BAB II
TINJAUAN PUSTAKA

2.1 *Text Mining*

Proses mengekstrak data dengan kualitas tinggi yang berasal dari teks dikenal sebagai *text mining*. Dengan memahami pola statistik, biasanya dimungkinkan untuk mendapatkan data berkualitas tinggi dengan memperhatikan pola serta tren data. Pembobotan kata merupakan aspek dari *text mining* yang bermaksud untuk memberikan bobot atau nilai pada istilah dalam dokumen (Deolika dkk., 2019). Metode yang dipakai menentukan besarnya bobot atau nilai pada istilah dalam dokumen.

2.2 Analisis Sentimen

Studi tentang komentar, perasaan, evaluasi, sikap, penilaian, dan perasaan orang terhadap hal-hal seperti organisasi, layanan, produk, orang, isu, kejadian, dan topik dikenal sebagai analisis sentimen (Bhatia dkk., 2017). Analisis sentimen merupakan studi komputasi dari opini, sentimen, serta emosi yang diekspresikan dalam teks. Untuk menentukan apakah opini yang diungkapkan dalam dokumen atau kalimat bersentimen positif, netral, atau negatif maka digunakan analisis sentimen yang bermaksud untuk mengklasifikasikan polaritas teks. Mengklasifikasikan polaritas teks dalam opini, kalimat, atau dokumen adalah tujuan mendasar dari analisis sentimen. (Amrullah dkk., 2020).

2.3 Algoritma *Multinomial Naïve Bayes*

Multinomial naïve bayes merupakan sebuah metode klasifikasi yang berakar pada teorema Bayes. Teorema Bayes dikemukakan oleh ilmuwan Inggris Thomas Bayes pada abad ke-18, yaitu memprediksi peluang dimasa depan berdasarkan pengalaman dimasa sebelumnya. Ciri utama dari klasifikasi *Naïve Bayes* adalah asumsi independensi yang kuat (Rinaldi dkk., 2021).

Secara umum, perhitungan teorema Bayes dapat dilihat pada persamaan berikut ini:

$$P(A|B) = \frac{P(A)\ P(B|A)}{P(B)} \tag{2.1}$$

Keterangan:

$P(A\|B)$	=	kemungkinan A akan terjadi jika B telah terjadi.
$P(A)$	=	kemungkinan terjadinya A.
$P(B\|A)$	=	kemungkinan B akan terjadi jika A telah terjadi.
$P(B)$	=	peluang terjadinya B.

Ide di balik algoritma ini adalah *Term Frequency* (TF), yang mengacu pada berapa kali sebuah kata muncul dalam dokumen. Frekuensi kemunculan kata dalam dokumen dan kehadirannya dalam dokumen adalah dua fakta yang dijelaskan oleh model ini. Berikut ini adalah cara untuk menghitung *multinomial naive bayes* (Singh dkk., 2019).

$$P(p|d) \propto P(p) \prod\nolimits_{1 \le k \le n} P(t_k|p) \tag{2.2}$$

Keterangan:

p	=	polaritas.
d	=	dokumen.
t_k	=	*term* ke-k yang muncul pada dokumen teks.
$P(t_k\|p)$	=	probabilitas munculnya *term* pada dokumen teks (t_k) yang memiliki polaritas p.
$P(p\|d)$	=	probabilitas munculnya dokumen d yang memiliki polaritas p.

Untuk menghitung polaritasnya atau dokumen yang mempunyai kemiripan dirumuskan sebagai berikut (Yuyun dkk., 2021).

$$P(t_k|p) = \frac{count(t_k|p) + 1}{count(t_p + |V|)} \quad (2.3)$$

Keterangan:

$(t_k\|p)$	=	jumlah t_k muncul di dokumen dengan polaritas p.
t_p	=	jumlah *term* yang terdapat pada dokumen yang memiliki polaritas p.
$\|V\|$	=	jumlah total *term* pada *vocabulary*.

2.4 *Text Preprocessing*

Langkah awal dalam mentransformasi teks menjadi data yang dapat diproses pada tahapan lebih lanjut disebut *text preprocessing*. Suatu *tweet* atau dokumen dapat dibagi menjadi bab, sub-bab, paragraf, kalimat dan kata-kata atau token. Pada titik ini keberadaan huruf kapital, digit angka, atau karakter lainnya diubah atau dihilangkan (Herdhianto, 2020).

Sebelum dokumen teks diproses pada tahap selanjutnya, dilakukan proses pemilihan data pada tiap dokumen menggunakan *text preprocessing*. (Herdhianto, 2020). Ada beberapa tahapan pada *text preprocessing*, yaitu *case folding, word normalization, cleansing, filtering*/*stopword removal, stemming* serta *tokenization.*

2.4.1 *Case Folding*

Membuat semua huruf pada dokumen menjadi huruf kecil adalah tahapan *case folding*. Karakter yang diterima adalah dari a sampai z. Karakter kecuali huruf akan dihapus dan disebut sebagai *delimiter* atau pembatas (Herdhianto, 2020). Data *tweet* hasil *scraping* akan diubah semua karakternya menjadi huruf kecil.

2.4.2 *Word Normalization*

Kata-kata tidak baku atau singkatan diubah menjadi kata-kata baku pada tahap *word normalization*. Selain itu, karakter berulang juga dihilangkan. Fungsi *regex* digunakan untuk menemukan kata yang tidak baku atau disingkat dalam teks dan menggantinya dengan pasangan kata baku dan tidak disingkat selama proses *word normalization* (Rasyadi,

2017). *Regex* merupakan sebuah metode untuk mengenali atau mendeteksi sebuah pola pencarian sehingga dapat membantu kita untuk melakukan *matching* (pencocokan), *locate* (pencarian), dan manipulasi teks.

2.4.3 *Cleansing*

Cleansing adalah tahapan untuk menghilangkan kata yang tidak digunakan (Herdhianto, 2020). Kata-kata yang dihilangkan pada tahapan ini adalah *hashtag (#),* alamat *website, username (@username),* angka, emoji serta *email.* Tahap *cleansing* perlu dilakukan untuk meningkatkan kualitas data *training*.

2.4.4 *Filtering/Stopword Removal*

Stopword removal adalah proses menghilangkan kata yang tidak penting pada dokumen dengan mengecek apakah kata tersebut masuk dalam daftar kata tidak penting *(stoplist)* atau tidak. Kata yang terdapat pada *stoplist* akan dihilangkan, sedangkan kata yang tidak dihilangkan dianggap sebagai *keywords* atau kata penting (Herdhianto, 2020). Pada penelitian ini, *stoplist* ditentukan dengan menggunakan salah satu *library* bahasa pemrograman Python yaitu Sastrawi.

2.4.5 *Stemming*

Stemming adalah proses menemukan akar kata dari sebuah kata. Cara kerja *stemming* adalah dengan menghapus imbuhan yang melekat pada suatu kata, sehingga diperoleh sebuah kata yang sesuai dengan struktur morfologi dalam Bahasa Indonesia.

2.4.6 *Tokenization*

Tokenization merupakan tahap dimana suatu kalimat dipotong dengan maksud untuk memperoleh kata-kata yang dapat digunakan pada tahapan selanjutnya (Prasidhatama & Suryaningrum, 2018). Metode tokenisasi yang digunakan adalah *unigram. Unigram* yaitu token data teks yang hanya terdiri dari satu kata.

2.5 *Feature Extraction* dengan *Term Frequency-Inverse Document Frequency* (TF-IDF)

Proses pencarian nilai fitur dalam dokumen untuk proses *text mining* dikenal dengan *feature extraction. Feature extraction* merupakan langkah penting dalam pemrosesan dokumen untuk mesin pencari karena menentukan kesuksesan proses *text mining* (Prihatini, 2016). TF-IDF adalah sebuah metode *feature extraction* yang cukup terkenal dan banyak digunakan.

2.5.1 TF-IDF

TF diperoleh berdasarkan banyaknya kemunculan kata pada setiap dokumen, dan *Inverse Document Frequency* (IDF) diperoleh berdasarkan banyaknya kemunculan kata pada seluruh dokumen (Prihatini, 2016). Setelah melewati tahap *preprocessing*, nilai TF dibandingkan dengan nilai IDF. Hasil akhirnya dalam bentuk matriks *term-document* X, dimana setiap kolom dokumen berisikan nilai TF-IDF.

Perhitungan untuk mencari nilai TF-IDF adalah sebagai berikut:

$$TF_{t,d}IDF_{t,d} = tf_{t,d} \times idf_t \tag{2.4}$$

$$idf_t = \log_e \frac{1+n}{1+df_t} + 1 \tag{2.5}$$

Keterangan:

n	=	jumlah seluruh dokumen dalam *dataset*.
t	=	sebuah *term* (t) dari *keywords*.
d	=	dokumen ke-d.
$TF_{t,d}IDF_{t,d}$	=	bobot dokumen d terhadap sebuah *term* (t).
$tf_{t,d}$	=	banyaknya kemunculan sebuah *term* (t) dalam dokumen (d).
idf_t	=	seberapa sering atau jarang sebuah *term* muncul pada seluruh dokumen.
df_t	=	banyaknya dokumen yang terdapat t.

2.6 *Feature Selection*

Seleksi fitur *(feature selection)* merupakan suatu tahapan yang berfungsi untuk mengurangi ukuran dimensi data, menghapus fitur yang tidak penting, dan meningkatkan akurasi (Herdhianto, 2020). Ada dua kegunaan utama dalam pemilihan fitur. Pertama, mengurangi ukuran fitur untuk membuat data *training* pengklasifikasi lebih efektif. Kedua, dengan menghilangkan *noise* fitur, pemilihan fitur biasanya meningkatkan akurasi klasifikasi.

Seleksi fitur berfungsi untuk mengurangi ukuran dimensi data serta bertujuan untuk memilih fitur terbaik dari suatu kumpulan data fitur. Fitur diseleksi dengan memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan.

Metode seleksi fitur yang digunakan pada penelitian ini adalah *chi square.* Perhitungan *chi square* adalah sebagai berikut.

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{2.6}$$

Keterangan:

k = banyaknya kategori, 1,2,…,k

O_i = nilai observasi untuk kategori i.

E_i = nilai ekspekstasi untuk kategori i

2.7 ADASYN

Suatu kondisi data yang dikenal sebagai *imbalanced class data* adalah kondisi di mana satu kelas data tidak seimbang dengan kelas yang lain. Kondisi tersbut akan menjadi masalah dalam klasifikasi karena *classifier learning* biasanya akan memprediksi kelas data mayoritas dibandingkan dengan kelas data minoritas. Kondisi tersebut juga akan mengakibatkan akurasi prediksi

baik untuk kelas mayoritas dan menyebabkan akurasi prediksi buruk untuk kelas minoritas (Fitriani dkk., 2021).

ADASYN merupakan metode untuk pendekatan sampling pada pembelajaran dengan dataset yang tidak seimbang (Haibo He, 2008). ADASYN meningkatkan pembelajaran dengan dua cara. Pertama, mengurangi bias yang diakibatkan oleh ketidakseimbangan kelas dan yang kedua secara adaptif menggeser batas keputusan klasifikasi terhadap kesulitan data.

Algoritma ADASYN dapat diformulasikan sebagai berikut (Haibo He, 2008).

a. Mengitung rasio kelas minoritas terhadap kelas mayoritas

$$d = \frac{m_s}{m_l} \tag{2.7}$$

Keterangan:

d = rasio kelas minoritas terhadap mayoritas.

m_s = banyaknya kelas minoritas.

m_l = banyaknya kelas mayoritas.

b. Menghitung banyak data sintetik yang akan dibuat.

$$G = (m_l - m_s)\beta \tag{2.8}$$

Keterangan:

G = jumlah data sintetik yang akan dibuat.

m_s = banyaknya kelas minoritas.

m_l = banyaknya kelas mayoritas.

β = rasio kelas minoritas terhadap mayoritas setelah diterapkan ADASYN. Jika $\beta = 1$ artinya banyak kelas minoritas sama dengan banyak kelas mayoritas setelah diterapkan ADASYN. Nilai β harus berada direntang 0 sampai 1.

c. Untuk setiap $x_i \in$ kelas minoritas, cari *k-Nearest Neighbors* berdasarkan jarak Euclidean dalam *n* dimensi ruang, dan hitung rasio r_i.

$$r_i = \frac{\Delta_i}{K}, \qquad i = 1,2, \dots, m_s \tag{2.9}$$

Keterangan:

r_i = nilai r_i menunjukkan dominasi kelas mayoritas di *neigbourhood* tertentu.

Δ_i = jumlah contoh kelas minoritas dalam *k-Nearest Neighbors.*

K = jumlah *nearest neighbors*.

d. Normalisasi nilai r_i

$$\hat{r}_i = \frac{r_i}{\sum r_i} \tag{2.10}$$

$$\sum \hat{r}_i = 1 \tag{2.11}$$

Keterangan:

$\hat{r}_i$ = nilai r_i setelah dinormalisasi

e. Menghitung jumlah data sintetik yang harus dibangkitkan per *neigbourhooods*.

$$G_i = G \times \hat{r}_i \tag{2.12}$$

Keterangan:

G_i = jumlah data sintetik yang harus dibangkitkan per *neigbourhooods*.

f. Hasilkan G_i data untuk setiap *neigbourhoods*. Pertama, ambil *neigbourhoods* untuk contoh kelas minoritas. Kemudian pilih secara acak kelas minoritas yang terdapat dalam *neighbourhoods* yang sama dengan tahap sebelumnya. Data sintetik dapat dihitung menggunakan persamaan berikut.

$$s_i = x_i + (x_{zi} - x_i) \times \lambda \quad (2.13)$$

Keterangan:

s_i = data sintetik.

x_i = *neigbourhoods* untuk contoh kelas minoritas.

x_{zi} = kelas minoritas yang terdapat dalam n*eighbourhoods* x_i

λ = bilangan acak antara 0 sampai 1.