# BAB II
# LANDASAN TEORI

## 2.1 *Text Mining*

*Text mining* adalah penemuan kembali informasi yang tersimpan dalam bentuk teks. *Text mining* melibatkan bidang ilmu *information retrieval* (IR), *text analysis*, *information extraction* (IE), *clustering*, *categorization*, *visualization*, *database technology*, *natural language processing* (NLP), *machine learning*, *artificial intelligence* dan *data mining* (Hearst, 2003).

*Text mining* pada umumnya mengacu pada proses ekstraksi informasi dari dokumen teks tidak terstruktur. *Text mining* dapat didefinisikan sebagai penemuan informasi baru dan tidak diketahui sebelumnya oleh komputer, dengan secara otomatis mengekstrak informasi dari sumber-sumber teks tak terstruktur yang berbeda. Hal utama dari proses ini adalah menggabungkan informasi yang berhasil diekstraksi dari berbagai sumber (Tan, 1999).

*Text mining* merupakan salah satu bidang khusus dari *data mining*. Pada buku "*The Text Mining Handbook"*, *text mining* dapat diartikan sebagai proses penggalian informasi dari sekumpulan dokumen dengan menggunakan perangkat analisis yang merupakan komponen dalam *data mining* yang salah satunya ialah kategorisasi.

*Text mining* dan *data mining* mempunyai perbedaan mendasar yang terletak pada sumber data yang digunakan. Pada *text mining*, pola-pola diekstrak dari data tekstual (*natural language*). Secara umum, basis data didesain untuk program dengan tujuan melakukan pemrosesan secara otomatis, sedangkan teks ditulis untuk dibaca langsung oleh manusia (Hearst, 2003), sedangkan di *data mining*, pola-pola diekstrak dari basis data yang terstruktur. Ada beberapa teknik yang dimiliki *data mining* berdasarkan tugas yang bisa dilakukan (Kusrini, 2009), yaitu: asosiasi, deskripsi, klasifikasi, estimasi, prediksi, *clustering.* Pada penelitian ini menggunakan teknik klasifikasi.

## 2.2 Lingkungan *Text Mining*

Ada empat tahap proses pokok dalam *text mining*, yaitu (Foltz, M, & Y, 2003) :

1. *Text preprocessing*
2. *Text transformation*
3. *Feature selection*
4. *Pattern discovery*

### 2.2.1 *Text Preprocessing*

Pada tahap ini akan dilakukan proses pembersihan teks atau menghilangkan karakter-karakter yang tidak relevan seperti: tanda baca, spasi,angka dan karakter lainnya. Hal tersebut dilakukan untuk mengubahnya menjadi data berkualitas yaitu data yang telah memenuhi persyaratan untuk dieksekusi pada sebuah algoritma.

*Text preprocessing* diterapkan pada algoritma untuk mendeteksi kemiripan isi dokumen, salah satunya merupakan algoritma *winnowing*. Hal yang dilakukan pada tahap *text preprocessing* pada *winnowing* adalah (Sanjaya & Absar, 2015) :

1. *Case Folding*, mengubah huruf besar menjadi huruf kecil.
2. Filtering, yaitu pengambilan kata penting. Penelitian ini menggunakan stopwords dengan menghapus kata penghubung yang terdapat pada penelitian (Zulfah et al., 2014).

### 2.2.2 *Text Transformation*

*Winnowing* merupakan algoritma yang digunakan untuk mendeteksi kemiripan dokumen hingga bagian kecil yang mirip dalam dokumen dengan menggunakan teknik *hash* (Elbegbayan, 2005). Input dari algoritma ini adalah dokumen teks (plaintext). Output dari algoritma ini berupa kumpulan nilai *hash*. Teknik *hash* pada dokumen menggunakan *k-gram*. Kumpulan nilai *hash* tersebut selanjutnya disebut *fingerprint*. *Fingerprint* tersebut akan dipilih nilai yang paling kecil dengan menggunakan *window*. *Fingerprint* inilah yang dijadikan rujukan sebagai pembanding antara file-file teks yang telah dimasukkan dan digunakan dalam deteksi penjiplakan (Schleimer et al, 2003).

Berikut adalah langkah selanjutnya dalam penerapan *winnowing* :

1. *K-gram*, merupakan proses membentuk *substring* sepanjang *k* karakter dari sebuah *string*. Contoh: "Logistic Regression termasuk metode klasifikasi dokumen". Memotong *string* sepanjang *k*. misalnya nilai *k*=8, dari kalimat diatas, sehingga diperoleh hasil sebagai berikut:

   logistic ogisticr gisticre isticreg sticregr ticregre icregres cregress regressi egressio gression .....

2. *Hash*, adalah fungsi yang menerima masukan *string* yang panjangnya tidak terstruktur dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula). Keluaran fungsi *hash* disebut juga nilai *hash*. Terdapat teori dari fungsi *hash* yang dikenal sebagai *rolling hash*. *Rolling hash* adalah teknik yang digunakan untuk mendapatkan nilai *hash* yang telah terbentuk dari metode *k-gram*. *Rolling hash* berfungsi untuk mempercepat kinerja komputer untuk menghitung nilai *hash* dari rangkaian *grams* selanjutnya yang telah terbentuk.

Berikut persamaan dari metode *hash* (Schleimer et al., 2003):

$$H_{(c1....ck)} = c_1 * b_{(k-1)} + c_2 * b_{(k-2)} + ..... + c_{(k-1)} * b_k + c_k \quad \text{.....(2.1)}$$

Keterangan:

c: nilai *ascii*

b: bilangan prima

k: banyak karakter

Penggunaan *rolling hash* memiliki keuntungan yaitu untuk menghitung nilai *hash* berikutnya menjadi lebih mudah (Schleimer et al., 2003):

$$H_{(c2....ck+1)} = (H_{(c1....ck)} - c_1 * b_{(k-1)}) * b + c_{(k+1)} \quad \text{.....(2.2)}$$

Keterangan:

$H_{(c1....ck)}$ : nilai *hash* sebelumnya

C1 : nilai ASCII karakter pertama

b : bilangan prima

Untuk mendapatkan nilai *hash* dari setiap *string* dibutuhkan nilai ASCII karakter. Berikut adalah tabel nilai ASCII karakter yang dapat digunakan untuk menghitung nilai *hash*:

**Tabel 2.1 Nilai karakter-karakter ASCII**

| Alphabet | Nilai ASCII | Alphabet | Nilai ASCII |
|---|---|---|---|
| A | 65 | a | 97 |
| B | 66 | b | 98 |
| C | 67 | c | 99 |
| D | 68 | d | 100 |
| E | 69 | e | 101 |
| F | 70 | f | 102 |
| G | 71 | g | 103 |
| H | 72 | h | 104 |
| I | 73 | i | 105 |
| J | 74 | j | 106 |
| K | 75 | k | 107 |
| L | 76 | l | 108 |
| M | 77 | m | 109 |
| N | 78 | n | 110 |
| O | 79 | o | 111 |
| P | 80 | p | 112 |
| Q | 81 | q | 113 |
| R | 82 | r | 114 |
| S | 83 | s | 115 |
| T | 84 | t | 116 |
| U | 85 | u | 117 |
| V | 86 | v | 118 |
| W | 87 | w | 119 |
| X | 88 | x | 120 |
| Y | 89 | y | 121 |
| Z | 90 | z | 122 |

Dengan demikian tidak perlu melakukan iterasi dari indeks pertama sampai terakhir untuk menghitung nilai *hash* untuk *gram* ke-2 sampai terakhir. Hal ini tentu dapat mengurangi waktu komputasi saat menghitung nilai *hash* dari sebuah *gram.* Berikut ini adalah contoh pembentukan nilai *hash* menggunakan persamaan *rolling hash* dengan nilai bilangan prima=2*:*

$$H_{(logistic)} = ascii(l) * 2^{(7)} + ascii(o) * 2^{(6)} + ascii(g) * 2^{(5)} + ascii(i) * 2^{(4)} + ascii(s) * 2^{(3)} + ascii(t) * 2^{(2)} + ascii(i) * 2^{(1)} + ascii(c) * 2^{(0)}$$

$$= 108 * 128 + 111 * 64 + 103 * 32 + 105 * 16 + 115 * 8 + 116 * 4 + 105 * 2 + 99 * 1$$

$$= 13824 + 7104 + 3296 + 1680 + 920 + 464 + 210 + 99$$

$$= 27597$$

$$H_{(ogisticr)} = (27597 - \text{ascii(l)} * 2^{(7)}) * 2 + \text{ascii(r)} * 2^{(0)}$$

$$= (27597 - 108 * 128) * 2 + 114 * 1$$

$$= (27597 - 13824) * 2 + 114 * 1$$

$$= (13773 * 2) + 114$$

$$= 27546 + 114$$

$$= 27660$$

3. Membentuk *window*

Nilai-nilai *hash* yang telah diperoleh dari perhitungan dibentuk menjadi beberapa *window*. Pada contoh dibawah ini menggunakan nilai $w = 5$:

[27597 27660 **27005** 27745 28724]

[27660 **27005** 27745 28724 28109]

[**27005** 27745 28724 28109 26637]

.....

.....

4. Pemilihan beberapa nilai *hash* menjadi *fingerprint.*

Pilih nilai *hash* minimum dari setiap *window* yang telah terbentuk untuk dijadikan *fingerprint* (penanda), jika nilai tersebut berada pada posisi sama maka kedua nilai *hash* tersebut dijadikan sebagai *fingerprint*.

### 2.2.3 *Feature Selection*

*Feature selection* merupakan tahap lanjut dari pengurangan dimensi pada proses transformasi teks. *Feature selection* merupakan bagian penting untuk mengoptimalkan kinerja dari klasifikasi (Yang, Shen, Ma, Huang, & Zhou, 2011). Empat macam *feature* yang sering digunakan adalah:

1. *Character*, merupakan komponan individual, bisa huruf, angka, karakter spesial dan spasi.

2. *Words*.
3. *Terms* merupakan *single word* dan *multiword phrase* yang terpilih secara langsung dari *corpus*.
4. *Concept*, merupakan *feature* yang di-*generate* dari sebuah dokumen secara manual, *rule-based*, atau metodologi lain.

### 2.2.4 Pembobotan TF-IDF (*TERMS FREQUENCY-INVERSE DOCUMENT FREQUENCY*)

Merupakan tahap pembobotan pada kata. Dalam hal ini yang akan dihitung bobotnya adalah hasil *fingerprint* karena menurut penelitian (Jurafsky & Martin, 2015) untuk mencari bobot yaitu memilih bobot yang membuat kelas pada data latih memungkinkan,dalam penelitian ini adalah hasil *fingerprint. Fingerprint* tersebut adalah *term* atau kata yang telah di proses menjadi nilai *hash* yang kemudian dipilih nilai terkecil untuk menjadi *fingerprint*. Penelitian ini menggunakan *Term Frequency Inverse Document Frequency* (*TF-IDF*) yaitu proses pembobotan dengan menghitung *term frequency* dengan menghitung jumlah munculnya *fingerprint* di dokumen terkait dan *inverse document frequency* dari *fingerprint* dalam koleksi dokumen menggunakan persamaan berikut.(Abghari, 2013)

Rumus umum untuk TF-IDF(Intan & Defeng, 2006):

$$W_{ij} = tf_{ij} \text{ x } \boldsymbol{log} \frac{N}{\text{n}} \quad \text{..... (2.3)}$$

Keterangan :

$W_{ij}$ : bobot *term*/*fingerprint* terhadap dokumen $d_i$

$tf_{ij}$ : jumlah kemunculan *term/fingerprint* dalam $d_i$

**N** : jumlah semua dokumen yang ada dalam *database*

n : jumlah dokuen yang menganduk *term/fingerprint* (minimal ada satu *fingerprint*)

Beberapapun besarnya nilai n, apabila N=n, maka akan didapatkan hasil 0 (nol),dikarenakan hasil dari log 1, untuk itu dapat ditambahkan nilai 1 pada sisi IDF, sehingga perhitungan bobotnya menjadi seperti berikut:

$$W_{ij} = tf_{ij} \text{ x } (\boldsymbol{log}\left(\frac{\mathrm{N}}{\mathrm{n}}\right) + 1) \quad \text{..... (2.4)}$$

### 2.2.5 *Pattern Discovery*

Tahap ini dilakukan pada proses *data mining* sehingga mendapatkan pengetahuan baru pada teks.

#### 2.2.5.1 Teknik Klasifikasi

Klasifikasi merupakan teknik *data mining* yang digunakan dalam pencarian sekumpulan model yang dapat menjelaskan dan membedakan kelas data atau konsep, yang bertujuan agar model tersebut dapat digunakan memprediksi objek kelas yang labelnya tidak diketahui atau dalam arti lain dapat memprediksi kecenderungan data-data yang akan muncul di masa depan (Han et al, 2006).

#### 2.2.5.2 *Multinomial Logistic Regression*

*Logistic Regression* termasuk ke dalam *family of classifiers*. *Logistic Regression* dikenal sebagai pengklasifikasi ekponensial atau *log-linier*. Secara teknis, *Logistic Regression* mengacu pada klasifikasi yang mengklasifikasikan observasi menjadi salah satu dari dua kelas. *Multinomial Logistic Regression* digunakan saat mengelompokkan ke dalam lebih dari dua kelas (Jurafsky & Martin, 2015).

Perbedaan mendasar antara *Logistic Regression dan Naive Bayes* adalah *Logistic Regression* merupakan *discriminative classifier*, sedangkan *Naive Bayes* merupakan *generative classifier* (Jurafsky & Martin, 2015).

(Jurafsky & Martin, 2015) berpendapat bahwa *denominator* dan *exp* dapat diabaikan jika tujuan dari penggunaan *Multinomial Logistic Regression* hanya untuk klasifikasi. Kita hanya akan memilih kelas dengan titik produk tertinggi antara bobot dan fitur. Persamaan *Multinomial Logistic Regression* untuk klasifikasi teks adalah (Jurafsky & Martin, 2015):

$$C = P(c|x)$$

$$= \frac{\exp(\sum_{i=1}^{N} wifi\ (c,x)}{\sum_{c1\ \epsilon\ c} \exp\ (\sum_{i=1}^{N} wifi\ (c1,x)}$$

$$= \exp \sum_{i=1}^{n} wifi\ (c,x)$$

$$= \sum_{i=1}^{n} wifi\ (c,x) \quad \text{.....(2.5)}$$

Keterangan :

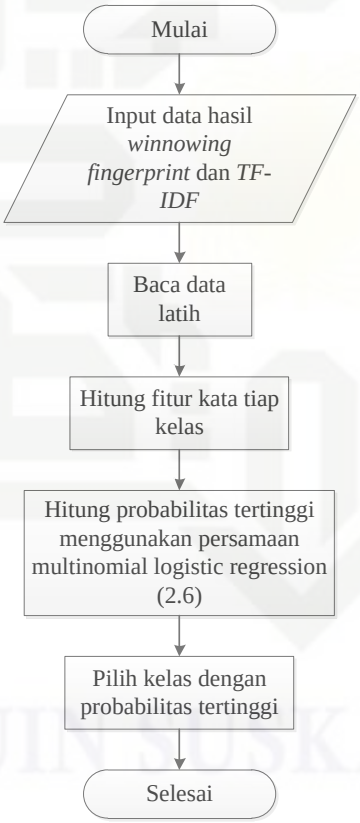
$w$ : bobot

$f$ : fitur

$c$ : kelas

$x$ : observasi

Berikut merupakan alur *Multinomial Logistic Regression*, yang dapat dilihat pada gambar 2.1.




**Gambar 2.1 Alur *Multinomial Logistic Regression***

## 2.3 Metode Kemiripan Dokumen

Terdapat tiga (3) metode yang dapat dilakukan untuk pendeteksian kesamaan dokumen (Kurniawati & Wicaksana, 2008) yaitu:

1. Perbandingan Teks Lengkap

   Metode ini diterapkan dengan membandingkan semua isi dokumen. Metode ini membutuhkan waktu yang lama tetapi cukup efektif. Algoritma yang digunakan adalah algoritma *Brute Force, Edit Distance, Boyer Moore*.

2. Dokumen *Fingerprint*

   Merupakan metode yang digunakan untuk mendeteksi keakuratan kesamaan antar dokumen. Prinsip kerja dari metode dokumen *fingerprint* ini dengan menggunakan teknik *hashing*. Teknik *hashing* adalah sebuah fungsi yang menkonversi setiap *string* menjadi bilangan.

3. Kesamaan Kata Kunci

   Prinsip dari metode kesamaan kata kunci adalah mencari kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen lain.

Sedangkan untuk melakukan pendeteksian kemiripan dokumen teks sebuah algoritma harus memenuhi salah satu persyaratan (Schleimer et al., 2003):

1. *Whitespace Insensitivity*

   Melakukan pencocokan terhadap dokumen teks seharusnya tidak terpengaruh oleh spasi, jenis huruf (kapital atau normal), tanda baca, simbol-simbol dan sebagainya.

2. *Noise Surpression*

   Menghindari penemuan kecocokan dengan panjang kata yang terlalu kecil atau kurang relevan.

3. *Position Independence*

   Penemuan kecocokan atau kesamaan tidak harus bergantung pada posisi kata-kata.

Pada penelitian ini menggunakan metode dokumen *fingerprint* untuk mendeteksi kemiripan dokumen dan persyaratan yang dipakai adalah *whitespace insensitivity*.

## 2.4 *Confusion Matrix*

Percobaan dari penelitian dievaluasi dengan pengukuran akurasi *Confusion Matrix* (Xhemali, et al., 2009).

**Tabel 2.2 Confusion Matrix** (Xhemali, et al., 2009)

| | | Predicted | |
|---|---|---|---|
| | | Irrelevant | Relevant |
| Actual | Irrelevant | TP | FN |
| | Relevant | FP | TN |

Keterangan:

*TP* (*True Positive*) : Jumlah prediksi yang benar dari data yang sakit.

*FP* (*False Positive*) : Jumlah prediksi yang salah dari data yang tidak sakit.

*FN* (*False Negative*) : Jumlah prediksi yang salah dari data yang sakit.

*TN* (*True Negative*) : Jumlah prediksi yang benar dari data yang tidak sakit.

$$\text{Akurasi} = \frac{(TP+TN)}{TP+FP+TN+FN} \quad (2.6)$$

Menghitung tingkat akurasi, dengan menghitung jumlah prediksi benar dan salah dari sebuah metode klasifikasi berbanding dengan data sesungguhnya atau prediksi target. Menggunakan *matrix* (NxN), yang dimaksud N adalah jumlah kelas. Berikut ini adalah contoh perhitungan akurasi dengan *confusion matrix* menggunakan 5 kelas:

**Tabel 2.3 Contoh Perhitungan *Confusion Matrix*** (Radili, 2016)

| | Teknologi | Berita | Hiburan | Olahraga | Lain-lain |
|---|---|---|---|---|---|
| Teknologi | A | B | C | D | E |
| Berita | F | G | H | I | J |
| Hiburan | K | L | M | N | O |
| Olahraga | P | Q | R | S | T |
| Lain-lain | U | V | W | X | Y |

Keterangan :

Kolom adalah prediksi kelas dan baris adalah kelas aktual

$$\text{Akurasi} = \frac{(A+G+M+S+Y)}{(A+B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R+S+T+U+V+W+X+Y)} \quad (2.7)$$