BAB 2
LANDASAN TEORI

2.1 Text Classification

Text Classification merupakan pemberian label dengan kategori yang sudah ditentukan pada sebuah teks dengan bahasa alami [13]. Klasifikasi teks diterapkan dalam berbagai konteks, mulai dari pengindeksan sebuah dokumen berdasarkan kosa kata, pemfilteran sebuah dokumen, pembuatan metadata otomatis, dan pada macam aplikasi lainnya [13]. Ada beberapa cara umum dalam klasifikasi teks secara otomatis, yaitu *preprocessing*, *feature extraction/selection*, *modeling* menggunakan teknik pembelajaran mesin, serta *training* dan *testing* pada classifier [14].

2.2 Text Preprocessing

Text Processing merupakan salah satu teknik dari *text mining*. *Text processing* dilakukan untuk mengubah data tekstual yang tidak terstruktur menjadi daya yang terstruktur lalu disimpan kedalam basis data [15].

2.2.1 Case Folding

Case Folding merupakan salah satu bentuk teknik *text preprocessing*. Tujuan proses ini adalah mengubah semua huruf dalam dokumen menjadi huruf kecil [16]. 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 [17]. Lalu pada tahap ini juga akan dilakukan penghapusan spasi di awal dan akhir, teknik ini biasa disebut *whitespace removal* [18].

2.2.2 Tokenisasi

Secara garis besar tokenisasi adalah tahap memecah sekumpulan karakter dalam suatu teks kedalam satuan kata [2]. Menurut [3] Tokenisasi merupakan proses pemotongan string input berdasarkan tiap kata penyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen [4].

2.2.3 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 *stopwords 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 [3].

2.2.4 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* [12].

2.3 Term Frequency Inverse Document Frequency

Metode Term Frequency-Inverse Document Frequency (TF-IDF) merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada informasi *retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat [13].

Metode TF-IDF adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. TF-IDF ini merupakan pengukuran statistik yang digunakan untuk mengevaluasi seberapa penting sebuah kata didalam sebuah dokumen atau dalam sekelompok kata. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Frekuensi kemunculan kata didalam dokumen yang diberikan menunjukkan seberapa penting kata itu didalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen [15].

Rumus yang digunakan untuk TF-IDF sebagai berikut.

$$w_{ij} = tf \times idf \quad (2.1)$$

$$w_{ij} = tf_{ij} \times log\left(\frac{N}{n}\right) \quad (2.2)$$

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

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

N = jumlah semua dokumen yang ada dalam basis data

n = jumlah dokumen yang mengandung kata (*term*) t_j

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

$$w_{ij} = tf_{ij} \times log\left(\frac{N}{n}\right) + 1 \quad (2.3)$$

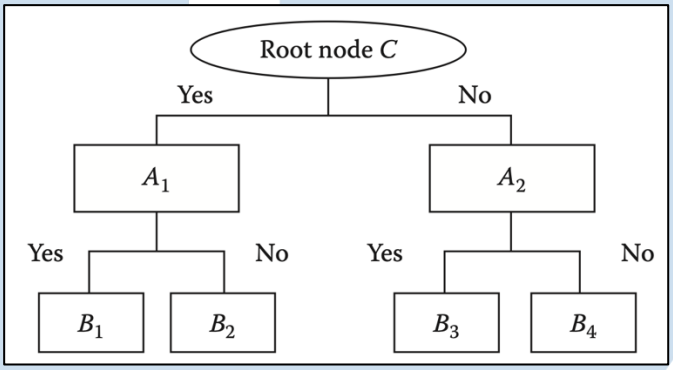
Rumus diatas dapat dinormalisasi, dengan tujuan untuk menstandarisasi nilai bobot ke dalam interval 0 s.d. 1. Rumus TF-IDF dengan menggunakan normalisasi sebagai berikut.

$$w_{ij} = \frac{tf_{ij} \times log\left(\frac{N}{n}\right) + 1}{\sqrt{\sum_{k=1}^{t} (tf_{ij})^2 \times \left[log\left(\frac{N}{n}\right) + 1\right]^2}} \quad (2.4)$$

2.4 Decision Tree

Decision Tree adalah sebuah pohon yang dimana setiap cabangnya menunjuk pilihan diantara sejumlah alternatif pilihan yang ada, dan setiap daunnya menunjukkan keputusan yang dipilih [8]. Algoritma *decision tree* didasarkan pada pendekatan *divide-and-conquer* untuk klasifikasi suatu masalah. Algoritma tersebut bekerja dari atas ke bawah, mencari pada setiap tahap atribut untuk membaginya ke dalam bagian terbaik class tersebut, dan memproses secara rekursif submasalah yang dihasilkan dari pembagian tersebut. Strategi ini menghasilkan sebuah *decision tree* yang dapat diubah menjadi satu set *classification rules* [3].

Pada *decision tree* terdapat 3 jenis *node*, yaitu *Root Node* merupakan *node* paling atas, pada *node* ini tidak ada input dan bisa tidak mempunyai output atau mempunyai output lebih dari satu, *Internal Node* merupakan *node* percabangan hanya terdapat satu input dan mempunyai output minimal dua, *Leaf Node* atau *Terminal Node* merupakan *node* akhir yang hanya terdapat satu input dan tidak mempunyai output [16]..



Gambar 2. 1 Struktur *Decision Tree* [17]

Berikut ini adalah rumus untuk perhitungan *dec*ision *tree* [16].

a. Entropi

$$Entropi\ (S) = \sum_{j=1}^{k} -p_j\ log_2\ p_j \qquad (2.5)$$

Keterangan variabel diatas sebagai berikut.

S = Himpunan (*dataset*) kasus.

K = Jumlah partisi S.

p_j = Probabilitas yang didapat dari Sum(Ya) dibagi Total Kasus

b. Gain

$$Gain\ (A) = Entropi\ (S) - \sum_{j=1}^{k} \frac{|S_i|}{|S|} \times Entropi\ (S_i) \qquad (2.6)$$

Keterangan penjelasan variabel diatas sebagai berikut.

S = Ruang (data) sample yang digunakan untuk *training*.

A = Atribut.

$|S_i|$ = Jumlah sample untuk nilai V.

$|S|$ = Jumlah seluruh sample data.

2.5 Ensemble

Ensemble merupakan teknik untuk metode pembelajaran algoritma yang dibangun dari beberapa model pengklasifikasi yang selanjutnya akan digunakan untuk mengklasifikasi data baru berdasarkan bobot prediksi yang dihasilkan sebelumnya[10]. Metode ini sudah dikembangkan sejak lama dan sampai saat ini terus diupayakan untuk dihasilkan metode *ensemble* yang lebih baik. Metode *ensemble* dianggap dapat menghasilkan akurasi yang lebih baik dibandingkan dengan penggunaan hanya dengan satu pengklasifikasi saja[10]. Metode *ensemble* memungkinkan untuk menggunakan beberapa model pengklasifikasi yang berbeda untuk mendapatkan gabungan model terbaik. Pemilihan pengklasifikasi ini ditentukan berdasarkan kebutuhan dan jenis dari data yang akan diolah.

2.6 Boosting

Konsep *ensemble* dengan *boosting* bekerja dengan melatih kelompok model secara berurutan dan kemudian menggabungkan seluruh model untuk membuat prediksi, kemudian model yang dihasilkan akan belajar dari kesalahan model sebelumnya [10]. Setiap iterasi menghasilkan model yang dihasilkan dari pembobotan proses sebelumnya. *Boosting* berfokus pada proses pembelajaran baru pada data dengan nilai akurasi yang rendah dari proses sebelumnya dan dilakukan dengan proses pelatihan yang berurutan. Data yang salah dari prediksi sebelumnya dikelompokkan sebagai nilai tetap dan akan digunakan pada proses prediksi selanjutnya sehingga nilai kesesuaian mencapai titik maksimal. Setelah seluruh proses prediksi dilakukan, langkah selanjutnya adalah menggabungkan seluruh model. *Boosting* mengubah model prediktif yang lemah menjadi prediktor kompleks yang andal. Tahapan proses pembelajaran ini adalah memprediksi regresi kemudian menghitung *error* dari residual dan terakhir proses mengolah residual [10].

2.7 Gradient Boosting

Gradient Boosting pertama kali diperkenalkan oleh J.H. Friedman [5]. *Gradient Boosted Trees* mampu membangun *decision tree* berdasarkan peningkatan dalam struktur pohon pada pembelajaran yang lemah untuk memperbaiki kesalahan pohon dan mencegah terjadinya potensi *overfitting*. Dalam membangun *decision tree*, dapat dilakukan penambahan jumlah iterasi yang sangat konservatif yang dapat menghasilkan dan meningkatkan kinerja model yang lebih baik. GBT mampu memecahkan masalah dengan menyesuaikan pembelajaran lemah dengan gradien negatif dari fungsi kerugian (*loss function*) dan meningkatkan pohon (*trees*) dengan parameter yang mewakili variabel *split* yang dipasang pada setiap *node* terminal pohon.

Gradient boosting mempunyai kemampuan untuk meningkatkan akurasi prediktif model. Konsep *gradient boosting* terletak pada pengembangannya yaitu memiliki ekspansi tambahan terhadap *fitting criterion* [5]. Metode awal menggunakan *bagging* yaitu mengambil sampel data secara acak, membangun algoritma dan menghitung rata-rata kemungkinan nilai *error* dan akurasi. Akurasi dihasilkan dapat ditingkatkan dengan cara membangun ulang model secara keseluruhan menggunakan input variabel yang baru.

Langkah-langkah pada teknik *gradient boosting*, dimulai dari memasukkan model ke dalam data dengan persamaan sebagai berikut

$$f_1(x) = y \tag{2.7}$$

Selanjutnya, model digunakan untuk menghitung *residual* pada proses sebelumnya, yaitu pada persamaan sebagai berikut

$$h_1(x) = y - F_1(x) \tag{2.8}$$

Kemudian, membuat model baru dengan persamaan sebagai berikut.

$$F_2(x) = F_1(x) + h_1(x) \tag{2.9}$$

Dari proses yang telah dilakukan, didapatkan model final yang merupakan gabungan dari kumpulan model sebanyak n iterasi sampai dihasilkan nilai error terkecil dari residual. Model final didefinisikan dengan persamaan sebagai berikut

$$F(x) = F_1(x) \rightarrow F_2(x) = F_1(x) + h_1(x) \ldots \rightarrow F_M(x) = F_{M-1}(x) + h_{M-1}(x) \quad (2.10)$$

Selanjutnya, model akhir dari *boosting* didefinisikan dengan persamaan sebagai berikut.

$$f(x) = \gamma_0 + \sum_{m=1}^{M} \gamma_m h_m(x) \quad (2.11)$$

$f_x = \gamma_0$ dan $f_m(x) = \gamma_m h_m(x)$ untuk $m = 1, 2, 3, \ldots, M$ dengan nilai $h_m(x) \in \{-1,1\}$. $\gamma_m(x)$ merupakan pengklasifikasian lemah, sedangkan γ_m adalah bobot pada tiap pengklasifikasian [6].

2.8 Extreme Gradient Boosting

Extreme Gradient Boosting merupakan teknik dalam *machine learning* untuk masalah regresi dan klasifikasi yang menghasilkan model prediksi dalam bentuk gabungan (*ensemble*) model prediksi yang lemah. Pembangunan model dilakukan dengan menggunakan metode *boosting*, yaitu dengan membuat model baru untuk memprediksi *error*/*residual* dari model sebelumnya. Model baru ditambahkan hingga tidak ada lagi perbaikan pada *error* yang dapat dilakukan. Algoritma ini dinamakan *gradient boosting* karena menggunakan *gradient descent* untuk memperkecil *error* saat membuat model baru [7].

Tabel 2.1 Parameter pada *Xtreme Gradient Boosting*

Parameter	Keterangan
Eta	*Learning rate* pada proses pelatihan
Gamma	Parameter *penalty* pada *regularization*
Max_depth	Tingkat kedalaman suatu pohon, semakin dalam pohon, maka proses akan semakin kompleks
Min_child_weight	Nilai minimal bobot yang dibutuhkan *child node*
Subsample	Jumlah sampel yang digunakan untuk proses pelatihan
Colsample_bytree	Jumlah sampel kolom untuk membuat *tree* baru

2.9 Evaluasi Klasifikasi

Evaluasi Klasifikasi bertujuan untuk mengetahui tingkat model gabungan dari berbagai metode pemilihan fitur. Ada beberapa cara untuk mengukur kinerja metode klasifikasi, antara lain menggunakan *F1-Score*, *Confusion Matrix*, *Precision* dan *Recall*.

Confusion Matrix adalah tabel yang terdiri dari jumlah baris data uji yang diprediksi benar dan salah oleh model klasifikasi, yang digunakan untuk menentukan kinerja model klasifikasi [13]. Tabel *Confusion Matrix* dapat membantu dalam mendapatkan nilai dari perhitungan karena berisi data prediksi positif dan negatif yang dihasilkan oleh sistem, dan data aktual positif dan negatif di dunia nyata [14]. Berikut adalah bentuk tabel *Confusion Matrix*.

Tabel 2. 2 Tabel *Confusion Matrix*

	Positif (Aktual)	**Negatif (Aktual)**
Positif (Sistem)	*True Positive*	*False Positive*
Negatif (Sistem)	*False Negative*	*True Negative*

Penjelasan variabel pada tabel *Confusion Matrix* sebagai berikut.

True Positive = perhitungan dari kelas positif sistem dan aktual yang positif.

True Negative = perhitungan dari kelas negatif sistem dan aktual yang negatif.

False Positive = perhitungan dari kelas positif sistem dan aktual yang negatif.

False Negative = perhitungan dari kelas negatif sistem dan aktual yang positif.

Precision merupakan rasio kategorisasi dokumen yang benar ke dalam kategori dengan jumlah total percobaan klasifikasi. *Recall* merupakan rasio klasifikasi dokumen yang benar ke dalam kategori dengan jumlah total data

berlabel di set pengujian [22]. Berikut rumus perhitungan *Precision* dan *Recall* [10].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2.12)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2.13)$$

Untuk menghindari perbedaan rasio yang cukup tinggi antara nilai *precision* dan *recall*. Maka, skor disamakan dengan menggunakan nilai *F1-Score* [15]. *F1-Score* akan menilai ketepatan sebuah *classifier* dengan rata-rata dari nilai *precision* dan *recall* [16].

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.14)$$