# BAB II

# LANDASAN TEORI

## 2.1 Text Classification dan Text Pre-processing

*Text Classification* merupakan pemberian label dengan kategori yang sudah ditentukan pada sebuah teks dengan bahasa alami (Sebastiani, 2001). Klasifikasi teks diterapkan dalam berbagai konteks, mulai dari pengindeksan sebuah dokumen berdasarkan kosa kata, pemfilteran sebuah dokumen, pembuatan *metadata* otomatis, dan pada macam aplikasi lainnya (Sebastiani, 2001). Ada beberapa cara umum dalam klasifikasi teks secara otomatis, yaitu *pre-processing*, *feature extraction/selection, modeling* menggunakan teknik pembelajaran mesin, serta *training* dan *testing* pada *classifier* (Dalal & Zaveri, 2011).

*Text Processing* merupakan salah satu teknik dari *text mining*. Text Processing dilakukan untuk mengubah data tekstual yang tidak terstruktur menjadi daya yang terstruktur lalu disimpan kedalam basis data (Langgeni, Baizal, & A.W., 2010).

### 2.1.1 Case Folding

*Case Folding* merupakan salah satu bentuk teknik text preprocessing. Tujuan proses ini adalah mengubah semua huruf dalam dokumen menjadi huruf kecil (Rahman, Wiranto, & Doewes, 2017). Pada proses ini juga dilakukan penghilangan tanda baca, angka dan karakter lain selain huruf alphabet. Hal ini dikarenakan karakter-karakter tersebut dianggap sebagai pemisah kata atau delimeter dan tidak memiliki pengaruh terhadap pemrosesan suatu teks (Hudin, 2018). Lalu pada tahap ini juga akan dilakukan penghapusan spasi di awal dan akhir, teknik ini biasa

disebut *whitespace removal* (Hudin, 2018). Tabel 2.1 menunjukkan contoh dari proses *case folding*.

Tabel 2.1 Contoh Proses *Case Folding*.

| Kalimat Awal Sebelum Case Folding | Hasil Kalimat Case Folding |
|---|---|
| Berikut ini adalah 5 negara dengan pendidikan terbaik di dunia adalah Korea Selatan, Jepang, Singapura Hong Kong, dan Finlandia. | berikut ini adalah negara dengan pendidikan terbaik di dunia adalah korea selatan jepang singapura hong kong dan finlandia |

### 2.1.2 Tokenisasi

Secara garis besar tokenisasi adalah tahap memecah sekumpulan karakter dalam suatu teks kedalam satuan kata (MA'ARIF, 2015). Menurut (Hudin, 2018) Tokenisasi merupakan proses pemotongan string input berdasarkan tiap kata penyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen (Asian, Williams, & Tahaghoghi, 2004). Tabel 2.2 menunjukkan contoh dari proses Tokenisasi.

Tabel 2.2 Contoh Proses Tokenisasi.

| Kalimat Awal Hasil Case Folding | Hasil Kalimat Tokenisasi |
|---|---|
| berikut ini adalah negara dengan pendidikan terbaik di dunia adalah korea selatan jepang singapura hong kong dan finlandia | berikut |
| | ini |
| | adalah |
| | negara |
| | dengan |

| | |
|---|---|
| berikut ini adalah negara dengan pendidikan terbaik di dunia adalah korea selatan jepang singapura hong kong dan finlandia | pendidikan |
| | terbaik |
| | di |
| | dunia |
| | adalah |
| | korea |
| | selatan |
| | jepang |
| | singapura |
| | hong |
| | kong |
| | dan |
| | finlandia |

### 2.1.3 Filtering

*Filtering* merupakan proses pemilihan kata-kata penting dari hasil tokenisasi, yaitu kata-kata yang bisa digunakan untuk mewakili isi dari sebuah teks atau dokumen. Proses filtering juga biasa disebut sebagai stopword removal. Pada proses ini terdapat dua teknik, yaitu stop list dan word list. Stop list merupakan proses membuang kata yang tidak deskriptif atau tidak penting. Sedangkan word list merupakan proses menyimpan kata yang dianggap penting (Hudin, 2018). Tabel 2.3 menunjukkan contoh proses *Filtering*.

Tabel 2.3 Contoh Proses *Filtering*.

| Kalimat Awal Hasil Tokenisasi | Hasil Kalimat Filtering |
|---|---|
| berikut | berikut |
| ini | - |
| adalah | - |
| negara | negara |
| dengan | - |
| pendidikan | pendidikan |
| terbaik | terbaik |
| di | - |
| dunia | dunia |
| adalah | - |
| korea | korea |
| selatan | selatan |
| jepang | jepang |
| singapura | singapura |
| hong | hong |
| kong | kong |
| dan | - |
| finlandia | finlandia |

### 2.1.4 Stemming

*Stemming* merupakan proses pengubahan bentuk kata menjadi kata dasar atau sebuah proses mencari akar kata dari setiap kata hasil filtering. Dengan proses stemming ini, setiap kata yang berimbuhan akan berubah menjadi kata dasar dan dapat lebih mengoptimalkan proses text mining (Hudin, 2018). Tabel 2.4 menunjukkan contoh proses *Stemming*.

Tabel 2.4 Contoh Proses *Stemming.*

| Kalimat Awal Sebelum Stemming | Hasil Kalimat Stemming |
|---|---|
| berpergian | pergi |
| pendidikan | didik |
| memakan | makan |
| memproses | proses |
| memetakan | peta |

Salah satu *library* yang dapat digunakan dalam melakukan proses *stemming* bahasa Indonesia adalah menggunakan *Library Python* Sastrawi. *Library* ini menerapkan algoritma Algoritma Nazief dan Adriani . Kemudian pada algoritma tersebut memiliki tahapan sebagai berikut (I, 2017).

1. Memeriksa kata tersebut dalam kamus. Jika ditemukan, maka diasumsikan bahwa kata tersebut merupakan akar kata, lalu algoritma berhenti.
2. Menghilangkan *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Jika kata merupakan *particles* (“-lah”, “-kah”, “-tah” atau “-pun”), maka langkah ini diulangi lagi dengan menghilangkan *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”).

3. Menghilangkan *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka lanjut ke langkah 3a.
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut di hapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak maka lanjut ke langkah 3b.
    b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan dan lanjut ke langkah ke 4.
4. Menghilangkan *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka lanjut ke langkah 4a, jika tidak maka lanjut ke langkah 4b.
    a. Periksa tabel kombinasi awalan – akhiran yang tidak diizinkan. Jika ditemukan maka algoritma berhenti, jika tidak maka lanjut ke langkah 4b. Tabel 2.5 menunjukkan tabel kombinasi awalan-akhiran yang tidak diizinkan.

Tabel 2.5 Kombinasi awalan-akhiran yang tidak diizinkan

| Awalan | Akhiran |
|---|---|
| be- | -i |
| di- | -an |
| ke- | -i, -kan |
| me- | -an |
| te- | -i, -kan |
| se- | -an |

b. Menentukan tipe awalan kemudian hapus awalan. Jika akar kata belum juga ditemukan, maka lanjut ke langkah 5. Jika sudah, maka algoritma berhenti jika awalan kedua sama dengan awalan pertama algoritma.

5. Melakukan *Recoding* (penyusunan kembali kata-kata yang mengalami proses *Stemming* berlebih).
6. Jika semua langkah telah selesai tetapi tidak juga berhasil, maka kata awal diasumsikan sebagai akar kata.

## 2.2 Term Frequency Inverse Document Frequency

Metode Term Frequency-Inverse Document Frequency (TF-IDF) merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada informasi retrieval. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat (MA'ARIF, 2015).

Metode TF-IDF adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. TF-IDF ini merupakan pengukuran statistik yang digunakan untuk mengevaluasi seberapa penting sebuah kata didalam sebuah dokumen atau dalam sekelompok kata. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Frekuensi kemunculan kata didalam dokumen yang diberikan menunjukkan seberapa penting kata itu didalam 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 (Putra, 2016).

Pada algoritma TF-IDF digunakan rumus untuk menghitung bobot (W) masing–masing dokumen terhadap kata kunci dengan rumus yaitu:

$$W_{dt} = tf_{dt} \times idf_t = tf_{dt} \times \log\left(\frac{N+1}{df_t+1}\right) \quad (2.1)$$

Di mana penjelasan variable sebagai berikut:

$t$ = Suatu kata (*word*)

$d$ = Suatu dokumen (*document*)

$W_{dt}$ = Bobot dokumen ke-d terhadap kata ke-t

$tf_{dt}$ = Banyaknya kata yang dicari pada sebuah dokumen

$idf_t$ = *Inversed Document Frequency* ($\log\left(\frac{N}{df_t}\right)$)

$N$ = Total dokumen

$df_t$ = Banyak dokumen yang mengandung kata yang dicari

## 2.3 Multinomial Naïve Bayes Classifier

Multinomial Naïve Bayes Classifier merupakan model pengembangan dari algoritma bayes yang cocok dalam pengklasifikasian teks atau dokumen. Pada formula Multinomial Naive Bayes Classifier, kelas dokumen tidak hanya ditentukan dengan kata yang muncul tetapi juga jumlah kemunculannya (Azuaje, Witten, & E., 2006).

Model Multinomial Naïve Bayes Classifier akan menghitung frekuensi setiap kata yang muncul pada dokumen. Untuk menentukkan persamaan formula Multinomial Naïve Bayes Classifier perlu diketahui rumus dari Naïve Bayes Classifier. Berikut rumus Naïve Bayes Classifier:

$$P(c|x) = \frac{P(x|c)\,P(c)}{\mathrm{P}(x)} \tag{2.2}$$

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \ldots \times P(x_n|c) \times P(c) \tag{2.3}$$

Di mana penjelasan variable sebagai berikut:

$P(c|x)$ = Probabilitas kelas (target) yang diberikan prediktor (atribut)

$P(x|c)$ = Probabilitas yang kemungkinan kelas prediktor (atribut) tertentu

$P(c)$ = Probabilitas *prior* dari kelas (target)

$P(x)$ = Probabilitas *prior* dari prediktor (atribut)

$P(x_n|c)$ = Probabilitas kata ke-n yang diketahui kelas c (target)

Probabilitas *prior* kelas c (target) ditentukan dengan rumus:

$$P(c) = \frac{N_c}{N} \tag{2.4}$$

Di mana penjelasan variable sebagai berikut:

$N_c$ = Jumlah seluruh dokumen dengan kelas c

$N$ = Jumlah seluruh dokumen

Probabilitas kata ke-n ($x_n$) untuk sebuah kelas c, $P(x_n|c)$, akan ditentukan menggunakan teknik *laplacian smoothing*:

$$P(x_n|c) = \frac{count\ (\mathrm{x_n},\mathrm{c})+1}{count(\mathrm{c})+|\mathrm{V}|} \tag{2.5}$$

Di mana penjelasan variable sebagai berikut:

$count(x_n, c)$ = Jumlah term $x_n$ yang ditemukan pada seluruh data latih dengan kategori c

$count(c)$ = Jumlah term diseluruh data latih dengan kategori c

V = seluruh term pada data latih

Berdasarkan rumus Naïve Bayes Classifier (persamaan 2.3), untuk mengetahui kelas dari sebuah dokumen d, maka dapat digunakan persamaan sebagai berikut:

$$MNBC(d) = argmax_{c \in C} P(c) \prod_{1<k<n_d} P(x_k|c) \tag{2.6}$$

Di mana penjelasan variable sebagai berikut:

*C* = himpunan seluruh kemungkinan kelas yang dimiliki oleh dokumen *d*

$n_d$ = jumlah seluruh kata unik yang muncul dalam dokumen *d*.

Akan tetapi, saat diimplementasikan ke dalam komputer, persamaan Multinomial Naïve Bayes Classifier (persamaan 2.6) dapat menyebabkan kondisi *floating point underflow,* lalu persamaan dituliskan kembali menjadi persamaan berikut (Manning, Raghavan, & Schutze, 2008):

$$MNBC(d) = argmax_{c \in C} [\log P(c) + \sum_{1<k<n_d} \log P(x_k|c)] \tag{2.7}$$

## 2.4 Complement Naïve Bayes Classifier

Algoritma Complement Naive Bayes Classifier merupakan algoritma yang akan melengkapi Algoritma Multinomial Naïve Bayes Classifier di mana mencari atau menghitung data latih dari seluruh kelas kecuali kelas c yang digunakan (Rennie, Shih, Teevan, & Krager, 2003). Berikut formula Algoritma Complement Naïve Bayes Classifier :

$$CNBC(d) = argmax_{c \in C}[\log P(c) - \sum_{1<k<n_d} \log P(x_k|`c)] \quad (2.8)$$

$$P(x_k|`c) = \frac{count\ (\mathrm{x_n}`,\mathrm{c})+1}{count(`\mathrm{c})+|\mathrm{V}|} \quad (2.9)$$

Di mana penjelasan variable (persamaan 6.4.2) sebagai berikut:

$count\ (\mathrm{x_n}`,\mathrm{c})$ = jumlah kata $x_n$ yang muncul pada dokumen bukan kelas c

$count(`\mathrm{c})$ = jumlah seluruh kata yang muncul pada dokumen bukan kelas c

## 2.5 Evaluasi Klasifikasi

Untuk mengetahui tingkat model gabungan dari berbagai metode seleksi fitur. Terdapat beberapa cara untuk mengukur performansi metode klasifikasi diantaranya dengan menggunakan *Confusion Matrix, Precision*, *Recall,* dan *F1-Score.*

*Confusion Matrix* merupakan sebuah tabel yang terdiri atas banyaknya baris data uji yang diprediksi benar dan tidak benar oleh model klasifikasi, digunakan untuk menentukan kinerja suatu model klasifikasi (Hamel, 2008). Tabel *Confusion Matrix* dapat membantu dalam memperoleh nilai dari perhitungan tersebut karena berisi data prediksi yang positif dan negatif yang dihasilkan oleh sistem, dan data aktual yang positif dan negatif di dunia nyata (Goutte & Gaussier, 2005). Berikut bentuk tabel *Confusion Matrix*:

Tabel 2.6 *Confusion Matrix Table*

| | **Positif (Aktual)** | **Negatif (Aktual)** |
|---|---|---|
| **Positif (Sistem)** | TP | FP |
| **Negatif (Sistem)** | FN | TN |

Di mana penjelasan *variable* sebagai berikut:

*True Positive* = perhitungan dari kelas positif sistem dan aktual yang positif.

*True Negative* = perhitungan dari kelas negatif sistem dan aktual yang negatif.

*False Positive* = perhitungan dari kelas positif sistem dan aktual yang negatif.

*False Negative* = perhitungan dari kelas negatif sistem dan aktual yang positif.

*Precision* merupakan rasio kategorisasi dokumen yang benar ke dalam kategori dengan jumlah total percobaan klasifikasi (Forman, 2003).

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{2.10}$$

*Recall* merupakan rasio klasifikasi dokumen yang benar ke dalam kategori dengan jumlah total data berlabel di set pengujian (Forman, 2003).

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{2.11}$$

Untuk menghindari perbedaan rasio yang cukup tinggi antara nilai *precision* dan *recall*. Maka, dilakukan penyetaraan nilai menggunakan *F1-Score* (Ponweiser, 2012). *F1-Score* akan menilai ketepatan sebuah classifier dengan rata-rata dari nilai *precision* dan *recall* (Singh, Tiwari, & Singh, 2018).

$$F1\text{-}Score = 2 \times \frac{Precission \times Recall}{Precission + Recall} \tag{2.12}$$