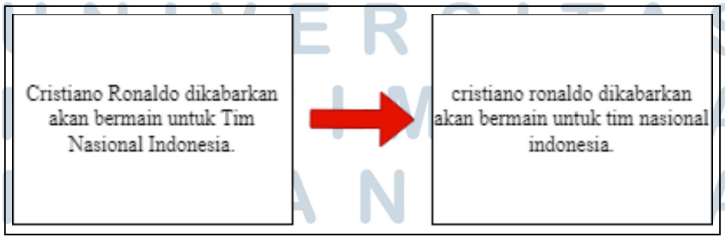
BAB 2
LANDASAN TEORI

2.1 Text Classification dan Text Preprocessing

Klasifikasi teks adalah pemberian label dengan kategori yang sudah ditentukan pada sebuah teks dengan bahasa alami [5]. Klasifikasi teks dapat dilakukan dengan dua cara berbeda: klasifikasi manual dan otomatis. Cara yang pertama, manusia menafsirkan konten teks dan mengkategorikannya. Metode ini biasanya dapat memberikan hasil yang berkualitas tetapi memakan waktu dan mahal. Kedua menerapkan pembelajaran mesin, pemrosesan bahasa alami dan teknik lainnya untuk secara otomatis mengklasifikasikan teks dengan cara yang lebih cepat dan hemat biaya. Ada beberapa cara umum dalam klasifikasi teks secara otomatis, yaitu *preprocessing, modeling*, dan *feature extraction* menggunakan teknik pembelajaran mesin serta *training* dan *testing* pada *classifier* [6]. Pemrosesan teks adalah salah satu teknik dari *text mining*. Pemrosesan teks dilakukan untuk mengubah data tekstual yang tidak terstruktur menjadi daya yang terstruktur lalu disimpan kedalam basis data [7].

2.1.1 Case Folding

Case Folding merupakan salah satu bentuk teknik pemrosesan teks. Tujuan proses ini adalah mengubah semua huruf dalam dokumen menjadi huruf kecil [8]. 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 [9].



Gambar 2.1. Proses *Case Folding*

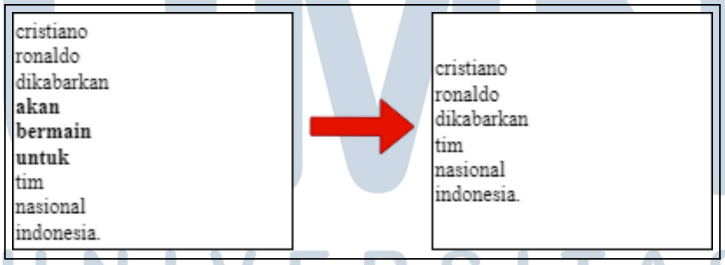
2.1.2 Tokenisasi

Secara garis besar tokenisasi adalah tahap memecah sekumpulan karakter dalam suatu teks kedalam satuan kata [10]. Menurut Hudin, tokenisasi merupakan proses pemotongan string input berdasarkan tiap kata penyusunnya [9].

Gambar 2.2. Proses Tokenisasi

2.1.3 Filtering

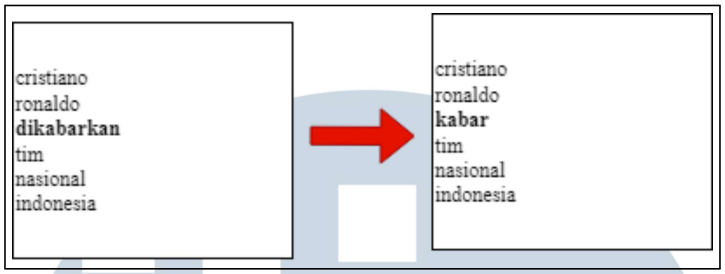
Filtering merupakan proses pemilihan kata-kata penting dari hasil tokenisasi, yaitu kata-kata yang bisa digunakan untuk mewakili isi dari sebuah teks atau dokumen. Proses *filtering* juga biasa disebut sebagai *stopword removal*. Pada proses ini terdapat dua teknik, yaitu *stop list* dan *word list*. *Stop list* merupakan proses membuang kata yang tidak deskriptif atau tidak penting. Sedangkan *word list* merupakan proses menyimpan kata yang dianggap penting [11].



Gambar 2.3. Proses *Stopword Removal*

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* [11].



Gambar 2.4. Proses *Stemming*

2.2 Support Vector Machine

Support Vector Machine (SVM) pertama kali diperkenalkan oleh Boser, Guyon, Vepnik, yang dipresentasikan untuk pertama kalinya pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep SVM merupakan kombinasi harmonis dari konsep komputer yang telah ada selama beberapa dekade, seperti *hyperplane*. Aturan utama penggunaan SVM adalah mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua kelas di ruang input. *Hyperplane* dapat berupa garis dua dimensi dan dapat berupa bidang di beberapa bidang. SVM adalah teknik pembelajaran mesin yang melakukan pelatihan menggunakan set data pelatihan dan menggeneralisasi dan membuat prediksi dari data baru. Prinsip dasar metode SVM adalah pengklasifikasi linier, dimana SVM dapat mengklasifikasikan data yang dapat dipisahkan secara linier, namun SVM dikembangkan untuk dapat mengatasi permasalahan non-linier dengan menerapkan konsep trik Nuklir dalam ruang kerja.

Prinsip dasar yang dimiliki oleh metode SVM adalah pengklasifikasi linier, dimana SVM dapat mengklasifikasikan data yang dapat dipisahkan secara linier, tetapi SVM telah dikembangkan untuk dapat mengatasi masalah non-linier dengan menerapkan konsep trik kernel di ruang kerja. Tujuan dari SVM adalah untuk mendapatkan *hyperplane* terbaik yang memisahkan dua kelas pada SVM linier dan beberapa lapisan pada SVM nonlinier [12]. Penelitian ini akan menggunakan beberapa kernel SVM yang akan diuji di antaranya kernel yang menghasilkan akurasi terbaik. Kernel yang akan digunakan antara lain *Linear* dan *Radial Basis Function* (RBF). Keluarannya adalah model klasifikasi. Setiap kernel dihitung dengan rumus berikut:

1. Kernel Linear

   Kernel *Linear* adalah fungsi kernel yang paling sederhana. Perkalian *linear* digunakan ketika data yang dianalisis dapat dipisahkan secara *linear*. Kernel *Linear* cocok di mana ada banyak fungsi, karena pemetaan ke ruang dimensi yang lebih tinggi tidak benar-benar meningkatkan kinerja seperti dalam klasifikasi teks. Dalam klasifikasi teks, jumlah kasus (dokumen) dan jumlah fitur (kata) adalah sama [13]. Berikut merupakan persamaan dari kernel *linear* SVM.

$$K(x,y) = x.y \tag{2.1}$$

   $K =$ Kernel

   $x =$ Titik data

   $y =$ Kelas data

2. Kernel *Radial Basis Function* (RBF)

   Kernel RBF merupakan fungsi kernel yang biasa digunakan dalam analisis ketika data tidak terpisah secara linear. Kernel RBF memiliki dua parameter yaitu *Gamma* dan *Cost*. Parameter *Cost* atau biasa disebut sebagai C merupakan parameter yang bekerja sebagai pengoptimalan SVM untuk menghindari misklasifikasi di setiap sampel dalam *training dataset*. Parameter *Gamma* menentukan seberapa jauh pengaruh dari satu sampel *training dataset* dengan nilai rendah berarti "jauh", dan nilai tinggi berarti "dekat". Dengan *gamma* yang rendah, titik yang berada jauh dari garis pemisah yang masuk akal dipertimbangkan dalam perhitungan untuk garis pemisah. Ketika *gamma* tinggi berarti titik – titik berada di sekitar garis yang masuk akal akan dipertimbangkan dalam perhitungan [14]. Berikut merupakan persamaan dari Kernel RBF.

$$K(x,y) = e[-y||x - x'||^2] \tag{2.2}$$

   $K =$ Kernel

   $x =$ Titik data

   $y =$ Kelas data

   $x - x' =$ Jarak Euclidean antara vektor (2 titik)

Metode yang akan digunakan untuk menghitung *Support Vector Machines* adalah dengan menggunakan *Metode Sequential Training*. Merupakan salah satu algoritma SVM yang memiliki algoritma yang lebih sederhana dan waktu yang dibutuhkan lebih cepat untuk menyelesaikan metode SVM. Algoritma *Sequential Training* adalah sebagai berikut:

1. Menginisialisasi α_i = 0 dan parameter lain, misalnya λ = 0,003, γ = 0,007, C = 1, IterasiMax = 30, dan ε = 0,00001. Kemudian menghitung matriks Hessian dapat dihitung dengan rumus:

$$D_{ij} = y_i y_j (K(x_i, x_j)) + \lambda^2 \tag{2.3}$$

   D_{ij} = Matriks Hessian

   y_i = Kelas data ke-i

   y_j = Kelas data ke-j

   $K(x_i, x_j)$ = Fungsi kernel yang digunakan

2. Mulai dari data *i* sampai *l*, hitung menggunakan persamaan berikut:

   - $E_i = \sum_{j=1}^{n} \alpha_j D_{ij}$
   - $\theta\alpha_i$ = min { max[$\lambda(1 - E_i), -\alpha_i$], C-$\alpha_i$ }
   - $\alpha_i = \alpha_i + \theta\alpha_i$

   α_j = Alfa ke-j

   E_{ij} = *Error rate*

   λ = Konstanta *Gamma*

   C = Konstanta C

   $\theta\alpha_i$ = Delta alfa ke-i

3. Langkah kedua dilakukan terus-menerus hingga kondisi iterasi maksimum tercapai.

2.3 Term Frequency Inverse Document Frequency

Metode *Term Frequency Inverse Document Frequency* (TF-IDF), salah satu fitur paling populer karena metode pembobotan yang digunakan untuk menggambarkan dokumen dalam model ruang vektor dan aplikasi yang terkait dengan *text*

mining dan pencarian informasi, dapat secara efektif mencerminkan pentingnya istilah dalam pengumpulan dokumen, di mana semua dokumen memainkan peran yang sama [15]. Namun, TF-IDF tidak memperhitungkan perbedaan bobot istialh IDF jika dokumen memainkan peran yang berbeda dalam pengumpulan dokumen, seperti *training* set positif dan negatif dalam klasifikasi teks [15].

Langkah dalam TF-IDF adalah untuk menemukan jumlah kata yang kita ketahui setelah dikalikan dengan berapa banyak *tweet* di mana suatu kata itu muncul. Rumus dalam menentukan pembobot dengan TF-IDF adalah sebagai berikut:

$$W_{ij} = tf_{ij} \times idf \tag{2.4}$$

$$idf = log\frac{(N)}{(n)} \tag{2.5}$$

W_{ij} = bobot kata (*term*) t_j terhadap dokumen d_i

tf_{ij} = jumlah kata (*term*) d_j terhadap dokumen d_i

N = jumlah semua dokumen yang ada dalam *database*

n = jumlah dokumen yang mengandung kata (term) t_j

Apabila didapatkan nilai 0 untuk idf, akan ada perubahan rumus sebagai berikut:

$$W_{ij} = tf_{ij} \times \log\frac{(N)}{(n)} + 1 \tag{2.6}$$

Rumus di atas dapat dinormalisasi, dengan tujuan untuk menstandarisasi nilai bobot ke dalam interval 0 sampai dengan 1. Rumus TF-IDF dengan menggunakan normalisasi sebagai berikut:

$$W_{ij} = \frac{tf_{ij} \times \log\frac{(N)}{(n)} + 1}{\sqrt{\sum_{k=1}^{t}(tf_{ij})^2 \times [\log\frac{(N)}{(n)} + 1]^2}} \tag{2.7}$$

2.4 F1-Score

Pengukuran kinerja model dalam penelitian akan menggunakan metrik F1-Score. Confusion Matrix adalah alat untuk memvisualisasi kualitas klasifikasi yang

dihasilkan menggunakan penggunaan matrix [16]. Dalam visualisasinya, confusion matrix terdiri dari empat faktor yang menentukan performa klasifikasi. Berikut adalah rincian empa faktor berdasarkan, yaitu:

1. TP = *True Positive* = Prediksi Positif yang benar
2. FP = *False Positive* = Prediksi Positif yang salah (eror klasifikasi)
3. TN = *True Negative* = Prediksi Negatif yang benar
4. FN = *False Negative* = Prediksi Negatif yang salah (eror klasifikasi)

Rumus untuk perhitungan *performance metric* yang umum digunakan dalam *confusion matrix* terdiri dari:

1. Akurasi = Keakuratan klasifikasi

$$\frac{TP+TN}{TP+FN+FP+TN} \tag{2.8}$$

2. *Precisssion* = Rasio prediksi benar positif dengan keseluruhan yang diprediksi positif

$$\frac{TP}{TP+FP} \tag{2.9}$$

3. *Recall* = Rasio prediksi benar positif dengan keseluruhan data yang benar positif

$$\frac{TP}{TP+FN} \tag{2.10}$$

Confusion Matrix menjadi penting karena dapat digunakan untuk mengevaluasi hasil klasifikasi. Dimana akurasi yang dimiliki pada suatu hasil klasifikasi berpengaruh terhadap performa dari sebuah klasifikasi [14]. Tabel 2.1 adalah contoh visualisasi dari *confusion matrix*.

Tabel 2.1. Contoh *Confusion Matrix*

	AKTUAL	
PREDIKSI	Positif	Negatif
Positif	TP	FP
Negatif	FN	TN