BAB 2
LANDASAN TEORI

2.1. Klasifikasi Teks

Klasifikasi teks adalah sebuah pekerjaan untuk menentukan apakah sebuah dokumen adalah milik dari sebuah kategori yang telah ditentukan sebelumnya. Tahapan dalam klasifikasi teks adalah *preprocessing*, seleksi fitur, *dictionary construction*, *features weighting*, generasi model klasifikasi, dan pengkategorian dokumen. **Error! Reference source not found.**

2.2. Seleksi Fitur

Seleksi fitur merupakan salah satu tahapan praproses pada *machine learning* yang bertujuan untuk mengurangi dimensi data dan menghilangkan data yang tidak relevan untuk meningkatkan efisiensi, performa sistem dan meningkatkan hasil akurasi. **Error! Reference source not found.** Banyaknya jumlah data dan jumlah fitur pada pengolahan data tentunya dapat menimbulkan permasalahan yang serius pada tingkat skalabilitas dan kinerja sistem untuk beberapa algoritma. Misalkan pada data dengan dimensi yang tinggi yaitu pada data yang memiliki ratusan bahkan ribuan fitur dapat berisi data-data yang tidak relevan yang bisa menurunkan kinerja dari algoritma. Oleh karena itu, pemilihan fitur menjadi sangat perlu dilakukan dalam menghadapi data-data dengan dimensi yang tinggi.

2.3. *Preprocessing*

Preprocessing adalah suatu proses pengubahan bentuk data yang belum terstruktur menjadi data yang sesuai dengan kebutuhan. Proses ini digunakan pada pengolahan teks atau dokumen seperti *sentiment analysis*, *summarization*, *clustering*, dan lain-lain. **Error! Reference source not found.** Beberapa proses yang digunakan adalah : *Cleaning* adalah poses untuk menghilangkan karakter yang tidak diperlukan seperti simbol, angka, dan *whitespace* berlebih. *Case folding* merupakan proses yang mengubah semua huruf dalam dokumen menjadi huruf kecil. *Tokenization* merupakan proses untuk memisahkan teks dapat berupa kalimat, paragraf, atau dokumen menjadi token-token. Token-token dipisahkan dengan tanda spasi antara kata. *Stemming* adalah proses untuk

menemukan bentuk dasar dari sebuah kata. Proses ini bertujuan untuk menormalisasi bentuk kata berdasarkan kata dasar. Proses normalisasi pada *stemming* mengindentifikasi dan menghapus prefiks serta suffiks dari sebuah kata. Pada penelitian ini, *tokenizer* dan *stemmer* yang digunakan untuk kata dalam bahasa Indonesia adalah *library* sastrawijs. *Stopword Removal* merupakan proses untuk menghilangkan kata yang tidak diperlukan, misalnya kata penghubung seperti "dan", "atau", "tapi" dan lainnya.

2.4. *Features Extraction*

Features Extraction merupakan tahapan untuk mengubah teks atau dokumen yang sebelumnya masih berbentuk kata kedalam bentuk vektor. **Error! Reference source not found.** Tahapan ini bertujuan agar teks atau dokumen dapat diklasifikasikan ke dalam kelas-kelas yang sudah ada. Pada penelitian ini, metode *features extraction* yang digunakan adalah *weighting TF-IDF* dengan *binary TF*.

2.4.1 TF

Menurut Karmayasa, *Term Frecuency* (TF) adalah algoritma pembobotan *heuristic* yang menentukan bobot dokumen berdasarkan kemunculan *term* (istilah). Semakin sering sebuah istilah muncul, semakin tinggi bobot dokumen untuk istilah tersebut, dan sebaliknya. Terdapat 4 macam algoritma TF : **Error! Reference source not found.**

1. *Binary TF*

Binary TF menyeragamkan bobot dokumen terhadap istilah dengan memberi nilai 0 dan 1. Nilai 1 menyatakan suatu istilah muncul minimal satu kali dalam suatu dokumen, sementara nilai 0 menyatakan suatu istilah tidak muncul sama sekali.

2. *Raw TF*

Raw TF menentukan bobot suatu dokumen terhadap istilah dengan menghitung frekuensi kemunculan suatu istilah tersebut pada dokumen.

3. *Logarithmic TF*

Logarithmic TF mengurangi tingkat kepentingan kemunculan kata dalam menghitung bobot dokumen terhadap suatu istilah dengan melakukan log terhadap TF.

4. *Augmented TF*

Augmented TF menyeragamkan bobot dokumen terhadap istilah dengan memberikan range antara 0.5 hingga 1 sebagai bobot dokumen.

2.4.2 IDF

Karen Spark Jones dan Robertson pada tahun 1972 menyusun interpretasi *statistic* kekhususan istilah yang disebut IDF. *Inverse Document Frecuency* (IDF) merupakan nilai kepentingan suatu dokumen terhadap term. Biasanya istilah "adalah" begitu sering muncul, sehingga kata ini bukan kata kunci yang baik untuk membedakan dokumen dan istilah yang relevan. Oleh karena itu IDF mengurangi bobot istilah yang sangat sering muncul dalam kumpulan dokumen. Formula untuk menghitung IDF dapat dilihat pada persamaan (2.1). **Error! Reference source not found.**

$$idf_j = \log\left(\frac{D}{df_j}\right) + 1 \tag{2.1}$$

Keterangan :

D = Jumlah semua dokumen

df_j = Jumlah dokumen yang mengandung sebuah kata j

2.4.3 TF-IDF

Menurut Maarif, metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat. Metode ini akan menghitung nilai *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Formula untuk menghitung TF-IDF dapat dilihat pada persamaan (2.2). **Error! Reference source not found.**

$$w_{ij} = tf_{ij} \times idf_j \tag{2.2}$$

2.5. Algoritma Naive Bayes

Naive Bayes merupakan suatu algoritma yang dapat mengklasifikasikan suatu variabel tertentu dengan menggunakan metode probabilitas dan statistik. Naive Bayes menggunakan cabang matematika yang dikenal dengan teori probabilitas untuk mencari peluang terbesar dari kemungkinan klasifikasi, dengan cara melihat frekuensi tiap klasifikasi pada data training. Berikut adalah Alur penyelesaian dari metode Naive Bayes : **Error! Reference source not found.**

1. Baca Vektor TF-IDF *Training Set*
2. Hitung *Mean* dan Standar Deviasi

Persamaan (2.3) berikut adalah rumus yang digunakan untuk menghitung nilai rata – rata (*mean*) :

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \text{ atau } \mu = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n} \quad (2.3)$$

Keterangan :

μ = rata – rata hitung (*mean*)
x_i = nilai sample ke–i
n = jumlah sampel

Kemudian persamaan (2.4) berikut ini adalah rumus yang digunakan untuk menghitung simpangan baku.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n - 1}} \quad (2.4)$$

Keterangan:

σ = standar deviasi
x_i = nilai x ke–i
μ = rata-rata hitung
n = jumlah sampel

3. Hitung Probabilitas *Prior* Setiap Kelas

Untuk mengetahui nilai probabilitas *prior* setiap kelas kita harus menghitung jumlah data yang sesuai dari kategori yang sama lalu dibagi dengan jumlah data pada kategori tersebut.

4. Hitung Nilai Distribusi Gaussian

Persamaan (2.5) berikut ini adalah rumus *probability density function* yang digunakan untuk mencari nilai distribusi gaussian. **Error! Reference source not found.**

$$pdf(x, mean, sd) = \frac{1}{\sqrt{2 \times \pi} \times sd} \times e^{-\frac{(x - mean)^2}{2 \times sd^2}} \quad (2.5)$$

5. Hitung Probabilitas Akhir

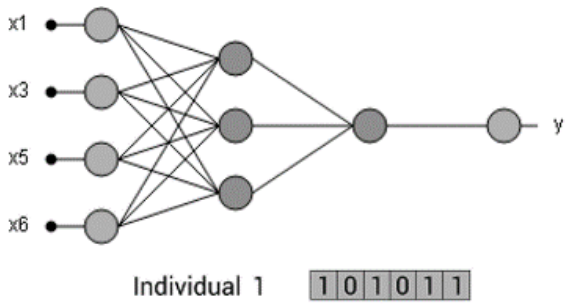
Probabilitas akhir didapatkan melalui perhitungan probabilitas *prior* dan distribusi gaussian dari masing-masing fitur kata pada dokumen *testing set*. Perhitungan probabilitas akhir dihitung menggunakan persamaan (2.6) berikut.

$$P(C|x_1, \dots, x_n) = P(c) \times P(x_1, \dots, x_n|C) \quad (2.6)$$

2.6. Algoritma Genetika

Algoritma genetika merupakan algoritma evolusioner yang paling populer. Algoritma ini menggunakan prinsip dasar dari seleksi alam yang diperkenalkan oleh Charles Darwin. Algoritma genetika diterapkan sebagai pendekatan untuk mengidentifikasi pencarian nilai dan solusi bagi berbagai permasalahan optimasi. **Error! Reference source not found.** Intisari dari algoritma genetika mencakup pengkodean fungsi optimasi sebagai *array* berisi bit-bit atau karakter berupa *string* untuk menggambarkan kromosom, operasi manipulasi *string* dengan operator genetik, dan seleksi sesuai dengan *fitness*, dengan tujuan untuk menemukan solusi yang baik dan optimal terhadap masalah yang sedang dihadapi.

Proses inisialisasi pada penelitian ini menggunakan pembulatan fungsi *random* pada setiap fitur kata, jika nilai *random* lebih besar atau sama dengan 0.5 maka simpan nilai 1 pada *knapsack*, jika nilai *random* kurang dari 0.5 maka simpan nilai 0 pada *knapsack*. Representasi kromosom dalam bentuk *neural network* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Kromosom dalam Representasi *Neural Network*

Proses berikutnya merupakan langkah yang berulang dari operator genetik. Algoritma genetika memiliki tiga operator genetik utama yaitu :

1. Seleksi

Seleksi ialah penggunaan solusi dengan nilai *fitness* yang tinggi untuk lulus ke generasi berikutnya dengan harapan untuk menemukan solusi terbaik. Proses seleksi populasi yang akan dilakukan menggunakan metode seleksi sebanding dengan nilai sesuaian (*fitness*), metode ini diimplementasikan dengan metode seleksi *elitism*.

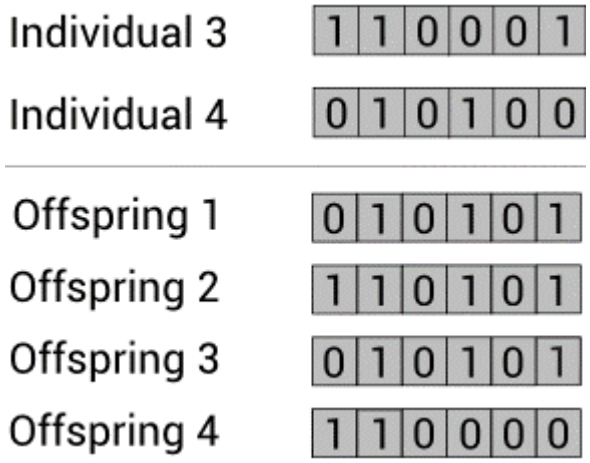
Seleksi *elitism* adalah seleksi dimana individu – individu yang terpilih untuk menjadi generasi selanjutnya berdasarkan pada nilai fitness tertinggi. Metode seleksi *elitism* bekerja dengan mengumpulkan semua individu baik populasi maupun offspring dalam satu penampungan. Individu terbaik dalam penampungan akan lolos untuk masuk dalam generasi berikutnya. Metode seleksi elitism menjamin individu yang terbaik akan selalu lolos. **Error! Reference source not found.**

2. Crossover

Crossover adalah proses menukar bagian dari solusi (kromosom) dengan bagian "*parent*" lain untuk menghasilkan jenis kromosom yang berbeda yang mungkin menjadi solusi baru untuk menyelesaikan permasalahan. Peran utamanya adalah untuk memberikan pencampuran solusi dan konvergensi dalam sub ruang (menghasilkan solusi yang baru).

Metode *crossover* yang digunakan pada penelitian ini adalah metode *uniform crossover*. Setelah operator seleksi telah memilih setengah dari populasi, operator *crossover* menggabungkan kembali individu yang dipilih untuk menghasilkan populasi baru. Operator ini mengambil individu terpilih dan menggabungkan gen mereka untuk

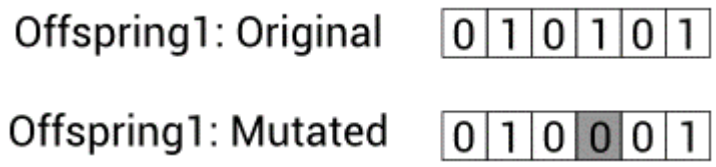
mendapatkan keturunan (*offspring*) untuk populasi baru, sampai populasi baru memiliki ukuran yang sama dari populasi yang lama dengan peluang *crossover* awal adalah 0.8, peluang ini dapat ditentukan oleh *user* nantinya. Contoh operasi *crossover* secara visual dapat dilihat pada Gambar 2.2.



Gambar 2.2 Operasi *Crossover*

3. Mutasi

Mutasi merupakan pergantian salah satu bagian solusi yang dipilih secara acak, yang meningkatkan keragaman dari populasi dan menghasilkan mekanisme untuk menghindari optimum lokal. Operator *crossover* dapat menghasilkan keturunan yang sangat mirip dengan orang tuanya. Hal ini memungkinkan generasi baru yang dihasilkan memiliki tingkat keberagaman yang rendah. Operator mutasi memecahkan masalah ini dengan mengubah nilai beberapa gen dalam *offspring* secara acak menggunakan peluang mutasi yang awalnya sebesar 0.10, peluang ini dapat ditentukan oleh *user* nantinya. Contoh operasi mutasi dapat dilihat pada Gambar 2.3.



Gambar 2.3 Operasi Mutasi

2.7. *Confusion Matrix*

Metode *confusion matrix* merepresentasikan hasil evaluasi model dengan menggunakan tabel matriks, jika dataset terdiri dari dua kelas, kelas pertama dianggap positif, dan kelas kedua dianggap negatif. Evaluasi menggunakan *confusion matrix* menghasilkan nilai akurasi, *precision*, dan *recall*. Akurasi dalam klasifikasi merupakan presentase ketepatan *record* data yang diklasifikasikan secara benar setelah dilakukan pengujian pada hasil klasifikasi. *Precision* atau *confidence* merupakan proporsi kasus yang diprediksi positif yang juga positif benar pada data yang sebenarnya. *Recall* atau sensitivity merupakan proporsi kasus positif yang sebenarnya yang diprediksi positif secara benar. **Error! Reference source not found.** *Confusion matrix* secara visual dapat dilihat pada Tabel 2.1.

Tabel 2.1 Confusion Matrix

Predicted Class	*Actual Class*	
	(+)	(-)
(+)	*True positives*	*False positives*
(-)	*False negatives*	*True negatives*

True positive (tp) merupakan jumlah *record* positif dalam dataset yang diklasifikasikan positif. *True negative* (tn) merupakan jumlah *record* negatif dalam dataset yang diklasifikasikan negatif. *False positive* (fp) merupakan jumlah record negatif dalam dataset yang diklasifikasikan potitif. *False negative* (fn) merupakan jumlah *record* positif dalam dataset yang diklasifikasikan negatif.

2.8. *Performance Measure*

Performance measure atau pengukuran performa dilakukan agar memperoleh tingkat kesesuain dari hasil klasifikasi dari sistem yang dibangun. Beberapa cara yang sering digunakan dalam pengukuran performa adalah dengan menghitung akurasi, *recall*, *precission* dan *f1-score*, **Error! Reference source not found.** dengan menggunakan persamaan-persamaan sebagai berikut : (2.7) (2.8) (2.9) (2.10)

$$Accuracy = \frac{total\ benar}{N} \times 100\% \quad (2.7)$$

$$Precision_i = \frac{tp_i}{tp_i + fp_i} \quad (2.8)$$

$$Recall_i = \frac{tp_i}{tp_i + fn_i} \quad (2.9)$$

$$F1\text{-}Score_i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i} \quad (2.10)$$

$Precision_i$ merupakan presentase dari nilai item yang diprediksi benar dan terbukti benar. Nilai dari *precision* digunakan untuk mengukur seberapa tepat sistem melakukan prediksi. $Recall_i$ adalah presentase dari nilai item yang memang benar dan berhasil diprediksi benar. Nilai *recall* digunakan untuk mengukur seberapa banyak item yang memang diprediksikan benar. $F1\text{-}score_i$ adalah presentase dari nilai item yang merupakan kombinasi dari *precison* dan *recall.* **Error! Reference source not found.**

Tingkat akurasi dapat dikelompokkan sebagai berikut : **Error! Reference source not found.**

Akurasi 0.90 – 1.00 = *Excellent classification*

Akurasi 0.80 – 0.90 = *Good classification*

Akurasi 0.70 – 0.80 = *Fair classification*

Akurasi 0.60 – 0.70 = *Poor classification*

Akurasi 0.50 – 0.60 = *Failure*

2.9. *K-Fold Cross Validation*

K-fold cross validation merupakan salah satu cara untuk untuk menemukan kombinasi *data training* yang baik untuk dijadikan model klasifikasi. Pada penelitian ini

k-fold dilakukan dengan cara membagi vektor TF-IDF dari *data training* menjadi 10 bagian dan secara bergantian setiap bagiannya menjadi *validation set*.