BAB 2
LANDASAN TEORI

Melihat perlunya pemahaman akan teori yang ada dalam penelitian ini, terdapat beberapa sumber yang diambil dari jurnal dan artikel yang telah disebarkan secara resmi serta dalam forum sebagai landasan dalam penelitian ini. Menelaah penelitian terdahulu juga membantu dalam penentuan serta penerapan algoritme SVM pada analisis sentimen ulasan aplikasi PeduliLindungi.

2.1 Sentimen Analisis

Sentimen analisis adalah instrumen yang umum dipakai untuk representasi sebuah emosi dari pengguna sosial media, konsumen dan publik dalam sekala besar, dan memudahkan untuk riset pasar, politik dan perhitungan politik. Sentimen analisis, juga disebut penggalian opini, menggambarkan cara untuk memetakan masalah melalui pengukuran opini, sentimen dan unsur subjektif [2]. Sentimen dalam garis besar dibagi dalam gambaran emosi dan sudut pandang dari seseorang terhadap sesuatu [6]. Sentimen dapat dengan mudah dianalisa dikarenakan adanya lonjakan popularitas dari sosial media, dan penggunaannya dapat dimaksimalkan oleh bisnis dengan adanya iklan dengan dilandaskan dengan sentimen analisis, untuk menggambarkan gambaran dari target pengguna melalui sentimen [2].

2.2 Preprocessing

Tahap *preprocessing* merupakan suatu proses pembersihan data dengan tujuan meningkatkan kualitas data yang dimiliki [7]. Tahapan yang dilakukan dalam *preprocessing* yaitu pengubahan kata pada huruf kapital, penghilangan tanda baca, simbol dan angka. Tahap selanjutnya yang dilakukan merupakan *stemming* yaitu penghilangan imbuhan, dan menghilangkan kata sambung atau *stopword*. Langkah terakhir yang dilakukan yaitu *tokenizing* yaitu memecah kalimat menjadi kumpulan kata yang telah terpisah. Pada Tabel 2.1 akan terlampir contoh dari pengimplementasian *preprocessing*.

Tabel 2.1. Contoh implementasi preprocessing

Sebelum *Preprocessing*	**Setelah *Preprocessing***
Gampang penggunaan aplikasi 9 :) !	[’gampang’, ’guna’, ’aplikasi’]

Dapat dilihat adanya beberapa perubahan setelah dilakukannya *preprocessing*. Setelah melalui proses, terlihat adanya pengilangan tanda seru, angka 9 dan *emoji* senyum. Perbedaan lainnya terdapat pada kata ’penggunaan’ menjadi ’guna’ setelah dilakukan *stemming*. Proses terakhir yang dilakukan merupakan *tokenizing*, yaitu pemecahan kata dari yang sebelumnya ’gampang guna aplikasi’ menjadi 3 kata yaitu ’gampang’, ’guna’, dan ’aplikasi’.

2.3 Support Vector Machine

Support Vector Machines (SVM) menyediakan metode klasifikasi, dan merupakan algoritme *supervised* [8]. Algoritme *supervised* merupakan algoritma yang berlatih menggunakan data yang telah ditandai sebagai dasar, atau dalam bahasa lain telah diajari sebelumnya. SVM dapat juga dibagi dalam model linear dan nonlinear [9]. Beberapa langkah yang harus dilakukan yaitu memetakan data yang telah ada, dan mencari data yang berada di ujung pemisah. Data domain yang dipisahkan kemudian dibagi dengan pemisah antara data yang telah dipisahkan menjadi beberapa *cluster* tergantung dengan *feature* yang dipilih.

Pemisah antara data yang menjadi *cluster* ini disebut dengan *hyperplane* [10]. Penentuan dari *hyperplane* tersebut berdasarkan titik yang berada di antara tiap *cluster* dengan jarak terdekat dengan *cluster* lainnya, sehingga dapat disebut *support vector* [11].

Penentuan dalam pemilihan *hyperlane* dengan menggunakan *support vector* dan mencari *margin* terbesar. *Margin* merupakan jarak yang ada terhadap *support vector* terhadap garis *hyperlane*. Dalam beberapa kasus, bentuk *hyperlane* dalam dua *feature* tidak selalu berbentuk garis lurus yang membagi dua *feature*. Jika menggunakan garis lurus, maka dapat disebut menggunakan *kernel* linear, sedangkan bila tidak dipisahkan dengan laris lurus maka akan disebut non-linear. Dikarenakan memiliki lebih dari banyak fitur, maka dilakukan metode *ovr* (*one versus rest*) yaitu dengan membagi perhitungan sesuai dengan fitur yang dimiliki, jika memiliki tiga fitur maka akan memiliki tiga perhitungan, dan dalam

perhitungan satu kelas akan dianggap positif dan kedua kelas lain akan dianggap negatif [5].

2.4 TF-IDF

TF-IDF merupakan sebuah metode dalam *feature extraction* yang menggunakan TF (*Term Frequency*) dan IDF (*Inverse document frequency*). TF merupakan Banyaknya muncul sebuah kata dalam suatu kalimat. IDF di sisi lain merupakan penggambaran dari *inverse* banyaknya kata dalam kalimat yang terdapat pada sebuah artikel. TF merupakan sebuah pengukuran dari banyaknya suatu kata dalam suatu kalimat. Perhitungan dalam TF dapat menggunakan banyaknya kata dalam suatu kalimat tersebut [12]. Di sisi lain, IDF atau *Inverse document frequency* merupakan kebalikan dari munculnya suatu kata dalam kalimat, dibandingkan dengan jumlah dari seluruh kalimat yang ada. Jika suatu kata terkandung dalam satu kalimat, maka dihitung satu, dan jika terkandung dalam dua kalimat, maka akan dihitung dua. Rumus perhitungan IDF ini merupakan logaritma dari jumlah dokumen dibagi dengan perhitungan jumlah kalimat yang mengandung kata yang dimaksud [13].

Rumus perhitungan TF.

$$TF_t = (t,d) \tag{2.1}$$

TF merupakan kemunculan variabel t dalam dokumen atau kata d. Variabel t merupakan jumlah kata yang terdapat dalam suatu kalimat pada dokumen d, dan TF merupakan jumlah pemunculan dari kata t. Setelah melakukan perhitungan TF, maka akan dilakukan perhitungan IDF.

Rumus perhitungan IDF.

$$IDF_t = \log \frac{n}{df(t)} \tag{2.2}$$

Dalam rumus 2.2, IDF didapat perhitungan logaritma dari jumlah dokumen yang diberikan dibagi dengan jumlah dari keberadaan kata t dalam suatu dokumen. Setelah didapatkan perhitungan TF dan IDF, maka akan dilakukan perhitungan bobot TF-IDF.

Rumus perhitungan TF-IDF.

$$W_t = TF_t \times IDF_t \tag{2.3}$$

TF-IDF merupakan perkalian dari perhitungan TF dan IDF yang telah dihitung dalam kata t. Dari perhitungan ini didapatkan bobot W yang dapat diterapkan pada model.

2.5 CountVectorizer

Selain dari TF-IDF, terdapat juga metode penelitian dengan menggunakan CountVectorizer. CountVectorizer merupakan suatu algoritme *feature extraction* yang bekerja dengan menghitung jumlah pemunculan kata pada suatu dokumen, sehingga akan membuat kata yang dominan menjadi lebih berpengaruh dengan hasil yang diberikan [12]. Data yang ada akan dibuat secara unik dalam suatu kamus untuk dihitung bobotnya.

1. Aplikasi mudah mudah digunakan.
2. Aplikasi lemot.
3. Aplikasi sangat lemot.

Terdapat tiga kata sebagai contoh perhitungan dari CountVectorizer. Contoh perhitungan dapat dilihat pada Tabel 2.2, menggunakan kalimat pertama sebagai contoh.

Tabel 2.2. Contoh perhitungan bobot CountVectorizer

Kata	Aplikasi	mudah	digunakan	lemot	sangat
Bobot	1	2	1	0	0

Terlihat bobot yang telah dihitung menggunakan CountVectorizer. Kata 'Aplikasi' dan 'mudah' memiliki bobot 1, dikarenakan hanya muncul satu kali dalam kalimat pertama. Kata 'mudah' muncul dua kali, sehingga diberi bobot 2. Sedangkan kata 'lemot' dan 'sangat' tidak terdapat dalam kalimat sehingga bobotnya 0.

2.6 Perhitungan Performa

Dalam membangun sebuah model, diperlukan sebuah pengukuran dari performa suatu model. Perhitungan dari performa suatu model didasari dengan perhitungan TP (*True Positive*), TN (*True Negative*), FP (*False Positive*) dan FN (*False Negative*). TP merepresentasikan jumlah data yang benar diprediksi oleh model dan bernilai positif. TN merupakan jumlah data yang tepat diprediksi oleh model namun bernilai negatif. FP merupakan kesalahan prediksi data, yang memiliki hasil negatif secara aktual namun diprediksi memiliki hasil positif. FN merupakan kesalahan prediksi di mana data diprediksi menjadi positif namun pada kenyataannya bernilai negatif. Berikut merupakan Confusion Matrix dari TP, TN, FP, FN.

Tabel 2.3. Confusion Matrix TP, TN, FP, FN

Aktual	**Prediksi**	
	1	0
1	TP	FP
0	FN	TN

Pada Tabel 2.1 kolom menggambarkan data yang sebenarnya, sedangkan baris menggambarkan prediksi dari suatu model. Tabel 2.3 merupakan acuan yang digunakan dalam perhitungan akurasi, *precision*, *recall*, dan *f1*.

Perhitungan Akurasi.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.4}$$

Perhitungan akurasi dengan cara menghitung jumlah *true positive* (TP) dan *true negative* (TN) dibagi dengan hasil *true positive*, *true negative*, *false positive* (FP) dan *false negative* (FN) [14]. Perhitungan ini adalah untuk melihat ketepatan model dalam melakukan prediksi.

Penghitungan *Precision*

$$Precision = \frac{TP}{TP+FP} \tag{2.5}$$

Penghitungan precision dengan cara menghitung TP dibagi dengan hasil TP danFP. Pada perhitungan lebih dari dua *class*, *precision*, akan mengambil semua yang diprediksi secara positif dan benar dibagi dengan semua yang diprediksi menuju *class* tersebut (pada perhitungan positif akan dihitung semua yang diprediksi positif, pada perhitungan negatif akan dihitung semua yang diprediksi negatif) [15].

Penghitungan *Recall*

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Penghitungan *recall* dengan cara menghitung TP dibagi dengan hasil TP dan FN. Pada perhitungan lebih dari dua *class*, maka akan diambil perhitungan positif akan dihitung semua yang memliki nilai aktual positif, pada perhitungan negatif akan dihitung semua memliki nilai aktual negatif.

Penghitungan *F1*

$$F1 = 2 * \frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

Penghitungan *F1* dengan cara menghitung perkalian antara *precision* dan *recall* dibagi dengan perkalian antara *precision* dan *recall* lalu dikalikan dengan dua. Perhitungan *f-score* atau skor F1 bertujuan sebagai alternatif pengukuran suatu metode selain dari akurasi [15].