BAB II
LANDASAN TEORI

2.1 Sistem Temu Kembali Informasi

Sistem temu kembali informasi adalah sebuah alat atau media layanan bagi pengguna untuk memperoleh informasi atau sumber informasi yang dibutuhkan oleh pengguna. Sulistyo Basuki (1991) mendefinisikan temu kembali informasi sebagai kegiatan yang bertujuan untuk menyediakan dan memasok informasi bagi pemakai sebagai jawaban atas permintaan atau berdasarkan kebutuhan pemakai. Dengan kata lain sistem temu kembali informasi bertujuan untuk menjembatani antara informasi dengan pengguna yang membutuhkan informasi.

Mooers (1948) berpendapat bahwa sistem temu kembali informasi adalah seni dan ilmu dalam mencari informasi pada dokumen, mencari untuk dokumen mereka sendiri, mencari untuk metadata dengan gambaran berbentuk dokumen, atau mencari dalam *database*, apakah itu hubungan *database* yang berdiri sendiri atau hiperteks jaringan *database* seperti internet atau intranet, untuk teks, suara, gambar atau data.

Dalam ODLIS (*Online Dictionary for Library and Information Science*), Dijelaskan bahwa sistem temu kembali informasi adalah *proses*, *metode*, dan prosedur yang digunakan untuk menyeleksi informasi yang relevan yang tersimpan dalam *database*. Dalam perpustakaan dan arsip, temu kembali informasi biasanya untuk dokumen yang diketahui atau untuk informasi mengenai subyek tertentu, dan *file* biasanya katalog atau indeks, atau penyimpanan informasi berbasis komputer dan sistem pencarian, seperti katalog *online* atau *database* bibliografi. Dalam merancang sistem tersebut, keseimbangan harus dicapai antara kecepatan, akurasi, biaya, kenyamanan, dan efektivitas.

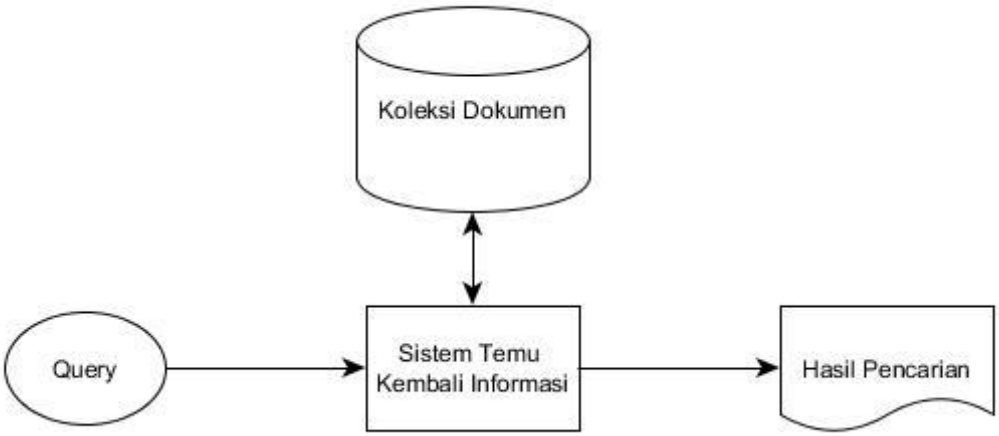
Tujuan dari sitem temu kembali informasi adalah untuk memberikan jawaban terbaik yang sesuai dengan kebutuhan pengguna informasi. Umumnya pada sebuah sistem informasi pengguna mengekspresikan kebutuhun informasi dalam bentuk kata kunci kemudian sistem mencocokan dengan data yang ada pada *database* dan menampilkan data sesuai dengan kata kunci. Dalam kasus sistem temu kembali informasi, kebutuhan informasi pengguna dapat di hubungkan

dengan seluruh data yang memiliki keterkaitan dengan *query* yang di masukan oleh pengguna.

Sistem Temu Kembali Informasi adalah ilmu mencari informasi dalam suatu dokumen, mencari dokumen itu sendiri dan mencari metadata yang menggambarkan suatu dokumen. Sistem Temu Kembali Informasi merupakan cabang dari ilmu komputer terapan (*applied computer science*) yang berkonsentrasi pada representasi, penyimpanan, pengorganisasian, akses dan distribusi informasi.

2.2 Perancangan Sistem temu kembali informasi

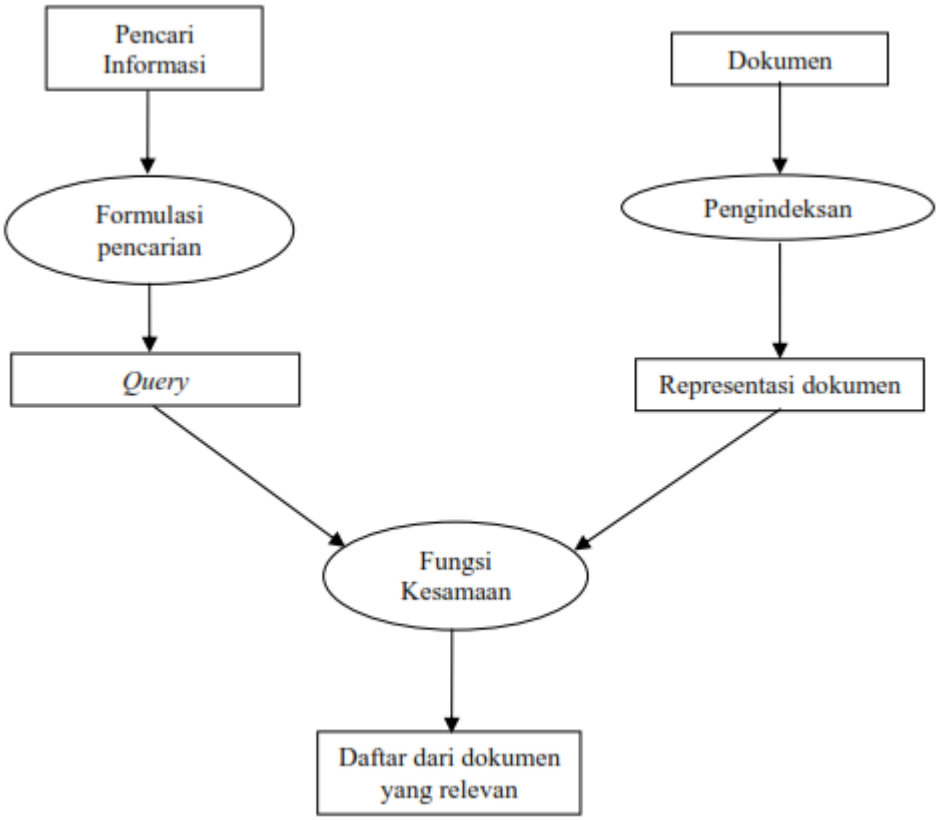
Proses dalam Sistem Temu Kembali Informasi dapat digambarkan sebagai sebuah proses untuk mendapatkan dokumen yang memiliki keterkaitan dari kumpulan dokumen melalui pencocokan *query* yang di masukan oleh pengguna. Sistem Temu Balik Informasi terdiri dari tiga komponen utama, yaitu masukan (*input*), pemroses *(processor*) dan keluaran (*output*). Komponen-komponen ini digambarkan pada Gambar 2.1.



Gambar 2.1 Proses Sistem Temu Kembali Informasi

Kerangka dari sistem temu-kembali informasi sederhana terbagi menjadi dua bagian. Bagian yang pertama adalah bagian si pencari informasi atau pengguna dari sistem. Pengguna dari sistem temu-kembali informasi harus menerjemahkan informasi yang dicarinya agar dapat diproses oleh sistem dengan cara memasukan *query*. *Query* tersebut nanti di proses menjadi sebuah *query* yang dapat dimengerti oleh komputer. Bagian yang kedua adalah bagian dari dokumen. Pada bagian ini

dokumen-dokumen direpresentasikan dalam bentuk indeks. Nanti *query* dari pengguna akan diproses melalui fungsi kesamaan untuk membandingkan *query* dengan indeks dari dokumen untuk mendapatkan dokumen yang relevan. Untuk lebih jelasnya mengenai kerangka sistem temu kembali informasi dapat dilihat pada Gambar 2.2.



Gambar 2.2 Kerangka Sistem Temu Kembali Informasi

2.2.1 Pengindeksan

Mencari sebuah informasi yang relevan sangat tidak mungkin dapat dilakukan oleh sebuah komputer, meskipun dilakukan oleh sebuah komputer yang memiliki spesifikasi yang canggih. Agar komputer dapat mengetahui sebuah dokumen itu relevan terhadap sebuah informasi, komputer memerlukan sebuah model yang mendeskripsikan bahwa dokumen tersebut relevan atau tidak. Salah satu caranya adalah dengan menggunakan indeks istilah.

Indeks adalah bahasa yang digunakan di dalam sebuah buku konvensional untuk mencari informasi berdasarkan kata atau istilah yang mengacu ke dalam suatu halaman. Dengan menggunakan indeks si pencari informasi dapat dengan mudah menemukan informasi yang diinginkannya. Pada sistem temu-kembali informasi, indeks ini nantinya yang digunakan untuk merepresentasikan informasi di dalam sebuah dokumen.

Elemen dari indeks adalah istilah indeks (*index term*) yang didapatkan dari teks yang dipecah di dalam sebuah dokumen. Elemen lainnya adalah bobot istilah (*term weighting*) sebagai penentuan rangking dari kriteria relevan sebuah dokumen yang memiliki istilah yang sama.

Baeza-Yates dan Ribeiro-Neto (1999) menjelaskan tentang proses pembuatan indeks dari sebuah dokumen teks atau dikenal dengan proses analisis teks (*automatic teks analysis*) melalui beberapa tahap:

a. Proses penentuan digit, tanda hubung, tanda baca dan penyeragaman dari huruf yang digunakan.
b. Penyaringan kata meliputi penghilangan kata yang memiliki arti niliai paling rendah (*stopwords*) untuk proses penemu-kembalian.
c. Penghilangan imbuhan kata, baik awalan maupun akhiran kata. Penghilangan imbuhan kata ini dikenal dengan *stemming*.
d. Pemilihan istilah untuk menentukan kata atau stem (atau kelompok kata) yang akan digunakan sebagai elemen indeks.
e. Pembentukan kategori istilah terstruktur seperti kelompok persamaan kata yang digunakan untuk perluasan dari *query* dasar yang diberikan oleh pengguna sistem temu-kembali informasi dengan istilah lain yang sesuai.

Pengindeksan dapat dilakukan dengan dua cara yaitu manual dan otomatis. Idealnya, untuk mendapatkan indeks istilah yang sempurna sebuah pengindeksan dilakukan secara manual (konvensional). Akan tetapi, menurut Salton (1968) sistem pencarian dan analisa teks yang sepenuhnya otomatis tidak menghasilkan kinerja temu-kembali yang lebih buruk dibandingkan dengan sistem konvensional yang menggunakan pengindeksan dokumen manual dan formulasi pencarian manual.

2.2.2 *Stemming*

Stemming adalah proses konversi term ke bentuk umumnya, sebagaimana dijelaskan sebelumnya. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi term-term tertentu di dalamnya. Sinonim adalah kata-kata yang mempunyai pengertian serupa tetapi berbeda dari sudut pandang morfologis.
Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1

2.2.2.1 *Stemming* Nazief-Andriani

Algoritma Nazief & Adriani yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Pertama cari kata yang akan distem dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata adalah *root word*. Maka algoritma berhenti.

2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa particles (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus Possesive Pronouns (“-ku”, “-mu”, atau “-nya”), jika ada.

3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.

4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.

b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.

5. Melakukan Recoding.

6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai. Tipe awalan ditentukan melalui langkah-langkah berikut:
   a. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
   b. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
   c. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
   d. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah “bukan “none” maka awalan dapat dilihat pada Tabel 2.1 Hapus awalan jika ditemukan.

Tabel 2.1 Kombinasi Awalan Akhiran Yang Di Hapus

Awalan	**Akhiran yang di hapus**
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

Tabel 2.2 Cara Menentukan Tipe Awalan Untuk awalan “te-”

Following Characters				**Tipe Awalan**
Set 1	**Set 2**	**Set 3**	**Set 4**	
“-r-“	“-r-“	-	-	None
“-r-“	-	-	-	ter-luluh
“-r-“	not (vowel or “-r-”)	“-er-“	vowel	Ter
“-r-“	not (vowel or “-r-”)	“-er-“	not vowel	ter-
“-r-“	not (vowel or “-r-”)	not “-er-“	-	Ter
not (vowel or “-r-”)	“-er-“	vowel	-	None
not (vowel or “-r-”)	“-er-“	not vowel	-	Te

Tabel 2.3 Jenis Awalan Berdasarkan Tipe Awalannya

Tipe Awalan	**Awalan yang harus di hapus**
di-	di-
ke-	ke-
se-	se-
te-	te-
ter-	ter-

Untuk mengatasi keterbatasan pada algoritma di atas, maka ditambahkan aturan-aturan berikut:

1. Aturan untuk reduplikasi.
    a. Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka *root word* adalah bentuk tunggalnya, contoh: “buku-buku” *root word*-nya adalah “buku”.
    b. Kata lain, misalnya “bolak-balik”, “berbalas-balasan, dan ”seolaholah”. Untuk mendapatkan *root word*-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki *root word* yang sama maka diubah menjadi bentuk tunggal, contoh: kata “berbalasbalasan”, “berbalas” dan “balasan” memiliki *root word* yang sama yaitu “balas”, maka *root word* “berbalas-balasan” adalah “balas”. Sebaliknya, pada kata “bolak-balik”, “bolak” dan

"balik" memiliki root word yang berbeda, maka *root word*-nya adalah "bolak-balik".

2. Tambahan bentuk awalan dan akhiran serta aturannya.
   a. Untuk tipe awalan "mem-", kata yang diawali dengan awalan "memp-" memiliki tipe awalan "mem-".
   b. Tipe awalan "meng-", kata yang diawali dengan awalan "mengk-" memiliki tipe awalan "meng-".

2.2.3 Term-document Matriks

Setelah melalui pengindeksan dan *stemming*, matriks term-document dibangun dengan menempatkan kata hasil proses *stemming* (term) ke dalam baris. Matriks ini disebut *term-document* matriks. Setiap baris mewakili sebuah kata yang unik, sedangkan setiap kolom mewakili konteks dari mana kata-kata tersebut diambil. Konteks yang dimaksud bisa berupa kalimat, paragraf, atau seluruh bagian dari teks. Contoh Term-document Matrixs dapat di lihat pada Tabel 2.4.

Tabel 2.4 Term-document Matriks

	Doc 1	**Doc 2**	**Doc 3**	**Doc n**
Term 1	1	2	1	N
Term 2	1	0	1	N
Term 3	2	2	1	N
Term 4	1	1	0	N
Term 5	1	0	1	N
Term 6	1	1	0	N
Term 7	1	1	0	N
Term 8	1	1	2	N
Term n	N	n	n	N

Pada tabel di atas, baris pertama mewakili stemmed term (term 1, term 2, n), dan bagian kolom mewakili konteks, yaitu teks. Nilai yang terletak pada setiap cell pada tabel menunjukkan berapa kali sebuah term muncul dalam sebuah

dokumen. Contohnya, term 1 muncul 1 kali pada dokumen ke-1, dan muncul 2 kali pada dokumen ke-2, namun term 1 tidak muncul pada dokumen 3, dan seterusnya.

2.2.4 *Singular Value Decomposition*

Singular Value Decomposition (SVD) adalah suatu pemfaktoran matrik dengan mengurai suatu matrik ke dalam dua matrik P dan Q. Jika diketahui suatu matrik adalah matrik A berukuran m×n dengan rank r > 0 , maka dekomposisi dari matrik A dinyatakan sebagai

$$\boxed{\begin{matrix} A = P\Delta \\ Q^T \end{matrix}} \quad \text{.....(2.1)}$$

Rank (r) menyatakan banyaknya jumlah baris atau kolom yang saling independen antara baris atau kolom lainnya dalam suatu matrik. P merupakan matrik orthogonal berukuran m×r sedangkan Q merupakan matrik orthogonal berukuran n×r. Δ adalah matrik diagonal berukuran r×r yang elemen diagonalnya merupakan akar positif dari eigenvalue matrik A. Terbentuknya matrik Δ tergantung kondisi matrik A, yaitu diantaranya:

a. Δ bila r = m = n .....(2.2)

b. $\begin{bmatrix} \Delta \\ (0) \end{bmatrix}$ bila r = n dan r<m .....(2.3)

c. $\begin{bmatrix} \Delta & (0) \end{bmatrix}$ bila r=m dan r<n .....(2.4)

d. $\begin{bmatrix} \Delta & (0) \\ (0) & (0) \end{bmatrix}$ bila r<m dan r<n .....(2.5)

Matrik P dapat diperoleh melalui perkalian antara A, Q dan Δ^{-1} sehingga dapat dinyatakan :

$$P = AQ\Delta^{-1} \quad \text{.....(2.6)}$$

Singular Value Decomposition (SVD) adalah salah satu teknik reduksi dimensi yang bermanfaat untuk memperkecil nilai kompleksitas dalam pemrosesan term-document matrix. SVD merupakan teorema aljabar linier yang menyebutkan

bahwa persegi panjang dari term-document matriks dapat dipecah/didekomposisikan menjadi tiga matriks, yaitu :

a. Matriks ortogonal U.
b. Matriks diagonal S.
c. Transpose dari matriks ortogonal V

Hasil dari proses SVD adalah vektor yang akan digunakan untuk menghitung similaritasnya dengan pendekatan *cosine similarity*.

2.2.5 Metode Cosine Similarity

Metode *Cosine Similarity* merupakan metode yang digunakan untuk menghitung *similarity* (tingkat kesamaan) antar dua buah objek. Secara umum penghitungan metode ini didasarkan pada *vector space similarity measure*. Metode *cosine similarity* ini menghitung similarity antara dua buah objek (misalkan D1 dan D2) yang dinyatakan dalam dua buah vector dengan menggunakan *keywords* (kata kunci) dari sebuah dokumen sebagai ukuran. Formula dari *Cosine Similarity* dapat di lihat sebagai berikut :

$$\cos \alpha = \frac{A.B}{|A||B|} = \frac{\Sigma_{i=1}^{n} A_i \; X \; B_i}{\sqrt{\Sigma_{i=1}^{n}} (A_i)^2 \; X \; \sqrt{\Sigma_{i=1}^{n}} (B_i)^2} \quad \text{.....(2.7)}$$

A = Vektor dokumen

B = Vektor query

$A.B$ = Perkalian vektor A dan vektor B

$|A|$ = Panjang Vektor A

$|B|$ = Panjang Vektor B

$|A||B|$ = Cross product antara A dan B

α = Sudut yang terbentuk antara vektor A dan B

Cosine similarity digunakan untuk menghitung nilai kosinus sudut antara vektor dokumen dengan vektor kueri. Semakin kecil sudut yang dihasilkan, maka tingkat kemiripan esai semakin tinggi. Dari hasil *cosine similarity*, akan didapatkan nilai yang akan dibandingkan dengan penilaian manusia untuk diuji selisih nilainya.

2.3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) adalah sebuah teknik statistik terotomasi untuk membandingkan kesamaan semantik dari beberapa kata atau beberapa dokumen. LSA bukanlah sebuah program yang tradisional dari natural *language processing* ataupun *artificial intelligence*. Teknik ini digunakan dalam menganalisis dokumen untuk menemukan arti atau konsep dari dokumen tersebut. LSA lahir karena didasarkan pada pemikiran bahwa *syntax* dan *style* saja tidak mencukupi untuk menilai sebuah esai. Hal yang menjadi kesulitan mendasar adalah ketika kita hendak membandingkan kata-kata untuk menemukan dokumen yang relevan. Hal yang sebenarnya dibandingkan adalah arti atau konsep di balik kata-kata tersebut.

Metode LSA melakukan mapping dari kata ataupun dokumen menjadi sebuah *concept space* dan perbandingan dilakukan pada *space* ini. *Concept space* tersebut atau yang lebih sering disebut sebagai *latent semantic space* merupakan hasil mapping dari matriks dimensi tinggi menjadi dimensi yang lebih kecil. Meskipun dalam dimensi yang lebih kecil, matriks tersebut merupakan matriks yang merepresentasikan isi dari keseluruhan dokumen. Ciri khas dari LSA adalah teknik yang dinamakan *Singular Value Decomposition* (SVD). SVD digunakan untuk melakukan dekomposisi matriks setelah diberikan pembobotan untuk kemudian diukur kesamaannya dengan data yang akan diujicobakan (Suhartono, 2014).

2.3.1 Konsep Latent Semantic Analysis

Konsep *Latent Semantic Analysis* (LSA) merupakan metode IR yang membangun struktur koleksi dokumen dalam bentuk ruang vektor dengan menggunakan teknik aljabar linier, yaitu *singular value decomposition*. Menurut Bunyamin (2005), secara umum konsep LSA meliputi beberapa point sebagai berikut :

1. *Text Operations pada Query dan Document Collection*

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

   a. Mem-parsing setiap kata dari koleksi dokumen.
   b. Membuang kata-kata yang merupakan *stop words*.
   c. 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,

   a. Menghitung frekuensi kemunculan dari kata,
   b. Membangun matriks kata-dokumen, dimana 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*

   a. Matriks kata-dokumen yang terbentuk, A berukuran m×n, selanjutnya dikenakan dekomposisi SVD (*singular value decomposition*). Hasil SVD berupa 3 (tiga) buah matriks, sehingga matriks A dapat ditulis menjadi :

   $$A = U\ S\ V\ T \quad \text{.....(2.8)}$$

   b. Untuk mempermudah penjelasan, misalkan u1, u2, …, uk adalah vektor-vektor kolom dari matriks U, σ1, σ2, …, σk adalah entry-entry di diagonal utama dari matriks S, dan v1, v2, …, vk adalah vektor-vektor kolom dari matriks V.
   c. Rank dari matriks A, k adalah banyaknya entry tak nol yang terletak pada diagonal utama matriks S , yaitu σ1, σ2, …, σk . k juga merupakan banyaknya nilai singular dari A. d. Dari k buah nilai singular dari A , dipilih r buah nilai singular yang terbesar, yaitu $\sigma 1 \geq \sigma 2 \geq \ldots$, $\sigma r > 0$, dengan $r < k$.

4. *Query Vector Creation*

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

5. *Query Vector Mapping*

   Point (3)(e) diatas 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 , yaitu :

   $$Q = qTUrSr - 1 \quad \text{.....(2.9)}$$

6. *Ranking*

   Kolom-kolom pada matriks VrT pada point (3)(e) adalah vektor-vektor dokumen yang digunakan untuk menghitung sudut antara vektor 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.

7. Hasil Akhir

   Perhitungan cosinus sudut antara *query*, Q dan dokumen Dj, j = 1, 2, …,n diperoleh dan diurutkan berdasarkan dari yang paling besar sampai yang terkecil. Nilai cosinus sudut yang terbesar menunjukkan dokumen yang paling relevan dengan *query*.