BAB II

LANDASAN TEORI

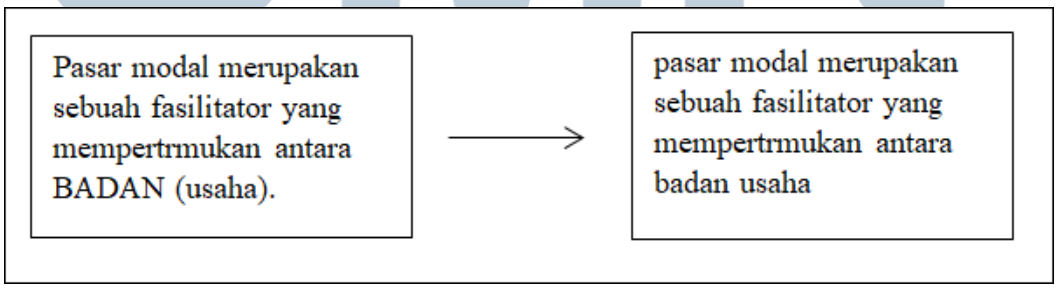
Adapun literatur yang berhubungan dengan implementasi metode TF-IDF dan *Support Vector Machine* pada aplikasi pendeteksi komentar berifat *bullying* adalah sebagai berikut.

2.1 Text Preprocessing

Tahap *pre-processing* atau praproses data merupakan proses untuk mempersiapkan data mentah sebelum dilakukan proses lain. Pada umumnya, praproses data dilakukan dengan cara mengeliminasi data yang tidak sesuai atau mengubah data menjadi bentuk yang lebih mudah diproses oleh sistem (Mujilahwati, 2016). Menurut Marfian (2015) terdapat beberapa tahapan *text preprocessing* sebagai berikut.

1. *Case Folding*

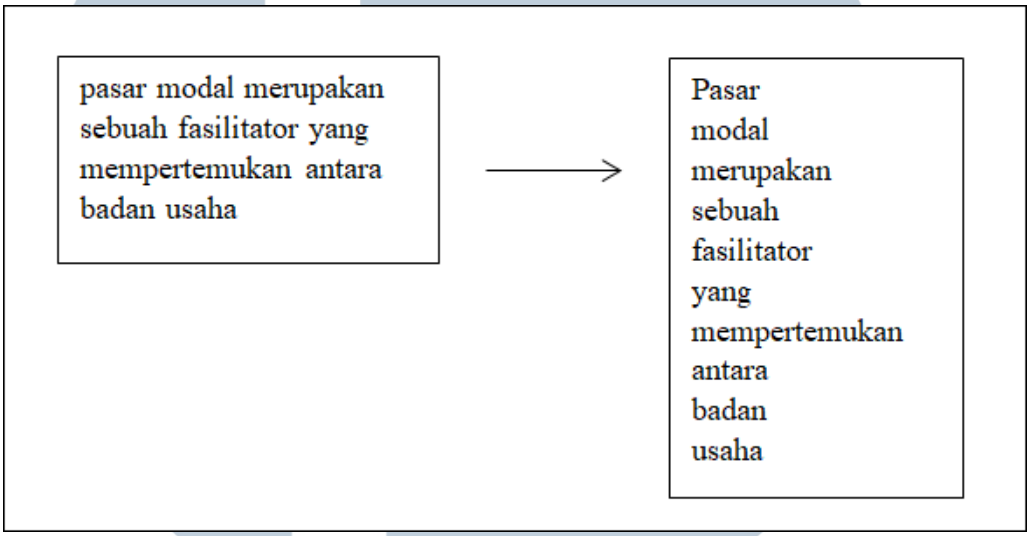
Pada tahap ini semua huruf dalam dokumen diubah menjadi huruf kecil, dan karakter yang diambil adalah karakter dari ‘a’ sampai ‘z’ dimana karakter selain itu akan dihilangkan dan dianggap delimiter (Ronen Feldman ,2007).



Gambar 2.1 Proses *Case Folding.* (Sumber : Marfian, 2015)

2. *Tokenizing*

Tahap *Tokenizing* adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya.



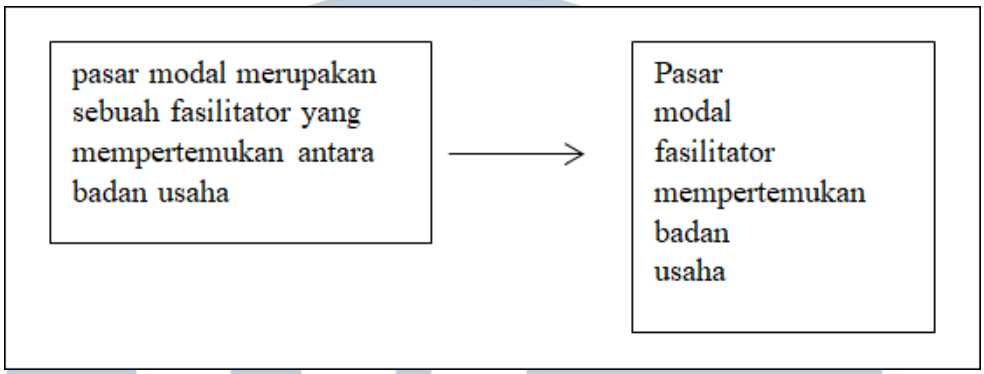
Gambar 2.2 Proses *Tokenizing*. (Sumber : Marfian, 2015)

2.2 Text Transformation

Text Transformation adalah tahapan yang dipergunakan untuk mengubah kata-kata ke dalam bentuk dasar, sekaligus untuk mengurangi jumlah kata-kata tersebut. Pendekatan yang dapat dilakukan adalah dengan *stemming* dan penghapusan *stopwords* (Larasati, 2015).

1. *Stopword removal* atau *filtering*

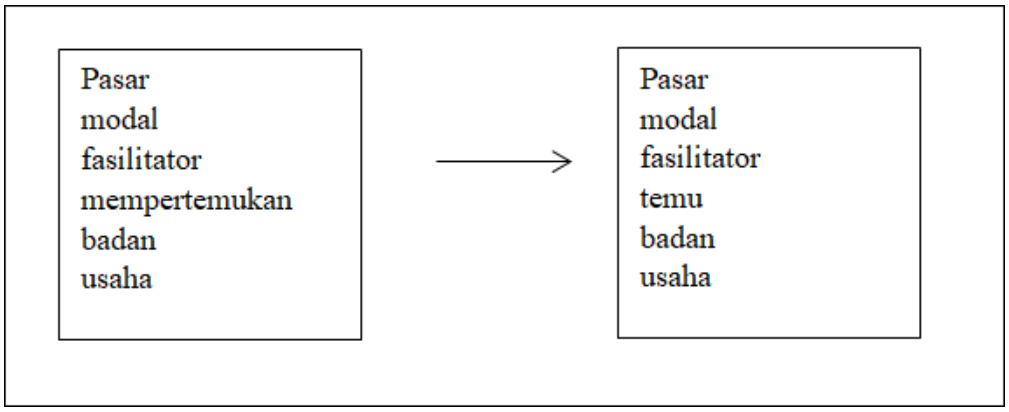
Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist* / *stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words* (Ronen Feldman,2007).



Gambar 2.3 Proses *Stopword Removal*. (Sumber : Marfian, 2015)

2. *Stemming*

Stemming merupakan proses membentuk suatu kata menjadi kata dasarnya. Agusta (2009) dalam penelitian perbandingan algoritma *stemming* Porter dengan algoritma Nazief & Adriani menyimpulkan bahwa proses *stemming* dengan algoritma Porter memiliki persentase keakuratan (presisi) yang lebih kecil dibandingkan steeming menggunakan algoritma Nazief & Adriani.



Gambar 2.4 Proses *Stemming*. (Sumber : Marfian, 2015)

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

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut 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
   a. 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.
   b. 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.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b. *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. 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 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF adalah metode yang digunakan untuk menghitung bobot setiap kata yang telah diekstrak (Luqyana dkk., 2018). Model pembobotan TF-IDF merupakan metode yang mengintegrasikan model *term frequency* (TF) dan *inverse document frequency* (IDF). *Term frequency* (TF) merupakan proses untuk menghitung jumlah kemunculan term dalam satu dokumen dan *inverse document frequency* (IDF) digunakan untuk menghitung *term* yang muncul di berbagai dokumen (komentar) yang dianggap sebagai *term* umum, yang dinilai tidak penting (Akbari dkk., 2012). Untuk mendapatkan bobot dari sebuah *term* dapat menggunakan rumus sebagai berikut.

$$TF(t) = \frac{\textbf{jumlah kemunculan term t dalam dokumen}}{\textbf{jumlah term dalam dokumen}} \quad ...(2.1)$$

$$IDF(t) = \log\left(\frac{\textbf{jumlah seluruh dokumen}}{\textbf{jumlah dokumen yang mengandung term t}}\right) \quad ...(2.2)$$

$$W(t) = TF(t) * IDF(t) \quad ...(2.3)$$

dimana TF(t) merupakan nilai *Term Frequency* dari *term* t, IDF(t) merupakan nilai *Inverse Document Frequency* dari *term* t, dan W(t) adalah bobot dari sebuah *term*.

2.4 Support Vector Machine Nonlinear

Menurut Fachrurrazi (2011) SVM merupakan suatu teknik untuk melakukan prediksi, baik prediksi dalam kasus regresi maupun klasifikasi. SVM bertujuan

untuk menemukan *hyperlane* terbaik yang memisahkan dua buah *class* pada *input space* (Satriyo dkk.,2003). Dinyatakan x merupakan *dataset* dan *n* merupakan banyaknya data *training* dan rumus untuk SVM mencari *margin hyperlane* maksimal yang memisahkan kelas-kelas yang berbeda dapat dilihat pada rumus 2.4.

$$x = (x_n, y_n), x_n \in R^p, y_n \in \{-1,1\}, \forall\, n = 1,2,...,N \quad ...(2.4)$$

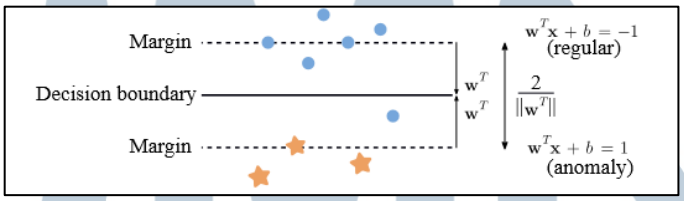
dimana x_n adalah *p-dimensional input vector* dan y_n adalah output yang bernilai (1 atau -1). Rumus untuk *vector* yang memisahkan dua kelas dapat dilihat pada rumus 2.5.

$$w^T . x + b = 0 \quad ...(2.5)$$

dimana w^T adalah *optimal weighing vector* dan b adalah *bias*. Untuk data *training* yang dapat dipisahkan secara linear, *margins* dirumuskan sebagai berikut.

$$w^T . x + b = 1 \text{ dan } w^T . x + b = -1 \quad ...(2.6)$$

dimana jarak antara dua margin dituliskan dalam $2/\|w^T\|$.



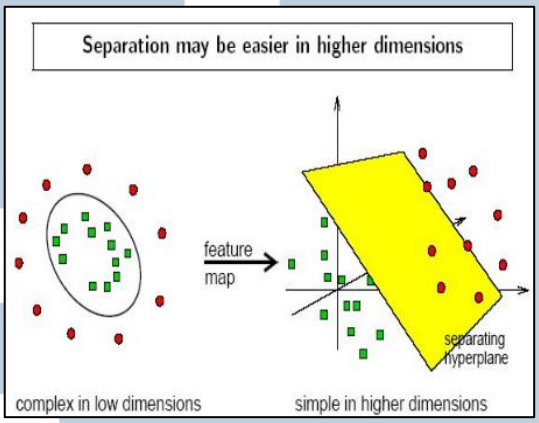
Gambar 2.5 *Support Vector Machine linear kernel.* (Sumber : Batta, 2018)

Prinsip dasar SVM adalah *linear classifier*, dan selanjutnya dikembangkan agar dapat bekerja pada *problem non-linear*. Dengan memasukkan konsep *kernel trick* pada ruang kerja berdimensi tinggi (Satriyo dkk.,2003).

Murfi (2014) menyatakan fungsi kernel adalah suatu fungsi k yang mana untuk semua vektor input x,z akan memenuhi kondisi

$$k(x,z) = \varphi(x)^T \varphi(z) \quad ...(2.7)$$

dimana φ (.) adalah fungsi pemetaan dari ruang input ke ruang fitur. Dengan kata lain, fungsi kernel adalah fungsi hasil kali dalam (*inner product*) pada ruang fitur.



Gambar 2.6 *Support Vector Machine Nonlinear Kernel.* (Sumber : Murfi, 2014)

Fungsi kernel memungkinkan untuk mengimplementasikan suatu model pada ruang dimensi lebih tinggi (ruang fitur) tanpa harus mendefinisikan fungsi pemetaan dari ruang input ke ruang fitur. Salah satu kernel yang terdapat pada SVM adalah *polynomial* kernel dengan fungsi sebagai berikut.

$$k(x_n, x_m) = (<x_n, x_m>+ 1)^p \quad ...(2.8)$$

dimana p dinyatakan sebagai parameter yang mendefinisikan *degree of polynomial.* Dengan menggunakan *polynomial kernel* (p > 1) berbeda dengan *linear kernel* (p = 1) sehingga menghasilkan batas keputusan SVM yang fleksibel. (Batta et. al.,2018).

2.5 Confusion Matrix

Confusion Matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya (Solichin, 2017). Dengan

menggunakan *confusion matrix* maka nilai akurasi, presisi, *recall*, dan *f-measure* dapat dihitung dengan rumus sebagai berikut.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} * 100\% \quad ...(2.9)$$

$$\text{Presisi} = \frac{TP}{TP+FP} * 100\% \quad ...(2.10)$$

$$\text{Recall} = \frac{TP}{FN+TP} * 100\% \quad ...(2.11)$$

$$\text{F-Measure} = 2 * \frac{\text{Presisi} * Recall}{\text{Presisi} + Recall} * 100\% \quad ...(2.12)$$

dimana TP adalah *True Positive*, yaitu jumlah data positif yang terklasifikasi dengan benar oleh sistem. TN adalah *True Negative*, yaitu jumlah data negatif yang terklasifikasi dengan benar oleh sistem. FN adalah *False Negative*, yaitu jumlah data negatif namun terklasifikasi salah oleh sistem. FP adalah *False Positive*, yaitu jumlah data positif namun terklasifikasi salah oleh sistem.