BANGUN SISTEM INFORMASI GEOGRAFIS PENYELA...
6	13	D:\DataSkripsi\multi1.txt	PERANGKAT LUNAK PENGENALAN BENTUK BANGUN UNTUK ...
7	14	D:\DataSkripsi\multi10.txt	PROGRAM BANTU PEMBELAJARAN MOTOR 4 LANGKAHBERB...
8	15	D:\DataSkripsi\multi11.txt	RANCANG BANGUN APLIKASI BRAIN GAME TENTANG SEJARA...
9	16	D:\DataSkripsi\multi12.txt	Rancang Bangun Aplikasi Pembelajaran Bahasa Inggris Untuk Anak ...
10	17	D:\DataSkripsi\multi13.txt	PROGRAM BANTU BELAJAR SENI MELIPAT KERTAS UNTUK P...
11	18	D:\DataSkripsi\multi14.txt	APLIKASI PEMBELAJARAN CHORD PIANO BERBASIS MULTIME...

Gambar 4. 2 Tabel Corpus

4.3. Implementasi Proses Preprosesing

4.3.1. Implementasi proses tokenisasi

Proses preprosesing dimulai dengan proses *tokenization* (tokenisasi) yaitu proses pemotongan string input berdasarkan tiap kata yang menyusunnya. Implementasi proses tokenisasi menggunakan *method* processLineByLine(). Proses tokenisasi dimulai dari 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 tokenisasi terlihat pada gambar 4. 3

```
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(),listword.
        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 4.3 Implementasi Proses Tokenisasi

4.3.2. Implementasi proses pembuangan stopword

Setelah proses tokenisasi akan dihasilkan dokumen abstrak skripsi dalam format teks dalam bentuk kata per kata. Proses akan dilanjutkan dengan proses *Stopword Removal* (pembuangan *stopword*). Proses pembuangan *stopword* adalah proses pembuangan *term* (kata) yang tidak memiliki arti atau tidak *relevan*.

Implementasi proses pembuangan *stopword* pada penelitian ini dilakukan dengan mencocokan *term* hasil proses *tokenisasi* dengan tabel *stopword* yang ada di database. Proses pembuangan *stopword* dilakukan dengan menggunakan *method* hilangkan(), perintah untuk proses pembuangan stopword adalah:

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

Term hasil proses tokenisasi jika cocok dengan tabel *stopword* maka akan dibuang dan tidak akan diikutkan pada proses *stemming*.

4.3.3. Implementasi proses stemming

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 (*stopword*). Proses penghilangan partikel/tanda baca diimplementasikan pada *method* processLine() dan proses menghilangkan *stopword* diimplementasikan pada *method* hilangkan(). Kedua method tersebut telah dilakukan pada proses tokenisasi dan proses pembuangan *stopword*.

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”, ”an” akan diimplementasikan pada *method* step1() dan *method* setto(). Pada *method* step1() jika ditemukan kata seperti ”lah”, ”pun”, ”ku”, ”mu”, ”nya”, ”kan”, ”an” akan memanggil *method* setto(). *Method* setto() akan melakukan proses jika term memiliki akhiran ”kan” akan dicek apakah *term* tersebut lebih dari 4 karakter, jika lebih maka akan

dilakukan penghilangan akhiran. Gambar 4.4 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;

      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(""); else
      if (ends("an")) setto("");
    }
```

Gambar 4.4 Implementasi Menghilangkan Kata Sandang dan Kepunyaan

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 4.5 adalah implementasi yang digunakan untuk menghilangkan awalan. Proses menghilangkan awalan dilakukan setelah proses proses menghilangkan kata sandang dan akhiran selesai dilakukan. Term hasil proses terlebih dahulu akan dicek apakah karakter lebih dari 3. Jika karakter kurang dari 3 maka proses tidak dilakukan. Proses menghilangkan awalan akan dilakukan jika karakter lebih dari 3.

Proses menghilangkan awalan yang diimplementasikan pada *method* step2() akan dilakukan proses jika dalam term yang dibaca ditemukan awalan seperti "meny", "peny" akan akan dipanggil ke *method* setto() bahwa awalan tersebut akan diganti dengan string ”s”. Jika dalam term yang dibaca

ditemukan awalan seperti ”mem”, ”pem” maka awalan akan diganti dengan string ”p”. Jika awalan ditemukan term ”me”, ”di”, ”ter”, ”ke”, ”ber”, ”per” maka awalan akan dihapus.

```
  private final void setinit(String s)
  {  String meny = "meny";
     String mem = "mem";
     String peny = "peny";
     String pem ="pem";
     int t=0;
     int l = s.length();
     int o = j+1;
      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 4.5 Implementasi Menghilangkan Awalan

4.4. Implementasi Proses Indexing

. Pada proses indeksing seluruh dokumen dalam koleksi disimpan dalam suatu file dengan format yang dapat menyimpan dan mengidentifikasi tiap-tiap term yang masuk ke dalam koleksi dokumen. Sehingga term yang tersimpan dalam tabel koleksi dapat teridentifikasi keberadaannya berdasarkan nama dokumen. Penelitian ini proses indeksing 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 yang telah dijelaskan pada implementasi bab 4.2 proses baca dokumen teks.

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 stopword 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 cacah 1, sedangkan apabila belum ada maka sisipkan record term baru pada tabel koleksi dengan kolom frek = 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.

4.5. Implementasi Proses Hitung Simmilaritas

Proses hitung simmilaritas adalah proses yang digunakan untuk menghitung nilai simmilaritas antar dokumen. 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 simmilaritas dokumen tertinggi dapat dianggap bahwa dokumen tersebut paling simmilar, yaitu bahwa dokumen tersebut memiliki banyak kesamaan.

Seperti terlihat pada gambar 3.6 class cosin 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 sampai 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
[KorpusSkripsi].[dbo].[koleksi] 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 [KorpusSkripsi].[dbo].[koleksi] group by
namadok order by namadok");
```

Dengan menggunakan persamaan (2.1) pada *method* hitung() pada class cosin digunakan sebagai perantara untuk menghitung nilai similaritas antar dokumen dari semua teks dokumen abstrak skripsi yang ada di dalam korpus. Class cosin menggunakan class proses untuk melakakukan proses hitung. Dengan perantara class proses, *method* hitung() pada class cosin melakukan proses hitung cosine similaritas.

Implementasi hitung cosine dimulai dengan koneksi ke database SQL Server untuk membaca 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 2.1

Hasil dari hitung similaritas akan disimpan dalam tabel cosin. Dokumen yang dihitung nilai similaritasnya disimpan dengan *field* dok1 dan dok2. Nilai similaritas dari dokumen disimpan dengan *field* similaritas. Implementasi SQL untuk menyimpan hasil hitung similaritas adalah :

```
String           SQLCommand4           ="INSERT           INTO
[KorpusSkripsi].[dbo].[cosin] ([dok1] ,[dok2] ,[similaritas])
VALUES  ('"+dokx+"','"+doky+"',"+hasil+")";
```

Tabel cosin menyimpan hasil implementasi dari *method* hitung() yang ada di class cosin dengan *atribute* dok1, dok2 dan similaritas. Hasil dari proses hitung cosin similaritas dapat dilihat pada gambar 4.7

	dok1	dok2	similaritas
1	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\gis2.txt	0,369250191184874
2	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\gis3.txt	0,358278798063606
3	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\gis4.txt	0,481751937243714
4	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\gis5.txt	0,288619821096139
5	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi1.txt	0,124269814547925
6	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi10.txt	0,184477722628018
7	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi11.txt	0,255624336899862
8	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi12.txt	0,204264327063324
9	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi13.txt	0,122931942641402
10	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi14.txt	0,252350211517174
11	D:\DataSkripsi\gis1.txt	D:\DataSkripsi\multi15.txt	0,245283035653856

Gambar 1.7 Tabel cosin

4.6. Implementasi Proses Temu Kembali

Query (kata kunci) yang diinputkan user terlebih dahulu akan dilakukan proses preprosesing. Yaitu dimulai dari proses menghilangkan partikel kata, penghilangan stopword dan proses stemming (pembentukan kata dasar). Implementasi proses preprosesing untuk *query,* sama dengan proses preprosesing pembentukkan klaster dokumen dari corpus. Term *query* yang telah melalui proses preprosesing akan dilakukan proses pencocokkan dengan tabel koleksi yang memuat *term* hasil indexing dari korpus. Proses pencocokkan term ini menggunakan operator *boolean* OR. Implementasi

untuk mencocokkan *query* dengan tabel koleksi dapat dilihat pada gambar 4.8.

Proses dimulai dengan mencocokkan *query* yang dimasukkan dengan *term* yang ada di tabel koleksi. Hasil mencocokkan *query* dihasilkan dokumen yang *match* dengan *query*. Perintah *query* dengan SQL untuk mencocokkan *query* dengan tabel koleksi adalah :

```
String teksqu =" (koleksi.namadok = corpus2.namadok) and
(koleksi.term like '%";
```

```
  String teks = request.getParameter("textfield2");
  String teksqu =" (koleksi.namadok = corpus2.namadok) and
(koleksi.term like '%";
  String hasil ="";
  String bantu="";

if (teks!=null && teks.length() > 0)
{
String delimiters = "\\s+|,\\s*|\\.\\s*";
String[] tokens = teks.split(delimiters);
StemmerIndo s = new StemmerIndo();
for (int i=0;i<tokens.length;i++)
{
for (int x=0;x<tokens[i].toString().length();x++)
s.add(tokens[i].toString().charAt(x));
s.stem();
String termin2 =s.toString();
hasil= hasil +teksqu +termin2+"%')";

teksqu =" or (koleksi.namadok = corpus2.namadok) and
(koleksi.term like '%";
}
String termin2 =s.toString();

hasil= hasil +teksqu +termin2+"%')";

teksqu =" or (koleksi.namadok = corpus2.namadok) and
(koleksi.term like '%";
}
```

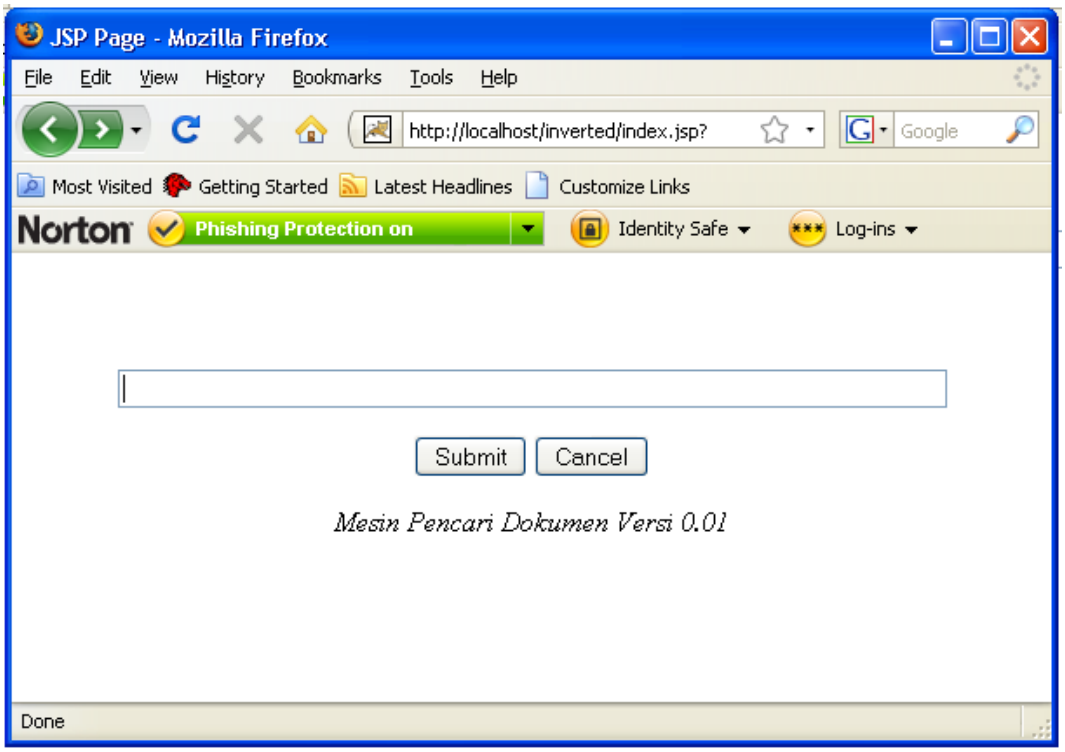
Gambar 4.8 Implementasi Mencocokkan Query (kata kunci)

Jika *query* yang dimasukkan lebih dari 1 (satu) *term*, digunakan operator boolean ”OR” diartikan sebagai ”atau”. Jadi jika ada 2 (dua) *term query* yang dimasukkan maka akan diproses apakah *term* yang ada di dalam dokumen *match* dengan *term* pertama atau *macth* dengan *term* kedua atau

match dengan kedua *term query*. Perintah *query* dengan SQL untuk operator *boolean* dalam implementasi adalah :

```
teksqu =" or (koleksi.namadok = corpus2.namadok) and
(koleksi.term like '%";
```

Setelah melalui proses mencocokkan *query* (kata kunci) dengan tabel koleksi, akan dihasilkan dokumen-dokumen yang mengandung *term* dari *query* yang dimasukkan. Gambar 4.9 adalah implementasi tampilan menu untuk *query* yang dimasukkan user dengan menggunakan Java Server Page (JSP).

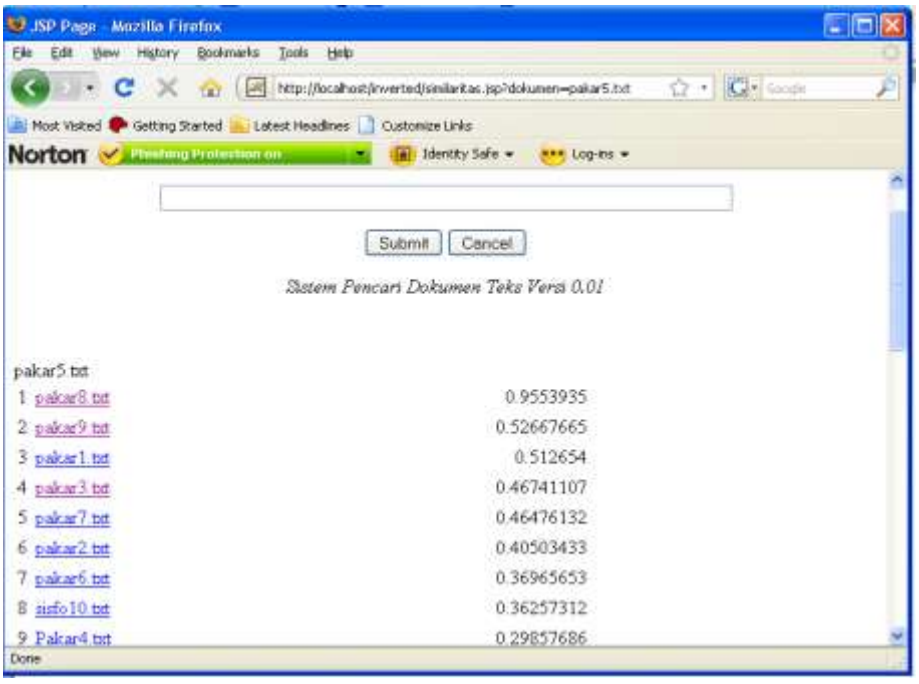


Gambar 4.9 Tampilan Menu untuk Memasukkan *Query*

Jika dimasukkan *query* maka aplikasi akan menampilan dokumen abstrak skripsi yang memiliki nilai similaritas terdekat dengan *query* yang dimasukkan. Pada implementasi aplikasi akan ditampilkan dokumen abstrak skripsi dengan nilai similaritas 10 terdekat dengan *query* yang dimasukkan. Gambar 4.10 adalah contoh hasil jika dimasukkan *query* "pakar" maka akan ditampilkan dokumen abstrak skripsi dengan nilai similaritas yang terdekat.

Gambar 4.10 Contoh Tampilan Hasil Query "pakar"

Aplikasi yang dibuat dilengkapi dengan menu "similaritas", yaitu menu yang memberikan informasi nilai similaritas dari masing-masing dokumen abstrak skripsi yang memiliki kedekatan dengan *query* yang telah dimasukkan. Gambar 4.11 adalah hasil tampilan jika dipilih menu similaritas.



Gambar 4.11 Contoh Tampilan Nilai Similaritas Dokumen Teks

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

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

1. Penelitian ini mempunyai proses yaitu proses membaca data dokumen teks, proses indeks term yang telah terbaca dan proses pembobotan term frekuensi yang dilanjutkan dengan hitung *cosine similaritas* dokumen.
2. Pada proses indeks term diperlukan proses pembentukkan kata dasar (stemming). Pada penelitian ini digunakan *stemming* Tala (2003) yang mengadopsi *English Porter Stemmer*.
3. Pembobotan term frekuensi dan *cosine similaritas* digunakan untuk menunjukkan kemiripan antar dokumen.
4. Sistem dapat menampilkan dokumen yang mempunyai kedekatan similaritas dengan rangking 10 tertinggi dari *query* yang diinputkan user .

5.2. Saran

Penelitian yang telah dilakukan mempunya beberapa kekurangan. Untuk perbaikan penelitian yang telah dilakukan, penulis menyarankan beberapa hal:

1. Sistem ini mempunyai kemampuan untuk melakukan pencarian dengan menggunakan metode boolean terbatas pada penggunaan operator OR. Untuk pengembangan lebih lanjut dapat dilengkapi dengan operator boolean AND, NOT serta operator boolean yang diperluas.
2. Selain itu untuk sistem ini dapat juga dikembangkan lebih lanjut dengan menerapkan metode stemming yang lain, atau modifikasi algoritma stemming Bahasa Indonesia sehingga dihasilkan kata dasar yang lebih baik dari penelitian yang telah dilakukan.

DAFTAR PUSTAKA

Djuandi, F., 2008, Jurus Bau Pemrograman SQL Server 2005, Elex Media Komputindo, Jakarta.

Foenadioen, 2008, Web Database menggunakan Java Server Page, Andi, Yogyakarta.

Pressman R, 2001, *Software Engineering,* Mc Graw Hill, USA.

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

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

Yuliana, 2009, Pengenalan JSP, http://lecturer.eepis-its.edu/~yuliana/ ProLanjut/ JSP/JSPdenganNetbeansversi6.pdf.