Unified Messaging System
Information Retrieval & Klasifikasi Teks

Paper ini menjelaskan konsep dasar yang berkaitan erat dengan penelitian saya mengenai *content server* pada suatu *Unified Messaging System* (UMS). Sebagai gambaran, content server yang sedang dikembangkan mengelola content iklan baris yang mempunyai sifat unik dan tidak terstruktur. *Content server* ini menerima query atau pesan teks dari pengguna UMS (dalam hal ini adalah UMS NoteBOX di Laboratorium Sistem Tersebar, Informatika, STEI ITB, Bandung) dan web. Query yang diterima oleh server diproses untuk mendapatkan jawaban yang tepat (dengan nilai presisi dan *recall* tinggi), kemudian jawaban tersebut dikembalikan kepada pengguna atau di*forward* ke suatu alamat email, sms atau fax. Pesan dari pengguna juga dapat ditambahkan sebagai content baru, memperkaya sistem content dengan melibatkan pengguna secara aktif, sehingga tidak selalu bergantung pada pengelola content. Konsep yang dibahas mencakup sekilas tentang UMS NoteBOX dan iklan baris, *information retrieval* (IR) dan klasifikasi dokumen teks. Referensi yang dicantumkan diakhir tulisan dapat memberikan penjelasan yang *panjang x lebar x tinggi = volume* mengenai konsep yang disampaikan di sini ☺.

UMS NOTEBOX

Unified messaging system (UMS) adalah suatu solusi yang memungkinkan terwujudnya saling komunikasi dan pertukaran pesan antara berbagai layanan *messaging*. Sistem *messaging* memungkinkan pengguna mengirim dan menerima pesan secara asinkron sehingga pesan yang dikirim tidak langsung diterima oleh penerima tetapi akan disimpan terlebih dahulu oleh sistem sebelum diteruskan kepada penerima. Sistem demikian beroperasi menggunakan mekanisme *store-and-forward*. Aplikasi sistem *messaging* yang telah digunakan secara luas mencakup email, *voicemail*, SMS, MMS dan faksimili [1].

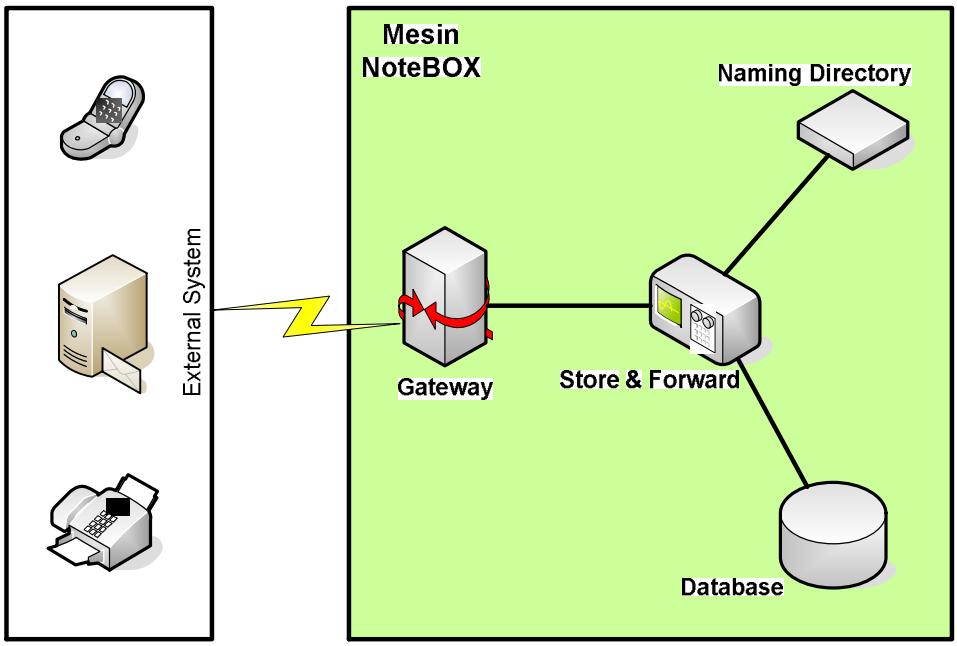
Contoh implementasi dari UMS adalah NoteBOX. Pengguna dapat mengirimkan pesan ke suatu tujuan melalui NoteBOX. Pengirim dan penerima pesan dapat menggunakan layanan *messaging* yang sama atau berbeda, misalnya pengiriman SMS ke suatu alamat email. NoteBOX akan menerima pesan dari pengirim, memeriksa tujuan dan menformat pesan agar sesuai dengan layanan penerima. Layanan *messaging* yang telah tersedia dan berjalan dengan baik pada NoteBOX adalah SMS, MMS, email, dan *search engine*. Semua layanan ini ditempatkan dalam satu mesin, dinamakan mesin NoteBOX [1,2].

NoteBOX dapat menerima dan meneruskan pesan ke berbagai layanan yang menggunakan format pesan berbeda tetapi NoteBOX hanya menyimpan satu format pesan yang seragam (*uniform*) selama operasionalnya. Pesan *uniform* ini terdiri dari:

1. Alamat pengirim pesan
2. Layanan messaging yang digunakan oleh pengirim pesan
3. Alamat penerima pesan
4. Layanan messaging yang digunakan oleh penerima pesan
5. Pesan yang diterima
6. Tipe pesan yang diterima (tipe MIME, misalnya text/plain, image/jpeg)

Mengikuti mekanisme *store-and-forward*, penanganan pesan di dalam NoteBOX dibagi ke dalam dua proses yaitu *store* dan *forward*. Proses *store* dimulai dengan pengambilan pesan dari sistem eksternal oleh gateway. Gateway kemudian memberikan tanda sedang diproses, mengubah pesan ke bentuk uniform dan mengirimkan pesan *uniform* tersebut ke core server. Core server menyimpan pesan yang diterima dari gateway, selanjutnya mengirimkan *delivery report PENDING* atau *FAILED* kepada gateway. Gateway akan menghapus pesan tersebut dari sistem eksternal jika menerima *delivery report PENDING* dan akan menghapus tanda sedang diproses jika menerima *delivery report FAILED*. Gateway hanya akan memroses pesan dari sistem eksternal yang tidak mempunyai tanda sedang diproses.

Proses *forward* dimulai oleh *core server* dengan mengambil pesan *uniform* dari database. *Core server* kemudian memberikan tanda sedang diproses pada pesan tersebut. Pesan uniform ini dikirimkan ke gateway yang sesuai dengan informasi layanan messaging penerima. Gateway mengubah pesan uniform ke format yang sesuai dengan sistem eksternal dan meneruskannya ke sistem eksternal. Setelah pesan diterima oleh sistem eksternal, gateway mengirimkan *delivery report DELIVERED* kepada core server. Core server akan menghapus pesan dari database jika menerima *delivery report DELIVERED*, dilanjutkan dengan mengirimkan *delivery report DELIVERED* kepada gateway yang menerima pesan tersebut. Jika menerima error, core server menghilangkan tanda sedang diproses dan mengirimkan *delivery report FAILED* kepada gateway penerima pesan. Core server hanya akan memroses pesan yang tidak mempunyai tanda sedang diproses.



Gambar 1 Arsitektur UMS NoteBOX

Gambar 1 memperlihatkan arsitektur dari NoteBOX [2]. Arsitektur tersebut menunjukkan adanya beberapa komponen yang digunakan dalam operasional NoteBOX, yaitu:

1. ***Gateway***. Komponen ini bertanggungjawab menghubungkan NoteBOX dengan sistem eksternal.

2. ***Naming Directory***. Komponen ini bertanggungjawab mengatur nama-nama yang berasosiasi dengan gateway termasuk penambahan nama, pengurangan nama, pembuatan kelas yang berasosiasi dengan nama tersebut, dan implementasi mekanisme object pool untuk gateway.

3. **Database**. Komponen ini bertanggungjawab menyimpan pesan dan informasi yang diperlukan oleh NoteBOX.

4. ***Store and forward***. Komponen ini bertanggungjawab mengeksekusi penerimaan pesan termasuk di dalamnya pemilihan *instance* gateway yang digunakan dalam penerimaan dan pengiriman pesan.

IKLAN BARIS

Iklan atau *advertising* merupakan suatu bentuk komunikasi yang ditujukan untuk membujuk penonton, pembaca atau pendengar untuk mengambil tindakan tertentu. Iklan biasanya menyertakan nama produk atau jasa yang akan menguntungkan pelanggan, mempengaruhi pelanggan potensial untuk membeli atau menggunakan merk tertentu. Banyak media dapat digunakan untuk menyampaikan pesan (iklan) termasuk pada media tradisional seperti surat kabar, majalah, televisi, radio, *billboard* dan surat [5].

Menurut *Cambridge Advanced Learner's Dictionary* [6], iklan baris atau *classified advertising* adalah suatu iklan kecil yang ditempatkan di dalam surat kabar atau majalah untuk menjual atau membeli sesuatu atau untuk memberikan tawaran pekerjaan. Iklan baris juga terdapat pada media online dan media lain yang terbit secara periodik. Iklan baris berbeda dengan iklan *standard* karena memungkinkan seseorang (bukan perusahaan) menawarkan produk dan jasanya. Iklan baris biasanya bersifat *text-only* dan berisi informasi singkat mengenai jenis item yang dijual dan nomor *telephone* yang dapat dihubungi, meskipun dapat mengandung informasi lebih detail seperti nama atau alamat yang dapat dikunjungi, deskripsi barang. Iklan baris terbagi ke dalam beberapa kelas yang ditandai oleh suatu *heading* yang mengklasifikasi produk atau jasa yang ditawarkan (contoh: Properti, Otomotif, Pakaian) dan dikelompokkkan secara

total pada bagian berbeda agar berbeda dari display advertising yang sering mengandung grafik atau karya seni lain [7].

Berikut ini adalah contoh beberapa iklan baris yang diambil dari surat kabar nasional dan daerah (Kompas, Pikiran Rakyat, dan Tribun Jabar):

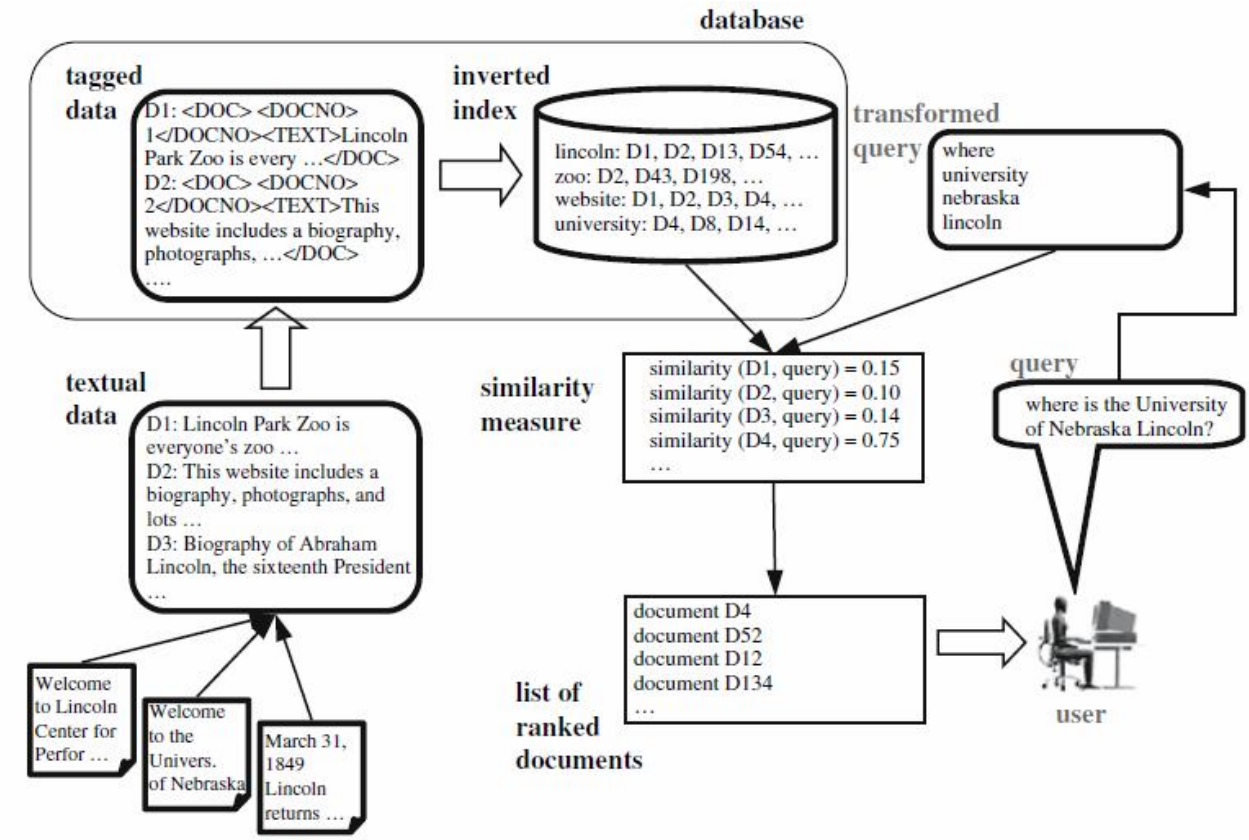
Dibeli Printer/LQ 2180 Lx 300Jt+comp N.Book T:703687292	Jl/Sw Apartment Cimbeuluit Setiabudi,Majesty,Braga FF C21 Avenue-Yesssi-08527899092
Laptop Toshiba Grs 1Th-93C:2Jtan "KAISAR" T:57378638/5747326	Kont.Toko 2,5x7m ramai,dktPasr,KopoSayati 93C-687979762
Jual MTR SUZUKU Shogun 02, HondaC800'82 Hrg:Nego Jl.Veteran 39-41	DIJUAL RUKO JL KOPO 284 Sebelah POM BENSIN-087967970879
READY STOCK Mio Vega Jup Mx, VxionPrsCpt,PrsytDjemputT:578676	
Jl H Sivic VVTI S 5A/Automatic'01 Hitam (D) Hrg:Nego T:6867879898	Cr:PENJAHIT u/Butik,Hub:Sasmitamaja 37 KbJati-0978787866
Jl Fiat 127'75 10Jt Siap Pakai Jl Permai II ME 76 Mrghy Prmi 08797979623	Video Klip(DVD,MiniDV,Beta) T.78788686
Jl Supra X 125 CW'06 Merah Htm 9,25Jt 6786687687	DICARI DibeliTinggi:SOFA, TV,L.Es,M.Cuci,KmrSet-7066655767
Cpt Supra X 125 '06 CW Mulus Jl Cihampelas 230C 5768768768	S280'05 Hitam Km.30rb CBU MobilLand-K.gdg 78786876876
C200 97 Jklt silver istw +E260 Elg 03 km 40rb Hitam Yessuri Jl.Cideng Barat 35 Ph 67686868	Avanza E VVTi'07 Silver a/n.Sndr,Bagus. Hub.08873893202
	HIKMAT STEEL,Canopy DenganBahan Berkualitas 942527277/0842424243

INFORMATION RETRIEVAL

D e f i n i s i

ISO 2382/1 mendefinisikan IR sebagai tindakan, metode dan prosedur demi menemukan kembali data yang tersimpan untuk menyediakan informasi mengenai subyek yang dibutuhkan. Tidakan tersebut mencakup text indexing, inquiry analysis, dan relevance analysis; data mencakup text, tabel, gambar, ucapan, dan video; sedangkan informasi termasuk pengetahuan relevan yang dibutuhkan untuk mendukung penyelesaian masalah dan akuisisi pengetahuan [8]. Tujuan dari sistem IR adalah memenuhi kebutuhan informasi pengguna dengan me-*retrieve*

semua dokumen yang mungkin relevan, pada waktu yang sama me-*retrieve* sesedikit mungkin dokumen yang tak-relevan. Sistem ini menggunakan fungsi heuristik untuk mendapatkan dokumen-dokumen yang relevan dengan query pengguna. Sistem IR yang baik memungkinkan pengguna menentukan secara cepat dan akurat apakah isi dari dokumen yang diterima memenuhi kebutuhannya. Agar representasi dokumen lebih baik, dokumen-dokumen dengan topik atau isi yang mirip dikelompokkan bersama-sama [9].



Gambar 2 Arsitektur dasar dari sistem IR

Arsitektur Sistem IR

Secara garis besar arsitektur sistem IR diperlihatkan pada gambar 2 [8]. Ada dua pekerjaan yang ditangani oleh sistem ini, yaitu melakukan *pre-processing* terhadap database dan kemudian menerapkan metode tertentu untuk menghitung kedekatan (relevansi atau *similarity*) antara dokumen di dalam database yang telah di*preprocess* dengan query pengguna. Pada tahapan *preprocessing*, sistem yang berurusan dengan dokumen *semi-structured* biasanya memberikan *tag* tertentu

pada term-term atau bagian dari dokumen; sedangkan pada dokumen tidak terstruktur proses ini dilewati dan membiarkan term tanpa imbuhan *tag*. Query yang dimasukkan pengguna dikonversi sesuai aturan tertentu untuk mengekstrak term-term penting yang konsisten dengan term-term yang sebelumnya telah diekstrak dari dokumen dan menghitung relevansi antara query dan dokumen berdasarkan pada term-term tersebut. Hasilnya, sistem mengembalikan suatu daftar dokumen terurut *descending* (ranking) sesuai nilai kemiripannya dengan query pengguna.

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. Kata 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.

Pembuatan Index

Pembangunan *index* dari koleksi dokumen merupakan tugas pokok pada tahapan preprocessing di dalam IR. Kualitas *index* mempengaruhi efektifitas dan efisiensi sistem IR [10]. 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 mengakibatkan hasil buruk dan mungkin dapat kehilangan beberapa item yang relevan. Index yang besar memungkinkan *retrieval* banyak dokumen bermanfaat sekaligus dapat menaikkan jumlah dokumen yang tidak relevan dan juga dapat menurunkan kecepatan pencarian (*searching*) [11].

Pembuatan *inverted index* harus melibatkan konsep *linguistic processing* yang bertujuan mengekstrak term-term penting dari dokumen yang direpresentasikan sebagai *bag-of-words*. Ekstraksi term biasanya melibatkan dua operasi utama berikut [8]:

1. Penghapusan ***stop-words***. Stop word didefinisikan sebagai term yang tidak berhubungan (*irrelevant*) dengan subyek utama dari database meskipun kata

tersebut sering kali hadir di dalam dokumen. Contoh *stop words* adalah a, an, the, this, that, these, those, her, his, its, my, our, their, your, all, few, many, several, some, every, for, and, nor, bit, or, yet, so, also, after, although, if, unless, because, on, beneath, over, of, during, beside, dan etc.

Stop-words termasuk pula beberapa kata yang didefinisikan tertentu yang terkait dengan subyek database, misal pada database yang menampung daftar paper penelitian terkait dengan heart *diseases*, maka kata *heart* dan *disease* sebaiknya dihapus.

2. ***Stemming***. Kata-kata yang muncul di dalam dokumen sering mempunyai banyak varian morfologik. Karena itu, setiap kata yang bukan *stop-words* direduksi ke *stemmed word* (*term*) yang cocok yaitu kata tersebut distem untuk mendapatkan bentuk akarnya dengan menghilangkan awalan atau akhiran. Dengan cara ini, diperoleh kelompok kata yang cocok dimana kata-kata di dalam kelompok tersebut merupakan varian sintaktis dari satu sama lain dan dapat menghimpun hanya satu kata per kelompok. Sebagai contoh, kata *disease*, *diseases*, *diseased* berbagi-pakai term stem umum *disease*, dan dapat diperlakukan sebagai bentuk lain dari kata ini.

   Algoritma *stemming* paling umum untuk bahasa Inggris dan dinyatakan efektif adalah algoritma Porter [12]. Algoritma ini terdiri dari lima fase reduksi kata yang diterapkan secara urut. Di dalam setiap fase terdapat berbagai konvensi untuk memilih aturan seperti memilih aturan dari setiap grup aturan yang menerapkan terhadap akhiran paling panjang. Di dalam fase pertama, konvensi tersebut digunakan mengikuti aturan grup berikut:

Rule		**Contoh**	
SSES	$\rightarrow$ SS	caresses	$\rightarrow$ caress
IES	$\rightarrow$ I	ponies	$\rightarrow$ poni
SS	$\rightarrow$ SS	caress	$\rightarrow$ caress
S	$\rightarrow$	cats	$\rightarrow$ cat

   Banyak aturan-aturan berikutnya menggunakan konsep ukuran kata, dengan memeriksa jumlah dari suku kata untuk mengetahui apakah suatu kata cukup

panjang sehingga beralasan untuk menganggap bagian yang cocok dari aturan sebagai suatu akhiran bukan sebagai bagian dari akar kata. Sebagai contoh, aturan:

(m>1) EMENT →

memetakan "*replacement*" menjadi "*replac*" tetapi "*cement*" tidak menjadi "*c*".

Menurut [13, 14] terdapat 5 langkah pembangunan *inverted index*, yaitu:

1. Penghapusan format dan *markup* dari dalam dokumen
   Tahap ini menghapus semua tag *markup* dan format khusus dari dokumen, terutama pada dokumen yang mempunyai banyak tag dan format seperti dokumen (X)HTML. Jika isi dokumen telah berada di dalam database maka tahapan ini sering ditiadakan.

2. Pemisahan rangkaian term (*tokenization*)
   Tokenization adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman 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 (*lower case*).

3. 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, setidaknya karena dua hal. Pertama, jumlah dokumen yang relevan terhadap suatu query kemungkinan besar merupakan bagian kecil dari koleksi. Term yang efektif dalam pemisahan dokumen yang relevan dari tidak relevan kemungkinan besar adalah term yang muncul pada sedikit dokumen. Ini berarti bahwa *term* dengan frekuensi tinggi merupakan *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 *stop-word* dan dihapus [14].

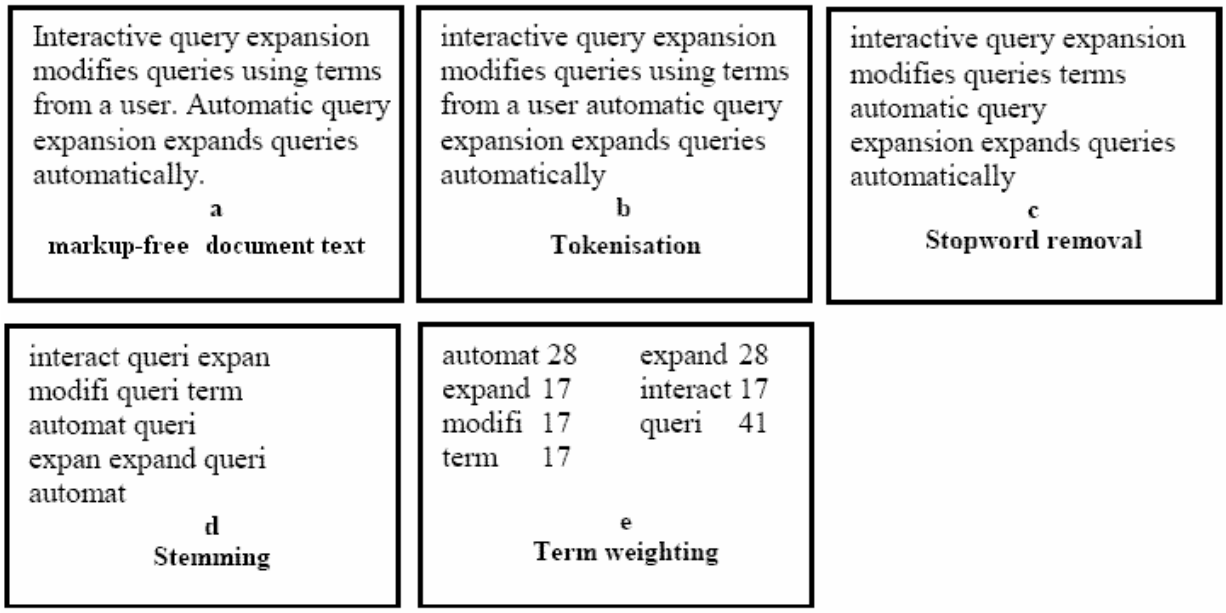
Pengahapusan *stop-word* dalam suatu koleksi dokumen pada satu waktu akan cukup memakan waktu. Solusinya adalah dengan menyusun suatu pustaka *stop-word* atau *stop-list* dari *term* yang akan dihapus. Strategi umum dalam menentukan *stop-list* adalah dengan mengurutkan term berdasarkan frekuensi koleksi (jumlah total kemunculan setiap term di dalam koleksi dokumen), dan kemudian memasukkan term yang paling sering muncul sebagai *stop-word* [13].

4. Pengembalian term ke bentuk akar kata (*stemming*)

   Selain itu, 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. Menyerupai stemming, operasi ini bertujuan menemukan suatu kelompok kata-kata terkait. Perbedaan utama adalah bahwa sinonim tidak berbagi-pakai term stem tetapi dijumpai berdasarkan pada thesaurus. Jika pengguna memasukkan query "*heart disease*" maka query diekspansi untuk mengakomodasi semua sinonim dari *disease* seperti *ailment*, *complication*, *condition*, *disorder*, *fever*, *ill*, *illness*, *infirmity*, *malady*, *sickness*, dan lain-lain [8].

5. Pemberian bobot terhadap term (*weighting*).

   Setiap term diberikan bobot sesuai dengan model pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya (dijelaskan pada bagian lain). Banyak aplikasi menerapkan pembobotan kombinasi berupa perkalian bobot lokal *term frequency* dan global *inverse document frequency*, ditulis *tf .idf*.



Gambar 3 Contoh lima tahap *indexing* pada sistem berbasis *content* secara urut mulai dari *markup removal* (a), *tokenization* (b), *stopwords filtration* (c), *stemming* (d) dan *weighting* (e).

Tabel 1 *Contoh struktur index dalam bentuk tabel*

Term	**Posisi (Nomor dokumen)**
automat	(1,2);(3,5);(4,3); dst s.d ke-28
expan	(1,3);(6,5);(10,13); dst s.d ke-28
expand	(7,2);(8,5);(9,1); dst s.d ke-17
interact	(21,7);(30,5);(34,2); dst s.d ke-17
modify	(12,1);(13,5);(14,12); dst s.d ke-17
query	(9,8);(13,5);(14,1); dst s.d ke-41
term	(6,2);(22,5);(23,3); dst s.d ke-17

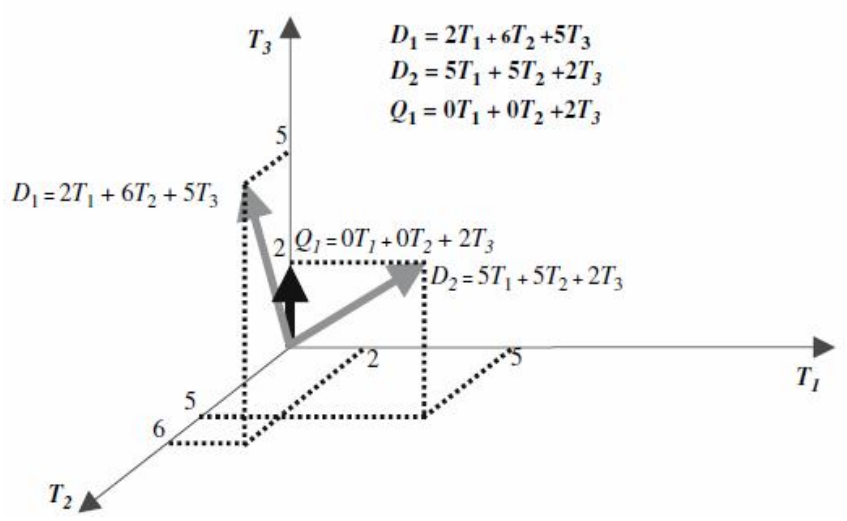
Gambar 3 memberikan ilustrasi dari proses *indexing* [15] sedangkan contoh struktur index diperlihatkan pada tabel II.1 [16, 17].

Model Ruang Vektor

Pada sistem IR, kemiripan antara dokumen teks didefinisikan berdasarkan representasi ***bag-of-words*** dan dikonversi ke suatu model ruang vektor (*vector space model*, VSM). Model ini diperkenalkan oleh Salton [18] dan telah digunakan secara luas. Pada VSM setiap dokumen di dalam database dan query pengguna direpresentasikan oleh suatu vektor multi-dimensi [8, 19]. Dimensi sesuai dengan jumlah term dalam database. Pada model ini:

- *Vocabulary* merupakan kumpulan semua term berbeda yang tersisa setelah *preprocessing* dari dokumen di dalam database, dan mengandung t index terms. Term-term "orthogonal" ini membentuk suatu ruang vektor.
- Setiap term i, di dalam dokumen atau query j, diberikan suatu bobot (*weight*) bernilai *real* w_{ij}.
- Dokumen dan query diekspresikan sebagai vektor t dimensi $d_j = (w_1, w_2, ..., w_{tj})$. Dianggap terdapat n dokumen di dalam database, yaitu j = 1, 2, ..., n.

Contoh dari model ruang vektor tiga dimensi untuk dua dokumen D_1 dan D_2, satu query pengguna Q_1, dan tiga term T_1, T_2 dan T_3 diperlihatkan pada gambar 4.



Gambar 4. Contoh model ruang vektor dengan dua dokumen D_1 dan D_2, serta query Q_1

Dalam model ruang vektor, suatu database dari semua dokumen direpresentasikan oleh matriks *term-document* (atau matriks *term-frequency*). Setiap sel dalam

matriks bersesuaian dengan bobot yang diberikan yang diberikan dari suatu term dalam dokumen yang ditentukan. Nilai nol berarti bahwa term tersebut tidak hadir di dalam dokumen. Gambar 5 mempertegas penjelasan ini [8].

$$\begin{bmatrix} & T_1 & T_2 & \cdots & T_t \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} \end{bmatrix}$$

Gambar 5. Contoh matriks *term-document* untuk database dengan n dokumen dan t term.

Keberhasilan dari model VSM ini ditentukan oleh skema pembobotan terhadap suatu term baik untuk cakupan lokal maupun global, dan faktor normalisasi [19]. Pembobotan lokal hanya berpatokan pada frekuensi munculnya term dalam suatu dokumen dan tidak melihat frekuensi kemunculan term tersebut di dalam dokumen lainnya. Pendekatan dalam pembobotan lokal yang paling banyak diaplikasikan adalah *term frequency* (*tf*) meskipun terdapat skema lain seperti pembobotan biner, *augmented normalized tf*, logaritmik *tf* dan logaritmik alternatif.

Pembobotan global digunakan untuk memberikan tekanan terhadap *term* yang mengakibatkan perbedaan dan berdasarkan pada penyebaran dari *term* tertentu di seluruh dokumen. Banyak skema didasarkan pada pemikiran bahwa semakin jarang suatu *term* muncul di dalam keseluruhan koleksi maka *term* tersebut menjadi semakin berbeda. Pemanfaatan pembobotan ini dapat menghilangkan kebutuhan *stop word removal* karena *stop word* mempunyai bobot global yang sangat kecil. Namun pada prakteknya akan lebih baik menghilangkan *stop word* di dalam fase *pre-processing* sehingga semakin sedikit *term* yang harus ditangani. Pendekatan yang digunakan dalam pembobotan global mencakup *inverse document frequency* (*idf*), *squared idf*, *probabilistic idf*, *GF-idf*, *entropy*. Pendekatan *idf* merupakan pembobotan yang paling banyak digunakan saat ini. Beberapa aplikasi tidak melibatkan bobot global, hanya memperhatikan *tf*, yaitu

ketika *tf* sangat kecil atau saat diperlukan penekanan terhadap frekuensi *term* di dalam suatu dokumen. [19].

Faktor normalisasi digunakan untuk menormalkan vektor dokumen sehingga proses *retrieval* tidak terpengaruh oleh panjang dari dokumen. Normalisasi demikian diperlukan karena dokumen panjang biasanya mengandung *term* yang sama secara berulang sehingga menaikkan frekuensi term (*tf*). Dokumen panjang juga mengandung banyak *term* yang berbeda sehingga menaikkan ukuran kemiripan antara query dengan dokumen tersebut, meningkatkan peluang di-*retrieve*nya dokumen yang lebih panjang. Beberapa pendekatan normalisasi adalah normalisasi cosinus, penjumlahan bobot, normalisasi ke-4, normalisasi bobot maksimal, normalisasi *pivoted unique*.

Bobot lokal suatu term i di dalam dokumen j (tf_{ij}) dapat didefinisikan sebagai:

$$tfi_j = \frac{f_{ij}}{\max_i(f_{ij})}$$

Dimana f_{ij} adalah jumlah berapa kali term i muncul di dalam dokumen j. Frekuensi tersebut dinormalisasi dengan frekuensi dari ***most common term*** di dalam dokumen tersebut.

Bobot global dari suatu term i pada pendekatan *inverse document frequency* (idf_i) dapat didefinisikan sebagai

$$idf_i = \log_2(\frac{n}{df_i})$$

Dimana df_i adalah frekuensi dokumen dari term i dan sama dengan jumlah dokumen yang mengandung term i. Log_2 digunakan untuk memperkecil pengaruhnya relatif terhadap tf_{ij}.

Bobot dari term i di dalam sistem IR (w_{ij}) dihitung menggunakan ukuran tf-idf yang didefinisikan sebagai berikut [8, 20]:

$$w_{ij} = tf_{ij} \times idf_i$$

Bobot tertinggi diberikan kepada term yang muncul sering kali dalam dokumen j tetapi jarang dalam dokumen lain. Gambar 6 memperlihatkan proses perhitungan bobot *tf-idf* bagi suatu dokumen yang menyertakan kalimat "data cube contains x data dimension, y data dimension, and z data dimension".

"data cube contains x data dimension, y data dimension, and z data dimension"

(teks bergarisbawah diabaikan sebagai akibat lingustic preprocessing)

- Term frequency dari term yang tersisa setelah preprocessing:

  Data (4), dimension (3), cube (1), contain (1)

- Jika dianggap total terdapat 10000 dokumen dan frekuensi dokumen yang mengandung 4 term di atas adalah

  data (1300), dimension (250), cube (50) dan contain (3100)

maka bobot untuk setiap term berdasarkan pendekatan tf.idf adalah

data: tf = 4/4 idf = log2(10000/1300) = 2.94 tf.idf = 2.94

dimension: tf= 1/4 idf = log2(10000/250) = 5.32 tf.idf = 3.99

cube: tf = 1/4 idf = log2(10000/50) = 7.64 tf.idf = 1.99

contain: tf = 1/4 idf = log2(10000/3100) = 1.69 tf.idf = 0.42

Gambar 6. Contoh perhitungan bobot *tf-idf*

II.3.5 Ukuran Kemiripan

Model ruang vektor dan pembobotan *tf-idf* digunakan untuk merepresentasikan nilai numerik dokumen sehingga kemudian dapat dihitung kedekatan antar dokumen. Semakin dekat dua vektor di dalam suatu VSM maka semakin mirip dua dokumen yang diwakili oleh vektor tersebut. Kemiripan antar dokumen dihitung menggunakan suatu fungsi ukuran kemiripan (*similarity measure*). Ukuran ini memungkinkan perankingan dokumen sesuai dengan kemiripan (relevansi)nya terhadap query. Setelah dokumen diranking, sejumlah tetap dokumen *top-scoring* dikembalikan kepada pengguna. Alternatifnya, suatu *threshold* dapat digunakan untuk memutuskan berapa banyak dokumen akan dikembalikan. *Threshold* dapat digunakan untuk mengontrol tarik-ulur antara presisi dan *recall*. Nilai *threshold* tinggi biasanya akan menghasilkan presisi tinggi dan *recall* rendah.

Ukuran kemiripan teks cukup populer [21] adalah *cosine similarity*. Ukuran ini menghitung nilai cosinus sudut antara dua vektor. Diberikan vektor yang

merepresentasikan dokumen d_j dan query q, dan t term diekstrak dari database, nilai cosinus antara d_j dan q didefinisikan sebagai [8]:

$$similarity\ (\vec{d}_j, \vec{q}) = \frac{\vec{d}_j . \vec{q}}{|\vec{d}_j| . |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

Contoh:

Jika dua dokumen D1 = 2T1 + 6T2 + 5T3 dan D2 = 5T1 + 5T2 + 2T3 dan query Q1 = 0T1 + 0T2 + 2T3 sebagaimana diperlihatkan pada gambar 3, berikut ini adalah nilai *cosine* yang diperoleh:

$$similarity(\vec{D}_1, \vec{Q}_1) = \frac{(2.0 + 6.0 + 5.2)}{\sqrt{(4 + 36 + 25).(0 + 0 + 4)}} = \frac{10}{\sqrt{65.4}} = 0.62$$

$$similarity(\vec{D}_2, \vec{Q}_1) = \frac{(5.0 + 5.0 + 2.2)}{\sqrt{(25 + 25 + 4).(0 + 0 + 4)}} = \frac{4}{\sqrt{54.4}} = 0.27$$

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

Terdapat beberapa variasi dari kemiripan cosinus terkait dengan pembobotan terhadap *term* seperti menghilangkan *tf*, *idf*, atau keduanya. Lee [20] menyarankan untuk mengikutsertakan *tf* dan *idf* dalam menghitung kemiripan antar dokumen. Menurutnya, meninggalkan salah satu *tf* atau *idf* akan memberikan *ranking* yang buruk. Guo [22] mengusulkan agar memberikan bobot khusus untuk term tertentu pada kondisi tertentu dan mengubah perhitungan bobot menjadi:

$$w_{ij} = w_i \times tf_{ij} \times idf_i$$

Selain ukuran kemiripan cosinus, beberapa ukuran kemiripan lain yang dapat digunakan dalam ruang vektor adalah Dice, Jaccard dan Overlap [23].

Dice: $$similarity\ (\vec{d}_j, \vec{q}) = \frac{2 \times \sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sum_{i=1}^{t} w_{ij}^2 + \sum_{i=1}^{t} w_{iq}^2}$$

Jaccard: $$similarity(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sum_{i=1}^{t} w_{ij}^2 + \sum_{i=1}^{t} w_{iq}^2 - \sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}$$

Overlap: $$similarity(\vec{d}_j, \vec{q}) = \frac{2 \times \sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\min\left(\sum_{i=1}^{t} w_{ij}^2, \sum_{i=1}^{t} w_{iq}^2 \right)}$$

Meskipun banyak diaplikasikan karena sederhana dan cepat, VSM dengan *tf.idf* mempunyai beberapa kekurangan. Pendekatan *tf.idf* tidak mampu mencerminkan kemiripan kata, hanya menghitung jumlah kata yang *overlap* dan mengabaikan informasi sinonim dan sintaktis. *tf.idf* menggunakan kata-kunci untuk menemukan dokumen yang mengandung kata yang cocok dengan query. Ini dapat memberikan banyak dokumen yang tidak relevan dan mungkin beberapa dokumen yang relavan hilang karena menggunakan kata berbeda untuk mengekspresikan ketertarikan yang sama. Setiap penambahan dan penghapusan terhadap koleksi dokumen juga mengharuskan perubahan nilai *idf* sehingga sistem harus sering melakukan *update* [9].

Murad [9] mengusulkan algoritma untuk menghitung kesamaaan kata berdasarkan himpunan fuzzy dari kata. Algoritma ini fokus pada penggunaaan kata, bukan artinya. Kata-kata di dalam dokumen dianggap serupa jika kata-kata tersebut tampil dalam konteks yang sama. Ada dua fase perhitungan kemiripan yang dilakukan setelah *preprocessing* terhadap dokumen, yaitu perhitungan distribusi frekuensi pada himpunan fuzzy dan perhitungan probabilitas kata.

Atlam [35] mengusulkan ukuran perhitungan bernama P-SIM (*passage similarity*). Pada pendekatan ini dilakukan pengambilan bagian (*passage*) terpenting dari suatu dokumen berdasarkan keberadaan artikel “the”. Kemiripan antara query dan dokumen adalah tingkat kemiripan antara query dengan *passage* yang diperoleh dari dokumen

panjang pada suatu ruang vektor. Keuntungan pendekatan ini adalah dimensi vektor dari dokumen yang lebih kecil sehingga proses komputasi menjadi lebih cepat. Kekurangannya, ini hanya dapat diterapkan pada teks bahasa inggris.

Kemiripan pada Teks Pendek

Ukuran kemiripan teks *standard*, yaitu kemiripan cosinus di dalam ruang vektor seperti di atas dapat memberikan hasil tidak memuaskan jika diterapkan pada teks sangat pendek, seperti pada perhitungan kemiripan antar query. Sebagai contoh, teks "UAE" dan "United Arab Emirates" sebenarnya mengacu pada maksud sama tetapi dianggap tidak mirip (jauh) pada pendekatan standard. Masalah lain adalah terkait *context* dari teks. Teks pendek "apple pie" dan "Apple computer" mengandung term "apple" tetapi berada pada topik berbeda. Perhitungan kemiripan standard mengatakan bahwa dua teks pendek ini sangat mirip [24].

Beberapa pendekatan diusulkan untuk mengatasi masalah ketidakcocokan *vocabulary* mencakup *stemming* [12], LSI, model-model translasi dan ekspansi query. Model translasi bertujuan mengukur kemiripan antara dokumen atau kalimat yang merupakan translasi atau tranformasi dari query. Model demikian dianggap kurang efektif pada teks sangat pendek karena kesulitan memperkirakan kemungkinan translasi yang *reliable* bagi suatu teks. Teknik yang umum digunakan adalah ekspansi query. Teknik ini mengkonversi suatu query awal ke representasi yang lebih kaya informasi. Ini disusun dengan menambahkan *term* yang kemungkinan besar hadir dalam dokumen yang relevan dengan representasi query asli.

Sahami [25] mengusulkan metode *web-based similarity kernel* untuk mengekspasi teks pendek menggunakan hasil pencarian web. Kemiripan antara dua teks pendek dapat dihitung dalam ruang representasi yang diekspansi. Meek [26] mengusulkan ukuran kemiripan *web-relevance* yang tetap memanfaatkan *search engine* untuk mendapatkan n top dokumen web tetapi hanya mengambil judul dan rangkumannya untuk membangun dokumen baru. Kemiripan antar dokumen dihitung menggunakan pendekatan *tf.idf* dalam ruang vektor sebagaimana dibahas di atas. Quan [27] mengusulkan metode baru dalam perhitungan kemiripan antar

teks pendek dengan membandingkan topik probabilitasnya. Metode ini dimulai dengan menghimpun term-term pembeda antara dua teks pendek dan membandingkan keduanya dengan serangkaian topik yang diekstrak dengan algoritma *Gibbs sampling*. Kedekatan antara term pembeda ditentukan oleh probabilitasnya di dalam setiap topik. Kemiripan antara dua teks dihitung berdasarkan pada common term dan kedekatan dari term-term pembedanya.

Paragraf di atas memperlihatkan bahwa representasi teks merupakan bagian penting dari suatu ukuran kemiripan, dan ini berlaku pada teks panjang maupun pendek. Ada tiga reprensentasi teks yang umum digunakan [24], yaitu:

1. Representasi *surface*. Teks tetap sebagaimana aslinya, tidak dilakukan modifikasi sama sekali dan kemungkinan mengandung *noice*.

2. Representasi hasil *stemming*. Teks pendek yang asli digeneralisasi atau dinormalisasi, terutama untuk mengatasi masalah ketidakcocokan *vocabulary*. *Stemming* masih mungkin meninggalkan *noise*, misalnya "*marine vegetation*" dan "*marinated vegetables*" akan diubah menjadi "marin veget" jika menggunakan *Porter Stemmer*.

3. Representasi hasil ekspansi. Representasi ini mencoba menyelesaikan masalah ketidak-cocokan *context*, misalnya kata "bank" di dalam "*Bank of America*" dan "*river bank*". Biasanya teks direpresentasi memanfaatkan sumber eksternal dari informasi yang terkait dengan *term* dalam query. Sumber yang sering digunakan sebagai acuan ekspansi adalah hasil query dari suatu *search engine*, Wikipedia dan catatan reformulasi query.

Kualitas Text Retrieval

Sistem IR mengembalikan sekumpulan dokumen sebagai jawaban dari query pengguna. Terdapat dua kategori dokumen yang dihasilkan oleh sistem IR terkait pemrosesan query, yaitu *relevent documents* (dokumen yang relevan dengan query) dan *retrieved documents* (dokumen yang diterima pengguna). Kombinasi dua kategori ini memberikan 4 kemungkinan jawaban dimana 2 jawaban salah dan 2 jawaban benar, yaitu [8]:

1. Mengembalikan dokumen yang tidak relevan atau tidak mengembalikan dokumen yang relevan. Ini merupakan jawaban yang salah.
2. Mengembalikan dokumen yang relevan atau tidak mengembalikan dokumen yang tidak relevan. Ini adalah jawaban yang benar.

Ukuran umum yang digunakan untuk mengukur kualitas dari *text retrieval* adalah kombinasi *precision* dan *recall*. Presisi mengevaluasi kemampuan sistem IR untuk menemukan kembali dokumen *top-ranked* yang paling relevan, dan didefinisikan sebagai persentase dokumen yang diretrieve yang benar-benar relevan terhadap query pengguna.

$$presisi = \frac{X}{retrieved\_documents}$$

Recall mengevaluasi kemampuan sistem IR untuk menemukan semua item yang relevan dalam database dan didefinisikan sebagai persentase dokumen yang relevan terhadap query pengguna dan yang diterima.

$$recall = \frac{X}{relevant\_documents}$$

Terdapat tarik-ulur antara presisi dan recall sehingga dapat terjadi dua situasi eksterm berikut:

- Presisi terlalu tinggi dan *recall* rendah. Sistem mengembalikan beberapa dokumen dan hampir semuanya relevan, tetapi sejumlah besar dokumen relevan lain terabaikan.
- *Recall* sangat tinggi dan presisi relatif rendah. Sistem mengembalikan sejumlah besar dokumen yang mengikutsertakan hampir semua dokumen relevan tetapi juga mencakup sebagian besar dokumen yang tak diharapkan.

Peningkatan Kualitas

Rancangan dasar dari sistem IR dapat ditingkatkan untuk menaikkan presisi dan *recall* serta memperbaiki matriks *term-document*. Isu pertama sering diselesaikan menggunakan mekanisme *relevance feedback*, sedangkan yang terakhir dilakukan menggunakan *latent semantic indexing*.

Latent Semantic Indexing

Latent semantic indexing bertujuan meningkatkan efektifitas dari sistem IR dengan me-*retrieve* dokumen-dokumen yang lebih relevan terhadap query pengguna dengan memanipulasi matriks *term-document*. Matriks asli biasanya sangat besar bagi sumberdaya komputasi yang tersedia. Terdapat pula kemungkinan *noisy* misalnya beberapa term yang bersifat anekdot atau terlalu jauh dari topik dokumen. Karena itu dibuat sebuah matriks baru yang lebih kecil dan bersih dari term-term yang tidak diperlukan [8].

Matriks baru dihitung menggunakan teknik *singular value decomposition* (SVD). Diberikan matriks *term-document* t x n yang merepresentasikan t term dan n dokumen, SVD menghilangkan beberapa baris dan kolom dari matriks asli sehingga ukurannya menjadi k x k, dimana k biasanya cukup kecil, misalnya ratusan (pada dokumen umum), sedangkan pada matriks awal t dan n dapat bernilai ribuan. SVD bertujuan menghapus bagian *least significant* dari matriks asli dengan langkah-langkah berikut:

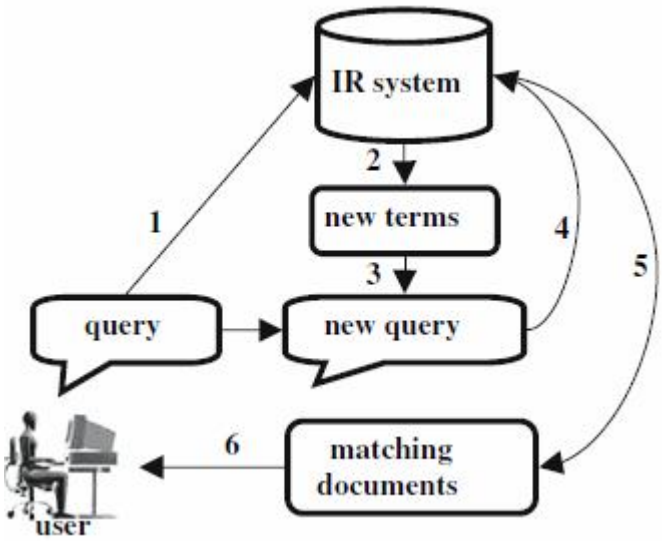
- Buat suatu matriks *term-document.*
- Hitung SVD dari matriks *term-document* tersebut dengan membaginya ke dalam tiga matriks yang lebih kecil U, S, dan V dimana U dan V bersifat *orthogonal*, yaitu $U^T U = 1$, dan S adalah suatu matriks *diagonal singular value*. Matriks terakhir berukuran k x k dan merupakan versi tereduksi dari matriks *term-document* asli.
- Bagi setiap dokumen, modifikasi vektor *term-document*-nya berdasarkan matriks baru telah menghapus term tidak perlu.
- Simpan semua vektor baru dan buat suatu index multidimensi untuk menfasilitasi prosedur pencarian.

Relevance Feedback

Relevance feedback bertujuan meningkatkan relevansi dari dokumen yang dikembalikan dengan mengubah query pengguna. Beberapa term ditambahkan ke dalam query awal agar dapat menemukan dokumen yang lebih relevan. Proses

dari sistem *relevance feedback* diperlihatkan pada gambar 7 dengan langkah-langkah sebagai berikut:

1. Kerjakan suatu pencarian IR pada query awal
2. Terima *feedback* dari pengguna, dokumen mana saja yang relevan dan temukan term-term baru (sesuai query) dari dokumen relevan yang diketahui.
3. Tambahkan term-term baru ke query awal untuk membentuk query baru
4. Ulangi pencarian menggunakan query baru tersebut
5. Kembalikan sekumpulan dokumen relevan berdasarkan query baru
6. Pengguna mengevaluasi dokumen yang dikembalikan sistem IR



Gambar 7. Relevance feedback pada Sistem IR

Relevance feedback (langkah 2) dapat dikerjakan secara manual maupun otomatis. Pada pendekatan pertama, pengguna secara manual mengidentifikasi dokumen yang relevan dan term baru dipilih secara manual atau otomatis. Pada pendekatan kedua, dokumen relevan diidentifikasi secara otomatis dengan menggunakan dokumen *top-ranked*, dan term-term baru dipilih secara otomatis memanfaatkan langkah-langkah berikut:

1. Temukan N dokumen *top-ranked*
2. Identifikasi semua term dari N dokumen *top-ranked*

3. Pilih term-term *feedback*, misalnya berdasarkan pemilihan beberapa top term yang sesuai dengan nilai maksimal dari bobot *tf-idf* dan tidak mengikutkan term dari query awal pengguna.

Relevance feedback akan meningkatkan presisi dengan menaikkan jumlah dari "good" term dalam query tetapi pada waktu yang sama membutuhkan lebih banyak komputasi dan kadang dapat menurunkan efektifitas terutama saat terjadi penambahan "bad" term yang dapat merusak apa yang telah dihasilkan oleh query "good" term [8].

KLASIFIKASI TEKS

Klasifikasi atau kategorisasi teks adalah proses penempatan suatu dokumen ke suatu kategori atau kelas sesuai dengan karakteristik dari dokumen tersebut. Dalam *text mining*, klasifikasi mengacu kepada aktifitas menganalisis atau mempelajari himpunan dokumen teks *pre-classified* untuk memperoleh suatu model atau fungsi yang dapat digunakan untuk mengelompokkan dokumen teks lain yang belum diketahui kelasnya ke dalam satu atau lebih kelas kelas *pre-defined* tersebut [28, 29, 30].

Dokumen yang digunakan untuk pembelajaran dinamakan contoh (*sample* atau *training data set*) yang dideskripsikan oleh himpunan atribut atau variabel. Salah satu atribut mendeskripsikan kelas yang diikuti oleh suatu contoh, hingga disebut atribut kelas. Atribut lain dinamakan atribut independen atau *predictor*. Klasifikasi termasuk pembelajaran jenis *supervised learning*. Jenis lain adalah *unsupervised learning* atau dikenal sebagai *clustering*. Pada *supervised learning*, data latihan mengandung pasangan data input (biasanya vektor) dan output yang diharapkan, sedangkan pada *unsupervised learning* belum terdapat target output yang harus diperoleh.

Proses klasifikasi teks dapat dibagi ke dalam dua fase, yaitu [31]:

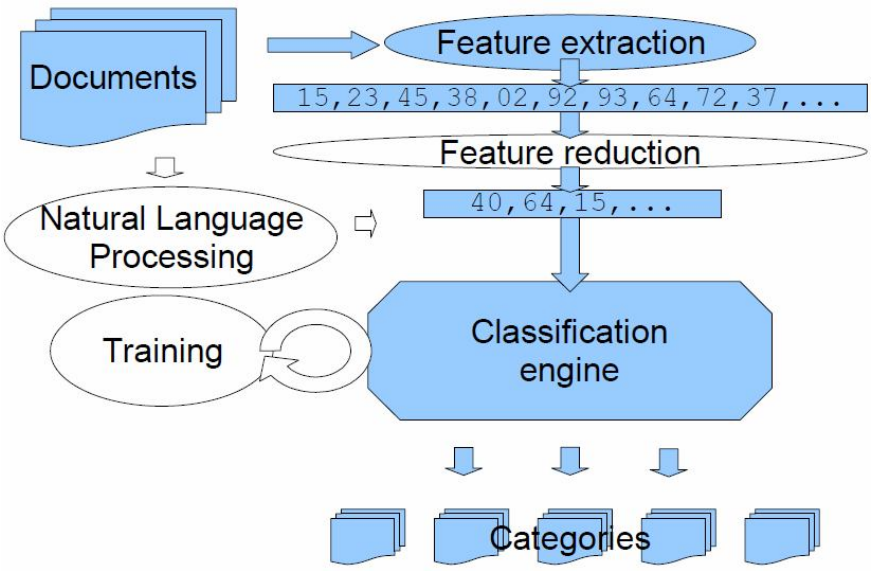
1. Fase *information retrieval* (IR) untuk mendapatkan data numerik dari dokumen teks. Langkah pertama yang dilakukan pada fase ini adalah feature extraction. Pendekatan yang umum digunakan adalah distribusi frekuensi kata.

Nilai numerik yang diperoleh dapat berupa berapa kali suatu kata muncul di dalam dokumen, 1 jika kata ada di dalam dokumen atau 0 jika tidak ada (biner), atau jumlah kemunculan kata pada awal dokumen. Feature yang diperoleh dapat direduksi agar dimensi vektor menjadi lebih kecil. Beberapa pendakatan *feature reduction* dapat diterapkan seperti menghapus *stop-words*, *stemming*, *statistical filtering*. Teknik lebih lanjut seperti SVD dan genetic algoritm akan menghasilkan vektor berdimensi lebih rendah.

2. Fase klasifikasi utama ketika suatu algoritma memroses data numerik tersebut untuk memutuskan ke kategori mana teks ditempatkan. Terdapat beberapa algoritma klasifikasi yang merupakan kajian di bidang statistika dan *machine learning* yang dapat diterapkan pada fase ini, di antaranya adalah *Naive Bayesian*, *Rocchio*, *Decision Tree*, k-*Nearest Neighbor*, *Neural Network*, dan *Support Vector Machines*. Teknik-teknik tersebut berbeda dalam mekanisme pembelajaran dan representasi model yang dipelajari [29].

Menurut Sebastiani [28] proses klasifikasi teks melibatkan banyak teknik IR mulai dari preprocessing, pengukuran kemiripan selama operasional klasifikasi sampai dengan evaluasi kinerja algoritma.

Gambar 8 memperlihatkan diagram proses klasifikasi teks secara garis besar [32].



Gambar 8 Diagram proses klasifikasi teks

Asirvatham [34] mengusulkan suatu metode klasifikasi otomatis dari halaman web ke dalam beberapa kategori berdasarkan pada struktur dari dokumen dan gambar yang terdapat di dalamnya. Informasi struktur dokumen dapat berupa rasio jumlah link dengan teks normal, adanya logo pada bagian atas halaman, bar navigasi, jumlah teks yang banyak, kehadiran persamaan dan grafik.

Chenometh [30] merangkum perbandingan antara 5 algoritma yang sering digunakan dalam kategorisasi teks dan hasilnya diperlihatkan pada tabel 2.

Tabel 2 Perbandingan algoritma klasifikasi teks

Classfier	**Metode**	**Kinerja**
Naive Bayes	Menghitung probabilitas dari suatu dokumen untuk ikut ke suatu kategori berdasarkan pada kehadiran dari kata yang sama di dalam dokumen lain yang telah ada di dalam kategori tersebut.	Lebih rendah daripada model lain
Metode *Rocchio*	Membandingkan dokumen terhadap suatu daftar term positif dan negatif bagi setiap katagori dan mengklasifi sesuai dengan kehadiran atau bobot dari term-term tersebut.	Rendah, terutama saat mengkasifikasi ke dalam kategori dengan banyak term representatif
k-Nearest Neighbor	Mencari sebanyak k dokumen paling mirip dan menempatkan dokumen ke kategori dimana k dokumen tersebut ditempatkan sebelumnya	Baik, terutama dengan penempatan banyak kategori, tetapi lambat karena setiap dokumen harus dibandingkan ke semua dokumen lain
Decision Tree	Memisahkan dokumen-dokumen secara hirarki di dalam struktur pohon, dimana setiap node merupakan term yang relevan dan ujung setiap cabang adalah kategori.	Baik tetapi memerlukan optimisasi untuk menyelesaikan *overfitting*.
Support Vector Machines	Menggambar antara term yang berkontribusi dan tidak terhadap suatu dokumen yang akan ditempatkan ke suatu kategori tertentu. Kategori didasarkan pada kehadiran dari term yang berkontribusi.	Terbaik meskipun sangat mudah terjadi error dalam data training.

REFERENSI

[1] Manaf, Afwarman, Catur, M. Zuhri (2006) *NoteBOX with a Unified Messaging System*. International Telecommunicatios Network Strategy and Planning Symposium, New Delhi. November 2006.

[2] Kurniawan, Robbi (2008), *Peningkatan Performansi NoteBOX Unified Messaging System*, Tugas Akhir S1 Teknik Informatika ITB, Bandung

[3] Indah, Mia Nur, (2008) Desain dan Implementasi Sistem NOteBOX Berbasis Unified Messaging: Subsistem Search Server, Tesis Magister Informatika ITB, Bandung

[4] Sondakh, Debby E. (2008) Pengembangan Model dan Struktur Informasi untuk Konten Berbasis Teks pada Sistem NoteBox, Tesis Magister Informatika ITB, Bandung

[5] Advertising. http://en.wikipedia.org/wiki/Advertising, 22 Desember 2009

[6] Classified Ad Definition. http://dictionary.cambridge.org/topic.asp?key =classifiedad, 22 Desember 2009

[7] Classified Advertising. http://en.wikipedia.org/wiki/Advertising-advertising, 22 Desember 2009

[8] Cios, Krzysztof J. Etc. (2007) Data Mining A Knowledge Discovery Approach, Springer

[9] Murad, Azmi MA., Martin, Trevor. (2007) Word Similarity for Document Gouping using Soft Computing. *IJCSNS International Journal of Computer Science and Network Security*, Vol.7 No.8, August 2007, pp. 20- 27

[10] Chu W. Liu Z, Mao W. (2002) Textual Document Indexing and Retrieval via Knowledge Sources and Data Mining.

[11] Hyusein, Byurhan, Patel, Ahmad (2003) Web Document Indexing and Retrieval, LNCS 2588 pp. 573-579, Springer Verlag Berlin

[12] Porter, Martin (1980) An algorithm for suffix stripping, http://tartarus.org/~martin/PorterStemmer/def.txt, Akses 03 Juni 2009

[13] Manning, Christopher D, Ragnavan Prabhakar, Schutze, Hinrich (2008) Introduction to Information Retrieval, Cambridge University Press

[14] Document Indexing Tutorial for Information Retrieval Students and Search Engine Marketers, http://www.miislita.com/information-retrieval-tutorial/indexing.html, 10 November 2008

[15] Ruthven I., Lalmas M. (2003) A survey on the use of relevance feedback for information access systems, *Knowledge Engineering Review, 18(1):2003*, http://inex.is.informatik.uni-duisburg.de:2004/pdf/ker_ruthven_lalmas.pdf

[16] Ristov, Strahil (2003) Using Inverted Files to Compress Text

[17] Moffat A. Bell, Witten IH (1999) Managing Gigabytes, 2nd Edition, Morgan Kaufmann

[18] Salton, Gerard (1983) Introduction to Modern Information Retrieval, McGraw Hill

[19] Polettini, Nicola (2004) The Vector Space Model in Information Retrieval – Term Weighting Problem

[20] Lee D.L. (1997). Document Ranking and the Vector-Space Model. *IEEE March-April 1997*

[21] Tata, Sandeep, Patel M, Jignesh (2007) Estimating he Selectivity of tf-idf based Cosine Similarity Predicates, Sigmod Record December 2007 Vol 36 No. 4

[22] Guo,Qinglin(2008) The Similarity Computing of Documents Based on VSM, Springer Verlag Berlin

[23] Dunham, Margareth H (2003) Data Mining Introductory and Advanced Topics, New Jersey: Prentice Hall

[24] Metzler, Donald, Dumais, Susan, Meek, Christopher (2007) Similarity Measures for Short Segments of Text, *ECIR 2007 LNCS 4425, pp 16-27*, Springer Verlag Berlin.

[25] Sahami, M, Heilman, T (2006) A web-based kernel function for measuring the similarity of short text snippets, In Proceedings of WWW 2006, pages 377-386.

[26] Meek, Christopher, Yih, Wen-tau (2007) Improving Similarity Measures for Short Segments of Text

[27] Quan, Xiaojun. Etc (2009) Short text similarity based on probabilistic topics. Knowledge Information Systems Regular Paper, 17 September 2009

[28] Sebastiani, Fabrizio (2002) Machine Learning in Automated Text Categorization. ACM Computing Survey, Vol.34 No.1, March 2002, pp. 1-47

[29] An, Aijun (2009) Classification Methods, dalam Encyclopedia of Data Warehouse & Data Mining, IGI Global, hal. 196 – 201

[30] Chenometh, Megan, Song, Min (2009) Text Categorization, dalam Encyclopedia of Data Warehouse & Data Mining, IGI Global, hal. 1936-1941

[31] Mahinovs, Aigars, TiwariText, Ashutosh (2007) Classification Method Review. *Decision Engineering Report Series*, Cranfield University

[32] Pant, G., Srinivasan, P. (2005) Learning to Crawl: Comparing Classification Schemes. ACM Transactions on Information Systems, 23, pp. 430 -462.

[33] Klensin , J. (2001) RFC-2821 - Simple Mail Transfer Protocol, http://www.ietf.org/rfc/rfc2821.txt, 10 November 2009

[34] Asirvatham, Arul Prakash, Ravi, Kranthi Kumar (200x) Web Page Categorization Based on Document Structure

[35] Atlam, Elsayed (2008) A New Approach For Text Similarity Using Articles, International Journal Of Information Technology & Decision Making, Vol.7 No. 1 (2008) pp. 23–34