BAB 2
LANDASAN TEORI

2.1 *Coreference Resolution*

Coreference resolution merupakan salah satu penelitian dalam bidang pemrosesan bahasa alami atau NLP (*Natural Language Processing*). *Coreference resolution* bertugas untuk mencari semua ekspresi yang merujuk pada entitas yang sama dalam suatu teks. Sistem *coreference* merupakan sistem yang tipikalnya beroperasi dengan membuat urutan dari keputusan lokal. Hal ini menjadi tahapan penting sebagai kebanyakan level dari tugas NLP yang dapat membantu bahasa natural memahami setiap dokumen *summarization* (peringkasan), *question answering* (menjawab pertanyaan), dan *information extraction* (ekstraksi informasi)[6].

Coreference resolution mendeteksi kata benda yang biasanya menyatakan orang yang mana sering kali diganti kedudukannya di dalam pertuturan dengan sejenis kata yang lazim yang disebut kata ganti[3]. Untuk lebih memahami bagaimana cara kerja *coreference resolution*, perhatikan gambar berikut.

Andi₁ *hendak pergi berbelanja ke pasar. Namun,* ***dia₂*** *melupakan* ***dompetnya₃*** *yang tertinggal di rumah.*

Gambar 2. 1. Contoh Kalimat Coreference

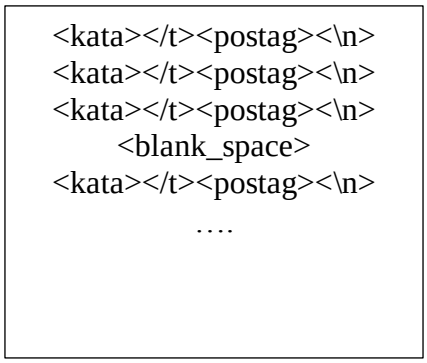
Pada contoh kalimat di atas, ketika dua kata mengalami *coreferential*, kata ganti ***dia₂*** pada kalimat kedua merupakan kata ganti yang menggantikan kedudukan kata ***Andi₁*** yang disebutkan pada kalimat pertama. Dan kata ***dompetnya₃*** yang memiliki imbuhan –nya merupakan kata ganti kepemilikan yang mana imbuhan -nya tersebut merujuk pada ***dia₂***, dan ***dia₂*** merujuk pada ***Andi₁***. Proses *coreferential resolution* di atas harus mengidentifikasi bahwa ***dompetnya₃*** merujuk pada ***dia₂*** dan ***Andi₁***, juga kata ***dia₂*** merujuk pada ***Andi₁***. Dengan kata lain sistem harus menghasilkan { ***dompetnya₃*** dan ***dia₂*** }, { ***dompetnya₃*** dan ***Andi₁***}, { ***dia₂*** dan ***Andi₁***}.

2.2 *Dataset Indonesian Manually Tagged Corpus*

Korpus *Part-of-Speech Tag* Bahasa Indonesia merupakan korpus yang berisi dokumen teks kalimat dalam bahasa Indonesia yang telah dianotasi nilai *part-of-speech tag* secara manual oleh manusia. Korpus ini dipublikasikan oleh Arawinda Dinakaramani, Fam Rashel, Andry Luthfi, dan Ruli Manurung[7].

Data kalimat-kalimat yang terdapat dalam korpus didapatkan dari PAN *Localization*. Kalimat-kalimat yang telah ditokenisasi ulang dengan memperhatikan ekspresi frasa menggunakan kamus bahasa Indonesia Kateglo. Korpus terdiri dari sepuluh ribu kalimat yang dibangun dari 256683 token.

Format yang digunakan oleh korpus adalah *tab separated value* atau yang berformat .tsv. Setiap baris terdiri dari token dengan nilai postag yang dipisahkan oleh karakter tab. Baris kosong menandakan akhir kalimat. Berikut adalah contoh corpus berformat .tsv pada gambar berikut.



Gambar 2. 2. Format Indonesian Manually Tagged Corpus

2.3 *Preprocessing*

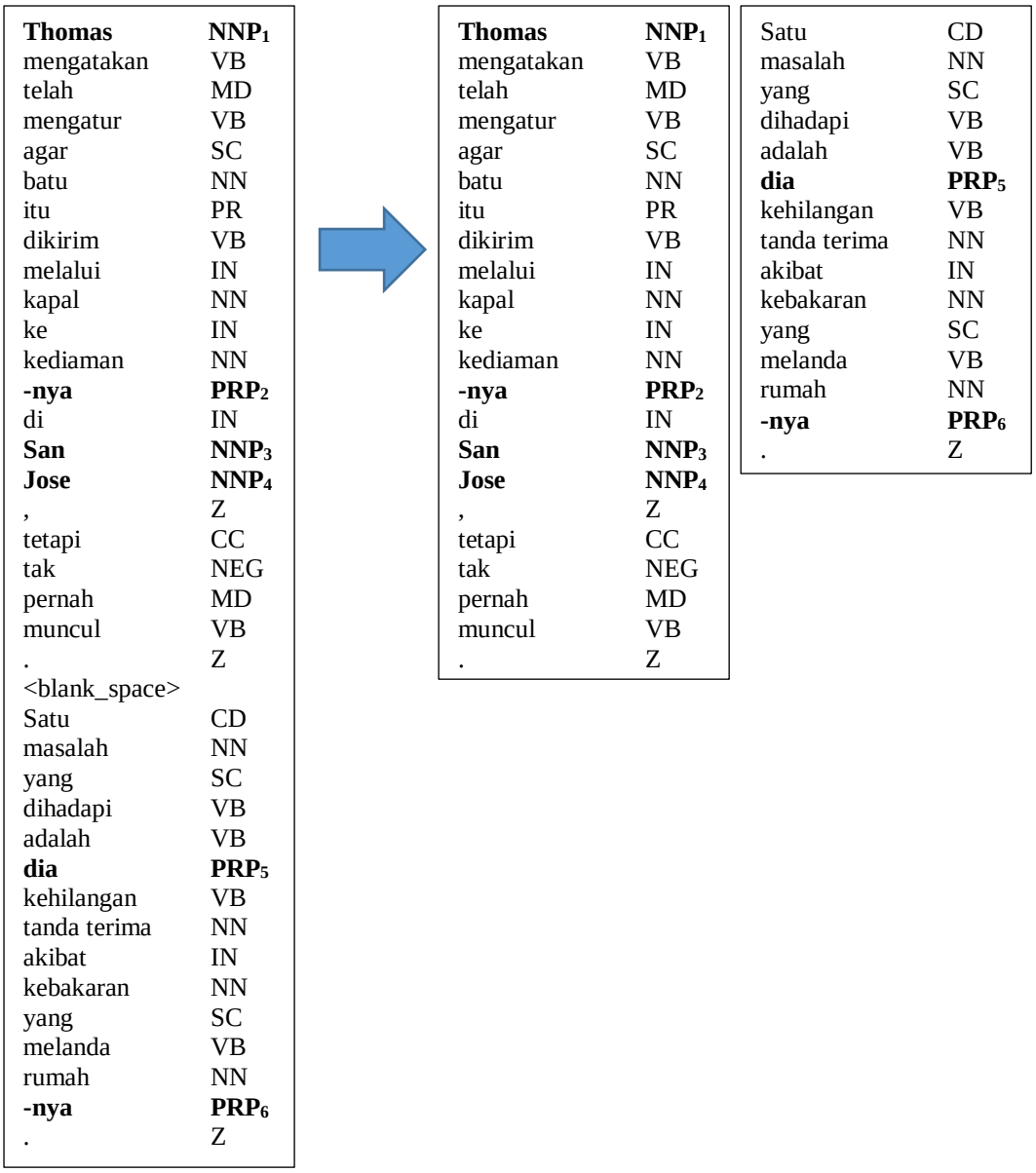
Preprocessing adalah proses mengolah data asli yang akan digunakan, kemudian diproses ke tahap selanjutnya sehingga data siap digunakan sesuai kebutuhan[8]. Proses ini sangat penting agar data yang dibutuhkan bersih dari *noise*, mengubah data yang tidak terstruktur menjadi terstruktur. *Preprocessing* yang akan digunakan pada penelitian ini terdiri dari tahapan-tahapan tersendiri diantaranya tokenisasi kata, tokenisasi kalimat, dan ekstraksi fitur.

2.3.1 Tokenisasi

Proses tokenisasi adalah proses pemotongan *string* atau masukan teks dari dokumen menjadi beberapa bagian. Tokenisasi dilakukan dengan melihat delimiternya seperti keberadaan digit, tanda baca, karakter spesial, dan sebagainya yang mana dapat menjadi suatu patokan untuk memisahkan string tersebut. Pemecahan dokumen menjadi kata-kata tunggal dilakukan dengan dua kali tahap tokenisasi yaitu tokenisasi kalimat, kemudian dilanjutkan dengan tokenisasi kata[9].

2.3.1.1 Tokenisasi kalimat

Tokenisasi kalimat merupakan proses pemotongan teks dalam suatu dokumen menjadi beberapa kalimat terpisah. Teknik ini menggunakan pemisahan kalimat dengan tanda (.) sebagai delimiter. Pada penelitian ini tokenisasi kalimat berguna untuk proses penentuan *coreference* dengan menggunakan delimiter *<blank_pace>* yang dapat dilihat pada Gambar 2. 3. Tokenisasi Kalimat



Gambar 2. 3. Tokenisasi Kalimat

2.3.1.2 Tokenisasi Kata

Tokenisasi kata merupakan proses lanjutan dari tokenisasi kalimat dimana penggalan kalimat yang sudah diproses sebelumnya, akan dilakukan pemotongan lagi menjadi kumpulan kata-kata tunggal dan dipisahkan dari postagnya dengan *tab* (\t) sebagai *delimiter*. Hasil keluaran dari proses tokenisasi kata ini akan digunakan sebagai masukan dalam tahap *casefolding*. Berikut adalah contoh proses tokenisasi kata pada Gambar 2. 4. Tokenisasi Kata.

Gambar 2. 4. Tokenisasi Kata

2.3.2 POS Tagging

Part of Speech Tagging (POS-Tagging) adalah suatu proses yang memberikan label kelas kata secara otomatis pada suatu kata dalam kalimat. Hasil dari POS ini sangat berpengaruh terhadap keluaran dari proses *parsing*[10]. Berikut adalah daftar *tagset* yang dapat diidentifikasi oleh POS *tagger* yang bersangkutan yang dapat dilihat pada Tabel 2. 1 Daftar Tagset.

Tabel 2. 1 Daftar Tagset

No.	Tag	Deskripsi	Contoh
1	CC	Konjungtor koordinatif, atau disebut juga koordinator. Konjungtor koordinatif menghubungkan dua satuan bahasa atau lebih yang sederajat (kata dengan kata, frasa dengan frasa, atau klausa dengan klausa) yang masing-masing mempunyai kedudukan yang setara dalam struktur kalimat.	dan, tetapi, atau
2	CD	Numeralia kardinal. Numeralia kardinal, yaitu numeralia yang menjadi jawaban atas pertanyaan "Berapa?", meliputi: a. bilangan pokok, misalnya dua, b. bilangan gugus, misalnya *juta*, c. bilangan penuh, misalnya *enam* dan *7916*, d. bilangan pecahan, misalnya *sepertiga*, e. bilangan desimal, misalnya *0,025* dan *0,525*, f. numeralia pokok taktentu, misalnya *banyak*, g. numeralia pokok kolektif, misalnya *kedua*, *berpuluh-puluh*, dan *ribuan*, h. tanggal,	dua, juta, enam, 7916, sepertiga, 0,025, 0,525, banyak, kedua, ribuan, 2007, 25
3	OD	Numeralia ordinal. Numeralia ordinal menyatakan urutan dan menjadi jawaban atas pertanyaan "Yang keberapa?"	ketiga, ke-4, pertama
4	DT	Artikel, atau disebut juga artikula. Artikel bertugas membatasi makna nomina.	para, sang, si
5	FW	Kata bahasa asing. Kata bahasa asing adalah kata yang berasal dari bahasa asing yang belum diserap ke dalam bahasa Indonesia. Pada dasarnya, kata bahasa asing adalah kata yang tidak terdapat di dalam kamus bahasa Indonesia.	*climate change*, *terms and conditions*

		Jika sebuah kata bahasa asing menjadi bagian dari *proper noun* atau nama, kata bahasa asing tersebut diberi *POS tag* NNP.	
6	IN	Preposisi, atau disebut juga kata depan. Preposisi menghubungkan kata atau frasa dengan konstituen di depan preposisi tersebut sehingga terbentuk frasa preposisional.	dalam, dengan, di, ke, oleh, pada, untuk
7	JJ	Adjektiva, atau disebut juga kata sifat. Adjektiva adalah kata yang memberikan keterangan yang lebih khusus tentang sesuatu yang dinyatakan oleh nomina dalam kalimat. Adjektiva meliputi: a. sifat, misalnya *bersih*, b. ukuran, misalnya *panjang* dan *kecil*, c. warna, misalnya *hitam*, d. waktu atau durasi, misalnya *lama*, e. jarak, misalnya *jauh*, f. sikap batin atau perasaan, misalnya *marah*, g. cerapan, yaitu adjektiva yang bertalian dengan pancaindra, misalnya *manis*, h. keanggotaan dalam suatu golongan, misalnya *nasional*, dan i. bentuk, misalnya *bulat*.	bersih, panjang, hitam, lama, jauh, marah, suram, nasional, bulat
8	MD	Verba modal dan verba bantu (*modal and auxiliary verb*)	boleh, harus, sudah, mesti, perlu
9	NEG	Kata ingkar.	tidak, belum, jangan
10	NN	Nomina, atau disebut juga kata benda. Nomina, yaitu kata yang mengacu pada manusia, binatang, benda, konsep, atau pengertian, meliputi: a. flora dan fauna, misalnya *monyet*, b. nomina lokatif dan nomina yang menunjukkan tempat atau arah, misalnya *bawah*, c. nomina penunjuk waktu, misalnya *sekarang*, dan d. mata uang yang tidak ditulis dalam bentuk simbol, misalnya *rupiah*.	monyet, bawah, sekarang, rupiah
11	NNP	*Proper noun.*	Boediono, Laut Jawa, Indonesia, India, Malaysia, Bank

		Proper noun adalah nama spesifik dari seseorang, sesuatu, atau sebuah tempat. *Proper noun* meliputi: a. nama diri atau individu, misalnya *Boediono*, b. nama tempat geografis, misalnya *Laut Jawa*, c. nama negara, wilayah, atau daerah pemerintahan, misalnya *Indonesia*, d. nama organisasi, institusi, atau perusahaan, misalnya *Bank Mandiri*, e. kode emiten, misalnya *BBKP*, f. nama bulan dalam setahun, misalnya *Januari*, g. nama hari dalam sepekan, misalnya *Senin*, h. nama hari raya, misalnya *Idul Fitri*, i. nama kompetisi, kejuaraan, ajang penghargaan, atau peristiwa bersejarah, misalnya *Piala Dunia*, dan j. judul karya, acara televisi, atau film, misalnya *Lord of the Rings: The Return of the King*. *Proper noun* yang ditulis dalam bahasa asing diberi *POS tag* NNP. *Proper noun* yang ditulis dalam bentuk singkatan atau inisial diberi *POS tag* NNP. Jika *proper noun* terdiri atas lebih dari satu kata atau bagian, setiap kata atau bagian dari *proper noun* tersebut diberi *POS tag* NNP.	Mandiri, BBKP, Januari, Senin, Idul Fitri, Piala Dunia, Liga Primer, Lord of the Rings: The Return of the King
12	NND	Penggolong atau nomina ukuran. Penggolong menempatkan nomina ke dalam sebuah kelompok tertentu dalam jumlah tertentu, misalnya *orang* dalam *dua orang prajurit.* Nomina ukuran merujuk pada ukuran, jarak, volume, kecepatan, berat, atau temperatur, misalnya *ton*.	orang, ton, helai, lembar
13	PR	Demonstrativa atau pronomina penunjuk.	ini, itu, sini, situ
14	PRP	Pronomina persona. Pronomina persona, yaitu pronomina yang dipakai untuk mengacu pada orang, meliputi: a. pronomina persona pertama tunggal, misalnya *saya*, b. pronomina persona pertama jamak eksklusif, misalnya *kami*,	saya, kami, kita, kamu, kalian, dia, mereka

		c. pronomina persona pertama jamak inklusif, misalnya *kita*, d. pronomina persona kedua tunggal, misalnya *kamu*, e. pronomina persona kedua jamak, misalnya *kalian*, f. pronomina persona ketiga tunggal, misalnya *dia*, dan g. pronomina persona ketiga jamak, misalnya *mereka*.	
15	RB	Adverbia, atau disebut juga kata keterangan.	sangat, hanya, justru, niscaya, segera
16	RP	Partikel. Dalam penelitian ini, *POS tag* RP menandai partikel penegas, yaitu partikel yang digunakan untuk menegaskan kalimat interogatif, imperatif, atau deklaratif.	pun, -lah, -kah
17	SC	Konjungtor subordinatif, atau disebut juga subordinator. Konjungtor subordinatif menghubungkan dua klausa atau lebih dan salah satu dari klausa-klausa tersebut merupakan klausa subordinatif.	sejak, jika, seandainya, supaya, meski, seolah-olah, sebab, maka, tanpa, dengan, bahwa, yang, lebih ... daripada ..., semoga
18	SYM	Simbol. Simbol, yang diberi *POS tag* SYM, meliputi simbol matematis, misalnya +, dan simbol mata uang, misalnya *IDR*.	IDR, +, %, @
19	UH	Interjeksi. Interjeksi mengungkapkan rasa hati atau perasaan pembicara dan secara sintaktis tidak berhubungan dengan kata-kata lain di dalam kalimat atau ujaran.	brengsek, oh, ooh, aduh, ayo, mari, hai
20	VB	Verba, atau disebut juga kata kerja. Verba, yang diberi POS tag VB, dapat berupa verba transitif, verba intransitif, verba aktif, verba pasif, dan kopula. Jika kata dasar dari verba berimbuhan berupa kata bahasa asing, verba berimbuhan tersebut tetap diberi *POS tag* VB, misalnya *di-arrange*.	merancang, mengatur, pergi, bekerja, tertidur
21	WH	Pronomina penanya, atau disebut juga kata tanya.	siapa, apa, mana, kenapa, kapan, di mana, bagaimana, berapa

		Pronomina penanya digunakan dalam kalimat interogatif sebagai pemarkah pertanyaan. Klausa tanya yang ada di dalam kalimat deklaratif merupakan klausa subordinatif sehingga pronomina penanya yang menghubungkan klausa tanya dengan klausa lain di dalam kalimat deklaratif menjadi konjungtor subordinatif dan diberi *POS tag* SC.	
22	X	Kata atau bagian dari kalimat yang tidak diketahui atau belum diketahui secara pasti kategorinya. *POS tag* X juga digunakan untuk menandai kesalahan penulisan (*typo*).	*state*men
23	Z	Tanda Baca	"...", ?, .

2.4 Ekstraksi Fitur

Ekstraksi fitur adalah sebuah proses untuk mengambil atau mengekstrak karakteristik atau informasi yang penting dari suatu data untuk digunakan dalam proses klasifikasi. Pada penelitian yang dilakukan sebelumnya yang berjudul "*Research of Noun Phrase Coreference Resolution*" menerapkan beberapa fitur untuk mendeteksi *corference resolution* diantaranya adalah *Distance, String Match, Alias, I-pronoun, J-pronoun, Demonstrative Noun Phrase, Semantic Class, I-propernoun, J-propernoun, I-arg0, I-arg1*, *J-arg0, J-arg1* dan *Similarity*. Pada penelitian ini hanya digunakan beberapa fitur saja diantaranya *i-pronoun, i-propernoun, j-pronoun, j-propernoun, Distance of Word, Distance of Senctence,* dan *String Match*. Semua fitur tersebut akan diekstrak menjadi nilai-nilai yang akan digunakan sebagai ciri untuk tahap pengklasifikasian[1]. Berikut adalah penjelasan setiap fitur yang akan digunakan pada Tabel 2. 2 Fitur *Coreference Resolution*.

Tabel 2. 2 Fitur *Coreference Resolution*

Fitur yang mendeskripsikan *coreference*-1(i)	
i-Pronoun	Mendeskripsikan nilai i yang dideteksi oleh POS tag dari entitasnya adalah PRP. Jika i adalah kata ganti nilainya adalah 1 jika tidak maka nilainya 0.
i-Proper Noun	Mendeskripsikan nilai i yang dideteksi oleh POS tag dari entitasnya adalah NNP. Jika i adalah proper noun nilainya adalah 1 jika tidak maka nilainya 0.

Fitur yang mendeskripsikan *coreference*-2(j)	
j-Pronoun	Mendeskripsikan nilai j yang dideteksi oleh POS tag dari entitasnya adalah PRP. Jika j adalah kata ganti nilainya adalah 1 jika tidak maka nilainya 0.
j-Proper Noun	Mendeskripsikan nilai j yang dideteksi oleh POS tag dari entitasnya adalah NNP. Jika j adalah proper noun nilainya adalah 1 jika tidak maka nilainya 0.
Fitur untuk relasi antar keduanya	
Distance Of Words (DOW)	Mendeskripsikan jarak baris kata atau data antara i dan j. Nilai yang akan mengisi pada fitur ini adalah letak kata *j* dikurangi kata ke *i* lalu dibagi jumlah baris kata pada dataset.
Distance Of Sentences (DOS)	Mendeskripsikan jarak kalimat antara kata i dan j. Nilai yang akan mengisi pada fitur ini adalah letak kalimat pada kata *j* dikurangi letak kalimat pada kata ke *i* lalu dibagi jumlah kalimat pada dataset.
String_match	Mendeskripsikan string pada i dan j. Jika string i sesuai dengan string j, maka nilainya adalah 1 jika tidak nilainya 0.

Berikut adalah contoh data masukan yang akan digunakan pada Gambar 2.

5. Contoh Data Masukan.

Thomas	**NNP_1**
mengatakan	VB
telah	MD
mengatur	VB
agar	SC
batu	NN
itu	PR
dikirim	VB
melalui	IN
kapal	NN
ke	IN
kediaman	NN
-nya	**PRP_2**
di	IN
San	**NNP_3**
Jose	**NNP_4**
,	Z
tetapi	CC
tak	NEG
pernah	MD
muncul	VB
.	Z

Satu	CD
masalah	NN
yang	SC
dihadapi	VB
adalah	VB
dia	**PRP_5**
kehilangan	VB
tanda terima	NN
akibat	IN
kebakaran	NN
yang	SC
melanda	VB
rumah	NN
-nya	**PRP_6**
.	Z

Gambar 2. 5. Contoh Data Masukan

Pada ekstraksi fitur data *training* memanfaatkan label.txt untuk membuat pasangan *i* dan *j*. Jika kata *i* memiliki data label yang berbeda dengan kata *j* maka akan dijadikan pasangan. Contohnya adalah kata ***Thomas*** memiliki label ***1;1;1*** dan ***-nya*** memiliki label ***13;1;1*** karena angka-angka pada label yang dimiliki ***Thomas*** dan ***-nya*** berbeda maka ***Thomas*** dan ***-nya*** akan dijadikan pasangan *i* dan *j*. Dimana *i* adalah ***nya*** dan *j* adalah ***Thomas*** karena letak *j* harus berada setelah kata *i* ($i > j$) . Setelah tahap *Preprocessing* maka didapatkan kata dan tag. Berdasarkan label yang dimiliki maka sistem akan mencari kata yang dimaksud. Kata ganti ***-nya*** sebagai *i* memiliki tag PRP sehingga fitur i-pronoun diberi nilai 1 dan i-propernoun diberi nilai 0. Begitu pun dengan ***Thomas*** sebagai *j, k*arena ***Thomas*** memiliki tag NNP sehingga fitur j-pronoun diberi nilai 0 dan j-propernoun diberi nilai 1. Untuk mencari nilai *Distance Of Words* seperti pada rumus (2.1). Dengan jarak kata *i* didapat dari jarak kata *i* dari awal kalimat, begitupun dengan kata *j*.

$$DOW = \frac{jarak\ kata\ i - jarak\ kata\ j}{jumlah\ kata\ dalam\ dataset} \qquad (2.1)$$

Sedangkan untuk mencari nilai *Distance Of Sentences* seperti pada rumus (2.2). Dengan jarak kalimat kata *i* didapat dari jarak kalimat kata *i* dari awal dataset, begitupun dengan kalimat kata *j*.

$$DOS = \frac{jarak\ kalimat\ kata\ i - jarak\ kalimat\ kata\ j}{jumlah\ kalimat\ dalam\ dataset} \qquad (2.2)$$

2.5 *Recurrent neural network*

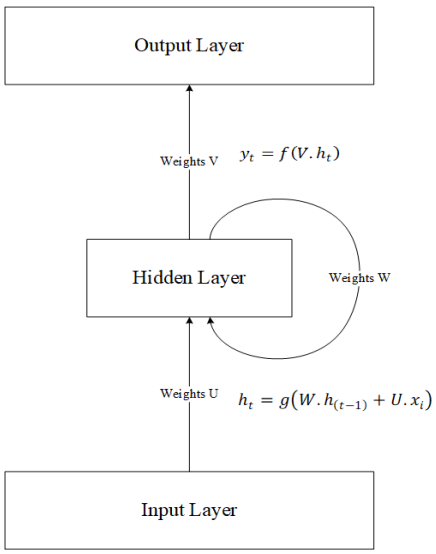
Recurrent neural network (RNN) merupakan kategori *deep learning* karena mempunya banyak lapisan atau layer yaitu hidden layer dan dapat diaplikasikan pada penelitian NLP dan *text mining*[11]. RNN pada penelitian ini menggunakan arsitektur *Elman Recurrent neural network* (ERNN) yang masih termasuk ke dalam *Simple Recurrent neural network* (SRNN).

Recurrent neural network (RNN) adalah jaringan saraf tiruan yang berulang, yang memungkinkan informasi bertahan dalam jaringan. RNN memiliki keterkaitan umpan balik ke jaringan itu sendiri yang memungkinkan aktivasi mengalir kembali dalam satu lingkaran mempelajari urutan dan informasi sebelumnya[12]. RNN sangat *powerful* dalam memodelkan data sekuensial, ucapan atau teks dan diterapkan pada data non-sekuensial untuk melatih secara non-

sekuensial, RNN bisa digunakan dalam kasus *image, video captioning, word prediction, word translation, image processing, speech recognition, natural language processing, music processing* dan sebagainya.

2.5.1 Elman *Recurrent neural network*

Elman Recurrent Neural Netwrok (ERNN) merupakan salah satu dari arsitektur *Recurrent neural network* (RNN), *Elman Recurrent neural network* (ERNN) termasuk ke dalam *Simple Recurrent neural network* (SRNN)[12]. ERNN melakukan proses *learning* dengan membuat salinan *neuron layer hidden* pada *layer input* disebut sebagai *context input* (*hidden layer* sebelumnya), sehingga salinan ini berfungsi sebagai perpanjangan dari *input* layer. Fungsi dari *context input* ini adalah untuk menyimpan status ataupun keadaaan sebelumnya dari *layer hidden*, untuk kemudian disampaikan kembali kepada layer *hidden*. Hubungan antara *context input* dan *layer hidden* adalah terhubung penuh (*fully connected*) dan diberi bobot 1. Selain ERNN arsitektur yang termasuk *Simple Recurrent neural network* adalah *Jordan Recurrent neural network*, perbedaannya adalah model Jordan melakukan proses learning dengan membuat salinan layer *output* pada layer *input* yang disebut sebagai *state layer*. Dengan proses ini, hasil *output* pada iterasi sebelumnya akan menjadi bagian dari *input* pada iterasi selanjutnya.



Gambar 2.6 Arsitektur Elman Recurrent Neural Network

Pada Gambar 2.6 Elman *Recurrent neural network* memerlukan 3 parameter bobot diantarannya, V bobot dari *input* ke *hidden layer*, U bobot dari *context layer*

(*hidden layer* sebelumnya) ke *hidden layer*, dan W bobot dari *hidden layer* ke *output layer*. Pada proses pelatihan ketiga bobot tadi akan diperbaharui hingga mendapatkan bobot yang optimal. Dalam melakukan *feedforward* ERNN menggunakan persamaan sebagai pada (2.3).

$$h_t = g\left(W.h_{(t-1)} + U.x_i\right) \quad (2.3)$$

Keterangan :

h_t = *hidden layer* pada *timestep* ke-t

W = bobot dari *neuron hidden* sebelumnya menuju ke *neuron hidden* selanjutnya

x_i = nilai *neuron input* ke-i

U = bobot dari *neuron input* menuju ke *neuron hidden*

t = *timestep*

g = *activation function* untuk mendapat nilai keluaran pada *hidden layer*, pada penelitian ini *activation function* yang digunakan pada *hidden layer* adalah *Tanh*

Sedangkan nilai *output layer* waktu ke t didapatkan dengan persamaan sebagai berikut.

$$y_t = f(V.h_t) \quad (2.4)$$

Dengan ketentuan :

y_t = *output layer* pada waktu ke t

V = bobot dari *neuron hidden* menuju ke *neuron output*

h_t = nilai *neuron hidden* ke-t

f = *activation function* untuk mendapat nilai keluaran pada *output layer*, pada penelitian ini *activation function* yang digunakan pada *output layer* adalah *Sigmoid.*

2.5.2 *Activation Function* (Fungsi Aktivasi)

Fungsi aktivasi sangat umum digunakan dalam *neural network*. Alasan utama menggunakan fungsi aktivasi adalah agar *neural network* mengenali data yang non-linear, karena *output* yang dihasilkan dari *neural network* jarang sekali bersifat linear[13]. Fungsi aktivasi juga digunakan untuk mengaktifkan dan menonaktifkan *neuron*, karakteristik *neural network* ditentukan oleh bobot dan *input-output* fungsi aktivasi yang diterapkan. Terdapat banyak jenis fungsi aktivasi

pada *neural network*, namun yang di gunakan dalam penelitian ini fungsi aktivasi *hyperbolic tangent* dan *sigmoid*.

2.5.2.1 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[14]. 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 berikut.

$$tanh(x) = \frac{\sin(x)}{\cos(x)} = \left(\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\right) = \left(\frac{e^{2x}-1}{e^{2x}+1}\right) \qquad (2.5)$$

Adapun untuk turunan aktivasi fungsi *tanh* yang nantinya akan digunakan pada proses BPTT dalam algoritma ERNN[11] seperti pada rumus (2.6).

$$derivative(tanh) = (1 - o^{2}) \qquad (2.6)$$

Pada rumus (2.6) terdapat simbol o dimana o adalah *output* dari *hidden layer*. Pada penelitian ini fungsi aktifasi *tanh* digunakan untuk menghitung nilai keluaran pada *hidden layer* untuk waktu t.

2.5.2.2 *Sigmoid*

Fungsi aktivasi *sigmoid* digunakan untuk menghitung probabilitas pada hasil *output* , yang terjadi di *output* layer dimana akan diambil nilai probabilitas paling besar sebagai prediksi. *Sigmoid* digunakan untuk menghitung probabilitas distribusi dari vektor bilangan real. Fungsi *sigmoid* menghasilkan *output* yang merupakan kisaran nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1[13]. Persamaan dari fungsi *sigmoid* dapat dilihat pada persamaan (2.7).

$$sigmoid(xi) = \left(\frac{1}{1+e^{-xi}}\right) \qquad (2.7)$$

Adapun turunan dari *sigmoid* yang nantinya akan digunakan pada proses BPTT algoritma ERNN[15] dapat dilihat pada persamaan (2.8).

$$derivative(sigmoid) = (y_t - label_t) \otimes s_t \qquad (2.8)$$

Dimana y_t adalah prediksi label ERNN ke t, $label_t$ adalah label sebenarnya (*true* label) ke t dan s_t *hidden layer* ke t. Pada penelitian ini fungsi aktivasi *sigmoid* digunakan untuk menghitung nilai keluaran pada *output* layer waktu t.

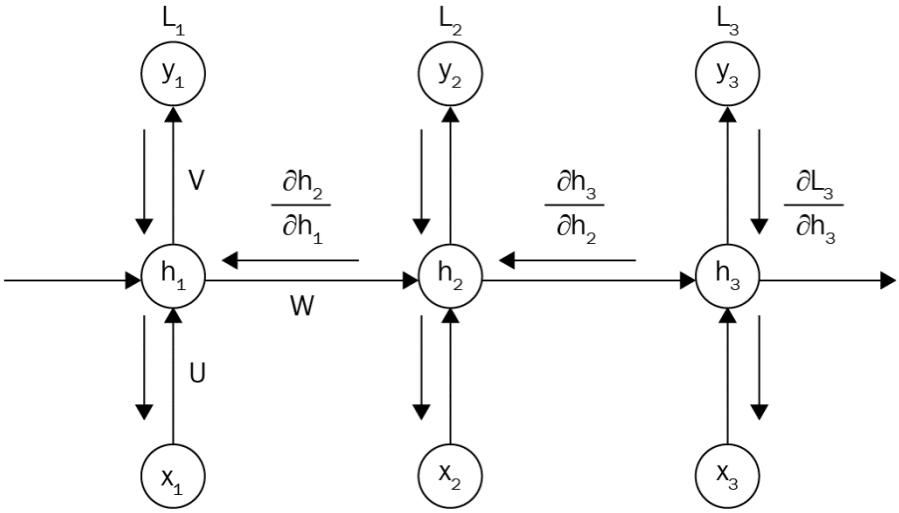
2.5.3 *Loss Function*

Loss function bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai yang sebenarnya[16]. Ada beberapa jenis *loss function* diantaranya *Mean Square Error* (MSE) dan *Cross Entropy* (CE). Namun dalam praktiknya, CE lebih cepat dalam konvergensi dan mendapatkan hasil lebih baik dalam tingkat kesalahan klasifikasi[16]. *Cross entropy* biasa digunakan pada *neural network* dan dalam kasus *multi class classification*. Pada penelitian ini *loss function* yang digunakan untuk menghitung nilai *error* hasil prediksi menggunakan *Cross Entropy* (CE). Persamaan *Cross Entropy* dapat dilihat pada persamaan (2.9).

$$L_{CE}(\hat{y}, y) = -\sum_{i=1} y_i log \hat{y}_i \qquad (2.9)$$

2.5.4 *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*[14]. 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 ada 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.



Gambar 2. 7. Backpropagation Through Time

Pada ERNN dalam melakukan BPTT akan menghitung turunan bobot U, W, dan V. Adapun rumus penurunan dari setiap bobot adalah sebagai berikut.

1. Turunan bobot V

Dalam menghitung turunan bobot V akan menggunakan rumus pada persamaan (2.10).

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

$$\frac{\partial E_t}{\partial V} = (y_t - label_t) \otimes h_t \qquad (2.10)$$

Keterangan :

$\frac{\partial E_t}{\partial V}$ = 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

$label_t$ = vektor dari label sebenarnya ke t

t = *time* atau urutan token

2. Turunan bobot W

Dalam menghitung turunan bobot W dimana setiap *hidden state* h_t terhubung dengan *hidden state* sebelumnya $h_{(t-1)}$ dan begitu seterusnya hingga mencapai *hidden state* pertama h_1 yang terhubung dengan h_0 . Maka dalam

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

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

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-1)}}\frac{\partial h_{(t-2)}}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-2)}}\frac{\partial h_{(t-2)}}{\partial W}\right) + \cdots + \left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial W}\right)$$

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W} = \delta_t \otimes h_{t-1} \qquad (2.11)$$

Keterangan :

$\frac{\partial E_t}{\partial W}$ = 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 layer* ke t

t = *time* atau urutan token

Dimana δ_t didapat dari penurunan. Rumus penurunan δ_t seperti pada rumus (2.12).

$$\delta_t = [V^T.(y_t - label_t)] \times \left(1 - {h_t}^2\right) \qquad (2.12)$$

Keterangan:

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

y_t = Nilai dari *output* layer ke t

$label_t$ = vektor dari label sebenarnya ke t

h_t= *Hidden state* ke t

t = *time* atau urutan token

3. Turunan bobot U

Pada penurunan bobot U mempunyai kesamaan dengan penurunan bobot W. Letak perbedaannya jika bobot W melakukan penurunan terhadap h_t sedangkan bobot U melakukan penurunan terhadap x_t. Rumus turunan bobot W ditunjukan persamaan (2.13).

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

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

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U} = \delta_t \otimes x_t \qquad (2.13)$$

Keterangan :

$\frac{\partial E_t}{\partial U}$ = 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 = *time* atau urutan token

2.5.5 *Stochastic Gradient Descent* (SGD)

Stochastic Gradient Descent adalah salah satu metode dalam dengan meminimalkan nilai *error* untuk mengukur keakuratan jaringan dalam melakukan tugas yang diberikan[17]. SGD bertujuan untuk mengatasi nilai *error* (*loss*) yang tinggi dengan cara memperbarui parameter. Disebut "*stochastic*" karena pada prosesnya sampel dipilih secara acak sebagai kelompok tunggal (seperti dalam penurunan *gradien* standar) atau dalam urutan, sampel yang dipilih diambil dari dataset pelatihan. Adapun rumus dari SGD ditunjukan persamaan (2.14).

$$\theta_{(baru)} = \theta_{(lama)} - a \times \frac{\partial E}{\partial \theta} \qquad (2.14)$$

Keterangan :

$\theta_{(baru)}$ = Nilai bobot yang baru

$\theta_{(lama)}$ = Nilai bobot yang lama

a = *learning rate*

$\frac{\partial E}{\partial \theta}$ = Nilai turunan antara nilai *error* dan bobot

2.6 Akurasi

Penelitian ini menggunakan perhitungan akurasi untuk mengetahui tingkat akurasi dari klasifikasi *coreference resolution*[14]. Untuk menghitung akurasi dari RNN dilakukan perhitungan dengan rumus berikut.

$$Akurasi = \frac{Jumlah\ yang\ diklasifikasi\ dengan\ benar}{jumlah\ sampel\ data\ uji}\ x\ 100\% \qquad (13)$$

2.7 Black Box Testing

Black Box adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian *white-box*). Pengetahuan khusus dari kode aplikasi atau struktur internal

dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih *input* yang valid dan tidak valid dan menentukan *output* yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga.

Pengujian pada *black box* berusaha menemukan kesalahan seperti:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

2.8 Pemodelan Sistem

Pemodelan sistem merupakan proses pengembangan model abstrak dari sistem, dengan tiap model memperlihatkan pandangan berbeda dari sistem. Adapun yang digunakan dalam pemodelan sistem pada penelitian tugas akhir ini adalah sebagai berikut.

2.8.1 Flowchart

Flowchart adalah penggambaran secara grafik dari langka-langkah dan urut-ututan prosedur dari suatu program. *Flowchart* menolong *analyst* dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang leboh kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

2.8.2 Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem. Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Diagram konteks merupakan level 0 atau level tertinggi dari DFD dimana diagram konteks akan menggambarkan seluruh *input*, proses dan *output* pada sistem[19].

2.8.3 Data Flow Diagram

DFD merupakan tampilan *input*, proses, dan *output* dari suatu sistem dimana objek data akan mengalir ke dalam sistem melalui *input* sistem dan ditransformasikan oleh elemen pemrosesan sistem kemudian data yang dihasilkan akan keluar dari sistem. Objek data diwakili oleh panah berlabel dan tranformasi diwaliki oleh lingkaran. DFD digambarkan secara hierarki dimana model aliran data pertama (kadang disebut DFD level 0) mewakili sistem secara keseluruhan. Model aliran data selanjutnya memberikan masing-masing detail yang semakin meningkat[19].

2.9 Bahasa Pemrograman

Bahasa pemrograman adalah *software* bahasa komputer yang digunakan dengan cara merancang atau membuat program sesuai dengan struktur dan metode yang dimiliki oleh bahasa program itu sendiri. Komputer mengerjakan transformasi data berdasarkan kumpulan perintah program yang telah dibuat oleh program. Kumpulan perintah ini harus dimengerti oleh komputer, berstruktur tertentu (*syntax*), dan bermakna. Bahasa pemrograman merupakan notasi untuk memberikan secara tepat program komputer.

2.9.1 *Python*

Pada saat ini bahasa pemrograman *python* sering digunakan untuk melakukan penelitian atau pembuatan aplikasi machine learning dikarenakan banyak *library* yang mendukung dan bahasa pemrograman *python* sama seperti matematika[19]. Kelebihan dari bahasa pemrograman *python* yaitu banyaknya *library* yang sedang dikembangkan, mudah untuk dipelajari dan dapat diintegrasikan dengan bahasa pemrograman lain seperti C, C++ atau java.

Walaupun begitu bahasa pemrograman *python* memiliki beberapa kelemahan yaitu lemah dalam bidang teknologi *mobile*, lemah dalam pengembangan *database layer* dan *run-time error* (*error* hanya terlihat pada saat program sudah dijalankan).

Python adalah bahasa pemrograman yang bersifat *open source*. Bahasa pemrograman ini dioptimalisasikan untuk *software quality*, *developer productivity*, *program portability*, dan *component integration*. *Python* telah digunakan untuk mengembangkan berbagai macam perangkat lunak, seperti *internet scripting*, *systems programming*, *user interfaces*, *product customization*, *numberic programming*, dan lain-lain.

2.9.2 Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman web. Di dalamnya berisi kumpulan informasi yang disimpan dalam *tag* tertentu, dimana *tag* tersebut digunakan untuk melakukan format terhadap informasi yang dimaksud. Berbagai pengembangan telah dilakukan terhadap kode HTML dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman web[21]. Kendati demikian, sampai sekarang HTML tetap berdiri kokoh sebagai dasar dari bahasa web seperti PHP, ASP, JSP dan lainnya. Bahkan secara umum, mayoritas situs web yang ada di Internet pun masih tetap menggunakan HTML sebagai teknologi utama mereka.


UNIVERSITAS KOMPUTER INDONESIA
UNIKOM