BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis Sentimen, atau yang sering disebut juga dengan Opinion Mining adalah ilmu yang menganalisis pendapat seseorang, sentimen, perilaku, penilaian dan emosi terhadap suatu entitas, seperti produk, pelayanan, jasa, organisasi, individu, dan entitas lainnya. Analisis sentimen ini dapat mengelompokkan polaritas dari teks dalam kalimat atau dokumen untuk mengetahui apakah opini pada kalimat atau dokumen tersebut termasuk positif atau negative. Pada analisis sentimen terdapat tiga klasifikasi utama, yaitu :

1. Level dokumen

   Tujuan dari analisis sentimen level dokumen adalah untuk mengklasifikasikan sebuah dokumen opini yang menunjukkan bahwa dokumen itu berisi opini positif, netral, atau negatif dan menganggap bahwa seluruh isi dokumen tersebut membahas satu topik saja.

2. Level kalimat

   Pada level kalimat, analisis sentimen bertujuan untuk mengklasifikasikan sentimen yang ada pada tiap-tiap kalimat. Analisis sentimen pada level kalimat akan menentukan apakah sebuah kalimat tersebut menyatakan ekspresi opini negatif, netral, atau positif.

3. Level aspek

   Pada level ini, analisis sentimen bertujuan untuk mengklasifikasikan sentimen berdasarkan aspek khusus dari sebuah entitas. Pemberi opini dapat memberikan opini yang berbeda untuk aspek-aspek yang berbeda dari entitas yang sama seperti ”Kualitas suara dari telepon ini jelek, tapi baterainya tahan lama”.

2.1.1 Analisis Sentimen Berdasarkan Aspek

Analisis sentimen berdasarkan aspek adalah salah satu area kasus *opinion mining* yang bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Klasifikasi teks opini pada level dokumen atau kalimat seringkali tidak cukup karena level tersebut hanya bisa mengidentifikasi sentimen secara umum. Penelitian yang dilakukan oleh Sasmita, Wicaksono, Louvan, dan Adriani [1] task dalam analisis sentimen berdasarkan aspek terdiri dari 2 task, yaitu Aspect Extraction dan Aspect Sentiment Orientation Classification.

1. Aspect Extraction

   Pada tahap ini mengekstrak aspek yang sudah ditentukan sebelumnya. Misalnya, dalam kalimat "harga makanan disini murah", aspeknya adalah "harga" entitasnya "makanan". Pada kata "makanan" tidak menunjukan aspek umum karena evaluasi bukan tentang makanan secara keseluruhan tetapi hanya tentang "harga".

2. Aspect Sentiment Orientation Classification

   Pada tahap ini menentukan apakah pendapat dari berbagai aspek kedalam sentimen positif, atau negatif. Pada contoh kalimat "pelayanan yang diberikan memuaskan". Sentimen dari aspek "pelayanan" adalah positif.

Tujuan analisis sentimen berdasarkan aspek adalah untuk mengidentifikasi aspek dari suatu entitas, dan sentimen yang diungkapkan oleh penulis komentar tentang aspek tersebut. Misalnya, dari kalimat "suasana di restoran ini sangat nyaman tapi harga makanannya mahal" pada kalimat tersebut kata "suasana" dan "harga" diidentifikasikan sebagai aspek yang kemudian ditentukan sentimennya dengan kata "nyaman" untuk aspek "suasana" yang mengandung sentimen positif sedangkan kata "mahal" untuk aspek "harga" dan mengandung sentimen negatif. Pada tahap mengidentifikasikan aspek ada beberapa pendekatan yaitu, *frequency based, relation based, supervised learning,* dan *topic modelling* dan untuk penentuan sentimen terhadap aspek mempunyai 2 pendekatan yaitu, *supervised learning* dan *lexicon based*.

2.2 Preprocessing

Metode *preprocessing* memainkan peran yang sangat penting dalam teknik dan aplikasi *text mining*. Ini adalah langkah pertama dalam proses pengolahan teks agar data siap untuk diproses. Adapun tahapan dari *preprocessing* pada penelitian ini antara lain, *case folding*, *filtering*, *word normalization*, *tokenization*, dan *stopword removal*, dan *word embedding*.

2.2.1 Case Folding

Case Folding merupakan proses yang dilakukan untuk menyeragamkan huruf pada semua teks di dokumen ke dalam bentuk huruf kapital (*uppercase*) atau huruf kecil (*lowercase)*. Pada penelitian ini, teks akan diubah kedalam bentuk *lowercase*.

2.2.2 Filtering

Filtering bertujuan untuk menghilangkan elemen-elemen yang tidak diperlukan dalam proses analisis sentimen. Salah satu elemen yang tidak diperlukan adalah tanda baca, seperti koma (,) dan titik (.).

2.2.3 Word Normalization

Word Normalization merupakan proses untuk mengubah kata tidak baku menjadi kata baku.

2.2.4 Tokenization

Tokenization bertujuan untuk mengubah struktur teks atau kalimat ke dalam bentuk token atau potongan kata. Proses pemisahan kata dilakukan berdasarkan spasi diantara setiap kata dalam kalimat.

2.2.5 Stopwords Removal

Stopwords pada penelitian ini bertujuan untuk menghilangkan kata-kata yang tidak memberikan makna secara signifikan pada kalimat, umumnya kata-kata tersebut masuk ke dalam kategori *stop-word*. Kata "*stop-word*" biasanya merujuk pada kata-kata yang paling umum seperti kata "dan", "dengan", "di", dan lain-lain.

2.2.6 Word Embedding

Word embedding dilakukan untuk mengubah sebuah kata menjadi kumpulan angka dalam bentuk sebuah vektor agar data dapat dengan mudah diolah oleh algoritma.

Metode yang umum digunakan agar algoritma dapat membaca data berupa teks adalah dengan mengonversi setiap kata menjadi integer dengan memberinya nomor. Angka tersebut diubah menjadi vektor dengan panjang yang sama dengan data dalam kamus. Vektor angka tersebut hanya memiliki nilai 1 atau 0 yang juga dikenal sebagai *one-hot-encoding*. Nilai 1 ditempatkan di indeks yang mewakili kata, nilai lainnya adalah 0. Dalam hal ini, jika kamus data sangat besar, maka akan dikonversi kata demi kata menjadi vektor yang besar, dengan hampir semua elemen memiliki nilai 0. Oleh karena itu, metode *one-hot-encoding* yang satu ini tidak efisien pada memori dan tidak memberikan banyak informasi.

Metode Fasttext dapat mengubah kata menjadi vektor yang berisi angka. Ukuran vektor ini cukup kecil untuk menampung lebih banyak informasi daripada jika *one-hot-encoding* digunakan.

2.2.7 FastText

FastText merupakan pengembangan dari library Word2Vec yang menggunakan CBOW atau Skip-Gram untuk merepresentasi kata menjadi vektor, tetapi secara default FastText menggunakan Skip-Gram. FastText tidak menggunakan seluruh kata untuk pemrosesan, tapi menggunakan n-gram. Contoh implementasi n-gram untuk kata "pintar" menggunakan trigram (n = 3) dalam bentuk "pin", "int", "nta", "tar".

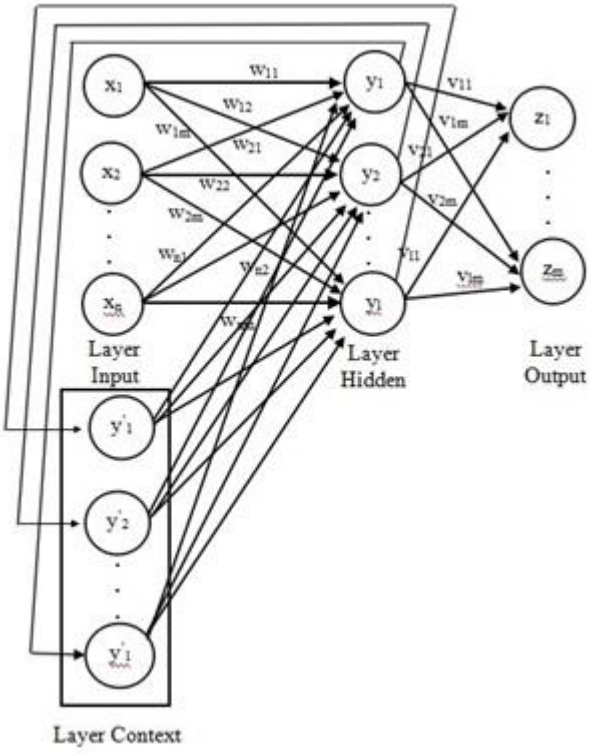
Model FastText memiliki keunggulan yakni kemampuan FastText untuk menangani kata yang tidak pernah kita jumpai sebelumnya (*Out Of Vocabulary word*). Library Word2Vec ataupun teknik *one hot encoding* tradisional akan menghasilkan *error* ketika menerima kata yang tidak pernah ada di kamus.

2.3 Recurrent Neural Network

Recurrent Neural Network (RNN) merupakan jaringan saraf berulang. Dikatakan jaringan saraf berulang karena nilai neuron pada hidden layer sebelumnya akan digunakan kembali sebagai information input. Penggunaan neuron pada hidden layer akan disimpan ke dalam sebuah layer yang dinamakan context layer. Nilai neuron pada context layer akan terus update hingga kondisi RNN terpenuhi [9]. RNN adalah jaringan syaraf tiruan yang menggunakan rekurensi dengan memanfaatkan data masa lalu.

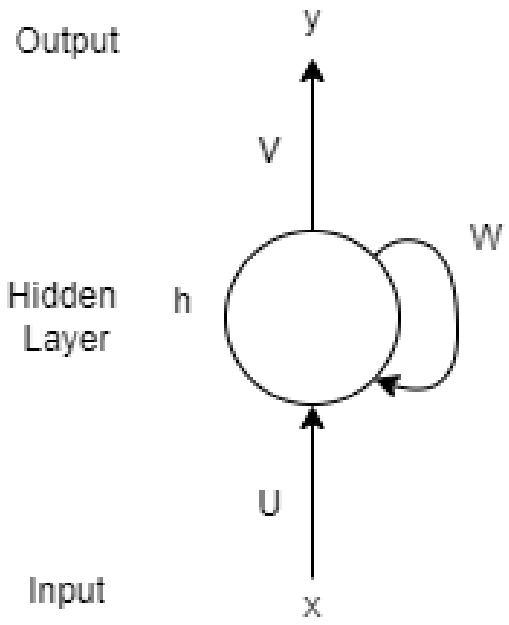
2.3.1 Elman Recurrent Neural Network

Elman Recurrent Neural Network (ERNN) merupakan salah satu arsitektur dari *Recurrent Neural Network* (RNN), *Elman Recurrent Neural Network* (ERNN) juga merupakan salah satu pengembangan dari algoritma neural network Backpropagation [10]. Perbedaan antara kedua algoritma ini adalah bahwa ERNN memiliki umpan balik (*feedback*) di lapisan *hidden*. Dari hasil umpan balik tersebut, dibuatlah lapisan tambahan yang disebut *context layer*. *Context layer* dapat meningkatkan kecepatan iterasi dan pembaruan parameter. Ini memungkinkan perhitungan dilakukan berdasarkan nilai perhitungan sebelumnya.



Gambar 2.1 Arsitektur Elman Recurrent Neural Network

ERNN memiliki loop tertutup dalam topologi jaringan. Unit-unit dalam kelompok tertentu yang menerima umpan balik dari tahap sebelumnya disebut unit kompleks. Arsitektur jaringan ERNN umumnya merupakan jaringan dua lapis dengan proses umpan balik dari keluaran (*output*) lapisan pertama dan masukan (*input*) lapisan pertama. Koneksi berulang yang ditemukan di jaringan Elman memungkinkan Elman mendeteksi dan menghasilkan pola waktu yang beragam dengan lebih baik. Jaringan Elman memiliki neuron tansig pada lapisan tersembunyi dan neuron purelin pada lapisan keluaran (*output*) [10].



Gambar 2.2 Bobot Dalam Elman Recurrent Neural Network

Beberapa parameter yang mempengaruhi proses pelatihan ERNN, yaitu inisialisasi bobot, jenis input, jumlah neuron tersembunyi, kecepatan belajar, dan faktor impuls. Jika terjadi kesalahan dalam menentukan parameter, prosesnya akan memakan waktu lama [10]. Penelitian ini membutuhkan 3 parameter bobot seperti pada Gambar 2.2, yaitu bobot U dari lapisan masukan (*input layer*) ke lapisan tersembunyi (*hidden layer*), bobot W dari lapisan konteks (*context layer*) ke lapisan tersembunyi (*hidden layer*), dan bobot V dari lapisan tersembunyi (*hidden layer*) ke lapisan keluaran (*output layer*). Selama latihan, ketiga bobot akan diperbarui untuk mendapatkan bobot yang optimal. Untuk melakukan *forward propagation* ERNN menggunakan persamaan (2.1).

$$h_t = tanh(W.h_{(t-1)} + U.x_i) \quad \textbf{(2.1)}$$

Dimana:

h_t : *hidden state* pada timestep ke-t

W : bobot dari neuron *hidden layer* sebelumnya ke neuron *hidden layer* selanjutnya

x_i : nilai neuron *input* ke-i

U : bobot dari neuron *input layer* ke neuron *hidden layer*

t : timestep

Sedangkan nilai output layer waktu ke t didapatkan dengan persamaan (2.2).

$$y_t = softmax(V.h_t) \quad \textbf{(2.2)}$$

Dimana :

y_t : output layer pada waktu ke-t

V : bobot dari neuron hidden ke neuron output

h_t : nilai neuron *hidden state* pada waktu ke-t

2.3.2 Fungsi Aktivasi

Fungsi aktivasi sangat umum digunakan dalam *neural network*. Alasan utama menggunakan fungsi aktivasi adalah agar *neural network* mengenali data yang nonlinear, karena output yang dihasilkan dari *neural network* jarang sekali bersifat linear [11]. Fungsi aktivasi juga digunakan untuk mengaktifkan dan menonaktifkan neuron, sifat-sifat jaringan syaraf tiruan ditentukan oleh bobot serta input dan output dari fungsi aktivasi yang diterapkan. Ada banyak jenis fungsi aktivasi dalam jaringan syaraf tiruan, namun yang digunakan dalam penelitian ini adalah fungsi aktivasi *hyperbolic tangent* dan *softmax*.

2.3.2.1 Tanh (Hyperbolic Tangent)

Fungsi aktivasi *hyperbolic tangent* adalah tipe aktivasi fungsi dalam *deep learning* dan memiliki beberapa varian, fungsi aktivasi *hyperbolic tangent* dikenal juga sebagai fungsi *tanh* yang menghasilkan nilai -1 sampai 1 [11]. Fungsi *tanh* memberikan kinerja pelatihan yang lebih baik untuk *multi layer neural network* dibandingkan fungsi sigmoid. Fungsi *tanh* telah banyak digunakan dalam *recurrent neural network* untuk kasus *natural language processing* dan *speech recognition*. Persamaan dari fungsi tanh dapat dilihat pada persamaan (2.3).

$$tanh(x) = \frac{sin(x)}{cos(x)} = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right) = \left(\frac{e^{2x} - 1}{e^{2x} + 1}\right) \quad \textbf{(2.3)}$$

Turunan aktivasi dari fungsi tanh yang nantinya akan digunakan pada proses BPTT dapat dilihat pada persamaan (2.4).

$$tanh'(tanh) = (1 - y^2) \quad \textbf{(2.4)}$$

Dimana y merupakan output dari lapisan tersembunyi.

2.3.2.2 Softmax

Fungsi aktivasi *softmax* digunakan untuk menghitung probabilitas pada hasil *output* yang terjadi di *output layer* dimana akan diambil nilai probabilitas paling besar sebagai prediksi. *Softmax* digunakan untuk menghitung probabilitas distribusi dari vektor bilangan real. Fungsi *softmax* menghasilkan keluaran (*output*) yang merupakan kisaran nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1 [11]. Persamaan dari fungsi *softmax* dapat dilihat pada persamaan (2.5).

$$softmax(x_i) = \frac{exp(x_i)}{\Sigma_j exp(x_j)} \quad \textbf{(2.5)}$$

Turunan aktivasi dari fungsi *softmax* yang nantinya akan digunakan pada proses BPTT dapat dilihat pada persamaan (2.6).

$$softmax'(softmax) = (y_t - label_t) \otimes h_t \quad \textbf{(2.6)}$$

Dimana :

y_t : prediksi label ERNN ke-t

$label_t$: label sebenarnya ke-t

h_t : nilai neuron *hidden state* pada waktu ke-t

2.3.3 Loss Function

Loss Function bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai sebenarnya. Terdapat beberapa jenis *loss function* diantaranya *Mean Square Error* (MSE) dan *Cross Entropy* (CE). Namun pada prakteknya, CE lebih cepat dalam konvergensi dan mendapatkan hasil lebih baik dalam tingkat klasifikasi. CE biasa digunakan pada *neural network* dan dalam

kasus *multi class classification*. Pada penelitian ini *loss function* yang digunakan adalah *Cross Entropy* (CE). Persamaan CE dapat dilihat pada persamaan (2.7).

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) \quad \textbf{(2.7)}$$

Pada Persamaan (2.7) terdapat beberapa variabel yaitu, t_i adalah label sebenarnya pada timestep ke-i, p_i adalah label prediksi pada timestep ke-i.

2.3.4 Backpropagation Through Time (BPTT)

Backpropagation through time merupakan algoritma yang biasa digunakan untuk memperbaharui bobot pada *Recurrent Neural Network*. Pada *Recurrent Neural Network* nilai *error* dapat di *backpropagate* lebih jauh daripada *neural network* biasa. Prinsip dasar dari BPTT adalah *unfolding.* Secara konseptual, BPTT bekerja dengan membuka gulungan (*unfolding*) setiap *input layer* pada setiap *timestep* (t), kemudian nilai *error* dihitung dan diakumulasikan untuk setiap *timestep* (t). Jaringan kemudian diputar kembali untuk memperbaharui bobot. Pada pelatihan RNN terdapat 3 bobot yang akan diperbaharui dimana U bobot dari *input layer* ke *hidden layer*, W bobot dari *hidden layer* sebelumnya ke *hidden layer* selanjutnya, dan V bobot dari *hidden layer* ke *output layer*. Pada ERNN dalam melakukan BPTT akan mencari nilai turunan dari 3 bobot yaitu U, W, dan V. Adapun rumus penurunan dari setiap bobot adalah sebagai berikut.

1. Penurunan Bobot U

   Rumus turunan bobot U dapat dilihat pada Persamaan (2.8).

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \quad \textbf{(2.8)}$$

   Dimana :

   $\frac{\partial E_t}{\partial U}$: turunan dari nilai *error* ke-t terhadap bobot U

   E_t : nilai *error* ke-t

   U : bobot dari input layer ke hidden layer

   x_t : input ke-t

   t : timestep atau urutan kata.

Persamaan (2.8) dapat diuraikan menjadi Persamaan (2.9).

$$\frac{\partial E_t}{\partial U} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U}\right)\left(\frac{\partial E_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial U}\right)\left(\frac{\partial E_t}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial U}\right)\ldots\left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial U}\right)$$

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U} = \delta_t \otimes x_t \quad \textbf{(2.9)}$$

Rumus penurunan δt dapat dilihat pada persamaan (2.10).

$$\delta_t = [V^T \cdot (y_t - label_t)] \times (1 - h_t^2) \quad \textbf{(2.10)}$$

Dimana :

V : bobot dari *hidden layer* ke *output layer*
y_t : *output state* ke-t
$label_t$: label sebenarnya ke-t
t : *timestep* atau urutan kata
T : simbol Transpose

2. Penurunan Bobot W

Dalam menghitung turunan bobot W dimana setiap *hidden state* (st) terhubung dengan *hidden state* sebelumnya (s_{t-1}) dan begitu seterusnya hingga mencapai *hidden state* pertama s_1 yang terhubung dengan s_0 .

Maka dalam penurunan bobot W harus memperhatikan seluruh *hidden state*. Rumus turunan bobot W ditunjukan persamaan (2.11).

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \quad \textbf{(2.11)}$$

Persamaan (2.11) dapat diuraikan menjadi persamaan (2.12).

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W}\right)\left(\frac{\partial E_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W}\right)\left(\frac{\partial E_t}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial W}\right)\ldots\left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial W}\right) \quad \textbf{(2.12)}$$

Persamaan (2.12) dapat diuraikan menjadi persamaan (2.13).

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W} = \delta_t \otimes h_{t-1} \quad \textbf{(2.13)}$$

Dimana :

$\frac{\partial E}{\partial W}$: turunan dari nilai *error* ke t terhadap bobot W

E_t : nilai *error* ke t

W : bobot dari *hidden layer* sebelumnya ke *hidden layer* sesudahnya

h_t : *hidden state* ke t

t : *timestep* atau urutan token

δ_t didapat dari penurunan. Penurunan δ_t menggunakan Persamaan (2.10).

3. Penurunan Bobot V

Rumus menghitung turunan bobot V dapat dilihat pada Persamaan (2.14).

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \quad \textbf{(2.14)}$$

Persamaan (2.14) dapat diuraikan menjadi Persamaan (2.15).

$$\frac{\partial E_t}{\partial V} = (y_t - label_t) \otimes h_t \quad \textbf{(2.15)}$$

Dimana :

$\frac{\partial E_t}{\partial V}$: turunan dari nilai *error* ke t terhadap bobot V

E_t : nilai *error* ke t

V : bobot dari *hidden layer* ke *output layer*

y_t : nilai dari *output layer* ke t

$label_t$: label sebenarnya ke-t

t : *timestep* atau urutan token

2.3.5 Minibatch Stochastic Gradient Descent

Algoritma pembelajaran mesin atau *neural network* dapat dianggap baik jika dalam prediksi, kesalahan (error) yang diberikan kecil, untuk mengukur kesalahan digunakan *loss function*, seperti *cross entropy*. Jika nilai *loss function* kecil, kemungkinan mendapatkan performa yang baik dalam memprediksi. Untuk

mendapatkan nilai *loss function* yang kecil dipengaruhi oleh nilai bobot yang mana dalam tahap pelatihan, bobot tersebut akan terus diperbaharui untuk mendapatkan bobot optimal. Salah satu algoritma pelatihan adalah *Gradient Descent* (GD).

Gradient Descent adalah metode pelatihan yang digunakan untuk mengukur keakuratan *neural network* dalam melakukan tugas yang diberikan dengan meminimalkan nilai kesalahan. GD bertujuan untuk mengatasi nilai kesalahan (*loss*) yang tinggi dengan memperbarui parameter. Rumus GD diwakili oleh persamaan (2.16).

$$\theta(baru) = \theta(lama) - a \times \frac{\partial E(error)}{\partial \theta(bobot)} \quad \textbf{(2.16)}$$

Dimana :

$\theta(baru)$: nilai bobot yang baru

$\theta(lama)$: nilai bobot yang lama

a : laju pembelajaran (*learning rate*)

$\frac{\partial E(error)}{\partial \theta(bobot)}$: nilai turunan nilai *error* dan bobot

Setelah melakukan pelatihan biasanya didapatkan bobot yang optimal, pada pelatihannya GD menggunakan semua data untuk mendapatkan bobot baru, sedangkan *Stochastic* GD menggunakan satu data. Jika hanya sebagian dari data yang digunakan, maka disebut *Minibatch Stochastic Gradient Descent* (*Minibatch* SGD).

2.4 Klasifikasi Multi-label

Klasifikasi multi-label menjadi perhatian pada peran penelitian dalam pengembangan dan implementasi pembelajaran mesin. Pendekatan klasifikasi multi-label terbagi dalam dua kategori utama, yaitu *Problem Transformation Methods* dan *Algorithm Adaption Methods* [12].

Klasifikasi teks merupakan bagian dari *supervised learning* yang bertujuan untuk mengklasifikasikan data berdasarkan label. Setiap data dapat

dikelompokkan ke dalam satu label, beberapa label, atau tanpa label. Terdapat dua jenis model klasifikasi yaitu single-label dengan panjang label hanya satu dan multi-label dengan panjang label lebih banyak.

Dalam penelitian yang dilakukan oleh Min-Ling Zhang terkait kasus multi-label, salah satu pendekatan yang digunakan dalam kasus multi-label adalah *Problem Transformation*. Pendekatan Problem Transformation memperlakukan masalah multi-label menjadi masalah klasifikasi biner independent. Di sini, setiap masalah klasifikasi biner sesuai dengan label yang dapat disebut relevansi biner [13].

Penelitian ini menggunakan metode klasifikasi biner yang mengambil satu label dari seluruh rangkaian label dan mengklasifikasikan setiap label secara terpisah. Pengklasifikasi biner dilatih untuk setiap label dan output digabungkan untuk menghasilkan sekumpulan label yang diprediksi.

2.5 Akurasi

Akurasi adalah salah satu metode pengukuran performa suatu algoritma. Dalam penelitian ini, perhitungan akurasi digunakan untuk menentukan tingkat akurasi pengklasifikasian ERNN. Persamaan presisi dapat dilihat pada persamaan (2.17)

$$Akurasi = \frac{Jumlah\ Label\ Benar}{Total\ Label} \quad \textbf{(2.17)}$$

2.6 F1 Score

Dalam hal klasifikasi, *F1 Score* dimaksudkan untuk menentukan keseimbangan kinerja pada kelas minoritas dan mayoritas, oleh karena itu *F1 Score* cocok untuk mengukur kinerja algoritma pada kondisi data yang tidak seimbang [14]. Selain *F1 Score*, terdapat juga presisi, akurasi dan recall. Presisi adalah tingkat keakuratan antara informasi yang diminta oleh pengguna dan jawaban yang diberikan oleh sistem, sedangkan recall adalah tingkat keberhasilan sistem dalam mengakses informasi. Dalam penelitian ini, aspek yang berlabel

bukan sentimen lebih banyak dari label positif dan label negatif. Persamaan untuk *F1 Score* dapat dilihat pada persamaan (2.20).

$$recall = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Kelas\ X} \quad \textbf{(2.18)}$$

$$precision = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Prediksi\ terhadap\ Kelas\ X} \quad \textbf{(2.19)}$$

$$F_1\ score(X) = \left(2 \times \frac{(precision \times recall)}{(precision + recall)}\right) \quad \textbf{(2.20)}$$

$$weighted\ F_1\ score = \frac{1}{\Sigma_i |x_i|} \times \sum_i |X|_1\ score(x_i) \quad \textbf{(2.21)}$$

Dimana:

$|x_i|$: jumlah anggota kelas x_i

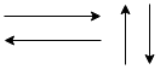
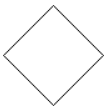
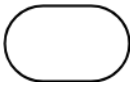
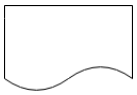
2.7 Pemodelan Sistem

Pemodelan sistem merupakan proses yang menggambarkan rancangan dari sistem yang akan dibangun. Adapun pemodelan yang akan dibagun pada tugas akhir ini digambarkan menggunakan flowchart.

2.7.1 Flowchart

Flowchart adalah jenis diagram yang menampilkan langkah-langkah dan keputusan dalam bentuk simbol-simbol grafis, dan urutannya dihubungkan dengan panah untuk melakukan sebuah proses di dalam suatu program atau prosedur sistem. Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut. Berikut akan dijelaskan mengenai simbol-simbol flowchart yang biasanya dipakai pada Tabel 2.1. [15] :

Tabel 2.1 Simbol Flowchart

Simbol	Nama	Fungsi
	Flow Direction Symbol/Connecting Line	Menghubungkan simbol yang satu dengan yang lainnya, menyatakan arus suatu proses
	Processing	Menunjukkan pengolahan yang akan dilakukan dalam komputer
	Decision	Memilih proses yang akan dilakukan berdasarkan kondisi tertentu
	Predefined Process	Mempersiapkan penyimpanan yang sedang/akan digunakan dengan memberikan harga awal
	Terminal	Memulai atau mengakhiri program
	Input / Output	Menyatakan input dan output tanpa melihat jenisnya
	Document	Menyatakan masukan dan keluaran yang berasal dari dokumen

2.8 Bahasa Pemrograman

Bahasa pemrograman umumnya dikenal sebagai bahasa komputer, adalah kumpulan aturan sintaksis dan semantik yang disusun sedemikian rupa sehingga pengguna komputer dapat membuat program komputer berdasarkan aturan

tersebut. Ada banyak jenis bahasa pemrograman diantaranya PHP, Python, Java, Pascal, C++, dan lain-lain.

2.8.1 Python

Python adalah bahasa pemrograman interpretatif multiguna yang mengusung konsep desain bagus dan sederhana. Python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Selain itu, python juga digunakan untuk membersihkan data, membuat visualisasi, dan membangun model oleh para data scientist saat ini. Pada penelitian ini python digunakan sebagai bahasa pemrograman dalam membangun sistem ASBA.