BAB II
LANDASAN TEORI

II.1 *Text Mining*

Text Mining merupakan penerapan konsep dan teknik data mining untuk mencari pola dalam teks, proses penganalisaan teks guna menemukan informasi yang bermanfaat untuk tujuan tertentu [2]. Definisi lain berkaitan dengan text mining dikatakan bahwa text mining merupakan penambangan data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen dan tujuannya adalah mencari kata- kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antara dokumen [3]. Penambangan teks juga memiliki tujuan dan menggunakan proses yang sama dengan penambangan data, hanya saja memiliki masukan yang berbeda. Masukan untuk penambangan teks adalah data yang tidak atau kurang terstrukur, seperti dokumen Word, PDF, kutipan teks, dll, sedangkan masukan untuk penambangan data adalah data yang terstruktur. [4]

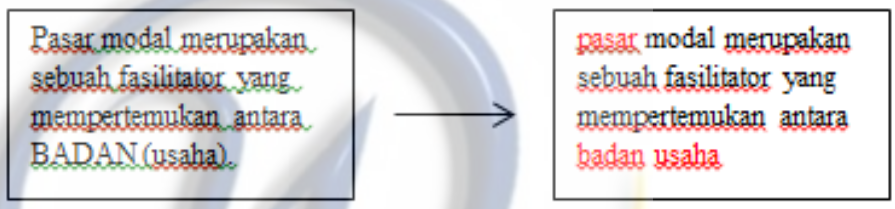
II.1.1 Text Preprocessing

Text Preprocessing merupakan tahapan dari proses awal terhadap teks untuk mempersiapkan teks menjadi data yang akan diolah lebih lanjut. Sebuah teks yang ada harus dipisahkan, hal ini dapat dilakukan dalam beberapa tingkatan yang berbeda. Suatu dokumen dapat dipecah menjadi bab, sub-bab, paragraf, kalimat dan pada akhirnya menjadi potongan kata/*token.* Selain itu pada tahapan ini keberadaan digit angka, huruf kapital, atau karakter- karakter yang lainnya dihilangkan dan dirubah. [4]

Berikut ini proses penjelasan tahapan *text preprocessing* :

1. *Case Folding*

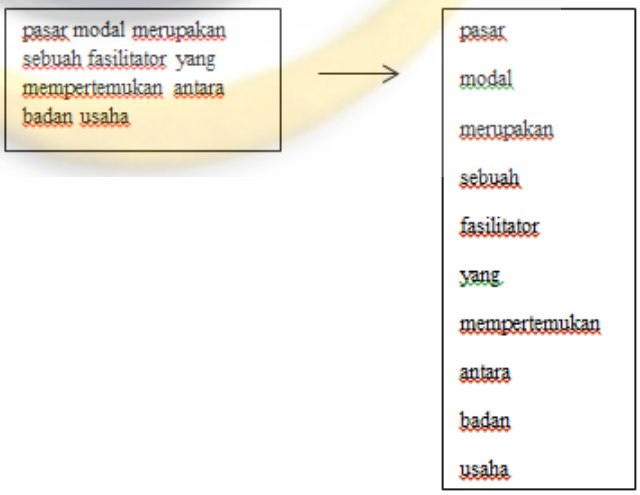
   Pada penjelasan diatas, telah dijelaskan bahwa pada tahapan *text preprocessing* terdapat beberapa hal yang dirubah, semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai dengan ‘z’ yang diterima. Karakter selain huruf dihilangkan dan dianggap delimiter. [4]



Gambar 2.1 Proses *Case Folding*

2. *Tokenizing*

   Tahap tokenizing adalah tahap pemotongan string input berdasarkan pada tiap kata yang menyusunnya. [3]



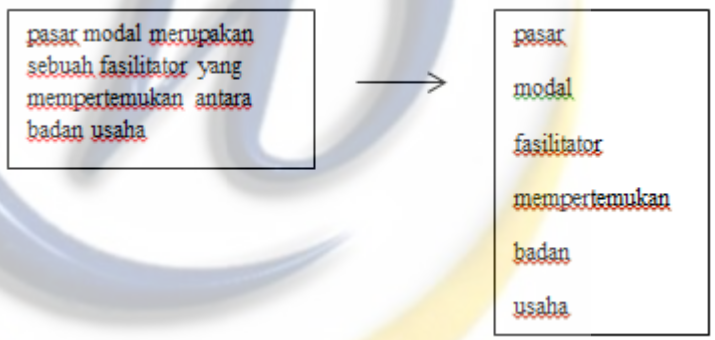
Gambar 2.2 Proses *Tokenizing*

II.1.2 *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 dengan *stemming* dan penghapusan *stopwords.*[3]

1. *Stopword Removal / Filtering*

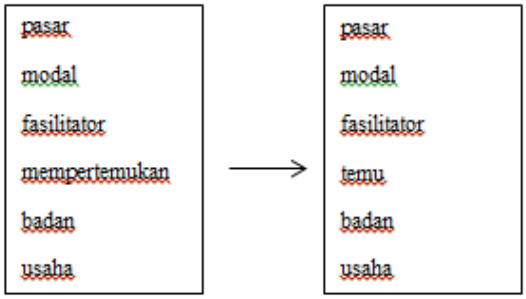
   Tahap *filtering* adalah tahap mengambil kata - kata penting dari hasil token. Bisa menggunakan algoritma stoplist (membuang kata yang kurang penting) atau wordlist (menyimpan kata penting). [3]



Gambar 2.3 Proses *Filtering*

2. *Stemming*

   Tahap *stemming adalah* tahap mencari root kata dari tiap kata hasil filtering. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama. [3]



Gambar 2.4 Proses *Stemming*

II.2 *Stemming* Bahasa Indonesia

Stemming dapat dikatakan proses membentuk suatu kata menjadi kata dasarnya. Misalnya :

berkata ⟶ kata

mengatakan ⟶ kata

perkataan ⟶ kata

Untuk bahasa Indonesia beberapa algoritma yang biasanya digunakan antara lain :

a. *Porter Stemmer*. Algoritma ini terkenal digunakan sebagai stemmer untuk bahasa Inggris. Porter Stemmer dalam bahasa Indonesia akan menghasilkan kambiguan karena aturan morfologi bahasa Indonesia.[5]

b. *Nazief & Adriani Stemmer*. Algoritma ini paling sering dibicarakan dalam *stemming* bahasa indonesia. Algoritma ini merupakan hasil penelitian internal UI (Universitas Indonesia) dan tidak dipublish secara umum [6]. Namu algoritma ini mempunyai dua masalah yang pertama kemampuanya tergantung dari besarnya database kata dasar, dan yang kedua, hasil *stemming* tidak selalu optimal untuk aplikasi *information retrieval*.[5]

Proses stemming dokumen teks menggunakan Algoritma Porter membutuhkan waktu yang lebih singkat dan presentase keakuratan yang lebih kecil dibandingkan dengan Algoritma Nazief & Adriani.

II.2.1 *Stemming* Bahasa Indonesia Algoritma Nazief & Andriani

Adapun langkah-langkah yang digunakan oleh algoritma Nazief dan Adriani yaitu sebagai berikut :

1. Kata dicari di dalam daftar kamus. Bila kata tersebut ditemukan di dalam kamus maka dapat diasumsikan kata tersebut adalah kata dasar sehingga algoritma dihentikan.
2. Bila kata di dalam langkah pertama tidak ditemukan di dalam kamus, maka diperiksa apakah surfiks tersebut yaitu sebuah partikel ("-lah" atau "-kah"). Bila ditemukan maka partikel tersebut dihilangkan.
3. Pemerikasaan dilanjutkan pada kata ganti milik ("-ku","-mu","-nya"). Bila ditemukan maka kata ganti tersebut dihilangkan.
4. Memerikas akhiran ("-i", "-an"). Bila ditemukan maka akhiran tersebut dihilangkan.

   Hingga langkah ke-4 dibutuhkan ketelitian untuk memeriksa apakah akhiran "-an" merupakan hanya bagian dari akgiran "-kan" dan memerikas lagi apakah partikel ("-lah", "-kah") dan kata ganti milik ("-ku","-mu","-nya") yang telah dihilangkan pada langkah 2 dan 3 bukan merupakan bagian dari kata dasar.
5. Memeriksa awalan ("se-","ke-","di-","te-","be-","pe-","me-"). Bila ditemukan, maka awalan tersebut dihilangkan. Pemerikasaan dilakukan dengan berulang mengingat adanya kemungkinan *multi-prefix*.

   Langkah ke-5 ini juga membutuhkan ketelitian untuk memeriksa kemungkinan peluluhan awalan, perubahan *prefix* yang disesuaikan dengan huruf awal kata dan aturan kombinasi prefix-suffix yang diperbolehkan.

6. Setelah menyelesaikan semua langkah dengan sukses, maka algoritma akan mengembalikan kata dasar yang ditemukan.

II.3 *K-nearest Neighbor Classifier*

Algoritma *k-nearest Neighbor* (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN termasuk algoritma supervised learning dimana hasil dari query instance yang baru diklasifikan berdasarkan mayoritas dari kategori pada KNN. Kelas yang paling banyak muncul yang akan menjadi kelas hasil klasifikasi[7].

Tujuan dari algoritma ini adalah mengklasifikasikan objek baru berdasarkan atribut dan training sample. Classifier tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik query, akan ditemukan sejumlah k obyek atau (titik training) yang paling dekat dengan titik query. Klasifikasi menggunakan voting terbanyak diantara klasifikasi dari k obyek.. algoritma k-nearest neighbor (KNN) menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari query instance yang baru[7].

Algoritma metode *K-Nearest Neighbor* (KNN) sangatlah sederhana, bekerja berdasarkan jarak terpendek dari query instance ke training sample untuk menentukan KNN-nya. Training sample diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi training sample. Sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan Distance[7].

II.3.1 *Euclidean Distance*

Jarak Euclidean paling sering digunakan menghitung jarak. Jarak euclidean berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek. yang direpresentasikan sebagai berikut :

$$\boldsymbol{dist} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{p}} (\boldsymbol{X_2} - \boldsymbol{X_1})^2}$$

Keterangan :

- *dist* = Jarak
- X_1 = Data Sample
- X_2 = Data Uji / Training
- *i* = Variable Data
- *p* = Dimensi Data

Semakin besar nilai D akan semakin jauh tingkat keserupaan antara kedua individu dan sebaliknya jika nilai D semakin kecil maka akan semakin dekat tingkat keserupaan antar individu tersebut.

Nilai k yang terbaik untuk algoritma ini tergantung pada data. Secara umum, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi semakin kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus dimana klasifikasi diprediksikan berdasarkan training data yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor[8].

II.2.2 *Minkowski Distance*

Minkowski Distance merupakan generalisasi dari *Euclidean Distance* dan dinyatakan dengan persamaan :

$$\boldsymbol{dist} = (\sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} |\boldsymbol{Pk} - \boldsymbol{Qk}|^{\boldsymbol{r}})^{\frac{\boldsymbol{1}}{\boldsymbol{r}}}$$

Keterangan :

- *dist* = Jarak
- *Qk* = Data Sample
- *Pk* = Data Uji / Training
- *k* = Variable Data
- *n* = Dimensi Data
- *r* = paramater

II.4 *Term Frequency*

Term frequency merupakan salah satu metode untuk menghitung bobot tiap *term* dalam teks. Dalam metode ini, tiap *term* diasumsikan memiliki nilai kepentingan yang sebanding dengan jumlah kemunculan *term* tersebut pada teks (Mark Hall & Lloyd Smith, 1999). Bobot sebuah *term* t pada sebuah teks dirumuskan dalam persamaan berikut :

$$W(d,t) = TF(d,t)$$

Dimana TF(d,t) adalah *term frequency* dari *term* t di teks d. *Term frequency* dapat memperbaiki nilai *recall* pada *information retrieval*, tetapi tidak selalu memperbaiki nilai *precision*. Hal ini disebabkan *term* yang *frequent* cenderung muncul di banyak teks, sehingga *term-term* tersebut memiliki kekuatan diskriminatif/keunikan yang kecil. Untuk memperbaiki permasalahan ini, *term*

dengan nilai frekuensi yang tinggi sebaiknya dibuang dari *set term*. Menemukan *threshold* yang optimal merupakan fokus dari metode ini.

II.5 *Document Frequency*

Document frequency adalah jumlah dokumen yang mengandung suatu *term* tertentu. Tiap *term* akan dihitung nilai *document frequency*-nya (DF). Lalu *term* tersebut diseleksi berdasarkan jumlah nilai DF. Jika nilai DF berada di bawah *threshold* yang telah ditentukan, maka *term* akan dibuang. Asumsi awalnya adalah bahwa *term* yang lebih jarang muncul tidak memiliki pengaruh besar dalam proses pengelompokkan dokumen. Pembuangan *term* yang jarang ini dapat mengurangi dimensi fitur yang besar pada *text mining*. Perbaikan dalam pengelompokkan dokumen ini juga dapat terjadi jika *term* yang dibuang tersebut juga merupakan *noise term*. *Document frequency* merupakan metode *feature selection* yang paling sederhana dengan waktu komputasi yang rendah.[10]