Prototipe Mesin Pencari Dokumen Teks

Herny Februariyanti, Eri Zuliarso dan Mardi Siswo Utomo
Fakultas Teknologi Informasi, Universitas Stikubank
Email : herny@unisbank.ac.id, ezuliarso@yahoo.com, mardiutomo@gmail.com

Abstrak

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 diinputkan 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.

Penelitian ini menggunakan algoritma indeks inverted untuk proses indeks kata (term), cosine similaritas untuk menghitung kesamaan kata dalam dokumen. Dokumen yang digunakan pada penelitian adalah dokumen teks abstrak skripsi mahasiswa Fakultas Teknologi Informasi Unisbank Semarang.
Algoritma yang dikembangkan diuji dengan menggunakan dokumen teks naskah abstrak skripsi Fakultas Teknologi Informasi Unisbank Semarang. Hasil uji menunjukan bahwa algoritma dapat digunakan untuk menghitung tingkat similaritas (kesamaan) dokumen berdasarkan kata kunci yang diinputkan oleh pengguna. Pemilihan kata kunci sangat mempengaruhi hasil pencarian dokumen teks.

Kata kunci : sistem temu kembali dokumen, similaritas, *indeks inverted*, dokumen teks

PENDAHULUAN

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 diinputkan 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

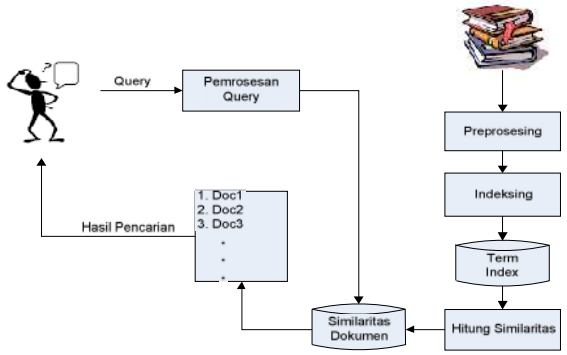
data pada koleksi dokumen yang disebut indeks untuk mempercepat proses pencarian.

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.

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 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 1. Proses Pencarian Dokumen Teks

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.

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.

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).

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.

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 :

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

Dimana :

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

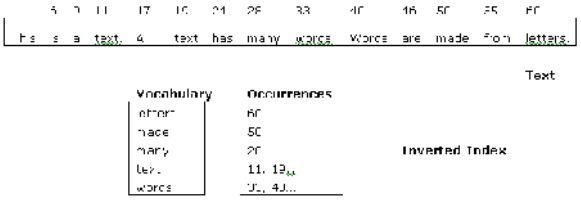
| X | adalah jumlah term yang ada di dokumen X

| Y | adalah jumlah term yang ada di dokumen Y

INDEKS INVERTED

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.



Gambar 2. Contoh Teks dan Inverted File-nya

Pada gambar 2 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.

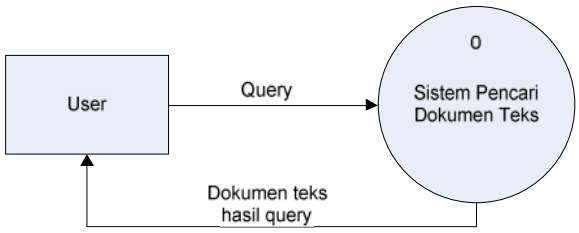
PERANCANGAN

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.

Perancangan Menggunakan DFD

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 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.

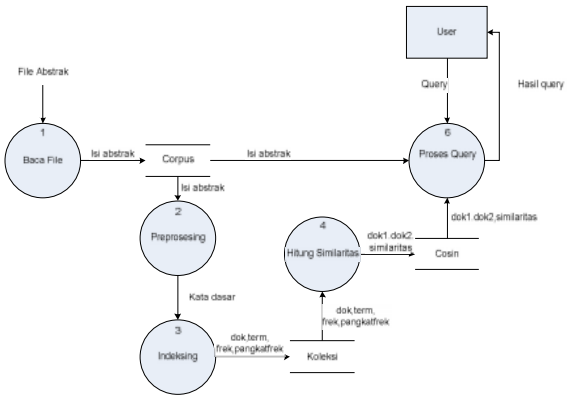


Gambar 3. Context Diagram Sistem Pencari Dokumen Teks

Dari context diagram lebih lanjut akan dijabarkan DFD level 1 sistem pencari dokumen teks seperti terlihat pada gambar 4.

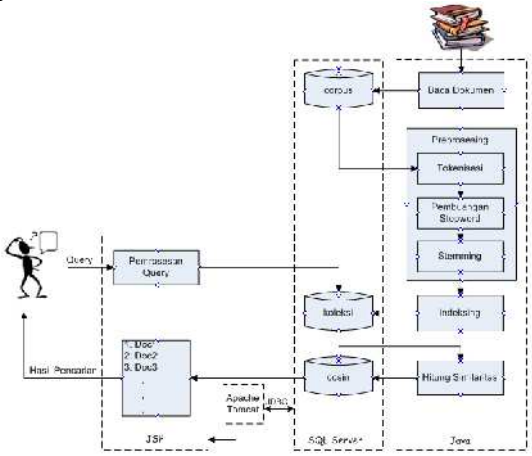
IMPLEMENTASI

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.



Gambar 4. DFD Level 1 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 5.



Gambar 5. Implementasi Sistem Pencari Dokumen Teks

Proses Baca Dokumen

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. Hasil proses baca data dapat dilihat pada tabel 1.

Tabel 1. Hasil Proses Baca Dokumen Teks

	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 LALU LIN...
3	10	D:\DataSkripsi\gis3.txt	SISTEM INFORMASI GEOGRAFI WILAYAH 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	[illegible] BANTU PEMBELAJARAN MUTU [illegible]...
8	15	D:\DataSkripsi\multi11.txt	RANCANG BANGUN APLIKASI BRAIN GAME TENTANG SEJARAH
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...

Proses Indeksing

Proses indeksing dilakukan setelah proses preprosesing. Proses preprosesing adalah proses dimana data yang telah disimpan dalam table corpus akan dilakukan proses tokenisasi, proses pembuangan stopword dan proses stemming. Dari preprosesing didapat data dalam bentuk kata dasar, kemudian akan dilakukan proses indeksing, yang akan menghasilkan tabel seperti dalam tabel 2.

Tabel 2. Tabel Hasil Proses Indeksing

	namadok	term	frek	pangkatfrek
1	gis1.txt	abstraksi	1	1
2	gis1.txt	ada	1	1
3	gis1.txt	akses	1	1
4	gis1.txt	ampil	1	1
5	gis1.txt	aplikasi	1	1
6	gis1.txt	arik	1	1
7	gis1.txt	bangun	1	1
8	gis1.txt	bantu	1	1
9	gis1.txt	basis	3	9
10	gis1.txt	berbasis	1	1

Term yang telah diindeks akan disimpan dalam tabel koleksi. Selanjutnya akan dilakukan proses hitung similaritas dengan menggunakan rumus (1). Dari hasil perhitungan cosine similaritas akan dihasilkan similaritas antara dokumen yang satu dengan dokumen yang lainnya. Pada penelitian ini implementasi dari hitung similaritas disimpan dalam tabel cosin seperti terlihat pada tabel 3.

Tabel 3. Tabel Hasil Proses Hitung Similaritas

	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

Implementasi Proses Temu Kembali

Setelah dihasilkan database yang menyimpan dokumen teks, selanjutnya adalah implementasi proses temu kempali. Yaitu proses mencocokkan query (kata kunci) yang dimasukkan user. *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.

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 :

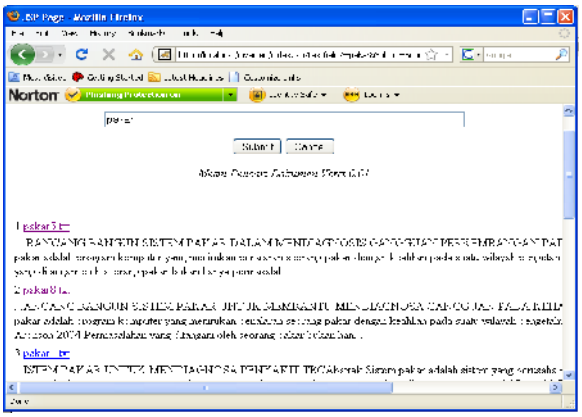
```
String teksqu =" (koleksi.namadok = corpus2.namadok) and (koleksi.term like '%'";
```

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 '%'";
```

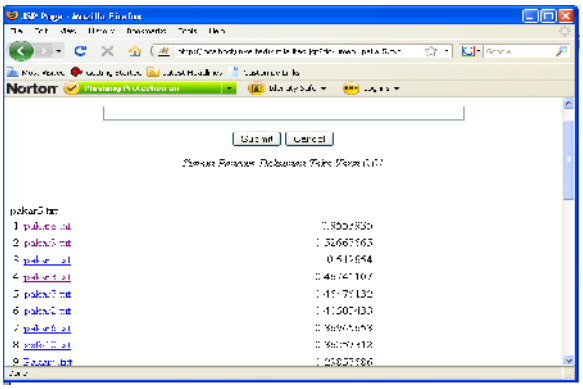
Setelah melalui proses mencocokkan *query* (kata kunci) dengan tabel koleksi, akan dihasilkan dokumen-dokumen yang mengandung *term* dari *query* yang dimasukkan. Jika dimasukkan *query* maka aplikasi akan menampilan dokumen abstrak skripsi yang memiliki nilai similaritas terdekat dengan *query* yang dimasukkan. Pada implementasi aplikasi akan ditampilkan dokumen abstrak skripsi

dengan nilai similaritas 10 terdekat dengan *query* yang dimasukkan. Gambar 9 adalah contoh hasil jika dimasukkan *query* "pakar" maka akan ditampilkan dokumen abstrak skripsi dengan nilai similaritas yang terdekat.



Gambar 9. 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 10. Tampilan Nilai Similaritas Dokumen Teks

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 proseos 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 .

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.