BAB II
TINJAUAN PUSTAKA

2.1 *Abstrak*

Menurut Kamus Besar Bahasa Indonesia (KBBI), pengertian abstrak adalah sesuatu yang tidak berwujud; tidak berbentuk; mujarad; niskala, abstrak juga dapat diartikan sebagai ikhtisar (karangan, laporan, dan sebagainya, ringkasan, inti.

Secara umum, abstrak dapat diartikan sebagai ringkasan dari sebuah karya ilmiah. Abstrak dapat didefinisikan sebagai rangkuman informasi yang terdapat dalam sebuah dokumen (Houghton, 1975). Menurut *American National Standards Institute* (1977), abstrak yang dipersiapkan dengan baik akan memungkinkan pembaca untuk mengidentifikasi materi inti dari sebuah dokumen secara cepat dan akurat, sehingga pembaca dapat mengetahui apakah dokumen tersebut terkait dengan kebutuhan pembaca.

Di dalam abstrak, terutama abstrak karya ilmiah terdapat beberapa hal yang harus dipaparkan, yaitu tujuan utama dan ruang lingkup penelitian, bahan dan metode yang digunakan, memberikan ringkasan hasil, dan simpulan untuk hal-hal yang mendasar (Day, 1993). Pada abstrak laporan penelitian juga terdapat lima hal penting yaitu latar belakang, tujuan, metode, hasil, dan simpulan (Weisberg & Bunker, 1990).

2.2 *Text Mining*

Text Mining yaitu sebuah analisis yang mengumpulkan *keywords* atau terms. *Text mining* melibatkan pembentukan *text* (kata) yang lebih terstruktur dan penggalian informasi yang relevan dan akurat dari teks (Miller, 2005;104). *Text Mining* merupakan tahapan untuk menentukan seberapa jauh keterhubungan dengan term atau kata. Kata tersebut biasanya berbentuk dokumen yang akan diproses, tetapi dokumen-dokumen tersebut belum terstruktur. Text mining dipergunakan untuk ekstraksi informasi yang mengeksplorasi pola yang sangat

menarik (Sya'bani, 2019). Pada *fase pre-processing* dalam *text mining* di bagi menjadi beberapa tahapan yaitu:

- Tahap pertama *Tokennizing* yaitu proses pemotongan string input.
- Tahap kedua *Filtering* proses penyaringan kata
- Tahap ketiga *CaseFolding* merupakan proses untuk merubah semua huruf besar dalam dokumen menjadi huruf kecil, *Tagging* proses mencari bentuk asal dari kata lampau, dan *Analyzing* merupakan proses untuk menentukan hubungan antara kata-kata dengan dokumen yang sudah ada.

Text mining dengan pencarian otomatis sangat berkaitan karena tujuan *text mining* dan pencarian otomatis yaitu untuk mendapatkan atau menghasilkan informasi yang berguna dari beberapa dokumen

2.3 *Pre-processing Text*

Pre-processing text merupakan suatu proses pengubahan data berbentuk teks yang belum terstruktur menjadi data yang terstruktur sesuai dengan kebutuhan. Menurut Triawati dkk. (2009), tahapan pada *pre-processing* antara lain sebagai berikut:

- *Case folding,* merupakan proses mengonversi keseluruhan tulisan dalam data menjadi huruf kecil.
- *Tokenizing*, merupakan proses pemotongan tulisan berupa huruf untuk dipisahkan menjadi kata per kata.
- *Filtering*, merupakan proses menghapus kata-kata yang kurang penting dari hasil pemotongan tulisan. Kata-kata yang dianggap tidak penting, antara lain seperti kata "yang", "dan", "di", "dengan" dan lain sebagainya.
- *Stemming*, merupakan proses mencari kata dasar dari tiap kata pembentuknya. Proses ini digunakan untuk menyeragamkan dan mengefisiensikan penggunaan kata yang memiliki kata dasar sama tetapi memiliki perbedaan pada penulisan seperti pada kalimat berimbuhan.

Berikut contoh proses *pre-processing text* yang dilakukan pada sebuah data komentar seperti ditunjukkan pada Tabel 2.1.

Tabel 2.1 Contoh *Pre-processing Text*

Data	Membeli barang berkualitas bagus dengan biaya pengiriman murah.
Case folding	membeli barang berkualitas bagus dengan biaya pengiriman murah.
Tokenizing	‘membeli’ ‘barang’ ‘berkualitas’ ‘bagus’ ‘dengan’ ‘biaya’ ‘pengiriman’ ‘murah’
Filtering	‘membeli’ ‘barang’ ‘berkualitas’ ‘bagus’ ‘biaya’ ‘pengiriman’ ‘murah’
Stemming	‘beli’ ‘barang’ ‘kualitas’ ‘bagus’ ‘biaya’ ‘kirim’ ‘murah’

Pada Tabel 2.1, tanda petik hanya merepresentasikan pemisahan setiap kata untuk contoh, penerapan sesungguhnya karakter selain huruf akan dihilangkan.

2.4 Pembobotan TF-IDF

Algoritma TF-IDF merupakan suatu metode pembobotan dengan menggunakan penghitungan *term frequency* dan *inverse document frequency*. Metode ini dilakukan dengan mencari representasi nilai dari tiap-tiap kata atau term pada sekumpulan data yang akan dibentuk menjadi sebuah vektor. Penggunaan TF-IDF dilakukan dengan cara pemberian bobot hubungan suatu kata atau fitur (*t*) sebanyak *m* terhadap data (*d*) sebanyak *n*, serta *w* merupakan hasil pembobotan TF-IDF seperti yang ditunjukkan pada Tabel 2.2. (Mulyana dkk, 2012).

Tabel 2.2 *Term-document Matrix*

	t_1	t_2	...	t_m
d_1	w_{11}	w_{21}	...	w_{m1}
d_2	w_{12}	w_{22}	...	w_{m2}
...	...	...	...	...
d_n	w_{1n}	w_{2n}	...	w_{mn}

2.4.1 *Term Frequency*

Term frequency (TF) merupakan frekuensi suatu term atau kata yang terdapat pada setiap data berbentuk dokumen. Nilai TF diperoleh berdasarkan jumlah kemunculan term tersebut pada setiap dokumen. Contohnya, apabila suatu term muncul sebanyak dua kali dalam dokumen, maka nilai TF term tersebut bernilai 2.

2.4.2 *Inverse Document Frequency*

Inverse document frequency (IDF) menunjukan hubungan ketersedian suatu term dalam seluruh dokumen (Harvlant dan Kreinovich, 2014). Berbeda dengan TF yang jika semakin sering frekuensi kata muncul maka nilai semakin besar. Dalam IDF, semakin jika sedikit frekuensi kata muncul dalam dokumen, maka makin besar nilainya.

. Rumus yang digunakan dalam *Inverse document frequency* ditunjukkan pada persamaan 2.1 (Havrlant dan Kreinovich, 2014).

$$idf(t,d) = log\frac{N}{df(t)} \tag{2.1}$$

Keterangan:

$idf(t,d)$ = IDF pada term *t* di dalam data *d*.

N = jumlah data.

$df(t)$ = jumlah data yang mengandung term *t*.

2.4.3 *Term Weighting* TF-IDF

Term weighting TF-IDF merupakan penggabungan dari rumus *term frequency* dengan *inverse document frequency* dengan mengalikan kedua rumus tersebut menjadi sebuah nilai pembobotan. Rumus TF-IDF ditunjukkan pada persamaan 2.2 (Havrlant dan Kreinovich, 2014).

$$w(t,d) = tf(t,d) \times idf(t,d) \tag{2.2}$$

Keterangan:

$w(t,d)$ = hasil pembobotan TF-IDF pada term *t* di dalam data *d*.

$tf(t,d)$ = nilai kemunculan term *t* di dalam data *d*.

2.5 *Naïve Bayes Classifier*

Naive Bayes Classifier merupakan salah satu metode yang populer untuk keperluan data mining karena penggunaannya yang mudah (Hall, 2006) dan dalam pemroresan memiliki waktu yang cepat, mudah diimplementasikan dengan strukturnya yang cukup sederhana dan untuk tingkat efektifitasnya memiliki efektifitas yang tinggi (Taheri & Mammadov, 2013).

Secara umum proses dari klasifikasi *Naive Bayes* dapat dilihat pada Persamaan 2.3.

$$P(c_j|w_i) = \frac{P(c_j)P(w_i|c_j)}{P(w_i)} \tag{2.3}$$

Keterangan:

$P(c_j|w_i)$: Peluang kategori *j* ketika terdapat kemunculan kata *i*.

$P(w_i|c_j)$: Peluang sebuah kata *i* masuk ke dalam kategori *j*.

$P(c_j)$: Peluang kemunculan sebuah kategori *j*.

$P(w_i)$: Peluang kemunculan sebuah kata.

Pada proses perhitungan klasifikasi peluang kemunculan kata sebenarnya dapat dihilangkan, hal ini dikarenakan peluang tersebut tidak berpengaruh pada

perbandingan hasil klasifikasi dari setiap kategori. Sehingga proses pada klasifikasi dapat disederhanakan dengan Persamaan 2.4.

$$P(c_j|w_i) = P(c_j)P(w_i|c_j) \quad (2.4)$$

Dalam proses klasifikasi terdapat prior yang digunakan untuk menghitung peluang kemunculan kategori pada semua dokumen, perhitungan prior dapat dilihat pada Persamaan 2.5.

$$P(c_j) = \frac{N_c}{N} \quad (2.5)$$

Keterangan:

N_c : Banyak dokumen berkategori c pada dokumen latih.

N : Jumlah keseluruhan dokumen latih yang digunakan.

2.6 *Multinomial Naïve Bayes*

Multinomial Naive Bayes adalah probabilitas bersyarat yang dilakukan tanpa memperhitungkan urutan kata dan informasi yang ada dalam kalimat atau dokumen secara umum. Algoritma ini juga memperhitungkan jumlah kata yang muncul dalam dokumen (Destuardi dan Surya, 2009).

Misal terdapat dokumen d dan himpunan kelas c. Untuk memperhitungkan kelas dari dokumen d, maka dapat dihitung dengan rumus :

$$P(c|term\ dokumen\ d) = P(c) \times P(t_1|c) \times P(t_2|c) \times P(t_3|c) \times \ldots \times P(t_n|c) \quad (2.6)$$

Keterangan :

$P(c)$: Probabilitas *prior* dari kategori c

t_n : Kata dokumen d ke-n

$P(c|term\ dokumen\ d)$: Probabilitas suatu dokumen termasuk kategori c

$P(t_n|c)$: Probabilitas kata ke-n dengan diketahui kategori c

Sementara rumus Multinomial yang digunakan dengan pembobotan kata TF-IDF adalah sebagai berikut :

$$P(wk|vj) = \frac{Wct + 1}{(\sum W' \in VW'ct) + B'} \quad (2.7)$$

Keterangan :

$P(wk|vj)$: Probabilitas kemunculan kata wk pada suatu dokumen jika diketahui dokumen tersebut berkategori tertentu.

Wct : Nilai pembobotan $tfidf$ atau W dari term t di kategori c.

$(\sum W' \in VW')$: Jumlah total W dari keseluruhan term yang berada di kategori c.

B' : Jumlah W kata unik (nilai idf tidak dikali dengan tf) pada seluruh dokumen.

2.7 *K-Nearest Neighbor*

Algoritma *k-nearest neighbor* (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN bekerja dengan mencari sejumlah k-objek data atau pola (dari semua pola latih yang ada) yang paling dekat dengan pola masukan, kemudian memilih kelas dengan jumlah pola terbanyak di antara k pola tersebut (Suyanto, 2018). Dekat atau jauhnya lokasi (jarak) bisa dihitung melalui salah satu dari besaran jarak yang telah ditentukan yakni jarak *Euclidean*, jarak *Minkowski*. Namun dalam penerapannya seringkali digunakan jarak *Euclidean* karena memiliki tingkat akurasi dan juga *productivity* yang tinggi. Jarak *Euclidean* adalah besarnya jarak suatu garis lurus yang menghubungkan antar objek. Rumus jarak *Euclidean* adalah sebagai berikut:

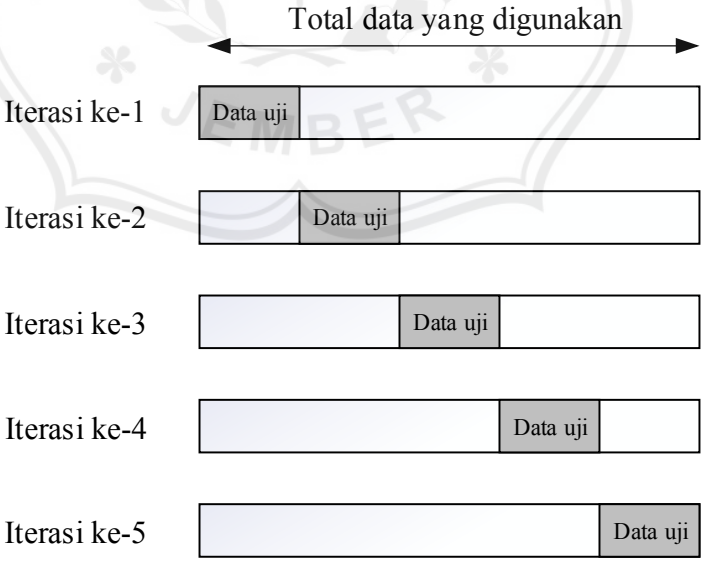
$$d(x_i, x_j) = \sqrt{\sum_{n=1}^{p} (x_{ip} - x_{jp})^2} \quad (2.8)$$

Dari rumus di atas dijelaskan bahwa, $d(x_i, x_j)$ merupakan jarak *euclidean* dari data test dengan data training sedangkan x_{ip} dan x_{jp} merupakan data testing ke i

dan data training ke j. Optimalnya nilai k pada algoritma ini tergantung pada sebuah data. Secara umum, nilai k yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi semakin kabur.

2.8 *K-Fold Cross-Validation*

Metode *K-Fold Cross-Validation* membagi himpunan data secara acak menjadi *K* subhimpunan (biasanya disebut dengan *fold*) yang saling bebas: $f_1, f_2, \dots, f_K$, sehingga masing-masing *fold* berisi $\frac{1}{k}$ bagian data. Selanjutnya dapat dibangun *k* himpunan data: $D_1, D_2, \dots, D_k$, yaitu masing-masing berisi $(K-1)$ *fold* untuk data latih, 1 *fold* untuk data uji. Dimisalkan, dengan *K*=5, maka dapat dibangun himpunan data sebanyak lima himpunan. Himpunan D_1 berisi empat *fold*: f_2, f_3, f_4, f_5, untuk data latih dan satu *fold* f_1 untuk data uji. Himpunan D_2 berisi *fold*: f_1, f_3, f_4, f_5, untuk data latih dan satu *fold* f_2 untuk data uji. Demikian seterusnya untuk himpunan data D_3, D_4, D_5 sehingga setiap *fold* pernah menjadi data uji sebanyak satu kali. Contoh penggunaan *K-Fold Cross Validation* dengan $K = 5$ ditunjukkan pada Gambar 2.1.



Gambar 2.1 Contoh *K-Fold Cross Validation*

Keuntungan dari metode ini adalah keseluruhan *dataset* digunakan sebagai *data training* dan *data testing* untuk meningkatkan akurasi pemodelan klasifikasi. Nilai *K* yang direkomendasikan tergantung besarnya jumlah data set.

2.9 *Python*

Python merupakan bahasa pemrograman yang memiliki banyak fungsi, interaktif yang sering digunakan dalam pengembangan aplikasi. Bahasa pemrograman *Python* dikembangkan oleh Guido van Rossum pada tahun 1990 dan diperkenalkan pertama kali pada tahun 1991. Bahasa pemrograman ini dirancang untuk memberikan kemudahan kepada *programmer* dalam segi efisiensi waktu ataupun kemudahan dalam pengembangan program. Saat ini *Python* telah mencapai versi *Python* 3.6 yang dirilis pada 23 Desember 2016. Beberapa fitur yang dimiliki bahasa pemrograman P*ython* adalah sebagai berikut:

1. Memiliki kepustakaan yang luas.
2. Memiliki tata bahasa yang mudah dipelajari.
3. Mudah dikembangkan dengan menciptakan modul-modul baru dengan bahasa pemrograman *Python* ataupun C / C++.

2.10 *Natural Language Toolkit* (NLTK)

Natural Language Toolkit adalah sebuah *library python* yang berfungsi untuk permodelan teks. NLTK memproses sebuah teks sebelum teks digunakan pada *machine learning* atau *deep learning*. NLTK menyediakan berbagai fungsi dan *wrapper*, serta *corpora* standar baik itu mentah atau pun *pre-processed* yang digunakan dalam materi pengajaran *Natural Language Processing* (NLP).

2.11 *Scikit-Learn*

Scikit-learn atau *sklearn* adalah modul untuk bahasa pemograman python atau dapat di sebut juga sebagai *machine learning library. Sklearn* menyediakan banyak skali fungsi untuk membantu pemrosesan algoritma, *datasets, utilities,* dan lain sebagainya. Ada banyak fungsi atau kegunaan yang dapat dilakukan *sklearn* seperti *Classification, Regression, Clustering, Dimensionality Reduction, Model*

Selection, dan *Preprocessing Data.* Adapun beberapa fungsi yang harus diinputkan sebelum menggunakan *sklearn* pada *pyhton* sebagai berikut :

- *NumPy*: *library python* yang fokus pada *scientific computing.*
- *SciPy*: *library* dasar *python* yang menangani *scientific computing*
- *Matplotlib*: *library python* yang fokus pada visualisasi data
- *IPython*: *shell* interaktif yang dibangun dari *python*
- *Sympy*: simbol-simbol matematika pada *python*
- *Pandas*: *library python* yang fokus untuk proses analisis data

2.12 *Library* Sastrawi

Library Sastrawi merupakan sebuah *library stemmer* berbasis algortima Nazief dan Adriani yang mengalami beberapa peningkatan-peningkatan. Fungsi dari *library* Sastrawi yaitu untuk mengubah semua kata berimbuhan dalam Bahasa Indonesia menjadi bentuk dasarnya. Karna *library* Sastrawi menerapkan algoritma Nazief dan Adriani tentunya alur yang digunakan *library* Sastrawi sama dengan alur algoritma Nazief dan Adriani. Berikut adalah alur algortima Nazief dan Adriani :

1. Memeriksa kata, apakah kata merupakan akar kata *(root)* yang terdapat dalam daftar akar kata *(root)*. Jika kata merupakan akar kata, maka proses berhenti pada tahap ini.
2. Menghilangkan *inflection Suffixes* (“-lah”,”- kah”, “-ku”, “-mu”, atau “-nya”). Jika kata berupa particles (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus Possesive Pronouns (“-ku”, “-mu”, atau “-nya”).
3. Menghilangkan *derivational suffix* (imbuhan turunan). Menghilangkan imbuhan -i, -kan, -an.
4. Menghilangkan *derivational prefix* (awalan turunan). Menghilangkan awalan be-, di-, ke-, me-, pe-, se- dan te-.
5. Bila dari langkah 4 di atas belum ketemu juga. Maka lakukan analisis apakah kata tersebut terdapat dalam tabel ambiguitas kolom terakhir atau tidak.

6. Bila semua proses gagal, maka algoritma akan mengembalikan kata aslinya.

2.13 *Jupyter Notebook*

Jupyter Notebook adalah sebuah aplikasi open source yang fungsinya untuk membuat dan berbagi dokumen yang berisi *live code,* persamaan, visualisasi dan teks penjelasan.

Project Jupyter adalah proyek nirlaba, sumber terbuka, yang lahir dari Proyek *IPython* pada tahun 2014 karena berevolusi untuk mendukung ilmu data interaktif dan komputasi ilmiah di semua bahasa pemrograman. *Jupyter* akan selalu menjadi perangkat lunak *open source,* gratis digunakan untuk semua dan dirilis di bawah persyaratan liberal dari lisensi BSD yang dimodifikasi.

Jupyter dikembangkan di *GitHub,* melalui konsensus komunitas *Jupyter*. Semua interaksi dan komunikasi daring dan langsung yang terkait langsung dengan proyek dicakup oleh Pedoman Perilaku Jupyter.

2.14 *Confusion Matrix*

Confusion matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya (Prasetyo, 2012)

Pada pengukuran kinerja menggunakan confusion matrix, terdapat 4 istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positve* (FP), *False Negative* (FN). Berikut adalah penjelasan mengenai istilah representasi hasil proses klasifikasi di tujukan pada Tabel 2.3.

Tabel 2.3. Representasi hasil proses klasifikasi *Confusion Matrix*

True Positive (TP)	data positif yang terdeteksi benar oleh model klasifikasi.
True Negative (TN)	data negatif yang terdeteksi dengan benar oleh model klasifikasi.
False Positve (FP)	data negatif namun terdeteksi sebagai data positif oleh model klasifikasi.
False Negative (FN)	data positif terdeteksi sebagai data negatif oleh model klasifikasi.