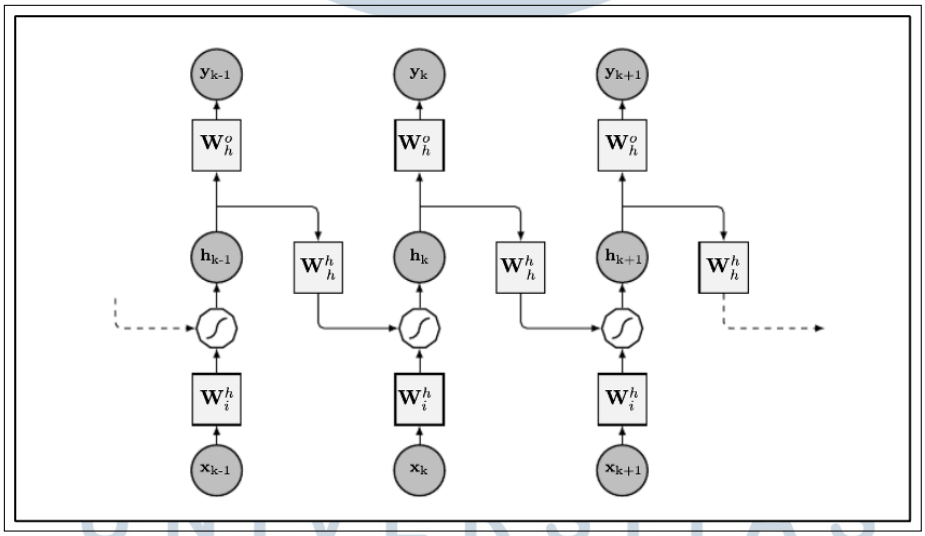
BAB 2
LANDASAN TEORI

Pada penelitian ini, terdapat beberapa literatur yang perlu dipahami, yaitu teori RNN, LSTM, BiLSTM, CRF, Viterbi, BiLSTM + CRF, NER, BIO-*Format*, dan POS-*Tagging*.

2.1. Recurrent Neural Network (RNN)

Menurut Abdullah Aziz Sharfuddin, dkk. [13], RNN merupakan sebuah cabang dari sistem *neural network* yang membentuk suatu memori melalui keterhubungan setiap *node*-nya. Dalam RNN, suatu masukan pada sebuah *node* atau *layer* akan mempengaruhi masukan pada *node* atau *layer* berikutnya. Karena sifat tersebut, RNN banyak digunakan untuk melakukan klasifikasi pada data *time series* atau data yang bersifat sekuensial [14]. Arsitektur dari RNN dapat diamati pada gambar berikut [15]:



Gambar 2.1. Arsitektur *Recurrent Neural Network*
Sumber: [15]

Pada Gambar 2.1, simbol x_t melambangkan masukan pada time step t, simbol h_t merupakan hidden state pada time step t, dan y_t adalah keluaran dari *time step* t. Selain itu, adapun bentuk formal dari fungsi RNN yang dapat diamati sebagai berikut [15]:

$$h[t] = f(W_i^h(x[t] + b_i) + (W_h^h(h[t-1] + b_h)) \tag{2.1}$$

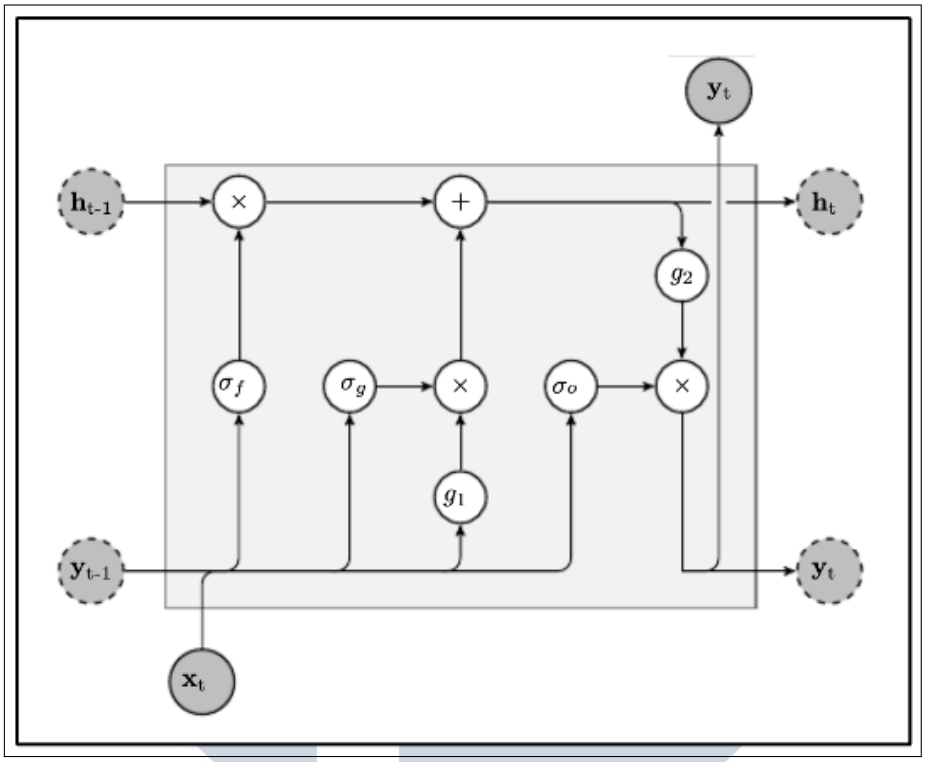
$$y[t] = g(W_h^o(h[t] + b_o) \tag{2.2}$$

Pada persamaan tersebut, simbol f melambangkan fungsi aktivasi yang digunakan pada *neuron* dalam RNN, yang biasa berupa fungsi *sigmoid* atau *hyperbolic tangent*.Fungsi aktivasi tersebut menerima masukan berupa matriks *input weight* (W_i^h) dari *input* terkini ($x[t]$) dan bias *input* (b_i) dan matriks *hidden weight* (W_h^h) dari *hidden state* sebelumnya ($h[t-1]$) dan bias *hidden* (b_h). Selain itu, terdapat simbol g yang melambangkan fungsi transformasi yang digunakan pada sebagai keluaran dari *time step* t. Fungsi transformasi yang digunakan biasanya berupa fungsi *linear* yang menerima masukan berupa matriks *output weight* (W_h^o) dari *hidden state* terkini ($h[t]$) dan bias *output* (b_o).

2.2. Long Short-Term Memory (LSTM)

LSTM atau *Long Short-Term Memory* merupakan sebuah metode hasil pengembangan dari RNN [16]. LSTM dirancang untuk mengatasi permasalahan *vanishing gradient* yang terjadi pada RNN. LSTM bekerja dengan mempertahankan nilai *error* yang konstan dan tidak menambahkan bias pada beberapa *state* terbaru [15]. Karena hal tersebut, LSTM lebih baik digunakan dalam mengolah data sekuensial yang besar jika dibandingkan dengan RNN.

Unit pemrosesan *internal* pada LSTM dikenal dengan sebutan *cell* [15]. Isi dari *cell* tersebut dikontrol dan dijaga oleh gerbang atau *gates*. Ada 3 macam *gates* yang digunakan dalam LSTM, yaitu *input* (*update*) *gate*, *output gate*, dan *forget gate*. *Input* atau *update gate* akan mengatur nilai yang akan dimasukkan dan diperbarui, *output gate* akan mengatur nilai yang akan dikeluarkan, dan *forget gate* akan mengatur nilai yang akan diingat dan dilupakan [14].



Gambar 2.2. Arsitektur *Long Short-Term Memory*

Sumber: [15]

Lalu, bentuk umum dari persamaan LSTM adalah sebagai berikut [15]:

$$updategate : \alpha_u[t] = \alpha(W_u x[t] + R_U y[t-1] + b_u) \tag{2.3}$$

$$outputgate : \alpha_o[t] = \alpha(W_o x[t] + R_o y[t-1] + b_o) \tag{2.4}$$

$$forgetgate : \alpha_f[t] = \alpha(W_f x[t] + R_f y[t-1] + b_f) \tag{2.5}$$

$$candidategate : \bar{h}[t] = g_1(W_h x[t] + R_h y[t-1] + b_h) \tag{2.6}$$

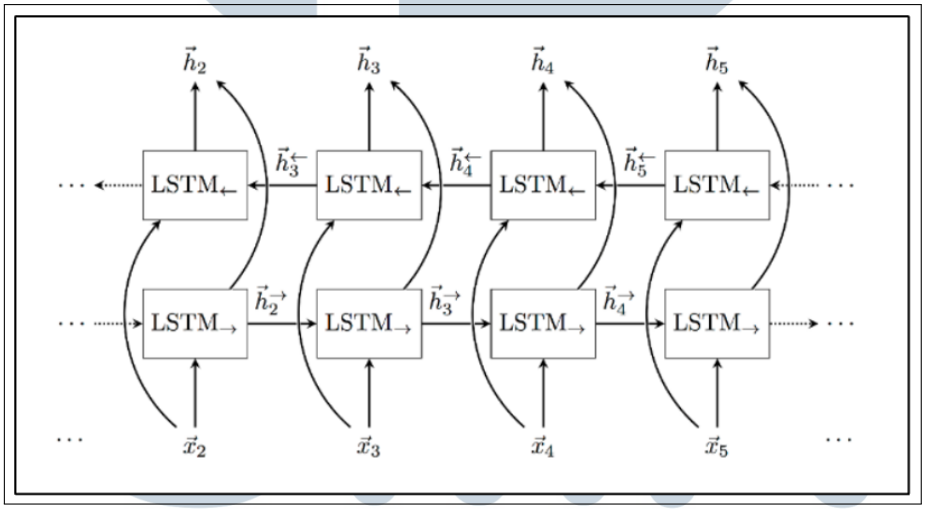
$$cellstate : h[t] = \alpha_u[t] \odot \bar{h}[t] + \alpha_f[t] \odot h[t] \tag{2.7}$$

$$output : y[t] = \alpha_o[t] \odot g_2(h[t]) \tag{2.8}$$

Pada persamaan-persamaan tersebut, terdapat simbol W yang merupakan matriks yang mengatur *weight* untuk setiap masukan yang digunakan pada *cell*, simbol R yang melambangkan matriks yang mengatur *weight* untuk *recurrent connections*, dan b yang merupakan nilai vektor untuk bias. Selain itu, terdapat simbol α yang merupakan fungsi *sigmoid*, serta simbol g_1 dan g_2 yang merupakan fungsi aktivasi yang biasanya berupa fungsi *hyperbolic tangent*.

2.3. Bidirectional Long Short-Term Memory (BiLSTM)

Pada *unidirectional* LSTM, informasi hanya diproses dari salah satu arah (*backward* atau *forward*) [13]. Berbeda dengan LSTM, BiLSTM bekerja dengan memproses informasi dari dua arah (*backward* dan *forward*) dan *hidden layer* yang terpisah [17]. *Forward layer* berguna untuk memproses informasi dari data sebelumnya dan *backward layer* berguna untuk memproses informasi dari data setelahnya. Cara tersebut dapat membantu meringankan ambiguitas pada data [18].



Gambar 2.3. Arsitektur BiLSTM

Sumber: [17]

Nilai keluaran dari BiLSTM dapat dilihat pada persamaan berikut [17]:

$$y_t = W_{\overrightarrow{h}_y} \overrightarrow{h}_t + W_{\overleftarrow{h}_y} \overleftarrow{h}_t \tag{2.9}$$

Pada persamaan tersebut, y_t melambangkan nilai keluaran dari *state* t dengan cara menerima nilai *weight forward layer* dikalikan dengan *hidden state forward layer* ($W_{\overrightarrow{h}_y} \overrightarrow{h}_t$) dan nilai *weight backward layer* dikalikan dengan *hidden state*

backward layer ($W_{\overleftarrow{h}_y} \overleftarrow{h}_t$).

2.4. Conditional Random Field (CRF)

Menurut Wallach [19], CRF merupakan sebuah model probabilistik yang digunakan untuk melakukan segmentasi dan pemberian label pada sekuens data. CRF berguna untuk memprediksi sejumlah keluaran dari jumlah fitur masukan yang banyak [10]. Salah satu model CRF sederhana yang cukup banyak digunakan adalah *Linear-chain* CRF. *Linear-chain* CRF banyak digunakan untuk melakukan *sequence labelling*. Adapaun persamaan *Linear-chain* CRF yaitu sebagai berikut [20]:

$$p(y|x) = \frac{1}{Z(x)} \prod_c \phi_c(y_c, x) \tag{2.10}$$

$$Z(x) = \sum_y \prod_c \phi_c(y_c, x) \tag{2.11}$$

Pada persamaan tersebut, $p(y|x)$ merupakan probabilitas kondisional kelas y terhadap kata x, ϕ_c melambangkan fungsi positif atau potensial, dan $Z(x)$ melambangkan fungsi normalisasi distribusi probabilitas label terhadap data x.

2.5. Algoritma Viterbi

Algoritma Viterbi merupakan suatu algoritma pencarian solusi optimal melalui sebuah jalur yang paling memungkinkan. Algoritma Viterbi bekerja dengan cara meminimalisir probabilitas kesalahan dengan membandingkan kemungkinan dari suatu *state* transisi yang dapat terjadi dan memutuskan suatu *state* dengan probabilitas yang paling tinggi [21]. Algoritma Viterbi berperan dalam efisiensi algoritma *Linear-chain* CRF, karena algoritma Viterbi memiliki metode yang cukup efektif dalam mencari dan mempertimbangkan *path* terbaik dari seluruh *state* yang tersedia [22]. Adapun persamaan algoritma Viterbi yaitu sebagai berikut[23]:

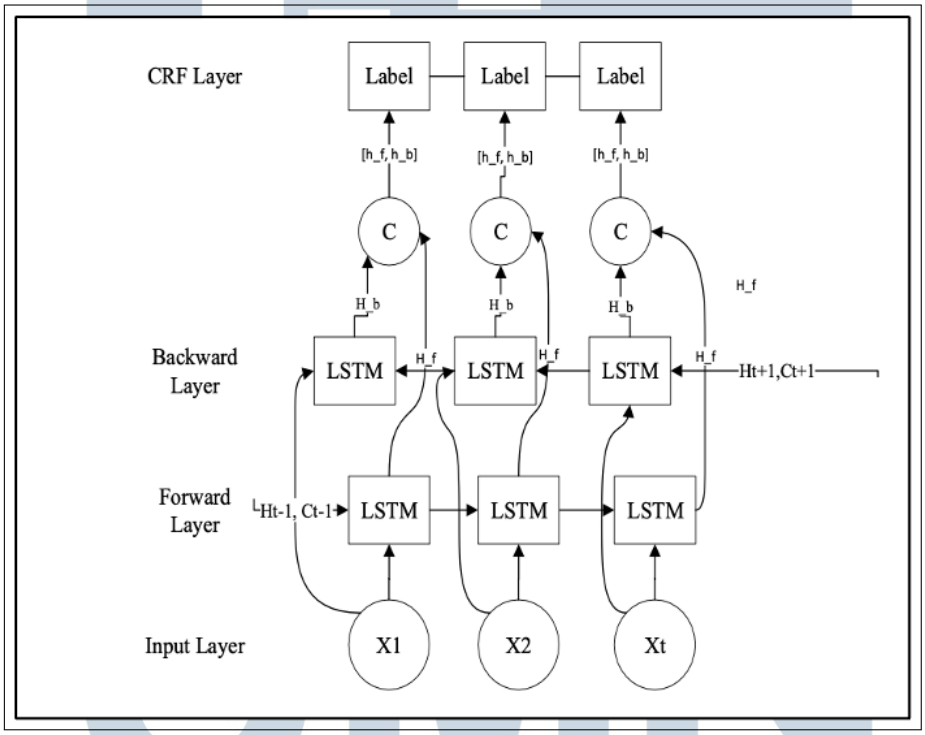
$$\delta_t(j) = \max_{i \in S} \Psi_t(j, i, x_t) \delta_{t-1}(i) \tag{2.12}$$

Pada persamaan tersebut, $\delta_t(j)$ melambangkan nilai probabilitas tertinggi

untuk pada pengamatan data ke-j dan $\Psi_t(j,i,x_t)$ melambangkan matriks yang berisi *weight* transisi dari *state* yang sudah ditentukan.

2.6. BiLSTM-CRF

Model BiLSTM-CRF merupakan sebuah model yang dikembangkan berdasarkan gabungan metode BiLSTM dan CRF. Berikut adalah contoh arsitektur model BiLSTM-CRF yang dikembangkan pada penelitian Permana dan Purnamasari [17]:



Gambar 2.4. Arsitektur BiLSTM-CRF

Sumber: [17]

Model tersebut menggunakan BiLSTM untuk melakukan perhitungan vektor kata untuk mendapatkan nilai kemungkinan *tag* dari setiap kata dan CRF untuk melakukan perhitungan probabilitas dari *tag* ke setiap kata. Pada penelitian ini, BiLSTM akan digunakan untuk melakukan pemrosesan fitur kata dan POS-*tag* dan CRF digunakan untuk mencari *tag* dengan probabilitas terbaik dari fitur yang digunakan.

2.7. Named Entity Recognition (NER)

Menurut Mansouri, dkk. [24], NER merupakan sebuah tugas atau masalah yang merupakan bagian dari ekstraksi informasi yang melibatkan suatu pemrosesan dokumen yang terstruktur maupun tidak terstruktur dan melakukan identifikasi suatu pernyataan. Pada NER, sebuah pernyataan akan diidentifikasi dan diklasifikasi ke dalam beberapa kelas, misalnya nama orang, tempat, organisasi, dan lain-lain. Kelas tersebut diperkenalkan dengan istilah entitas bernama atau named entity pada *Message Understanding Conference* keenam (MUC-6)[24]. Seringkali *named entity* memegang informasi yang penting dalam sebuah pernyataan atau dokumen, maka biasanya hasil dari proses NER akan dijadikan masukan untuk tugas NLP lainnya, seperti machine translation, question answering, dan lain-lain [25].

2.8. BIO-Format

BIO-*Format* merupakan salah satu bentuk tagging format yang banyak digunakan pada penugasan NLP [26]). Terdapat 3 bagian tagging dalam BIO-*Format*, yaitu B (*Beginning*), I (*Inside*), O (*Outside*). *Tag Beginning* melambangkan kata pertama dalam suatu entitas, *tag Inside* melambangkan kata lanjutan dalam suatu entitas, dan *tag Outside* melambangkan kata yang tidak termasuk dalam sebuah entitas.

Pada penelitian ini, entitas yang digunakan adalah sebagai berikut:

1. Person (PER): B-PER dan I-PER
2. Location (LOC): B-LOC dan I-LOC
3. Organization (ORG): B-ORG dan I-ORG
4. Others (O): O

2.9. Part-of-Speech Tagging (POS-Tagging)

Part-of-Speech Tagging atau POS-*Tagging* merupakan suatu metode untuk mengolah data berupa kata yang tidak teranotasi ke dalam beberapa label seperti *noun*, *verb*, *adjective*, dan lain-lain [27]. POS-*Tagging* memiliki peranan penting dalam tugas-tugas NLP, seperti *machine translation*, *information retrieval*, *informaton extraction*, dan lain-lain.

Saat ini, terdapat 3 jenis teknik dalam POS-*Tagging*, yaitu *Rule-based Tagging*, *Stochastic Tagging*, dan *Transformation-based* (*hybrid*). *Rule-based Tagger* menggunakan aturan-aturan yang spesifik atau *dictionary* untuk melakukan *labelling* pada sebuah kata. *Stochastic Tagging* menggunakan pendekatan secara statistik kepada sebuah label yang paling cocok dengan kata tersebut. *Stochastic Tagging* biasanya menggunakan *Hidden Markov Model* (HMM) untuk menganalisis sebuah kata yang ambigu. *Transformation-based Tagging* adalah mengaplikasikan penggunaan aturan-aturan spesifik seperti *Rule-based Tagging* dengan pencarian label data menggunakan statistik atau probabilistik seperti yang digunakan pada *Stochastic Tagging*.