BAB 2
TINJAUAN PUSTAKA

2.1. Ekstraksi Informasi

Ekstraksi informasi adalah suatu cara untuk mengambil informasi yang berasal dari teks. Teks biasanya ditulis secara bebas, tidak terstruktur, atau dalam bahasa natural. Hasil dari ekstraksi informasi berupa teks yang terstruktur [6]. Hasil ekstraksi informasi digunakan sebagai informasi yang berguna. Sistem ekstraksi informasi menggunakan teknik pencocokan pola. Setiap pola terdiri dari suatu ekspresi regular dan pemetaan dari bentuk sintaktis ke logis [7] [8] [9]. Fungsi ekstraksi informasi digunakan untuk mencari kata (token) dan entitas yang dapat mewakili isi dari data tekstual. Pengenalan entitas bernama merupakan komponen dasar dari ekstrasi informasi yang bertugas untuk mengenali entitas tersebut. Entitas yang telah dikenali digunakan sebagai metadata pada ekstraksi informasi. Terdapat beberapa metode yang dapat digunakan dalam ekstraksi informasi, di antaranya yaitu [9] :

1. *Hand-coded* atau *Learning-based*

Pendekatan ini membutuhkan ahli untuk mendefinisian aturan (*rules*) atau *regular expressions* yang digunakan untuk mengekstrak informasi yang relevan. Pengetahuan umum atau intuisi juga diterapkan dalam perancangan aturan. Pendekatan ini menggunakan proses iteratif. Ketika proses iterasi sedang berlangsung beberapa aturan dapat dimodifikasi untuk meningkatkan kualitas dari sistem. Hasil sistem ekstraksi informasi tergantung pada pengetahuan pengembang atau pakar yang dilibatkan untuk mendefinisikan aturan. Teks masukan untuk sistem ini adalah teks.

2. *Rule-based* atau *Statistical*

Pendekatan ini menggunakan korpus yang telah dianotasi terlebih dahulu. Tidak diperlukan pakar pada saat sistem berjalan. Ahli hanya diperlukan ketika perancangan sistem dan proses menganotasi teks. Teks yang telah dianotasi digunakan sebagai input sistem untuk menjalankan algoritma pelatihan. Sistem

mendapatkan pengetahuan dari teks yang telah dianotasi dan menggunakannya untuk memperoleh informasi yang dibutuhkan dari teks-teks yang dimasukkan.

2.2. Pengenalan Entitas Bernama

Pengenalan entitas bernama adalah komponen dari ekstraksi informasi yang berfungsi untuk mengenali entitas nama (nama orang, lokasi, organisasi), ekspresi waktu (tanggal, waktu, durasi), dan ekspresi bilangan (uang, persen, numerik, kardinal) [10]. Entitas nama terdiri dari nama orang termasuk nama keluarga, nama lokasi seperti lokasi geografis, dan nama organisasi meliputi nama perusahaan, nama organisasi lain, dsb.

2.3. Pendekatan Pengenalan Entitas Bernama

Pengenalan entitas dapat dilakukan menggunakan tiga jenis pendekatan yakni *rule based*, *machine learning*, dan *hybrid*.

1. ***Rule based***

Pendekatan *rule-based* merupakan cara pengenalan entitas bernama yang telah lama digunakan. Pendekatan ini menggunakan aturan atau *rule* yang didefinisikan manual oleh ahli bahasa berdasarkan pengamatan untuk mengenali entitas. Aturan yang digunakan seperti kelas kata atau *part-of-speech*, sintaksis penulisan, dan fitur ortografi [11]. Proses pengenalan entitas dengan cara ini memiliki ketergantungan yang besar terhadap bahasa yang digunakan dan keterbatasan para ahli dalam mendefinisikan semua aturan untuk mengenali entitas bernama.

2. ***Machine learning***

Metode *machine learning* dalam pengenalan entitas bernama digunakan untuk melakukan klasifikasi menggunakan model klasifikasi statistik. Pada pendekatan ini dibuat *pattern* atau pola dan model menggunakan pendekatan statistik. Pengenalan entitas bernama menggunakan *machine learning* merupakan masalah klasifikasi pada umumnya sehingga terdapat tiga pendekatan dalam *machine learning*, yaitu *supervised*, *semi-supervised*, dan *unsupervised*.

Supervised learning adalah pendekatan pembelajaran menggunakan data yang telah dianotasi. Model yang dibuat akan menghasilkan performa yang bagus jika pada pelatihan menggunakan data masukkan yang berkualitas dan dalam jumlah data yang besar. Pada pendekatan *unsupervised learning* pembelajaran menggunakan data yang tidak dianotasi sehingga diharapkan dapat menghasilkan dan membangun representasi dari data. Representasi data digunakan untuk kompresi data, klasifikasi, pengambilan keputusan dan beberapa tujuan lain. Implementasi *unsupervised* biasanya digabungkan dengan metode-metode lain.

3. ***Hybrid***

Hybrid merupakan metode penggabungan dari *rule based* dan *machine leaning*. Pengenalan entitas dengan cara ini memaksimalkan kelebihan-kelebihan masing-masing metode *rule based* dan *machine learning*.

2.4. Pengenalan Kelas Kata

Pengenalan kelas kata atau *part of speech (POS) Tagging* adalah proses penentuan jenis kata berdasarkan tata bahasa. Setiap tag pengenalan kelas kata terdiri dari kategori dari sebuah kata seperti kata benda umum, kata benda, kata sifat, dan sebagainya dan juga terdapat informasi morfologi yaitu nomor, jenis kelamin, orang, dan sebagainya.

Pengenalan kelas kata membantu proses sintaksis atau semantik *parser* karena tidak melakukan kalimat ambigu. Hal ini dapat dimasukkan dalam sistem *Natural Language Processing* (NLP) yang berkaitan dengan teks terbatas, seperti ekstraksi informasi, pencarian informasi, peringkasan, terjemahan mesin, dan lain-lain. Hampir semua area NLP membutuhkan pengenalan kelas kata untuk meningkatkan performa dalam akurasi dan efisiensi komputasi.

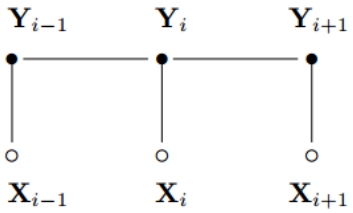
POS Tag yang digunakan adalah IPOSTagger hasil penelitian Alfan Wicaksono menggunakan metode *Hidden Markov Model* karena memberikan akurasi yang baik yaitu sebesar 96,50% [12]. Tag kelas kata *START* dan *END* ditambahkan untuk menandakan awal dan akhir kalimat. Kedua kelas kata tersebut tidak berada pada sebuah kata. Tag kelas kata yang digunakan beserta contohnya dapat dilihat pada Tabel 2.1 berikut ini [12].

Tabel 2.1 Tag Kelas Kata

No.	Tag Kelas Kata	Kelas Kata	Contoh
1.	OP	Buka kurung	({[
2.	CP	Tutup kurung	)}]
3.	GM	Garis miring	/
4.	;	Titik koma	;
5.	:	Titik dua	:
6.	“	Tanda kutip	“ ’
7.	.	Pengakhir kalimat	. ! ?
8.	,	Koma	,
9.	-	Garis	-
10.	…	Ellipsis	…
11.	JJ	Adjektiva	Kaya, Manis
12.	RB	Adverbia	Sementara, Nanti
13.	NN	Common Noun	Mobil
14.	NNP	Proper Noun	Bekasi, Indonesia
15.	NNG	Genitive Noun	Bukunya
16.	VBI	Intransitive Verb	Pergi
17.	VBT	Transitive Verb	Membeli
18.	IN	Preposisi	Di, Ke, Dari
19.	MD	Modal	Bisa
20.	CC	Coor-Conjunction	Dan, Atau, Tetapi
21.	SC	Subor-Conjunction	Jika, Ketika
22.	DT	Determiner	Para, Ini, Itu
23.	UH	Interjection	Wah, Aduh, Oi
24.	CDO	Ordinal Numerals	Pertama, Kedua
25.	CDC	Collective Numerals	Bertiga
26.	CDP	Primary Numerals	Satu, Dua
27.	CDI	Irregular Numerals	Beberapa
28.	PRP	Personal Pronouns	Saya, Kamu
29.	WP	WH-Pronouns	Apa, Siapa
30.	PRN	Number Pronouns	Kedua-duanya
31.	PRL	Locative Pronouns	Sini, Situ, Sana
32.	NEG	Negasi	Bukan, Tidak
33.	SYM	Simbol	@#$%^&
34.	RP	Partikel	Pun, Kah
35.	FW	Kata asing	Foreign, Word
36.	START	Penanda awal kalimat	
38	END	Penanda akhir kalimat	

2.5. Conditional Random Fields

Conditional random fields merupakan suatu model probabilistik untuk segmentasi dan pelabelan suatu sekuen data [2]. CRF berbentuk model grafik tidak berarah yang mendefinisikan sebuah distribusi *log-linear* tunggal dari label berurutan terhadap urutan data observasi tertentu [13]. CRF memiliki banyak kelebihan dibandingkan model probabilitas lain seperti *Hidden Markov Model* (HMM) dan *Maximum Entropy Markov Model* (MEMM). CRF mengatasi permasalahan ketergantungan asumsi yang tinggi pada HMM. Hal ini karena CRF dapat menentukan sendiri seberapa banyak fitur yang diinginkan untuk membangun sebuah model CRF tidak seperti HMM yang bersifat lokal dimana setiap kata hanya bergantung pada label saat ini dan setiap label sebelumnya. Selain itu CRF dapat mempunyai bobot yang bebas sedangkan HMM harus memenuhi bobot tertentu. CRF juga mengatasi permasalahan label bias pada MEMM karena CRF memformulasikan distribusi kondisional label untuk sekuens data secara kesuluruhan dibandingkan MEMM yang memformulasikan distribusi kondisional label untuk setiap elemen data [14]. CRF digunakan pada banyak aplikasi termasuk *natural language processing*, *computer vision*, dan *bioinformatic*. Salah satu bentuk dari CRF adalah linier-chain CRF yang ditunjukkan oleh Gambar 2.1.



Gambar 2.1 Linear Chain Conditional Random Field

Diberikan dua peubah acak dimana $\bar{x} = (x_1, x_2, \ldots, x_n)$ adalah himpunan data observasi dan $\bar{y} = (y_1, y_2, \ldots, y_n)$ adalah himpunan label yang mungkin. Probabilitas kondisional dari *y* terhadap *x* dapat dituliskan dalam persamaan 2.1.

$$\mathrm{p(y|x)} = \frac{1}{Z(x)} \prod_c \psi_c (y_c, x) \qquad (2.1)$$

Dimana ψ_c($y_{c,}$,x) adalah fungsi potensial dan z(x) adalah fungsi normalisasi dari distribusi probabilitas kondisional label untuk semua sekuens data *x* dan dinotasikan dengan persamaan 2.2.

$$Z(x) = \sum_y \prod_c \psi_c (y_c, x) \quad (2.2)$$

Fungsi potensial adalah jumlah yang mengikat label data dengan fitur pada saat yang sama. Fitur juga sebagai *data pattern* pada beberapa literatur. Dalam kasus sederhana fungsi potensial merupakan eksponensial dari jumlah bobot semua fungsi fitur. Ada dua jenis potensial yakni *node* potensial dan *edge* potensial.

a. Node Potensial

Fungsi node potensial dinotasikan dalam persamaan 2.3.

$$\phi_t (y_t, x) = exp\left(\sum_k \lambda_k f_k (y_t, x, t) \right) \quad (2.3)$$

Dimana :

t : bernilai 1 sampai banyaknya kata dalam 1 kalimat.

y_t : label (*other*, *person*, *location*, atau *organization*)

x : kata dalam kalimat

λ_k : parameter fungsi fitur, diinisialisasi 0 untuk kalimat ke-1

$f_k (y_t, x, t)$: fungsi fitur node

k : bernilai 1 sampai banyaknya fungsi fitur node

b. Edge Potensial

Fungsi edge potensial dinotasikan dalam persamaan 2.4.

$$\psi_t (y_t, y_{t+1}, x) = exp\left(\sum_k \lambda_k f_k (y_t, y_{t+1}, x, t) \right) \quad (2.4)$$

Dimana :

y_t : label (*other*, *person*, *location*, atau *organization*)

y_{t+1} : label yang dimiliki kata ke-t+1

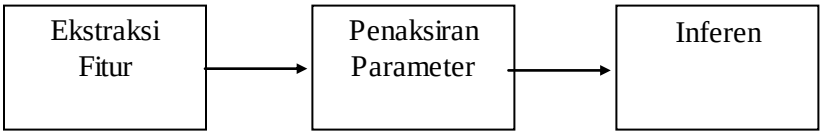
x : kata pada data pelatihan

t : bernilai 1 sampai banyaknya kata dalam kalimat

λ_k : parameter fungsi fitur edge

$f_k(y_t, y_{t+1}, x, t)$: fungsi fitur edge

Secara umum proses CRF ditunjukkan pada Gambar 2.2 berikut ini.



Gambar 2. 2 Tahapan Proses CRF

2.6. Ekstraksi Fitur

Fitur adalah sebuah elemen penting untuk membuat model. Fitur yang tepat dapat meningkatkan akurasi sistem yang dibuat. Pada implementasinya, fungsi fitur adalah sebuah fungsi yang mengkombinasikan sebuah fitur dengan labelnya yang bersesuaian dan antara label yang berdekatan. Adapun contoh fungsi fitur dapat dilihat pada persamaan 2.5.

$$f_{1000}(y_{10}, x, 10) = \begin{cases} 1, \ if\ x_{10} = "today"\ and\ y_{10} = B-NP \\ 0, \ otherwise \end{cases} \quad (2.5)$$

Dimana, fungsi fitur ke-1000 akan menghasilkan nilai 1 bila kata ke-10 adalah “today” dan label saat ini adalah B-NP dan akan bernilai 0 untuk label lain.

2.7. Penaksiran Parameter

Penaksiran parameter adalah proses untuk mendapatkan nilai optimal parameter fungsi fitur yakni λ_k. Nilai λ_k dihitung mengunakan prosedur maksimum *likelihood*. Maksimum *likelihood* merupakan kuantitas yang menunjukkan berapa banyak parameter yang didukung oleh data pelatihan. Maksimum *likelihood* dapat memaksimalkan kemiripan ciri dari kumpulan data pelatihan yang telah dimodelkan dengan CRF dengan kumpulan data pelatihan. Metode ini membutuhkan dua perhitungan yaitu *log-likelihood* dan turunan pertamanya [14]. *Log-likelihood* secara matematis dapat dituliskan dalam persamaan 2.6 berikut.

$$\mathcal{L} = \sum_{t\epsilon[1,T]} \sum_k \lambda_k f_k(y'_t, x, t) - \sum_{t\epsilon[1,T-1]} \sum_k \lambda_k f_k(y'_t, y'_{t+1} x, t) - \log Z(x) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \quad (2.6)$$

Dimana :

$\mathcal{L}$: log-likelihood

t : banyaknya kata dalam sebuah kalimat

k : banyaknya fitur bernilai 1 sampai banyaknya fitur

y'_t : label ke-t dari sebuah kalimat dari data pelatihan

y'_{t+1} : label ke-t+1 dari sebuah kalimat dari data pelatihan

x : kata

λ_k, : parameter fungsi fitur

$f_k(y'_t, x, t)$: fungsi fitur ke-k dari data pelatihan yang telah diberi label

$f_k(y'_t, y'_{t+1} x, t)$: fungsi fitur ke-k dari data pelatihan yang telah diberi label

$Z(x)$: fungsi normalisasi

σ : standar deviasi distribusi Gauss

Dari persamaan 2.6 maka didapatkan turunan dari *log-likelihood feature function* ke-*k* yaitu pada persamaan 2.7 dan 2.8

$$G_k^x = \sum_{t\epsilon[1,T]}(f_k(y_t', x, t) - \sum_{y_t} P_t(y_t|x) f_k(y_t, x, t)) - \frac{\lambda_k}{\sigma^2} \quad (2.7)$$

$$G_{k'}^x = \sum_{t\epsilon[1,T-1]}(f_k(y'_t, y'_{t+1} x, t) - \sum_{y_t}\sum_{y_{t+1}} P_t(y_t, y_{t+1}|x) f_k(y_t, y_{t+1} x, t) - \frac{\lambda_{k'}}{\sigma^2} \quad (2.8)$$

Keterangan :

G_k^x : nilai gradien ke-k untuk fungsi fitur node

$G_{k'}^x$: nilai gradien ke-k untuk fungsi fitur edge

k : bernilai 1 sampai banyaknya fitur node

k' : bernilai 1 sampai banyaknya fitur edge

T : banyaknya kata dalam 1 kalimat

$P_t(y_t|x)$: probabilitas lokal node ke-t

$P_t(y_t, y_{t+1}|x)$: probabilitas lokal edge ke-t

$f_k(y'_t, x, t)$: fungsi fitur node ke-k dari data pelatihan yang telah diberi label

$f_k(y_t, x, t)$: fungsi fitur node ke-k untuk semua label y_t yang mungkin

$f_k(y_t, y_{t+1} x, t)$: fungsi fitur edge ke-k untuk label y_t dan y_{t+1} yang mungkin

$f_k(y'_t, y'_{t+1} x, t)$: fungsi fitur edge ke-k dari data pelatihan yang telah diberi label

σ : standar deviasi distribusi Gauss nilainya 0.1 sampai 10 [14]

λ_k : nilai parameter fungsi fitur node ke-k

$\lambda_{k'}$: nilai parameter fungsi fitur edge ke-k

Untuk mendapatkan nilai parameter fungsi fitur λ_k yang optimal persamaan 2.7 dan 2.8 tidak dapat diselesaikan menggunakan sistem persamaan linier dikarenakan nilai probabilitas kondisional label untuk data x_t yakni $P_t(y_t|x)$ dan data yang berdampingan $P_t(y_t, y_{t+1}|x)$ tidak diketahui sehingga persamaan 2.7 dan 2.8 diselesaikan menggunakan prosedur *forward-backward pass* untuk menelusuri sekuens data. Penelusuran dengan *forward-backward pass* bertujuan untuk mendapatakan nilai dari seluruh probabilitas lokal label pada sekuens data. *Forward pass* pada tiap data x_t secara matematis dapat dilihat pada persamaan 2.9 dan 2.10.

Untuk kata pertama

$$\alpha_1[y_1] = 1/S \tag{2.9}$$

Untuk kta kedua sampai kata ke-T

$$\alpha_t[y_t] = \kappa_t \sum_{y_{t-1}} \alpha_{t-1}[y_{t-1}]\, \phi_t(y_t, x)\, \psi_t(y_t, y_{t+1}, x) \tag{2.10}$$

Dimana :

S : jumlah label

κ_t : faktor penskalaan sehingga $\sum_{y_t} \alpha_t[y_t] = 1, \kappa_t > 0$

$\alpha_{t-1}[y_{t-1}]$: nilai forward pass ke t-1

$\phi_t(y_t, x)$: nilai node potensial ke-t

$\psi_t(y_t, y_{t+1}, x)$: nilai edge potensial ke-t

Backward pass pada tiap *data* x_t secara matematis dapat dituliskan dalam persamaan 2.11 dan 2.12.

Untuk kata terakhir

$$\alpha_1[y_1] = 1/S \tag{2.11}$$

Untuk pertama sampai kata T-1

$$\beta_t[y_t] = \gamma_t \sum_{y_{t+1}} \beta_{t+1}[y_{t+1}] \phi_{t+1}(y_{t+1}, x)\, \psi_t(y_t, y_{t+1}, x) \tag{2.12}$$

Dimana:

γ_t : faktor penskalaan sehingga $\sum_{x_t} \beta_t[x_t] = 1, \gamma_t > 0$

$\beta_{t+1}[y_{t+1}]$: nilai backward pass ke-t+1

$\phi_{t+1}(y_t, x)$: nilai node potensial ke-t+1

$\psi_t(y_t, y_{t+1}, x)$: nilai edge potensial ke-t

Setelah semua *forward pass* dan *backward pass* untuk tiap *data* x telah memiliki nilai, maka perhitungan probabilitas lokal untuk setiap label yang mungkin pada data x_t dapat dituliskan dalam persamaan matematis 2.13 dan 2.14 sebagai berikut.

$$P_t(y_t|x) = w_t \alpha_t[y_t] \phi_t(y_t, x) \beta_t[y_t] \quad (2.13)$$

$$P_t(y_t, y_{t+1}|x) = \text{Ш}_t \alpha_t[y_t] \phi_t(y_t, x) \psi_t(y_t, y_{t+1}, x) \phi_{t+1}(y_t, x) \beta_{t+1}[y_{t+1}] \quad (2.14)$$

Dimana

$P_t(y_t|x)$: probabilitas lokal untuk setiap label y_t yang mungkin

w_t : faktor normalisasi $\sum_{y_t} P_t(y_t|x) = 1$.

$\alpha_t[y_t]$: nilai *forward pass* ke-t

$\phi_t(y_t, x)$: nilai node potensial ke-t didapatkan dari persamaan 2.3

$\psi_t(y_t, y_{t+1}, x)$: nilai edge potensial ke-t didapatkan dari persamaan 2.4

$\beta_t[y_t]$: nilai *backward pass* ke-t

$P_t(y_t, y_{t+1}|x)$: probabilitas lokal untuk setiap label y_t dan y_{t+1} yang mungkin

Ш_t : faktor normalisasi memastikan $\sum_{y_t} \sum_{y_{t+1}} P_t(y_t, y_{t+1}|x) = 1$

$\phi_{t+1}(y_t, x)$: nilai node potensial ke-t+1 untuk label y_{t+1}

$\beta_{t+1}[y_{t+1}]$: nilai *backward pass* ke-t+1 untuk label y_{t+1}

Setelah nilai probabilitas lokal tiap label yang mungkin untuk tiap data x_t pada sekuens *x* didapatkan, maka nilai turunan pertama *log-likelihood* akan didapatkan berdasarkan persamaan 2.7 dan 2.8. Kemudian nilai turunan pertama *log-likelihood* G_k^x digunakan untuk memperbarui parameter fungsi fitur λ_k di setiap iterasi berdasarkan persamaan berikut 2.15 dan 2.16.

$$\lambda_k \leftarrow \lambda_k + \omega G_k^x \quad (2.15)$$

$$\lambda_k \leftarrow \lambda_k + \omega G_{k'}^x \quad (2.16)$$

Dimana :

ω : *learning rate* dengan $\omega \in$ [0.001,0.1] [15].

G_k^x : nilai gradien node

$G_{k,l}^x$: nilai gradien edge

2.8. Inferen

Proses berikutnya setelah mendapatkan nilai parameter fitur dari proses pelatihan yakni mengimplementasikan nilai parameter tersebut pada data pengujian. Data uji merupakan sekumpulan data yang telah memiliki nilai-nilai fitur. Pada tahap ini, semua data berurutan pada sebuah kalimat akan mendapatkan label. Himpunan parameter fungsi fitur yang didapatkan dari proses pelatihan akan digunakan untuk melakukan inferen terhadap sekumpulan data uji yang akan diprediksi semua labelnya dengan menemukan sekuens label yang paling optimal. Ada tiga tahap untuk melakukan *decoding* yaitu menghitung fungsi potensial, *maximal forward pass*, dan *backtracking*. *Maximal Forward Pass* merupakan proses penelusuran sekuens data mulai dari data ke-1 hingga data ke-*T* untuk mendapatkan nilai maksimal variabel *maximal forward* ($\alpha_t^{max}[x_t]$) untuk tiap data pada posisi t. Variabel *maximal forward* ini adalah fungsi normalisasi bagi setiap label untuk data ke-*t*.

Pada awal iterasi, $\alpha_1^{max}[x_1]$ diinisialisasi dengan nilai $1/S$ untuk tiap label dengan *S* adalah jumlah label yang mungkin diberikan pada sekuens data uji. Selanjutnya, penelusuran *maximal forward pass* dilakukan terhadap sekuens data secara matematis dapat dituliskan :

Untuk kata terakhir

$$\alpha_1[y_1] = 1/S \tag{2.15}$$

Untuk kedua sampai kata T-1

$$\alpha_t^{max}[y_t] = \kappa_t max_{yt-1}(\alpha_{t-1}[y_{t-1}^{max}]\phi_t(y_t, x)\psi_t(y_t, y_{t+1}, x)) \tag{2.16}$$

Dimana :

S : jumlah label

κ_t : faktor normalisasi

$\alpha_{t-1}[y_{t-1}^{max}]$: nilai maksimal *forward pass* saat t-1

$\phi_t(y_t, x)$: nilai node potensial ke-t

$\psi_t(y_t, y_{t+1}, x)$: nilai edge potensial ke-t

Pada tahap selanjutnya yaitu b*acktracking* untuk menelusuri kembali dan meninjau label optimal pada tiap data berdasarkan nilai dari *maximal forward pass* yang dilakukan sebelumnya. Persamaan *backtracking* secara matematis pada persamaan 2.17.

$$y_t^* = arg\ max_{y_t}(a_t^{max}[y_t]\ \phi_t(y_t, z)) \quad (2.17)$$

sehingga sekuens label optimal adalah $\{y_1^*, y_2^*, \dots, y_T^*\}$ untuk sekuens data pengujian sejumlah *T*.

Dimana :

$a_t^{max}[y_t]$: nilai *maksimal forward* pass ke-t

$\phi_t(y_t, z)$: nilai node potensial ke-t

2.9. Pengenalan Entitas Menggunakan Conditional Random Fields (CRF)

Pengenalan entitas menggunakan CRF menggunakan *supervised learning* yakni pendekatan pembelajaran dengan menggunakan data yang sudah anotasi atau diberi labelentitas. Label entitas yang digunakan adalah <PERSON> yang menunjukkan entitas nama orang, <LOCATION> untuk entitas lokasi, label <ORGANIZATION> yang menujukkan nama organisasi, dan label <OTHER> yang menujukkan bukan entitas. Data pelatihan digunakan untuk membuat model yang akan diterapkan pada data pengujian.

2.10. *Precision, Recall dan F-Measure*

Pengukuran evaluasi dilakukan untuk menguji keakuratan simulator yang telah dibuat. Pengukuran evaluasi yang digunakan adalah parameter akurasi untuk mengukur performansi dari simulator. Akurasi adalah tingkat kedekatan pengukuran kuantitas terhadap nilai yang sebenarnya. Perhitungan *precision*, *recall*, dan *f-measure* adalah sebagai berikut [16]:

$$Precision = \frac{true\ positif}{true\ positif + false\ positif} \quad (2.18)$$

$$Recall = \frac{true\ positif}{true\ positif + false\ negatif} \quad (2.19)$$

$$F - Measure = \frac{true\ positif + true\ negatif}{true\ positif + false\ positif + true\ negatif + false\ negatif} \quad (2.20)$$

Untuk entitas *person*, *location*, dan *organization* yang diidentifikasi secara benar dihitung sebagai *true positif*, sedangkan yang salah dihitung sebagai *false positif*. Entitas *other* yang diidentifikasikan secara benar dihitung sebagai *true negatif*, sedangkan yang salah dihitung sebagai *false negatif*.

Tabel 2.2 Contoh Perhitungan Akurasi

Kata	Label Data Uji	Label Prediksi
Gatot	PER	O
Berasal	O	O
Dari	O	O
Depok	LOC	LOC

Perhitungan akurasi dilakukan dengan cara membandingkan hasil prediksi label entitas dari simulator dengan label entitas yang sebenarnya. Dari Tabel 2.2 di atas diketahui bahwa label yang diprediksi dengan benar berjumlah 3, yaitu pada kata "berasal", "dari" dan "Depok" sedangkan label yang diprediksi salah adalah 1, yaitu kata "Gatot". Berdasarkan persamaan 2.20 didapatkan nilai akurasi dari sistem yaitu 33.3%.

2.11. Model Analisis dan Perancangan

Model analisis yang digunakan untuk membangun simulator ini adalah *Unified Modeling Language* (UML). UML adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan

perangkat lunak). *Artifact* dapat berupa model, deskripsi atau perangkat lunak dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem nonperangkat lunak lainnya [17]. Bagian-bagian UML adalah sebagai berikut :

1. *Use case diagram*

   Use case diagram adalah suatu aktivitas sistem menanggapi permintaan pengguna dalam sebuah masalah. Sistem digunakan oleh pengguna (*actor*) [17].

2. *Activity diagram*

   Activity diagram digunakan untuk menggambarkan aliran kontrol dari sistem, seperti aturan bisnis, menggambarkan jalannya fungsi yang ada, dan menggambarkan kasus penggunaan juga proses bisnis [17].

3. *Sequence diagram*

   Sequence diagram merupakan kolaborasi objek berdasarkan urutan waktu. *Sequence diagram* menunjukkan bagaimana objek berinteraksi dengan objek lain dalam skenario tertentu dari sebuah *use case* [17].

4. *Class diagram*

   Class diagram menggambarkan objek dan kelas dalam sistem dan hubungan antara mereka. *Class diagram* juga terdapat atribut dan *method-method* yang ada pada sistem [17].

2.12. Java Netbeans

NetBeans mengacu pada dua hal, yakni *platform* untuk pengembangan aplikasi *desktop* java, dan sebuah *Integrated Development Environment* (IDE) yang dibangun menggunakan *platform* NetBeans. Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut modul. Sebuah modul adalah suatu arsip Java (*Java archive*) yang memuat kelas-kelas Java untuk berinteraksi dengan *NetBeans Open API* dan file manifestasi yang mengidentifikasinya sebagai modul [18].

2.12.1. IDE NetBeans 8.0.2

NetBeans IDE 8.0.2 ditulis dalam bahasa pemrograman Java dan dapat dijalankan pada sistem operasi yang telah terinstal Java Virtual Machine (JVM). NetBeans IDE 8.0.2 dapat digunakan pada sistem operasi Windows, Mac OS, Linux, dan Solaris. Sebuah JDK dibutuhkan untuk fungsi pengembangan Java, tetapi tidak diperlukan untuk pemgembangan aplikasi dengan bahasa pemrograman lain [18].

2.12.2. Kelemahan Netbeans

Beberapa keunggulan Netbeans adalah sebagai berikut [18]:

1. *Netbeans GUI Builder* didapatkan secara gratis dan memiliki banyak *plug in* yang bisa didapatkan langsung pada website resmi maupun dari pihak ketiga.
2. *Netbeans GUI Builder* sangat kompatibel dengan *swing* karena dikembangkan langsung oleh sun Microsystem sebagai pengembang *swing*.
3. *Netbeans* tidak hanya dapat digunakan buat java saja, karena Netbeans dapat digunakan untuk bahasa pemograman lain seperti C/C++,Ruby, dan PHP.
4. *Netbeans GUI Builder* sangat cocok digunakan dalam pengembangan sistem berskala Enterprise.
5. Pada paket tertentu, *Netbeans* menyertakan *GlassFish* V2 UR2 dan *Apache Tomcat* 6.0.16.

2.12.3. Kelemahan Netbeans

NetBeans juga memiliki kelemahan seperti berikut [18].

1. NetBeans hanya mensupport satu penembangan java GUI (*swing)*
2. NetBeans mempatenkan *source* untuk java GUI yang sedang dikerjakan dalam sebuah *Generated Code*, sehingga programmer tak dapat mengeditnya secara manual.
3. Dari segi sumber daya, NetBeans memerlukan sumber daya yang besar, seperti memory dan ruang *hard disk*.
4. Netbeans memerlukan dukungan prosesor yang cukup handal untuk mendapatkan performa maksimalnya.