BAB II
LANDASAN TEORI

2.1 *Text mining*

Menurut Feldman dan Sanger (Feldman dan Sanger, 2007), *text mining* dapat didefinisikan secara luas sebagai proses pengetahuan intensif yang memungkinkan pengguna berinteraksi dengan koleksi dokumen dari waktu ke waktu menggunakan berbagai macam analisis. Dalam cara yang sejalan dengan *data mining*, *text mining* berusaha mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi patterns. *Text mining* menjadi menarik karena sumber data koleksi dokumen dan pola yang menarik tidak ditemukan dari database formal namun ditemukan dalam data tekstual yang tidak terstruktur pada kumpulan dokumen. *Text mining* juga dapat diartikan sebagai sebuah proses untuk menemukan suatu informasi atau tren baru yang sebelumnya tidak terungkap dengan memroses dan menganalisis data dalam jumlah besar.

Text mining bertujuan untuk mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisis keterhubungan antar dokumen. Pada dasarnya proses kerja dari *text mining* banyak mengadopsi penelitian data mining, namun yang menjadi perbedaan adalah pola yang digunakan oleh *text mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur, sedangkan dalam data mining pola yang diambil adalah dari data yang terstruktur. (Han & Kamber, 2006).

2.2 *Text Preprocessing*

Pada *text mining* data yang di gunakan berasal dari dokumen atau teks yang tidak terstruktur. Oleh karena itu, dibutuhkan suatu proses yang dapat mengubah bentuk data yang sebelumnya tidak terstruktur menjadi data yang terstruktur. Proses ini bertujuan agar data yang akan digunakan nantinya bersih dari *noise* atau ciri-ciri yang tidak berpengaruh pada klasifikasi sentimen seperti link, “@”, “RT”,

stopword. Proses preprosesing juga mempunyai tujuan agar data yang digunakan memiliki dimensi yang lebih kecil dan lebih terstruktur, sehingga dapat diolah lebih lanjut.

Pada penelitian ini, tahapan *Preprocessing* adalah *case folding*, *tokenizing*, *stopword removal*, dan *stemming*. Keseluruhan tahapan memiliki fungsi dan perannya masing-masing. Untuk mendapatkan dataset yang berdimensi lebih kecil dari data sebelumnya, terstruktur, serta bersih dari *noise*, maka ke semua tahap harus berkesinambungan. Secara umum proses yang dilakukan dalam tahapan *preprocessing* adalah sebagai berikut

1. *Case folding*

   Case folding merupakan proses *text preprocessing* yang dilakukan untuk menyeragamkan karakter pada data (dokumentasi/teks). Pada proses ini, semua huruf besar (*uppercase*) dijadikan huruf kecil (*lowercase*).

2. *Tokenizing*

   Tokenizing adalah proses pemotongan sebuah dokumen menjadi bagian-bagian, yang disebut dengan token. Pada saat bersamaan tokenizing juga berfungsi untuk membuang beberapa karakter tertentu yang dianggap tanda baca.

3. *Stopword Removal*

   Stopword removal adalah proses penghilangan kata-kata yang tidak berkontribusi banyak pada isi dokumen. Kata-kata yang termasuk ke dalam *stopword* dihilangkan karena memberikan pengaruh yang tidak baik dalam proses *text mining* seperti kata-kata “di”, ”oleh”, “pada”, ”sebuah”, ”karena” dan lain sebagainya.

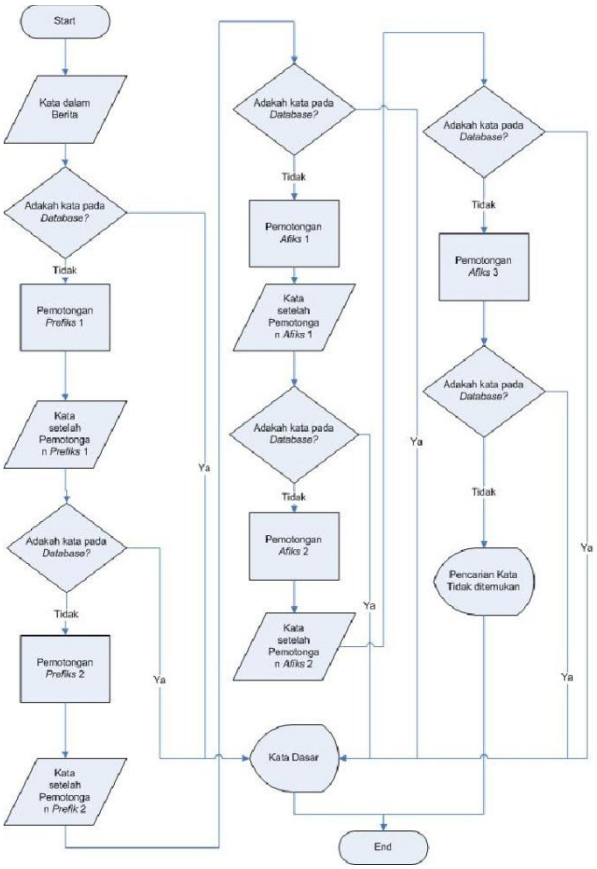
4. *Stemming*

   Stemming adalah suatu proses pengembalian suatu kata berimbuhan ke dalam bentuk kata dasarnya *(root). Stemming* adalah alat pemprosesan teks dasar yang sering digunakan untuk meningkatkan kinerja pada *text retrieval* dan *text classification*. Namun sama pada halnya *stopword*, kinerja *stemming* juga bervariasi dan sering bergantung pada domain bahasa yang digunakan.

Stemming merupakan bagian yang tidak terpisahkan dalam *Information Retrieval* (*IR*). Tidak banyak algoritma yang dikhususkan untuk *stemming* bahasa Indonesia dengan berbagai keterbatasan didalamnya. Algoritma Porter salah satunya, Algoritma ini membutuhkan waktu yang lebih singkat dibandingkan dengan *stemming* menggunakan Algoritma Nazief & Adriani, namun proses *stemming* menggunakan Algoritma Porter memiliki prosentase keakuratan (presisi) lebih kecil dibandingkan dengan *stemming* menggunakan Algoritma Nazief & Adriani. Algoritma Nazief & Adriani sebagai algoritma *stemming* untuk teks berbahasa Indonesia yang memiliki kemampuan prosentase keakuratan (presisi) lebih baik dari algoritma lainnya. Algoritma ini sangat dibutuhkan dan menentukan dalam proses IR dalam dokumen Indonesia.

Stemming adalah salah satu cara yang digunakan untuk meningkatkan performa IR dengan cara mentransformasi kata-kata dalam sebuah dokumen teks ke bentuk kata dasarnya. Algoritma *stemming* untuk bahasa yang satu berbeda dengan algoritma *stemming* untuk bahasa lainnya. Sebagai contoh bahasa Inggris memiliki morfologi yang berbeda dengan bahasa Indonesia sehingga algoritma *stemming* untuk kedua bahasa tersebut juga berbeda. Proses *stemming* pada teks berbahasa Indonesia lebih rumit/kompleks karena terdapat variasi imbuhan yang harus dibuang untuk mendapatkan *root word* (kata dasar) dari sebuah kata.. Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

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



Gambar 2.1 Alur Proses *Stemming*

Algoritma Nazief & Adriani yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Pertama cari kata yang akan di *stem* dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
    1. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut

ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.

2. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.

4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   1. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak
   2. pergi ke langkah 4b.
   3. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika root word belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
   5. Melakukan Recoding.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

2.3 *Sentiment analysis*

Sentiment analysis atau *opinion mining* mengacu pada bidang yang luas dari pengolahan bahasa alami, komputasi linguistik dan *text mining* yang bertujuan menganlisa pendapat, sentimen, evaluasi, sikap, penilaian dan emosi seseorang apakah pembicara atau penulis berkenaan dengan suatu topik , produk, layanan, organisasi, individu, ataupun kegiatan tertentu (Liu, 2010).

Tugas dasar dalam analisis sentimen adalah mengelompokkan teks yang ada dalam sebuah kalimat atau dokumen kemudian menentukan pendapat yang dikemukakan dalam kaliamat atau dokumen tersebut apakah bersifat positif, negatif atau netral. *Sentiment analysis* juga dapat menyatakan perasaan emosional sedih, gembira, atau marah.

Kita dapat mencari pendapat tentang produk-produk, merek atau orang-orang dan menentukan apakah mereka dilihat positif atau negatif di web (Saraswati, 2011). Hal ini memungkinkan kita untuk mencari informasi tentang:

a. Deteksi Flame (rants buruk)
b. Persepsi produk baru.
c. Persepsi Merek.
d. Manajemen reputasi.

Ekspresi atau sentimen mengacu pada fokus topik tertentu, pernyataan pada satu topik mungkin akan berbeda makna dengan pernyataan yang sama pada *subject* yang berbeda. Oleh karena itu pada beberapa penelitian, terutama pada review produk, pekerjaan didahului dengan menentukan elemen dari sebuah produk yang sedang dibicarakan sebelum memulai proses *opinion mining*.

Sentiment analysis dapat dibedakan berdasarkan sumber datanya, beberapa level yang sering digunakan dalam penelitian adalah *Sentiment analysis* pada level dokumen dan *Sentiment analysis* pada level kalimat. Berdasarkan level sumber datanya *Sentiment analysis* terbagi menjadi 2 kelompok besar yaitu :

1. Coarse-grained *Sentiment analysis*
2. Fined-grained *Sentiment analysis*

2.3.1 *Coarse-grained Sentiment analysis*

Pada coarse-grained *sentiment analysis* dilakukan proses analysis pada level dokumen. Singkatnya adalah kita mencoba menklasifikasikan orientasi sebuah dokumen secara keseluruhan sebagai sentiment positif atau sentimen negatif (Fink Clayton, 2011).

2.3.2 *Fined-grained Sentiment analysis*

Fined-grained *sentiment analysis* adalah anaisis sentimen pada level kalimat. Fokus utama fined-grained *sentiment analysis* adalah menetukan sentimen pada setiap kalimat pada suatu dokumen, dimana kemungkinan yang terjadi adalah terdapat sentimen pada level kalimat yang berbeda pada suatu document (Fink Clayton, 2011).

2.4 *Lexicon based*

Salah satu pendekatan yang umum digunakan dalam melakukan analisis sentimen adalah dengan menggunakan *dictionary based approach*. Dalam paper

Yan Dang et. al., metode ini disebut juga sebagai *lexical based approach*. *Lexical based approach* merupakan sebuah metode untuk melakukan analisis sentimen dengan menggunakan sebuah kamus sebagai sumber bahasa atau *lexical*. *Opinion lexicon* yang digunakan dalam penelitian kali ini adalah *opinion lexicon* milik Hu dan Liu. Terdiri dari kurang lebih 6800 kata yang dibagi ke dalam dua kelas, yaitu kata positif dan negatif. Kamus ini telah disusun oleh Hu dan liu dalam beberapa tahun terakhir dimulai dari paper pertama yang mereka buat untuk acara *Conference on Knowledge and Data Mining* pada tahun 2004.

2.5 *Naïve Bayes*

Algoritma *naive bayes classifier* merupakan algoritma yang digunakan untuk mencari nilai probabilitas tertinggi untuk mengklasifikasi data uji pada kategori yang paling tepat (Feldman & Sanger 2007). Dalam penelitian ini yang menjadi data uji adalah dokumen *tweets*. Ada dua tahap pada klasifikaasi dokumen. Tahap pertama adalah pelatihan terhadap dokumen yang sudah diketahui kategorinya. Sedangkan tahap kedua adalah proses klasifikasi dokumen yang belum diketahui kategorinya.

Dalam algoritma *Naïve Bayes classifier* setiap dokumen direpresentasikan dengan pasangan atribut "x1, x2, x3,...xn" dimana x1 adalah kata pertama, x2 adalah kata kedua dan seterusnya. Sedangkan V adalah himpunan kategori *tweet*. Pada saat klasifikasi algoritma akan mencari probabilitas tertinggi dari semua kategori dokumen yang diujikan (VMAP), dimana persamaannya adalah sebagai berikut :

$$V_{MAP} = \underset{V_j \, e \, V}{\arg\max} \frac{P(x_1,x_2,x_3,...x_n|V_j)P(V_j)}{P(x_1,x_2,x_3,...x_n)}$$

Untuk P(x1, x2, x3,...xn) nilainya konstan untuk semua kategori (Vj) sehingga persamaan dapat ditulis sebagai berikut :

$$V_{MAP} = \underset{V_j \, e \, V}{\arg max} P(x_1, x_2, x_3, \ldots x_n | V_j) P(V_j)$$

Persamaan diatas dapat disederhanakan menjadi sebagai berikut :

$$V_{MAP} = \underset{V_j \, e \, V}{\arg max} \prod_{i=1}^{n} P(x_i | V_j) P(V_j)$$

Keterangan :

Vj = Kategori *tweet* j =1, 2, 3,…n. Dimana dalam penelitian ini j1 = kategori *tweet* sentimen negatif, j2 = kategori *tweet* sentimen positif, dan j3 = kategori *tweet sentimen* netral

P(xi|Vj) = Probabilitas xi pada kategori Vj

P(Vj) = Probabilitas dari Vj

Untuk P(Vj) dan P(xi|Vj) dihitung pada saat pelatihan dimana persamaannya adalah sebagai berikut :

$$P(V_j) = \frac{|docs\ j|}{|contoh|}$$

$$P(x_i|V_j) = \frac{n_k + 1}{n + |kosakata|}$$

Keterangan :

|docs j| = jumlah dokumen setiap kategori j

|contoh| = jumlah dokumen dari semua kategori

Nk = jumlah frekuensi kemunculan setiap kata

n = jumlah frekuensi kemunculan kata dari setiap kategori

|kosakata| = jumlah semua kata dari semua kategori

2.6 Bahasa pemograman PHP

PHP adalah *script* bersifat *sever-side* yang ditambahkan ke dalam HTML. *Script* ini akan membuat suatu aplikasi yang dapat diintegrasikan ke dalam HTML sehingga suatu halaman *web* tidak lagi bersifat statis namun bersifat dinamis. Sifat *sever-side* berarti pengerjaan *script* dilakukan di *server* baru kemudian hasilnya dikirimkan ke *browser*. (Sutabri, 2009). PHP merupakan *software* yang *open source* bebas. Jadi *source code* dapat dirubah dan didistribusikan secara bebas dan gratis. PHP juga dapat berjalan lintas *platform*, yaitu dapat digunakan dengan sistem operasi(Windows dan Linux) dan *web server* apapun (misalnya: PWS, IIS, Apache).

PHP (*Hypertext Preprocessor*) adalah bahasa computer yang dibuat untuk pengembangan web dinamis. Pada umumnya PHP digunakan di server namun juga dapat berdiri sendiri sebagai aplikasi *graphical* (www.php.net, 2008).

Penggunaan PHP dan MySQL dipilih karena PHP dan MySQL memiliki beberapa kelebihan seperti dinyatakan oleh Nugroho, B (2008) kelebihannya sebagai berikut:

1. Bahasa pemograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. Web *Server* yang mendukung PHP dapat ditemukan dimana-mana dari mulai IIS sampai dengan Apache dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis-milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena referensi yang banyak.
5. PHP adalah bahasa *opensource* yang dapat digunakan di berbagai mesin (Linux, Unix, Windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

2.7 *Twitter*

Twitter adalah sebuah situs web yang dimiliki dan dioperasikan oleh *Twitter* Inc., yang menawarkan jaringan sosial berupa mikroblog sehingga memungkinkan

penggunanya untuk mengirim dan membaca pesan *Tweets* (*Twitter*, 2013). Mikroblog adalah salah satu jenis alat komunikasi *online* dimana pengguna dapat memperbarui status tentang mereka yang sedang memikirkan dan melakukan sesuatu, apa pendapat mereka tentang suatu objek atau fenomena tertentu. *Tweets* adalah teks tulisan hingga 140 karakter yang ditampilkan pada halaman profil pengguna. *Tweets* bisa dilihat secara publik, namun pengirim dapat membatasi pengiriman pesan ke daftar teman-teman mereka saja. Pengguna dapat melihat *Tweets* pengguna lain yang dikenal dengan sebutan pengikut *(follower)*.

Tidak seperti Facebook, LinkedIn, dan MySpace, *Twitter* merupakan sebuah jejaring sosial yang dapat digambarkan sebagai sebuah graph berarah (Wang, 2010), yang berarti bahwa pengguna dapat mengikuti pengguna lain, namun pengguna kedua tidak diperlukan untuk mengikutinya kembali. Kebanyakan akun berstatus publik dan dapat diikuti tanpa memerlukan persetujuan pemilik.

Semua pengguna dapat mengirim dan menerima *Tweets* melalui situs *Twitter*, aplikasi eksternal yang kompatibel (telepon seluler), atau dengan pesan singkat (SMS) yang tersedia di negara-negara tertentu (*Twitter*, 2013). Pengguna dapat menulis pesan berdasarkan topik dengan menggunakan tanda # (*hashtag*). Sedangkan untuk menyebutkan atau membalas pesan dari pengguna lain bisa menggunakan tanda @.

Pesan pada awalnya diatur hanya mempunyai batasan sampai 140 karakter disesuaikan dengan kompatibilitas dengan pesan SMS, memperkenalkan singkatan notasi dan slang yang biasa digunakan dalam pesan SMS. Batas karakter 140 juga meningkatkan penggunaan layanan memperpendek URL seperti bit.ly, goo.gl, dan tr.im, dan jasa hosting konten, seperti Twitpic, Tweephoto, memozu.com dan NotePub untuk mengakomodasi multimedia isi dan teks yang lebih panjang daripada 140 karakter (*Twitter*, 2013). *Twitter* menggunakan bit.ly untuk memperpendek otomatis semua URL yang dikirim-tampil. Fitur yang terdapat dalam *Twitter*, antara lain:

1. Laman Utama (*Home)*

   Pada halaman utama kita bisa melihat *Tweets* yang dikirimkan oleh orang-orang yang menjadi teman kita atau yang kita ikuti (*following*).

2. Profil (*Profile*)

   Pada halaman ini yang akan dilihat oleh seluruh orang mengenai profil atau data diri serta *Tweets* yang sudah pernah kita buat.

3. *Followers*

   Pengikut adalah pengguna lain yang ingin menjadikan kita sebagai teman. Bila pengguna lain menjadi pengikut akun seseorang, maka *Tweets* seseorang yang ia ikuti tersebut akan masuk ke dalam halaman utama.

4. *Following*

   Kebalikan dari pengikut, *following* adalah akun seseorang yang mengikuti akun pengguna lain agar *Tweets* yang dikirim oleh orang yang diikuti tersebut masuk ke dalam halaman utama.

5. *Mentions*

   Biasanya konten ini merupakan balasan dari percakapan agar sesama pengguna bisa langsung menandai orang yang akan diajak bicara.

6. *Favorite*

   Tweets ditandai sebagai favorit agar tidak hilang oleh halaman sebelumnya.

7. Pesan Langsung (*Direct Message*)

   Fungsi pesan langsung lebih bisa disebut SMS karena pengiriman pesan langsung di antara pengguna.

8. *Hashtag*

   Hashtag "#" yang ditulis di depan topik tertentu agar pengguna lain bisa mencari topik yang sejenis yang ditulis oleh orang lain juga.

9. *List*

   Pengguna *Twitter* dapat mengelompokkan ikutan mereka ke dalam satu grup sehingga memudahkan untuk dapat melihat secara keseluruhan para nama pengguna (*username*) yang mereka ikuti (*follow*).

10. Topik Terkini (*Trending Topic*)

    Topik yang sedang banyak dibicarakan banyak pengguna dalam suatu waktu yang bersamaan.

2.8 *Confusion matrix*

Confusion matrix (Kohavi & Provost, 1998) berisi informasi mengenai hasil klasifikasi actual dan yang telah dipresiksi oleh sistem klasifikasi. Performa dari sistem tersebut biasanya dievacluasi menggunakan data dalam sebuah matrix.

Tabel dibawah ini menampilkan sebuah *confusion matrix* untuk pengklasifikasian ke dalam dua kelas.

a-> jumlah prediksi yang benar untuk data aktual negatif

b-> jumlah prediksi yang salah untuk data aktual positif

c-> jumlah prediksi yang salah untuk data aktual negatif

d-> jumlah prediksi yang benar untuk data aktual positif.

Tabel 2.1 *Confusion matrix*

		PREDICTED	
		NEGATIVE	***POSITIVE***
ACTUAL	***NEGATIVE***	**A**	**C**
	POSITIVE	**B**	**D**

Beberapa term standar yang telah ditetapkan untuk matrix dua kelas diatas adalah sebagai berikut:

1. Accuracy (AC) adalah proporsi jumlah prediksi yang benar. Hal ini ditentukan dengan menggunakan persamaan

$$AC = \frac{a + d}{a + b + c + d}$$

2. Recall atau True *Positive* Rate (FP) adalah proposi dari kasus positif yang diidentifikasi dengan benar, dihitung dengan menggunakan persamaan :

$$TP = \frac{d}{c + d}$$

3. False *Positive* Rate (FP) adalah proporsi dari kasus negatif yang salah diklasifikasi sebagai positif, dihitung dengan menggunakan persamaan

$$FP = \frac{b}{a+b}$$

4. True *Negative* Rate (TN) didefinisikan sebagai proporsi untuk kasus negatif yang diklasifikasikan dengan benar, dihitung dengan menggunakan persamaan.

$$TN = \frac{a}{a+b}$$

5. False *Negative* Rate (FN) adalah proposi dari kasus positif yang salah diklasifikasikan sebagai negatif, dihitung dengan menggunakan persamaan

$$FN = \frac{c}{c + d}$$

6. Precision (P) adalah proporsi kasus dengan hasil positif yang benar dengan menggunakan persamaan.

$$P = \frac{d}{b + d}$$

2.9 Metode TF IDF

Metode Term Frequency-Inverse Document Frequency (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu Term frequency (TF)

merupakan frekuensi kemunculan kata (t) pada kalimat (d). Document frequency (DF) adalah banyaknya kalimat dimana suatu kata (t) muncul. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen (Robertson, 2005). Pada Metode ini pembobotan kata dalam sebuah dokumen dilakukan dengan mengalikan nilai TF dan IDF. Pembobotan diperoleh berdasarkan jumlah kemunculan term dalam kalimat (TF) dan jumlah kemunculan term pada seluruh kalimat dalam dokumen (IDF). Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen. Nilai IDF sebuah term dihitung menggunakan persamaan di bawah:

$$W_{d,t} = TF_{d,t} * IDF_t$$

dengan :
d = kalimat ke-d
t = kata(*term*) ke –t
TF = *term fregency*
W = bobot kalimat ke-d terhadap kata(*term*)ke- t
IDF = *inverse document f regency*

Menghitung bobot masing-masing dokumen dengan persamaan di bawah:

$$IDF = log\left(\frac{N}{df_t}\right)$$

dengan:
N = jumlah kalimat yang berisi *term*(t)
df_t = jumlah kemunculan kata (*term*) terhadap D

Kemudian baru melakukan proses pengurutan (sorting) nilai kumulatif dari W untuk setiap kalimat. Tiga kalimat dengan nilai W terbesar dijadikan sebagai hasil dari ringkasan atau sebagai output dari peringkasan teks otomatis.