BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada sub bab ini menjelaskan tentang konsep yang terkait dengan penelitian.

2.1.1 Analisis Sentimen

Analisis sentimen atau penggalian opini merupakan sebuah studi komputasi yang didasarkan oleh opini, sentimen, emosi, penilaian, dan sikap terhadap sebuah entitas seperti produk, layanan, organisasi, individu, masalah, kejadian, dan topik yang berkaitan lainnya [16].

Jadi Sentimen analisis dapat dijadikan sebagai acuan dalam mengklasifikasikan opini masyrakat mengenai sebuah peristiwa atau konflik yang terjadi di kalangan masyrakat tentang polemik Undang-undang Cipta kerja *Omnibus law* atau permasalahan lain sebagainya, Opini yang dimaksudkan adalah opini yang tengah beredar luar di kalangan Warganet/internet pada media sosial *Twitter*.

2.1.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) adalah salah satu bidang ilmu komputer yang merupakan cabang dari kecerdasan buatan, dan bahasa (linguistik) yang berkaitan dengan interaksi antara komputer dan bahasa alami manusia, seperti bahasa Indonesia atau bahasa Inggris. Tujuan utama dari studi NLP adalah membuat mesin yang mampu mengerti dan memahami makna bahasa manusia lalu memberikan respon yang sesuai [17].

2.1.3 *Text Mining*

Text Mining dapat didefinisikan sebagai proses penggalian informasi, di mana pengguna menggunakan alat analisis untuk berinteraksi dengan sekumpulan

dokumen, dan alat analisis merupakan bagian integral dari data mining, salah satunya adalah peningkatan dokumen. Tujuan penambangan teks adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Oleh karena itu, sumber data yang digunakan dalam text mining adalah kumpulan teks dengan format tidak terstruktur atau minimal semi terstruktur [18].

Adapun tugas khusus dari *text* mining antara lain yaitu pengkategorisasian teks (*text categorization*) dan pengelompokan teks (*text clustering*). Permasalahan yang dihadapi pada *text mining*. sama dengan permasalahan yang terdapat pada data *mining*, yaitu jumlah data yang besar, dimensi yang tinggi, data dan struktur yang terus berubah, dan data *noise*. Perbedaan di antara keduanya adalah pada data yang digunakan. Pada *data mining*, data yang digunakan adalah struktur data, sedangkan pada *text mining*, data yang digunakan text mining pada umumnya adalah *unstructured* data, atau minimal *semistructured*. Hal ini menyebabkan adanya tantangan tambahan pada text mining yaitu struktur teks yang kompleks dan tidak lengkap, arti yang tidak jelas dan tidak standar, dan bahasa yang berbeda ditambah translasi yang tidak akurat.

2.1.4 *Preprocessing*

Preprocessing adalah mempersiapkan dokumen teks yang tidak terstruktur menjadi data terstruktur yang siap digunakan untuk proses selanjutnya. Tahapan text processing yang dilakukan[19] [20], diantaranya adalah:

2.1.4.1 *Cleansing*

Cleansing, yaitu proses membersihkan *Tweets* dari kata yang tidak diperlukan untuk mengurangi *noise*. Kata yang dihilangkan adalah karakter HTML, kata kunci, ikon emosi, hashtag (#), username (@username), url (http://situs.com), dan email (nama@situs.com).

2.1.4.2 *Tokenizing*

Tokenisasi adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi token atau potongan kata tunggal atau *trimmed word*. Pada saat

bersamaan, tokenisasi juga membuang beberapa karakter tertentu yang dianggap sebagai tanda baca.

2.1.4.3 *Case Folding*

Case-folding adalah proses penyamaan *case* dalam sebuah dokumen.Ini dilakukan untuk mempermudah pencarian.

2.1.4.4 Penghilangan *Stopword*

Stopword didefinisikan sebagai term yang tidak berhubungan (*irrelevant*) dengan subjek utama dari database meskipun kata tersebut sering kali hadir di dalam dokumen. Berikut ini adalah contoh *stopwords* dalam bahasa Indonesia: yang, juga, dari, dia, kami, kamu, aku, saya, ini, itu, atau, dan, tersebut, pada, dengan, adalah, yaitu, ke,tak, tidak, di, pada, jika, maka, ada, pun, lain, saja, hanya, namun, seperti, kemudian, dll.

2.1.4.5 *Stemming*

Kata-kata yang muncul di dalam dokumen sering mempunyai banyak varian morfologik. Karena itu, setiap kata yang bukan stop-words direduksi ke bentuk stemmed word (term) yang cocok. Kata tersebut diatur untuk mendapatkan bentuk akarnya dengan menghilangkan awalan atau akhiran. Dengan cara ini, diperoleh kelompok kata yang mempunyai makna serupa tetapi berbeda wujud sintaksis satu dengan lainnya. Kelompok tersebut dapat direpresentasikan oleh satu kata tertentu. Sebagai contoh, kata menyebutkan, tersebut, disebut dapat dikatakan serupa atau satu kelompok dan dapat diwakili oleh satu kata umum sebut.

2.1.5 Klasifikasi

Klasifikasi Merupakan suatu proses untuk menilai objek data agar dapat dimasukan ke dalam sebuah kelas tertentu dari beberapa kelas yang tersedia, terdapat 2 buah proses utama yang dilakukan dalam melakukan klasifikasi yaitu:

1. Pembuatan Model menjadi Purwarupa agar dapat disimpan dalam bentuk memori.

2. Penggunaan model digunakan untuk melakukan pengenalan/klasifikasi/prediksi pada suat objek data lain agar diketahui di kelas mana objek data tersebut dalam model yang sudah sebelumnya disimpan di memori.

2.1.6 Pengukuran Kinerja Klasifikasi

Sebuah sistem yang melakukan klasifikasi diharapkan dapat melakukan klasifikasi semua set data dengan benar, tetapi tidak dapat dimungkiri bahwa kinerja suatu sistem tidak bisa 100% benar sehingga sebuah sistem klasifikasi juga harus diukur kinerjanya. Umumnya, pengukuran kinerja klasifikasi dilakukan dengan matriks konfusi (*confusion matrix*). Matriks konfusi merupakan tabel pencatat hasil kerja klasifikasi. Tabel 2.1 merupakan contoh matriks konfusi yang melakukan klasifikasi masalah biner (dua kelas), hanya ada dua kelas, yaitu kelas 0 dan 1. Setiap sel dalam matriks menyatakan jumlah *record*/data dari kelas i yang hasil prediksinya masuk ke kelas j. Misalnya, sel f : adalah jumlah data dalam kelas l yang secara benar dipetakan ke kelas l, dan fij adalah data dalam kelas 1 yang dipetakan secara salah ke kelas 0 [21].

Tabel 2. 1 Matriks Konfusi Untuk Klasifikasi Dua Kelas

Fij		kelas hasil prediksi	
		Kelas = 1	Kelas = 0
Kelas Asli (i)	Kelas = 1	f_{11}	f_{10}
	kelas = 0	f_{01}	f_{00}

Berdasarkan isi matriks konfusi, kita dapat mengetahui jumlah data dari masing-masing kelas yang diprediksi secara benar, yaitu (f_{11} + f_{00}), dan data yang diklasifikasikan salah, yaitu (f_{10} + f_{01}). Kuantitas matriks konfusi dapat diringkas menjadi dua nilai, yaitu akurasi, presisi, *recall*, *f1-score* dan *Specificity.*

Dengan mengetahui jumlah data yang diklasifikasikan secara benar, kita dapat mengetahui akurasi hasil prediksi, dan dengan mengetahui jumlah data yang diklasifikasikan secara salah, kita dapat mengetahui laju eror dari prediksi yang dilakukan [22] [23]. Dua kuantitas ini digunakan sebagai metrik kinerja klasifikasi. Untuk menghitung akurasi digunakan formula :

$$\text{Akurasi} = \frac{f_{11}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}}\ x\ 100\%$$

$$\text{Presisi} = \frac{Jumlah\ Data\ kelas\ 1\ yang\ diprediksi\ tidak\ tepat}{Jumlah\ prdiksi\ kelas\ 1} = \frac{f_{11}}{f_{11+f_{10}}} x\ 100\%$$

$$Recall = \frac{Jumlah\ data\ kelas\ 1\ yang\ diprediksi\ tepat}{Jumlah\ data\ kelas\ 1} = \frac{f_{11}}{f_{11+f_{01}}} x\ 100\%$$

$$F1\text{-}score = \frac{2*recall*presisi}{recall+presisi}\ \text{x}\ 100$$

$$Specificity = \frac{True\ Negative}{True\ Negative+False\ Positive}$$

Semua algoritma klasifikasi berusaha membentuk model yang mempunyai akurasi tinggi (laju error yang rendah). Umumnya, model yang dibangun dapat memprediksi dengan benar pada semua data yang menjadi data latihnya, tetapi ketika model berhadapan dengan data uji, barulah kinerja model dari sebuah algoritma klasifikasi ditentukan, adapun pengukuran *performance* pada akurasi, presisi, *recall,* F-*1 Score,* dan *Specificity* dapat dikatakan sebagai acuan performasi algoritma, dimana akurasi digunaka sebagai acuan jika dataset memiliki jumlah data *False Negative* dan *False Positive* yang mendekati *simetric* [25].

Kemudian presisi digunakan sebagai acuan bila terjadi *True Positive* dan tidak terjadi *False Negative,* sedangkan *Recall* adalah sebagai rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif, lalu *F-1 Score* sebagai acuan perbandingan rata-rata presisi dan recall yang di bobotkan, lalu yang terakhir adalah Specificity merupakan kebenaran prediksi negatif untuk menjawab pertanyaan mengenai berapa banyak sentimen negatif yang dinyatakan benar dibandingkan dengan keseluruhan sentimen yang sebenarnya benar.

2.1.7 *Term Frequency – Inverse Document Frequency* (TF-IDF)

Metode TF-IDF merupakan suatu cara untuk memberikan bobot hubungan suatu kata (term) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu dan inverse frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebut tinggi di dalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen [26].

Pada algoritma TF-IDF digunakan rumus untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci dengan rumus yaitu:

$$w_{dt} = TF_{dt} * (IDF_{ft} + 1) \ldots.$$

Dimana :

w_{dt} = Bobot dokumen ke-d terhadap kata ke-t

TF_{dt} = Banyaknya kata yang dicari dari sebuah dokumen

IDF_{ft} = *Inversed Document Frequency* ($log\ (\frac{n}{df})$)

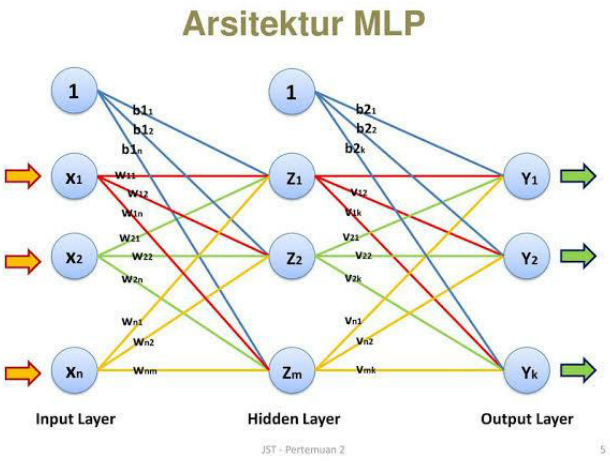
N = Total Dokumen

df = Banyak dokumen yang mengandung kata yang dicari.

2.1.8 *Multilayer Perceptron*

Multilayer Perceptron (MLP) merupakan ANN turunan dari *Perceptron*, berupa ANN umpan balik (feedforward) dengan satu atau lebih layer tersembunyi (*hidden layer*). Biasanya, jaringan terdiri atas satu layer masukan, setidaknya satu layer neuron komputasi di tengah (tersembunyi), dan sebuah *layer neuron* komputasi

keluaran. Sinyal masukan ditambatkan dengan arah maju pada layer-per-layer[27]. Contoh arsitektur MLP adalah sebagai berikut :



Gambar 2. 2 Arsitektur *Multilayer Perceptron*

Pada gambar tersebut ada satu layer tersembunyi dengan empat neuron, dan satu layer keluaran dengan tiga neuron. Sebenarnya ada satu layer lagi dalam MLP, yaitu layer masukan berupa vektor masukan. Namun, dalam layer ini tidak ada komputasi, yang dilakukan hanya meneruskan sinyal/vektor masukan yang diterima ke layer di depannya.

Setiap layer dalam MLP mempunyai fungsi khusus. Layer masukan berfungsi menerima sinyal/ vektor masukan dari luar dan mendistribusikannya ke semua neuron dalam layer tersembunyi. Layer keluaran menerima sinyal keluaran (atau dengan kata lain, stimulus pola) dari layer tersembunyi dan memunculkan sinyal/nilai/kelas keluaran dari keseluruhan jaringan.

Neuron dalam layer tersembunyi mendeteksi fitur-fitur tersembunyi. Bobot dari neuron dalam layer tersembunyi merepresentasikan fitur tersembunyi dalam vektor masukan. Fitur-fitur tersembunyi ini kemudian digunakan oleh layer keluaran dalam penentuan pola/kelas keluaran. Dengan satu layer tersembunyi, kita dapat merepresentasikan sembarang fungsi kontinu dari sinyal masukan, dan dengan dua layer tersembunyi, fungsi diskontinu pun dapat direpresentasikan. Layer tersembunyi "menyembunyikan" keluaran yang diinginkan. Neuron dalam layer

tersembunyi tidak dapat diamati melalui perilaku masukan/keluaran jaringan secara keseluruhan.

Berikut ini adalah algoritma *Multilayer Perceptron*) :

1. Inisialisasi semua bobot dengan bilangan acak kecil.
2. Jika kondisi penghentian belum dipenuhi, lakukan langkah 2-8.
3. Untuk setiap pasang data pelatihan, lakukan langkah 3-8
4. Tiap unit masukan menerima sinyal dan meneruskan ke unit tersembunyi diatasnya.
5. Hitung semua keluaran di unit tersembunyi *Zj* (j = 1, 2,..., p).

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i \, v_{ij}$$

voj = bias pada unit tersembunyi *j* aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, *zj = f* (*z_inj*), dan kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit keluaran). Terapkan fungsi aktivasi dan hasilnya dikirim ke *output unit*.

6. Tiap unit keluaran (*yk, k = 1,..., m*) jumlahkan bobot sinyal masukannya, $y_{in_k} = w_{0k} + \sum_{j=1}^{p} z_j \, w_{jk}$*wok* = bias pada unit keluaran *k* dan aplikasikanfungsi aktivasinya untuk menghitung sinyal keluarannya,

$$y_k = f\left(y_{in_k}\right)$$

7. Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k(k = 1, 2,..., m). δk = (tk − yk)f'(y_netk) = (tk − yk)yk(1 − yk), tk = target δk merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layer dibawahnya. Hitung perubahan bobot wkj dengan laju pemahaman α. $\Delta W_{0k} = \alpha \delta_k$
8. Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk}$$

Faktor δ unit tersembunyi Kalikan dengan nilai fungsi turunan aktivasi untuk menghitung *error*,

$$\delta_j = \delta_{in_j} f\left(Z_{in_j}\right)$$

Hitung nilai perbaikan bobot, (v), 2.$\Delta V_{ij} = \alpha \delta_j x_i$ 16 Hitung nilai perbaikan bias (v), $\Delta V_j = \alpha \delta_j$.

9. Hitung semua perubahan bobot. Perubahan bobot garis yang menuju ke unit keluaran, yaitu:

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk})$$

Perubahan bobot garis yang menuju ke unit tersembunyi, yaitu:

$$V_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$$

2.1.9 Python

Python adalah bahasa pemrograman interpretatif yang dianggap mudah dipelajari serta berfokus pada keterbacaan kode. Dengan kata lain, Python diklaim sebagai bahasa pemrograman yang memiliki kode-kode pemrograman yang sangat jelas, lengkap, dan mudah untuk dipahami. Python secara umum berbentuk pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional, Python dapat digunakan untuk berbagai keperluan pembangunan perangkat lunak dan dapat berjalan diberbagai platform sistem operasi.

Python memiliki beberapa fitur dan kelebihan sebagai berikut:

1. Memiliki koleksi kepustakaan yang banyak, itu artinya, telah tersedia modul-modul "siap pakai" untuk berbagai keperluan.
2. Memiliki struktur bahasa yang jelas, sederhana, dan mudah dipelajari.
3. Berorientasi objek.

   Memiliki sistem pengelolaan memori otomatis (garbage collection) seperti halnya Java Bersifat modular sehingga mudah dikembangkan dengan menciptakan modul – modul baru, baik dibangun dengan bahasa Python maupun C/C++..