BAB 2
TINJAUAN PUSTAKA

2.1 Ekstraksi Informasi

Ekstraksi Informasi merujuk pada ekstraksi otomatis dari informasi terstruktur seperti entitas, hubungan antar entitas, dan atribut deskripsi entitas dari sumber yang tidak terstruktur [1]. Dalam sebagian besar kasus, kegiatan ini berkaitan dengan pemrosesan teks bahasa manusia melalui pemrosesan bahasa alami (NLP). Kegiatan terbaru dalam pemrosesan dokumen multimedia seperti anotasi otomatis dan ekstraksi konten dari gambar / audio / video / dokumen dapat dilihat sebagai ekstraksi informasi. Dalam penelitian ini akan melakukan ekstraksi informasi pada dokumen Surat Keputusan.

2.2 Dokumen Surat Keputusan

Menurut Kamus Besar Bahasa Indonesia, dokumen adalah surat yang tertulis atau tercetak yang dapat dipakai sebagai bukti keterangan. Surat tugas atau surat keputusan adalah surat yang menerangkan bahwa orang yang diberi surat itu diperintahkan atau diberi tugas untuk menjalankan sesuatu.

Pada penelitian ini, dokumen yang akan digunakan adalah dokumen surat keputusan. Dokumen yang digunakan hanya dokumen surat keputusan yang ditujukan kepada 1 (satu) orang. Lembar dokumen surat keputusan terdiri dari Kepala Surat, Jenis Surat, Nomor Surat, Tentang, Jabatan yang Menetapkan Surat, Menimbang, Mengingat, Putusan, Nama yang Dituju, Nomor Induk yang Dituju, Isi Putusan, Tempat Diputuskan, Tanggal Diputuskan, Organisasi, Nama yang Menetapkan Surat, dan Tembusan. Berikut adalah bagian-bagian atau kategori yang terdapat pada sebuah lembar dokumen surat keputusan.

Tabel 2. 1 Bagian Pada Dokumen Surat Keputusan

No	Kategori	Kelas
1	Kepala Surat	0
2	Jenis Surat	1
3	Nomor Surat	2
4	Tentang	3
5	Jabatan yang Menetapkan Surat	4
6	Menimbang	5
7	Mengingat	6
8	Memutuskan	7
9	Nama yang Dituju	8
10	Nomor Induk yang Dituju	9
11	Tempat Diputuskan	10
12	Tanggal Diputuskan	11
13	Organisasi	12
14	Nama yang Menetapkan Surat	13
15	Tembusan	14

2.3 Algoritma

Pada bidang teknologi, untuk menyusun suatu rencana proses sistem diperlukan pembuatan algoritma. Menurut Ritayani, algoritma adalah urutan langkah – langkah logis penyelesaian masalah yang disusun secara sistematis [8].

Algoritma disusun secara sistematis, lalu diimplementasikan ke dalam bentuk notasi. Notasi algoritma bukan merupakan notasi bahasa pemrograman, melainkan notasi yang dapat diterjemahkan kedalam berbagai bahasa pemrograman. Berikut tiga macam dalam pembuatan notasi algoritma.

1. Deskriptif

   Algoritma yang ditulis dalam bahasa Indonesia atau bahasa asing. Berbentuk kalimat, namun untuk notasinya kurang efektif dan terkadang relatif sukar untuk diterjemahkan ke bahasa pemrograman.

2. *Pseudocode*

   Pseudocode terbagi kepada 2 kata, *pseudo* dan *code. Pseudo* memiliki arti imitasi, sedangkan *code* adalah intruksi yang ditulis dalam bahasa komputer.

Jika didefinisikan, *pseudocode* adalah logika urutan – urutan dari program tanpa memandang bahasa pemrograman.

3. *Flowchart*

   Algoritma yang digambarkan dalam bentuk diagram alir. Diagram alir merupakan aliran yang berlangsung ketika suatu bagian (prosedur) dalam program dieksekusi.

Pada penelitian ini, algoritma deskriptif akan digunakan untuk menjelaskan fungsi pembobotan token pada bagian ekstraksi fitur. Sedangkan untuk algoritma berbentuk *flowchart,* akan digunakan untuk menggambarkan prosedur yang terdapat pada sistem ekstraksi informasi menggunakan CRF.

2.4 Pemodelan Sistem

Suatu sistem yang memiliki beberapa proses di dalamnya harus dimodelkan untuk memperjelas gambaran yang terjadi pada proses tersebut. Berikut pemodelan sistem yang digunakan pada penelitian ini, yang terdiri dari Blok Diagram, *DFD*, Diagram Konteks, dan *Flowchart*.

2.4.1 *Blok Diagram*

Proses yang terjadi di dalam suatu sistem, dapat digambarkan dengan mudah melalui data masukan, proses, serta keluaran yang dihasilkan. Model yang memiliki ketiga tahapan dinamakan dengan model blok diagram. Blok diagram digunakan untuk merepresentasikan suatu sistem atau sejumlah blok yang berhingga dalam rangkaian beberapa proses menggunakan blok. Elemen yang terdapat pada diagram blok terdiri dari sebab akibat *input* dan *outpu t*[9]. Berikut gambaran blok diagram secara umum.

Gambar 2. 1 Blok Diagram

Penggunaan blok diagram pada penelitian ini untuk menggambarkan beberapa proses yang terjadi pada sistem dari mulai data masukan sampai hasil yang diharapkan.

2.4.2 *Data Flow Diagram*

Model diagram yang dapat menggambarkan keterkaitan proses yang ada pada sistem secara mendalam dapat dibuat dengan menggunakan diagram *Data Flow Diagram* (DFD). DFD adalah suatu *model* logika data atau proses yang dibuat untuk menggambarkan darimana asal data, dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut, dan interaksi antara data yang tersimpan, dan proses yang dikenakan pada data tersebut [10]. Terdapat 4 simbol utama pada *DFD* yang terdiri dari:

1. Entitas Luar (*External Entity*)
   Entitas luar digunakan untuk menyatakan departemen, kantor, organisasi, orang, maupun sekelompok orang.
2. Arus Data (*Data Flow*)
   Arus data digunakan untuk menggambarkan masukan maupun keluaran dari proses.
3. Proses (*Process*)
   Proses digunakan untuk menggambarkan pekerjaan yang dilakukan oleh manusia maupun sistem.
4. Simpanan Data (*Store Data*)
   Simpanan data digunakan untuk penyimpanan data dari suatu sistem maupun memanggil suatu data dari simpanan data.

Pada penelitian ini, *DFD* digunakan untuk menggambarkan beberapa proses yang terjadi pada sistem ekstraksi informasi menggunakan CRF.

2.4.3 Diagram Konteks

Suatu *model* sistem yang belum dilakukan analisis secara mendalam terhadap proses yang terdapat pada sistem, dinamakan dengan diagram konteks. Diagram konteks merupakan bagian dari *Data Flow Diagram* (DFD) yang hanya memiliki satu simbol. Istilah lain diagram konteks adalah *context diagram* yang merupakan

top level pada penggambaran *Data Flow Diagram* (DFD). Diagram konteks memiliki karakteristik, yaitu:

1. Hanya mengandung satu proses saja.
2. Tidak diberikan penomoran khusus pada prosesnya.
3. Semua arus data dan datanya sendiri digambarkan secara jelas.

Pada penelitian ini, diagram konteks digunakan untuk menjelaskan alur masukan, proses dan keluaran data secara menyeluruh.

2.4.4 *Flowchart*

Suatu aliran data yang terjadi pada sistem dapat digambarkan dengan *model* diagram *flowchart*. Menurut Rasim, Setiawan, dan Rahman, *Flowchart* atau bagan alir adalah suatu bagan yang berisi simbol – simbol grafis yang menunjukan arah aliran kegiatan dan data-data yang dimiliki program sebagai suatu proses eksekusi [11]. *Flowchart* memiliki konsep dasar, sama halnya seperti pada blok diagram, yaitu terdapat *input*, proses dan *output*, dan sebagai tambahannya memiliki simbol kondisional diantara *input* dan *output*. *Flowchart* pada penelitian ini digunakan untu menggambarkan perancangan prosedur – prosedur yang terdapat pada sistem ekstraksi informasi.

2.5 Tokenisasi

Suatu proses untuk memisahkan setiap kata dari rangkaian kalimat dinamakan dengan proses tokenisasi. Menurut Amin, token seringkali disebut sebagai istilah (*term*) atau kata. Sebuah token merupakan suatu urutan karakter dari dokumen tertentu yang dikelompokkan sebagai unit semantik yang berguna untuk diproses [12].

2.6 Ekstraksi Fitur

Menurut Prihatini, ekstraksi fitur merupakan proses untuk mencari nilai - nilai fitur yang terkandung dalam dokumen [13]. Fitur dapat diartikan sebagai ciri dari setiap data yang dikenali oleh sistem sehingga menghasilkan nilai fitur. Ekstraksi fitur merupakan topik penting dalam klasifikasi, karena fitur-fitur yang

baik akan sanggup meningkatkan tingkat akurasi, sementara fitur-fitur yang tidak baik cenderung memperburuk tingkat akurasi [14].

Pada penelitian ini, terdapat tiga fitur, yaitu fitur kamus, *node* dan *edge*. Fitur kamus adalah sebuah fungsi dari masukan bernama pola data yang menangani data mentah dan mengembalikan nilai, berikut adalah contoh fitur kamus.

$$g_1(x,2)\begin{cases}1\ if\ x_1 = "AKADEMISEKRETARI" \\ 0 \qquad\qquad otherwise\end{cases}$$

Fitur diatas merupakan fitur ke-2 yang mengembalikan nilai 1 pada kata ke 2 jika kata sebelumnya adalah “AKADEMISEKRETARI”. Fitur *node* merupakan asosiasi antara label *y* dengan data masukan *x*, berikut contoh fitur *node*.

$$f_1(y_1,x)\begin{cases}1\ if\ y_1 = KS, x = "AKADEMISEKRETARI" \\ 0 \qquad\qquad otherwise\end{cases}$$

Fitur diatas merupakan fitur ke-t yang bernilai 1 jika label KS dan kata yang sedang diproses adalah “AKADEMISEKRETARI”, jika bukan maka bernilai 0. Sedangkan fitur *edge* merupakan asosiasi antara label kelas kata untuk kata yang sedang diproses dengan label kelas kata untuk kata setelahnya, berikut contoh fitur *edge*.

$$f_1(y_1,y_2,x)\begin{cases}1\ if\ y_1 = KS, y_2 = KS \\ 0 \qquad\qquad otherwise\end{cases}$$

Fitur diatas merupakan fitur ke-t yang bernilai 1 jika label kelass kata untuk kata yang sedang diproses merukapakan KS dan label kelas kata untuk kata setelahnya adalah KS, jika bukan maka bernilai 0.

2.7 *Conditional Random Fields*

CRF merupakan *framework* untuk membangun model probabilistik untuk segmentasi dan pelabelan data yang berurutan. Kelebihan utama CRF dibandingkan dengan *hidden markov model* (HMM) adalah sifat bersyaratnya (*conditional*), sehingga mengurangi ketergantungan asumsi yang dibutuhkan oleh HMM untuk memastikan inferensi mudah dikerjakan [15]. Selain itu CRF juga menghindari masalah label bias, kelemahan yang ditunjukkan oleh *maximum entropy markov*

model (MEMM) dan model *conditional markov* lain yang merupakan *directed graphical models* [16].

Diperkenalkan oleh Lafferty et al (2001) [15], *Conditional Random Field* (CRF) merupakan model probabilistik untuk menghitung probabilitas $p(y|x)$ dari output yang memungkinkan $y = (y_1, y_2, \dots, y_n)$ terhadap input yang diberikan $x = (x_1, x_2, \dots, x_n)$ yang disebut juga data observasi [17]. Probabilitas kondisional dari *y* terhadap *x* dapat dituliskan dalam persamaan [18]:

$$P(y|x) = \frac{1}{Z(x)} \prod_c \psi_c(y_c, x) \tag{2.1}$$

$$Z(x) = \sum_y \prod_c \psi_c(y_c, x) \tag{2.2}$$

$\psi_c(y_{c,}x)$ merupakan fungsi positif atau fungsi potensial, dan $Z(x)$ merupakan fungsi partisi atau fungsi normalisasi. Dalam implementasinya, ada dua jenis fungsi potensial, yaitu *node* potensial $\phi(y_t, x)$ dan *edge* potensial $\psi(y_t, y_{t+1}, x)$. *Node* potensial dapat dinotasikan kedalam persamaan berikut [18].

$$\phi_t(y_t, x) = \exp(\sum_k \lambda_k f_k(y_t, x, t)) \tag{2.3}$$

Sedangkan untuk *edge* potensial dapat dituliskan sebagai berikut [18].

$$\psi_t(y_t, y_{t+1}, x) = \exp(\sum_k \lambda_{k'} f_{k'}(y_t, y_{t+1}, x, t)) \tag{2.4}$$

Dimana:

x	: data masukan
y_t	: label untuk data x
y_{t+1}	: label setelah y_t
t	: indeks kata ke-t dari data input
λ_k	: parameter fitur *node*
$\lambda_{k'}$	: parameter fitur *edge*
$f_k(y_t, x, t)$	: fitur *node* ke-k

$f_{k'}(y_t, y_{t+1}, t)$: fitur *edge* ke-k

Proses *training* dilakukan untuk mendapatkan nilai optimal untuk parameter fitur *node* dan parameter fitur *edge*. Nilai parameter didapatkan dengan menggunakan *maximum likelihood* yang dapat dituliskan [18]:

$$\mathcal{L} = \sum_{t\in[1,T]}\sum_k \lambda_k f_k(y_t, x, t) - \sum_{t\in[1,T-1]}\sum_{k'} \lambda_{k'} f_{k'}(y_t, y_{t+1}, x, t) - \log Z(x) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \quad (2.5)$$

Dimana:

$t \in [1, T]$: banyaknya kata

λ_k , $\lambda_{k'}$: parameter fitur ke-k

$f_k(y_t, x, t)$, $f_{k'}(y_t, y_{t+1}, x, t)$: fitur ke-k

$Z(x)$: fungsi partisi

σ : standar deviasi

Untuk mendapatkan parameter λ untuk tiap fungsi fitur, maka digunakan turunan *log-likelihood*, yaitu [18]:

$$G_k^x = \sum_{t\in[1,\mathrm{T}]}\left(f_k(y_t, x, t) - \sum_{y_t} P_t(y_t|x) f_k(y_t, x, t)\right) - \frac{\lambda_k}{\sigma^2} \quad (2.6)$$

$$G_{k'}^x = \sum_{t\in[1,\mathrm{T}-1]}\left(f_{k'}(y_t, y_{t+1}, x, t) - \sum_{y_t}\sum_{y_{t+1}} P_t(y_t, y_{t+1}|x) f_{k'}(y_t, y_{t+1}, x, t)\right) - \frac{\lambda_{k'}}{\sigma^2} \quad (2.7)$$

Dimana:

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

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

$f_k(y_t, x, t)$: fitur *node* ke-k

$f_{k'}(y_t, y_{t+1}, x, t)$: fitur *edge* ke-k

$P_t(y_t|x)$: probabilitas label y_t terhadap data x

$P_t(y_t, y_{t+1}|x)$: probabilitas label y_t diikuti label y_{t+1} ketika data x

λ_k : parameter fitur *node* ke-k

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

σ : Standar deviasi

Persamaan di atas merupakan *stochastic gradient method.* Untuk menghitung $P_t(y_t|x)$ dan $P_t(y_t, y_{t+1}|x)$ digunakan prosedur *two-pass* terhadap data. Prosedur ini dikenal sebagai *forward-backward.*

a. *Forward Pass*

Forward pass melakukan penelusuran dari indeks t = 1 hingga t = T. Untuk menghitung seluruh probabilitas lokal diperlukan penelusuran yang berlawanan arah (*backward pass*).

Variabel *forward* dilambangkan oleh $\alpha_t[y_t]$. Ketika t =1 inisialisasi $\alpha_1[y_1] = 1/S$ untuk seluruh label y_t, dimana S merupakan jumlah seluruh label kelas kata. Ketika $t \geq 2$ maka dapat digunakan persamaan sebagai berikut [18].

$$\alpha_t[y_t] = k_t \sum_{x_{t-1}} \alpha_{t-1}[y_{t-1}]\phi_t(y_t, x)\psi_{t-1}(y_{t-1}, y_t, x) \quad (2.8)$$

Dimana:

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

k_t : faktor penskalaan untuk memastikan $\sum_{y_t} \alpha_t[y_t]$=1

$\alpha_{t-1}[y_{t-1}]$: variabel *forward* pass sebelumnya

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

$\psi_{t-1}(y_{t-1}, y_t, x)$: nilai *edge* potensial sebelumnya

b. *Backward Pass*

Backward pass hampir sama dengan *forward pass.* Perbedaannya hanya menelusuri dari indeks t = T sampai t = 1. Variabel *Backward* dilambangkan $\beta_t[y_t]$. Ketika t = T, inisialisasi $\beta_T[y_T] = 1/S$ untuk semua

label y_T ketika $t < T$ maka dapat digunakan persamaan sebagai berikut [18].

$$\beta_t[y_t] = \mu_t \sum_{y_{t+1}} \beta_{t+1}[y_{t+1}] \phi_{t+1}(y_{t+1}, x) \psi_t(y_t, y_{t+1}, x) \quad (2.9)$$

Dimana:

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

μ_t : faktor penskalaan untuk memastikan $\sum_{y_t} \beta_t[y_t] = 1$

$\beta_{t+1}[y_{t+1}]$: nilai variabel *backward pass* setelahnya

$\phi_{t+1}(y_{t+1}, x)$: nilai *node* potensial setelahnya

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

Selanjutnya dilakukan perhitungan probabilitas *node* dan probabilitas *edge*.

a. Probabilitas *Node*

Untuk mendapatkan probablitas *node* yang menghitung probabilitas label kelas kata y_t terhadap data masukan x, dapat dihitung dengan menggunakan persamaan berikut [18].

$$P_t(y_t|x) = w_t \alpha_t[y_t] \, \phi_t(y_t, x) \beta_t[y_t] \quad (2.10)$$

Dimana:

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

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

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

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

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

b. Probabilitas *Edge*

Sedangkan untuk menghitung probabilitas *edge* yang menghitung probabilitas label y_t untuk data x diikuti oleh label y_{t+1} yang merupakan label kelas kata untuk kata setelahnya dapat dihitung menggunakan persamaan berikut [18].

$$P_t(y_t, y_{t+1}|x) = \gamma_t \alpha_t[y_t]\, \phi_t(y_t, x)\psi_t(y_t, y_{t+1}, x)\phi_{t+1}(y_{t+1}, x)\beta_{t+1}[y_{t+1}] \quad (2.11)$$

Dimana:

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

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

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

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

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

$\phi_{t+1}(y_{t+1}, x)$: nilai *node* potensial setelahnya

$\beta_{t+1}[y_{t+1}]$: nilai *backward pass* setelahnya

Proses *testing* dilakukan untuk menentukan label kelas kata untuk seluruh data, bukan hanya satu *node* saja. Secara teori, untuk menemukan urutan label yang paling memungkinkan untuk diberikan terhadap data masukan. Pada langkah ini digunakan metode *Viterbi Decoding.* Ada dua langkah prosedur pada metode ini, yaitu [18]:

a. *Maximal Forward Pass*

Maximal forward pass digunakan untuk menemukan probabilitas paling optimal untuk data masukan dengan mempertimbangkan hubungan antara label kelas kata untuk kata yang sedang diproses dengan label kelas kata untuk kata setelahnya. Ketika t = 1, inisialisasi $\alpha_1^{max}[y_1] = 1/S$ untuk semua label y_1 Ketika $t \geq 2$ maka dapat digunakan persamaan sebagai berikut.

$$\alpha_t^{max}[y_t] = k_t \max_{y_{t-1}} (\alpha_{t-1}[y_{t-1}^{max}]\phi_t(y_t, x)\psi_{t-1}(y_{t-1}, y_t, x)) \quad (2.12)$$

Dimana:

$\alpha_t^{max}[y_t]$: nilai maksimal untuk $\alpha_t[y_t]$

k_t : faktor penskalaan untuk memastikan $\sum_{y_t} \alpha_t^{max}[y_t] = 1$

$\alpha_{t-1}[y_{t-1}^{max}]$: nilai maksimal untuk label kelas kata untuk kata sebelumnya

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

$\psi_{t-1}(y_{t-1}, y_t, x)$: nilai *edge* potensial sebelumnya

b. *Backtracking Pass*

Proses *backtracking* digunakan untuk menemukan label paling optimal. Adapun persamaannya adalah sebagai berikut.

$$y_t^* = \arg max_{y_t}\left(\alpha_t^{max}[y_t]\phi_t(y_t, x)\right) \quad (2.13)$$

Dimana:

$\alpha_t^{max}[y_t]$: nilai maksimal untuk $\alpha_t[y_t]$

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

2.8 Bahasa Pemrograman

Bahasa pemrograman merupakan bahasa yang digunakan dalam pembangunan sistem ekstraksi informasi menggunakan CRF. Bahasa pertama yang digunakan adalah *PHP* (*Personal Home Page*) sebagai pemrosesan Ekstraksi Fitur, dan proses *Training* dan *Testing* CRF.

2.8.1 PHP

PHP (akronim dari PHP: Hypertext Preprocessor) adalah bahasa pemrograman yang berfungsi untuk membuat website dinamis maupun aplikasi web. Berbeda dengan HTML yang hanya bisa menampilkan konten statis, PHP bisa berinteraksi dengan database, file dan folder, sehingga membuat PHP bisa

menampilkan konten yang dinamis dari sebuah website. Blog, Toko Online, CMS, Forum, dan Website Social Networking adalah contoh aplikasi web yang bisa dibuat oleh PHP. PHP adalah bahasa scripting, bukan bahasa tag-based seperti HTML. PHP termasuk bahasa yang cross-platform, ini artinya PHP bisa berjalan pada sistem operasi yang berbeda-beda (Windows, Linux, ataupun Mac). Program PHP ditulis dalam file plain text (teks biasa) dan mempunyai akhiran ".php" [19].

2.9 Perangkat Lunak Pendukung

Perangkat Lunak Pendukung merupakan suatu kebutuhan perangkat yang digunakan untuk pembangunan sistem ekstraksi informasi menggunakan CRF.

2.9.1 *Apache*

Apache adalah sebuah nama web server yang bertanggung jawab terhadap *request-response HTTP* dan *logging* informasi secara detail. Selain itu, *apache* juga diartikan sebagai suatu *web server* yang kompak, modular, mengikuti standar protocol HTTP, dan tentu saja sangat digemari [20]. Pada penelitian ini, a*pache* terinstall pada *XAMPP*.

2.10 CSV

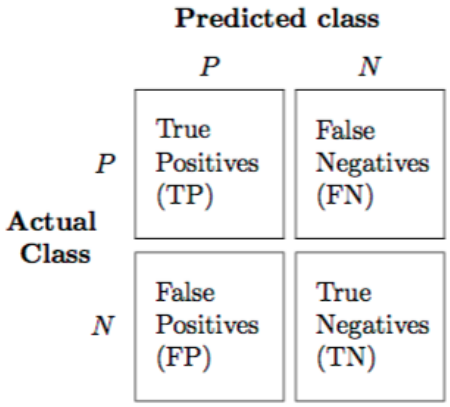
Comma Separated Values (CSV) adalah format dasar yang terdiri dari file teks yang berisi *line* dan *value*[21]. Pada file *CSV*, terdapat beberapa istilah seperti *Value*, *Line*, *Header*, *Row*, dan *Cell*. *Line* adalah setiap baris *header* yang tidak termasuk sebagai *row*. *Value* adalah konten pada *cell* untuk *row*. *Cell* merupakan kolom, dan *row* merupakan baris.

Pada penelitian ini, file *CSV* digunakan untuk keperluan *data training* dan *data testing*. *Data training* yang disimpan pada file *CSV*, memiliki tujuan agar data dapat tersusun dengan rapih terhadap banyaknya data yang digunakan.

2.11 *Confusion Matrix*

Confusion Matrix adalah sebuah tabel yang menyatakan jumlah data uji yang terklasifikasi dengan benar dan jumlah data uji yang terklasifikasi dengan salah [22]. *Confusion Matrix* merupakan matriks yang terdiri dari jumlah *True*

Positive, True Negative, Falsse Positive, dan *False Negative.* TP dan TN menggambarkan klasifikasi yang benar sedangkan FP dan FN menggambarkan klasifikasi yang salah. Adapun gambaran *confusion matrix* ditunjukan pada gambar berikut.



Gambar 2. 2 *Confusion Matrix*

Adapun untuk perhitungan tingkat akurasi menggunakan persamaan berikut.

$$Akurasi = \frac{TP + TN}{TP + FN + FP + TN} \text{ x } 100\%$$

Keterangan: TP : *True Positive*

TN : *True Negative*

FP : *False Positive*

FN : *False Negative*