|  |
| --- |
|  **DOCUMENT MINING****11** |
| JUMLAH PERTEMUAN : 1 PERTEMUANTUJUAN INSTRUKSIONAL KHUSUS :1. Memahami pengertian Documents mining
2. Memahami proses dalam document mining
3. Memahami metode untuk menentukan dokument yang sesuai dengan pencarian
 |

**Materi :**

* 1. **Pendahuluan**

Penyimpanan dokumen secara *digital* berkembang dengan pesat seiring meningkatnya penggunaan komputer. Kondisi tersebut memunculkan masalah untuk mengakses informasi yang diinginkan secara akurat dan cepat. Oleh karena itu, walaupun sebagian besar dokumen *digital* tersimpan dalam bentuk teks dan berbagai algoritma yang efisien untuk pencarian teks telah dikembangkan, teknik pencarian terhadap seluruh isi dokumen yang tersimpan bukanlah solusi yang tepat mengingat pertumbuhan ukuran data yang tersimpan umumnya.

Tujuan dari *document mining* atau sering disebut *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Jadi, sumber data yang digunakan pada *text mining* adalah kumpulan teks yang memiliki format yang tidak terstruktur atau minimal semi terstruktur. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks (*text categorization*) dan pengelompokan teks (*text clustering*). Algoritma yang digunakan pada *text mining*, biasanya tidak hanya melakukan perhitungan hanya pada dokumen dan juga *feature*. Empat macam *feature* yang sering digunakan yaitu :

* *Character*, merupakan komponan individual, bisa huruf, angka, karakter spesial dan spasi, merupakan *block* pembangun pada level paling tinggi pembentuk semantik *feature*, seperti kata, *term* dan *concept*. Pada umumnya, representasi *character-based* ini jarang digunakan pada beberapa teknik pemrosesan teks.
* *Words*.
* *Terms* merupakan *single word* dan frasa *multiword* yang terpilih secara langsung dari corpus. Representasi *term-based* dari dokumen tersusun dari subset *term* dalam dokumen.
* *Concept*, merupakan *feature* yang degenerate dari sebuah dokumen secara manual, *rule-based*, atau metodologi lain. Pada tugas akhir ini, konsep *degenerate* dari *argument* atau *verb* yang sudah diberi label pada suatu dokumen.
	1. **Komponen Document Mining**

Dokumen Mining memiliki definisi menambang data yang berupa text dimana sumber data biasanya didapat dari dokumen, dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. Tahapan yang dilakukan secara umum dalam Dokumen Mining adalah : tokenizing, filtering, stemming, tagging dan analyzing.

****

**Gambar 11.1.** Tahapan Secara Umum dalam *Text Mining*

**Tahap *tokenizing***adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Contoh dari tahap ini adalah sebagai berikut :

****

**Gambar 11.2.** Contoh Tahap *Tokenizing*



**Gambar 11.3. Flowchart Parsing**

**Tahap *filtering***adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata-kata yang kurang penting) atau *word list*. Contoh dari tahap ini adalah sebagai berikut :



**Gambar 11.3.** Contoh Tahap *Filtering*

**Tahap *Stemming***adalah tahap mencari *root* kata dari tiap kata hasil *filtering*. Tahap ini kebanyakan dipakai untuk text berbahasa inggris. Hal ini dikarenakan karena bahasa Indonesia tidak memiliki rumus bentuk baku yang permanen. Penggunaannya dalam text bahasa Indonesia adalah ketika menghilangkan imbuhan pada suatu kalimat. Seperti kata “melihat” di *stemming* sehingga menghasilkan kata “lihat” karena dihilangkan imbuhan me-. Contoh lain dari tahap *stemming* adalah sebagai berikut :



**Gambar 11.4.** Contoh Tahap *Stemming*



**Gambar 11.5.** Bagan metode *stemming*

*Porter stemmer*[3] merupakan algoritma penghilangan akhiran *morphological* dan *infleksional* yang umum dari bahasa Inggris. Step-step algoritma *Porter Stemmer* :

1. Step 1a : *remove plural suffixation*

2. Step 1b : *remove verbal inflection*

3*.* Step 1b1 : *continued for -ed and –ing rules*

4. Step 1c : y and i

5. Step 3

6. Step 4 : delete last suffix

7. Step 5a : *remove e*

8. Step 5b : reduction



**Gambar 11.6.** *Control flow* algoritma *Porter Stemmer*

*Porter Stemmer for Bahasa Indonesia* dikembangkan oleh Fadillah Z. Tala pada tahun 2003. Implementasi *Porter Stemmer for Bahasa Indonesia* berdasarkan English *Porter* *Stemmer* yang dikembangkan oleh W.B. Frakes pada tahun 1992. Karena bahasa Inggris datang dari kelas yang berbeda, beberapa modifikasi telah dilakukan untuk membuat Algoritma *Porter* dapat digunakan sesuai dengan bahasa Indonesia[8]. Desain dari *Porter Stemmer for Bahasa Indonesia* dapat dilihat pada gambar 11.7.

Pada gambar 11.7 terlihat beberapa langkah *'removal'* menurut aturan yang ada pada tabel 1 sampai dengan tabel 5.

**Tabel 11.1: Kelompok *rule* pertama : *inflectional particles*[8]**



**Tabel 11.2: Kelompok *rule* kedua :*inflectional possesive pronouns*[8]**





**Gambar 11.7. Desain dari *Porter Stemmer for Bahasa Indonesia*[8]**

**Tabel 11.3: Kelompok *rule* ketiga: *first order of derivational prefixes*[8]**



**Tabel 11.4: Kelompok *rule* keempat: *second order of derivational prefixes*[8]**



**Tabel 11.5: Kelompok *rule* kelima: *derivational suffixes*[8]**



**Tahap *tagging***adalah tahap mencari bentuk awal / *root* dari tiap kata lampau atau kata hasil *Stemming*. Tahap ini banyak dipakai untuk text yang berbahasa inggris karena bahasa Indonesia tidak memiliki bentuk lampau. Contoh dari tahap ini adalah sebagai berikut

****

**Gambar 11.8.** Contoh Tahap *Tagging*

Dalam pengklasifikasian jurnal tugas akhir ini, tahapan *text mining* dilakukan hanya sampai tahap *filtering*. Hal ini disebabkan karena pengklasifikasian berdasarkan judul dan abstrak tidak menangani kalimat maupun paragraf yang berbahasa inggris.

* 1. **Model Ruang Vektor (Vector Space Model)**

Tiap *term* atau kata yang ditemukan pada dokumen dan *query* diberi bobot dan disimpan sebagai salah satu elemen vektor dan dihitung nilai kemiripan antara *query* dan dokumen.

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.

1. **Model Ruang Vektor**

Pada document mining terdapat beberapa metode yang digunakan dalam *analisa* 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:

* 1. *Vocabulary* merupakan kumpulan semua *term* berbeda yang tersisa dari dokumen setelah *preprocessing* dan mengandung *t term index*. *Term-term* ini membentuk suatu ruang vektor.
	2. Setiap *term i* di dalam dokumen atau *query j*, diberikan suatu bobot (*weight*) bernilai *real* W*ij*.
	3. Dokumen dan *query* diekspresikan sebagai vektor *t* dimensi *dj* = (W1, W2, ..., W*tj*) dan terdapat *n* dokumen di dalam koleksi, yaitu *j* = 1, 2, ..., n.

Contoh dari model ruang vektor tiga dimensi untuk dua dokumen D1 dan D2, satu query pengguna Q1, dan tiga term T1, T2 dan T3 diperlihatkan pada gambar 2.5.

****

**Gambar 11.9. Contoh Model Ruang Vektor dengan dua dokumun D1 dan D2,**

**serta *query* Q1. [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 11.10 Contoh matriks *term-document* untuk *database* dengan *n***

**document dan *t term*. [1]**

1. **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

Faktor yang memegang peranan penting dalam pembobotan kata, yaitu :

1. ***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.

* 1. *Logarithmic* Tf

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

𝑇𝑓=1+log (𝑇𝑓) (2.1)

* 1. *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.

* 1. Augmented Tf

𝑇𝑓=0.5+0.5 ×$\frac{Tf}{max (Tf) }$ (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.

1. ***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 :

$Idf\_{i}=log\_{2}\left(\frac{N}{dj\_{i}}\right)$ (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} x idf $ (2.4)

1. **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 \left(D,Q\right)=\cos(θ= \frac{\sum\_{i=1}^{n}(WDi ×WQi)}{\sqrt{\sum\_{i=1}^{n}(WD\_{i})^{2}x \sum\_{i=1}^{n}(WQ\_{i})^{2}} }) $ (2.5)

Dengan W adalah bobot dari *query* dan dokumen.

1. **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 *dj* dan*query q,* serat *t term* diekstrak dari koleksi dokumen maka nilai kosinus antara *dj* dan *q* didefinisikan sebagai [1] :

$$Similarity \left(\vec{d\_{j}},\vec{q}\right)=\frac{\vec{d\_{j}}.\vec{q}}{\left|\vec{d\_{j}}\right|.\left|\vec{q}\right|}θ= \frac{\sum\_{i=1}^{n}(W\_{ij} ×W\_{iq})}{\sqrt{\sum\_{i=1}^{n}(WD\_{i})^{2}x \sum\_{i=1}^{n}(WQ\_{i})^{2}} }$$

Contoh:

Jika dua dokumen D1 = 2T1 + 6T2 + 5T3 dan D2 = 5T1 + 5T2 + 2T3 dan query Q1 = 0T1 + 0T2 + 2T3 sebagaimana diperlihatkan pada gambar 11.9, berikut ini adalah nilai kosinus yang diperoleh:

$$Similarity \left(\vec{D\_{1}},\vec{Q\_{1}}\right)=cosθ=\frac{(2.0+6.0+5.2)}{\sqrt{\left(4+36+25\right).(0+0+4)}}=\frac{10}{\sqrt{65.4}}=0.62$$

$$Similarity \left(\vec{D\_{2}},\vec{Q\_{1}}\right)=cosθ=\frac{(5.0+5.0+2.2)}{\sqrt{\left(25+25+4\right).(0+0+4)}}=\frac{4}{\sqrt{54.4}}=0.27$$

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

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:

$W\_{ij}=W\_{i}x tf\_{ij} x idf\_{i}$(2.7)

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

Dice $Similarity \left(\vec{d\_{j}},\vec{q}\right)=\frac{2 x \sum\_{i=1}^{t}(W\_{ij} ×W\_{iq})}{\sum\_{i=1}^{t}(W\_{ij})^{2}x \sum\_{i=1}^{t}(W\_{iq})^{2} }$:

Jaccard : $Similarity \left(\vec{d\_{j}},\vec{q}\right)=\frac{2 x \sum\_{i=1}^{t}(W\_{ij} ×W\_{iq})}{\sum\_{i=1}^{t}(W\_{ij})^{2}x \sum\_{i=1}^{t}(W\_{iq})^{2}-\sum\_{i=1}^{t}(W\_{ij}.W\_{iq}) }$

Overlap : $Similarity \left(\vec{d\_{j}},\vec{q}\right)=\frac{2 x \sum\_{i=1}^{t}(W\_{ij} ×W\_{iq})}{min\sum\_{i=1}^{t}(W\_{ij})^{2}x \sum\_{i=1}^{t}(W\_{iq})^{2} }$