BAB II
LANDASAN TEORI

II.1 Information Retrieval

Information Retrieval System(Sistem Temu Balik Informasi) merupakan bagian dari bidang ilmu komputer yang bertujuan untuk pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri. Menurut Gerald J. Kowalski di dalam bukunya "*Information Storage and Retrieval System Theory and Implementation*", sistem temu balik informasi adalah suatu sistem yang mampu melakukan penyimpanan, pencarian, dan pemeliharaan informasi. Informasi dalam konteks yang dibahas dalam buku ini dapat berupa informasi teks (termasuk data numerik dan tanggal), gambar, audio, video, dan objek multimedia lainnya [7].

Menurut Manning, *Information Retrieval* adalah kemampuan untuk memberikan informasi dan menemukan dokumen dari koleksi dokumen yang sangat besar yang tidak terstruktur dan tersimpan dalam komputer sesuai dengan yang dibutuhkan pengguna [5].

Tujuan dari sistem *Information retrieval* adalah untuk memenuhi kebutuhan informasi pengguna dengan me-*retrieve* semua dokumen yang mungkin relevan, tetapi pada waktu yang sama me-*retrieve* sesedikit mungkin dokumen yang tidak relevan. Sistem IR yang baik memungkinkan pengguna menentukan secara cepat dan akurat apakah isi dari dokumen yang diterima memenuhi kebutuhannya. Di dalam *information retrieval* mendapatkan dokumen yang relevan tidaklah cukup. Tujuan yang harus dipenuhi dalam suatu IR adalah bagaimana menyusun dokumen yang telah didapatkan tersebut ditampilkan terurut dari dokumen yang memiliki tingkat relevansi tinggi ke tingkat relevansi yang lebih rendah. Penyusunan urutan dokumen tersebut disebut sebagai perangkingan dokumen [14].

Query dalam *information retrieval* merupakan sebuah *formula* yang

digunakan untuk mencari informasi yang dibutuhkan oleh *user*, dalam bentuk yang paling sederhana, sebuah *query* merupakan suatu *keywords* (kata kunci) dan dokumen yang mengandung *keywords* merupakan dokumen yang dicari dalam *information retrieval*.

Model *Information Retrieval* adalah model yang digunakan untuk melakukan pencocokan antara term-term dari *query* dengan term-term dalam *document collection,* Model yang terdapat dalam *Information retrieval* terbagi dalam 3 model besar, yaitu :

1. *Set-theoritic models,* model merepresentasikan dokumen sebagai himpunan kata atau frase. Contoh model ini ialah *standard Boolean model* dan *extended Boolean model.*
2. *Algebraic model,* model merepresentasikan dokumen dan *query* sebagai vektor *similarity* antara vektor dokumen dan vektor *query* yang direpresentasikan sebagai sebuah nilai skalar. Contoh model ini ialah *vektor space model* (model ruang vektor) , *latent semantic indexing* (LSI) dan *generalized vector space model* (GVSM).
3. *Probabilistic model,* model memperlakukan proses pengambilan dokumen sebagai sebuah *probabilistic inference.* Contoh model ini ialah penerapan teorema bayes dalam model probabilistic [12].

Secara garis besar terdapat dua proses besar yang dilakukan oleh sistem information retrieval yaitu preprocessing dan proses searching yang akan dijelaskan pada subab II.3 dan subab II.4

II.2 Algebraic Model

Algebraic model adalah salah satu dari 3 model besar *information retrieval*. Secara bahasa *algebraic* memiliki arti pertemuan, hubungan, atau penyelesaian. *Algebraic* atau aljabar adalah cabang ilmu matematika yang dapat dicirikan sebagai generalisasi dari bidang aritmatika. Pada *information retrieaval,* aljabar yang digunakan adalah aljabar linier yang mempelajari sistem persamaan linier dan solusinya, vektor, serta transformasi linier. Matriks dan operasinya juga merupakan hal yang berkaitan erat dengan bidang aljabar linier.

Pada *information retrieval* yang menggunakan *algebraic* model, seluruh dokumen dan *query* direpresentasikan menjadi vektor, untuk menemukan *similiarity* antara dokumen dan *query* maka nilai skalar antara vektor *query* dan vektor dokumen akan dikaitkankan sehingga muncul nilai skalar sebagai acuan pengurutan dokumen [12].

Hal yang perlu diperhatikan dalam pencarian informasi menggunakan *algebraic* model dari koleksi dokumen yang heterogen adalah pembobotan term. Term dapat berupa kata, frase atau unit hasil *indexing* lainnya dalam suatu dokumen yang dapat digunakan untuk mengetahui konteks dari dokumen tersebut [4].

Karena setiap kata memiliki tingkat kepentingan yang berbeda dalam dokumen, maka untuk setiap kata tersebut diberikan sebuah indikator, yaitu term *weight.* Term *weighting* atau pembobotan term sangat dipengaruhi oleh hal-hal berikut ini [9]:

1. Term *Frequency* (tf) *factor*, yaitu faktor yang menentukan bobot term pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan suatu kata (term *frequency*) diperhitungkan dalam pemberian bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu term(tf tinggi) dalam dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuian yang semakin besar.
2. *Inverse Document Frequency* (idf) *factor*, yaitu pengurangan dominansi term yang sering muncul di berbagai dokumen. Hal ini diperlukan karena term yang banyak muncul di berbagai dokumen, dapat dianggap sebagai term umum (*common* term) sehingga tidak penting nilainya. Sebaliknya faktor kejarang munculan kata (term *scarcity*) dalam koleksi dokumen harus diperhatikan dalam pemberian bobot. Menurut Mandala (dalam Witten, 1999) 'Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon term*) 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 proposional dengan urutannya.'

3. collection frequency (*cfi) factor* : banyaknya kemunculan istilah *ti* dalam koleksi.

Contoh model Algebraic adalah *vector space model*, *generalized vector space model, topic-based vector space model, extended boolean model*, dan *latent semantic indexing*. Pada subab berikutnya akan dijelaskan mengenai *vector space model* dan *generalized vector space model.*

II.2.1 Vector Space Model

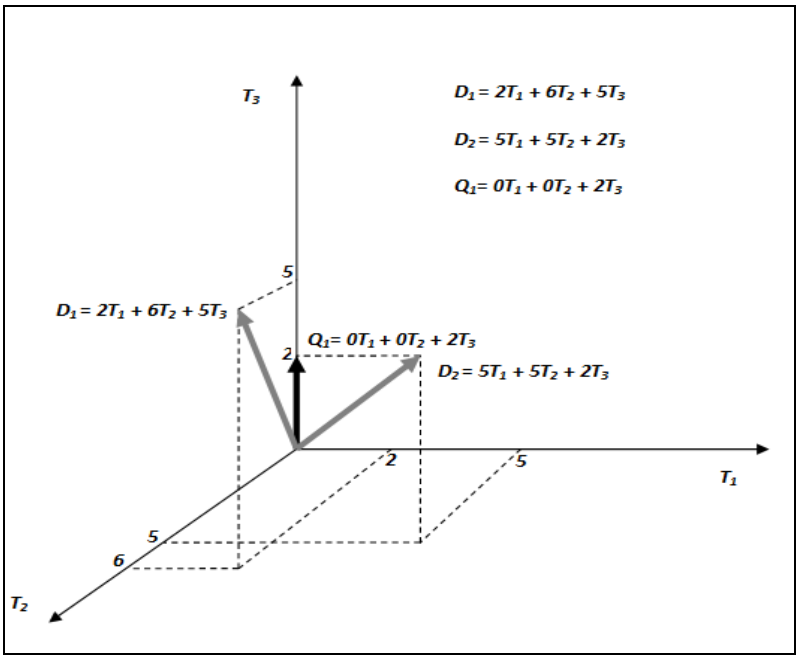
Setelah menghitung pembobotan, dilakukan suatu perhitungan ukuran kemiripan antar dokumen dengan *query*. Ukuran ini memungkinkan perankingan dokumen sesuai dengan kemiripan relevansinya terhadap *query*.

Vector Space Model adalah model sistem temu balik informasi yang mengibaratkan masing-masing *query* dan dokumen sebagai sebuah vektor n-dimensi. Tiap dimensi pada vektor tersebut diwakili oleh satu term. Term yang digunakan biasanya berpatokan kepada term yang ada pada *query* atau *keyword*, sehingga term yang ada pada dokumen tetapi tidak ada pada *query* biasanya diabaikan [5].

Model ini menentukan kemiripan (*similarity)* antara dokumen dengan *query* dengan cara merepresentasikan dokumen dan *query* masing-masing ke dalam bentuk vektor. Tiap kata yang ditemukan pada dokumen dan *query* diberi bobot dan disimpan sebagai salah satu elemen vektor.

Cosine Similarity tidak hanya digunakan untuk menghitung normalisasi panjang dokumen tapi juga menjadi salah satu ukuran kemiripan yang popular. Ukuran ini menghitung nilai kosinus sudut antara dua vektor. Jika terdapat dua vektor dokumen *dj* dan *query q*, serat *t* term diekstrak dari koleksi dokumen maka nilai kosinus antara *dj* dan *q* didefinisikan sebagai berikut:

$$\text{Sim}\left(\vec{d_j}.\vec{q}\right) = \frac{\vec{d_j}.\vec{q}}{|\vec{d_j}||\vec{q}|} = \frac{\sum_{i=1}^{t}(w_{ij}\,x\,w_{iq})}{\sqrt{\sum_{i=1}^{t}(w_{ij})^2\,x\,\sum_{i=1}^{t}(w_{iq})^2}} \qquad (1)$$



Gambar II.1 Contoh Model Ruang Vector

Jika dua dokumen D1 = 2T1 + 6T2 + 5T3 dan D2 = 5T1 + 5T2 + 2T3 dan *query* Q1 = 0T1 + 0T2 + 2T3 sebagaimana diperlihatkan pada Gambar II.1, berikut ini adalah nilai kosinus yang diperoleh:

$$\text{Sim}\left(\vec{d_1}.\vec{q}\right) = \frac{(2x0)+(6x0)+(5x2)}{\left(\sqrt{4+36+25}\right)\left(\sqrt{0+0+4}\right)} = \frac{10}{\sqrt{65,4}} = 0,62$$

$$\text{Sim}\left(\vec{d_2}.\vec{q}\right) = \frac{(5x0)+(5x0)+(2x2)}{\left(\sqrt{25+25+4}\right)\left(\sqrt{0+0+4}\right)} = \frac{4}{\sqrt{54,4}} = 0,27$$

Contoh di atas memperlihatkan bahwa sesuai dengan perhitungan kosinus, dokumen D2 lebih mirip dengan *query* daripada dokumen D1. Terlihat sudut antara D2 dan Q1 lebih kecil daripada sudut antara D1 dan Q1.

II.2.2 Generalized Vector Space Model

Algoritma *Generalized Vector Space Model* yang dibahas menggunakan konsep ruang vektor. Masukan dari pengguna dan kumpulan dokumen diterjemahkan menjadi vektor-vektor. Kemudian vektor-vektor tersebut dikenakan operasi perkalian titik dan hasilnya menjadi acuan dalam menentukan relevansi masukan pengguna (*query*) terhadap kumpulan dokumen.

Ada beberapa langkah atau proses untuk mendapatkan hasil dari *query* yang dimasukkan, yang disebut algoritma *Generalized Vector Space Model* [4]:

1. Membuang kata depan dan kata penghubung.
2. Menggunakan *stemmer* pada kumpulan dokumen dan *query*, yaitu aplikasi yang digunakan untuk menghilangkan imbuhan (awalan, akhiran).

   Contoh : keagungan = agung, keabadian = abadi.

3. Menentukan *min*term untuk menentukan kemungkinan pola frekuensi kata. Panjang *min*term ini didasarkan pada banyak kata yang diinput pada *query*. Kemudian diubah menjadi vektor ortogonal sesuai dengan pola *min*termyang muncul. Kemungkinan pola yang akan muncul adalah :

Tabel II.1 Pola Minterm

M_1= (0,0,0)
M_2= (1,0,0)
M_3= (0,1,0)
.....
M_n= (0,0,0)

4. Menghitung banyaknya frekuensi atau kemunculan kata dalam kumpulan dokumen yang sesuai dengan *query*

5. Menghitung *index* term yang dapat dinyatakan dengan :

$$\overrightarrow{Ki} = \frac{\sum_{\forall r, gi(Mr)=1} Ci,r.\overrightarrow{Mr}}{\sqrt{\sum_{\forall r, gi(Mr)=1} Ci,r^2}} \qquad (2)$$

Dimana :

$\overrightarrow{Ki}$: *Index* term ke-i

$\overrightarrow{Mr}$: vektor ortogonal sesuai pola *min*termyang terpakai

$C_{i,r}$: Faktor korelasi antara *Index* term ke-i dengan minterm r

Sedangkan faktor korelasi sebagai berikut :

$$C_{i,r} = \sum_{d_j | g_i(\overrightarrow{dj}) = g_i(M_r)} W_{i,j} \qquad (3)$$

Dimana:

$C_{i,r}$: Faktor korelasi antara *Index* term i dengan minterm r

$W_{i,j}$: Berat *Index* term i pada dokumen j

$g_i(M_r)$: bobot *Index* term K_i dalam minterm M_r

6. Mengubah dokumen dan *query* menjadi vektor

$$\overrightarrow{d_j} = \sum_{i=1}^{n} W_{i,j} \text{x} \overrightarrow{K_i} \quad (4)$$

$$\vec{q} = \sum_{i=1}^{n} q_i \text{x} \overrightarrow{K_i} \quad (5)$$

Dimana :

d_j : vektor dokumen ke-J

$\vec{q}$:vektor *query*

$W_{i,j}$:Berat *Index* term i pada dokumen j

q_i :berat *Index* term pada *query* i

n :jumlah *Index* term

7. Mengurutkan dokumen berdasarkan similiritas, dengan menghitung perkalian vektor

$$\text{Sim}\left(\overrightarrow{d_j}.\vec{q}\right) = \frac{\overrightarrow{d_j}.\vec{q}}{|\overrightarrow{d_j}||\vec{q}|} \quad (6)$$

Dimana :

d_j : vektor dokumen ke-J

$\vec{q}$: vektor *query*

II.2.3 Perbedaan VSM dan GVSM

Untuk dapat membedakan antara metode *vector space model* (VSM) dan *generalized vector space model* (GVSM) diberikan contoh kasus sebagai berikut :

Terdapat Sebuah *query* (q) dan 2 buah dokumen, yaitu dokumen ke-1 (d1) dan dokumen ke-2 (d2) Sebagai berikut :

Tabel II.2 Contoh Kasus

Query	Terjadinya Kerusakan Sinyal
Judul dokumen ke-1 (d1)	Kerusakan Pemancar di atap gedung, teknisi memeriksa dua kali
Judul dokumen ke-2 (d2)	Sinyal Kompleks rusak karena gangguan cuaca

II.2.3.1 Menggunakan Vector Space Model

1. Menentukan Minterm

Menentukan minterm berdasarkan banyak kata yang diinput pada *query* dan kemungkinan pola yang muncul . Sebelumnya *query* dilakukan proses *break into token*, *filtration*, dan *stemming* sehingga didapatkan *root word*. Berdasarkan *query* tersebut *minterm* yang dipakai adalah Mx,My,Mz

Tabel II.3 Minterm

Mx	terjadi
My	rusak
Mz	sinyal

2. Menghitung Banyaknya frekuensi

Table I Menghitung banyaknya frekuensi VSM

Tabel II.4 Frekuensi Kemunculan Kata Pada Dokumen

	Terjadi	Rusak	Sinyal	Vektor Orthogonal
D1	2	4	0	$\overrightarrow{M_1}$
D2	1	3	3	$\overrightarrow{M_2}$
Q	1	1	1	

3. Menghitung *Similiarity* Dokumen dan *Query*

Sebelum menghitung *similiarity* seluruh term dokumen dan term *query* ditransformasikan kedalam vektor sebagai berikut:

$D_1 = 2T_1 + 4T_2$

$D_2 = 1T_1 + 3T_2 + 3T_3$

$q = T_1 + T_2 + T_3$

Menghitung nilai similiarity berdasarkan vektor :

$$\text{Sim}\left(\overrightarrow{d_1}.\vec{q}\right) = \frac{(2x1)+(4x1)}{(\sqrt{2^2+4^2})(\sqrt{1^2+1^2+1^2})} = \frac{6}{\sqrt{20}} = 1{,}341$$

$$\text{Sim}\left(\overrightarrow{d_2}.\vec{q}\right) = \frac{(1x1)+(3x1)+(3x1)}{(\sqrt{1^2+3^2+3^2})(\sqrt{1^2+1^2+1^2})} = \frac{7}{\sqrt{19}} = 1{,}605$$

Berdasarkan hasil perhitungan *similiarity* antara dokumen dan *query* , dapat disimpulkan bahwa dokumen yang memiliki persamaan yang dekat dengan *query* adalah :

1. Dokumen 2 (D_2) = 1,605
2. Dokumen 1 (D_1) = 1,341

Dikarenakan nilai *similiarity* dokumen dua lebih besar dibandingkan dengan nilai *similiarity* dokumen 1 (Sim $(\vec{d_2}.\vec{q})$ >Sim $(\vec{d_1}.\vec{q})$).

II.2.3.2 Menggunakan Generalized Vector Space Model

1. Menentukan Minterm

Menentukan minterm berdasarkan banyak kata yang diinput pada *query* dan kemungkinan pola yang muncul . sebelumnya *query* dilakukan proses *break into token*, *filtration*, dan *stemming* sehingga didapatkan *root word*. Berdasarkan *query* tersebut *minterm* yang dipakai adalah Mx,My,Mz

Tabel II.5 Minterm

Mx	terjadi
My	rusak
Mz	sinyal

2. Menghitung Banyaknya frekuensi

Menghitung banyaknya frekuensi GVSM

Tabel II.6 Frekuensi Kemunculan Kata Pada Dokumen

	terjadi	Rusak	Sinyal	Vektor Orthogonal
D1	2	4	0	$\vec{M_1}$
D2	1	3	3	$\vec{M_2}$
Q	1	1	1	

3. Menghitung Korelasi Setiap Term

Tabel II.7 Korelasi Setiap Term

$C_{1,1} = 2$	$C_{2,1} = 4$	$C_{3,1} = 0$
$C_{1,2} = 1$	$C_{2,2} = 3$	$C_{3,2} = 3$

4. Menghitung *Index* term

$$\vec{K_1} = \frac{C1,1\vec{M_1} + C1,2\vec{M_2}}{\sqrt{C1,1^2 + C1,2^2}} = \frac{2\vec{M_1} + 1\vec{M_2}}{\sqrt{5}}$$

$$\overrightarrow{K_2} = \frac{C2,1\overrightarrow{M_1} + C2,2\overrightarrow{M_2}}{\sqrt{C2,1^2+C2,2^2}} = \frac{4\overrightarrow{M_1} + 3\overrightarrow{M_2}}{5}$$

$$\overrightarrow{K_3} = \frac{C3,1\overrightarrow{M_1} + C3,2\overrightarrow{M_2}}{\sqrt{C3,1^2+C3,2^2}} = \frac{3\overrightarrow{M_2}}{\sqrt{9}} = \overrightarrow{M_2}$$

5. Mengubah dokumen dan *query* kedalam bentuk vektor

$\overrightarrow{d_1} = 2\ \overrightarrow{K_1} + 4\ \overrightarrow{K_2}$

$= 2\left[\frac{2\overrightarrow{M_1} + 1\overrightarrow{M_2}}{\sqrt{5}}\right] + 4\left[\frac{4\overrightarrow{M_1} + 3\overrightarrow{M_2}}{5}\right]$

$= 4{,}988\overrightarrow{M_1} + 3.294\ \overrightarrow{M_2}$

$\overrightarrow{d_2} = 1\ \overrightarrow{K_1} + 3\ \overrightarrow{K_2} + 3\ \overrightarrow{K_3}$

$= 1\left[\frac{2\overrightarrow{M_1} + 1\overrightarrow{M_2}}{\sqrt{5}}\right] + 3\left[\frac{4\overrightarrow{M_1} + 3\overrightarrow{M_2}}{5}\right] + 3\ \overrightarrow{M_2}$

$= 3{,}294\overrightarrow{M_1} + 5{,}247\ \overrightarrow{M_2}$

$\vec{q} = 1\ \overrightarrow{K_1} + 1\ \overrightarrow{K_2} + 1\ \overrightarrow{K_3}$

$= 1\left[\frac{2\overrightarrow{M_1} + 1\overrightarrow{M_2}}{\sqrt{5}}\right] + 1\left[\frac{4\overrightarrow{M_1} + 3\overrightarrow{M_2}}{5}\right] + 1\ \overrightarrow{M_2}$

$= 1{,}694\overrightarrow{M_1} + 2{,}047\ \overrightarrow{M_2}$

6. Menghitung Similaritas dokumen dan meranking

$$\text{Sim}\left(\overrightarrow{d_1}.\vec{q}\right) = \frac{(4{,}988x1{,}694)+(3{,}294x2{,}047)}{\left(\sqrt{4{,}988^2+3{,}294^2}\right)\left(\sqrt{1{,}694^2+2{,}047^2}\right)} = 0{,}95655$$

$$\text{Sim}\left(\overrightarrow{d_2}.\vec{q}\right) = \frac{(3{,}294x1{,}694)+(5{,}247x2{,}047)}{\left(\sqrt{3{,}294^2+5{,}247^2}\right)\left(\sqrt{1{,}694^2+2{,}047^2}\right)} = 0{,}99146$$

Berdasarkan hasil *Similiarity* antara dokumen dan *query* maka dapat disimpulkan bahwa urutan dokumen yang sesuai dengan *query* adalah :

1. Dokumen 2 (D_2) = 0,99146
2. Dokumen 1 (D_1) = 0,95655

Dikarenakan nilai *similiarity* dokumen dua lebih besar dibandingkan dengan nilai *similiarity* dokumen 1 (Sim $\left(\overrightarrow{d_2}.\vec{q}\right)$ >Sim $\left(\overrightarrow{d_1}.\vec{q}\right)$) .

Berdasarkan contoh kasus diatas dapat disimpulkan bahwa perbedaan antara *vector space model* dan *generalized vector space model* terdapat pada

perhitungan korelasi antar *query* dan dokumen. Dimana pada *generalized vector space model* semua term dijadikan vektor orthogonal untuk menghitung *Index* term dan setelah itu setiap term pada dokumen digeneralisasi menjadi vektor orthogonal dengan mengkalikan hasil *Index* term dengan term dokumen dan *query*, yang kemudian setiap vektor tersebut dikenakan operasi perkalian titik dan hasilnya menjadi acuan dalam menentukan relevansi masukan pengguna (*query*) terhadap kumpulan dokumen.

II.3 Preprocessing

Pada tahapan *preprocessing*, setiap dokumen (termasuk *query*) direpresentasikan menggunakan model *bag-of-words* yang mengabaikan urutan dari kata-kata di dalam dokumen, struktur sintaktis dari dokumen dan kalimat. Dokumen ditransformasi ke dalam suatu "tas" berisi kata-kata independen. Term disimpan dalam suatu *database* pencarian khusus yang ditata sebagai sebuah *inverted index*. *Index* ini merupakan konversi dari dokumen asli yang mengandung sekumpulan kata ke dalam daftar kata yang berasosiasi dengan dokumen terkait dimana kata-kata tersebut muncul [7].

Preprocessing merepresentasikan koleksi dokumen kedalam bentuk tertentu untuk memudahkan dan mempercepat proses pencarian dan penemuan kembali dokumen yang relevan. Pembangunan *index* dari koleksi dokumen merupakan tugas pokok pada tahapan preprocessing di dalam IR. Kualitas *index* mempengaruhi efektifitas dan efisiensi sistem IR. *index* dokumen adalah himpunan term yang menunjukkan isi atau topik yang dikandung oleh dokumen. *Index* akan membedakan suatu dokumen dari dokumen lain yang berada di dalam koleksi. Ukuran *index* yang kecil dapat memberikan hasil buruk dan mungkin beberapa item yang relevan terabaikan, *index* yang besar memungkinkan ditemukan banyak dokumen yang relevan tetapi sekaligus dapat menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pencarian (*searching*) [4].

Pembuatan *inverted index* harus melibatkan konsep *linguistic processing* yang bertujuan mengekstrak term-term penting dari dokumen yang

direpresentasikan sebagai *bag-of-words*. Konsep linguistic processing terdiri dari tokenizing, filtration (stop list),stemming, dan *parse query* [5].

II.3.1 Tokenizing

Tokenization adalah proses memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi *token* atau potongan kata tunggal atau *termmed word* yang berdiri sendiri. Di dalam *tokenizing* karakter dan symbol selain a-z dihilangkan, pemecahan kalimat dan kata dilakukan berdasarkan pada spasi di dalam kalimat tersebut. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*).

Contoh proses *tokenizing*

Input: Sinyal adalah besaran fisis yang berubah menurut waktu, ruang, atau variabel-variabel bebas lainnya.

Output: sinyal adalah besaran fisis yang berubah menurut waktu ruang atau variabel variabel bebas lainnya

II.3.2 Filtration

Filtration atau *stop word removal* merupakan merupakan proses lanjutan dari *tokenizing* di dalam *preprocessing* kalimat. Proses *filtration* merupakan proses untuk menghilangkan kata yang ‘tidak relevan’ pada hasil *parsing* sebuah dokumen teks dengan cara membandingkannya dengan *stoplist* yang ada. *Stoplist* disebut juga dengan *stopword*. *Stoplist* berisi sekumpulan kata yang 'tidak relevan', namun sering sekali muncul dalam sebuah dokumen. Dengan kata lain *Stoplist* berisi sekumpulan *stopword* [6].

Stopword adalah daftar kata-kata yang tidak dipakai di dalam pemrosesan bahasa alami. Hasil penelitian sebelumnya menyatakan bahwa penggunaan *stopword* meningkatkan kemampuan pemrosesan bahasa alami. Kata-kata hasil tokenisasi yang termasuk dalam *stopword* selanjutnya dihilangkan [2].

Contoh Proses *Filtration*:

Input :

Secara matematis, sinyal adalah fungsi dari satu atau lebih variabel independen. Proses ini dilakukan melalui pemodelan sinyal.

Outpu :

matematis sinyal fungsi satu lebih variable independen proses dilakukan pemodelan sinyal

II.3.3 Stemming

Stemming adalah proses pencarian bentuk dasar suatu kalimat dengan cara menghilangkan imbuhannya. *Stemming* merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu.

Stemming sangat penting dalam mendukung efektivitas pencarian informasi dalam bahasa Indonesia, penerjemahan dokumen, dan pencarian dokumen teks. Imbuhan bahasa Indonesia lebih kompleks dari pada bahasa inggris karena di dalam bahasa Indonesia terdapat awalan (*prefiks*), *infiks* (sisispan), akhiran (*sufiks*), konfiks (gabungan *prefiks* dan *sufiks*). Sehingga stemming bahasa Indonesia harus mampu menemukan akar kata sesuai dengan aturan baku bahasa Indonesia [3]. Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

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

Contoh proses Stemming :

Sebelum Stemming:

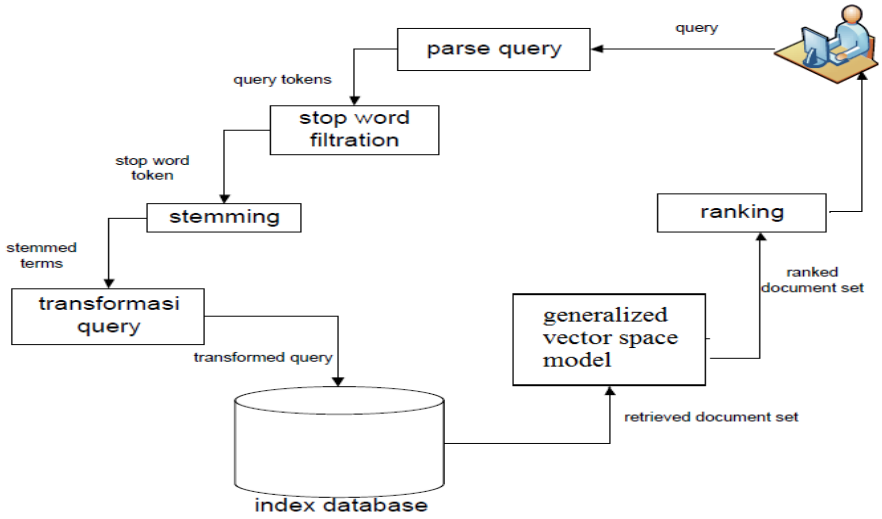
Matematis sinyal berfungsi satu atau lebih variabel independen maka proses dilakukan pemodelan sinyal

Setelah Stemming:

matematis sinyal fungsi satu atau lebih variable independen maka proses laku model sinyal

II.4 Searching Process

Setelah proses *preprocessing* dilakukan, selanjutnya adalah melakukan *searching process*. *Searching process* terbagi menjadi dua tahapan besar yaitu parse *query* dan tahapan pemodelan menggunakan *generalized vector space model*. adalah gambar ilustrasi proses pencarian dalam information retrieval.



Gambar II.2 Skema Searching Process

Parse *query* adalah tahapan preprocessing yang merubah kalimat *query* kedalam kelompok kata (term) dengan menggunakan beberapa tahapan yaitu :

1. Tokenizing

Tokenizing adalah proses pengenalan *token* yang terdapat dalam rangkaian teks. Proses tokenisasi adalah pemecahan sebuah kalimat menjadi kata(*token*) yang berdiri sendiri. Di dalam *tokenizing* karakter dan symbol selain a-z dihilangkan, pemecahan kalimat dan kata dilakukan berdasarkan pada spasi di dalam kalimat tersebut.

2. Proses *Stopword filtration*

Token-token *query* yang telah dihasilkan pada proses parse *query* kemudian di filter melalui proses pembuangan *token* yang termasuk *Stopword.*

3. Proses *Stemming*

Setelah proses *filtering* pada *Stop Word Filtration* maka token-token *query* yang dihasilkan akan dirubah ke bentuk dasarnya menjadi *stemmed term query*.

Sistem akan membandingkan *Stemmed* term *query* yang dihasilkan tersebut dengan koleksi dokumen sehingga mengembalikan dokumen-dokumen yang relevan. Setelah proses *parse query* dilakukan tahapan selanjutnya adalah memodelkan *query* dengan kumpulan term menggunakan salah satu metode dari information retrieval. Metode yang digunakan pada penelitian ini adalah metode *Generalized Vector Space* model.Tiap term atau kata yang ditemukan pada dokumen dan *query* diberi bobot dan disimpan sebagai salah satu elemen vektor dan dihitung nilai kemiripan antara *query* dan dokumen. Setelah itu perankingan dokumen dapat dilakukan berdasarkan kemiripan antara *query* dan dokumen.