BAB 2

LANDASAN TEORI

2.1 Text Classification

Text Classification merupakan kegiatan mengklasifikasikan secara otomatis dokumen berupa teks ke dalam kategori-kategori yang telah ditetapkan sebelumnya (Korde & Mahender, 2012). Text Classification diawali dengan merepresentasikan dokumen/data sebagai vektor dalam sebuah vektor dengan dimensi tinggi, dimana masing-masing dimensi menyesuaikan dengan nilai fitur, atau secara spesifik, sebuah *term*. Kemudian program mengklasifikasi dokumen sesuai dengan kategori yang tepat dengan menggunakan vektor (Nedjah, et al., 2009). Tujuan dari *text classification* adalah untuk melatih pengklasifikasi berdasarkan model yang sudah dibuat sebelumnya kemudian contoh yang tidak diketahui dikategorikan secara otomatis (Gaikwad, et al., 2014).

2.2 Sentimen Analisis

Sentimen analisis merupakan salah satu cara untuk melakukan analisis teks dengan pendekatan analitis. (Rajput, et al., 2019). Sentimen analisis adalah suatu proses yang mengotomasikan penambangan dan klasifikasi pendapat, pandangan, emosi, dan sentiment *dataset* teks yang tidak terstruktur dengan Bahasa mesin dan pemrograman komputer (Fiarni, et al., 2016). Sentimen analisis dilakukan dengan mengklasifikasi teks ke dalam kategori atau jenis seperti positif, negatif dan netral (Boiy & Moens, 2008)

2.3 Text Preprocessing

Text preprocessing merupakan tahapan awal dalam *pipeline* dari Natural Language Processing (NLP). *Text preprocessing* berpotensi besar pada performa akhir (Camacho-Collados & Pilehvar, 2018). *Text preprocessing* merupakan suatu cara untuk mengurangi noise dari data yang akan digunakan pada tahapan berikutnya (Kumar & Harish, 2018). *Preprocessing* merupakan tahapan yang penting untuk tahapan selanjutnya yaitu klasifikasi (Srividhya & Anitha, 2010). Berikut beberapa prosedur yang biasa digunakan dalam tahap *preprocessing*.

2.3.1 Case Folding

Dalam dokumen atau data yang akan digunakan seringkali terdiri atas huruf kapital dan huruf kecil sehingga tidak memiliki kesamaan. Hal ini dapat di sebabkan karena kesalahan dalam penulisan, Tujuan dari *case folding* yaitu untuk merubah semua huruf pada dokumen atau data yang digunakan menjadi huruf kecil, missal kata “Baik” menjadi “baik” (Rosid, et al., 2020).

2.3.2 Remove Number and Punctuation

Remove number and punctuation sering kali dilakukan karena angka dan tanda baca dianggap tidak informatif. Namun, dalam beberapa kasus terkadang angka dan tanda baca menjadi penting seperti penggunaan *hashtag* yang muncul di twitter dan penggunaan angka pada referensi kode Undang-Undang (Denny & Spirling, 2017).

2.3.3 Stop Word Removal

Stop Word Removal dilakukan dengan menghilangkan kata yang paling sering digunakan namun tidak berguna dalam Information Retrieval (IR) dan Text

Mining. Biasanya kata-kata yang termasuk dalam *stop-word* seperti kata ganti, preposisi, dan kata hubung (Srividhya & Anitha, 2010).

2.3.4 Stemming

Stemming digunakan untuk mengetahui kata dasar dari sebuah kata. Stemming mengubah kata yang berimbuhan menjadi kata dasar. Hipotesis dibalik *stemming* adalah kata-kata dengan kata dasar yang sama kebanyakan menggambarkan konsep yang sama atau relatif dekat dalam sebuah teks (Srividhya & Anitha, 2010).

2.3.5 Document Indexing

Tujuan utama dilakukannya *document indexing* adalah untuk meningkatkan efisiensi dengan mengekstraksi dari dokumen yang menghasilkan sekumpulan *term* yang dipilih untuk digunakan dalam mengindeks dokumen. *Document indexing* terdiri dari membuat daftar kata kunci (*vocabulary*) yang sesuai berdasarkan keseluruhan dokumen kemudian menetapkan bobot pada kata kunci tersebut pada masing-masing dokumen sehingga mengubah setiap dokumen kedalam bentuk vektor bobot kata kunci. Biasanya bobot terkait dengan frekuensi kemunculan *term* dalam dokumen dan jumlah dokumen yang menggunakan *term* tersebut (Srividhya & Anitha, 2010). Salah satu d*ocument indexing* yaitu dengan menggunakan Term Frequency-Inverse Document Frequency (TF-IDF).

2.4 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF adalah metode yang sering digunakan untuk membangun *vector space model* dalam Information Retrieval. TF-IDF mengevaluasi pentingnya kata

dalam dokumen. (Erra, et al., 2014). TF-IDF merupakan gabungan dari Term Frequency (TF) dan Inverse Document Frequency (IDF). TF digunakan untuk mengukur berapa kali sebuah kata muncul dalam sebuah dokumen. IDF digunakan untuk menghitung berapa kali sebuah kata yang muncul di berbagai dokumen yang dianggap sebagai kata umum yang dinilai tidak penting (Hakim, et al., 2014). Untuk mendapatkan bobot dari sebuah *term*/kata dapat digunakan rumus sebagai berikut (Paltoglou & Thelwall, 2010).

$$W(t) = TF(t) * IDF(t) \quad (2.1)$$

$$TF(t) = \frac{Jumlah\ kemunculan\ term\ t\ dalam\ dokumen}{jumlah\ term\ dalam\ dokumen} \quad (2.2)$$

$$IDF(t) = \log\left(\frac{Jumlah\ seluruh\ dokumen}{jumlah\ dokumen\ yang\ mengandung\ term\ t}\right) \quad (2.3)$$

Keterangan:

W(t) = bobot dari TF-IDF.

TF(t) = banyaknya *term* t yang dicari pada sebuah dokumen.

IDF(t) = nilai Inverse Document Frequency dari *term* t.

2.5 Logistic Regression

Logistic Regression (LR) adalah metode statistic yang mirip dengan Linear Regression di mana Logistic Regression menggunakan persamaan yang memprediksi hasil untuk variabel biner *y* (variabel dependen) dari satu atau lebih variabel respon variabel *x* (variabel independen) (A.DIGangi & K.Moore, 2013). Secara matematis fungsi Logistic Regression memperkirakan fungsi linear berganda yang didefinisikan sebagai berikut:

$$\text{Logit}(S) = \log_b \frac{p}{1-p} = b0 + b1M1 + b2M2 + b3M3 + bnMn \quad (2.4)$$

Keterangan:

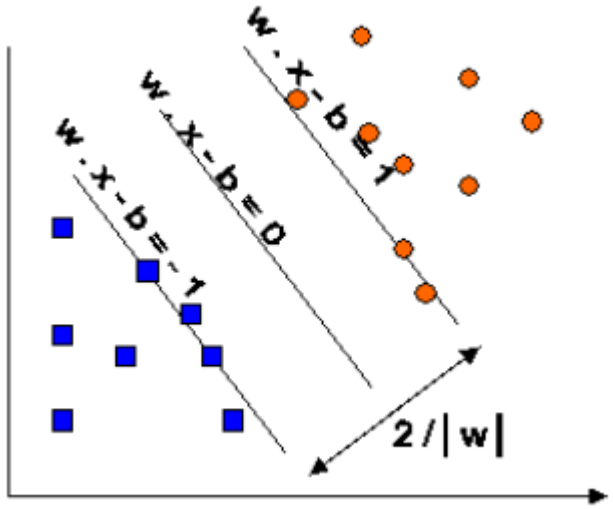
S = probabilitas keberadaan fitur yang diminati

M = nilai predictor

B = parameter model

2.6 Support Vector Machine

Support Vector Machine (SVM) merupakan metode klasifikasi data linear dan non-linear. SVM menggunakan pemetaan non-linear untuk mengubah data pelatihan menjadi dimensi yang lebih tinggi dan mencari *hyperplane* pemisah linear yang optimal (Patra & Singh, 2013). SVM mencari sebuah *hyperplane* yang dapat dengan baik memisahkan data sesuai dengan kelasnya.



Gambar 2. 1 Jarak Maksimal Hyperlane pada SVM dengan Sampel dari 2 Kelas **(Srivastava & Bhambhu, 2009)**

Pada dasarnya SVM menggunakan prinsip Linear Classifier, namun dalam domain dunia nyata jarang yang bersifat masalah linear. kemudian dikembangkan

dengan memasukkan *kernel function* agar SVM tetap dapat digunakan saat data tidak dapat dipisahkan secara linear (agar dapat bekerja pada permasalahan non-linear) (Nugroho, et al., 2003). Berikut merupakan beberapa *kernel function* yang paling umum digunakan. (Srivastava & Bhambhu, 2009).

1. Linear Kernel
   $$K(x_i, x_j) = {x_i}^T x_j \quad (2.5)$$
2. Polynomial Kernel
   $$K(x_i, x_j) = (\gamma {x_i}^T x_j + r)^d, \gamma > 0 \quad (2.6)$$
3. Radian Basis Function (RBF) Kernel
   $$K(x_i, x_j) = \exp\left(-\gamma \left|\left|{x_i}^T x_j\right|\right|^2\right), \gamma > 0 \quad (2.7)$$
4. Sigmoid Kernel
   $$K(x_i, x_j) = \tanh(\gamma\, {x_i}^T x_j + r) \quad (2.8)$$

   Di mana γ, r, dan d adalah parameter kernel.

2.7 Multinomial Naïve Bayes Algorithm

Multinomial Naïve Bayes merupakan metode yang menggunakan probabilitas yang termasuk dalam *supervised learning* untuk kasus klasifikasi teks. Metode ini mengikuti prinsip dari distribusi multinomial dalam probabilitas bersyarat (Manning, et al., 2009). Meskipun mengikuti prinsip dari distribusi multinomial, algoritma Multinomial Naïve Bayes dapat diterapkan untuk kasus teks dengan mengkonversinya ke bentuk nominal yang dapat dihitung dengan nilai integer. Rumus probabilitas dijelaskan pada rumus 9 (Abdurrahman, et al., 2019).

$$P(c|d) \propto \mathrm{P(c)} \prod_{1 \le k \le n_d} P(t_k|c) \quad (2.9)$$

P(c|d) merupakan probabilitas kondisional dari kata t_k yang berada dalam dokumen yang memiliki kelas c. Dalam persamaan P(t_k|c) merupakan probabilitas *likelihood* t_k di kelas c. P(c) merupakan probabilitas prior dari dokumen tersebut

muncul di kelas c. Untuk menentukan kelas yaitu dengan membandingkan hasil probabilitas posterior yang diperoleh, untuk kelas dengan probabilitas posterior terbesar adalah kelas yang dipilih sebagai hasil prediksi. Rumus probabilitas prior dapat dilihat pada rumus 10.

$$P(c) = \frac{N_c}{N} \tag{2.10}$$

N_c merupakan penjumlahan dari kategori c, N merupakan jumlah dari semua kategori. Rumus probabilitas *likelihood* dapat dilihat pada rumus 11.

$$P(t_k|c) = \frac{T_{tc}}{\sum t' \epsilon V \, T_{ct'}} \tag{2.11}$$

T_{tc} merupakan jumlah kemunculan kata t dalam dokumen yang memiliki kelas c, dan $\sum t' \epsilon V \, T_{ct'}$ merupakan jumlah total kemunculan semua kata di kelas c.

2.8 Ensemble Learning

Ensemble Learning bertujuan untuk menggabungkan keputusan dari beberapa algoritma pembelajaran untuk meningkatkan hasil akurasi (terutama algoritma pembelajaran yang lemah) dan membuat model yang lebih baik untuk melakukan prediksi. Penggunaan metode *ensemble* lebih baik jika dibandingkan dengan hanya menggunakan satu algoritma pembelajaran (Onan, et al., 2016). Secara umum *ensemble learning* dibagi menjadi dua kelompok yaitu dependen dan independen. Dalam metode dependen *output* dari hasil pembelajaran digunakan untuk membangun klasifikasi berikutnya, dengan kata lain hasil pembelajaran iterasi sebelumnya dapat ditransfer ke pembelajaran iterasi berikutnya. Kemudian, metode independen hasil pembelajaran dibangun secara independen dan *output* dari

algoritma pembelajaran digunakan melalui metode kombinasi seperti *majority voting* atau metode *meta-learning* (Breiman, 1996).

Majority voting merupakan salah satu pendekatan *voting based method*. *Voting based method* merupakan metode berbasis pemungutan suara yang beroperasi hanya pada label (Polikar, 2009). Dalam pengimplementasian *majority voting*, untuk melakukan klasifikasi *instance* yang tidak berlabel yaitu dengan melakukan pemungutan suara dan memperoleh jumlah suara terbanyak (Roakch, 2019). Salah satu algoritma yang menerapkan konsep *majority voting* adalah *voting classifier algorithm* (Mahabub, et al., 2019).

2.9 Voting Classifier

Voting Classifier merupakan salah satu algoritma ensemble learning. *Voting classifier* adalah *metaclassifier* yang melakukan prediksi dengan menggabungkan hasil prediksi dari beberapa *base classifier* berdasarkan strategi pemungutan suara yang telah ditentukan sebelumnya. Voting Classifier memanfaatkan kelebihan dari masing-masing *base classifier* di mana setiap *classifier* melakukan yang terbaik sambil meratakan dampak dari kelemahan masing-masing *base classifier* di bagian lain dari *dataset*. Dalam Voting Classifier setiap *base classifier* melatih dan disetel pada *dataset* yang sama secara paralel. Dalam membuat prediksi akhir, Voting Classifier mengkombinasikan hasil prediksi *base classifier* dengan menggunakan *soft voting* atau *hard voting* (Misra, et al., 2020). *Soft voting* memprediksi target berdasarkan argmax dari jumlah prediksi probabilitas kelas. *Hard voting* memprediksi target berdasarkan suara mayoritas (Mahabub, et al., 2019).

2.10 Evaluasi Performa

Evaluasi performa dari *ensemble learning* dengan menggunakan Logistic Regression, Support Vector Machine, dan Multinomial Naïve Bayes dengan TF-IDF *feature extraction* yang dilakukan di penelitian ini ditampilkan dalam hasil pengukuran dengan menghitung akurasi (*recognition rate*), *sensitivity* (*recall*), *precision* dan F1-score. Nilai *accuracy* digunakan untuk mengukur ketepatan prediksi yang dihasilkan. 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. F1-score merupakan rata-rata berbobot dari nilai precision dan recall. Berikut ini merupakan perhitungan untuk evaluasi performa (Han, et al., 2012).

$$Accuracy = \frac{TP+TN}{P+N} \tag{2.12}$$

$$Recall = \frac{TP}{P} \tag{2.13}$$

$$Precision = \frac{TP}{(P\prime)} \tag{2.14}$$

$$F1 - score = \frac{(2 \times Precision \times Recall)}{(Precision+Recall)} \tag{2.15}$$

Tabel 2. 1 Tabel Confusion Matrix

Actual class \ Predicted class	**True**	**False**	**Total**
True	TP	FN	P
False	FP	TN	N
Total	P’	N’	P + N

P merupakan jumlah total dari data dengan kelas sebenarnya bernilai *true*, N merupakan jumlah total dari data dengan kelas sebenarnya bernilai *false*, TP merupakan True Positive, TN merupakan True Negative, FP merupakan False Positive, FN merupakan False Negative yang didapat dari Confusion Matrix.