BAB II
LANDASAN TEORI

Pada bab ini akan dibahas semua teori yang berhubungan dengan *Information Retrieval System.*

2.1 Sistem Temu Kembali Informasi

Sistem temu kembali informasi *(information retrieval)* adalah sebuah tindakan, metode dan prosedur untuk menemukan kembali data yang tersimpan dalam *database*, dan selanjutnya menyediakan informasi mengenai subyek yang dibutuhkan. Informasi yang ditemukan adalah informasi yang relevan berdasarkan kata kunci yang diinputkan oleh pengguna. Manning (2009), menjelaskan *Information Retrieval* adalah proses menemukan kembali informasi yang biasanya berbentuk dokumen dari lingkungan bersifat tidak terstruktur (biasanya teks) untuk memenuhi kebutuhan informasi dari dalam koleksi dokumen yang besar. Menurut Ramadhany (2008), sistem temu kembali informasi adalah salah satu cara untuk mendapatkan informasi yang akurat dan relevan dengan membuat perhitungan untuk menentukan apakah sebuah informasi relevan dengan kebutuhan penggunanya.

Tujuan dari sistem temu kembali informasi adalah memenuhi kebutuhan informasi pengguna dengan me-*retrieve* semua dokumen yang mungkin relevan, pada waktu yang sama melakukan *retrieve* sesedikit mungkin dokumen yang tak-relevan. Dokumen sebagai objek data dalam sistem temu kembali informasi merupakan sumber informasi.

2.2 Tujuan dan Fungsi Sistem Temu Kembali Informasi

Sistem Temu Kembali Informasi dibuat untuk menemukan dokumen atau informasi yang diperlukan oleh pengguna. Sistem Temu Kembali Informasi

bertujuan untuk menjembatani kebutuhan informasi pengguna dengan sumber informasi yang tersedia. Seperti dikemukakan oleh Belkin (1980) sebagai berikut:

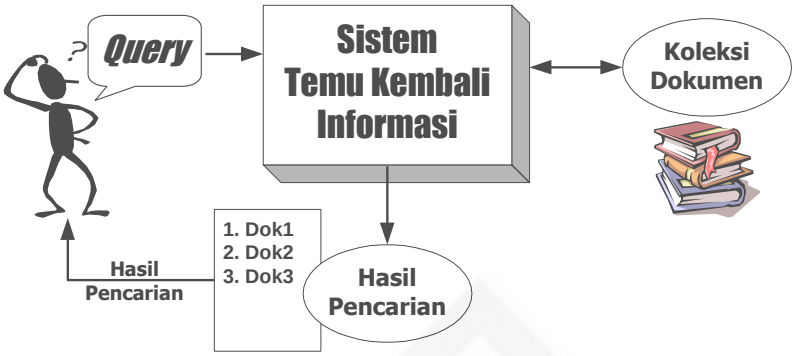
1. Penulis mempresentasikan sekumpulan ide dalam sebuah dokumen menggunakan sekumpulan konsep atau aturan yang berlaku dalam sistem temu kembali informasi.
2. Penulis merepresentasikan ide dari beberapa pengguna yang memerlukan ide tersebut untuk kebutuhannya namun pengguna tersebut tidak bisa merepresentasikan ide nya dengan baik dan benar.
3. Sistem temu kembali informasi bertujuan untuk mempertemukan ide yang dikemukakan oleh penulis dalam dokumen dengan kebutuhan informasi pengguna yang dinyatakan dalam bentuk pertanyaan (*query*).

Peran sistem temu kembali informasi jika dikaitkan dengan sumber informasi dan kebutuhan informasi pengguna adalah:

1. Menganalisis isi sumber informasi dan pertanyaan pengguna *(query).*
2. Mempertemukan pertanyaan pengguna dengan sumber informasi untuk mendapatkan dokumen yang relevan.

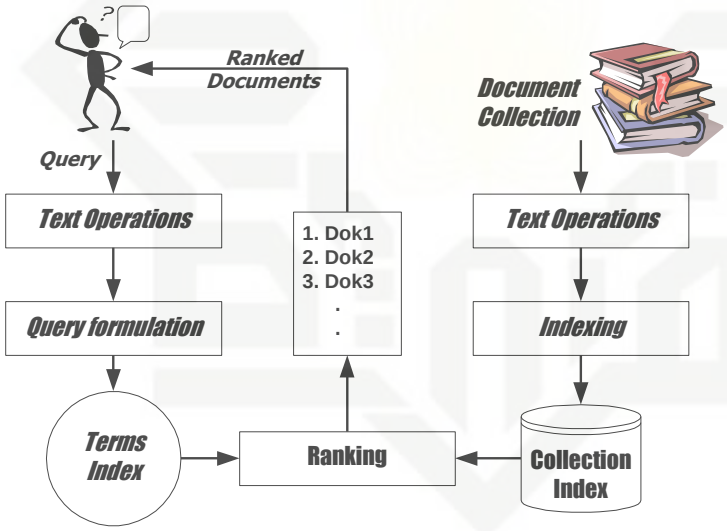
Adapun fungsi utama Sistem Temu Kembali Informasi seperti dikemukakan oleh Lancaster (1979) adalah sebagai berikut:

1. Mengidentifikasi sumber informasi yang relevan dengan kebutuhan pengguna.
2. Menganalisis isi sumber informasi (dokumen).
3. Merepresentasikan pertanyaan *(query)* pengguna dengan cara tertentu (aturan) yang memungkinkan untuk dipertemukan sumber informasi yang terdapat dalam basis data.
4. Mempertemukan pencarian dengan data yang tersimpan dalam basis data.
5. Menemu-kembalikan informasi yang relevan dengan *query*.



Gambar 2.1 Ilustrasi Sistem Temu Kembali Informasi
(Mandala dan Setiawan, 2002)

Sebagai suatu sistem, sistem temu kembali informasi memiliki beberapa bagian yang membangun sistem secara keseluruhan. Gambaran bagian-bagian yang terdapat pada suatu sistem temu kembali informasi digambarkan pada Gambar 2.2.



Gambar 2.2 Bagian-bagian Sistem Temu Balik Informasi
(Mandala dan Setiawan, 2002)

Dari gambar 2.2 terdapat dua buah alur operasi pada sistem temu kembali informasi. Alur pertama dimulai dari koleksi dokumen dan alur kedua dimulai dari *query* pengguna. Alur pertama yaitu pemrosesan terhadap koleksi dokumen menjadi basis data indeks yang tidak tergantung pada alur kedua. Sedangkan alur kedua tergantung dari keberadaan basis data indeks yang dihasilkan pada alur pertama.

Bagian-bagian dari sistem temu balik informasi menurut gambar 2.2 meliputi :

1. Text Operations (operasi terhadap teks) yang meliputi pemilihan kata atau kalimat dalam *query* maupun dokumen (*term selection*) dalam pemisahan *query* menjadi *term*s index (indeks dari kata-kata).
2. Indexing (pengindeksan), membangun basis data indeks dari koleksi dokumen. Dilakukan terlebih dahulu sebelum pencarian dokumen dilakukan.
3. *Query* formulation (formulasi terhadap *query*) yaitu memberi bobot pada indeks kata-kata *query*.
4. Ranking (perangkingan), mengurutkan dokumen tersebut berdasarkan kesesuaiannya dengan *query*.

2.3 Arsitektur Sistem Temu Kembali Informasi

Arsitektur dalam sistem temu kembali informasi antara lain adalah koleksi dokumen, *preprocessing* dan pembuatan index.

2.1. 2.3.1 Koleksi Dokumen (corpus)

Korpus adalah kumpulan dari potongan-potongan teks bahasa dalam bentuk elektronik, dipilih sesuai dengan kriteria *query* untuk mewakili satu atau berbagai ragam bahasa sebagai sumber data untuk penelitian linguistik. (Sinclair, 2004)

2.2. 2.3.2 Pre-Processing

Pre-processing dilakukan setelah dataset terbentuk. Tahapan dalam preprocessing data meliputi tokenizing, filtering dan stemming. Setiap tahapan mememiliki proses tersendiri dalam mengelola *term* dan kemudian disimpan kedalam sebuah variabel (Darmawan, 2011).

2.3. 2.3.3 Pembuatan Index

Pembangunan *index* dari koleksi dokumen merupakan tugas pokok pada tahapan *preprocessing* di dalam *information retrieval system*. *Index* dokumen

adalah himpunan *term* yang menunjukkan isi atau topik yang dikandung oleh dokumen. Menurut Cios dkk (2007), pembuatan *inverted index* melibatkan konsep *linguistic preprocessing* yang bertujuan mengekstrak *term-term* penting dari dokumen yang direpresentasikan sebagai *bag of words*.

Ekstraksi *term* biasanya melibatkan dua operasi utama berikut:

1. Penghapusan *stop-words*. *Stop-words* didefinisikan sebagai *term* yang tidak berhubungan (*irrelevant*) dengan subyek utama dari database meskipun kata tersebut seringkali hadir di dalam dokumen. Kata-kata tersebut termasuk kata penghubung, kata depan, dan sejenisnya. Contoh *stop-words* adalah ini, itu, dia, kami, pada, juga, jika, karena, meskipun, dan sebagainya.
2. *Stemming*.

   Kata yang muncul di dalam dokumen sering mempunyai banyak varian makna. Karena itu, setiap kata yang bukan *stop-words* direduksi ke bentuk *stemmed word* (*term*) yang cocok. Kata tersebut di-*stem* untuk mendapatkan bentuk dasarnya dengan menghilangkan awalan atau akhiran.

Menurut Manning dkk (2009) terdapat 4 langkah pembangunan *inverted index*, yaitu:

1. Mengumpulkan dokumen yang akan di-*index* (dikenal dengan nama *corpus*/koleksi dokumen).
2. Pemisahan rangkaian kata (*tokenization*).

   Pada tahapan ini, seluruh kata di dalam kalimat ataupun paragraf dipisahkan menjadi *token* atau potongan kata tunggal atau *termmed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lowercase*).
3. Melakukan *linguistic preprocessing* untuk menghasilkan *token*/*term* yang telah dinormalisasi. Dua hal yang dilakukan dalam tahap ini adalah:
   a. Penyaringan (*filtration*)

Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. *Term* yang sering dipakai tidak dapat digunakan untuk tujuan ini karena dua alasan. Pertama, jumlah dokumen yang relevan terhadap suatu *query* kemungkinan besar merupakan bagian kecil dari koleksi. *Term* yang efektif dalam pemisahan dokumen yang relevan dari dokumen tidak relevan kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. Ini berarti bahwa *term* dengan frekuensi kemunculan tinggi bersifat *poor descriminator*. Kedua, *term* yang muncul dalam banyak dokumen tidak mencerminkan definisi dari topik atau sub-topik dokumen. Karena itu, *term* yang sering digunakan dianggap sebagai *stopwords* dan dihapus dari dokumen.

b. Konversi *term* ke bentuk akar atau dasar (*stemming*)

Stemming adalah proses konversi *term* ke bentuk dasarnya. Hal ini bisa dilakukan dengan cara menghilangkan akhiran atau awalan dari sebuah kata. Tidak banyak algoritma yang dikhususkan untuk *stemming* bahasa Indonesia dengan berbagai keterbatasan di dalamnya, diantaranya:

a. Algoritma Porter, Algoritma ini membutuhkan waktu yang lebih singkat dibandingkan dengan *stemming* menggunakan Algoritma Nazief dan Adriani, namun proses *stemming* menggunakan Algoritma Porter memiliki presentase keakuratan (presisi) lebih kecil dibandingkan dengan *stemming* menggunakan Algoritma Nazief dan Adriani.
b. Algoritma Nazief dan Adriani, algoritma *stemming* untuk teks berbahasa Indonesia yang memiliki kemampuan presentase keakuratan (presisi) lebih baik dari algoritma lainnya. Algoritma ini sangat dibutuhkan dan menentukan dalam proses sistem temu kembali informasi dalam dokumen Indonesia. Algoritma

Nazief dan Adriani mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan. Pengelompokan ini termasuk imbuhan di depan (awalan), imbuhan kata di belakang (akhiran), imbuhan kata di tengah (sisipan) dan kombinasi imbuhan pada awal dan akhir kata (konfiks).

Tabel 2.1 berikut akan menunjukan secara lengkap aturan pemenggalan kata yang dilakukan oleh Nazief dan Adriani (Tahitoe, 2010).

Tabel 2.1 Aturan pemenggalan kata Nazief dan Adriani

Aturan	**Format Kata**	**Pemenggalan**
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... dimana C!="r" & P!="er"
3	berCAerV...	ber-CaerV... dimana C!="r"
4	Belajar	bel-ajar
5	beC1erC2...	be-C1erC2... dimana C1!={"r"|"l"}
6	terV...	ter-V... | te-rV..
7	terCerV...	ter-CerV... dimana C!="r"
8	terCP...	ter-CP... dimana C!="r" dan P!="er"
9	teC1erC2...	te-C1erC2... dimana C1!="r"
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe{r|l}...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV
16	meng{g|h|q}...	meng-{g|h|q}..
17	mengV...	meng-V... | meng-kV...
18	menyV...	meny-sV…

Aturan	Format Kata	Pemenggalan
19	mempV...	mem-pV... dengan V!="e"
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
22	perCAP	per-CAP... dimana C!="r" dan P!="er"
23	perCAerV...	per-CAerV... dimana C!="r"
24	pem{b|f|V}...	pem-{b|f|V}...
25	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	peng{g|h|q}...	peng-{g|h|q}...
29	pengV...	peng-V... | peng-kV...
30	penyV...	peny-sV…
31	pelV...	pe-lV... kecuali "pelajar" yang menghasilkan "ajar"
32	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
33	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!="er"

Keterangan simbol huruf :

C : huruf konsonan

V : huruf vokal

A : huruf vokal atau konsonan

P : partikel atau fragmen dari suatu kata, misalnya "er"

c. *Algoritma Confix Stripping Stemmer* Algoritma *Confix Stripping (CS) Stemmer* dikembangkan oleh Jelita Asian (2007) dengan referensi dari algoritma *stemming* Nazief-Adriani untuk memperbaiki kesalahan-kesalahan *stemming* yang masih dilakukan. Kesalahan-kesalahan tersebut adalah sebagai berikut:

1. Tidak terdapat aturan pemenggalan awalan untuk kata-kata dengan format "mempeng...", "mengk...", "ter*C1*er*C2*...", "pe*C1*er*C2*...",

2. Tidak dapat untuk melakukan proses pemenggalan kata-kata dengan bentuk perulangan, misalnya pada kata "buku-buku".

3. Algoritma Nazief-Adriani melakukan proses *stemming* dengan menghilangkan akhiran terlebih dahulu, kemudian penghilangan awalan. Langkah yang seharusnya dilakukan adalah penghilangan awalan terlebih dahulu yang dilakukan oleh *Confix Stripping*.

Tabel 2.2 berikut aturan pemenggalan kata yang diperbaiki dan ditambahkan oleh *Confix Stripping* dengan referensi tabel aturan *stemming* Nazief dan Adriani (Tahitoe, 2010).

Aturan	**Format Kata**	**Pemenggalan**
12	mempe...	mem-pe...
16	meng{g|h|q|k}...	meng-{g|h|q|k}...
34	terC1erC2...	ter-C1erC2... dimana C1!= „r‟
35	peC1erC2...	pe-C1erC2... dimana C1!={r|w|y|l|m|n}

d. *Algoritma Enhanded Confix Stripping (ECS) Stemmer* menurut Tahitoe (2010) setelah dilakukan beberapa percobaan dan analisis ditemukan beberapa kata yang tidak dapat di-*stemming* menggunakan *Confix Stripping Stemmer*. Analisis terhadap kata-kata yang gagal di-*stemming* tersebut sebagai berikut :

1) Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format "mem+p...", "men+s...", dan "peng+k...". Hal ini terjadi pada kata "mempromosikan", "memproteksi", "mensyaratkan", "mensyukuri", dan "pengkajian".

2) Kurang relevannya beberapa aturan pada algoritma yang sudah ada sebelum algortima ECS untuk pemenggalan awalan pada kata-kata dengan format "menge+kata dasar" dan "penge+kata dasar", seperti pada kata "mengerem" dan "pengeboman".
3) Adanya elemen pada beberapa kata dasar yang menyerupai suatu imbuhan. Kata-kata seperti "perpolitikan", dan "pelaku" gagal distemming karena akhiran "-kan" dan "-ku".

Untuk memperbaiki kesalahan-kesalahan tersebut, algoritma *ECS Stemmer* melakukan beberapa buah perbaikan sebagai berikut :

1) Menambahkan suatu algoritma tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Algoritma ini disebut *loop Pengembalian Akhiran*. Algoritma *loop Pengembalian Akhiran* dideskripsikan sebagai berikut:
   a. Kembalikan seluruh awalan yang telah dihilangkan sebelumnya.
   b. Kembalikan akhiran sesuai dengan urutan model pada bahasa Indonesia.
   c. Lakukan pengecekan di kamus kata dasar. Apabila ditemukan, proses dihentikan. Apabila tidak ditemukan, maka lakukan proses pemenggalan awalan.
   d. Lakukan *recoding* apabila diperlukan.
   e. Apabila pengecekan di kamus kata dasar tetap gagal setelah *recoding*, maka awalan-awalan yang telah dihilangkan dikembalikan lagi.

Tabel 2.3 berikut aturan pemenggalan kata yang diperbaiki dan ditambahkan oleh ECS berdasarkan tabel aturan pemenggalan Nazief dan Adriani (Tahitoe, 2010).

Aturan	**Format Kata**	**Pemenggalan**
14	men{c|d|j|s|z}...	men-{c|d|j|s|z}...
17	mengV	meng-V... | meng-kV... | (mengV-... jika V=‟e‟)
19	mempA...	mem-pA... dengan A!=‟e‟
28	pengC...	peng-C...
29	pengV	peng-V... | peng-kV... | (pengV-... jika V=‟e‟)

Tabel 2.4 berikut akan menunjukan secara lengkap aturan pemenggalan kata yang dilakukan oleh ECS yang telah diperbaiki berdasarakan aturan pemenggalan dari algoritma Nazief dan Adriani untuk menghasilkan proses *stemming* yang lebih baik (Tahitoe, 2010).

Tabel 2.4 Aturan pemenggalan kata ECS

Aturan	**Format Kata**	**Pemenggalan**
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... dimana C!=‟r‟ & P!=‟er‟
3	berCAerV...	ber-CaerV... dimana C!=‟r‟
4	Belajar	bel-ajar
5	beC1erC2...	be-C1erC2... dimana C1!={‟r‟|‟l‟}
6	terV...	ter-V... | te-rV..
7	terCerV...	ter-CerV... dimana C!=‟r‟
8	terCP...	ter-CP... dimana C!=‟r‟ dan P!=‟er‟
9	teC1erC2...	te-C1erC2... dimana C1!=‟r‟
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe...	mem-pe...

Aturan	Format Kata	Pemenggalan
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|s|z}...	men-{c|d|j|s|z}...
15	menV...	me-nV... | me-tV
16	meng{g|h|q|k}...	meng-{g|h|q|k}...
17	mengV...	meng-V... | meng-kV... | (mengV-... jika V=‟e‟)
18	menyV...	meny-sV…
19	mempA...	mem-pA... dengan A!=‟e‟
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
22	perCAP	per-CAP... dimana C!=‟r‟ dan P!=‟er‟
23	perCAerV...	per-CAerV... dimana C!=‟r‟
24	pem{b|f|V}...	pem-{b|f|V}...
25	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	pengC...	peng-C...
29	pengV...	peng-V... | peng-kV... | (pengV-... jika V=‟e‟)
30	penyV...	peny-sV…
31	pelV...	pe-lV... kecuali “pelajar” yang menghasilkan “ajar”
32	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
33	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!=‟er‟
34	terC1erC2...	ter-C1erC2... dimana C1!= „r‟
35	peC1erC2...	pe-C1erC2... dimana C1!={r|w|y|l|m|n}

Keterangan simbol huruf :

C : huruf konsonan

V : huruf vokal

A : huruf vokal atau konsonan

P : partikel atau fragmen dari suatu kata, misalnya "er"

4. Pemberian bobot terhadap *term*

Setiap *term* diberikan bobot sesuai dengan skema pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya. Jika menggunakan pembobotan lokal maka, pembobotan *term* diekspresikan sebagai *tf* (*term frequency*). Namun, jika pembobotan global yang digunakan maka, pembobotan *term* didapatkan melalui nilai *idf* (*inverse document frequency*).

a. Pembobotan TF-IDF (kombinasi)

Analisis tahap penentuan seberapa jauh keterhubungan antar kata-kata pada dokumen yang ada dengan menghitung frekwensi term pada dokumen. Tahap ini disebut juga tahap pembobotan, yaitu dijelaskan sebagai berikut:

1. Pembobotan Term

Term adalah suatu kata atau suatu kumpulan kata yang dalam *information retrieval* sebuah term perlu diberi bobot, karena semakin sering suatu term muncul pada suatu dokumen maka kemungkinan term tersebut semakin penting dalam dokumen. Dari proses pembobotan term maka akan didapatkan hasil akhir berupa Term Frequency (TF), yaitu merupakan frekwensi atau jumlah masing masing kata. Hasil pembobotan term kemudian akan digunakan sebagai dasar perhitungan pada basis Term Frequency-Inverse Document Frequency (TF-IDF).

Term Frequency-Inverse Document Frequency (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. Basis ini menggabungkan dua konsep untuk perhitungan bobot, yaitu Term Frequency (TF) merupakan frekwensi

kemunculan kata (t) pada dokumen (d). Document Frequency (DF) adalah banyaknya dokumen yang mengandung kata (t). TF-IDF dapat dirumuskan sebagai berikut:

TF-IDF (tk, dj) = TF (tk, dj) * IDF (tk) **(2.1)**

Keterangan:

dj = Dokumen ke-j.

tk = Term ke-k.

Dimana sebelumnya dihitung terlebih dahulu Term Frequency (TF) yaitu frekwensi kemunculan suatu term ditiap dokumen. Kemudian dihitung Inverse Document Frequency (IDF) yaitu nilai bobot suatu term dihitung dari seringnya suatu term muncul dibeberapa dokumen. Semakin sering suatu term muncul dibanyak dokumen, maka nilai IDF-nya akan kecil. Berikut rumus-rumus TF dan IDF:

TF (tk, dj) = f (tk, dj) **(2.2)**

Keterangan:

TF = Jumlah frekwensi term.

F = Jumlah frekwensi kemunculan.

dj = Dokumen ke-j.

tk = Term ke-k.

Kemudian untuk menghitung nilai IDF bisa menggunakan persamaan sebagai berikut:

IDF(tk) = 1 / df (t) Atau IDF (tk) = log (N / df (t)) **(2.3)**

Keterangan:

IDF = Bobot term.

N = Jumlah total dokumen.

df = Jumlah kemunculan dokumen.

dj = Dokumen ke-j.

tk = Term ke-k.

Persamaan pertama hanya boleh digunakan apabila hanya terdapat satu buah dokumen saja yang diproses sedangkan

persamaan kedua digunakan pada proses yang melibatkan banyak dokumen.

Kemudian hitung bobot tiap term i di dokumen j dengan rumus berikut.

$$\mathbf{Wi,j = TF\ i,j\ x\ IDF} \qquad \mathbf{(2.4)}$$

2.3.4 Ekspansi *Query*

Ekspansi Query dengan menggunakan kamus bahasa Indonesia yang mempunyai daftar sinonim dan hiponim mampu meningkatkan jumlah dokumen hasil temu kembali informasi. *Ekspansi Query* dengan kamus kata sinonim dan hiponim perlu dilakukan untuk mengatasi kata yang tidak mampu dikembalikan oleh sistem (Thamrin, 2015).

Ekspansi Query dilakukan dengan memeriksa inputan dari pengguna apakah kata tersebut merupakan kata sinonim atau hiponim. Jika inputan merupakan sinonim atau hiponim maka inputan tersebut akan dikembalikan dengan daftar kata sinonim dan hiponim yang akan menjadi tambahan sebagai *query*.

Sinonim dalam Kamus Besar Bahasa Indonesia dapat diartikan bentuk bahasa yang maknanya mirip atau sama dengan bentuk bahasa lain. Misalnya perahu adalah sinonim dari bahtera, binatang adalah sinonim dari hewan, cadar adalah sinonim dari burkak.

Hiponim dalam Kamus Besar Bahasa Indonesia adalah hubungan antara makna spesifik antara anggota taksonomi dan nama taksonomi. Misalnya Kucing, Ayam, Kelinci adalah hiponim dari Hewan. Khuldi, anggur, jeruk, nenas, markisa, kurma, delima, pisang, zaitun, tin adalah hiponim dari buah.

2.4 Model Dalam Sistem Temu Kembali Informasi

Beberapa model yang sering digunakan dalam sistem temu kembali informasi adalah model boolean, model ruang vektor dan model probabilistik.

2.4. 2.4.1 Model Boolean

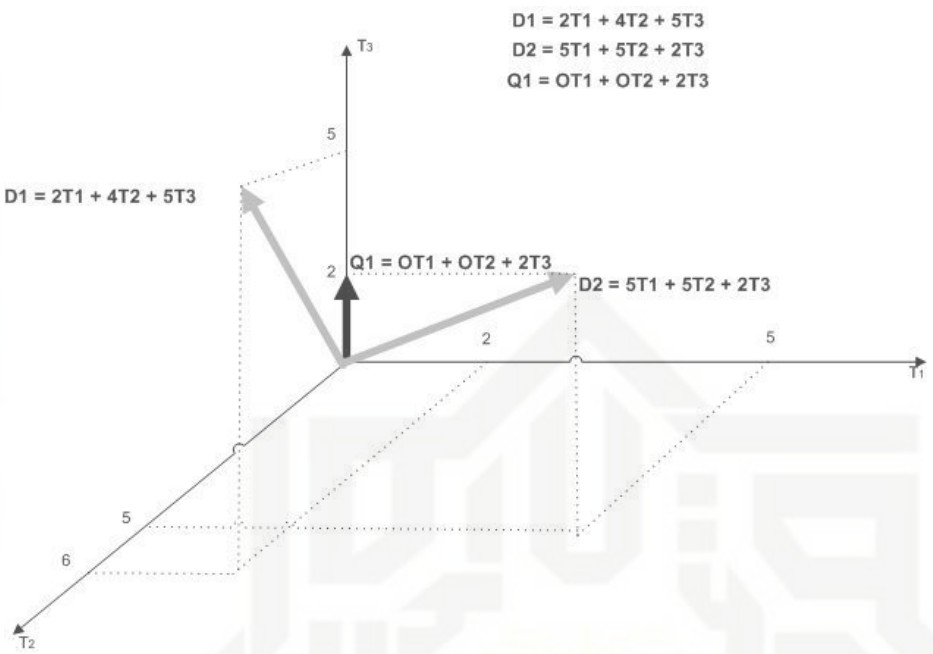
Model boolean, dalam sistem temu kembali informasi, merupakan salah satu model dimana proses pencarian informasi dari *query* yang diterima diperlakukan dengan ekspresi *boolean*. Ekspresi boolean yang dimaksud dapat berupa *operator* logika *AND, OR,* dan *NOT*. Dokumen yang di *retrieve* adalah dokumen yang benar-benar sesuai dengan *query*.

Kelemahan model *boolean*, yaitu:

1. Semua *term* dalam *query* dianggap mempunyai bobot yang sama, sehingga tidak ditetapkan tingkat kepentingan untuk *term* dalam suatu *query*.
2. Tidak dilakukan perangkingan (peringkat) terhadap dokumen yang terambil.
3. Tidak bisa menyelesaikan *partial matching* pada *query*. Dokumen yang terambil hanya dokumen yang benar.
4. Benar sesuai dengan pernyataan boolean atau *query* yang diberikan.

2.5. 2.4.2 Model Ruang Vektor (Vector Space Model – VSM)

Menurut Bunyamin (2005), dalam sistem temu kembali informasi, kemiripan antar dokumen didefinisikan berdasarkan representasi *bag of words* dan dikonversikan ke suatu model ruang vektor (*vector space model* - VSM). Pada VSM, setiap dokumen di dalam *database* dan *query* pengguna direpresentasikan oleh suatu vektor multi-dimensi seperti yang ditunjukkan oleh Gambar 2.3 berikut ini:



Gambar 2.3 Contoh VSM Dengan Dua Dokumen D_1, D_2, Dan *Query* Q_1 (Sumber: Bunyamin, 2005)

Penentuan relevansi dokumen dengan *query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara vektor dokumen dengan vektor *query*. Model ruang vektor sangat mementingkan *frequency* kemunculan *term* yang dicari pada dokumen. Namun tidak diimbangi dengan faktor panjang dokumen itu sendiri. Yang menyebabkan dokumen yang lebih panjang memiliki kemungkinan lebih besar untuk menjadi dokumen yang lebih relevan karena kemunculan *term* yang lebih banyak.

Maka perlu dilakukan normalisasi, yaitu membagi relevansi dokumen yang didapatkan dengan perkalian panjang vektor *query* dengan panjang vektor dokumen bersangkutan. Panjang vektor *query* atau dokumen secara matematis adalah akar kuadrat dari penjumlahan nilai kuadrat dari panjang vektor linier pembentuk *query* atau dokumen. Dan berikut adalah rumus untuk normalisasi bobot *term*, menghitung bobot setiap *query*, serta menghitung normalisasi pembobotan *query* untuk mendapatkan pengukuran kesamaan (*similarity measure*) antara vektor dokumen dengan vektor *query*.

1. Normalisasi panjang *term*, merupakan jumlah perbandingan antara bobot *term* terhadap panjang *term*.

$$\bar{w}_{i,j} = \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}} \quad \textbf{(2.5)}$$

2. Pembobotan masing-masing *query*

$$w_{i,q} = \left(0.5 + \frac{0.5 \text{freq}_{i,q}}{1}\right) \text{x } idf(i) \quad \textbf{(2.6)}$$

3. Normalisasi *query*, merupakan jumlah perbandingan antara bobot *query* terhadap panjang *query*.

$$\bar{w}_{i,q} = \frac{w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \quad \textbf{(2.7)}$$

Perhitungan kesamaan antara vektor *query* dan vektor dokumen (*similarity*) dilihat dari sudut yang kecil. Sudut yang dibentuk oleh dua buah vektor dapat dihitung dengan rumus:

$$R(Q,D) = \cos\theta = \frac{D.Q}{|Q||D|} \quad \textbf{(2.8)}$$

Dimana:

D = normalisasi *query* pada dokumen
Q = normalisasi *query*
|Q| = panjang *query*
|D| = panjang ayat yang mengandung *query*

2.6. 2.4.3 Model Probabilistik

Model probabilistik adalah model *Information retrieval system* yang mengurutkan dokumen dalam urutan menurun terhadap peluang relevansi sebuah dokumen terhadap informasi yang dibutuhkan. Beberapa model yang juga dikembangkan berdasarkan perhitungan probabilistik yaitu, *Binary Independence Model*, model Okapi BM25, dan *Bayesian Network Model* (Manning dkk, 2009).

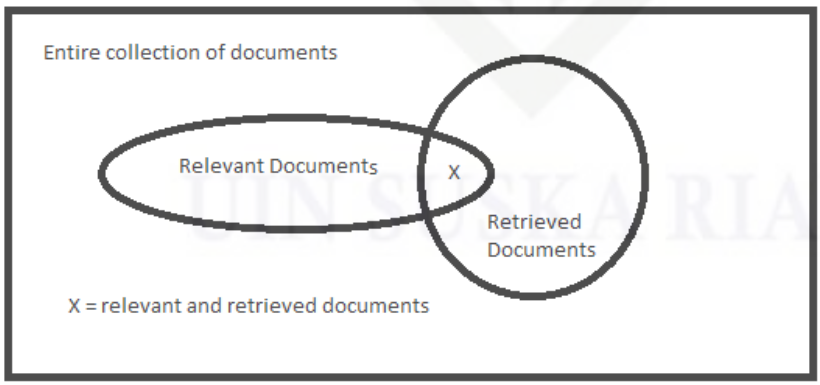
2.5 Kualitas Text Retrieval

Terdapat dua kategori dokumen yang dihasilkan oleh sistem temu kembali informasi terkait pemrosesan *query*, yaitu *relevant documents* (dokumen yang relevan dengan *query*) dan *retrieved documents* (dokumen yang diterima pengguna). Ukuran umum yang digunakan untuk mengukur kualitas dari *text retrieval* adalah kombinasi *precision* dan *recall*. *Precision* mengevaluasi kemampuan sistem temu kembali informasi untuk menemukan kembali dokumen *top-ranked* yang paling relevan, dan didefinisikan sebagai persentase dokumen yang di-*retrieve* yang benar-benar relevan terhadap *query* pengguna (Cios dkk, 2007).

$$\boldsymbol{precision} = \frac{\{\boldsymbol{relevant\ docs}\} \cap \{\boldsymbol{retrieved\ docs}\}}{\{\boldsymbol{retrieved\ docs}\}} \quad \textbf{(2.9)}$$

Recall mengevaluasi kemampuan sistem temu kembali informasi untuk menemukan semua dokumen yang relevan dari dalam koleksi dokumen dan didefinisikan sebagai persentase dokumen yang relevan terhadap *query* pengguna dan yang diterima oleh pengguna.

$$\boldsymbol{recall} = \frac{\{\boldsymbol{relevant\ docs}\} \cap \{\boldsymbol{retrieved\ docs}\}}{\{\boldsymbol{relevant\ docs}\}} \quad \textbf{(2.10)}$$



Gambar 2.4 Hubungan Antara *Relevant Documents* dan *Retrieved Documents* (Sumber: Cios dkk, 2007).

Menurut (Hinrich, 2008), pengujian kemampuan sistem *information retrieval* dilakukan dengan menghitung nilai *precision* dan *recall.* Nilai *precision* adalah keakurasian antara *query* dan dokumen yang dikembalikan oleh sistem temu kembali. Kemudian *recall* adalah proporsi jumlah dokumen yang dapat ditemu kembalikan oleh sebuah proses pencarian pada sistem temu kembali informasi berdasarkan *query*. Berikut ini adalah Tabel parameter untuk menghitung *precision* dan *recall*:

Tabel 2.5 Parameter Menghitung *Precision* dan *Recall*.

keterangan	***relevan***	**Tidak *relevan***
Terambil	*True positive* (tp)	False positve (fp)
Tidak terambil	false negative (fn)	True negative (tn)

Rumus untuk menghitung *Precission:*

$$\mathbf{P = tp / (tp + fn)} \qquad \mathbf{(2.11)}$$

Rumus 2.7 *Precission* (Sumber: Hinrich, 2008)

Keterangan :

P = *Precision*

tp = *true positive*

fn = *false negative*

Rumus untuk menghitung *Recall:*

$$\mathbf{R = tp / (tp + fp)} \qquad \mathbf{(2.12)}$$

Rumus 2.8 *Recall* (Sumber: Hinrich, 2008)

Keterangan :

R = *Recall*

Tp = *true positive*

Fn = *false positive*

2.6 Pengujian Untuk Menilai Kemampuan Sistem

Menurut (Ramadhany, 2008) Pengujian kemampuan sistem merupakan pengujian untuk memperoleh nilai kesesuaian antara *query* dan dokumen yang relevan. Oleh karena itu, dibutuhkan nilai perbandingan *precision* dan *recall*. Untuk menginterpretasikan nilai kesesuaian, ditetapkan tiga kategori yaitu: kesesuaian rendah, sedang dan tinggi. Kemudian tolak ukur yang digunakan adalah skala interval, dengan mencari selisih kemungkinan angka kesesuaian tinggi (1) dengan kemungkinan kesesuaian rendah (0) dibagi dengan 3 sesuai dengan kategori penilaian seperti:

(1 – 0) : 3 = 0.33

Dari hasil pembagian tersebut dapat disimpulkan nilai interval untuk menentukan sistem *information retrieval* memiliki nilai kesesuaian yang baik dalam hal *precision* dan *recall* adalah sebagai berikut:

1. Kerelevanan rendah berada apabila angka pada rentang 0.00- 0.33.
2. Kerelevanan sedang berada apabila angka pada rentang 0.34- 0.66.
3. Kerelevanan tinggi berada apabila angka pada rentang 0.67-1.00.