BAB 2
LANDASAN TEORI

2.1 Hoax

Hoax merujuk kepada informasi yang mencakup elemen yang tidak dapat dipercaya dan juga *hoax* memiliki informasi yang tidak sesuai dengan suatu kejadian yang terjadi pada kenyataannya, dalam konteks *hoax* itu sendiri memiliki tanda-tanda bahwa berita tersebut pasti berasal dari sumber yang tidak jelas keberadaannya jika informasi *hoax* tersebut masuk ke dalam kalangan masyarakat tentunya bisa berdampak negatif dan juga dengan adanya *hoax* bisa merusak kepercayaan publik, dan konsekuensi lainya[2].

Hoax adalah informasi yang dirancang untuk menyembunyikan informasi sebenarnya. Dengan kata lain, hoax juga dapat dipahami sebagai upaya memutarbalikkan kebenaran dengan menggunakan informasi yang terkesan meyakinkan dan tidak dapat diverifikasi. *Hoax* juga dapat dipahami sebagai tindakan menyembunyikan informasi yang sebenarnya, dengan cara membanjiri suatu media dengan pesan-pesan palsu guna menyembunyikan pesan yang sebenarnya. Tujuan dari *hoax* yang disengaja adalah untuk membuat orang merasa tidak aman, kesal, dan bingung. Pada saat terjadi kebingungan, masyarakat akan mengambil keputusan yang lemah, tidak meyakinkan, bahkan salah[11].

2.2 Text Mining

Text mining umumnya digunakan untuk menganalisis informasi serta bisa juga dengan menjalankan tugas-tugas manajemen informasi lainnya yang berhubungan dengan data teks yang berjumlah besar[11]. Data tidak terstruktur semakin banyak digunakan dengan format teks dan juga data teks contohnya klasifikasi, *clustering*, analisis sentimen, dan lain sebagainya untuk memenuhi kebutuhan analisis informasi, dalam data teks mempunyai karakter yang lebih rumit dibandingkan dengan data biasa atau bahkan data yang teratur. Karena bentuk data teks banyak yang tidak beraturan sehingga dengan metode ini diperlukan sebagai langkah untuk menyusun dasar teks yang dapat dimodifikasi agar memiliki struktur yang lebih konsisten[13].

2.3 Preprocessing

PreProcessing adalah langkah-langkah ini ditujukan untuk mengubah data teks yang awalnya tidak beraturan yang kemudian dengan langkah-langkah ini menjadikan data

tersebut menjadi terstruktur sehingga bisa dilanjutkan dengan proses klasifikasi. Adapun tahapan yang dilakukan pada penelitian ini dapat dilihat sebagai berikut:

1. Case Folding

Case Folding digunakan untuk menyamakan bentuk pada kata, contohnya seperti yang dilakukan pada data narasi yang digunakan pada penelitian ini yaitu mengubah semua kata menjadi huruf kecil atau disebut *lowercase*. Pada penelitian ini karakter-karakter 'A' hingga 'Z' dalam data diubah menjadi karakter 'a' hingga 'z'[14].

2. Cleaning Data

Berita yang ditulis yang terdapat angka, *emoji*, dan tanda baca pada ketiganya tidak penting untuk diolah, sehingga perlu dihapus untuk mempermudah dalam pengolahan data[15]. Pada penelitian ini proses *Cleaning data* menggunakan library re (*Regular expression*).

3. Tokenization

Tokenization digunakan untuk membagi *string input* menjadi bagian-bagian berdasarkan kata-kata yang membentuknya, dengan tahapan ini untuk memecah kalimat menjadi kata-kata atau secara umumnya tahapan ini akan dilakukan tahap pemotongan memisahkan kata-kata berdasarkan *white space* atau spasi[15]. Tahap *Tokenization* membagi urutan karakter menjadi kalimat dan kemudian kalimat menjadi token. Pada penelitian ini proses *tokenization* menggunakan *library NLTK*.

4. Stopword

Stopword merupakan kata-kata umum yang sering muncul tanpa makna atau tanpa pengaruh yang berarti dalam sebuah kalimat, oleh karena itu penghapusan stopword sangat diperlukan dimana penulis melakukan proses penghapusan berdasarkan daftar kata-kata *stopword* pada dataset[15]. Pada penelitian ini proses *stopword* menggunakan *library NLTK*.

5. Stemming

Stemming merupakan sebuah proses pemetaan dan penguraian bentuk dari suatu bentuk kata ke bentuk kata dasarnya dengan menghilangkan semua imbuhan, baik yang terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (suffixes) dan kombinasi dari awalan dan akhiran (*confixes*), sehingga dapat memperkecil panjang string dan jumlah

kata yang berbeda[15]. Pada penelitian ini, perubahan kata menjadi kata dasar menggunakan *library streamer Sastrawi*.

2.4 Pembobotan TF-IDF

Pembobotan merujuk pada proses konversi kata-kata yang bersifat kualitatif dalam dokumen menjadi data kuantitatif sehingga proses selanjutnya bisa di proses dengan komputer, dan metode ini biasanya dilakukan setelah melalui proses *preprocessing*, tahap pembobotan pada penelitian ini penulis akan menggunakan fitur TF-IDF (*Term Frequency Inverse Document Frequency*). Algoritma TF-IDF itu sendiri merupakan kombinasi dari dua buah aturan dalam pembentukan fiturnya yang dimana setiap aturannya menangkap salah satu dari dua intuisi yang disajikan[16]. Pada penelitian ini pembobotan TF-IDF akan menggunakan library dari sklearn maka persamaannya akan seperti berikut:

1. *Term Frequency* (TF), didefinisikan sebagai banyaknya kata yang muncul pada dokumen tersebut, frekuensi dihitung sebagai hasil bagi banyaknya kata terhadap keseluruhan kata di dokumen tersebut atau secara matematis dapat dibentuk sebagai berikut (2.1):

$$tf_i = \frac{freq_i(d_j)}{\sum_{i=1}^{k} freq_i(d_j)} \tag{2.1}$$

Dimana :

tf = *term* frekuensi atau banyaknya kata pada dokumen

$freq_i(d_j)$ = frekuensi kemunculan term atau *term* ke-i dalam dokumen ke-j

2. *Inverse Document Frequency* (IDF), mengidentifikasi frekuensi kemunculan yang tinggi atau rendah suatu istilah dalam dokumen untuk menentukan nilai idfnya, di mana semakin rendah frekuensi istilah dalam dokumen, semakin tinggi nilai idfnya, dapat dilihat pada persaaman (2.2) sebagai berikut :

$$idf_j = log\left(\frac{1+N}{1+df_j}\right) + 1 \quad (2.2)$$

Dimana :

idf_j = adalah jumlah dokumen dalam kumpulan dokumen yang mengandung term j

N = total sebuah dokumen

Oleh karenanya algoritma TF-IDF merupakan kombinasi kedua intuisi tf dan idf, dan perhitungan bobot untuk fitur TF-IDF didefinisikan sebagai:

$$w_{i,j} = tf_{i,j} * idf_i \quad (2.3)$$

Dimana :

$w_{i,j}$ = besar bobot TF-IDF

3. Kemudian lakukan normalisasi pada bobot yang telah didapatkan. Untuk penggunaan library sklearn maka akan menggunakan persamaan 2.4

$$w_{i,j} = \frac{tf_{i,j} * idf_i}{\sqrt{\sum_{i=1}^{n}(tf_{i,j} * idf_i)^2}} \quad (2.4)$$

Dimana :

n = banyak kata

i = iterasi ke-

$w_{i,j}$ = nilai bobot akhir kata

2.5 Neighbor-Weighted K-Nearest Neighbor (NWKNN)

Pandangan bahwa data latih tersebar secara merata dan seimbang untuk semua kelas namun bisa saja ada peluang bahwasanya ada data latih yang tidak merata dan seimbang sehingga bisa artikan pada metode ini pada kelas mayoritas cenderung memiliki representasi yang jauh lebih besar dalam data latih, sementara kelas minoritas memiliki representasi yang lebih sedikit. Metode ini didefinisikan akan memberikan bobot pada k tetangga terdekat yang berasal dari kelas mayoritas, bobotnya akan kecil, sementara pada k tetangga dari kelas minoritas, bobotnya akan besar. Dalam proses metode NWKNN memilih k tetangga terdekat yang berasal pada kelas k untuk semua data uji adapun langkah-langkah algoritma NWKNN mirip dengan langkah-langkah algoritma KNN bedanya pada bagian perhitungan bobot dan perhitungan skor untuk mengklasifikasi data uji[16]. Bobot kelas dapat dihitung menggunakan rumus atau persamaan (2.5):

$$weight_i = \frac{1}{\left(\frac{Num\left(c\frac{d}{i}\right)}{Min\left\{Num\left(c\frac{d}{j}\right)\middle| j = 1 \dots, k^*\right\}}\right)\frac{1}{Exp}} \tag{2.5}$$

Dimana :

$Num(C\frac{d}{i})$ = Jumlah data pelatihan (data latih) dalam kelas i.

$Num(C\frac{d}{J})$ = Jumlah data pelatihan (data latih) dalam kelas j.

Di mana j adalah bagian dari kumpulan k tetangga terdekat.

Exp = Eksponen (angka yang lebih besar dari 1).

K = Total jumlah tetangga terdekat.

Langkah selanjutnya setelah diperoleh nilai bobot akan digunakan untuk mengkomputasi nilai skor data uji pada kelas. Perhitungan skor pada metode NWKNN dapat dilakukan dengan persamaan (2.6):

$$score(x, c_i) = weight_i \left(\sum_{d_j \in KNN(X)} \sqrt{\sum_{i=0}^{n} (x_i - y_j)^2} * \delta(d_j, C_i) \right) \tag{2.6}$$

Dimana :

$Weight_i$ = Bobot kelas pada *i*

$d_j \in$ NWKNN(X) = Data Latih dj, pada kumpulan tetangga terdekat dari data uji X

$\delta(X, d_j)$ = Akan diberi nilai 1 jika nilai jarak ∈ C*i* dan

akan diberi nilai 0 jika nilai jarak ∉ C*i*

C_i = Jenis atau kelas pada *i*

2.6 Jarak Euclidean

Euclidean distance adalah perhitungan yang digunakan untuk mengukur jarak dua titik dalam *euclidean space* yang mempelajari hubungan antara sudut dan jarak[17]. Untuk menghitung jarak kedekatan dengan menggunakan *Euclidean Distance*. Metode pencarian antara dua titik variabel, semakin dekat dan mirip maka semakin kecil jarak antara dua titik tersebut. Perhitungan jarak pada metode KNN dengan nilai kedekatan ketetanggaan menggunakan perhitungan persamaan 2.7 *Euclidean Distance*:

$$d(x_1, x_2) = \sqrt{\sum_{i=0}^{n} (x_{1i} - x_{2i})^2} \quad (2.7)$$

Dimana :

X_1 = nilai setiap data latih

X_2 = nilai setiap data uji

n = banyak data

i = data ke i

2.7 Pengukuran Akurasi

Salah satu alat yang dapat digunakan sebagai ukuran performansi adalah *confusion matrix*. *Confusion matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya dilakukan[18]. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi terlihat pada Tabel 2.1 berikut.

Tabel 2.1 Tabel Kontigensi

		Kelas Prediksi	
		Positive	Negative
Kelas Aktual	Positive	TP	FN
	Negative	FP	TN

Dengan keterangan :

TP = True Positive atau jumlah tupel positif yang dilabeli dengan benar oleh clasiffier.

TN = True Negative atau jumlah tupel negatif yang dilabeli dengan benar oleh clasiffier.

FP = False Positive atau jumlah tupel positif yang salah dilabeli oleh clasiffier.

FN = False Negative atau jumlah tuple negatif yang salah dilabeli oleh clasiffier.

Sedangkan untuk mengukur dari segi akurasi, *Precision*, *Recall*, dan *F-1 Score* itu sendiri menggunakan persamaan berikut:

1. *Accuracy,* menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \quad (2.8)$$

2. *Precision*, menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model. *Precision* dihitung dari jumlah pengenalan bernilai benar oleh sistem dibagi dengan jumlah keseluruhan pengenalan yang dilakukan oleh sistem[19].

$$Precision = \frac{(TP)}{(TP + FP)} \quad (2.9)$$

3. *Recall*, menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. *Recall* menyatakan jumlah pengenalan entitas bernilai benar dilakukan sistem dibagi dengan jumlah entitas yang seharusnya dapat dikenali sistem[19].

$$Recall = \frac{(TP)}{(TP + FN)} \quad (2.10)$$

4. *F-1 Score* menggambarkan perbandingan rata-rata *Precision* dan *Recall* yang dibobotkan.

$$\mathrm{F-1\ Score} = \frac{2 * Recall * Precision)}{(Recall + Precision)} \quad (2.11)$$