BAB II

LANDASAN TEORI

2.1 Information Retrieval

2.1.1 Definisi

Information Retrieval System atau Sistem Temu Balik Informasi merupakan bagian dari *computer science* tentang pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri. Menurut Gerald J. Kowalski [19] di dalam bukunya "*Information Storage and Retrieval Systems Theory and Implementation*", sistem temu balik informasi adalah suatu sistem yang mampu melakukan penyimpanan, pencarian, dan pemeliharaan informasi. Informasi dalam konteks ini dapat terdiri dari teks (termasuk data numerik dan tanggal), gambar, audio, video, dan objek multimedia lainnya.

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 tidak relevan.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 [2].

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 [3] :

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

2.1.2 Arsitektur *Information Retrieval System*

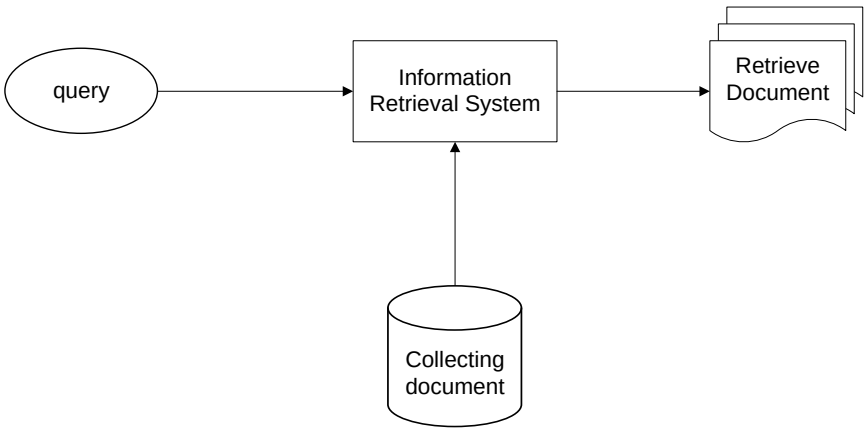
Ada dua pekerjaan yang ditangani oleh sistem ini, yaitu melakukan *pre-processing* terhadap *database* dan kemudian menerapkan metode tertentu untuk menghitung kedekatan (relevansi atau *similarity*) antara dokumen di dalam *database* yang telah di*preprocess* dengan *query* pengguna.

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 sejalan dengan *term-term* yang sebelumnya telah diekstrak dari dokumen dan menghitung relevansi antara *query* dan dokumen berdasarkan pada *term-term* tersebut. Sebagai hasilnya, sistem mengembalikan suatu daftar dokumen terurutsesuai nilai kemiripannya dengan *query* pengguna [1].

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.

Proses dalam *Information Retrieval*dapat digambarkan sebagai sebuah proses untuk mendapatkan *retrieve document* dari *collection documents* yang ada melalui pencarian *query* yang diinputkan user.



Gambar 2.1Proses dalam *Information Retrieval System*

Proses yang terjadi di dalam *Information Retrieval System* terdiri dari 2 bagian utama, yaitu *Indexing subsystem* dan *Searching subsystem (matching system).*

2.1.3 Proses Indexing

Indexing subsystem adalah proses *subsystem* yang 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 [4]. 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*) [5].

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 [1]:

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. Berikut ini adalah Contoh *stop words*dalam bahasa inggris :*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*. Contoh *stop words* dalam bahasa Indonesia : yang, juga, dari, dia, kami, kamu, aku, saya, ini, itu, atau, dan, tersebut, pada, dengan, adalah, yaitu, ke, tak, tidak, di, pada, jika, maka, ada, pun, lain, saja, hanya, namun, seperti, kemudian, dll. *Stop-words* termasuk pula beberapa kata tertentu yang didefinisikan terkait dengan topik *database*, misal pada *database* yang menampung daftar karya tulis

(*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 bentuk *stemmed word* (*term*) yang cocok. Kata tersebut distem untuk mendapatkan bentuk akarnya dengan menghilangkan awalan atau akhiran. Dengan cara ini, diperoleh kelompok kata yang mempunyai makna serupa tetapi berbeda wujud sintaktis satu dengan lainnya. Kelompok tersebut dapat direpresentasikan oleh satu kata tertentu. Sebagai contoh, kata menyebutkan, tersebut, disebut dapat dikatakan serupa atau satu kelompok dan dapat diwakili oleh satu kata umum sebut.

Menurut [6, 7] terdapat 5 langkah pembangunan *inverted index*, yaitu:

a. **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.

b. Pemisahan rangkaian kata (*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*).

c. 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 dokumen tidak relevan kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. 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 [6].

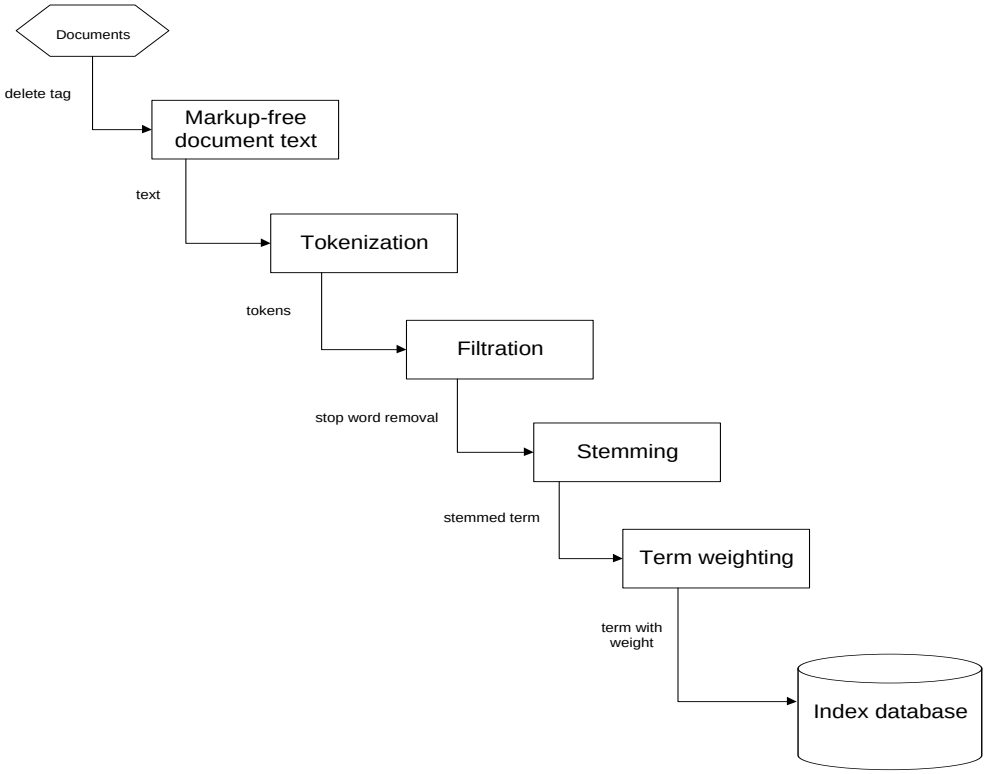
d. Konversi *term* ke bentuk dasar (*stemming*).

Stemming adalah proses konversi *term* ke bentuk umumnya, sebagaimana dijelaskan sebelumnya. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi term-term tertentu di dalamnya. Sinonim adalah kata-kata yang mempunyai pengertian serupa tetapi berbeda dari sudut pandang morfologis. Seperti *stemming*, operasi ini bertujuan menemukan suatu kelompok kata terkait. Akan tetapi sinonim bekerja berdasarkan pada

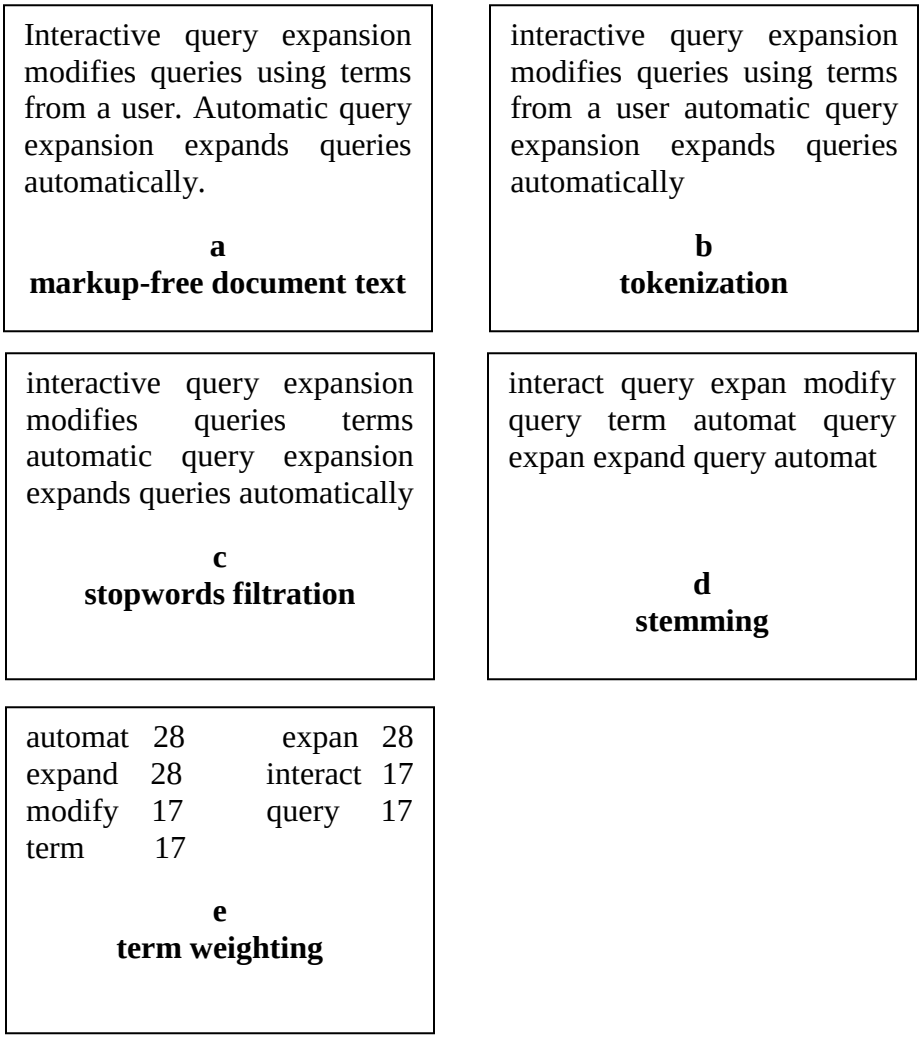
thesaurus, tidak berbagi-pakai term stem. Jika pengguna memasukkan *query* "*heart disease*" maka *query* diekspansi untuk mengakomodasi semua sinonim dari *disease* seperti *ailment, complication, condition, disorder, fever, ill, illness, infirmity, malady, sickness*, dan lain-lain [1].

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

Setiap *term* diberikan bobot sesuai dengan skema pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya. Banyak aplikasi menerapkan pembobotan kombinasi berupa perkalian bobot lokal *term frequency* dan *global inverse document frequency*, ditulis tf .idf.



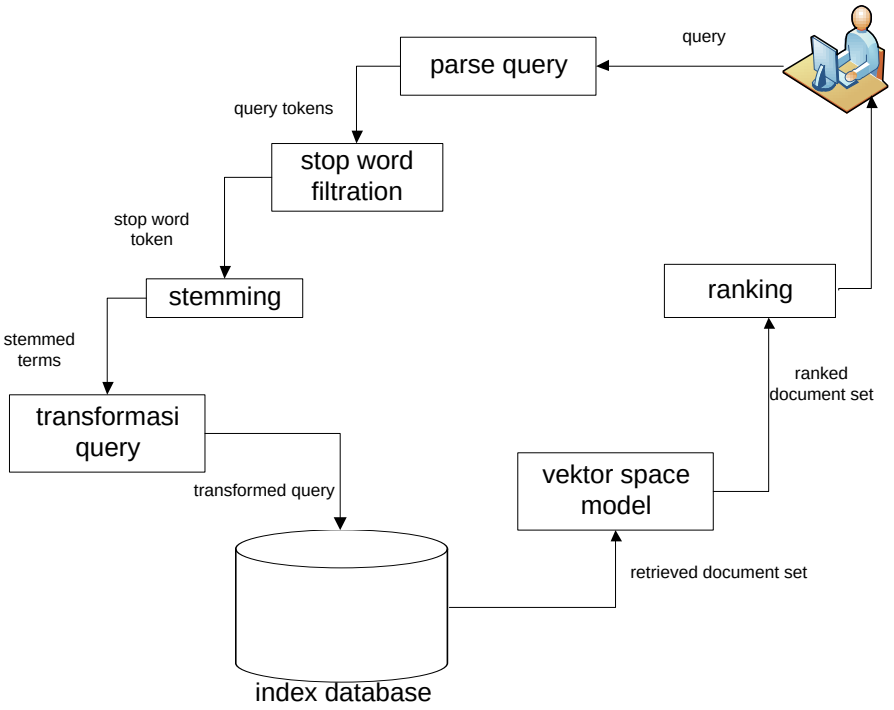
Gambar 2.2Proses *Indexing*



Gambar 2.3 Contoh lima tahap *indexing* berbasis content secara urut mulai dari*markup removal* (a), *tokenization* (b), *stopword filtration* (c), *stemming* (d), *weighting*(e) [8]

2.1.4 Proses *Searching*

Dibawah ini adalah gamabar ilustrasi proses pencarian dalam *Information Retrieval System.*



Gambar 2.4 Proses *Searching*

Beberapa proses yang terjadi saat melakukan *search* sesuai dengan ilustrasi gambar 2.4 yaitu :

1. *Parse query* yaitu memecah query menjadi bentuk token
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*

   Stopword tokens dari proses *stopword* sebelumnya kemudian di filter kembali melalui proses *Stemming* sehingga menghasilkan *stemmed term query.*

4. Transformasi *Query*

   Stemmed term query yang dihasilkan kemudian ditransformasikan apabila memerlukan. Artinya, apabila *query* yang diinputkan membutuhkan terjemahan ke dalam bentuk *query* bahasa lain maka sebelum mencari dokumen pada koleksi dokumen, *query* tersebut diterjemahkan duhulu melalui proses penerjemahan *query.* Sistem akan membandingkan *query* tersebut dengan koleksi dokumen sehingga mengembalikan dokumen-dokumen yang relevan dalam suatu bahasa yang berbeda dengan bahasa *query*.

5. Pemodelan dalam model ruang vektor

   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.

6. Perangkingan dokumen atau konten berdasarkan nilai kemiripan antara *query* dan dokumen.

2.2 Model Ruang Vektor (*Vector Space Model*)

2.2.1 Definisi

Vektor Space Model adalah model sistem temu balik informasi yang mengibaratkan masing-masing *query* dan dokumen sebagai sebuah vektor n-dimensi [10]. 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.

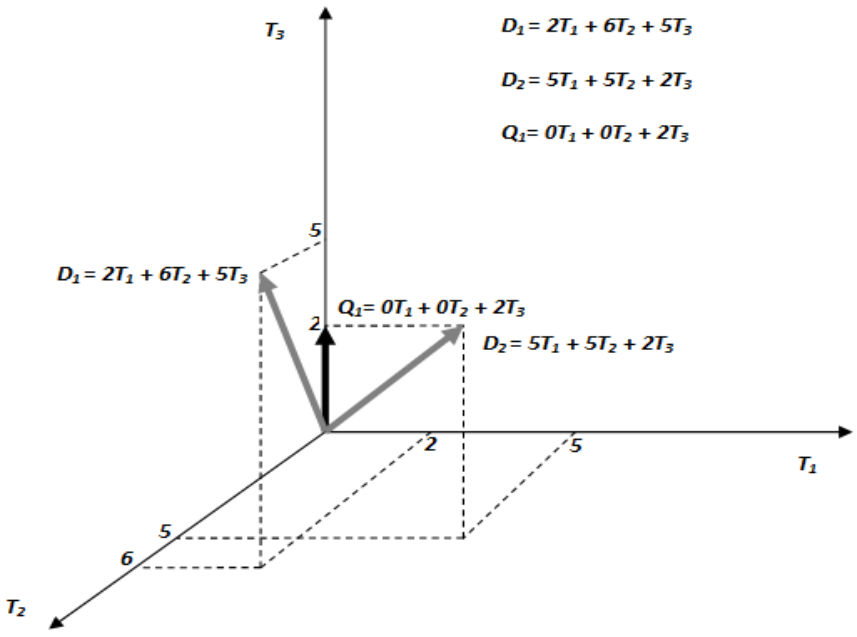
2.2.2 Model Ruang Vektor Dalam Information Retrieval

Pada *Information Retrieval System* terdapat beberapa metode yang digunakan dalam *Searching* salah satunya adalah dengan merepresentasikan proses *Searching* menggunakan Model Ruang Vektor. Model ruang vektor dibuat berdasarkan pemikiran bahwa isi dari dokumen ditentukan oleh kata-kata yang digunakan dalam dokumen tersebut. 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.

Kemiripan antar dokumen didefinisikan berdasarkan representasi *bag-of-words* dan dikonversi ke suatu model ruang vektor (*vector space model*, VSM). Model ini diperkenalkan oleh Salton [12] dan telah digunakan secara luas. Pada VSM, setiap dokumen di dalam database dan query pengguna direpresentasikan oleh suatu vektor multi-dimensi [1, 13]. Dimensi sesuai dengan jumlah term dalam dokumen yang terlibat Pada model ini:

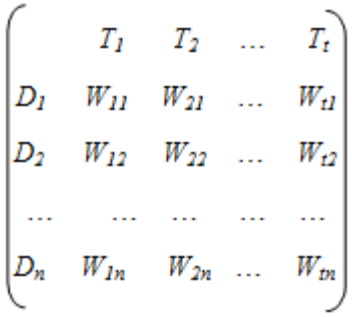
a. *Vocabulary* merupakan kumpulan semua *term* berbeda yang tersisa dari dokumen setelah *preprocessing* dan mengandung *t term index*. *Term-term* ini membentuk suatu ruang vektor.
b. Setiap *term i* di dalam dokumen atau *query j*, diberikan suatu bobot (*weight*) bernilai *real* W*ij*.
c. Dokumen dan *query* diekspresikan sebagai vektor *t* dimensi $d_j = (W_1, W_2, ..., Wtj)$ dan terdapat *n* dokumen di dalam koleksi, 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 2.5.



Gambar 2.5 Contoh Model Ruang Vektor dengan dua dokumun D_1 dan D_2, serta *query* Q_1. [1]

Dalam model ruang vektor, koleksi dokumen direpresentasikan oleh matriks *term-document* (atau matriks *term-frequency*). Setiap sel dalam matriks bersesuaian dengan bobot yang diberikan dari suatu term dalam dokumen yang ditentukan. Nilai nol berarti bahwa *term* tersebut tidak hadir di dalam dokumen. [1]



Gambar 2.6 Contoh matriks *term-document* untuk *database* dengan *n* document dan *t term*. [1]

2.2.3 Pembobotan Kata (*Term Weighting)*

Pembobotan kata sangat berpengaruh dalam menentukan kemiripan antara dokumen dengan *query*. Apabila bobot tiap kata dapat ditentukan dengan tepat, diharapkan hasil perhitungan kemiripan teks akan menghasilkan perangkingan dokumen yang baik.

Keberhasilan dari model VSM ini ditentukan oleh skema pembobotan terhadap suatu *term* baik untuk cakupan lokal maupun global, dan faktor normalisasi [13]. Pembobotan lokal hanya berpedoman pada frekuensi munculnya *term* dalam suatu dokumen dan tidak melihat kemunculan *term* tersebut di dalam dokumen lainnya.

Faktor yang memegang peranan penting dalam pembobotan kata, yaitu :

a. *Term Frequency (tf)*

Pendekatan dalam pembobotan lokal yang paling banyak diterapkan adalah *term frequency (tf)*.Faktor ini menyatakan banyaknya kemunculan suatu kata dalam suatu dokumen.Semakin sering suatu kata muncul dalam sebuah dokumen, berarti semakin penting kata tersebut.

Ada empat cara yang bias digunakan untuk mendapatkan nilai TF (Ramadhany, 2008; Karhendana, 2008):

1. *Raw* Tf

   Nilai Tf sebuah *term* dihitung berdasarkan kemunculan *term* tersebut dalam dokumen.

2. *Logarithmic* Tf

   Dalam memperoleh nilai Tf, cara ini menggunakan fungsi logaritmik dalam matematika.

   $Tf = 1 + \log(Tf)$ (2.1)

3. *Binary* Tf

   Cara ini, akan menghasilkan nilai boolean berdasarkan kemunculan term pada dokumen tersebut. Akan bernilai 0 apabila term tidak ada pada sebuah dokumen, dan bernilai 1 apabila term tersebut ada dalam

dokumen. Sehingga banyaknya kemunculan term pada sebuah dokumen tidak berpengaruh.

4. Augmented Tf

$$Tf = 0.5 + 0.5 \times \frac{Tf}{\max(Tf)} \quad (2.2)$$

- Nilai *Tf* adalah jumlah kemunculan *term* pada sebuah dokumen
- Nilai *max(Tf)* adalah jumlah kemunculan terbanyak *term* pada dokumen yang sama.

Perhitungan Tf yang akan digunakan dalam implementasi *Information Retrieval System* ini adalah Raw Tf.

b. *Inverse Document Frequency* (*idf*)

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 pertimbangan bahwa semakin jarang suatu *term* muncul di dalam total koleksi maka *term* tersebut menjadi semakin berbeda. Pemanfaatan pembobotan ini dapat menghilangkan kebutuhan *stopwordremoval* karena *stopword* mempunyai bobot global yang sangat kecil. Namun pada prakteknya lebih baik menghilangkan *stopword* di dalam fase *pre-processing* sehingga semakin sedikit *term* yang harus ditangani. Pendekatan terhadap

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. [13].

Bobot global dari suatu term *i* pada pendekatan *inverse document frequency* (*idfi*) dapat dirumuskan sebagai berikut :

$$Idf_i = log_2 (\frac{N}{d_{ji}}) \quad (2.3)$$

Dimana N menyatakan jumlah artikel dalam koleksi dokumen, d_{ji} 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 *term i* di dalam *Information Retrieval System* (W_{ij}) dihitung menggunakan *tf-idf* yang didefinisikan sebagai berikut. [1,15]

$$W_{ij} = tf_{ij} \times idf_i \quad (2.4)$$

c. Normalisasi panjang dokumen

Dokumen-dokumen yang panjang sering dianggap lebih relevan dibandingkan dokumen yang pendek, padahal belum tentu demikian.Untuk mengurangi pengaruh perbedaan panjang dokumen ini, pada pembobotan kata digunakan

satu faktor lagi yang disebut sebagai normalisasi panjang dokumen.Normalisasi yang digunakan adalah normalisasi kosinus. Berdasarkan [14] rumus normalisasi kosinus yaitu :

$$Similarity(D,Q) = \cos\theta = \frac{\sum_{i=1}^{n}(WD_i \times WQ_i)}{\sqrt{\sum_{i=1}^{n}(WD_i)^2 \times \sum_{i=1}^{n}(WQ_i)^2}} \quad (2.5)$$

Dengan W adalah bobot dari *query* dan dokumen.

2.2.4 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 relevansinya terhadap *query*. Setelah dokumen diranking, sejumlah tetap dokumen *top-scoring* dikembalikan kepada pengguna.

Cosine Similarity tidak hanya digunakan untuk menghitung normalisasi panjang dokumen tapi juga menjadi salah satu ukuran kemiripan yang popular[16]. Ukuran ini menghitung nilai kosinus sudut antara dua vektor. Jika terdapat dua vektor dokumen d_j dan*query* q, serat *t term* diekstrak dari koleksi dokumen maka nilai kosinus antara d_j dan q didefinisikan sebagai [1] :

$$Similarity(\overrightarrow{d_j},\vec{q}) = \frac{\overrightarrow{d_j}.\vec{q}}{|\overrightarrow{d_j}|.|\vec{q}|} = \frac{\sum_{i=1}^{t}(W_{ij} \times W_{iq})}{\sqrt{\sum_{i=1}^{t}(W_{ij})^2 \times \sum_{i=1}^{t}(W_{iq})^2}} \quad (2.6)$$

Contoh:

Jika dua dokumen D1 = 2T1 + 6T2 + 5T3 dan D2 = 5T1 + 5T2 + 2T3 dan query Q1 = 0T1 + 0T2 + 2T3 sebagaimana diperlihatkan pada gambar 2.5, berikut ini adalah nilai kosinus yang diperoleh:

$$Similarity(\overrightarrow{D_1},\overrightarrow{Q_1}) = \cos\theta = \frac{(2.0 + 6.0 + 5.2)}{\sqrt{(4 + 36 + 25).(0 + 0 + 4)}} = \frac{10}{\sqrt{65.4}} = 0.62$$

$$Similarity(\overrightarrow{D_2},\overrightarrow{Q_1}) = \cos\theta = \frac{(5.0 + 5.0 + 2.2)}{\sqrt{(25 + 25 + 4).(0 + 0 + 4)}} = \frac{4}{\sqrt{54.4}} = 0.27$$

Contoh di atas memperlihatkan bahwa sesuai dengan perhitungan kosinus, dokumen D2lebih mirip dengan *query* daripada dokumen D1. Terlihat sudut antara D2 dan Q1 lebih kecil daripada sudut antara D1 dan Q1.

Terdapat beberapa variasi dari kemiripan kosinus terkait dengan pembobotan terhadap *term* seperti menghilangkan *tf*, *idf*, atau keduanya.Lee [15] menyarankan untuk mengikutsertakan *tf* dan *idf* dalam menghitung kemiripan antar dokumen. Menurutnya, meninggalkan salah satu *tf* atau *idf* akan memberikan ranking yang buruk. Guo [17] mengusulkan agar memberikan bobot khusus untuk *term*tertentu pada kondisi tertentu dan mengubah perhitungan bobot menjadi:

$$W_{ij} = W_i \times tf_{ij} \times idf_i \quad (2.7)$$

Selain ukuran kemiripan kosinus, beberapa ukuran kemiripan lain yang dapat digunakan dalam ruang vektor adalah Dice, Jaccard dan Overlap [18].

Dice : $Similarity(\overrightarrow{d_j},\vec{q}) = \frac{2 \times \sum_{i=1}^{t}(W_{ij} \times W_{iq})}{\sum_{i=1}^{t}(W_{ij})^2 + \sum_{i=1}^{t}(W_{iq})^2}$ (2.8)

Jaccard : $Similarity(\overrightarrow{d_j},\vec{q}) = \frac{2 \times \sum_{i=1}^{t}(W_{ij} \times W_{iq})}{\sum_{i=1}^{t}(W_{ij})^2 + \sum_{i=1}^{t}(W_{iq})^2 - \sum_{i=1}^{t}(W_{ij} . W_{iq})}$ (2.9)

Overlap : $Similarity(\overrightarrow{d_j},\vec{q}) = \frac{2 \times \sum_{i=1}^{t}(W_{ij} \times W_{iq})}{\min(\sum_{i=1}^{t}(W_{ij})^2, \sum_{i=1}^{t}(W_{iq})^2)}$ (2.10)

2.3 *Software* Pendukung

2.3.1 MySQL

Pada awalnya, *MySQL* merupakan proyek internal sebuah firma asal Swedia, *TcXDataKonsult*.MySQL kemudian dirilis untuk publik pada tahun 1996. Karena *MySQL* menjadi sangat populer, pada tahun 2001 firma tersebut mendirikan sebuah perusahaan baru, *MySQLAB*, yang khusus menawarkan layanan dan produk berbasis *MySQL* (Gilmore, 2006).

Dari awal pembuatannya, para pengembang *MySQL* menitikberatkan pengembangan *MySQL* pada sisi performa dan skalabilitasnya. Hasilnya adalah sebuah perangkat lunak yang sangat teroptimasi, walaupun dari sisi fitur memiliki kekurangan dibandingkan solusi basis data kelas enterprise lain. Akan tetapi *MySQL* menarik minat banyak pengguna. Saat ini, tercatat lebih dari lima juta basis data *MySQL* yang terpasang dan aktif di seluruh dunia. Beberapa perusahaan dan instansi

penting dunia seperti *Yahoo!*, *Google* dan *NASA* menggunakan *MySQL* untuk mengolah basis data mereka.

Ada beberapa kelebihan yang dimiliki *MySQL* sehingga dapat menarik banyak pengguna. Kelebihan tersebut yaitu:

1. Fleksibilitas

   Saat ini, *MySQL* telah dioptimasi untuk duabelas platform seperti *HP-UX*, *Linux*, *Mac OS X*, *Novell Netware*, *OpenBSD, Solaris, Microsoft Windows* dan lain-lain. *MySQL* juga menyediakan *source code* yang dapat diunduh secara gratis, sehingga pengguna dapat mengkompilasi sendiri sesuai platform yang digunakan. Selain itu, *MySQL* juga dapat dikustomisasi sesuai keinginan penggunanya, misalnya mengganti bahasa yang digunakan pada antarmukanya.

2. Performa

   Sejak rilis pertama, pengembang *MySQL* fokus kepada performa. Hal ini masih tetap dipertahankan hingga sekarang dengan terus meningkatkan fiturnya.

3. Lisensi

   MySQL menawarkan berbagai pilihan lisensi kepada penggunanya. Lisensi *open source* yang ditawarkan yaitu lisensi *GNU General Public License* dan *Free/Libre and Open Source Software (FLOSS) License Exception*. Selain itu

ditawarkan juga lisensi komersil berbayar yang memiliki fasilitas dukungan teknis.

2.3.2 HTML (Hipertext Markup Language)

HTML adalah sebuah bahasa *markup* yang digunakan untuk membuat sebuah halaman web dan menampilkan berbagai informasi di dalam sebuah browser Internet.*HTML* diciptakan oleh Tim Berners-Lee, seorang peneliti CERN.Berners-Lee mendasarkan HTML pada *Standard Generalized Markup Language*. Dokumen HTML pada dasarnya adalah dokumen teks yang mengandung kode-kode tag yang sesuai dengan spesifikasi HTML. Kode-kode tag itu nantinya diterjemahkan oleh aplikasi browser sehingga dokumen HTML tadi bisa ditampilkan sesuai dengan yang diinginkan pembuatnya. Secara umum, HTML memiliki empat jenis elemen yaitu:

1. *Structural*, yaitu tanda yang menentukan level atau tingkatan sebuah teks (misalnya sebagai *heading*, paragraf, kutipan, dan sebagainya).
2. *Presentational*, yaitu tanda yang menentukan tampilan sebuah teks (misalnya cetak tebal, miring, garis bawah, dan lain-lain).
3. *Hypertext*, yaitu tanda yang menunjukkan *link* ke bagian lain pada teks tersebut atau ke dokumen lain.
4. *Widget*, yaitu tanda yang menghasilkan obyek-obyek tertentu seperti tombol, garis horisontal, dan lain-lain.

2.3.3 PHP

PHP berawal dari skrip *Perl/CGI* yang dibuat oleh seorang pengembang perangkat lunak bernama Rasmus Lerdorf untuk menghitung jumlah pengunjung *homepage*-nya. Karena banyaknya pengunjung yang meminta skrip tersebut, Lerdorf akhirnya membagi-bagikan skrip buatannya yang diberi nama *Personal Home Page (PHP)*.

Banyaknya permintaan membuat Lerdorf terus mengembangkan skripnya. Beberapa orang akhirnya bergabung membentuk tim untuk mengembangkan PHP. Sejak itu PHP berkembang pesat dengan banyak fungsi baru yang ditambahkan. Kepanjangan dari PHP kini berubah menjadi *PHP: Hypertext Preprocessor*.

Ada tiga macam penggunaan PHP:

1. *Server-side scripting*. Ini merupakan jenis penggunaan yang paling banyak dilakukan pengguna PHP. Untuk menggunakannya, dibutuhkan tiga hal: PHP *parser*, aplikasi *web server* yang terkoneksi dengan instalasi PHP, dan aplikasi *web browser*.
2. *Command line scripting*. Pada penggunaan PHP jenis ini hanya dibutuhkan PHP *parser*.
3. Pembuatan aplikasi berbasis *desktop*. Pada penggunaan PHP jenis ini, dibutuhkan ekstensi tambahan *PHP-GTK*.

PHP memiliki empat kelebihan utama yang menarik minat banyak pengguna. Kelebihan utama PHP tersebut diringkas dalam 4P berikut:

1. *Practicality*. PHP dibuat dengan menitikberatkan pada kepraktisan. Hasilnya, PHP adalah bahasa pemrograman minimalis, dilihat dari segi kebutuhan pengguna dan kebutuhan sintaks.
2. *Power*. PHP memiliki banyak kemampuan, mulai dari kemampuan untuk terhubung dengan basis data, membuat halaman *web* dinamis, membuat dan memanipulasi berkas gambar, *Flash* dan *PDF*, berkomunikasi dengan bermacam protokol seperti *IMAP* dan *POP3*, dan masih banyak lagi.
3. *Possibility*. PHP dapat menyediakan lebih dari satu solusi untuk suatu masalah
4. *Price*. PHP selalu dirilis kepada publik tanpa ada batasan untuk penggunaan, modifikasi, atau redistribusi.

2.3.4 Cascading Style Sheets (CSS)

CSS adalah sebuah bahasa *style sheet* (lembar gaya) yang digunakan untuk mengatur tampilan dokumen yang ditulis dalam bahasa *markup*. CSS Level 1 (CSS1) diperkenalkan pada tahun 1995 oleh *World Wide Web Consortium* (W3C) dan dimaksudkan untuk mengatasi keterbatasan HTML dalam hal keleluasaan pengaturan desain dan tampilan pada sebuah dokumen HTML.Setahun kemudian CSS1 diberi status rekomendasi penuh oleh W3C yang juga mengatur spesifikasi CSS. Saat ini ada tiga level CSS, yaitu CSS Level 1 (*Recommendation*), CSS Level 2 (Recommendation), dan CSS Level 2 Revision 1 (Candidate Recommendation).

Penggunaan CSS paling banyak untuk memformat halaman *web* yang ditulis dengan HTML dan XHTML. Walau demikian, CSS dapat dipergunakan untuk bahasa *markup* lain seperti SVG dan XUL.

2.3.5 Apache HTTP Server

Apache HTTP Server atau yang biasa disebut Apache, merupakan sebuah aplikasi *web server* yang dibuat oleh Robert McCool.Apache kini dikembangkan dibawah *Apache Software Foundation* dan tersedia untuk berbagai sistem operasi seperti Linux, UNIX, MS Windows, Mac OS X dan lain-lain. Sejak tahun 1996 Apache menjadi aplikasi *web server* paling populer, dan pada tahun 2009 menjadi aplikasi *web server* pertama yang digunakan oleh lebih dari 100 juta situs *web*.

2.3.6 Adobe Dreamweaver CS5

Dreamweaver merupakan sebuah aplikasi untuk merancang pembuatan *website*.*Dreamweaver* dibuat oleh perusahaan *Macromedia* sehingga dinamakan *Macromedia Dreamweaver*.Sejak *Macromedia* diakuisisi *Adobe Inc.*, namanya berubah menjadi *Adobe Dreamweaver*.Versi pertama *Dreamweaver* dibawah Adobe adalah CS5, mengikuti versi rilisnya yang dipaketkan dalam *Adobe Creative Suite 5*.

Dreamweaver memiliki kelebihan dalam hal kemudahan penggunaan. Untuk pengguna awam, *Dreamweaver* menyediakan fungsi tampilan *Design*, sehingga pengguna dapat merancang tampilan halaman web dengan konsep *WYSIWYG (What*

You See Is What You Get). Untuk pengguna tingkat lanjut, *Dreamweaver* menyediakan tampilan *Code* sehingga pengguna dapat merancang tampilan yang lebih lengkap menggunakan kode. Pengguna juga dimudahkan dengan berbagai fasilitas yang dimiliki *Dreamweaver* seperti *tag auto-completion*untuk penulisan kode *HTML*. Format yang didukung *Dreamweaver* juga cukup lengkap, mulai dari *HTML*, *JavaScript, CSS,* sampai *XML*.

2.4 Pemrograman Berorientasi Objek

Pemograman berorientasi objek *(Obhect oriented programming – OOP)* merupakan paradigma pemograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek.Bandingkan dengan logika pemrograman terstruktur.Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar.Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

2.4.1 Konsep Pemrograman Berorientasi Objek

Pemrograman berorientasi objek menekankan konsep berikut:

1. kelas, kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'class of dog'* adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi oobjek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang *non-programmer* sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
2. Objek, membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstraksi , kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat

melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. Enkapsulasi, memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
5. *Polimorfisme* melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut *polimorfisme* karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang

berbeda dalam pemanggilan yang sama. Hal ini berlawanan denganbahasa fungsional yang mencapai *polimorfisme* melalui penggunaan fungsi kelas-pertama.

6. Inheritas mengatur *polimorfisme* dan *enkapsulasi* dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas.)
7. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

2.4.2 *Unified Modelling Language (UML)*

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak.

Unified Modeling Language (UML) adalah himpunan struktur danteknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat *(tool)* desain berorientasi objek dari IBM

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson.Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

2.4.2.1 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence Diagram* untuk memodelkan pengiriman pesan *(message)* antar objek.
4. *Collaboration Diagram* untuk memodelkan interaksi antar objek.
5. *State Diagram* untuk memodelkan perilaku objek di dalam sistem.
6. *Activity Diagram* untuk memodelkan perilaku *user*dan objek di dalam sistem.
7. *Class Diagram* untuk memodelkan struktur kelas.
8. *Objek Diagram* untuk memodelkan struktur objek.
9. *Component Diagram* untuk memodelkan komponen objek.
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan 4 macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi objek, yaitu *use case diagram*, *sequence diagram, collaboration diagram,* dan *class diagram.*

2.4.2.2 Use Case Diagram

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk *elips* dengan nama operasi dituliskan didalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

2.4.2.3 Sequence Diagram

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*. Interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

2.4.2.4 Collaboration Diagram

Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration diagram* lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek.

2.4.2.5 Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.