BAB 2
LANDASAN TEORI

2.1 Bahasa Indonesia

Bahasa Indonesia terdiri dari dua kata yaitu bahasa dan Indonesia dimana tiap kata memiliki arti yang berbeda. Bahasa merupakan sebuah alat yang digunakan untuk berkomunikasi atau berinteraksi agar terjadi kesepahaman. Indonesia merupakan salah satu negara yang berada di Asia Tenggara. Sehingga Bahasa Indonesia merupakan bahasa nasional atau bahasa resmi yang digunakan di Negara Indonesia dan bahasa ini merupakan bahasa persatuan bangsa Indonesia yang dulu diresmikan penggunaannya setelah Proklamasi Kemerdekaan Indonesia.

2.2 Named Entity Recognition

Named Entity Recognition adalah sebuah sub tugas dari ekstraksi informasi yang tujuannya adalah untuk mengklasifikasikan teks dari suatu dokumen atau corpus ke dalam beberapa kategori yang telah ditentukan seperti nama orang, nama lokasi, nama organisasi, bulan, tanggal, waktu dll [1] [2] [3].

2.3 Penelitian Terdahulu

Berikut adalah penelitian-penelitian sebelumnya yang berhubungan dengan penelitian tentang Named Entity Recognition (NER).

1) Penelitian yang dilakukan oleh Hadi Permana [1], dalam Named entity Recognition menggunakan metode Bidirectional LSTM-CRF pada teks bahasa Indonesia. Data masukan yang digunakan adalah berita politik yang diambil dari berbagai situs berita online dan artikel berita tersebut disimpan dengan format file *.txt. preprocessing yang digunakan pada penelitian ini adalah tokenisasi kata menggunakan sebuah library nltk.word_tokenize. Fitur yang digunakan dalam penelitian ini adalah spelling feature dengan mengambil enam fitur berdasarkan dengan pertimbangan dari dataset yang digunakan, fitur-fitur yang digunakan adalah InitCap, AllCap, AllLower,

Digits, ContainDigits, dan Punctuataion. Hasil pengujian akurasi didapat dari dua skenario pengujian, yaitu mengubah jumlah epoch dan mengubah nilai learning rate. Pada skenario pengujian pertama akurasi terbaik yang didapatkan sebesar 87,26% dengan epoch sebanyak 40 dan learning rate sebesar 0.015 sedangkan pada skenario pengujian kedua didapatkan akurasi sebesar 87,77% dengan epoch sebanyak 50 dan learning rate sebesar 0.001.

2) Lalu pada penelitian yang dilakukan oleh Agus Willyawan [2], dalam Named Entity Recognition untuk bahasa Indonesia Menggunakan metode Conditional Random Field dan fitur yang digunakan adalah POS Tagging. Dataset yang digunakan pada penelitian ini adalah data Artikel Wikipedia dan Dbpedia Indonesia. Hasil yang didapatkan dari penelitian ini adalah F_1 Score pada PERSON, ORGANIZATION, dan LOCATION sebesar 72%, 66%, dan 84%.
3) Selanjutnya pada penelitian yang dilakukan oleh Li et al [3], dalam Biomedical Named Entity Recognition menggunakan Extended Recurrent Neural Network. Untuk merepresentasikan data pada penelitian ini menggunakan word embeddings yang dilatih oleh Word2Vec. Data masukan yang digunakan berasal dari BioCreative II GM corpus dimana mengandung 15000 kalimat untuk data training dan 5000 kalimat untuk data testing. Hasil yang didapatkan pada penelitian ini adalah F_1 Score terbaik sebesar 81,87 untuk Extended Recurrent Neural Network; F_1 Score terbaik untuk ERNN adalah 79,21; dan F_1 Score terbaik untuk CRF adalah 78,88.

2.4 Preprocessing

Preprocessing adalah sebuah tahap yang mengubah data masukan menjadi data yang layak diproses oleh Algoritma utama. Berikut adalah sebagian penjelasan dari Preprocessing yang digunakan.

2.4.1 Penggunaan Kata Spesial

Kata spesial adalah kata digunakan sebagai pengganti dari kata yang tidak ada pada kamus kata. Berikut adalah kata spesial yang digunakan beserta penjelasannya.

1) number_

   Kata ini berguna sebagai pengganti dari kata yang berupa kombinasi dari angka dan tidak ada pada kamus kata atau kata yang baru ditemukan pada data masukan.

2) unknown_

   Kata ini berguna sebagai pengganti dari kata yang tidak ada pada kamus kata atau kata yang baru ditemukan pada data masukan.

2.4.2 Penyusunan Kalimat

Penyusunan kalimat adalah proses menyusun beberapa kata menjadi sebuah kelompok kata berdasarkan pemisah atau *separator* yang ditentukan. Proses ini berguna untuk membagi kata-kata ke dalam kelompok kata.

2.4.3 Case Folding

Case Folding adalah sebuah proses yang mengubah tiap huruf pada kata menjadi huruf kecil atau huruf besar. Proses ini berguna untuk penerapan suatu algoritma agar tidak membedakan dua potongan kata yang sama hanya karena perbedaan huruf besar dan kecil.

2.4.4 Ekstraksi Fitur

Ekstraksi Fitur adalah sebuah proses pengambilan ciri suatu objek yang dapat menggambarkan objek tersebut. Proses ini berguna untuk mengekstrak ciri-ciri yang tidak disebutkan pada data masukan.

2.4.5 Kamus Kata

Kamus kata merupakan sejenis buku rujukan yang menyimpan kumpulan kata. Di dalam kamus kata tidak terdapat kata yang sama sehingga tiap kata yang ada pada kamus kata berbeda satu sama lain.

2.4.6 One Hot Encoding

One Hot Encoding merupakan salah satu teknik dalam merepresentasikan huruf ke dalam angka. Biasanya angka yang merepresentasikan huruf adalah Vektor. Karena banyak metode yang tidak bisa mengolah huruf sebagai data masukan sehingga huruf tersebut perlu di representasikan ke dalam angka. One Hot Encoding akan memberikan nilai 1 pada id yang sama dengan huruf yang sama pada kamus kata.

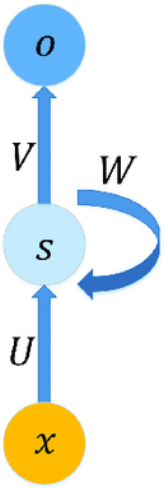
2.5 Recurrent Neural Network

Recurrent Neural Network adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memproses masukan yang biasanya berupa data sekuensial. RNN masuk dalam kategori *deep learning* karena data diproses melalui banyak lapis (layer). RNN telah mengalami kemajuan yang pesat dan telah merevolusi bidang-bidang seperti pemrosesan bahasa alami, pengenalan suara, sintesa musik, pemrosesan data finansial seri waktu, analisis deret DNA, analisis video, dan sebagainya.

2.5.1 Elman Recurrent Neural Network

Elman Recurrent Neural Network (ERNN) merupakan salah satu varian dari *Recurrent Neural Network*. ERNN juga termasuk ke dalam *Simple Recurrent Neural Network* [8]. ERNN membuat salinan hidden layer pada layer input yang disebut *context layer* (hidden layer sebelumnya). Context layer ini berfungsi sebagai perpanjangan dari input layer. Context layer memiliki fungsi sebagai penyimpanan status atau keadaan sebelumnya dari hidden layer, untuk kemudian disampaikan kembali pada hidden layer saat ini. Hubungan antara *context layer* dan

hidden layer adalah *fully connected* dan diberi bobot 1. Ilustrasi ERNN dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Elman Recurrent Neural Network [24]

Pada Gambar 2.1 tedapat parameter 3 parameter bobot yaitu U, V, dan W. Dimana U adalah bobot yang ada bobot antara *input layer* dan *hidden layer*, V adalah bobot antara *hidden layer* dan *output layer*, dan W adalah input antara *hidden layer* dan *context layer*. Pada proses training ketiga bobot tersebut akan terus berubah sampai mendapatkan bobot yang optimal atau sampai iterasi selesai. Saat melakukan *Forward Propagation* ERNN akan menggunakan Persamaan 2.1.

$$s_t = tanh(W.s_{(t-1)} + U.X_t) \quad (2.1)$$

Pada persamaan 2.1 dapat dilihat bahwa persamaan tersebut memiliki beberapa variabel yaitu, s_t merupakan variabel dari *hidden state* pada *timestep* ke-t, *W adalah* bobot dari *hidden layer* ke *hidden layer*, *U* adalah bobot dari *input layer* ke *hidden layer*, X_t adalah Vektor input, dan *t* adalah *timestep* atau kata.

Sedangkan nilai *output state* waktu ke-t didapatkan dengan menggunakan Persamaan 2.2.

$$o_t = softmax(V.s_t) \quad (2.2)$$

Pada persamaan 2.2 dapat dilihat bahwa persamaan tersebut memiliki beberapa variabel yaitu, o_t adalah *output state* pada waktu ke-t, *V* adalah bobot dari *hidden*

layer ke *output layer*, s_t adalah *hidden state* pada waktu ke-t, dan *t* adalah *timestep* atau kata.

2.5.2 Inisialisasi Bobot

Pada ERNN terdapat 3 bobot: yaitu U, V, dan W. Aturan dimensi dari ketiga bobot tersebut dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Aturan Inisialisasi Bobot Awal

Bobot	Keterangan
$U \in \mathbb{R}^{HxI}$	Adalah matriks berukuran HxI dimana H adalah dimensi hidden *(Hidden_dim)* yang ditentukan manual dan I adalah dimensi vektor data masukan *(Input_dim)*.
$V \in \mathbb{R}^{OxH}$	Adalah matriks berukuran OxH dimana O adalah dimensi vektor label *(Output_dim)* dan H adalah dimensi hidden *(Hidden_dim)* yang ditentukan manual.
$W \in \mathbb{R}^{HxH}$	Adalah matriks berukuran HxH dimana H adalah dimensi hidden *(Hidden_dim)* yang ditentukan manual.

Berikut adalah teknik inisialisasi yang digunakan dalam penelitian ini [18].

1) Elemen-elemen U merupakan **nilai acak** yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan 2.3,

$$BA(U) = -\frac{1}{\sqrt{Input\_dim}} \text{ dan } BB(U) = \frac{1}{\sqrt{Input\_dim}} \quad (2.3)$$

2) Elemen-elemen V merupakan **nilai acak** yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan 2.4,

$$BA(V) = -\frac{1}{\sqrt{Hidden\_dim}} \text{ dan } BB(V) = \frac{1}{\sqrt{Hidden\_dim}} \quad (2.4)$$

3) Elemen-elemen W merupakan **nilai acak** yang berada pada interval BA hingga BB dimana untuk menentukannya menggunakan persamaan 2.5,

$$BA(W) = -\frac{1}{\sqrt{Hidden\_dim}} \text{ dan } BB(W) = \frac{1}{\sqrt{Hidden\_dim}} \quad (2.5)$$

Keterangan, dimana *Input_dim* adalah ukuran dari dimensi Input dan *Hidden_dim* adalah ukuran dari dimensi Hidden.

2.5.3 Fungsi Aktivasi

Fungsi aktivasi sangat umum digunakan dalam *neural network*. Alasan utama menggunakan fungsi aktivasi agar neural network mengenali data *non-linear*, karena output yang dihasilkan dari *neural network* jarang sekali bersifat linear [9]. Fungsi aktivasi juga digunakan untuk mengaktifkan dan menonaktifkan neuron, karakteristik *neural network* ditentukan oleh bobot dan input-output fungsi aktivasi yang diterapkan. Terdapat jenis fungsi aktivasi pada neural network, namun yang digunakan dalam penelitian ini adalah *hyperbolic tangent* (*tanh*) dan *softmax*.

2.5.2.1 Tanh (Hyperbolic Tangent)

Fungsi aktivasi *hyperbolic tangent* adalah tipe fungsi aktivasi dalam *deep learning* dan memiliki bebrapa varian, fungsi aktivasi *hyperbolic tangent* dikenal juga sebagai fungsi *tanh* yang menghasilkan nilai -1 sampai 1 [9]. Fungsi tanh memberikan kinerja pelatihan yang lebih baik bagi *multi layer neural network* dibandingkan fungsi *sigmoid*. Fungsi tanha telah banyak digunakan dalam *Recurrent Neural Network* pada kasus *Natural Language Processing* dan *Speech Recognition*. Persamaan dari fungsi *tanh* dapat dilihat pada Persamaan 2.6.

$$\mathrm{Tanh}(x) = \frac{\sin(x)}{\cos(x)} = \left(\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\right) = \left(\frac{e^{2x}-1}{e^{2x}+1}\right) \tag{2.6}$$

2.5.2.2 Softmax

Fungsi aktivasi *softmax* digunakan untuk menghitung probabilitas pada hasil output, yang terjadi di *output layer* dimana akan diambil nilai probabilitas yang paling besar sebagai prediksi. *Softmax* digunakan untuk menghitung probabilitas distribusi dari Vektor bilangan real. Fungsi *softmax* menghasilkan output yg berada pada kisaran 0 dan 1, dengan jumlah probabilitas 1 [9]. Persamaan pada fungsi *softmax* dapat dilihat pada Persamaan 2.7.

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (2.7)$$

2.5.2.3 Loss Function

Loss Function bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai sebenarnya [10]. Ada beberapa jenis *loss function* diantaranya *Mean Square Error* (MSE) dan *Cross Entrophy* (CE). Namun pada praktiknya CE lebih cepat dalam konvergensi dan mendapatkan hasil lebih baik dalam tingkat klasisfikasi [10]. CE biasa digunakan pada *neural network* dan dalam kasus *multi class classification.* Pada penelitian ini *loss function* yang digunakan adalah *Cross Entrophy* (CE). Persamaan CE dapat dilihat pada Persamaan 2.8.

$$error = \sum_i E_t \quad (2.8)$$

Dimana E_t dapat dilihat pada Persamaan 2.9.

$$E_t = -\sum_i y_t \cdot \log o_t \quad (2.9)$$

Pada Persamaan 2.9 terdapat beberapa variabel yaitu, y_t adalah label sebenarnya pada *timestep* ke-t, o_t adalah label prediksi pada timestep ke-t, y_t adalah Vektor y_t (label sebenarnya), o_t adalah Vektor o_t (*output state* atau dugaan sistem), dan E_t adalah error dari output o_t.

2.6 Backpropagation Through Time

Backpropagation through time (BPTT) merupakan algoritma yang biasa digunakan untuk menghitung nilai turunan terhadap error. Pada *Recurrent Neural Network* nilai error dapat di *backpropagate* lebih jauh daripada *neural network* biasa. Prinsip dasar dari BPTT adalah *unfolding* [12]. 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 memperbarui bobot. Pada pelatihan ERNN terdapat 3 bobot yang akan dicari nilai turunannya yaitu U, V, dan W.

2.6.1 Penurunan Bobot U

Rumus turunan bobot *W* dapat dilihat pada Persamaan 2.10.

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \tag{2.10}$$

Keterangan, dimana $\frac{\partial E_t}{\partial U}$ adalah turunan dari nilai error ke-t terhadap bobot U, E_t adalah nilai error ke-t, U adalah bobot dari input layer ke hidden layer, X_t adalah input ke-t, dan t adalah timestep atau urutan kata. Persamaan **(2.10)** dapat diuraikan menjadi Persamaan 2.11.

$$\frac{\partial E_t}{\partial U} = \left(\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial U}\right)\left(\frac{\partial E_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial U}\right)\left(\frac{\partial E_t}{\partial s_{t-2}}\frac{\partial s_{t-2}}{\partial U}\right)\ldots\left(\frac{\partial E_t}{\partial s_1}\frac{\partial s_1}{\partial U}\right) \tag{2.11}$$

Persamaan 2.11 dapat diuraikan menjadi Persaman 2.12.

$$\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial U} = \delta_t \otimes X_t \tag{2.12}$$

Pencarian nilai δ_t dapat dilihat pada Persamaan 2.13.

$$\delta_t = [V^T \cdot (o_t - y_t)] \odot (1 - {s_t}^2) \tag{2.13}$$

Keterangan dari Persamaan 2.13, dimana V adalah bobot dari hidden layer ke output layer, o_t adalah output state ke-t, y_t adalah label sebenarnya ke-t, t adalah timestep atau urutan kata, dan *T* adalah simbol Transpose.

2.6.2 Penurunan Bobot V

Rumus menghitung turunan bobot V dapat dilihat pada Persamaan 2.14.

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \tag{2.14}$$

Persamaan 2.14 dapat diuraikan menjadi Persamaan 2.15.

$$\frac{\partial E_t}{\partial V} = \sum_t (o_t - y_t) \otimes s_t \tag{2.15}$$

2.6.3 Penurunan Bobot W

Bobot dari input dalam menghitung turunan bobot W dimana setiap hidden state (s_t) 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 *hidden state*. Rumus turunan bobot W ditunjukan oleh Persamaan 2.16.

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{2.16}$$

Persamaan 2.16 dapat diuraikan menjadi Persamaan 2.17.

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial W}\right) + \left(\frac{\partial E_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial W}\right) + \left(\frac{\partial E_t}{\partial s_{t-2}}\frac{\partial s_{t-2}}{\partial W}\right) + \cdots \left(\frac{\partial E_t}{\partial s_1}\frac{\partial s_1}{\partial W}\right) \tag{2.17}$$

Persamaan 2.17 dapat diuraikan menjadi Persaman 2.18.

$$\frac{\partial E_t}{\partial s_t}\frac{\partial s_t}{\partial W} = \delta_t \otimes s_{t-1} \tag{2.18}$$

Pada Persamaan 2.18 terdapat beberapa variabel yaitu, $\frac{\partial E}{\partial W}$ adalah 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, s_t adalah *hidden state* ke-t, t adalah *timestep* atau urutan kata, dan δ_t didapat dari penurunan. Penuruan δ_t menggunakan Persamaan 2.13.

2.7 Testing

Testing menggunakan Algoritma ERNN akan melakukan beberapa langkah seperti Training ERNN dimana Training ERNN melakukan proses Forward dan BPTT sedangkan Testing ERNN hanya akan melakukan proses Forward Propagation yang terdiri dari pencarian hidden state yang menggunakan Persamaan 2.1 dan pencarian output state yang menggunakan Persamaan 2.2. berikut adalah tahap-tahap yang dilakukan pada Testing ERNN :

1) Ambil satu data/kalimat.
2) Ambil satu kata.

3) Mencari hidden state dari kata yang diambil dengan menggunakan Persamaan 2.1.
4) Mencari output state dari kata yang diambil dengan menggunakan Persamaan 2,2.
5) Ambil data/kalimat selanjutnya.
6) Ulangi tahap 2, 3, dan 4 pada semua kata.

2.8 Minibatch Stochastic Gradient Descent

Algoritma *machine learning* atau *neural network* dapat dikatakan bagus bila dalam memprediksi memberikan sedikit kesalahan (error), untuk mengukur kesalahan digunakan *loss function*, seperti *cross entrophy*. Jika nilai *loss function* kecil kemungkinan dalam memprediksi mendapatkan performa yang baik. Dalam mendapatkan nilai *loss function* yang kecil tersebut dipengaruhi nilai bobot dimana dalam tahap pelatihan bobot akan terus diperbarui untuk mendapatkan bobot yang optimal. Salah satu algoritma pelatihan aalah *Gradient Descent* (GD).

Gradient Descent adalah salah satu metode pelatihan dengan meminimalkan nilai error untuk mengukur keakuratan *neural network* dalam melakukan tugas yang diberikan. GD bertujuan mengatasi nilai error (*loss*) yang tinggi dengan cara memperbarui parameter. Adapun rumus dari GD dapat dilihat pada Persamaan 2.19.

$$bobot_{baru} = bobot_{lama} - \left(\alpha \frac{\partial(error)}{\partial(bobot)}\right) \quad (2.19)$$

Pada persamaan 2.19 terdapat beberapa variabel yaitu, α adalah *learning rate* dan $\frac{\partial(error)}{\partial(bobot)}$ merupakan nilai turunan nilai error dan bobot. :

Setelah melakukan pelatihan biasanya bobot yang optimal didapatkan, dalam pelatihannya GD menggunakan seluruh data untuk mendapatkan bobot yang baru sedangkan *stochastic* GD menggunakan satu buah data. Jika yang digunakan dadalam pelatihan hanya sebagian saja maka biasa dikenal sebagai *Minibatch Stochastic Gradient Descent (Minibatch SGD)*.

2.9 Confusion Matrix

Confusion Matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining. Confusion matrix digambarkan dengan tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan [21].

Tabel 2. 2 Confusion Matrix

Prediction	**Actual**	
	Positive	**Negative**
Positive	TP	FP
Negative	FN	TN

Keterangan :

1. True Positive (TP) adalah jumlah data kelas positif yang diklasifikasikan ke dalam nilai positif.
2. False Positive (FP) adalah jumlah data pada kelas negatif yang diklasifikasikan ke dalam nilai positif.
3. False Negative (FN) adalah jumlah data pada kelas positif yang diklasifikasikan ke dalam nilai negatif.
4. True Negative (TN) adalah jumlah data kelas negatif yang diklasifikasikan ke dalam kelas negatif.

2.10 F_1 Score

Perhitungan F_1 Score dilakukan per kelas lalu diambil nilai keseluruhan dengan cara diratakan (*averaged*). F_1 Score memiliki beberapa versi; yaitu micro, macro, dan weighted macro F_1 Score yang persamaannya dapat dilihat pada Persamaan 2.20 [22].

$$weighted\ F_1\ score = \frac{1}{\sum |X_i|} \times \sum_i |X| F_1\ score(X_i) \quad (2.20)$$

Dimana untuk mencari F_1 Score akan menggunakan Persamaan 2.23. Pada Persamaan 2.23 terdapat Recall dan Precision dimana untuk mencari Recall akan menggunakan Persamaan 2.21 sedangkan untuk mencari Preicison akan menggunakan Persamaa 2.22 [21].

$$Recall = \frac{TP}{TP+FN} \tag{2.21}$$

$$Precision = \frac{TP}{TP+FP} \tag{2.22}$$

$$F_1\ score(X) = 2 \times \frac{Recall \times Precision}{Recall+Precision} \tag{2.23}$$

Keterangan dari Persamaan (2.21), (2.22), dan (2.23). TP merupakan jumlah total dari hasil prediksi yang sama dengan label sebenarnya, FN merupakan jumlah total dari hasil prediksi yang tidak sama dengan label sebenarnya. FP merupakan jumlah total dari hasil prediksi yang benar tetapi berbeda dengan label sebenarnya, *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi, *Precision* adalah ketepatan antara informasi yang diminta dengan hasil prediksi, dan *X* merupakan kelas atau label.

2.11 Akurasi

Akurasi merupakan salah satu metode pengukuran performa algoritma untuk kasus klasifikasi. Metode ini memberikan informasi mengenai jumlah persentase keberhasilan sebuah algoritma melakukan prediksi secara benar. Karena NER termasuk kedalam *multiclass labelling* maka diperlukan pencarian akurasi untuk setiap kelas dan untuk mencari akurasi tiap kelas akan menggunakan Persamaan 2.24 [21].

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{2.24}$$

Sedangkan untuk menghitung Akurasi total untuk seluruh data dapat dicari menggunakan Persamaan 2.25 [23].

$$Accuracy = \frac{Jumlah\ Prediksi\ Benar}{Jumlah\ Label\ Total} \times 100\% \tag{2.25}$$

2.12 Matriks

Matriks adalah himpunan skalar yang disusun secara empat persegi panjang menurut baris (m) dan kolom (n). Skalar-skalar tersebut disebut elemen matriks. Matriks biasanya menggunakan batas seperti ( ), [ ], atau || ||. Berikut adalah salah satu contoh dari matriks.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

Pada matriks A terlihat bahwa matriks A memiliki dimensi 3 x 2. Selain itu terdapat matriks baris atau matriks kolom dimana salah satu dimensi bernilai 1. Berikut adalah contoh dari matriks kolom.

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

2.12.1 Transpose Matriks

Transpose matriks yaitu suatu matriks yang melakukan pertukaran antara dimensi kolom dan barisnya atau sebuah matriks yang memindahkan elemen-elemen pada kolom menjadi elemen baris atau sebaliknya. Berikut adalah contoh dari Transpose matriks.

$$A = \begin{bmatrix} 1 & 7 \\ 2 & 8 \\ 3 & 9 \end{bmatrix} \; maka \; A^T = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

2.12.2 Outer Product

Outer product merupakan perkalian perkalian antara dua buah vektor yang akan menghasilkan sebuah matriks. Berikut adalah persamaan dari outer product.

$$C = A \otimes B$$

$$= A \cdot B^T$$

$$= \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \cdot [b_1 \quad b_2 \quad b_3]$$

$$= \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \\ a_4b_1 & a_4b_2 & a_4b_3 \end{bmatrix}$$

2.12.3 Hadamard Product

Hadamard Product merupakan perkalian dua matriks berdimensi sama yang akan menghasilkan matriks dengan dimensi yang sama.

$$C = A \odot B$$

$$= \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} & a_{21}b_{21} \\ a_{12}b_{12} & a_{22}b_{22} \end{bmatrix}$$

2.13 Pemodelan Sistem

Pemodelan sistem merupakan proses yang menggambarkan rancangan dari sistem yang akan dibangun.

2.13.1 Diagram Konteks

Diagram Konteks merupakan level nol dari Data Flow Diagram atau sebuah diagram yang menampilkan rancangan sistem secara garis besar. Diagram konteks hanya menampilkan aliran data dari luar sistem dan hasil dari sistem.

2.13.2 Data Flow Diagram

Data Flow Diagram menampilkan gambaran secara detail dari perancangan sistem yang akan dibangun. Data Flow Diagram juga menampilkan alur datanya dari luar sistem sampai mengeluarkan hasil dari sistem.

2.14 Bahasa Pemrograman

Bahasa pemrograman adalah sekumpulan aturan yang disusun sedemikian rupa sehingga pengguna komputer dapat membuat program komputer berdasarkan

aturan tersebut [6]. Terdapat banyak macam bahasa pemrograman, diantaranya Java, PHP, Phyton, C, C++, Pascal, dan lain-lain.

2.15 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python merupakan bahasa pemrograman yang populer dan banyak digunakan oleh Data Analysts, Data Scientists dan para Software Engineers untuk menjalankan proses pembangunan sebuah aplikasi dan untuk menggali lebih dalam machine learning.

2.16 Jupyter Notebook

Jupyter adalah organisasi non-profit untuk mengembangkan software interaktif dalam berbagai bahasa pemrograman. Notebook adalah satu software buatan Jupyter, adalah aplikasi web open-source yang memungkinkan dan berbagi dokumen interaktif yang berisi kode live, persamaan, visualisasi, dan teks naratif yang kaya.

2.17 Flask

Flask adalah sebuah aplikasi microframework untuk bahasa Python yang dibuat dengan toolkit Werkzeug dan template Jinja2. Penggunaan aplikasi tidak membutuhkan pekerjaan yang sangat kompleks dari sisi pengguna dalam mengelola websitenya. Kompleks disini dalam artian tidak membutuhkan installasi server kembali dengan kata lain installasi sudah dilakukan di flask.