BAB II

LANDASAN TEORI

2.1 Text Classification dan Text Pre-processing

Text classification atau klasifikasi teks sebuah kegiatan yang memberikan label dengan kategori tertentu yang telah ditetapkan pada teks dengan bahasa alami (Sebastian, 2002). Saat ini klasifikasi teks dapat diterapkan dalam berbagai konteks, mulai dari pengindeksan dokumen berdasarkan kosa kata yang terkontrol, pemfilteran dokumen, pembuatan *metadata* otomatis, dan pada berbagai aplikasi lainnya yang membutuhkan organisasi dokumen (Sebastian, 2002). Ada beberapa strategi umum untuk proses klasifikasi teks secara otomatis, yaitu *preprocessing, feature extraction/selection, modeling* menggunakan teknik pembelajaran mesin yang sesuai, serta *training* dan *testing* pada *classifier* (Dalal & Zaveri, 2011).

Pada tahapan *preprocessing,* terjadi pengurangan *input* pada dokumen teks secara signifikan (Dalal & Zaveri, 2011). Dalam proses *pre-processing* yaitu sebagai berikut (Huda & Z.Rugmina, 2011; Bhumika & A.Nayyar, 2013; Indriani, 2014; Chandrasekar & Qian, 2016; R.P & Juliet, 2017) :

1. Mengubah teks ke bentuk huruf kecil, *Case Folding* adalah proses penyamaan *case* dalam sebuah dokumen. Hal ini dilakukan untuk mempermudah pencarian. Tidak semua dokumen teks konsisten dalam penggunaan huruf kapital. Oleh karena itu peran *case folding* dibutuhkan dalam mengkonversi keseluruhan teks dalam dokumen menjadi suatu bentuk standar (dalam hal ini huruf kecil atau lowercase) (Sutami, 2015).

2. Mengubah angka menjadi kata atau menghapus angka yang tidak dibutuhkan (*conver number)*
3. Mengganti kata yang berhubungan dengan tautan dengan suatu kata pengganti
4. Menghapus tanda baca dan spasi yang berlebihan (*remove punctuation)*
5. *Tokenization* adalah proses pemotongan sebuah dokumen menjadi bagian-bagian, yang disebut dengan token. Pada saat bersamaan *Tokenization* juga berfungsi untuk membuang beberapa karakter tertentu yang dianggap sebagai tanda baca (Sutami, 2015).
6. *Netural language specific stop-word elimination* (Kim, et al., 2006). Menurut (Dalal & Zaveri, 2011) *Stop word* merupakan sebuah kata-kata yang sering muncul dalam teks contohnya '*a*',*'the','an',''of',* dalam bahasa inggris. *Stopword Removal* adalah proses penghilangan kata-kata yang tidak berkontribusi banyak pada isi dokumen (Baeza-Yates & Ribeiro-Neto, 1999). Kata-kata yang termasuk ke dalam *stopword* dihilangkan karena memberikan pengaruh yang tidak baik dalam proses *text mining* seperti kata *"and","i","you*" dan lain-lain (Sutami, 2015).
7. *Stemming* adalah proses merubah suatu kata menjadi bentuk dasar (Dalal & Zaveri, 2011). *Stemming* adalah suatu proses pengembalian suatu kata berimbuhan ke dalam bentuk dasarnya (*root)*. *Stemming* adalah alat pemrosesan teks dasar yang sering digunakan untuk meningkatkan kinerja pada *text retrieval* dan *text classification.* Namun sama pada halnya *stopword,* kinerja *stemming* juga bervariasi dan sering bergantung pada bahasa yang digunakan (Sutami, 2015).

2.2 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 (Setiawati, et al., 2016) .

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* (Witten, et al., 2011).

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 (Andriani, 2013).

Berikut ini adalah rumus untuk perhitungan *Decision tree* (Andriani, 2013):

a. *Entropi*

$$Entropi\ (S)\ = \sum_{j=1}^{k} -\ p_j\ log_2\ p_j \qquad (2.1)$$

dimana keterangan penjelasan rumus diatas sebagai berikut :

S = Himpunan (dataset) kasus.

K = Banyaknnya partisi ***S.***

p_j= Probabilitas yang di dapat dari Sum(Ya) dibagi Total Kasus.

b. ***Gain***

$$Gain\ (A) = Entropi\ (S) - \sum_{j=1}^{k} \frac{|S_i|}{|S|} \times Entropi\ (S_i) \quad (2.2)$$

dimana keterangan penjelasan rumus 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.

$Entropi\ (S)$ = Entropi untuk sample-sample yang memiliki nilai i.

2.3 Random Forest Classifier

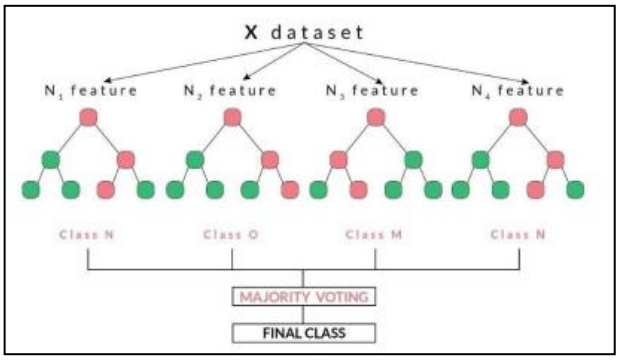
Random Forest adalah pengklasifikasi yang terdiri dari kumpulan pengklasifikasian pohon terstruktur di mana masing-masing pohon melemparkan unit suara untuk kelas paling populer dari input x. (Leo, 2001), Random Forest merupakan sebuah metode ensemble. Metode ensambel merupakan cara untuk meningkatkan akurasi metode klasifikasi dengan cara mengkombinasikan metode klasifikasi. (Han, 2012).

Random forest adalah metode klasifikasi yang supervised. Sesuai dengan namanya metode ini menciptakan sebuah hutan (forest) dengan sejumlah pohon (tree). Secara umum, semakin banyak pohon pada sebuah hutan maka semakin kuat juga hutuan tersbut terlihat, begitu juga semakin banyak pohon maka semakin besar akurasi yang didapatkan. (Haristu, 2019). Dalam perhitungan

membangung pohon (tree) Random Forest menggunakan informasi gain dan gini index untuk penghitungannya. (Han, 2012). Pohon (tree) yang dibangun menggunakan data set dengan atribut yang diambil secara acak dari data training. Dengan kata lain setiap pohon (tree) akan bergantung pada nilai dari sampel vektor yang independen dengan distribusi yang sama pada setiap pohon (tree). Selama proses klasifikasi setiap pohon (tree) akan memberikan voting kelas yang paling populer (Han, 2012).

Metode Random Forest merupakan percabangan dari metode decision tree (Islami, 2019). Decision tree atau pohon pengambil keputusan adalah diagram alir yang memiliki bentuk seperti pohon yang memiliki sebuah root node yang digunakan untuk mengumpulkan data, sebuah inner node yang berada pada root node yang berisikan tentang pertanyaan tentang data dan sebuah leaf node yang digunakan untuk memecahkan masalah serta membuat keputusan (Yanuar, 2018). Random Forest memiliki beberapa decision tree, kemudian algoritma Random Forest mengambil keputusan berdasarkan hasil voting terbanyak dari semua decision tree (Islami, 2019).

Kelebihan dari Random Forest adalah jika terdapat data yang hilang dengan jumlah tertentu, Random Forest masih dapat melakukan klasifikasi dengan akurasi yang stabil karena tidak bergantung dengan satu decision tree saja melainkan membandingkan data voting decision tree lainnya (Islami, 2019).



Gambar 2. 1 Struktur algoritma Random Forest (Islami, 2019)

2.4 Term Frequency Inverse Document Frequency

Data yang telah melalui tahapan *preprocessing* harus berbentuk *numeric* atau angka karena komputer hanya bisa membaca angka. Untuk merubah data tersebut dapat digunakan metode pembobotan TF-IDF. Metode *Term Frequency Inverse Document Frequency* (TF-IDF) merupakan metode yang digunakan menentukan seberapa jauh keterhubungan kata (term) terhadap dokumen dengan memberikan bobot setiap kata. Metode TF-IDF ini menggabungkan dua konsep yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen dan inverse frekuensi dokumen yang mengandung kata tersebut (Fitri, 2013).

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 (Robertson, 2004).

2.4.1 Rumus TF-IDF

Berikut ini adalah rumus TF-IDF untuk pembobotan setiap term dalam sebuah dokumen (Amelia Rahman, 2017)

$$W_{dt} = tf_{dt} \times idf_t = tf_{dt} \times log(\frac{N}{df_t}) \quad (2.3)$$

dimana keterangan penjelasan rumus diatas sebagai berikut :

W_{dt} = Bobot *term* ke-t terhadap dokumen d

tf_d = Jumlah kemunculan *term* t dalam dokumen –d

N = Jumlah dokumen secara keseluruhan

df_t = Jumlah dokumen yang mengandung *term* t

2.5 Pengukuran Kinerja Model *(Performance Measure)*

Dalam pengukuran kinerja model terdapat berbagai cara yaitu *Confusion matrix, Precision, Recall,* dan *F1-Score.*

Untuk pengukuran kinerja model dalam penelitian ini menggunakan pengukuran *F1-Score* karen dalam pengukuran ini dapat menggabungkan *Precision* dan *Recall* dari data yang didapat.

2.5.1 Tabel *Confusion Matrix*

Tabel *Confusion Matrix* dapat membantu dalam memperoleh nilai dari perhitungan tersebut karena tabel ini berisi data prediksi yang positif dan negatif yang dihasilkan oleh sistem, dan data aktual yang positif dan negatif di dunia nyata (Goutte & Gaussier, 2006).

a. Tabel *Confusion Matrix*

	Positif (Aktual)	**Negatif (Aktual)**
Positif (Sistem)	**TP**	**FP**
Negatif (Sistem)	**FN**	**TN**

Penjelaasan *confusion Matrix* (Rasyid, et al., 2019) :

- **TP (True Positive)** adalah perhitungan dari kelas positif sistem dan aktual yang positif.
- **TN (True Negative)** adalah perhitungan dari kelas negatif sistem dan aktual yang negatif.
- **FP (False Positive)** adalah perhitungan dari kelas positif sistem dan aktual yang negatif.
- **FN (False Negative)** adalah perhitungan dari kelas negatif sistem dan aktual yang positif.

2.5.2 *Precision*

Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem (Rasyid, et al., 2019).

a. Rumus *Precision*

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (2.4)$$

2.5.3 *Recall*

Recall merupakan sebuah pengukuran keberhasilan sistem dalam menemukan kembalinya sebuah informasi (Rasyid, et al., 2019)

a. *Rumus Recall*

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (2.5)$$

2.5.4 *F1-Score*

F1-score atau *F-measure* dapat didefinisika sebagai rata rata harmonik *precision* dan *recall.* Untuk pengklasifikasian yang baik dalam pengukuran performansi adalah pengukuran performansi yang memiliki hasil *F1-score* yang tinggi hasih performansi yang tinggi menunjukan bahwa pengklasifikasian berkinerja dengan baik terkait dengan *precision* dan *recal* (Meng, et al., 2011)

a. Rumus *F1-Score*

F1-Score merupakan pengukuran performansi yang menggabungkan perhitungan *precision* dan *recall* (Rasyid, et al., 2019)

$$F_{1=}\frac{2 \times Precision \times Recall}{Precision \times Recall} \times 100\% \quad (2.6)$$