INFORMATION RETRIEVAL SYSTEM
DENGAN
METODE *LATENT SEMANTIC INDEXING*

TESIS

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Magister dari
Institut Teknologi Bandung

Oleh

HENDRA BUNYAMIN

NIM : 23502013

Program Studi Rekayasa Perangkat Lunak

INSTITUT TEKNOLOGI BANDUNG

2005

ABSTRAK

INFORMATION RETRIEVAL SYSTEM DENGAN METODE *LATENT SEMANTIC INDEXING*

Oleh

Hendra Bunyamin

NIM : 23502013

Information retrieval (IR) *system* adalah sistem yang secara otomatis melakukan pencarian atau penemuan kembali informasi yang relevan terhadap kebutuhan pengguna. Kebutuhan pengguna, diekspresikan dalam *query*, menjadi input bagi IR *system* dan selanjutnya IR *system* mencari dan menampilkan dokumen yang relevan dengan *query* tersebut.

Salah satu metode mencari atau menemukan kembali informasi yang relevan dengan *query* adalah dengan melihat kecocokan semantik *query* dengan koleksi dokumen. Metode dalam menemukan kecocokan semantik antara *query* dengan koleksi dokumen adalah metode *Latent Semantic Indexing*. Contoh dua kata yang mempunyai kecocokan semantik adalah '*purchase*' dan '*buy*'. Kedua kata tersebut mempunyai arti yang sama. Jadi informasi yang mempunyai arti yang sama dengan query juga dicari atau ditemukan kembali.

Pada tesis ini dilakukan studi mengenai evaluasi perbandingan IR *system* dengan menggunakan metode *Latent Semantic Indexing* (LSI) dengan IR *system* menggunakan metode lain.

Kata kunci: *information retrieval system, Latent Semantic Indexing*.

ABSTRACT

INFORMATION RETRIEVAL SYSTEM USING LATENT SEMANTIC INDEXING METHOD

By

Hendra Bunyamin

NIM : 23502013

Information retrieval (IR) system is a system, which is used to search and retrieve information relevant to the users' needs. IR system retrieves and displays documents that are relevant to the users' input (query).

One of the methods to retrieve information relevant to the query is how to match the query **semantically** with document collection. Latent Semantic Indexing (LSI) is a method to match the query **semantically** with document collection. For example, there is a query 'purchase'. 'Purchase' and 'buy' are two words that have semantic matching. So, LSI retrieves documents, which have both or either one of those words.

This thesis explores the comparison between the performance of LSI method and that of vector method. The performance is measured by non-interpolated average precision (NIAP).

Keywords: information retrieval system, Latent Semantic Indexing, non-interpolated average precision.

INFORMATION RETRIEVAL SYSTEM

DENGAN

METODE *LATENT SEMANTIC INDEXING*

Oleh

HENDRA BUNYAMIN

NIM : 23502013

Program Studi Rekayasa Perangkat Lunak

Institut Teknologi Bandung

Menyetujui

Dosen Pembimbing

Tanggal .....

Dosen Pembimbing

_____

(Dr. Ir. Rila Mandala, M.Eng.)

PEDOMAN PENGGUNAAN TESIS

Tesis S2 yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin pengarang dan harus disertai dengan kebiasaan ilmiah untuk menyebutkan sumbernya.

Memperbanyak atau menerbitkan sebagian atau seluruh tesis haruslah seizin Direktur Program Pascasarjana Institut Teknologi Bandung.

UCAPAN TERIMA KASIH/KATA PENGANTAR

Penulis sangat berterima kasih pada Dr. Ir. Rila Mandala, M.Eng. sebagai Pembimbing, atas segala saran, bimbingan dan nasehatnya selama penelitian berlangsung dan selama penulisan tesis ini.

Terima kasih yang sebesar-besarnya juga disampaikan kepada

1. Mama, dan *my Brother* yang *always* / selalu / senantiasa mendukung penulis dalam pembuatan tesis ini.
2. Para Dosen Penguji, yaitu Bapak Dr. Oerip Santoso, M.Sc. dan Ibu Dra. Harlili, M.Sc. yang sudah memberikan masukan yang sangat berharga untuk tesis ini.
3. Ibu Tita, Ibu Susi, Ibu Titi, Pak Ade, dan Pak Kandayat yang sudah membantu penulis dalam urusan administrasi perkuliahan dan banyak urusan lainnya.
4. Ibu Yenni M. Djajalaksana dan para dosen Universitas Kristen Maranatha, yaitu Risal, Andi, Saron, Radiant, Djoni, Doro Edi, Elisabet, Indra, Peter Kim, dan banyak dosen lain yang tidak dapat disebutkan semua. Terima kasih atas perhatiannya, semangat, dan *friendship*.
5. Rekan RPL angkatan 2002, yaitu Robinhood, Jaw Thong, Arman Rahman, Jasman Pardede, Bambang Pramono, Rimba, Megah Mulya, dan kawan-kawan. Terima kasih atas kebersamaannya selama studi S2.
6. Bapak Dr. Ing. Farid Wazdi, sebagai dosen wali. Terima kasih atas konsep-konsep yang diajarkan sehingga konsep-konsep tersebut dapat membangun sebuah struktur bangunan yang disebut *knowledge*.
7. Para dosen jurusan Teknik Informatika yang sudah mengajarkan konsep-konsep yang sangat berharga.

Hendra Bunyamin

Institut Teknologi Bandung,
Januari 2005

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR DAN ILUSTRASI

Bab I Pendahuluan

I.1 Latar Belakang

Perkembangan pengetahuan dan teknologi yang begitu cepat membuat manusia harus terus belajar. Belajar akan lebih mudah apabila akses terhadap informasi mudah diperoleh. Semakin canggihnya teknologi di bidang komputasi dan telekomunikasi pada masa kini, membuat informasi dapat dengan mudah didapatkan oleh banyak orang. Kemudahan ini menyebabkan informasi menjadi semakin banyak dan beragam. Informasi dapat berupa dokumen, berita, surat, cerita, laporan penelitian, data keuangan, dan lain-lain.

Seiring dengan perkembangan informasi, banyak pihak menyadari bahwa masalah utama telah bergeser dari cara mengakses informasi menjadi memilih informasi yang berguna secara selektif. Usaha untuk memilih informasi ternyata lebih besar dari sekedar mendapatkan akses terhadap informasi. Pemilihan informasi ini tidak mungkin dilakukan secara manual karena kumpulan informasi yang sangat besar dan terus bertambah besar.

Suatu sistem otomatis diperlukan untuk membantu pengguna dalam menemukan informasi. *Information retrieval* (IR) *system* adalah sistem yang digunakan untuk menemukan informasi yang relevan terhadap kebutuhan penggunanya secara otomatis dari suatu koleksi informasi.

I.2 Rumusan Masalah

IR *system* bertujuan untuk mendapatkan atau menemukan kembali informasi yang sesuai atau relevan dengan kebutuhan informasi pengguna secara otomatis. IR *system* yang ideal adalah IR *system* yang

(1) Menemukan seluruh informasi yang relevan
(2) Menemukan hanya informasi yang relevan saja dan tidak menemukan informasi yang tidak relevan.

Kedua hal di atas menunjukkan kualitas atau performansi dari suatu IR *system*. Informasi yang dibutuhkan pengguna diekspresikan dalam *query*. *Query* dapat berupa kata atau kalimat. Dengan *query* sebagai input, IR *system* melakukan pencarian informasi di dalam koleksi informasi untuk mencari informasi yang relevan dengan *query* (3).

Pencarian informasi dilakukan dengan mencari informasi yang memuat *query*. Informasi yang relevan dapat diperoleh dengan menemukan informasi yang

(1) Memuat kata atau kalimat yang **sama** dengan query atau
(2) Memuat kata atau kalimat yang **bermakna sama** dengan query.

Sebagai contoh, terdapat *query* satu kata yaitu "pintar". Pada point 1, informasi yang memuat kata "pandai" atau "cerdas" dinilai tidak relevan karena informasi yang relevan adalah informasi yang memuat kata "pintar". Sedangkan pada point 2, informasi yang memuat kata "pandai" atau "cerdas" dinilai relevan karena "pandai" atau "cerdas" bermakna sama dengan "pintar".

Makna kata sama dapat ditinjau dari dua istilah, yaitu sinonim dan polisemi (8). Sinonim adalah istilah untuk kata yang bermakna sama. Contoh, kata "pintar" merupakan sinonim untuk "pandai" karena "pintar" dan "pandai" bermakna sama. Sedangkan polisemi adalah istilah untuk kata yang sama namun maknanya berbeda. Contoh, kata "membajak" dalam "membajak sawah" dan "membajak pesawat" merupakan polisemi karena kata "membajak" di kedua frase sama namun mempunyai arti yang berbeda.

Salah satu metode pencarian informasi dengan melihat kecocokan makna antara query dengan informasi adalah metode *Latent Semantic Indexing* (LSI). Metode *Latent Semantic Indexing* diajukan untuk memperbaiki performansi IR *system* dalam mencari informasi yang relevan dengan *query*. Informasi yang relevan bukan hanya sama kata namun makna kata juga.

I.3 Tujuan Penelitian

Tujuan yang ingin dicapai melaui penelitian ini adalah:

(1) Mempelajari metode *Latent Semantic Indexing* dan proses pembangunan IR *system*.

(2) Merancang dan mengimplementasikan metode *Latent Semantic Indexing* dalam IR *system*.

(3) Melakukan pengujian IR *system* metode *Latent Semantic Indexing* dengan menggunakan koleksi pengujian (*test collection*).

I.4 Batasan Masalah

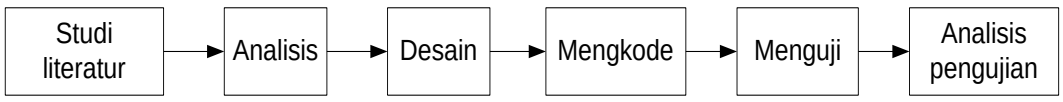
Batasan masalah dalam tesis ini adalah

(1) Koleksi dokumen yang digunakan untuk pengujian berupa dokumen tekstual tanpa format. Hal ini dimaksudkan untuk menghilangkan kebutuhan untuk mempelajari format dokumen seperti *Microsoft Word Document Format*, *Adobe Portable Document Format*, dan lain-lain.

(2) Bahasa yang digunakan dalam koleksi pengujian adalah bahasa Inggris saja.

I.5 Metoda Penelitian

Model proses dalam pengerjaan proyek ini menggunakan model sekuensial linier (Gambar I.1). Model ini sering juga disebut sebagai "daur hidup klasik" ("*classic life cycle*") atau model waterfall (16).



Gambar I-1 Model Sekuensial Linier

Aktivitas-aktivitas dalam model sekuensial sebagai berikut:

(1) Studi pustaka.

Studi pustaka adalah kegiatan mengumpulkan informasi dari pustaka-pustaka untuk menggali konsep-konsep dalam pengembangan perangkat lunak, khususnya perangkat lunak IR dengan metode LSI (9).

(2) Analisis perangkat lunak.

Analisa adalah proses memecah gambaran besar perangkat lunak menjadi gambaran yang lebih detil dan terfokus. Untuk memahami perangkat lunak yang dibuat, *software engineer* (analisis) harus memahami *domain* informasi untuk perangkat lunak.

(3) Desain perangkat lunak

Desain perangkat lunak merupakan proses yang mengutamakan 4 (empat) karakteristik program: struktur data, arsitektur perangkat lunak, representasi antarmuka, dan detil algoritma. Proses desain menerjemahkan *requirements* menjadi representasi perangkat lunak yang dapat diukur kesesuaiannya dengan *requirements* (*quality of conformance*) sebelum proses pengkodean dimulai.

(4) Pengkodean

Pengkodean adalah proses menerjemahkan desain menjadi bentuk kode program.

(5) Menguji

Pengujian adalah menguji program setelah kode program dihasilkan. Proses testing berfokus pada logika di dalam program, kepastian bahwa semua *statements* sudah diuji, dan pada fungsionalitas eksternal yaitu masukan sudah sesuai dengan hasil yang diinginkan.

(6) Analisis terhadap hasil pengujian

Analisis hasil pengujian adalah membandingkan performansi hasil pencarian IR *system* dengan metode LSI dengan metode lain..

I.6 Sistematika Pembahasan

Pembahasan tesis ini terdiri dari enam buah bab dengan perincian sebagai berikut:

Bab I. Pendahuluan, menguraikan tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, serta sistematika pembahasan tesis.

Bab II. *Information Retrieval System* berisi penjelasan mengenai IR *system*. Pembahasan meliputi gambaran IR *system* beserta bagian-bagiannya, model IR system.

Bab III. Teori Latent Semantic Indexing, menguraikan teori yang mendasari teknik *Latent Semantic Indexing*.

Bab IV, Analisis Perancangan Perangkat Lunak, menguraikan analisis perancangan perangkat lunak (deskripsi umum, diagram-diagram yang menggunakan notasi UML).

Bab V. Implementasi Perangkat Lunak, menguraikan aspek implementasi yang meliputi lingkungan implementasi pengembangan (perangkat keras dan perangkat lunak) dan implementasi pemrograman.

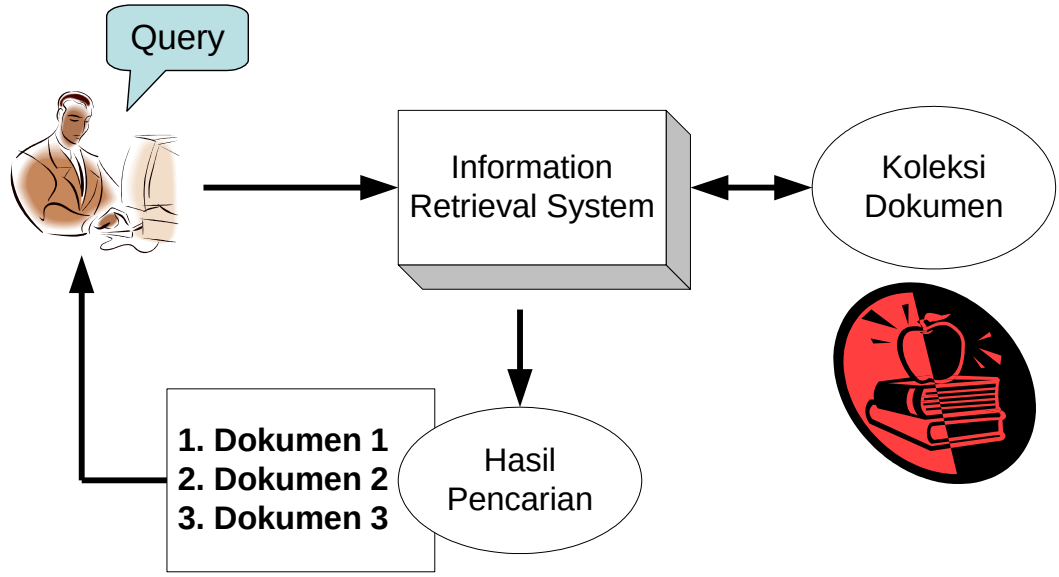
Bab VI. Evaluasi Perangkat Lunak, menguraikan aspek pengujian perangkat lunak yaitu tujuan pengujian, lingkungan pengujian, identifikasi dan rencana pengujian, perbandingan dengan metode lain dan kesimpulan hasil pengujian.

Bab VII. Kesimpulan dan saran, berisi kesimpulan dari pengembangan perangkat lunak dan saran bagi pengembangan perangkat lunak lebih lanjut.

Bab II *Information Retrieval System*

II.1 Perkenalan *Information Retrieval System*

Information retrieval (IR) *system* digunakan untuk menemukan kembali (*retrieve*) informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis.

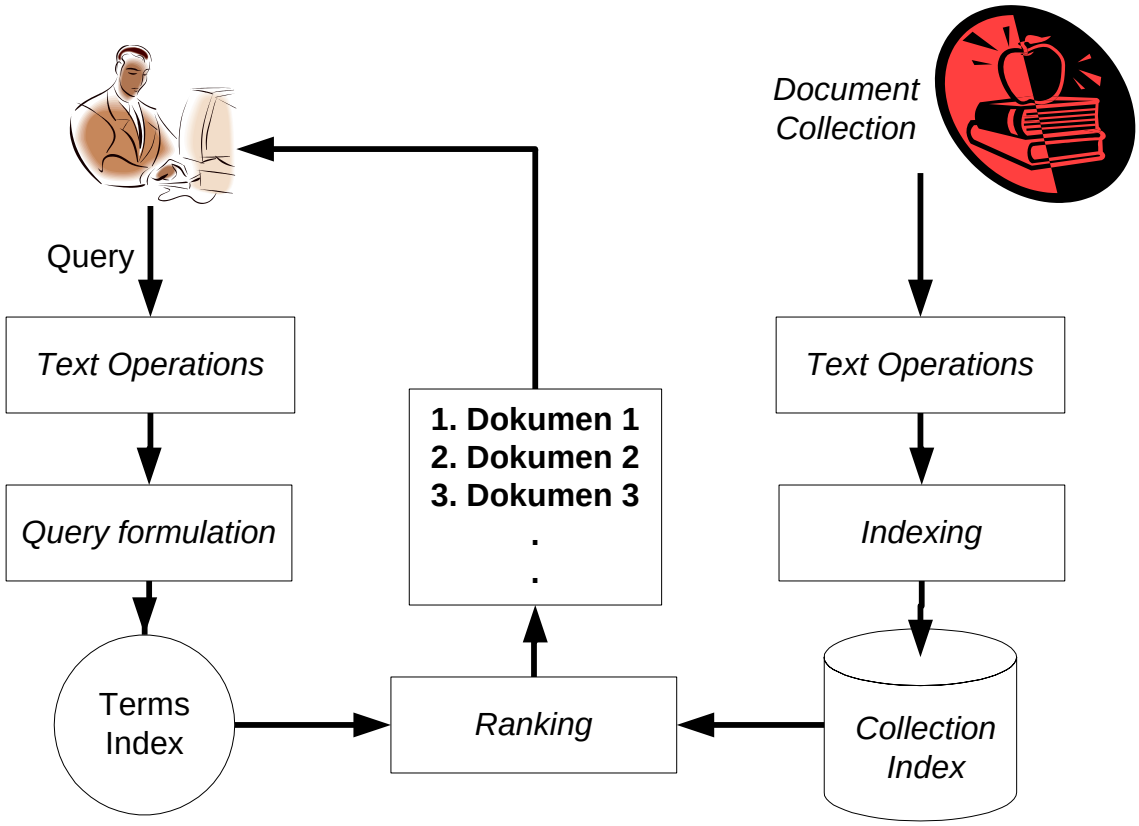


Gambar II-1 Ilustrasi *information retrieval system*

Salah satu aplikasi umum dari IR *system* adalah *search engine* atau mesin pencarian yang terdapat pada jaringan internet. Pengguna dapat mencari halaman-halaman web yang dibutuhkannya melalui *search engine*. Contoh lain dari IR *system* adalah sistem informasi perpustakaan.

IR *system* 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 IR *system* dengan sistem basis data. Dokumen adalah contoh informasi yang tidak terstruktur. Isi dari suatu dokumen sangat tergantung pada pembuat dokumen tersebut.

Sebagai suatu sistem, IR *system* memiliki beberapa bagian yang membangun sistem secara keseluruhan. Gambaran bagian-bagian yang terdapat pada suatu IR *system* digambarkan pada Gambar II.2



Gambar II-2 Bagian-bagian *information retrieval system*

Gambar II.2 memperlihatkan bahwa terdapat dua buah alur operasi pada IR *system*. Alur pertama dimulai dari koleksi dokumen dan alur kedua dimulai dari *query* pengguna. Alur pertama yaitu pemrosesan terhadap koleksi dokumen menjadi basis data indeks tidak tergantung pada alur kedua. Sedangkan alur kedua tergantung dari keberadaan basis data indeks yang dihasilkan pada alur pertama.

Bagian-bagian dari IR system menurut gambar II.2 meliputi (12):

(1) *Text Operations* (operasi terhadap teks) yang meliputi pemilihan kata-kata dalam query maupun dokumen (*term selection*) dalam pentransformasian dokumen atau query menjadi *term index* (indeks dari kata-kata).

(2) *Query formulation* (formulasi terhadap *query*) yaitu memberi bobot pada indeks kata-kata *query*.
(3) *Ranking* (perangkingan), mencari dokumen-dokumen yang relevan terhadap *query* dan mengurutkan dokumen tersebut berdasarkan kesesuaiannya dengan *query*.
(4) *Indexing* (pengindeksan), membangun basis data indeks dari koleksi dokumen. Dilakukan terlebih dahulu sebelum pencarian dokumen dilakukan.

IR *system* menerima query dari pengguna, kemudian melakukan perangkingan terhadap dokumen pada koleksi berdasarkan kesesuaiannya dengan *query*. Hasil perangkingan yang diberikan kepada pengguna merupakan dokumen yang menurut sistem relevan dengan *query*. Namun relevansi dokumen terhadap suatu *query* merupakan penilaian pengguna yang subjektif dan dipengaruhi banyak faktor seperti topik, pewaktuan, sumber informasi maupun tujuan pengguna.

Model IR *system* menentukan detil IR *system* yaitu meliputi representasi dokumen maupun *query*, fungsi pencarian (*retrieval function*) dan notasi kesesuaian (*relevance notation*) dokumen terhadap *query*.

Salah satu model IR *system* yang paling awal adalah model *boolean*. Model *boolean* merepresentasikan dokumen sebagai suatu himpunan kata-kunci (*set of keywords*). Sedangkan *query* direpresentasikan sebagai ekspresi *boolean*. *Query* dalam ekspresi *boolean* merupakan kumpulan kata kunci yang saling dihubungkan melalui operator *boolean* seperti *AND*, *OR*, dan *NOT* serta menggunakan tanda kurung untuk menentukan *scope* operator. Hasil pencarian dokumen dari model *boolean* adalah himpunan dokumen yang relevan.

Kekurangan dari model *boolean* ini antara lain:
(1) Hasil pencarian dokumen berupa himpunan, sehingga tidak dapat dikenali dokumen-dokumen yang paling relevan atau agak relevan (*partial match*).

(2) *Query* dalam ekspresi *boolean* dapat menyulitkan pengguna yang tidak mengerti tentang ekspresi *boolean*.

Kekurangan dari model *boolean* diperbaiki oleh model ruang vektor yang mampu menghasilkan dokumen-dokumen terurut berdasarkan kesesuaian dengan *query*. Selain itu, pada model ruang vektor, *query* dapat berupa sekumpulan kata-kata dari pengguna dalam ekspresi bebas.

II.2 Model Ruang Vektor

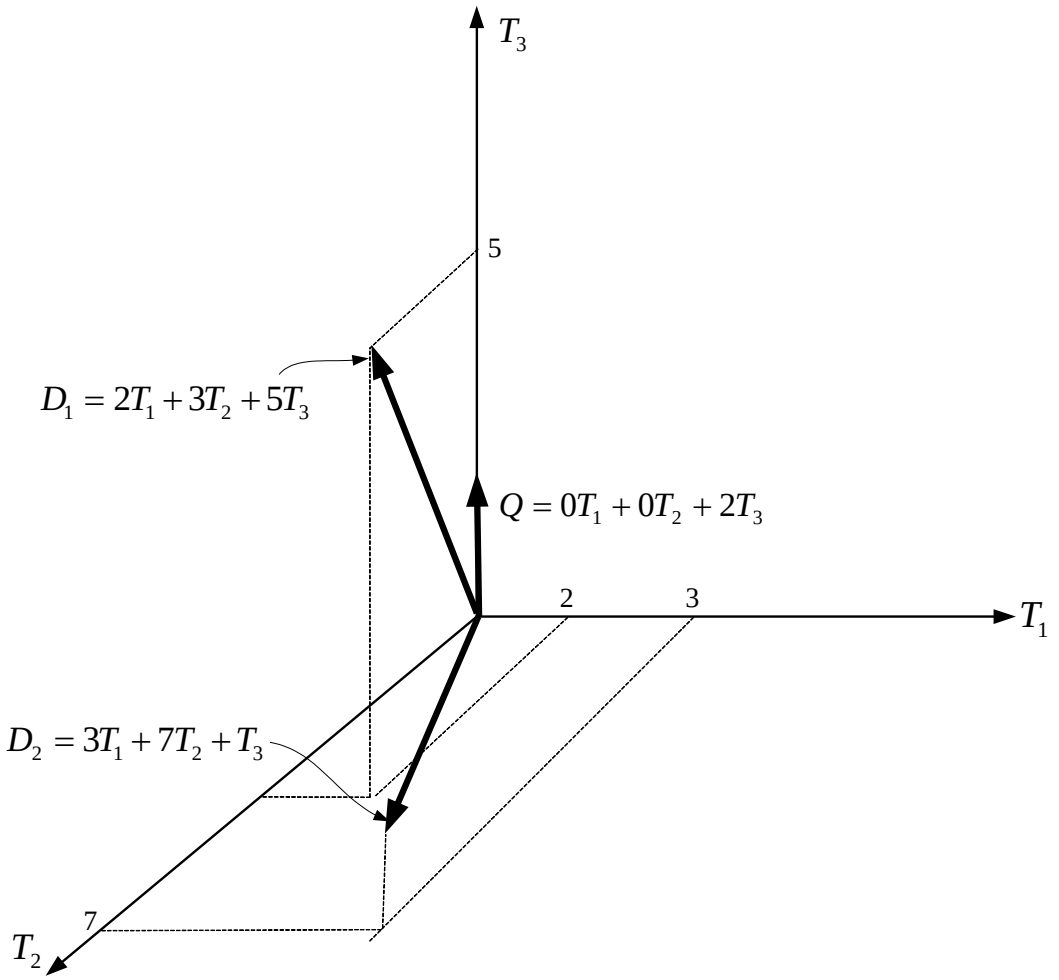
Misalkan terdapat sejumlah n kata yang berbeda sebagai kamus kata (*vocabulary*) atau indeks kata (*terms index*). Kata-kata ini akan membentuk ruang vektor yang memiliki dimensi sebesar n. Setiap kata i dalam dokumen atau *query* diberikan bobot sebesar w_i. Baik dokumen maupun *query* direpresentasikan sebagai vektor berdimensi n.

Sebagai contoh terdapat 3 buah kata (T_1, T_2 dan T_3), 2 buah dokumen (D_1 dan D_2) serta sebuah *query* Q. Masing-masing bernilai:

$$D_1 = 2T_1 + 3T_2 + 5T_3;\ D_2 = 3T_1 + 7T_2 + 0T_3;\ Q = 0T_1 + 0T_2 + 2T_3$$

Maka representasi grafis dari ketiga vektor ini adalah seperti pada gambar II.3

Koleksi dokumen direpresentasi pula dalam ruang vektor sebagai matriks kata-dokumen (*terms-documents matrix*). Nilai dari elemen matriks w_{ij} adalah bobot kata i dalam dokumen j.



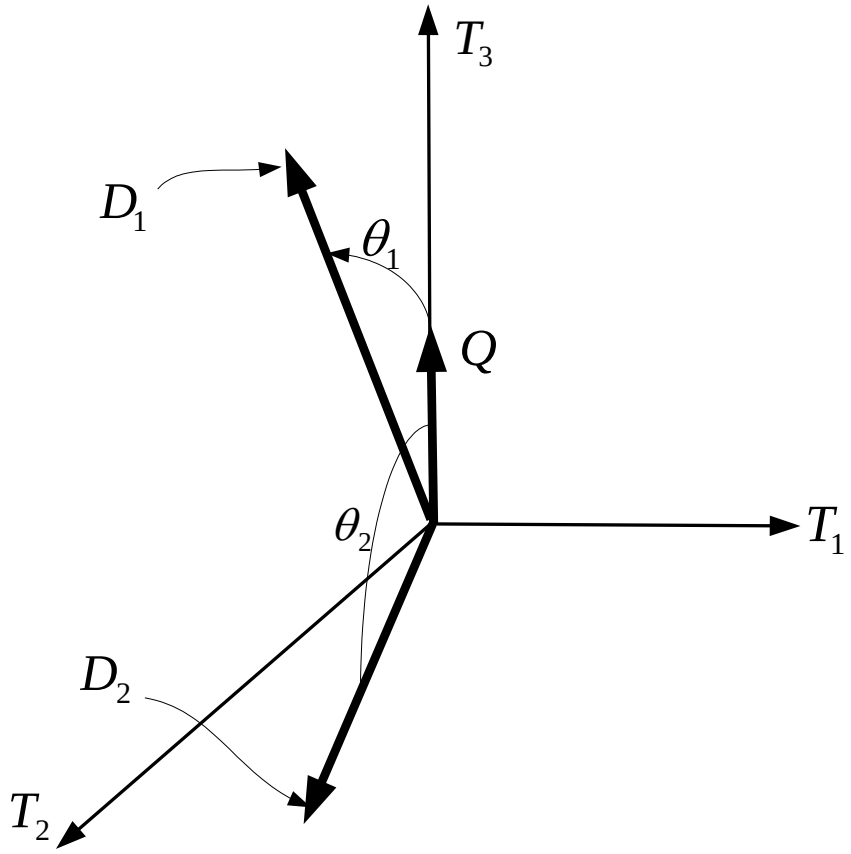
Gambar II-3 Contoh vektor-vektor D_1, D_2, D_3 dan Q

Misalkan terdapat sekumpulan kata T sejumlah m, yaitu $T = (T_1, T_2, \ldots, T_m)$ dan sekumpulan dokumen D sejumlah n, yaitu $D = (D_1, D_2, \ldots, D_n)$ serta w_{ij} adalah bobot kata i pada dokumen j. Maka gambar II.4 adalah representasi matriks kata-dokumen

$$\begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & \cdots & D_n \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \end{array}$$

Gambar II-4 Representasi matriks kata-dokumen

Penentuan relevansi dokumen dengan *query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara vektor dokumen dengan vektor *query*. Semakin "sama" suatu vektor dokumen dengan vektor *query* maka dokumen dapat dipandang semakin relevan dengan *query*. Salah satu pengukuran kesesuaian yang baik adalah dengan memperhatikan perbedaan arah (*direction difference*) dari kedua vektor tersebut. Perbedaan arah kedua vektor dalam geometri dapat dianggap sebagai sudut yang terbentuk oleh kedua vektor. Gambar II.5 mengilustrasikan kesamaan antara dokumen D_1 dan D_2 dengan *query* Q. Sudut θ_1 menggambarkan kesamaan dokumen D_1 dengan *query* sedangkan sudut θ_2 menggambarkan kesamaan dokumen D_2 dengan *query*.



Gambar II-5 Representasi grafis sudut vektor dokumen dan *query*

Jika Q adalah vektor *query* dan D adalah vektor dokumen, yang merupakan dua buah vektor dalam ruang berdimensi-n, dan θ adalah sudut yang dibentuk oleh kedua vektor tersebut. Maka

$$Q \bullet D = |Q|\,|D| \cos\theta \quad \text{.....(II.1)}$$

dengan $Q \bullet D$ adalah hasil perkalian titik (*dot product*) kedua vektor, sedangkan

$$|D| = \sqrt{\sum_{i=1}^{n} D_i^2} \text{ dan } |Q| = \sqrt{\sum_{i=1}^{n} Q_i^2} \quad \text{.....(II.2)}$$

merupakan *norm* atau panjang vektor di dalam ruang berdimensi-n.

Perhitungan kesamaan (*Similarity*) kedua vektor adalah sebagai berikut

$$Sim(Q, D) = \cos(Q, D) = \frac{Q \bullet D}{|Q|\,|D|} = \frac{1}{|Q|\,|D|} \sum_{i=1}^{n} Q_i \times D_i \quad \text{.....(II.3)}$$

dengan $Q_i \times D_i$ adalah perkalian antara Q_i dan D_i.

Metode pengukuran kesesuaian ini memiliki beberapa keuntungan, yaitu adanya normalisasi terhadap panjang dokumen. Hal ini memperkecil pengaruh panjang dokumen. Panjang kedua vektor digunakan sebagai faktor normalisasi. Hal ini diperlukan karena dokumen yang panjang cenderung mendapatkan nilai yang besar dibandingkan dengan dokumen yang lebih pendek.

Proses perangkingan dari dokumen dapat dianggap sebagai proses pemilihan (vektor) dokumen yang dekat dengan (vektor) *query*, kedekatan ini diindikasikan dengan sudut yang dibentuk. Nilai cosinus yang cenderung besar mengindikasikan bahwa dokumen cenderung sesuai *query*. Nilai cosinus sama dengan 1 mengindikasikan bahwa dokumen sesuai dengan *query*.

II.3 Pembobotan Kata

Bagian sebelumnya membahas mengenai metode pengukuran kesesuaian antara dokumen dan *query* dalam model ruang vektor. Dokumen maupun *query* direpresentasikan sebagai vektor berdimensi-n. Bagian ini akan membahas mengenai nilai dari vektor atau bobot kata dalam dokumen.

Salah satu cara untuk memberi bobot terhadap suatu kata adalah memberikan nilai jumlah kemunculan suatu kata (*term frequency*) sebagai bobot (11). Semakin besar kemunculan suatu kata dalam dokumen akan memberikan nilai kesesuaian yang semakin besar.

Faktor lain yang diperhatikan dalam pemberian bobot adalah kejarangmunculan kata (*term scarcity*) dalam koleksi. Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon terms*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebalikan secara proporsional dengan urutannya (7).

Faktor terakhirnya adalah faktor normalisasi terhadap panjang dokumen. Dokumen dalam koleksi dokumen memiliki karakteristik panjang yang beragam. Ketimpangan terjadi karena dokumen yang panjang akan cenderung mempunyai frekuensi kemunculan kata yang besar. Sehingga untuk mengurangi ketimpangan tersebut diperlukan faktor normalisasi dalam pembobotan.

Perbedaan antara normalisasi pada pembobotan dan perangkingan adalah normalisasi pada pembobotan dilakukan terhadap suatu kata dalam suatu dokumen sedangkan pada perangkingan dilakukan terhadap suatu dokumen dalam koleksi dokumen.

Pembobotan yang dianggap paling baik (14) adalah menggunakan persamaan

$$w_i = \frac{\log(tf_i) + 1.0}{\sqrt{\sum_{j=1}^{t} [\log(tf_j) + 1.0]^2}} \quad \text{.....(II.4)}$$

untuk pembobotan kata i (w_i) pada dokumen dan menggunakan persamaan

$$q_i = \frac{(\log(tf_i) + 1.0) + \log(N/n_i)}{\sqrt{\sum_{j=1}^{t} [(\log(tf_j) + 1.0) \times (\log(N/n_j))]^2}} \quad \text{.....(II.5)}$$

untuk pembobotan kata i (q_i) pada *query*. Dengan tf_i adalah frekuensi kemunculan kata i, n_i banyak dokumen yang mengandung kata i dan N jumlah dokumen dalam koleksi.

Bab III Metode Latent Semantic Indexing

III.1 Metode *Latent Semantic Indexing*

Mendapatkan hasil pencarian yang sesuai dengan kebutuhan dalam suatu koleksi dokumen yang besar merupakan hal sulit. Usaha pengguna secara manual untuk memilah-milah dokumen yang sesuai dengan kebutuhannya ternyata sangat besar. Hasil pencarian merupakan sejumlah dokumen yang relevan menurut sistem, namun relevansi merupakan hal yang subjektif.

Pada umumnya, dokumen dikatakan relevan dengan query apabila dokumen

(1) Memuat kata atau kalimat yang sama dengan query atau
(2) Memuat kata atau kalimat yang bermakna sama dengan query.

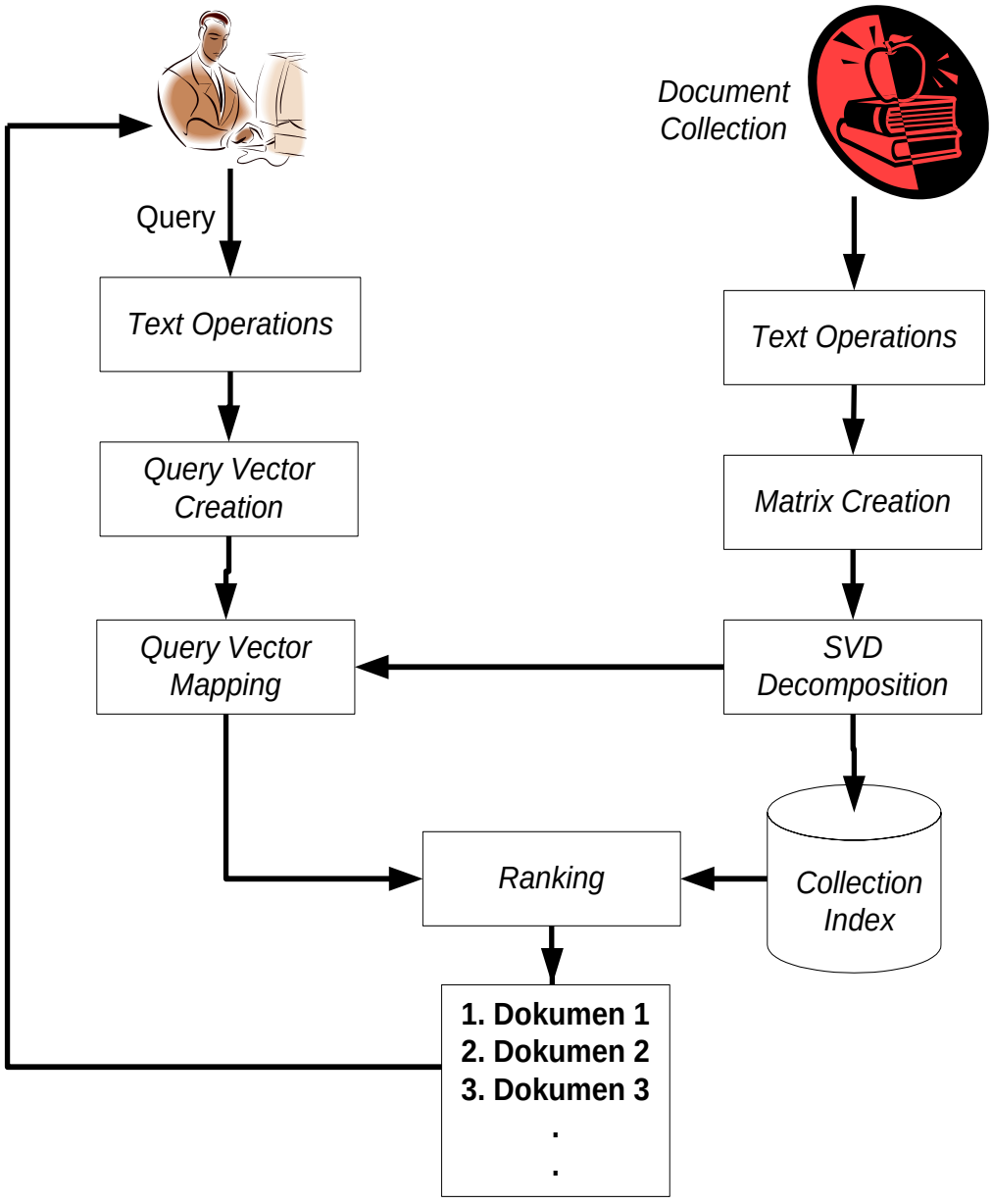
Sebagai contoh, terdapat query satu kata yaitu "sulit". Pada point 1, informasi yang memuat kata "susah" atau "sukar" dinilai tidak relevan karena informasi yang relevan adalah informasi yang memuat kata "sulit". Sedangkan pada point 2, informasi yang memuat kata "susah" atau "sukar" dinilai relevan karena "susah" atau "sukar" bermakna sama dengan "sulit".

Makna kata dapat ditinjau dari dua istilah, yaitu sinonim dan polisemi (8). Sinonim adalah istilah untuk kata yang bermakna sama. Contoh, kata "sulit" merupakan sinonim untuk "sukar" karena "sulit" dan "sukar" bermakna sama. Sedangkan polisemi adalah istilah untuk kata yang sama namun maknanya berbeda. Contoh, kata "membajak" dalam "membajak sawah" dan "membajak VCD" merupakan polisemi karena kata "membajak" di kedua frase sama namun mempunyai arti yang berbeda.

Metode *Latent Semantic Indexing* (LSI) adalah metode yang diimplementasikan di dalam IR system dalam mencari dan menemukan informasi berdasarkan makna keseluruhan (*conceptual topic* atau *meaning*) dari sebuah dokumen bukan hanya makna kata per kata.

III.2 Metode *Latent Semantic Indexing* Secara Keseluruhan

Secara global, alur proses metode *Latent Semantic Indexing* (LSI) dapat diilustrasikan dalam gambar III.1.



Gambar III-1 Alur proses dari metode *latent semantic indexing*

Alur proses dari metode *Latent Semantic Indexing* dibagi 2 (dua) kolom, yaitu kolom sebelah kiri yaitu *query* dan kolom sebelah kanan kanan yaitu, koleksi dokumen. Pada proses sebelah kiri, *query* diproses melalui operasi teks,

kemudian vektor *query* dibentuk. Vektor *query* yang dibentuk dipetakan menjadi vektor *query* terpeta (*mapped query vector*). Dalam membentuk *query* terpeta, diperlukan hasil dekomposisi nilai singular dari koleksi dokumen. Pada koleksi dokumen, dilakukan operasi teks pada koleksi dokumen, kemudian matriks kata-dokumen (*terms-documents matrix*) dibentuk, selanjutnya dilakukan dekomposisi nilai singular (*Singular Value Decomposition*) pada matriks kata-dokumen. Hasil dekomposisi disimpan dalam *collection index*. Proses *ranking* dilakukan dengan menghitung relevansi antara vektor *query* terpeta dan *collection index*. Selanjutnya, hasil perhitungan relevansi ditampilkan ke pengguna.

Dalam subbab-subbab berikutnya dibahas mengenai konsep aljabar linier elementer yang mendasari metode LSI.

III.3 Notasi dan Terminologi Matriks

Sebuah matriks adalah larik berbentuk persegi panjang yang terdiri dari angka-angka. Angka-angka di dalam larik disebut *entry* dalam matriks (2).

Ukuran dari sebuah matriks dideskripsikan dengan banyaknya baris dan kolom di dalamnya. Suatu matriks disebut matriks bujursangkar apabila banyak baris dan banyak kolom dari matriks tersebut sama.

Contoh:

Pandang matriks, $A = \begin{bmatrix} -1 & 0 & 1 \\ 2 & 1 & 2 \\ 1 & 3 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 2 \\ 3 & 0 \\ -1 & 4 \end{bmatrix}$.

Matriks A merupakan matriks bujursangkar yang berukuran 3×3.

Matriks B terdiri dari 3 baris dan 2 kolom, atau B matriks berukuran 3×2.

1 dan 2 disebut *entry* baris ke-1 kolom ke-1 dan *entry* baris ke-1 kolom ke-2.

3 dan 0 disebut *entry* baris ke-2 kolom ke-1 dan *entry* baris ke-2 kolom ke-2.

-1 dan 4 disebut *entry* baris ke-3 kolom ke-1 dan *entry* baris ke-3 kolom ke-2

III.4 Perkalian Matriks

Diketahui matriks $A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ dan $B=\begin{bmatrix} e & f \\ g & h \end{bmatrix}$ maka

perkalian matriks A dan matriks B yaitu $AB=C$ adalah

$$AB=\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix}=\begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}=C.$$

Perhatikan bahwa *entry* pada baris pertama dan kolom pertama di matriks C adalah hasil perkalian setiap *entry* dari baris pertama di matriks A dikalikan dengan setiap *entry* dari kolom pertama di matriks B dan hasil akhirnya adalah penjumlahan setiap perkalian *entry*.

Inti perkalian matriks adalah

(1) *entry* pada baris i dan kolom j dari matriks AB sama dengan perkalian baris i dari matriks A dengan kolom j dari matriks B.

(2) perkalian AB dapat dihitung jika dan hanya jika banyak *entry* pada baris A sama dengan banyak *entry* pada kolom B

Contoh:

Pandang, $A=\begin{bmatrix} 3 & -1 \\ -4 & 2 \end{bmatrix}$, $B=\begin{bmatrix} 5 & 2 \\ -7 & 3 \end{bmatrix}$ maka

$$AB=\begin{bmatrix} 3 & -1 \\ -4 & 2 \end{bmatrix}\begin{bmatrix} 5 & 2 \\ -7 & 3 \end{bmatrix}=\begin{bmatrix} 3\times 5-1\times(-7) & 3\times 2-1\times 3 \\ -4\times 5+2\times(-7) & -4\times 2+2\times 3 \end{bmatrix}=\begin{bmatrix} 22 & 3 \\ -34 & -2 \end{bmatrix}$$

$$BA=\begin{bmatrix} 5 & 2 \\ -7 & 3 \end{bmatrix}\begin{bmatrix} 3 & -1 \\ -4 & 2 \end{bmatrix}=\begin{bmatrix} 5\times 3+2\times(-4) & 5\times(-1)+2\times 2 \\ -7\times 3+3\times(-4) & -7\times(-1)+3\times 2 \end{bmatrix}=\begin{bmatrix} 7 & -1 \\ -33 & 13 \end{bmatrix}$$

III.5 Operasi Baris Elementer

Operasi baris elementer merupakan operasi dikenakan pada sebuah matriks sembarang meliputi:

(1) Mengalikan sebuah baris dengan skalar tak nol.

(2) Mengalikan sebuah baris dengan skalar tak nol dan menambahkan ke

baris lainnya.

(3) Menukar dua baris.

Jika sebuah matriks B diperoleh dari matriks A melalui operasi-operasi di atas, maka A dan B dikatakan ekivalen baris.

Contoh:

$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ dan $\begin{bmatrix} 1 & 2 & 1 \\ 6 & 3 & 3 \\ 1 & 2 & 1 \end{bmatrix}$ dikatakan ekivalen baris karena $\begin{bmatrix} 1 & 2 & 1 \\ 6 & 3 & 3 \\ 1 & 2 & 1 \end{bmatrix}$ diperoleh dari $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ setelah dilakukan operasi mengalikan baris kedua dengan 3.

$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ dan $\begin{bmatrix} 1 & 2 & 1 \\ 0 & -3 & -1 \\ 1 & 2 & 1 \end{bmatrix}$ dikatakan ekivalen baris karena $\begin{bmatrix} 1 & 2 & 1 \\ 0 & -3 & -1 \\ 1 & 2 & 1 \end{bmatrix}$ diperoleh dari $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ setelah dilakukan operasi mengalikan baris pertama dengan -2 kemudian menambahkan ke baris kedua.

$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ dan $\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$ dikatakan ekivalen baris karena $\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$ diperoleh dari $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$ setelah dilakukan operasi menukar baris kedua dan baris ketiga.

III.6 Matriks *Echelon* Baris Tereduksi (*reduced row-echelon matrix*)

Sebuah matriks dikatakan matriks *echelon* baris tereduksi jika

(1) Semua baris nol terdapat di bawah semua baris tak nol.

(2) Entry tak nol pertama dari baris tak nol adalah 1. *Entry* tersebut disebut *leading entry* dari baris tersebut.

(3) *Leading entry* dari setiap baris merupakan satu-satunya *entry* tak nol pada kolomnya.

Contoh matriks *echelon* baris tereduksi sebagai berikut

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \text{ dan } \begin{bmatrix} 1 & 2 & 0 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

III.7 *Rank* Matriks

Diketahui dua buah matriks A dan B. B merupakan ekivalen baris dari A. Jika B adalah matriks *echelon* baris tereduksi, dapat dikatakan bahwa B adalah bentuk *echelon* baris tereduksi dari A. *Rank* dari matriks A, $rank(A)$, adalah banyak baris tak nol dalam matriks *echelon* baris tereduksi dari A.

Contoh:

Diketahui $A = \begin{pmatrix} 1 & -1 & 3 & 7 \\ 1 & 0 & 6 & 5 \\ 0 & 2 & 6 & -4 \end{pmatrix}$

Gunakan operasi baris elementer, diperoleh

$$A \mapsto \begin{pmatrix} 1 & -1 & 3 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 2 & 6 & -4 \end{pmatrix} \mapsto \begin{pmatrix} 1 & -1 & 3 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 6 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Diperoleh bahwa

$\begin{pmatrix} 1 & 0 & 6 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ adalah bentuk matriks *echelon* tereduksi dari A. Maka

$rank(A) =$ banyak baris tak nol dalam matriks echelon baris tereduksi dari A.

III.8 Invers dari Sebuah Matriks

Jika A adalah sebuah matriks bujursangkar, dan jika sebuah matriks B berukuran sama dengan A dapat ditemukan sedemikian sehingga $AB = BA = I$ (matriks identitas), maka A dikatakan mempunyai invers dan B disebut invers dari A.

Contoh:

Matriks $B = \begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix}$ merupakan invers dari $A = \begin{bmatrix} 2 & -5 \\ -1 & 3 \end{bmatrix}$ karena

$$AB = \begin{bmatrix} 2 & -5 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \text{ dan}$$

$$BA = \begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 2 & -5 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

III.9 Transpose dari Sebuah Matriks

Jika A matriks sembarang berukuran $m \times n$, maka *transpose* dari A, A^T didefinisikan sebagai matriks berukuran $n \times m$ yang merupakan matriks dengan *entry* hasil pertukaran baris dan kolom dari A. Kolom ke-1 dari A^T adalah baris ke-1 dari A, kolom ke-2 dari A^T adalah baris ke-2 dari A, dan seterusnya.

Contoh:

Pandang $C = \begin{bmatrix} 2 & 1 & 5 \\ 3 & 4 & 6 \end{bmatrix}$, maka $C^T = \begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 5 & 6 \end{bmatrix}$.

III.10 Matriks *Unitary*

Suatu matriks A disebut matriks *unitary* jika transpose dan invers dari A adalah identik, yaitu $AA^{-1} = I = AA^T$.

Contoh:

$A = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}$ dan $B = \frac{1}{2}\begin{bmatrix} 1+i & 1+i \\ 1-i & -1+i \end{bmatrix}$ adalah contoh matriks unitary karena transpose dari matriks A dan invers A adalah identik, begitu juga dengan matriks B. (10).

III.11 Matriks Simetri

Suatu matriks A dikatakan simetri apabila $A^T = A$.

Contoh:

Pandang $A=\begin{bmatrix}-1 & 2 & 1\\ 2 & 2 & 2\\ 1 & 2 & 1\end{bmatrix}$, maka A merupakan matriks simetri karena

$$A^T=\begin{bmatrix}-1 & 2 & 1\\ 2 & 2 & 2\\ 1 & 2 & 1\end{bmatrix}=A.$$

III.12 Teorema Membangun Matriks Simetri

Jika A sebuah matriks berukuran $m\times n$, maka A^T adalah matriks berukuran $n\times m$, sehingga perkalian AA^T dan A^TA, keduanya merupakan matriks bujursangkar, yaitu AA^T berukuran $m\times m$ dan A^TA berukuran $n\times n$.

Perkalian matriks AA^T dan matriks A^TA selalu simetri karena

$(AA^T)^T=(A^T)^TA^T=AA^T$ dan $(A^TA)^T=A^T(A^T)^T=A^TA$

Contoh:

$A=\begin{bmatrix}1 & -1 & 0\\ 2 & 1 & -2\end{bmatrix}$, maka

$$A^TA=\begin{bmatrix}1 & 2\\ -1 & 1\\ 0 & -2\end{bmatrix}\begin{bmatrix}1 & -1 & 0\\ 2 & 1 & -2\end{bmatrix}=\begin{bmatrix}5 & 1 & -4\\ 1 & 2 & -2\\ -4 & -2 & 4\end{bmatrix}$$ dan

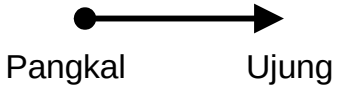
$$AA^T=\begin{bmatrix}1 & -1 & 0\\ 2 & 1 & -2\end{bmatrix}\begin{bmatrix}1 & 2\\ -1 & 1\\ 0 & -2\end{bmatrix}=\begin{bmatrix}2 & 1\\ 1 & 9\end{bmatrix}.$$

Dari contoh terlihat bahwa AA^T dan A^TA merupakan matriks simetri.

III.13 Definisi Vektor Secara Geometrik

Vektor digambarkan secara geometrik sebagai anak panah di dalam ruang vektor berdimensi 2 atau berdimensi 3. Arah anak panah menunjukkan arah vektor dan

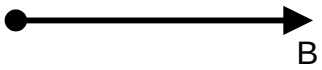
panjang anak panah menggambarkan besar atau panjang dari vektor (Gambar III.2).



Gambar III-2 Sebuah vektor dilihat secara geometri

Dua buah vektor dinamakan sama apabila keduanya sama besarnya (sama panjangnya) dan arahnya juga sama (2).

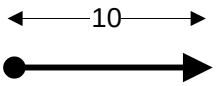
Sebuah vektor dapat ditulis dengan menggunakan notasi vektor $\overrightarrow{AB}$, mempunyai titik pangkal di A dan titik ujung di B (Gambar III.3); atau vektor **u** (diberi cetak tebal) atau $\vec{u}$.



Gambar III-3 Contoh vektor dengan notasi $\overrightarrow{AB}$

Sebuah vektor digambarkan secara aljabar menjadi sebuah matriks berukuran $n \times 1$, dengan n adalah banyaknya dimensi dari ruang vektor.

Contoh:



Gambar III-4 Contoh vektor di ruang vektor berdimensi 2

Vektor pada gambar III.4 secara geometris mempunyai arah ke timur dengan panjang sebesar 10. Secara aljabar, vektor di atas dapat ditulis dalam matriks $\begin{bmatrix} 10 \\ 0 \end{bmatrix}$.

III.14 Kombinasi Linier (Membangun)

Diketahui $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ adalah vektor-vektor dan $r_1, r_2, \ldots, r_n$ adalah skalar. Maka vektor

$$\vec{w} = r_1\vec{v}_1 + r_2\vec{v}_2 + \ldots + r_n\vec{v}_n$$

adalah kombinasi linier dari $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$. Himpunan semua kombinasi linier dari $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ disebut *span* dari $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$, diberi notasi $span\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$.

III.15 Definisi Ruang Vektor

Diketahui V merupakan himpunan tidak kosong yang terdiri dari objek-objek dengan dua operasi didefinisikan, yaitu operasi penambahan dan operasi perkalian skalar.

Operasi penambahan artinya aturan yang dikenakan pada setiap pasangan objek $\vec{u}$ dan $\vec{v}$, anggota V untuk menghasilkan $\vec{u} + \vec{v}$.
Operasi perkalian skalar artinya aturan yang dikenakan pada setiap pasangan skalar k dan objek $\vec{u}$, anggota V untuk menghasilkan objek $k\vec{u}$.

Jika aksioma berikut dipenuhi oleh semua objek $\vec{u}, \vec{v}, \vec{w}$ anggota V dan skalar k dan l, maka V adalah sebuah ruang vektor dan objek anggota V disebut vektor.

(1) Jika $\vec{u}$ dan $\vec{v}$ adalah objek anggota V, maka $\vec{u} + \vec{v}$ anggota V juga.
(2) $\vec{u} + \vec{v} = \vec{v} + \vec{u}$
(3) $\vec{u} + (\vec{v} + \vec{w}) = (\vec{u} + \vec{v}) + \vec{w}$
(4) Ada objek $\vec{0}$, anggota V, yang disebut vektor nol untuk V, sedemikian sehingga $\vec{0} + \vec{u} = \vec{u} + \vec{0} = \vec{u}$ untuk semua $\vec{u}$ anggota V.
(5) Untuk setiap $\vec{u}$ anggota V, ada objek $-\vec{u}$, anggota V, yang disebut negatif $\vec{u}$, sedemikian sehingga $\vec{u} + (-\vec{u}) = (-\vec{u}) + \vec{u} = \vec{0}$.
(6) Jika k adalah skalar dan $\vec{u}$ adalah objek anggota V, maka $k\vec{u}$ anggota V.
(7) $k(\vec{u} + \vec{v}) = k\vec{u} + k\vec{v}$

(8) $(k+l)\vec{u} = k\vec{u} + l\vec{u}$

(9) $k(l\vec{u}) = (kl)(\vec{u})$

(10) $1\,\vec{u} = \vec{u}$

Contoh ruang vektor adalah R (bilangan riil), R^2 (vektor-vektor di bidang), dan R^3 (vektor-vektor di ruang berdimensi 3). Bentuk umum ruang vektor bilangan riil atau (*Euclidean Space*) adalah R^n (2).

III.16 Subruang Vektor

Diketahui V merupakan ruang vektor. W merupakan subruang vektor dari V bila

(i) $W \subseteq V$.

(ii) $W \neq \{\}$.

(iii) Jika $\vec{v} \in W$ dan a adalah skalar, maka $a\vec{v} \in W$.

(iv) Jika $v_1, v_2 \in W$, maka $(v_1 + v_2) \in W$.

Contoh:

Diketahui R^3 merupakan ruang vektor untuk semua vektor berdimensi 3 (tiga), yaitu $R^3 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \middle| \, x, y, z \in R \right\}$. Maka $W = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \middle| \, x + y - z = 0 \; dan \; x, y, z \in R \right\}$ merupakan subruang vektor dari R^3.

III.17 Ruang Baris, Ruang Kolom dan Ruang *Null*

Misalkan A adalah matriks berukuran $m \times n$ dengan semua entry di $A \in R$. Maka

- Subruang vektor dari R^n yang dibangun oleh baris-baris dari A disebut ruang baris dari A, row(A).
- Subruang vektor dari R^m yang dibangun oleh kolom-kolom dari A disebut ruang kolom dari A, col(A).

- Subruang vektor dari R^n yang dibangun oleh semua vektor yang merupakan solusi $AX = \vec{0}$ disebut ruang *null* dari A, ker(A).

Contoh:

Pandang matriks dengan *entry* matriks bilangan riil

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 2 \end{bmatrix}$$

Ruang baris A adalah subruang yang dibangun oleh vektor $\begin{bmatrix} 1 & 3 & 1 \end{bmatrix}$ dan $\begin{bmatrix} 1 & 0 & 2 \end{bmatrix}$.

Ruang kolom A adalah subruang yang dibangun oleh vektor $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\begin{bmatrix} 3 \\ 0 \end{bmatrix}$, dan $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

Ruang *null* A adalah subruang yang dibangun oleh himpunan solusi dari sistem

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

yaitu $\begin{bmatrix} 6 \\ -1 \\ -3 \end{bmatrix}$.

III.18 Pemetaan Linier

Diketahui V dan W masing-masing merupakan ruang vektor. Pemetaan linier dari V ke W adalah fungsi $T : V \to W$ yang memenuhi (10):

(i) Jika $\vec{u}, \vec{v} \in V$, maka $T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$.

(ii) Jika $\vec{v} \in V$ dan k adalah skalar, maka $T(k\vec{v}) = kT(\vec{v})$.

Contoh:

Misalkan $F : R \to R$ didefinisikan $F(x) = 3x$ dan $G : R \to R$ didefinisikan $G(x) = 5x + 8$. Maka

$F(a+b) = 3(a+b) = 3a + 3b = F(a) + F(b)$ dan $F(ka) = 3(ka) = k(3a) = kF(a)$

sehingga F merupakan pemetaan linier.

Kemudian apabila $G(a+b) = 5(a+b) + 8 \neq (5a+8) + (5b+8) = G(a) + G(b)$

sehingga G tidak merupakan pemetaan linier.

III.19 Dimensi Ruang Kolom dan Ruang Baris

Misalkan $T = T_A : R^n \to R^m$ adalah **pemetaan linier** dengan A matriks berukuran $m \times n$ dengan semua entry di $A \in R$. Maka

(i) ker(T) adalah himpunan solusi $AX = \vec{0}$.

(ii) im(T) adalah ruang kolom A.

(iii) $rank(T) = rank(A)$.

(iv) $rank(T) + null(T) = n$, dengan $null(T)$ adalah dimensi dari ker(T).

III.20 Teorema *Norm* dari Suatu Vektor

Panjang dari suatu vektor $\vec{u}$ sering juga disebut *norm* dari $\vec{u}$ dilambangkan dengan $\|\vec{u}\|$. Bila $\vec{u} = (u_1, u_2, \cdots, u_n)$ merupakan vektor di ruang vektor berdimensi n, maka *norm* $\vec{u}$ ditulis

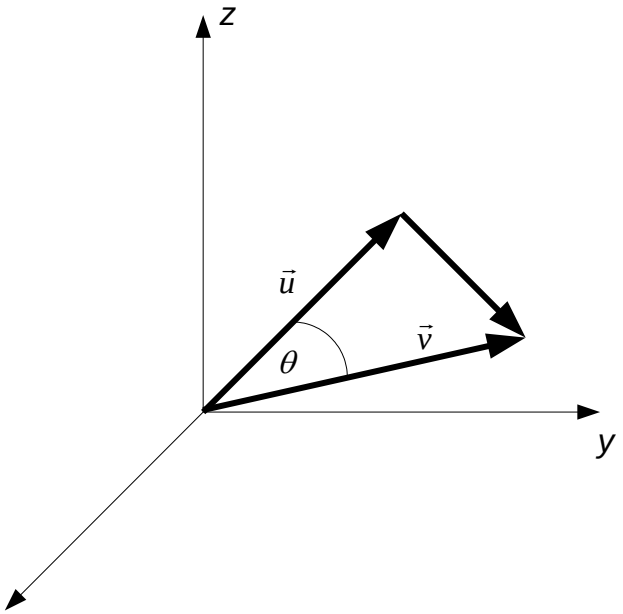
$$\|\vec{u}\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2} \quad \text{.....(III.1)}$$

Contoh:

Diketahui $\vec{u} = (3, 4)$, maka *norm* dari $\vec{u}$, $\|\vec{u}\| = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$

III.21 Teorema Sudut Antara Dua Vektor

Diketahui $\vec{u}$ dan $\vec{v}$ adalah vektor-vektor tak nol di sebuah ruang vektor yang berdimensi n.



Gambar III-5 Sudut yang dibentuk oleh $\vec{u}$ dan $\vec{v}$ di ruang vektor

Sudut yang dibentuk antara $\vec{u}$ dan $\vec{v}$ adalah (Gambar III.5)

$$\cos\theta = \frac{\vec{u} \bullet \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|} \quad \text{.....(III.2)}$$

dengan $\|\vec{u}\|$ adalah norm dari $\vec{u}$ dan $\|\vec{v}\|$ adalah norm dari $\vec{v}$.

Contoh:

Diketahui vektor $\vec{u} = (2, -1, 1)$ dan $\vec{v} = (1, 1, 2)$, maka $\vec{u} \bullet \vec{v} = u_1 v_1 + u_2 v_2 + u_3 v_3 = (2)(1) + (-1)(1) + (1)(2) = 3$.

$\|\vec{u}\| = \|\vec{v}\| = \sqrt{6}$ sehingga (III.2) memberikan

$$\cos\theta = \frac{\vec{u} \bullet \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|} = \frac{3}{\sqrt{6}\,\sqrt{6}} = \frac{1}{2}.$$

Diperoleh $\cos\theta = \frac{1}{2}$, maka $\theta = 60°$.

III.22 Nilai Eigen dan Vektor Eigen

Jika A adalah matriks $n \times n$, maka sebuah vektor tak nol $\vec{x}$ anggota R^n disebut vektor eigen dari A jika $A\vec{x}$ adalah perkalian skalar dari $\vec{x}$; yaitu,

$$A\vec{x} = \lambda\vec{x} \quad \text{.....(III.3)}$$

untuk beberapa nilai λ. Nilai skalar λ disebut nilai eigen dari A, dan $\vec{x}$ dikatakan vektor eigen dari A yang berkaitan (*corresponding*) dengan λ (2).

Contoh:

Vektor $\vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ merupakan vektor eigen dari $A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$ yang berkaitan dengan nilai eigen $\lambda = 3$, karena

$$A\vec{x} = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} = 3\vec{x}$$

III.23 Himpunan Vektor Ortonormal

Suatu himpunan yang terdiri beberapa vektor, $\{x_1, x_2, x_3, \cdots, x_n\}$ dikatakan himpunan vektor ortonormal bila

(1) vektor satu ortogonal (tegak lurus) dengan vektor lainnya, yaitu $x_i \bullet x_j = 0, \quad$ untuk $i \neq j$ dan $i = 1, 2, \ldots, n$.

(2) *norm* dari masing-masing vektor di dalam himpunan adalah 1, yaitu $\|x_i\| = 1, \quad$ untuk $i = 1, 2, \ldots, n$.

Contoh himpunan vektor ortonormal adalah $\left\{ \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \right\}$.

III.24 Teorema Pendiagonalan Ortogonal

Suatu matriks A berukuran $n \times n$ dikatakan dapat didiagonalkan ortogonal jika ada matriks ortogonal P sedemikian sehingga matriks $P^{-1}AP = P^T AP$ adalah

matriks diagonal (matriks yang *entry* di diagonal utamanya tak nol dan *entry* lainnya nol semua).

Jika A sebuah matriks berukuran $n \times n$, maka berikut ini adalah ekivalen

(1) A dapat didiagonalkan secara ortogonal,

(2) A mempunyai himpunan vektor eigen yang ortonormal sebanyak n buah

(3) A simetri

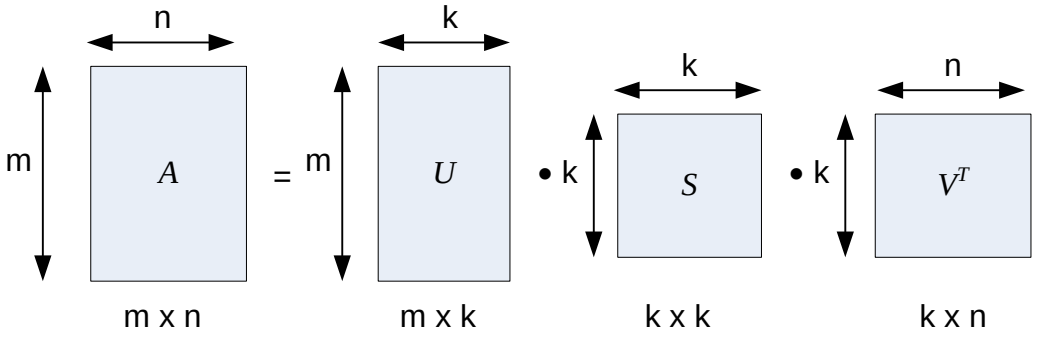
Contoh suatu matriks $\begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix}$ didiagonalkan ortogonal menjadi

$$\begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix}$$

III.25 Teorema *Singular Value Decomposition* (SVD)

Singular Value Decomposition (SVD) adalah suatu metode untuk mendekomposisi suatu matriks, A berukuran $m \times n$, menjadi 3 (tiga) buah matriks, yaitu U, S, dan V seperti pada ilustrasi di bawah ini (1).

$$A = U\, S\, V^T \quad \text{.....(III.4)}$$



Gambar III-6 Ilustrasi dekomposisi nilai singular (SVD) dari A

Hasil SVD adalah matriks U adalah matriks berukuran $m \times k$ dan V matriks bujursangkar $n \times k$, keduanya mempunyai kolom-kolom ortogonal sedemikian sehingga

$$U^T U = V^T V = I \quad \text{..... (III.5)}$$

dan S adalah matriks diagonal berukuran $k \times k$ (22). *Entry-entry* di diagonal utama matriks S adalah nilai singular dari matriks A.

Hasil SVD dapat lebih dipahami apabila matriks A ditulis dengan interpretasi yang berbeda. Bila $u_1, u_2, \ldots, u_k$ adalah vektor-vektor kolom dari matriks U, $\sigma_1, \sigma_2, \ldots, \sigma_k$ adalah *entry-entry* di diagonal utama dari matriks S, dan $v_1, v_2, \ldots, v_k$ adalah vektor-vektor kolom dari matriks V. Maka matriks A dapat ditulis sebagai

$$A = \sum_{i=1}^{k} \sigma_i \, u_i \, v_i^T \quad \text{.....(III.6)}$$

Nilai-nilai σ_i, untuk $i = 1, 2, \ldots, k$, pada persamaan (III.6) diurutkan menurun dari yang terbesar sampai terkecil. Apabila beberapa nilai σ_i yang besar diambil dan nilai σ_i yang kecil (mendekati nol) dibuang, kita memperoleh suatu aproksimasi dari A yang baik. Jadi, dengan SVD, suatu matriks dapat ditulis sebagai penjumlahan dari komponen-komponen ($u_i \, v_i^T$ untuk $i = 1, 2, \ldots, k$)dengan bobotnya adalah nilai singular (σ_i, untuk $i = 1, 2, \ldots, k$).

Contoh:

$$\begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} 0.84 & 0.24 \\ 0.08 & -0.12 \\ 0.24 & 0.64 \\ 0.48 & -0.72 \end{bmatrix} \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 0.6 & 0.8 \\ 0.8 & -0.6 \end{bmatrix}$$

$$A \quad = \quad U \qquad\qquad S \qquad\qquad V^T$$

Apabila matriks A ditulis dengan menggunakan bentuk (III.6), hasilnya adalah

$$\begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix} = 10 \begin{bmatrix} 0.84 \\ 0.08 \\ 0.24 \\ .48 \end{bmatrix} \begin{bmatrix} 0.6 & 0.8 \end{bmatrix} + 5 \begin{bmatrix} 0.24 \\ -0.12 \\ 0.64 \\ -0.72 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \end{bmatrix}$$

III.26 Makna Hasil *Singular Value Decomposition*

Berikut ini adalah konsep atau teorema yang dapat digunakan untuk memahami makna matriks hasil dekomposisi matriks dengan SVD (6). Untuk bukti teorema tidak ditulis di sini dan dapat dibaca pada referensi yang digunakan. (19).

Misalkan A adalah matriks berukuran $m \times n$ dan A mempunyai *rank* r, kemudian A didekomposisi SVD menjadi $A = USV^T$, dengan U matriks berukuran $m \times r$, S matriks diagonal berukuran $r \times r$, dan V^T berukuran $r \times n$. Misalkan U ditulis dengan menggunakan vektor-vektor kolom menjadi $U = [\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_r]$ dengan $\vec{u}_i$ adalah vektor kolom ke-i dari matriks U dan $V = [\vec{v}_1 \quad \vec{v}_2 \quad \cdots \quad \vec{v}_r]$ dengan $\vec{v}_i$ adalah vektor kolom ke-i dari matriks V. Maka

(i) $\text{im}(A) = \text{im}(U) = span\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\}$

(ii) $\text{im}(A^T) = \text{im}(V) = span\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r\}$

Dengan menggunakan konsep pada subbab III.25 dan point (i) di atas, dapat disimpulkan bahwa $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r$ membangun ruang kolom A dan $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r$ disebut vektor-vektor kata (*term*) dari koleksi dokumen.

Hal yang sama juga dilakukan point (ii) sehingga diperoleh bahwa $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_r$ membangun ruang kolom A^T atau $\vec{v}_1{}^T, \vec{v}_2{}^T, \ldots, \vec{v}_r{}^T$ membangun ruang baris A. Selanjutnya $\vec{v}_1{}^T, \vec{v}_2{}^T, \ldots, \vec{v}_r{}^T$ disebut vektor-vektor dokumen (*document*) dari koleksi dokumen.

III.27 Algoritma Memperoleh *Singular Value Decomposition*

Misalkan diketahui A matriks berukuran $m \times n$. Bagaimanakah mencari *singular value decomposition* dari matriks A? Berikut ini adalah langkah-langkah di dalam membangun *singular value decomposition* dari A (6)

(1) Bentuk $A^T A$ yang merupakan matriks simetri berukuran $n \times n$.

(2) Dekomposisi $A^T A$ dengan pendiagonalan ortogonal menjadi

$$A^T A = VSV^T$$

dengan V matriks ortogonal berukuran $n \times k$, dengan $k = rank$ dari matriks $A^T A$, $VV^T = V^T V = I$.

(3) Bentuk AA^T yang merupakan matriks simetri berukuran $m \times m$.

(4) Dekomposisi AA^T dengan pendiagonalan ortogonal menjadi

$$AA^T = USU^T$$

dengan U matriks ortogonal berukuran $m \times k$, dengan $k = rank$ dari matriks AA^T, $UU^T = U^T U = I$.

(5) Pandang

$$AA^T = USU^T$$

$$\Leftrightarrow AA^T = US^{\frac{1}{2}} IS^{\frac{1}{2}} U^T$$

$$\Leftrightarrow AA^T = US^{\frac{1}{2}} (V^T V) S^{\frac{1}{2}} U^T$$

$$\Leftrightarrow AA^T = (US^{\frac{1}{2}} V^T)(VS^{\frac{1}{2}} U^T)$$

$$\Leftrightarrow AA^T = US^{\frac{1}{2}} V^T (US^{\frac{1}{2}} V^T)^T$$

$$\Leftrightarrow A = US^{\frac{1}{2}} V^T \quad \text{.....(III.7)}$$

Hasil Dekomposisi Nilai Singular dari matriks A adalah $A = US^{\frac{1}{2}} V^T$.

Contoh:

Carilah dekomposisi nilai singular (*singular value decomposition*) dari matriks

$$F = \begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix} !$$

Algoritma di atas dijalankan

(1) Bentuk matriks $F^T F$, yaitu $F^T F = \begin{bmatrix} 52 & 36 \\ 36 & 73 \end{bmatrix}$

(2) Didiagonalkan $F^T F$ secara ortogonal menjadi

$$\begin{bmatrix} 0.6 & 0.8 \\ 0.8 & -0.6 \end{bmatrix} \begin{bmatrix} 100 & 0 \\ 0 & 25 \end{bmatrix} \begin{bmatrix} 0.6 & 0.8 \\ 0.8 & -0.6 \end{bmatrix}$$

(3) $FF^T = \begin{bmatrix} 72 & 6 & 24 & 36 \\ 6 & 1 & 0 & 6 \\ 24 & 0 & 16 & 0 \\ 36 & 6 & 0 & 36 \end{bmatrix}$

(4) FF^T juga dapat didiagonalkan secara ortogonal menjadi

$$\begin{bmatrix} 0.84 & 0.24 \\ 0.08 & -0.12 \\ 0.24 & 0.64 \\ 0.48 & -0.72 \end{bmatrix} \begin{bmatrix} 100 & 0 \\ 0 & 25 \end{bmatrix} \begin{bmatrix} 0.84 & 0.08 & 0.24 & 0.48 \\ 0.24 & -0.12 & 0.64 & -0.72 \end{bmatrix}$$

(5) Matriks F dapat dinyatakan dalam bentuk dekomposisi nilai singular (*singular value decomposition*),

$$F = U S^{\frac{1}{2}} V^T$$

$$\begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} 0.84 & 0.24 \\ 0.08 & -0.12 \\ 0.24 & 0.64 \\ 0.48 & -0.72 \end{bmatrix} \begin{bmatrix} \sqrt{100} & 0 \\ 0 & \sqrt{25} \end{bmatrix} \begin{bmatrix} 0.6 & 0.8 \\ 0.8 & -0.6 \end{bmatrix}$$

III.28 Konsep Metode *Latent Semantic Indexing* (LSI)

Konsep *Latent Semantic Indexing* (LSI) merupakan metode IR yang membangun struktur koleksi dokumen dalam bentuk ruang vektor dengan menggunakan teknik aljabar linier, yaitu *singular value decomposition*.

Secara umum, konsep LSI meliputi beberapa point seperti dilustrasikan pada gambar III.1 yaitu:

(1) *Text Operations* pada *Query* dan *Document Collection*.

Query dari pengguna dan koleksi dokumen dikenakan proses *text operations*. Proses *text operations* meliputi,

(i) mem-*parsing* setiap kata dari koleksi dokumen,

(ii) membuang kata-kata yang merupakan *stop words*,

(iii) mem-*stemming* kata-kata yang ada untuk proses selanjutnya.

(2) *Matrix Creation*.

Hasil *text operations* yang dikenakan pada koleksi dokumen dikenakan proses *matrix creation*. Proses *matrix creation* meliputi,

(i) menghitung frekuensi kemunculan dari kata,

(ii) membangun matriks kata-dokumen seperti dilustrasikan pada gambar II.4. Baris matriks menunjukkan kata dan kolom matriks menunjukkan dokumen. Sebagai contoh, elemen matriks pada baris ke-1 dan kolom ke-2 menunjukkan frekuensi kemunculan kata ke-1 pada dokumen ke-2.

(3) *SVD Decomposition*.

(i) Matriks kata-dokumen yang terbentuk, A berukuran $m \times n$, selanjutnya dikenakan dekomposisi SVD (*singular value decomposition*). Hasil SVD berupa 3 (tiga) buah matriks seperti yang dilustrasikan pada gambar III.6. Matriks A dapat ditulis menjadi $A = USV^T$.

(ii) Untuk mempermudah penjelasan, misalkan $u_1, u_2, \ldots, u_k$ adalah vektor-vektor kolom dari matriks U, $\sigma_1, \sigma_2, \ldots, \sigma_k$ adalah *entry-entry* di diagonal utama dari matriks S, dan $v_1, v_2, \ldots, v_k$ adalah vektor-vektor kolom dari matriks V, sehingga dapat ditulis

$$A = \quad U \qquad\qquad S \qquad\qquad V^T$$

$$A = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$$

(iii) *Rank* dari matriks A, k adalah banyaknya *entry* tak nol yang terletak pada diagonal utama matriks S, yaitu $\sigma_1, \sigma_2, \ldots, \sigma_k$.
k juga merupakan banyaknya nilai singular dari A.

(iv) Dari k buah nilai singular dari A, dipilih r buah nilai singular yang terbesar, yaitu $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, dengan $r < k$.

(v) Diperoleh hasil perkalian baru yaitu

$U_r \, S_r \, V_r^T$ dengan $U_r = \begin{bmatrix} u_1 & u_2 & \cdots & u_r \end{bmatrix}$,

$$S_r = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}, \text{ dan } V_r^T = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_r^T \end{bmatrix}.$$

(4) <u>*Query Vector Creation*</u>.

Vektor *query*, q dibentuk seperti membangun sebuah kolom dari matriks kata-dokumen.

Contoh vektor *query*, q adalah

$$q = \begin{matrix} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{matrix} \begin{matrix} \textit{Query} \\ \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} \end{matrix},$$

dengan q_j, $j = 1, 2, \ldots, m$ adalah frekuensi kemunculan kata T_j pada *Query*.

(5) <u>*Query Vector Mapping*</u>.

Point (3)(v) di atas telah memberikan nilai r yang merupakan dimensi dari ruang vektor hasil perkalian baru. Selanjutnya, vektor *query*, q dipetakan ke dalam ruang vektor berdimensi r menjadi Q (subbab III.30), yaitu

$$Q = q^T U_r S_r^{-1}$$

(6) <u>*Ranking*</u>.

Kolom-kolom pada matriks V_r^T pada point (3)(v) adalah vektor-vektor dokumen yang digunakan dalam menghitung sudut antara vekor dokumen dan vektor *query*.

Ranking dari dokumen relevan ditentukan oleh besar sudut yang dibentuk oleh vektor *query* dan vektor dokumen. Semakin kecil sudut yang dibentuk, semakin relevan *query* dengan dokumen.

Misalkan matriks V_r ditulis

$$V_r = \underbrace{\begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{bmatrix}}_{r} \text{,dengan } D_j^T = \begin{bmatrix} d_{j1} \\ \vdots \\ d_{jr} \end{bmatrix},\ j = 1, 2, \ldots, n$$

D_j, $j = 1, 2, \ldots, n$ adalah vektor dokumen untuk dokumen ke- j.

Kemudian misalkan vektor *query*, Q ditulis $Q = \begin{bmatrix} q_1 \\ \vdots \\ q_r \end{bmatrix}$ maka

dengan menggunakan rumus (II.3), cosinus sudut yang dibentuk adalah

$$Sim(Q, D_j) = \cos(Q, D_j) = \frac{Q \bullet D_j}{|Q|\,|D_j|} = \frac{1}{|Q|\,|D_j|} \sum_{i=1}^{r} q_j \times d_{ji}$$

(7) Hasil akhir.

Perhitungan cosinus sudut antara *query*, Q dan dokumen D_j, $j = 1, 2, \ldots, n$ diperoleh dan diurutkan berdasarkan dari yang paling besar sampai yang terkecil.

Nilai cosinus sudut yang terbesar menunjukkan dokumen yang paling relevan dengan *query*.

III.29 Hubungan Vektor *Query* Dengan Vektor Dokumen

Relevansi antara *query* dengan dokumen dihitung dengan menggunakan konsep hasil kali titik antara dua vektor (subbab III.21). Dalam hal ini, hasil kali titik antara vektor dokumen untuk *query* dengan vektor dokumen dari masing-masing dokumen. Vektor *query* dibentuk seperti membangun sebuah kolom pada matriks kata-dokumen sedangkan vektor dokumen adalah baris-baris pada matriks V, hasil dekomposisi SVD (subbab III.28). Selanjutnya hendak diturunkan hubungan antara vektor *query* dengan vektor dokumen dari masing-masing dokumen.

Misalkan A matriks berukuran $m \times n$ dan A didekomposisi SVD menjadi

$$A = USV^T,$$

dengan U matriks berukuran $m \times r$,

S matriks diagonal berukuran $r \times r$,

V^T berukuran $r \times n$, dan

$r = rank(A)$.

Misalkan juga A dapat ditulis dengan menggunakan vektor-vektor kolom, yang juga merupakan vektor-vektor kata untuk setiap dokumen, yaitu

$A = [\vec{a}_1 \quad \vec{a}_2 \quad \cdots \quad \vec{a}_n]$, dengan $\vec{a}_i$ berukuran $m \times 1$, $i = 1, 2, \ldots, n$.

Demikian juga U ditulis dengan menggunakan vektor-vektor kolom, yaitu

$U = [\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_r]$, dengan $\vec{u}_i$ berukuran $m \times 1$, $i = 1, 2, \ldots, r$.

Matriks S berukuran $r \times r$, yaitu

$$S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}, \text{ dengan } \sigma_i \text{ adalah nilai } \textit{singular}, \ i = 1, 2, \ldots, r$$

Terakhir matriks V ditulis dengan menggunakan vektor-vektor baris, yaitu

$$V = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}, \text{ dengan } \vec{v}_i \text{ berukuran } 1 \times r, \ i = 1, 2, \ldots, n.$$

Selanjutnya, dilakukan manipulasi aljabar sebagai berikut

$$A = USV^T$$

$$\Leftrightarrow A^T = VSU^T$$

$$\Leftrightarrow A^T U S^{-1} = V$$

$$\Leftrightarrow \underbrace{\begin{bmatrix} \vec{a}_1^T \\ \vec{a}_2^T \\ \vdots \\ \vec{a}_n^T \end{bmatrix}}_{A^T} \underbrace{\begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_1^{-1} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^{-1} \end{bmatrix}}_{S^{-1}} = \underbrace{\begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}}_{V}$$

$$\Leftrightarrow \begin{bmatrix} \vec{a}_1^T \vec{u}_1 & \vec{a}_1^T \vec{u}_2 & \cdots & \vec{a}_1^T \vec{u}_r \\ \vec{a}_2^T \vec{u}_1 & \vec{a}_2^T \vec{u}_2 & \cdots & \vec{a}_2^T \vec{u}_r \\ \vdots & \vdots & \ddots & \vdots \\ \vec{a}_n^T \vec{u}_1 & \vec{a}_n^T \vec{u}_2 & \cdots & \vec{a}_n^T \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1^{-1} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^{-1} \end{bmatrix} = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} \vec{a}_1^T \vec{u}_1 \sigma_1^{-1} & \vec{a}_1^T \vec{u}_2 \sigma_2^{-1} & \cdots & \vec{a}_1^T \vec{u}_r \sigma_r^{-1} \\ \vec{a}_2^T \vec{u}_1 \sigma_1^{-1} & \vec{a}_2^T \vec{u}_2 \sigma_2^{-1} & \cdots & \vec{a}_2^T \vec{u}_r \sigma_r^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{a}_n^T \vec{u}_1 \sigma_1^{-1} & \vec{a}_n^T \vec{u}_2 \sigma_2^{-1} & \cdots & \vec{a}_n^T \vec{u}_r \sigma_r^{-1} \end{bmatrix} = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_n \end{bmatrix}$$

Kemudian, ruas kiri dan ruas kanan dicocokkan sehingga diperoleh

$$\vec{v}_1 = \begin{bmatrix} \vec{a}_1^T \vec{u}_1 \sigma_1^{-1} & \vec{a}_1^T \vec{u}_2 \sigma_2^{-1} & \cdots & \vec{a}_1^T \vec{u}_r \sigma_r^{-1} \end{bmatrix}$$

$$= \vec{a}_1^T \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1^{-1} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^{-1} \end{bmatrix}$$

$$= \vec{a}_1^T US^{-1}$$

Hal yang sama dilakukan untuk $\vec{v}_2$, $\vec{v}_3$, …, $\vec{v}_n$ dan diperoleh $\vec{v}_2 = \vec{a}_2^T US^{-1}$, $\vec{v}_3 = \vec{a}_3^T US^{-1}$, …, $\vec{v}_n = \vec{a}_n^T US^{-1}$.

Jadi, misalkan diberikan suatu vektor *query*, $\vec{q}$ yang berisi frekuensi kemunculan kata di *query*. Vektor dokumen untuk vektor *query* tersebut adalah $\vec{q}^T US^{-1}$. Selanjutnya, hasil kali titik antara vektor dokumen untuk *query* dan vektor dokumen dalam koleksi dokumen dapat dihitung.

III.30 Contoh Penggunaan Teorema *Singular Value Decomposition*

Diketahui (7)

Q : "gold silver truck"
D_1 : "Shipment of gold damaged in a fire"
D_2 : "Delivery of silver arrived in a silver truck"
D_3 : "Shipment of gold arrived in a truck"

Matriks *terms-documents*, A yang terbentuk adalah

term	*D1*	*D2*	*D3*
a	1	1	1
arrived	0	1	1
damaged	1	0	0
delivery	0	1	0
fire	1	0	0
gold	1	0	1
in	1	1	1
of	1	1	1
shipment	1	0	1
silver	0	2	0
truck	0	1	1

Hasil dekomposisi matriks A adalah A sebagai perkalian dari USV^T.

Dalam contoh ini, A adalah perkalian dari

$$\begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2625 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \begin{bmatrix} 4.0989 & 0 & 0 \\ 0 & 2.3616 & 0 \\ 0 & 0 & 1.2737 \end{bmatrix} \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & -0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Low-dimensional space yang diperoleh dari SVD dibangun oleh beberapa vektor eigen dari $A^T A$. Banyaknya vektor eigen tersebut (k buah) ditentukan bebas namun $k <$ rank dari A. k juga merupakan dimensi dari *low-dimensional space*.

Misalkan, matriks kata-dokumen, A didekomposisi SVD menjadi $A = USV^T$. Kemudian $S_k = diag(\sigma_1, \sigma_2, \ldots, \sigma_k)$, $U_k = (u_1, u_2, \ldots, u_k)$ dan $V_k = (v_1, v_2, \ldots, v_k)$. Maka

$$A_k = U_k \; S_k \; V_k^T \quad \text{.....(III.8)}$$

A_k adalah matriks dengan rank k, yang merupakan aproksimasi dari matriks A. Baris-baris pada matriks V merupakan vektor-vektor yang mewakili dokumen-dokumen.

Di dalam contoh ini dipilih nilai k = 2. Sekarang diperoleh $A_2 = U_2 \; S_2 \; V_2^T$. Dengan k = 2 yang dipilih, maka perkalian matriks yang baru adalah

$$\begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ 0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} 4.0989 & 0 \\ 0 & 2.3616 \end{bmatrix} \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & -0.2469 \end{bmatrix}$$

Selanjutnya, vektor *query* q^T dibentuk dengan cara yang sama dengan membentuk matriks A. Vektor *query* dipetakan ke dalam ruang berdimensi 2 (*low-dimensional space*) dengan transformasi $q^T U_2 S_2{}^{-1}$.

Ini memberikan hasil

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ 0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} 0.2440 & 0 \\ 0 & 0.4234 \end{bmatrix} = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Baris dari matriks V_2 merupakan koordinat dari dokumen, sehingga

$D_1 = \begin{pmatrix} -0.4945 & 0.6492 \end{pmatrix}$
$D_2 = \begin{pmatrix} -0.6458 & -0.7194 \end{pmatrix}$
$D_3 = \begin{pmatrix} -0.5817 & -0.2469 \end{pmatrix}$

Sekarang dapat dihitung hasil kali titik antara vektor *query* yang sudah dipetakan dengan masing-masing vektor dokumen. Hasilnya adalah sebagai berikut

$$D_1 = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$D_2 = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$D_3 = \frac{(-0.2140)(-0.5817) + (-0.1821)(-0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.5817)^2 + (-0.2469)^2}} = 0.9543$$

D_1, D_2, D_3 merupakan hasil kali titik vektor query dengan vektor dokumen 1, dokumen 2, dokumen 3 berurutan (7).

Dari ketiga nilai tersebut dapat disimpulkan bahwa urutan relevansi untuk dokumen dari yang paling relevan adalah

1. Dokumen ke-2
2. Dokumen ke-3
3. Dokumen ke-1

Dokumen ke-2 merupakan dokumen paling relevan dengan query karena D_2 memiliki nilai *cosinus* yang paling besar atau θ antara *query* dan D_2 paling kecil.

Bab IV Analisis Perangkat Lunak

IV.1 Deskripsi Umum Analisis

Dalam tesis ini dikembangkan perangkat lunak yang diberi nama *Information Retrieval using Latent Semantic Indexing* (Matriulasi). Selanjutnya, perangkat lunak tersebut disebut sebagai perangkat lunak Matriulasi.

Pada tahapan analisis ini diidentifikasi fungsionalitas-fungsionalitas dari perangkat lunak Matriulasi. Setelah diperoleh analisis maka kemudian dibangun sebuah model perangkat lunak yang menjawab kebutuhan yang ada.

Perangkat lunak Matriulasi adalah perangkat lunak aplikasi yang berorientasi objek dan dikembangkan dengan menggunakan bahasa pemrograman berorientasi objek. Pengembangan perangkat lunak Matriulasi dilakukan dengan menggunakan kakas UML (*Unified Modeling Language*) yang mengakomodasi pengembangan perangkat lunak berorientasi objek.

UML adalah bahasa pemodelan visual serba guna yang digunakan untuk menspesifikasikan, menvisualisasikan, mengkonstruksi , dan mendokumentasikan rancangan suatu sistem perangkat lunak.

Ada tujuh jenis diagram yang digunakan dalam UML, yaitu *use case diagram*, *activity diagram*, *sequence diagram*, *class diagram*, *collaboration diagram*, *component diagram* dan *deployment diagram*.

Pada tahap analisis, diagram UML yang digunakan adalah *use case diagram* dan *sequence diagram*. *Use case diagram* digunakan untuk menggambarkan kelakuan sistem atau subsistem seperti yang terlihat oleh pengguna luar (4).

Dalam *use case diagram*, digambarkan aktor, *use case* dan relasi antar keduanya. Aktor menggambarkan peran yang dimainkan oleh pengguna *use case* pada saat melakukan interaksi dengan aktor tersebut. *Use case* mendeskripsikan aksi yang dilakukan oleh sistem. Sedangkan *sequence diagram* menggambarkan interaksi antar objek dan aktornya. *Use case* merupakan aspek dinamis dari suatu perangkat lunak.

Diagram lain yang digunakan pada tahap analisis adalah *sequence diagram*. *Sequence diagram* memodelkan interaksi objek berdasarkan urutan waktu dan menggambarkan kelakukan objek dalam *use case* (4).

IV.2 Deskripsi Global Spesifikasi Perangkat Lunak Matriulasi

Analisis perangkat lunak Matriulasi terdiri dari beberapa bagian, yaitu: spesifikasi perangkat lunak, fungsionalitas perangkat lunak, batasan perangkat lunak Matriulasi, kebutuhan antarmuka eksternal, dan kebutuhan fungsional.

IV.2.1 Spesifikasi Perangkat Lunak Matriulasi

Matriulasi adalah *information retrieval system* yang menerima *query*, memproses *query* dan memberikan hasil analisis berupa ranking dokumen berdasarkan tingkat relevan dengan *query*. Pemrosesan *query* menggunakan konsep *latent semantic indexing* (subbab III.21 dan III.22).

IV.2.2 Fungsionalitas Perangkat Lunak Matriulasi

Fungsionalitas-fungsionalitas dari perangkat lunak Matriulasi adalah:

(1) Membaca file yang berisi koleksi dokumen. Dokumen ini adalah tulisan, artikel tentang tema yang bebas dalam format file teks.
(2) Melakukan pemindaian untuk setiap kata dalam setiap dokumen. Dalam pemindaian tersebut, dilakukan pembuangan *stop words* dari dokumen.
(3) Mengerjakan *stemming* untuk dokumen yang sudah bebas *stop words*. Algoritma *stemming* yang dipakai adalah algoritma *Porter Stemming*.

(4) Membangun matriks kata-dokumen dari dokumen yang sudah di-*stemming*.
(5) Melakukan proses dekomposisi nilai singular (*singular value decomposition*) untuk matriks kata-dokumen.
(6) Melakukan pemetaan vektor *query* ke ruang vektor baru (*low-dimensional space*) yang berdimensi k, dengan $k < rang$ matriks kata-dokumen.
(7) Mengurutkan dokumen-dokumen sesuai dengan dengan tingkat relevansi vektor *query* dengan vektor-vektor dokumen.
(8) Menghitung performansi dari perangkat lunak dengan menghitung parameter *recall*, *precision*, dan *non-interpolated average precision* (NIAP) (5).

IV.2.3 Batasan Perangkat Lunak Matriulasi

Batasan perangkat lunak Matriulasi adalah perangkat lunak menganalisis untuk dokumen yang berbahasa Inggris dalam teks file.

IV.2.4 Kebutuhan Antarmuka Eksternal

Kebutuhan antarmuka eksternal pada perangkat lunak Matriulasi adalah meliputi antarmuka perangkat keras.

IV.2.4.1 Antarmuka Perangkat Keras

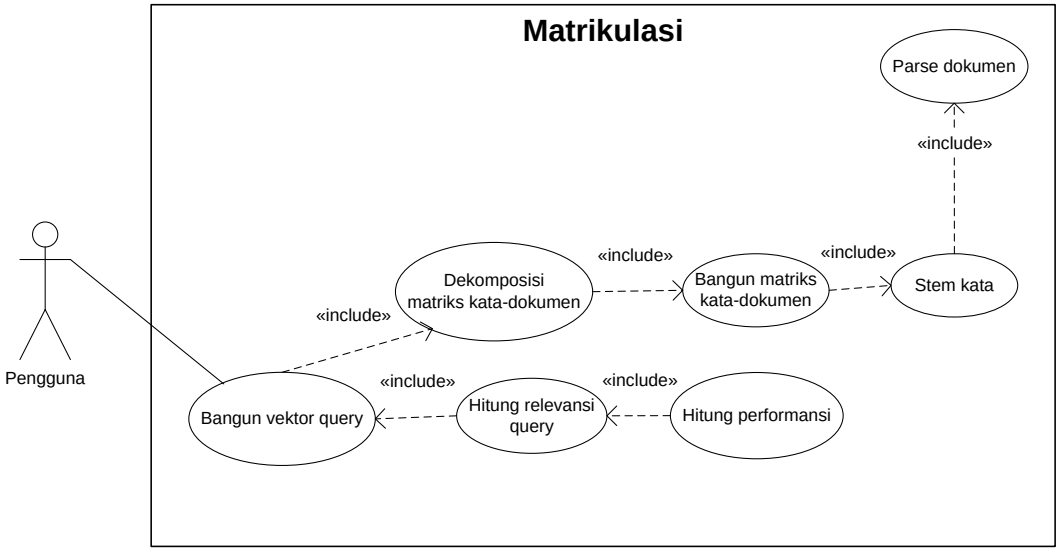
Kebutuhan perangkat lunak agar dapat menjalankan perangkat lunak Matriulasi adalah komputer yang kompatibel dengan IBM, keyboard, dan mouse.

IV.2.5 Kebutuhan Fungsional

Fungsionalitas perangkat lunak Matriulasi digambarkan dalam *use case diagram* seperti pada gambar IV.1. Subbab-subbab berikut menjelaskan fungsionalitas-fungsionalitas dari Matriulasi.

IV.2.5.1 *Use Case Parse* Dokumen (UC1)

Use case ini menggambarkan fungsi perangkat lunak (PL) Matriulasi yaitu membaca dan mem-*parse* koleksi dokumen. Hasil *parse* koleksi dokumen adalah kata-kata (*token-token*) yang disimpan dalam *array*. Koleksi dokumen yang dibaca adalah koleksi dokumen dengan nama file, ADI.ALL.



Gambar IV-1 *Use Case Diagram* dari perangkat lunak Matriulasi

IV.2.5.2 *Use Case Stem* Kata (UC2)

Use case ini memodelkan fungsi PL yaitu melakukan *stemming* untuk *token-token* hasil *parse* koleksi dokumen. *Token* yang merupakan *stop words* dibuang dan *token* hasil *stemming* disimpan.

IV.2.5.3 *Use Case* Bangun Matriks Kata-Dokumen (UC3)

Use case ini menggambarkan fungsi PL yaitu membentuk matriks kata-dokumen. Isi dari matriks kata-dokumen adalah frekuensi kemunculan token-token di dalam koleksi dokumen.

IV.2.5.4 *Use Case* Dekomposisi Matriks Kata-Dokumen (UC4)

Use case ini memodelkan fungsi PL yaitu melakukan dekomposisi matriks dengan menggunakan *Singular Value Decomposition* (SVD).

IV.2.5.5 *Use Case* Bangun Vektor Query (UC5)

Use case ini menggambarkan fungsi PL yaitu menerima *query* dari pengguna kemudian membentuk vektor *query*.

IV.2.5.6 *Use Case* Hitung Relevansi Query (UC6)

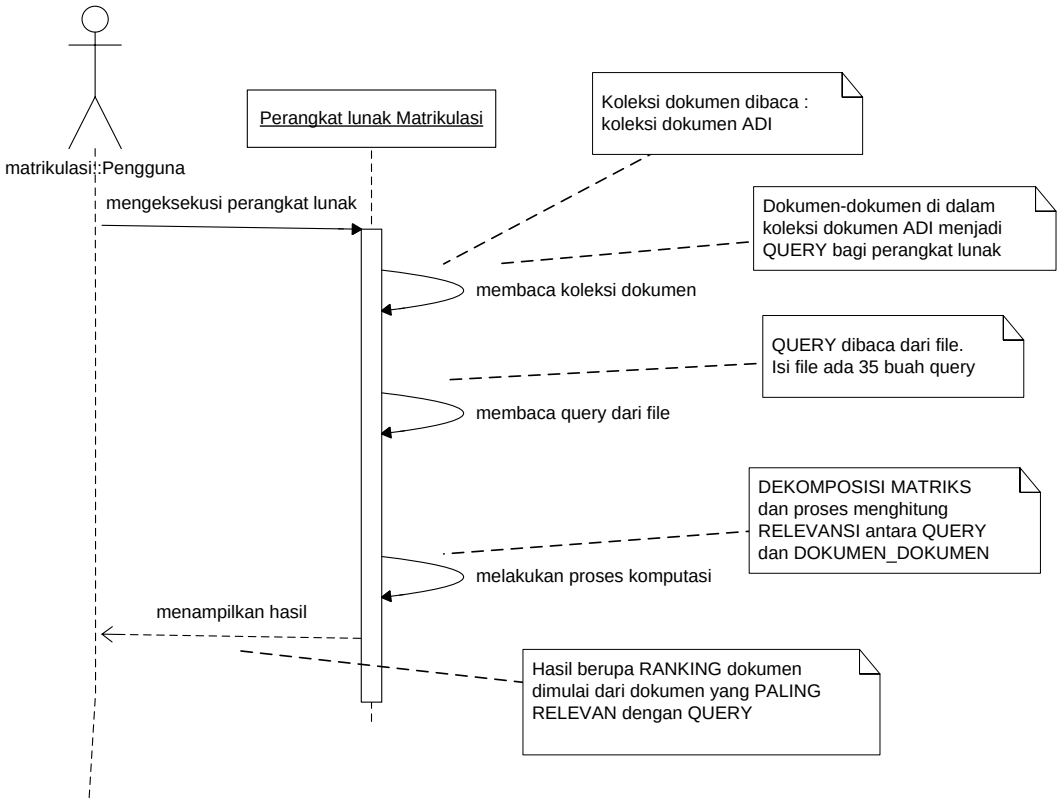
Use case ini memodelkan fungsi PL yaitu menghitung dokumen yang paling relevan dengan *query*.

IV.2.5.7 *Use case* Hitung Performansi (UC7)

Use case ini menggambarkan fungsi PL yaitu mengevaluasi performansi metode LSI dengan menghitung *non-interpolated average precision* (NIAP).

IV.2.6 Pendefinisian Skenario Penggunaan Perangkat Lunak

Skenario untuk pengguna PL Matriulasi secara *high-level* digambarkan dalam sequence diagram pada gambar IV.2.



Gambar IV-2 *Sequence diagram* dari sequence diagram (*high level*)

Urutan proses dalam gambar IV-2 adalah

(1) Pengguna mengeksekusi perangkat lunak (PL) Matriulasi.

(2) PL membaca koleksi dokumen.

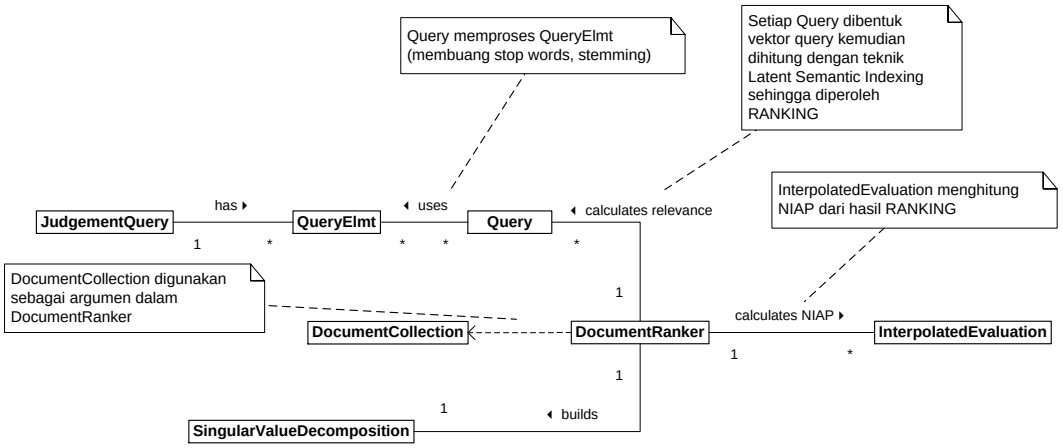
(3) PL membaca query dari file. Dalam file terdapat 35 buah query. Kelompok query ini data uji bagi PL.

(4) PL melakukan proses komputasi untuk menghitung relevansi query dengan dokumen-dokumen.

(5) PL menampilkan hasil berupa ranking dokumen-dokumen, dimulai dari dokumen yang paling relevan dengan *query*. Daftar ranking dokumen ada 35 buah. Masing-masing merupakan daftar ranking bagi setiap *query*. Selain ranking dokumen, juga ditampilkan nilai NIAP dari setiap query.

IV.2.7 Pendefinisian *Class Diagram* Tahap Awal

Class diagram tahap awal dari perangkat lunak Matriulasi digambarkan dalam *class diagram* pada gambar IV-3.



Gambar IV-3 *Class diagram* tahap awal

Kelas-kelas dalam *class diagram* awal digambarkan pada Gambar IV.3 meliputi:

(1) Kelas *JudgementQuery*

Kelas yang mewakili isi dari *query filename* dan *query relevance*.

(2) Kelas *QueryElmt*

Kelas yang mewakili query beserta *query relevance*-nya.

(3) **Kelas *Query***

Kelas yang mewakili *query* untuk perangkat lunak Matriulasi.

(4) **Kelas *DocumentCollection***

Kelas yang berfungsi melakukan *indexing* terhadap suatu koleksi dokumen.

(5) Kelas *DocumentRanker*

Kelas yang berfungsi membuat ranking dari hasil *indexing* suatu koleksi dokumen.

(6) Kelas *SingularValueDecomposition*

Kelas yang berfungsi membangun hasil dekomposisi nilai singular (*Singular Value Decomposition*) dari suatu matriks.

(7) Kelas *InterpolatedEvaluation*

Kelas yang berfungsi menghitung performansi dari ranking yang dihasilkan Matriulasi. Ukuran performansi yang dipakai adalah *non-interpolated average precision* (NIAP).

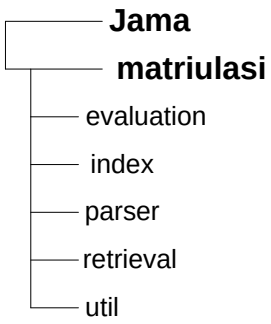
IV.3 Tahap Selanjutnya: Desain Perangkat Lunak Matriulasi

Bab ini membahas hasil analisis perangkat lunak Matriulasi secara garis besar. Dalam bab selanjutnya akan dibahas desain dari perangkat lunak Matriulasi secara mendetil.

Bab V Perancangan Perangkat Lunak Matriulasi

V.1 Perancangan *package* kode program Matriulasi

Kode program Matriulasi dibuat dalam bahasa pemrograman Java™ dan menggunakan kompiler Java™, disebut *Java Developer Kit* (JDK®) versi 1.4.1. Kode program dibagi dalam 2 (dua) *package* utama, yaitu *package* Jama dan *package* matriulasi. Struktur *package* digambarkan pada gambar V-1.



Gambar V-1 Struktur *package* PL Matriulasi

V.1.1 *Package* Jama

Jama merupakan singkatan dari *Java Matrix*®, berisi kelas-kelas yang berfungsi untuk melakukan operasi matriks seperti membuat struktur data matriks dan mendekomposisi suatu matriks (*singular value decomposition*).

V.1.2 *Package* matriulasi

Package ini merupakan inti (*core*) dari program Matriulasi. *Package* matriulasi dibagi menjadi 5 (lima) buah *sub-package*, yaitu evaluation, index, parser, retrieval, dan util. Penjelasan mengenai masing-masing *sub-package* adalah

- evaluation berisi kelas-kelas yang berfungsi menghitung performansi hasil ranking;
- index berisi kelas-kelas yang berfungsi melakukan proses *indexing* terhadap koleksi dokumen;

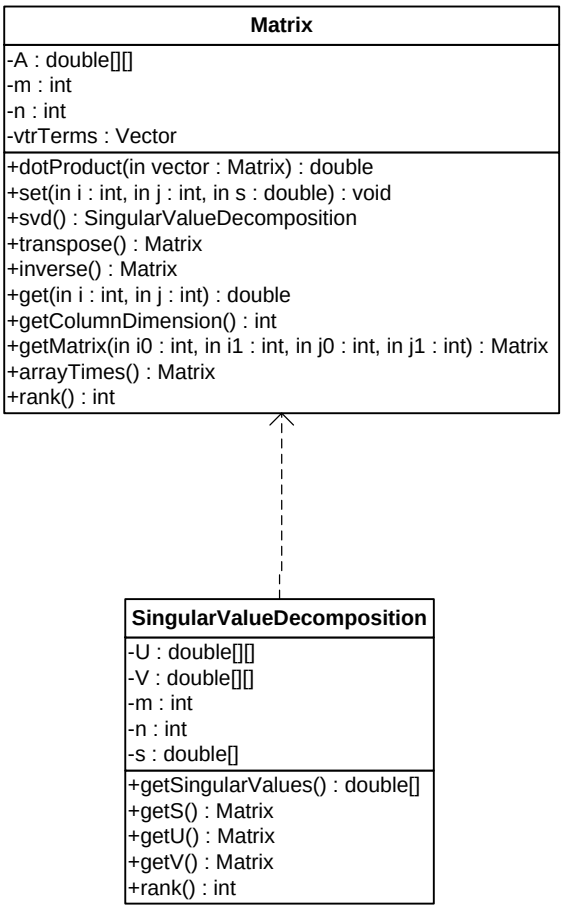
- parser berisi kelas-kelas yang berfungsi mengerjakan *stemming* terhadap hasil *indexing*;
- retrieval berisi kelas-kelas yang berfungsi melakukan komputasi untuk memperoleh nilai relevansi *query* dengan dokumen.
- util berisi kelas-kelas yang berfungsi sebagai *utility* dalam mengerjakan proses-proses.

V.2 Perancangan *Class Diagram* Kode Program Matriulasi

Pada bab sebelumnya, gambar IV-3 memperlihatkan *class diagram* awal. *Class diagram* awal pada gambar IV-3 merupakan garis besar kelas-kelas yang membangun perangkat lunak Matriulasi.

Selanjutnya, *class diagram* awal tersebut dipecah menjadi beberapa *class diagram* untuk setiap *package*.

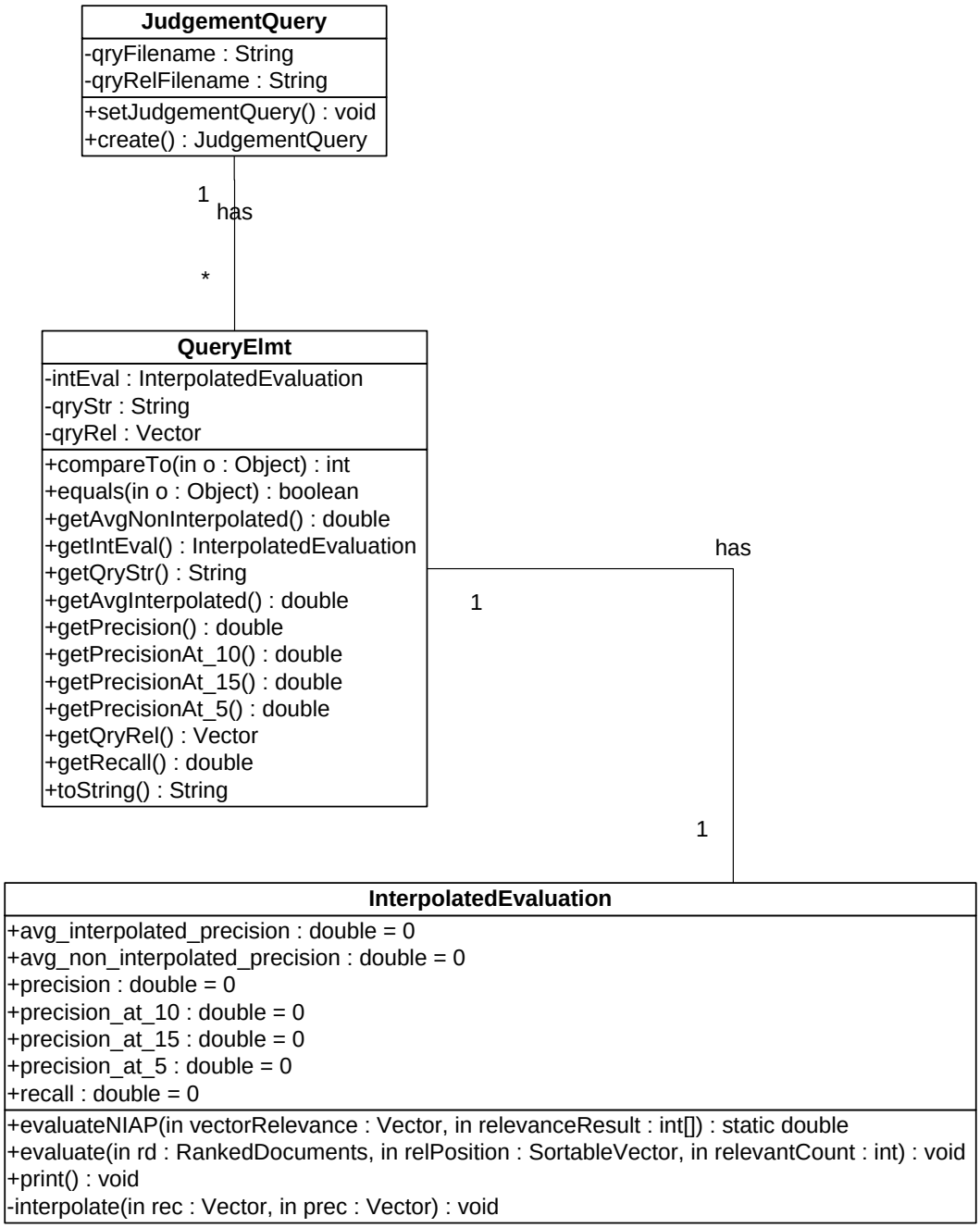
V.2.1 *Class Diagram* Package Jama



Gambar V-2 *Class diagram* package Jama

Gambar V-2 menggambarkan kelas *SingularValueDecomposition* yang mempunyai hubungan ketergantungan (*dependency*) dengan kelas *Matrix*. Hal ini disebabkan kelas *SingularValueDecomposition* mempunyai *method* dengan kelas Matrix sebagai argumennya.

V.2.2 *Class Diagram* Package matriulasi.evaluation

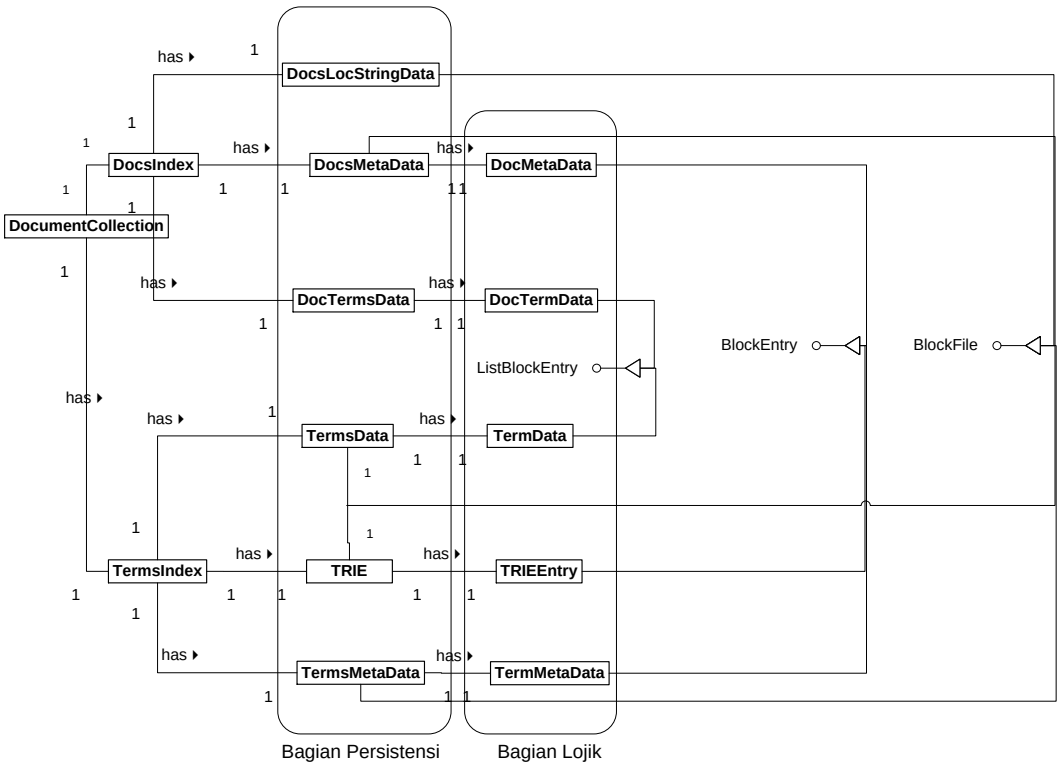


Gambar V-3 *Class diagram* package matriulasi.evaluation

Kelas *JudgementQuery* mempunyai elemen-elemen dari kelas *QueryElmt*. Dan masing-masing kelas *QueryElmt* mempunyai kelas *InterpolatedEvaluation*. Kelas

InterpolatedEvaluation mempunyai fungsionalitas untuk menghitung nilai *non-interpolated average precision* (NIAP).

V.2.3 *Class Diagram* Package matriulasi.index



Gambar V-4 *Class diagram* package matriulasi.index

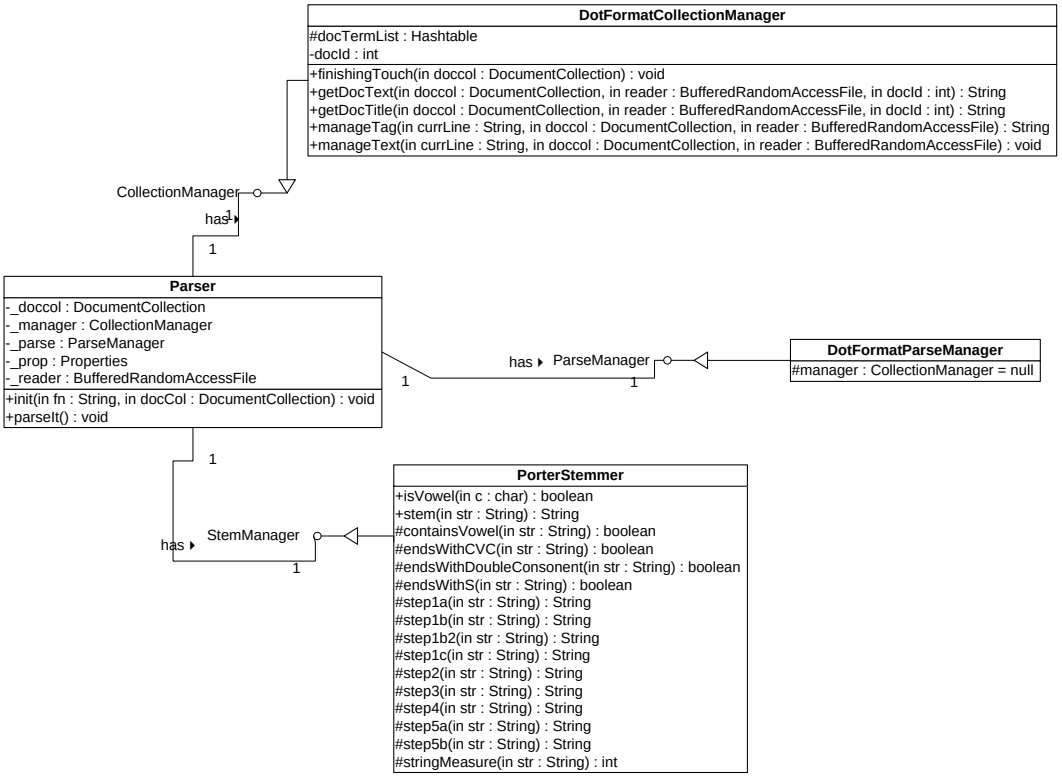
DocumentCollection merupakan gabungan dari (20)

(1) *Document Index* (*DocsIndex*), yang menyimpan data-data tentang dokumen;

(2) *Terms Index*, yang menyimpan informasi mengenai kata-kata (*terms*).

Detil dari kelas-kelas pada gambar V-4 dijelaskan di bagian Lampiran

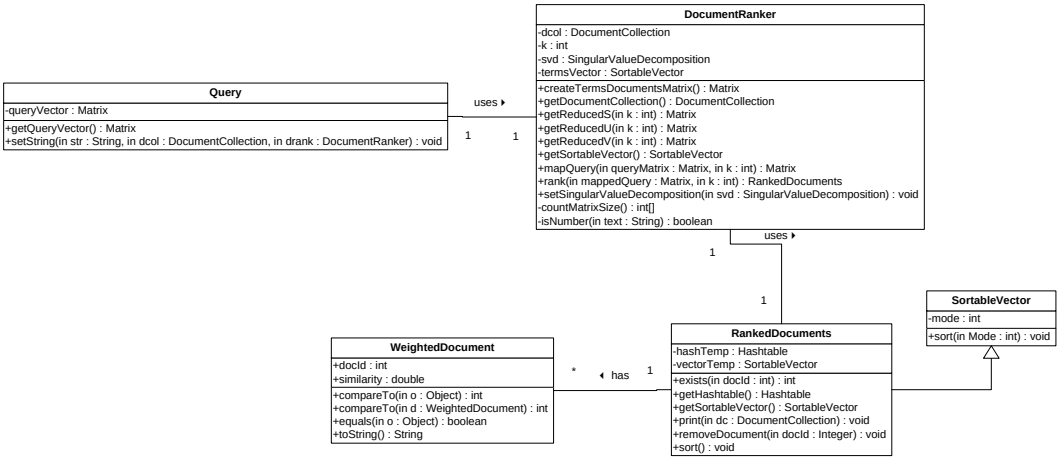
V.2.4 *Class Diagram* Package matriulasi.parser



Gambar V-5 *Class diagram* package matriulasi.parser

Dalam *package* matriulasi.parser, kelas *PorterStemmer* merupakan kelas utama yang berfungsi mem-*parse* dokumen-dokumen sesuai dengan *use case parse* dokumen (UC1).

V.2.5 *Class Diagram* Package matriulasi.retrieval



Gambar V-6 *Class diagram* package matriulasi.retrieval

Dalam *package* matriulasi.retrieval, Matriulasi menggunakan kelas *DocumentRanker* untuk memproses kelas *Query*. Kemudian kelas *DocumentRanker* menggunakan kelas *RankedDocuments* untuk menyimpan hasil perangkingan. Kelas *RankedDocuments* itu sendiri mewarisi kelas *SortableVector*.

V.3 Pembangunan Perangkat Lunak

Tahap selanjutnya adalah membangun perangkat lunak berdasarkan desain kelas yang telah dijabarkan dari subbab-subbab sebelumnya. Perangkat lunak Matriulasi merupakan aplikasi desktop yang dibangun dengan bahasa pemrograman Java™. Perangkat lunak Matriulasi dijalankan di *console* dan tidak memiliki tampilan antarmuka. Perangkat Matriulasi digunakan untuk tujuan eksperimen dan bukan tujuan komersil.

Langkah berikutnya adalah mengadakan eksperimen untuk menguji Matriulasi. Pengujian dilakukan untuk memperoleh nilai *non-interpolated average precision* (NIAP) dalam beberapa kasus. Bab selanjutnya membahas eksperimen PL Matriulasi.

Bab VI Evaluasi Perangkat Lunak Matriulasi

VI.1 Evaluasi *Information Retrieval System* Secara Umum

Performansi *Information Retrieval System* dapat diukur dalam 6 (enam) kualitas (18), yaitu:

- Banyak dokumen relevan yang berhasil di-*retrieve*.
- *Response time* antara setelah memasukkan *query* dan menerima *response* dari sistem.
- Efektivitas dari tampilan keluaran (contoh, dokumen-dokumen di-*ranking* dari relevansi terbesar menuju terkecil).
- Usaha yang melibatkan pengguna dalam menemukan jawaban bagi *request*-nya.
- Nilai *recall* dari sistem, yaitu

$$recall = \frac{banyak\ dokumen\ relevan\ dan\ ter-retrieved}{banyak\ semua\ dokumen\ relevan\ di\ koleksi\ dokumen} \quad \text{.....(VI.1)}$$

Contoh:

Misalkan terdapat sebuah *query* bagi sebuah IR *system*. Keseluruhan dokumen relevan dalam koleksi dokumen adalah dokumen ke-1 (D_1), dokumen ke-5 (D_5), dokumen ke-8 (D_8), dan dokumen ke-10 (D_{10}).

IR *system* tersebut memberikan hasil ranking:

1. D1
2. D2
3. D8
4. D20
5. D15

Berapakah nilai *recall* dari IR *system* di atas?

Jawab:

$$recall = \frac{banyak\ dokumen\ relevan\ dan\ ter-retrieved}{banyak\ semua\ dokumen\ relevan\ di\ koleksi\ dokumen} = \frac{2}{4} = \frac{1}{2}$$

- Nilai *precision* dari sistem, yaitu

$$precision = \frac{banyak\ dokumen\ relevan\ dan\ ter-retrieved}{semua\ dokumen\ ter-retrieved} \quad \text{.....(VI.2)}$$

Contoh:

Misalkan terdapat sebuah *query* bagi sebuah IR *system*. Keseluruhan dokumen relevan dalam koleksi dokumen adalah dokumen ke-1 (D_1), dokumen ke-5 (D_5), dokumen ke-8 (D_8), dokumen ke-10 (D_{10}).

IR *system* memberikan hasil ranking:

1. D1
2. D2
3. D8
4. D20
5. D15

Berapakah nilai *precision* dari IR *system* di atas?

Jawab:

$$precision = \frac{banyak\ dokumen\ relevan\ dan\ ter-retrieved}{semua\ dokumen\ ter-retrieved} = \frac{2}{5}$$

Selain 6 (enam) point di atas, ada juga parameter lain untuk mengukur performansi yaitu *non-interpolated average precision* (*NIAP*).

$$NIAP = \frac{\sum precision\ dari\ dokumen\ relevan\ yang\ ditemukan}{banyak\ semua\ dokumen\ relevan\ koleksi\ dokumen} \quad \text{.....(VI.3)}$$

Contoh:

Misalkan ada 2 (dua) buah IR *system* yang hendak diukur performansinya. Diketahui juga sebuah *query* dan keseluruhan dokumen relevan dalam koleksi dokumen adalah dokumen ke-1 (D_1), dokumen ke-5 (D_5), dokumen ke-8 (D_8), dokumen ke-10 (D_{10}).

Kedua IR *system* memberikan hasil berupa *ranking* dokumen-dokumen relevan, sebagai berikut:

Hasil *Ranking* IR *system* ke-1	**Hasil *Ranking* IR *system* ke-2**
1. Dokumen ke-99	1. Dokumen ke-1
2. Dokumen ke-95	2. Dokumen ke-5
3. Dokumen ke-88	3. Dokumen ke-8
4. Dokumen ke-71	4. Dokumen ke-10
⋮	⋮
997. Dokumen ke-1	997. Dokumen ke-99
998. Dokumen ke-5	998. Dokumen ke-45
999. Dokumen ke-8	999. Dokumen ke-34
1000. Dokumen ke-10	1000. Dokumen ke-43

Manakah dari kedua IR system tersebut yang terbaik?

Jawab:

IR *system* yang ke-1 memberikan nilai $recall = \frac{1}{1} = 1$, $precision = \frac{4}{1000}$.

IR *system* yang ke-2 memberikan nilai $recall = \frac{1}{1} = 1$, $precision = \frac{4}{1000}$.

Padahal hasil ranking menunjukkan bahwa IR system ke-2 lebih baik daripada IR system ke-1 karena dokumen relevan terdapat pada ranking ke-1, ke-2, ke-3, dan ke-4.

Apabila nilai *NIAP* dihitung untuk kedua IR *system* di atas, diperoleh bahwa

IR *system* ke-1 memberikan nilai $NIAP = \frac{1+1+1+1}{4} = 1$ dan IR *system* ke-2 memberikan nilai $NIAP = \frac{\frac{1}{997} + \frac{2}{998} + \frac{3}{999} + \frac{4}{1000}}{4} \approx 0.0025$.

Jadi *NIAP* IR *system* ke-2 jauh lebih besar daripada *NIAP* IR *system* ke-1 atau dengan kata lain performansi IR *system* ke-2 lebih baik daripada performansi IR *system* ke-1.

VI.2 Koleksi Dokumen

Koleksi dokumen yang digunakan untuk evaluasi perangkat lunak Matriulasi adalah koleksi dokumen ADI. Koleksi dokumen ADI terdiri 82 (delapan puluh dua) dokumen. Sebagai contoh, cuplikan dokumen ke-9 dari koleksi dokumen ADI adalah

```
.I 9
.T
analysis of the role of the computer in the reproduction
and distribution of scientific papers
.A
J. H. KUNEY
.W
the american chemical society has begun
 an analysis of the role of the computer in related aspects
 of the reproduction, distribution, and retrieval of scientific
 information .  initial work will attempt to solve problems
 of photocomposition via computer .
```

Gambar VI-1 Dokumen ke-9 dari koleksi dokumen ADI

Keterangan:

- `.I` menunjukkan nomor dokumen.
- `.T` menunjukkan judul dokumen.
- `.A` menunjukkan pengarang dokumen.
- `.W` menunjukkan isi dokumen

VI.3 Skenario Evaluasi Perangkat Lunak Matriulasi

Berdasarkan point-point yang terdapat dalam konsep *Latent Semantic Indexing* (subbab III.28), kasus uji yang dikenakan untuk perangkat lunak Matriulasi adalah kasus memilih parameter r dalam membangun submatriks. Pada tahap awal, hasil *Singular Value Decomposition* untuk matriks kata-dokumen, A dari koleksi dokumen ADI adalah

$$A = \quad U \qquad\qquad S \qquad\qquad V^T$$

$$A = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$$

dengan k adalah banyaknya nilai singular dari A ($\sigma_1, \sigma_2, \ldots, \sigma_k$). Selanjutnya, dari k buah nilai singular dari A, dipilih r buah nilai singular yang terbesar, yaitu $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, dengan $r < k$.

Kasus uji yang muncul adalah menguji perangkat lunak Matriulasi untuk beberapa nilai r. Dari beberapa nilai r yang diuji, dipilih nilai r yang memberikan nilai *NIAP* maksimum.

Koleksi dokumen ADI mempunyai 82 buah dokumen. Jadi nilai k dari matriks kata-dokumen koleksi dokumen ADI adalah 82. Dalam tesis ini, kasus uji nilai r untuk perangkat lunak Matriulasi adalah nilai $r = 10$, $r = 20$, $r = 30$, $r = 40$, $r = 50$, $r = 60$, $r = 70$, dan $r = 80$.

VI.4 Tujuan Evaluasi Perangkat Lunak Matriulasi

Secara garis besar, tujuan yang hendak dicapai dalam evaluasi adalah

1. Membandingkan nilai rata-rata *NIAP* antara metode *Latent Semantic Indexing* dengan metode *Vector*.
2. Menguji beberapa nilai r, memilih nilai r yang memberikan nilai *NIAP* maksimum dan kemudian mencoba menjelaskan mengapa nilai r tersebut memberikan nilai *NIAP* maksimum.
3. Mencoba menganalisis korelasi frekuensi kata di matriks kata-dokumen dengan angka-angka di dalam matriks hasil *singular value decomposition*.
4. Mencoba menganalisis kata-kata dalam *query* dan sinonim kata yang dimunculkan dalam konsep *Latent Semantic Indexing*.

VI.5 Keluaran Perangkat Lunak Matriulasi

Perangkat lunak Matriulasi melakukan proses pe-*ranking*-an untuk dokumen-dokumen yang relevan dengan *query*. *Ranking* ditentukan oleh nilai relevansi antara vektor *query* dengan vektor dokumen (subbab III.22). *Ranking* pertama

adalah dokumen yang paling relevan dengan *query*, ranking ke-dua adalah dokumen yang relevan ke-2 dengan *query*, dan seterusnya.

Masukan perangkat lunak Matriulasi adalah nilai r (subbab III.28) dan *query-query*. Nilai r dipilih berdasarkan $rank(A)$, dengan A adalah matriks kata-dokumen. Matriks kata-dokumen yang dibentuk dari koleksi dokumen ADI berukuran 895×82. $Rank(A)$ adalah 82. Jadi nilai r yang dapat dipilih $0 < r \leq 82$.

Query-query yang menjadi masukan disimpan dalam file yang selanjutnya dibaca oleh perangkat lunak. Jumlah total *query* adalah 35 (tiga puluh lima) buah. Salah satu contoh *query*, yaitu *query* ke-8 diilustrasikan dalam gambar VI-2.

```
.I 8
.W
Describe information retrieval and indexing in other languages.
What bearing does it have on the science in general?
```

Gambar VI-2 *Query* ke-8 dari koleksi dokumen ADI

Keterangan:

- `.I` menunjukkan nomor *query*.
- `.W` menunjukkan isi *query*.

Keluaran perangkat lunak Matriulasi berupa informasi sebagai berikut:

- Ukuran matriks kata-dokumen, A.
- Kata-kata di dalam matriks kata-dokumen.
- Ukuran matriks U.
- Ukuran matriks S.
- Ukuran matriks V.
- Informasi mengenai *query* yang diproses (secara keseluruhan terdapat 35 buah *query* yang diproses), yaitu
    - Informasi mengenai kata-kata di dalam setiap *query*.
    - Ukuran setiap vektor *query* yang telah dipetakan.
    - Vektor *query* yang telah dipetakan

- Hasil *ranking* setiap dokumen beserta nilai *similarity* (rumus II.3).

Keseluruhan informasi pada point-point di atas ditulis ke dalam file.

```
Matriks Terms-Documents berdimensi : 895 x 82

Term-term dari matriks Terms-Documents adalah :

process  technic  total  catalog  featur  output  format  integr  drawn
ibm  retriev  data  produc  comput  effici  overdu  base  tradit
copi  oper  machin  gap  user  document  librari  compat  organ
...

Matriks U berdimensi : 895 x 82
Matriks S berdimensi : 82 x 82
Matriks V berdimensi : 82 x 82

======================
   QUERY PROCESSING
======================
 QUERY KE-1 =
TERM = retriev FREQUENCY = 1.0
TERM = automat FREQUENCY = 1.0
TERM = make FREQUENCY = 1.0
TERM = problem FREQUENCY = 1.0
TERM = relev FREQUENCY = 1.0
TERM = titl FREQUENCY = 3.0
TERM = content FREQUENCY = 1.0
TERM = articl FREQUENCY = 2.0
TERM = involv FREQUENCY = 1.0
TERM = descript FREQUENCY = 1.0
TERM = approxim FREQUENCY = 1.0

Mapped Query berdimensi : 1 x 50
Mapped Query =
0.156   0.0062  0.0006  0.0740  0.0140  0.0237  ...

HASIL SIMILARITY UNTUK QUERY KE-1:
1.  DOKUMEN NO. 69    SIMILARITY : 0.6959592813405281
2.  DOKUMEN NO. 17    SIMILARITY : 0.3588976149196746
3.  DOKUMEN NO. 39    SIMILARITY : 0.35625813048314936
.
.
.
81.  DOKUMEN NO. 6    SIMILARITY : -0.14009685142140238
82.  DOKUMEN NO. 1    SIMILARITY : -0.15511590747098017

========================================
     Menghitung NIAP untuk query ke-1
========================================
NIAP untuk query ke-1 = 0.24814814814814815

QUERY KE-2 =
TERM = retriev FREQUENCY = 1.0
TERM = data FREQUENCY = 1.0
TERM = actual FREQUENCY = 1.0
.
.
.
Rata-Rata NIAP untuk semua query = 0.3558575175755285
```

Gambar VI-3 Keluaran perangkat lunak Matriulasi untuk $r = 50$

VI.6 Evaluasi Beberapa Nilai r

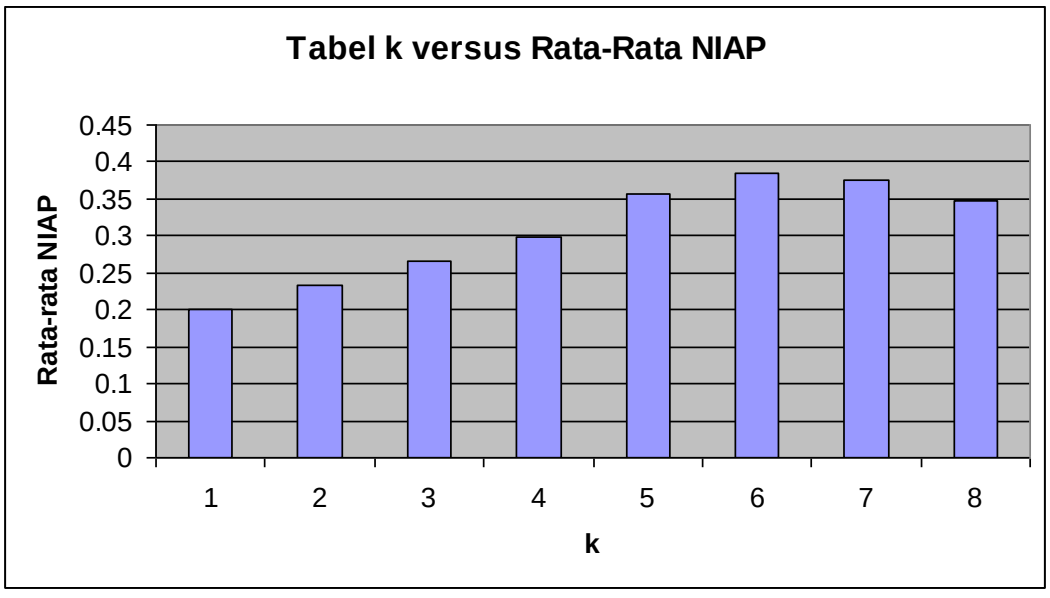
Dalam mencari nilai *NIAP*, perangkat lunak Matriulasi dievaluasi untuk beberapa nilai r. Dari beberapa nilai r yang dievaluasi, dipilih nilai r yang memberikan nilai rata-rata *NIAP* maksimum.

Koleksi dokumen ADI mempunyai 82 buah dokumen. Jadi nilai maskimum k yang dapat dipilih dari matriks kata-dokumen koleksi dokumen ADI adalah 82. Dalam tesis ini, evaluasi nilai r untuk perangkat lunak Matriulasi adalah nilai $r = 10$, $r = 20$, $r = 30$, $r = 40$, $r = 50$, $r = 60$, $r = 70$, dan $r = 80$.
Hasil evaluasi matriks kata-dokumen seperti pada gambar IV-4.

r	10	20	30	40	50	60	70	80
rata-rata *NIAP*	0.20127	0.23317	0.26648	0.29792	0.35586	0.38532	0.37543	0.34826

Gambar VI-4 Nilai r terhadap nilai Rata-rata *NIAP*

Dari gambar VI-5 dapat disimpulkan bahwa nilai *NIAP* maksimum dicapai pada saat $r = 60$, yaitu 0.38532.



Gambar VI-5 Tabel rata-rata NIAP terhadap k

Point penting yang dapat dijelaskan adalah pembentukan submatriks dengan $r = 60$ telah membuang nilai-nilai singular yang merupakan *noise* sehingga diperoleh performasi maksimum dari perangkat lunak . Dengan memilih $r = 60$, hasil dekomposisi nilai singular (SVD) matriks sebelumnya yaitu matriks U berukuran 895×82, matriks S berukuran 82×82, dan matriks V^T berukuran 82×82 menjadi matriks U baru berukuran 895×60, matriks S baru berukuran 60×60, dan matriks V^T baru berukuran 60×82. Matriks U baru dibentuk dengan mengambil vektor-vektor kolom matriks U sebanyak 60 buah vektor kolom. Matriks S baru dibentuk dengan mengambil 60 buah nilai singular dari matriks S. Matrik V^T baru dibentuk dengan mengambil 60 buah vektor baris dari matriks V^T. Nilai singular pada diagonal utama matriks S dimulai dari elemen ke-61 sampai dengan elemen ke-82 telah dibuang dan setelah nilai-nilai singular tersebut dibuang, nilai rata-rata *NIAP* maksimum diperoleh. Hal ini menunjukkan bahwa 22 (dua puluh dua) buah nilai singular pada matriks S adalah *noise* dan dapat dibuang untuk meningkatkan performansi perangkat lunak Matriulasi.

VI.7 Perbandingan Hasil Metode LSI dan Metode Vektor

Metode LSI dengan menggunakan vektor pembobotan yaitu frekuensi kemunculan kata (*term-frequency*) dibandingkan dengan metode vektor yang menggunakan vektor pembobotan yang sama ternyata memberikan hasil:

- *NIAP* metode LSI adalah 0.3853 dan *NIAP* metode vektor adalah 0.3286 sehingga performansi IR *system* dipandang dari parameter *NIAP* dengan metode LSI lebih baik daripada metode vektor.
- Bila metode LSI digunakan maka terjadi peningkatan sebesar 17.25 % daripada bila metode vektor digunakan.

VI.8 Pengkajian Polisemi dan Sinonim

Misalkan terdapat kata ‘komputer’ dan kata ‘hardware’ pada koleksi dokumen yang terdiri dari 4 buah dokumen. Polisemi dan sinonim antara kata ‘komputer’ dan kata ‘hardware’ dapat dihitung dengan langkah-langkah sebagai berikut:

1. Membangun vektor yang berisi frekuensi kemunculan kata pada dokumen-dokumen.

   Contoh:

   $$\vec{x}_1 = \begin{matrix} \\ komputer \end{matrix} \begin{bmatrix} dokumen1 & dokumen2 & dokumen3 & dokumen4 \\ 2 & 0 & 1 & 2 \end{bmatrix}$$

   $$\vec{x}_2 = \begin{matrix} \\ hardware \end{matrix} \begin{bmatrix} dokumen1 & dokumen2 & dokumen3 & dokumen4 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

   Vektor $\vec{x}_1$ menunjukkan frekuensi kemunculan kata ‘komputer’ pada dokumen 1, dokumen 2, dokumen 3, dan dokumen 4.

   Vektor $\vec{x}_2$ menunjukkan frekuensi kemunculan kata ‘hardware’ pada dokumen 1, dokumen 2, dokumen 3, dan dokumen 4.

2. Menghitung nilai *similarity* (subbab II.2) antara $\vec{x}_1$ dan $\vec{x}_2$, yaitu

   $$\frac{5}{\sqrt{2^2+0^2+1^2+2^2}\sqrt{1^2+0^2+1^2+1^2}} = 0.9622$$

Secara umum, langkah-langkah menghitung nilai hasil kali titik antara 2 (dua) kata adalah sebagai berikut:

1. Diketahui A adalah matriks kata-dokumen. Setelah langkah-langkah pada subbab III.29 dikerjakan, matriks U_r, S_r, dan V_r^T diperoleh, dengan $r < rank(A)$.
2. Matriks A_r adalah matriks kata-dokumen setelah *noise* dibuang $A_r = U_r\ S_r\ V_r^T \neq A$.
3. Vektor baris ke-i dari matriks A_r adalah vektor yang menunjukkan frekuensi kemunculan kata ke-i dalam dokumen-dokumen.
4. *Similarity* antara 2 (dua) kata untuk semua kata dalam dokumen dapat dihitung dengan

$$A_r\ A_r^T = U_r\ S_r\ V^T\ (U_r\ S_r\ V^T)^T$$

$$\Leftrightarrow A_r\ A_r^T = U_r\ S_r\ V^T V\ S_r^T U_r^T$$

$$\Leftrightarrow A_r\ A_r^T = U_r\ S_r\ S_r^T U_r^T$$

$$\Leftrightarrow A_r\ A_r^T = U_r\ S_r\ (U_r\ S_r)^T \quad \text{.....(VI.4)}$$

Contoh:

Vektor kolom ke-2 dari matriks kata-dokumen menunjukkan frekuensi kemunculan kata-kata yang terdapat dalam dokumen ke-2. Gambar VI-6 menunjukkan cuplikan beberapa kata di dokumen ke-2 matriks kata-dokumen. Frekuensi kemunculan kata ke-169 pada dokumen ke-2 di matriks kata-dokumen adalah 0 (nol) seperti pada gambar VI.6. Selanjutnya, matriks kata-dokumen didekomposisi SVD menjadi U, S, dan V^T. Matriks U merupakan matriks dengan kolom-kolom adalah vektor kata. Frekuensi kemunculan kata ke-169 di dokumen ke-2 pada matriks U adalah 0.5992 seperti pada gambar VI-7. Hal ini menunjukkan bahwa kata ke-169 di dokumen ke-2 mempunyai frekuensi kemunculan tidak sama dengan nol meskipun pada matriks kata-dokumen frekuensi kemunculan adalah nol.

	$Dokumen\ ke-2$	
⋮	⋮	⋮
$Kata\ ke-67$	0	1
$Kata\ ke-68$	0	1
$Kata\ ke-69$	0	1
⋮	⋮	⋮
$Kata\ ke-89$	0	1
$Kata\ ke-90$	0	2
⋮	⋮	⋮
$Kata\ ke-169$	0	0
⋮	⋮	⋮
$Kata\ ke-895$	0	0

Gambar VI-6 Cuplikan beberapa kata di dokumen ke-2 matriks kata-dokumen

	Dokumen ke − 2		
⋮	⋮	⋮	⋮
Kata ke − 67	0.0028	0.0010	0.0030
Kata ke − 68	0.0028	0.0010	0.0030
Kata ke − 69	0.0028	0.0010	0.0030
⋮	⋮	⋮	⋮
Kata ke − 89	0.0028	0.0010	0.0030
Kata ke − 90	0.0137	0.0065	0.0124
⋮	⋮	⋮	⋮
Kata ke − 169	0.2207	0.5992	− 0.2278
⋮	⋮	⋮	⋮
Kata ke − 895	0.0014	0.0002	0.0021

Gambar VI-7 Cuplikan beberapa kata di dokumen ke-2 matriks U

Kemudian pertanyaan yang muncul adalah:
"Kata apakah di dokumen ke-2 yang mempunyai nilai *similarity* paling tinggi dengan kata ke-169 ?"

Kata ke-169 dalam koleksi dokumen ADI adalah kata 'index', hendak dicari kata-kata dalam dokumen ke-2 yang memiliki nilai *similarity* paling tinggi. Nilai r yang memberikan nilai *NIAP* maksimum berdasarkan *trial-and-error* adalah 60. Jadi nilai $r = 60$ yang digunakan dalam perhitungan.

Langkah-langkah untuk menghitung nilai *similarity* antara kata ke-169 dengan kata-kata di dokumen ke-2 adalah

1. Berdasarkan rumus VI.4, dihitung matriks $U_r\, S_r = U_{60}\, S_{60}$.
2. Dari matriks $U_{60}\, S_{60}$ baris ke-169 diambil, $\vec{x}_{169}$. $\vec{x}_{169}$ merupakan vektor yang berisi frekuensi kemunculan untuk kata ke-169, yaitu 'index'.
3. Dari matriks kata-dokumen, terlihat bahwa kata-kata yang ada di dokumen ke-2 dimulai dari kata ke-69 sampai dengan kata ke-90. Jadi baris ke-69 sampai dengan kata ke-90 diambil dan dibentuk matriks M.
4. Nilai *similarity* dihitung antara $\vec{x}_{169}$ dan M.

Hasil perhitungan *similarity* antara kata ‘index’ dan kata-kata di dokumen ke-2 dapat dibaca pada gambar IV-8. Kata di dokumen ke-2 yang memiliki *similarity* paling besar dengan kata ‘index’ adalah kata ‘includ’. Apabila koleksi dokumen ADI ditilik, diperoleh bahwa kata ‘index’ muncul bersamaan sebanyak 3 (tiga) kali di dokumen yang sama, yaitu dokumen ke-6 bagian Title, kata ‘including‘ dan kata ‘index’, kata ‘include‘ dan kata index di bagian Isi; dokumen ke-15 bagian isi, kata ‘including’ dengan kata ‘indexes’.

No.	**Kata-Kata di Dokumen ke-2**	***Similarity* dengan Kata ‘index’**
1	twodimension	-0.0002
2	extract	-0.0002
3	pattern	-0.0002
4	structur	0.0054
5	line	-0.0008
6	present	0.1558
7	method	0.2303
8	automat	0.1536
9	search	0.1941
10	analyz	-0.0002
11	excerpt	-0.0002
12	includ	0.3423
13	consist	-0.0018
14	comparison	0.1630
15	chemic	0.0611
16	procedur	0.0766
17	identif	-0.0002
18	queri	-0.0002
19	dimension	-0.0002
20	request	0.0178
21	match	0.0002
22	graph	-0.0002
23	syntact	-0.0002
24	applic	-0.0021

Gambar VI-8 Tabel *Similarity* antara kata ‘index’ dan kata di dokumen ke-2

Bab VII Kesimpulan Dan Saran

Dalam tesis ini dikembangkan metode *Latent Semantic Indexing* untuk *Information Retrieval System*. Beberapa point yang dapat disimpulkan dalam tesis ini adalah

1. Metode LSI menghasilkan performansi yang lebih baik daripada metode vektor dalam IR *system* karena metode LSI memasukkan faktor polisemi dan sinonim dalam komputasi metode LSI. Faktor polisemi dan sinonim diperhitungkan ketika proses pembentukan vektor kata-vektor kata untuk ruang kolom dari matriks kata-dokumen.
2. Nilai *rank* yang direkomendasikan untuk koleksi dokumen ADI adalah sekitar 60. Dengan pemilihan nilai *rank* sekitar 60, rata-rata *NIAP* maksimum dapat dicapai.
3. Meskipun rata-rata *NIAP* maksimum dapat dicapai dengan nilai *rank* sekitar 60, terdapat nilai-nilai *NIAP* yang sangat kecil untuk *query-query* tertentu, seperti untuk query ke-8 diperoleh nilai *NIAP* adalah 0.04545. Oleh karena itu, masih harus dilakukan studi lanjut untuk meningkatkan nilai *NIAP* untuk semua *query*.

Saran untuk perbaikan metode LSI dalam tesis ini adalah

1. Pengelolaan memori yang lebih baik sehingga banyak dokumen yang diproses dapat lebih banyak.
2. Faktor normalisasi untuk dokumen perlu juga diperhatikan sehingga dokumen yang panjang dan dokumen yang pendek tidak dibedakan.

DAFTAR PUSTAKA

1. Aberer, Karl (2003), Latent Semantic Indexing, EPFL-SSC, Laboratoire de systemes d'informations repartis, 36 – 66.
2. Anton, Howard (1994), *Elementary Linear Algebra Seventh Edition*, John Wiley & Sons, 25 – 220.
3. Baeza–Yates, Ricardo, dan Berthier Ribeiro-Neto (1999), *Modern Information Retrieval*, Addison Wesley, 73 – 96.
4. Booch, Grady, James Rumbaugh, dan Ivar Jacobson (1999), *The Unified Modelling Language User Guide*, Addison Wesley, 233 – 241, 243 – 256.
5. Dowling, Jason (2002), *Information Retrieval using Latent Semantic Indexing and a Semi-Discrete Matrix Decomposition*, Monash University, 10 – 12.
6. Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W. dan Harshman, R. A. (1990), Indexing by latent semantic analysis, *Journal of American Society of Information Science*, 391 – 407.
   URL: *http://citeseer.nj.nec.com/deerwester90indexing.html*
7. Grossman, David A., dan Ophir Frieder (1998), *Information Retrieval Algorithms and Heuristics*, Kluwer Academic Publishers, 60 – 65.
8. Hong, Jason I (2000)., An Overview of Latent Semantic Indexing, Tuesday, 17 February 2004.
   URL: *http://www.cs.berkeley.edu/~jasonh/classes/sims240/sims-240-final-paper-lsi.htm*
9. Ingwersen, Peter (1992), *Information Retrieval Interaction*, Taylor Graham Publishing, 49 – 60.
10. Jacob, Bill (1990), *Linear Algebra*, W.H. Freeman and Company, 283.
11. Karlgren, Jussi (1998), The Basics of Information Retrieval , Tuesday, 7 February 2004.
    URL: *http://citeseer.nj.nec.com/146825.html*
12. Liddy, Elizabeth (2001), How a search engine works
    URL: *http://www.infotoday.com/searcher/may01/liddy.htm*

13. Konstathis, April, dan William M. Pottenger (1998), A Mathematical View of Latent Semantic Indexing: Tracing Term Co-occurences, Thursday , 19 February 2004.
URL: *http://www3.lehigh.edu/images/userImages/cdh3/Page_3456/LU-CSE-02-006.pdf*
14. Mandala, Rila (2000), Improving information retrieval system performance by combining different text mining techniques, *Intelligent Data Analysis 4*, 489 – 511.
15. Papadimitriou, Christos H., Prabhakar Raghavan, Hisao Tamaki, Santosh Vempala (1997), Latent Semantic Indexing: A Probabilistic Analysis, Wednesday, 25 February 2004.
URL: *http://www.cse.msu.edu/~cse960/Papers/LSI/LSI-papadimitriou.pdf*
16. Pressman, Roger (1997), *Software Engineering A Practitioner's Approach Fourth Edition*, McGraw-Hill, 31.
17. Purcell, Edwin J., dan Dale Varberg (1995), *Kalkulus dan Geometri Analitis Edisi Kelima*, Erlangga, 130 – 237.
18. Rijsbergen, C.J. van (1979), Information Retrieval, Butterworths, London, 112 – 140.
19. Rosario, Barbara (2000), Latent Semantic Indexing: An overview, Tuesday, 17 February 2004.
URL: http://www.sims.berkeley.edu/~rosario/projects/LSI.pdf
20. Setiawan, Hendra (2002), *Umpan Balik Relevansi pada Sistem Temu Kembali Informasi*, Tugas Akhir Departemen Teknik Informatika ITB, 6 – 14.
21. Witten, I. H., A. Moffat, T. C. Bell (1999), *Managing Gigabytes, 2nd edition*, Morgan Kaufmann Publishing.
22. Weisstein, Eric W.(1999), Singular Value Decomposition From MathWorld – A Wolfram Web Resource，Tuesday, 17 February 2004.
URL: http://mathworld.wolfram.com/SingularValueDecomposition.html

LAMPIRAN

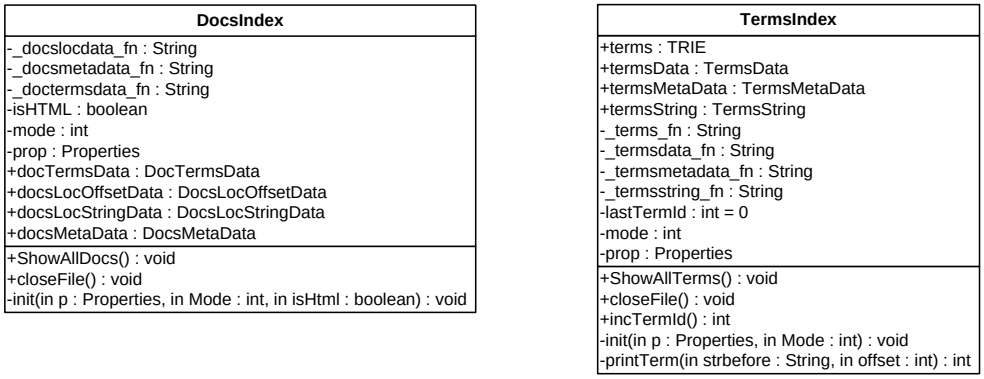
Lampiran 1 Detil *Class Diagram*

♦ Package index

DocumentCollection
+TestQuery : Vector +docsIndex : DocsIndex +isHtml : boolean = false +stemmer : StemManager = null +termsIndex : TermsIndex -colManager : CollectionManager -common : WordsHashtable -currDocId : int = -999 -dd : DocTermData -dm : DocMetaData -manager : ParseManager -prop : Properties -reader : BufferedRandomAccessFile = null -td : TermData -tm : TermMetaData
+addDocumentEntry() : void +addDocumentEntry(in doccount : int) : void +appendDocOffset(in linenumber : long) : void +appendDocString(in str : String) : void +avgDocLength() : double +closeFile() : void +computeDocLength(in docid : int) : void +containsTerm(in doclist : Vector, in termId : int) : int +getAllDocs(in termId : int) : ListBlockEntryEnumerator +getAllTerms(in docId : int) : ListBlockEntryEnumerator +getCollectionCount() : int +getCollectionManager() : CollectionManager +getCommonWords() : WordsHashtable +getDocFreq(in termId : int) : int +getDocFreq(in term : String) : int +getDocLength(in docId : int) : double +getDocMetaData(in docId : int) : DocMetaData +getDocOffset(in docId : int) : long +getDocString(in docId : int) : String +getDocText(in docId : int) : String +getDocTitle(in docId : int) : String +getParseManager() : ParseManager +getTermFreq(in term : String, in docid : int) : int +getTermFreq(in termId : int, in docid : int) : int +getTermFreq(in termId : int, in doclist : Vector) : int +getTermId(in term : String) : int +getTermMetaData(in termId : int) : TermMetaData +getTermString(in termid : int) : String +insertTerm(in term : String, in docid : int) : void +insertTerm(in term : String, in docid : int, in count : int, in mode : int) : void +isWordCommon(in word : String) : boolean +setCollectionManager(in Manager : CollectionManager) : void +setCommonWords(in Common : WordsHashtable) : void +setParseManager(in Manager : ParseManager) : void +setStemmer(in StemManager : StemManager) : void +stem(in input : String) : String -find(in v : Vector, in d : int) : int -init(in p : Properties, in Mode : int) : void -insertTermToDocIndex(in termid : int, in docid : int, in count : int, in mode : int) : void

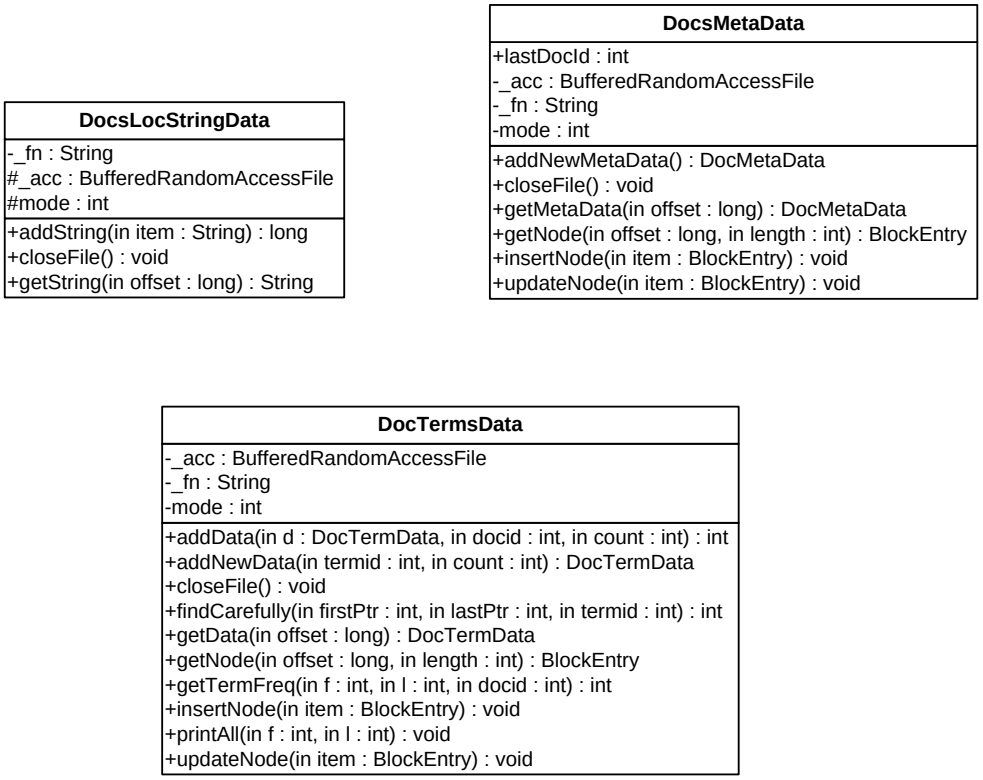
Gambar 1-1 Kelas *DocumentCollection*

Kelas *DocumentCollection* merupakan kelas yang berfungsi sebagai koleksi dokumen dalam proses *parsing, stemming* dan pembangunan matriks kata-dokumen. Asosiasi kelas *DocumentCollection* dengan kelas lainnya dapat dilihat pada gambar V-4



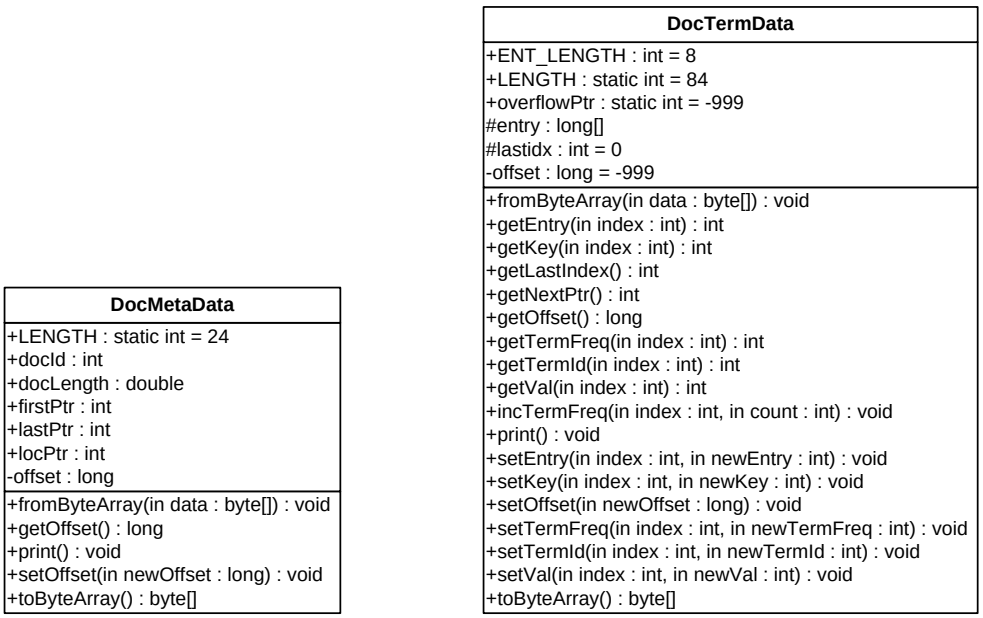
Gambar 1-2 Kelas *DocsIndex* dan *TermsIndex*

Kelas *DocsIndex* dan Kelas *TermsIndex* merupakan dua kelas yang berasosiasi dengan kelas DocumenCollection seperti pada gambar V-4.



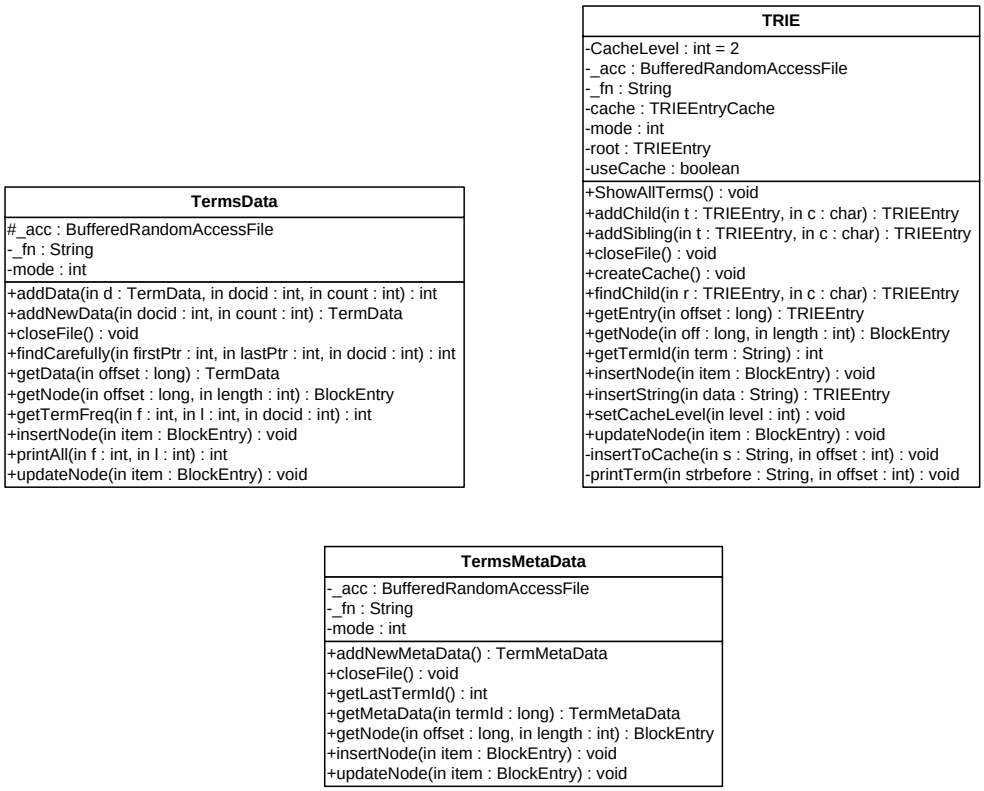
Gambar 1-3 Kelas *DocsLocStringData*, *DocsMetadata*, *DocTermsData*

Kelas *DocsLocStringData*, kelas *DocsMetaData*, dan kelas *DocTermsData* berasosiasi dengan kelas *DocsIndex*. Kelas *DocsMetaData* mengimplementasi *interface BlockFile* seperti pada gambar V-4.



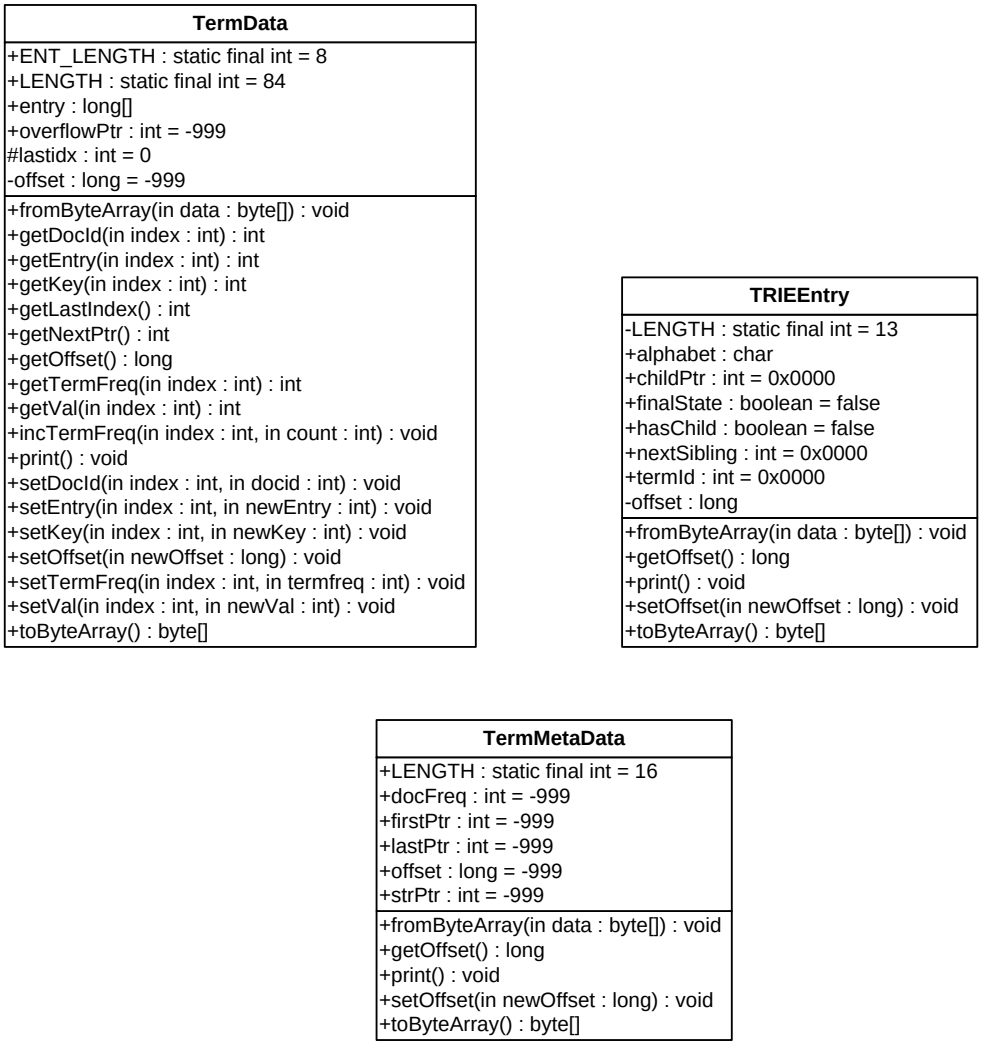
Gambar 1-4 Kelas *DocMetaData* dan *DocTermData*

Kelas *DocMetaData* berasosiasi dengan kelas *DocsMetaData* dan kelas *TermMetaData*. Kelas *DocTermData* berasosiasi dengan kelas *DocTermsData* dan kelas *DocTermData* mengimplementasi *interface ListBlockEntry* seperti pada gambar V-4.



Gambar 1-5 Kelas *TermsData*, kelas *TRIE*, dan kelas *TermsMetaData*

Kelas *TermsData*, kelas *TRIE*, dan kelas *TermsMetaData* berasosiasi dengan kelas *TermsIndex*. Kelas *TRIE* dan kelas *TermsData* mengimplementasi *interface BlockFile* seperti pada gambar V-4.



Gambar 1-6 Kelas *TermData*, kelas *TRIEEntry*, dan kelas *TermMetaData*

Kelas *TermData* berasosiasi dengan kelas *TermsData* dan mengimplementasi *interface ListBlockEntry*. Kelas *TRIEEntry* berasosiasi dengan kelas *TRIE* dan mengimplementasi *interface BlockEntry*. Kelas *TermMetaData* berasosiasi dengan kelas *TermsMetaData* dan kelas *DocMetaData* seperti pada gambar V-4.

Lampiran 2 Kamus Data

high level Cara memandang sistem pada tingkat tertinggi, tidak melihat detil bagian-bagian dari sistem.

query relevance Tabel yang terdiri dari 2 kolom. Kolom pertama adalah nomor query dan kolom kedua adalah nomor dokumen yang relevan dengan query.

stop words Kata-kata yang merupakan kata-kata umum dan bukan kata kunci. Contoh stop words: a, an, the, from.