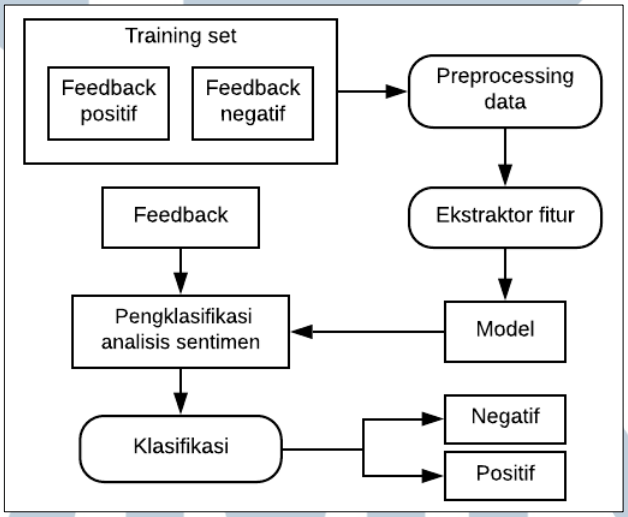
BAB II

LANDASAN TEORI

2.1 Sentiment Analysis

Analisis sentimen merupakan proses yang mengotomasikan penambangan dan klasifikasi pendapat, pandangan, emosi, dan sentimen *dataset* teks yang tidak terstruktur dengan bahasa mesin dan pemrograman komputer (Fiarni dkk., 2016). Dalam analisis sentimen, teks diklasifikasikan ke dalam kategori seperti "positif" atau "negatif" atau "netral". Secara garis besar struktur analisis sentimen *user feedback* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur Analisis Sentiment

2.2 Data Preprocessing

User feedback mengandung banyak kata dan kalimat, yang dapat diekspresikan dengan berbagai cara. Karena itu sebelum melakukan analisis sentimen, data yang tidak diperlukan dan berlebihan harus dihapus. Beberapa tahap

preprocessing data *user feedback* yang dilakukan dalam penelitian ini adalah penghapusan *user feedback* atau kata non bahasa Indonesia, angka, URL dan alamat email, serta karakter spesial termasuk *emoticon*.

2.3 Naïve Bayes Classifiers

Naïve Bayes (NB) *classifiers* adalah pengklasifikasi probabilistik yang berada dalam cakupan teknik *machine learning*. Teknik pengklasifikasi ini didasarkan pada penerapan teorema Bayes dengan asumsi kuat tentang ketergantungan antara tiap pasang fitur. Dalam NB *classifiers*, sebuah atribut merupakan kata yang berarti dalam teks dimana klasifikasi dilakukan (Wagh dkk., 2018). Dengan asumsi ini, pengklasifikasi dapat ditetapkan sebagai Persamaan 2.1.

$$c(d) = \arg\max_{c \in C} P(c) \prod_{i=1}^{m} P(a_i|c) \quad \dots (2.1)$$

dimana *d* merupakan dokumen *test*, *c(d)* merupakan *class* dari *d*, dan *c* merupakan label *class*.

Kelebihan utama dari NB *classifiers* adalah penanganan data yang terdistorsi karena dengan estimasi probabilitas bersyarat data dirata-ratakan, nilai *null* diabaikan dan fitur yang tidak relevan didistribusi secara seragam sehingga tidak memiliki pengaruh yang signifikan terhadap hasil klasifikasi (Ismail dkk., 2016).

Umumnya, tiga model distribusi, yaitu model Bernoulli, model multinomial, dan model Poisson, telah digabung ke dalam *framework* Bayesian dan masing-masing telah menghasilkan pnegklasifikasi Bernoulli Naïve Bayes (BNB), multinomial Naïve Bayes (MNB), dan Poisson Naïve Bayes (PNB).

2.4 Multinomial Naïve Bayes Classifiers

Multinomial Naïve Bayes mengembangkan penggunaan algoritma NB dan merupakan model berbasis frekuensi untuk klasifikasi teks, dimana suatu dokumen direpresentasikan oleh kumpulan kata yang muncul dari dokumen tersebut (Song dkk., 2017). MNB termasuk salah satu pendekatan klasifikasi Naïve Bayes yang paling dikenal, dimana *term frequency* digunakan untuk merepresentasikan dokumen (Tang dan He, 2016). Klasifikasi teks dengan MNB yang telah dimodifikasi oleh Song, dkk dapat dilihat pada Persamaan 2.2, dimana dokumen *d* ditunjukkan oleh vektor kata $< w_1, w_2, \ldots, w_m >$.

$$c(d) = \arg\max_{c \in C}[\log P(c) + \sum_{i=1}^{m} WT_i f_i \log P(w_i|c)] \quad \ldots (2.2)$$

dimana *m* merupakan jumlah kata yang berbeda dalam dokumen, w_i (i = 1,2,…, m) merupakan kata ke-*i* yang muncul dalam dokumen, dan f_i (i = 1,2,…, m) merupakan frekuensi dari w_i dalam *d*. Probabilitas kelas $P(c)$ dapat dihitung dengan Persamaan 2.3, sedangkan probabilitas kondisional $P(w_i|c)$ dapat dihitung menggunakan Persamaan 2.4.

$$P(c) = \frac{\sum_{j=1}^{n} \delta(c_j,c)+1}{n+l} \quad \ldots (2.3)$$

dimana *n* adalah jumlah dokumen *training*, *l* adalah jumlah kelas, c_j adalah kelas dari dokumen *training* ke-*j*. Fungsi biner untuk $\delta(c_j, c)$ dapat didefinisikan menggunakan Persamaan 2.5.

$$P(w_i|c) = \frac{\sum_{j=1}^{n} WT_i f_{ji} \delta(c_j,c)+1}{\sum_{i=1}^{m}\sum_{j=1}^{n} WT_i f_{ji} \delta(c_j,c)+m} \quad \ldots (2.4)$$

dimana f_{ji} adalah frekuensi w_i pada dokumen *training* ke-j dan WT_i merupakan bobot kata dari kelas *c*, yang diperoleh dari proses *attribute weighting* dan *feature selection*. Semakin besar bobot suatu kata maka semakin relevan kata tersebut dengan kelas *c*.

$$\delta(c_j, c) = \begin{cases} 1, \text{if } c_j = c \\ 0, \text{otherwise} \end{cases} \quad \dots (2.5)$$

2.4.1 Attribute Weighting

Song dkk. (2017) mengusulkan cara pembobotan atribut baru yang memiliki pendekatan serupa dengan *gain ratio based weighting* (GRW). Sebelumnya GRW merupakan pendekatan metode pembobotan atribut yang paling sering digunakan. Akan tetapi GRW memiliki kekurangan dimana jumlah atribut yang sama digunakan untuk menghitung bobot setiap *classs*. Cara pembobotan atribut yang diusulkan Song dkk. dapat dilihat pada Persamaan 2.6.

$$WT_{i,c} = \frac{IGR(c,w_{i,c}) \times m_c}{\sum_{i=1}^{m_c} IGR(c,w_{i,c})} \quad \dots (2.6)$$

dimana *c* adalah label dari kelas *v* ($\in \{positif, negatif\}$), m_c adalah jumlah kata berbeda dalam dokumen D_c, dan $IGR(c, w_{i,c})$ adalah rasio perolehan informasi dari $w_{i,c}$. Rasio perolehan informasi dapat dihitung dengan Persamaan 2.7.

$$IGR(c, w_{i,c}) = \frac{IG(c,w_{i,c})}{H(w_{i,c})} \quad \dots (2.7)$$

dimana $IG(c,w_{i,c})$ dan $H(w_{i,c})$ merupakan perolehan informasi dan entropi informasi dari $w_{i,c}$, yang didapat dari Persamaan 2.8 dan 2.9.

$$IG(c, w_{i,c}) = H(c) - H(c|w_{i,c}) \quad \dots (2.8)$$

dimana $H(c)$ merupakan entropi dari D dan $H(c|w_{i,c})$ merupakan entropi kondisional dari D untuk w_i. $H(c)$ dan $H(c|w_{i,c})$ masing-masing dapat dihitung dengan Persamaan 2.10 dan 2.11.

$$H(w_{i,c}) = -\sum_V \frac{|D_{v,c}|}{|D|} \log_2 \frac{|D_{v,c}|}{|D|} \quad \dots (2.9)$$

dimana $|D_{v,c}|$ adalah ukuran dari D_c dalam hal angka dari $w_{i,c}$ dimana nilai *v* ($\in \{0, \bar{0}\}$).

$$H(c) = -P(c) \log_2 P(c) \quad \dots (2.10)$$

dimana *c* adalah kelas target dan $P(c)$ adalah probabilitas dari kelas *c* dan dapat dihitung menggunakan Persamaan 2.3.

$$H(c|w_{i,c}) = -\sum_V \frac{|D_{v,c}|}{|D|} \sum_c P(c|v) \log_2 P(c|v) \quad \dots (2.11)$$

dimana $|D_v|$ adalah ukuran dari *D* dan $P(c|v)$ adalah probabilitas bersyarat kelas *c* terhadap nilai *v* dimana nilai *v* ($\in \{0, \bar{0}\}$).

2.4.2 Feature Selection

Song dkk. (2018) mengusulkan cara baru seleksi fitur, yang menggunakan selisih antara bobot postif dan bobot negatif semua kata untuk memodifikasi bobot kata-kata tidak berarti. Perhitungan selisih bobot dapat dilihat pada Persamaan 2.12.

$$WD_i = \begin{cases} |WT_{i,p} - WT_{i,n}|, \text{if } w_i \in D_p \text{and } w_i \in D_n \\ WT_{i,c}, \text{otherwise} \end{cases} \quad \dots (2.12)$$

dimana $WT_{i,p}$ adalah bobot positif tiap kata dan $WT_{i,n}$ adalah bobot negatif tiap kata.

Setelah perhitungan selisih antara bobot positif dan bobot negatif, maka rata-rata dari selisih bobot dapat dihitung dengan Persamaan 2.13.

$$Avg_{WD} = \frac{\sum_{i=1}^{m} WD_i}{m} \quad \text{… (2.13)}$$

dimana WD_i adalah selisih bobot kata *i* dan *m* adalah jumlah kata yang berbeda dalam dataset.

Seluruh bobot tiap kata kemudian dibandingkan dengan rata-rata selisih bobot di tiap dataset D_c. Jika nilai bobot lebih besar dari rata-rata selisih bobot, maka akan diubah menjadi 0. Selain itu akan tetap disimpan seperti pada Persamaan 2.14.

$$WT_{i,c} = \begin{cases} 0, \text{if } WT_{i,c} > Avg_{WD} \\ WT_{i,c}, \text{otherwise} \end{cases} \quad \text{… (2.14)}$$

Terakhir, label kelas dapat diprediksi dengan menggunakan Persamaan 2.2

2.4.3 Laplace Smoothing

Probabilitas bersyarat $P(c|v)$ pada Persamaan 2.11 dapat menimbulkan masalah karena dapat memberikan probabilitas bernilai 0 untuk dokumen yang memiliki kata yang tidak pernah ditemui sebelumnya. Cara umum untuk mengatasi masalah tersebut adalah dengan menggunakan Laplace Smoothing. Teknik Laplace Smoothing ini dilakukan dengan mengasumsikan bahwa dataset *training* sangat besar sehingga apabila menambahkan 1 ke setiap hitungan yang kita perlukan hanya akan membuat perbedaan yang diabaikan dalam nilai probabilitas estimasi nol (Medhekar dkk., 2013). Perhitungan Laplace Smoothing dapat dilihat pada Persamaan 2.15.

$$P(w|c) = \frac{count(w,c)+1}{count(c)+|V|} \quad \ldots (2.15)$$

dimana $count(w,c)$ adalah jumlah kata *w* dalam kelas *c*, $count(c)$ adalah jumlah kata dalam kelas *c*, dan $|V|$ adalah jumlah kosa kata dalam dokumen *training*.

2.5 Confusion Matrix

Confusion Matrix digunakan untuk menyimpulkan kinerja klasifikasi dari suatu pengklasifikasi terhadap suatu data uji (Ting, 2017). Confusion Matrix merupakan matriks dua dimensi, satu dimensi diindeks oleh kelas sebenarnya dari suatu objek dan dimensi lainnya diindeks oleh kelas yang ditentukan oleh pengklasifikasi. *Confusion matrix* secara umum dapat dilihat pada Tabel 2.1.

Tabel 2.1 Confusion Matrix

Prediksi / Aktual	Positif	Negatif
Positif	TP	FN
Negatif	FP	TN

TP, TN, FP, dan FN merupakan nilai dari *true positive*, *true negative*, *false positive*, dan *false negative*. Keempat nilai tersebut kemudian digunakan untuk perhitungan performa model. Nilai *accuracy* digunakan untuk mengukur ketepatan prediksi yang dihasilkan. Nilai *accuracy* sendiri tidak cukup untuk mengukur performa model, maka dari itu digunakan metric pengukuran lain untuk membantu pengukuran performa model. Nilai *precision*, *recall*, dan F1 *score* digunakan untuk mendukung pengukuran performa model. Nilai *precision* mengukur perbandingan prediksi positif yang benar (*true positif)* terhadap total prediksi positif. Nilai *recall*

mengukur perbandingan prediksi positif yang benar (*true positif)* terhadap semua prediksi di kelas aktual. Sedangkan F1 *score* merupakan rata-rata berbobot dari nilai *precision* dan *recall*. Perhitungan nilai accuracy, *precision, recall,* dan F1 *score* dapat dilihat pada Persamaan 2.16, 2.17, 2.18, 2.19.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \dots (2.16)$$

$$Precision = \frac{TP}{(TP+FP)} \quad \dots (2.17)$$

$$Recall = \frac{TP}{(TP+FN)} \quad \dots (2.18)$$

$$F1 = \frac{(2 \times Precision \times Recall)}{(Precision+Recall)} \quad \dots (2.19)$$