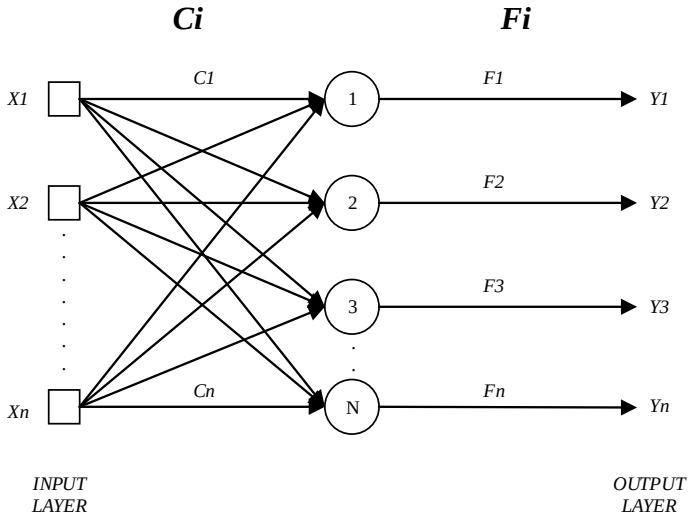
BAB 2
TINJAUAN PUSTAKA

2.1 *Learning Vector Quantization*

Leaning Vector Quantization (*LVQ*) adalah algoritma Jaringan Syaraf Tiruan, yang nilai keluarannya mengklasifikasikan kedalam sebuah kelas, metode *Learning Vector Quantization* digunakan untuk klasifikasi yang jumlah kelasnya sudah ditentukan [8]. Keunggulan dari metode *LVQ* adalah mampu untuk memberikan pelatihan terhadap lapisan-lapisan kompetitf sehingga secara otomatis dapat mengklasifikasikan vector masukan yang diberikan. Jaringan saraf tiruan merupakan representasi buatan yang mencoba mensimulasikan proses pembelajaran pada otak manusia. Jaringan *LVQ* dibuat oleh Teuvo Kohonen untuk versi *supervised learning* dari algoritma *Self-Organizing Maps* (*SOM*) [9]. Jaringan *LVQ* termasuk pada lapisan yang kompetitif, terdiri dari dua *layer* dan bisa bekerja untuk melakukan klasifikasi multi kelas [10], yang berarti jika 2 *vector input* mendekati sama, maka lapisan komptetitif akan meletakkan kedua *vector input* tersebut ke dalam kelas yang sama. Berikut struktur jaringan *LVQ* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur Jaringan *LVQ*

Penjelasan dari struktur jaringan *LVQ* yaitu, dimana *Xn* merupakan *input layer* atau dapat disebut sebagai *input vector*. Sedangkan simbol lingkaran yang memiliki angka dari 1 sampai dengan N, merupakan jarak (*distance)* atau *euclidean distance* yang berfungsi untuk menghitung nilai jarak antara *input vector Xn* dengan *codebook vectors* sebagai *Cn*, *Codebook vector* merupakan perwakilan *vector* bobot dari masing-masing kelas yang sudah ditentukan dari data *training* [10].

Untuk simbol *Y1* sampai dengan *Yn*, merupakan wilayah *output layer* yang terdiri dari kelas–kelas *Y* ke-n. Diantara *distance* dan *output layer* terdapat fungsi aktifasi yang direpresentasikan sebagai *Fn*, yang akan bekerja jika pada hasil perhitungan jarak seperti || X – W1 || < || X – W2|| Y1 akan bernilai 1 dan Y2 akan bernilai 0 [11]. Dalam artian lain, hal tersebut merupakan proses kompetitif yang terjadi pada kompetitif *layer LVQ*.

Dengan demikian, prinsip kerja algoritma *LVQ* adalah melakukan pembelajaran secara otomatis untuk mengklasifikasikan vektor *input* kedalam kelas-kelas tertentu. Kelas yang dihasilkan tergantung pada jarak terdekat antara vector–vector *input*. Untuk memperjelas algoritma *LVQ*, berikut parameter dan langkah–langkah untuk melakukan *training* dan *testing LVQ*.

Langkah Proses *Training LVQ* [12]

1. Inisialisasi bobot awal atau *Codebook Vectors* (C), *Epoch* ($Epoch$), *Max Epoch* ($MaxEpoch$), *Learning Rate* awal $\alpha(0)$, *Learning Rate* selanjutnya $\alpha(t+1)$, *Learning Rate* Baru $\alpha(t)$, Pengurangan *Learning Rate* $\alpha(u)$, dan *Error* minimum yang diharapkan (Eps).
2. Tetapkan kondisi awal untuk $Epoch = 0$.
3. Kerjakan $Epoch$ sampai $MaxEpoch$ jika ($Epoch < MaxEpoch$) atau ($\alpha > Eps$).
4. Tentukan nilai jarak minimum dengan *Euclidean Distance* [10]:

$$d_{i,j} = \sqrt{\sum_{j=0}^{J} \left(X_{i,j} - C_{i,j}\right)^2} \quad (2.1)$$

Keterangan :

$d_{i,j}$ = jarak *euclidean distance vector* i dengan banyak data j.

$\sum$ = Sigma atau jumlah dari perhitungan *input vector* dengan *codebook vector*.

$X_{i,j}$ = *Input Vector* ke-i,j

$C_{i,j}$ = *Codebook Vector* ke-i,j

- Nilai $d_{i,j}$ akan direpresentasikan sebagai kelas jarak minimum yaitu $c_w$dari *array* C dengan baris dan kolom $i\ x\ j$ dimana i adalah total *codebook vectors* ke-i dan *j* adalah total *neuron* ke-j. Sedangkan untuk w, merupakan indeks pemenang dari setiap iterasi yang dilakukan.

5. Perbaiki/Perbaharui/Update c_w dengan ketentuan [10] :
   - Jika $kelas\ x_r$ sama dengan kelas c_w maka :

     $$c_w = c_w + \alpha(t) \cdot (x_i - c_w) \quad (2.2)$$

     atau bisa dibaca dengan [5]

     $$c_{i,j(baru)} = c_{i,j(lama)} + \alpha(t) \cdot \left(x_{i,j} - c_{i,j(lama)}\right) \quad (2.3)$$

   - Jika kelas x_r tidak sama dengan kelas c_w maka :

     $$c_w = c_w - \alpha(t) \cdot \left(x_{ij} - c_w\right) \quad (2.4)$$

     atau bisa dibaca dengan [13]

     $$c_{i,j(baru)} = c_{i,j(lama)} - \alpha(t) \cdot \left(x_{ij} - c_{i,j(lama)}\right) \quad (2.5)$$

     Keterangan :

     $(baru)$: Bobot baru terhadap pembaharuan yang dilakukan.

     $(lama)$: Bobot lama hasil pembaharuan sebelumnya.

6. Pengurangan *Learning Rate* (α) , dengan rumus :

$$\alpha(t+1) = \alpha(t) - \frac{\alpha(0) - u}{K} \quad (2.6)$$

t : Berupa waktu (Menunjukan perubahan bobot pada satu waktu sesuai dengan banyaknya *input* yang masuk [5] atau bisa dilihat sebagai iterasi).

$\alpha(0)$: *Learning rate* awal yang telah ditentukan

$\alpha(t+1)$: *Learning rate* selanjutnya.

$\alpha(t)$: *Learning rate* sebelumnya

u : Nilai pengurangan *learning rate*

K : Jumlah Kelas

7. Penambahan *epoch*, dengan rumus :

$$Epoch = Epoch + 1 \qquad (2.7)$$

8. Setelah pembaharuan bobot dilakukan, maka diperoleh bobot akhir untuk c_w yang dapat digunakan untuk proses *testing*.

Testing LVQ

1. Inisialisasi *Input Vector* sebagai $x_{i,j}$.
2. Persiapkan bobot akhir yang telah diperoleh pada tahapan *training* sebagai $c_{i,j}$.
3. Tentukan nilai jarak minimum dengan *Euclidean Distance* [9] :

$$d_{i,j} = \sqrt{\sum_{j=1}^{J} \left(x_{i,j} - c_{i,j}\right)^2} \qquad (2.8)$$

$d_{i,j}$ = jarak *euclidean distance* vector i dengan banyak data j.

$\sum$ = Sigma atau jumlah dari perhitungan *input* vector dengan *codebook vector*.

$X_{i,j}$ = *Input Vector* ke-i,j

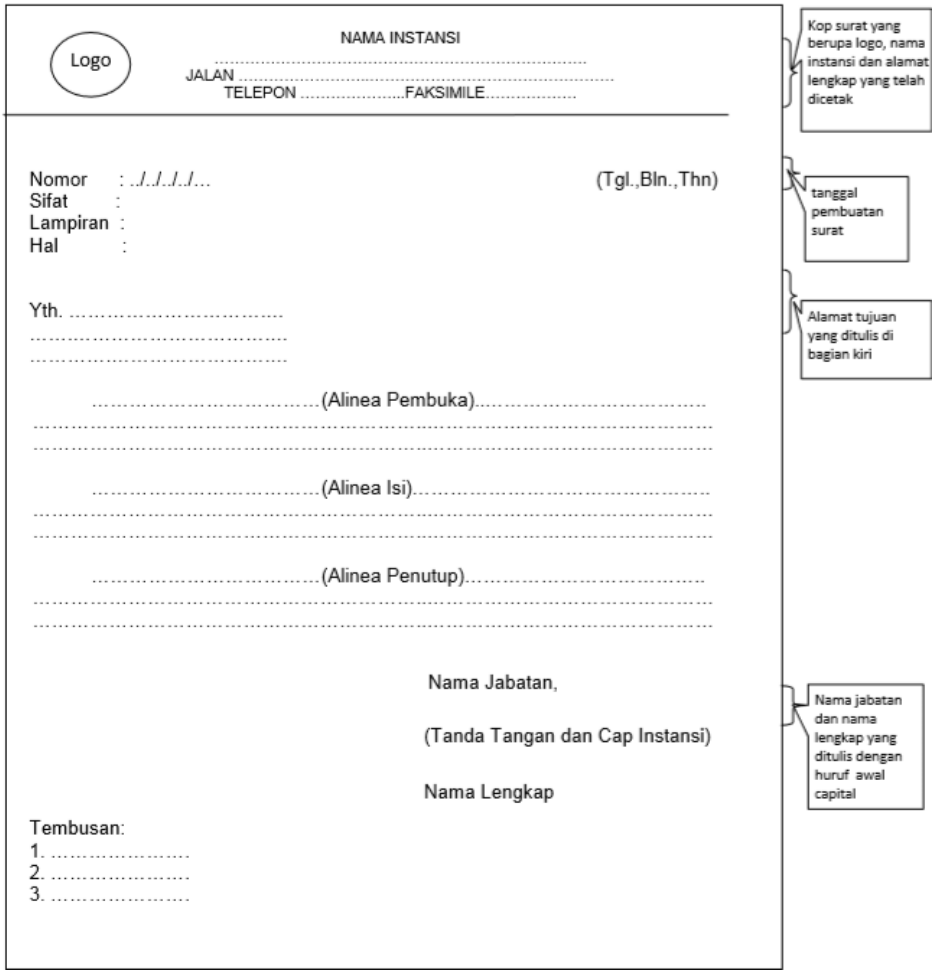
$C_{i,j}$ = *Codebook Vector* ke-i,j

4. Klasifikasikan nilai $d_{i,j}$ sebagai kelas dari $x_{i,j}$.

Perlu diketahui bahwa jika terdapat nilai minimum yang sama pada masing – masing bobotnya, maka untuk nilai minimum pertama kali yang diperoleh itulah yang menang. Mengingat bahwa pada algoritma *LVQ* ini mempunyai konsep *competitive learning*.

2.2 Surat Masuk

Surat masuk adalah surat yang diterima oleh lembaga atau organisasi yang memiliki aturan sesuai penyusunan surat resmi. Berikut ini kerangka bagian yang terdapat pada surat masuk [4]:



Gambar 2.2 Kerangka Surat Masuk

Berikut ini penjelasan mengenai kerangka pada surat masuk :

1. Kepala Surat

   Kop surat berfungsi sebagai identitas dari instansi dari pembuat surat. Komponen kepala surat terdiri dari logo, nama instansi, alamat instansi lengkap dan email.

2. Tanggal Surat

Tanggal surat bertujuan untuk mencatat informasi waktu dibuatnya surat.

3. Nomor Surat

   Komponen nomor surat terdiri dari kode surat, nomer surat yang dikeluarkan pada instansi tersebut, identitas instansi, dan tahun dibuatnya surat tersebut. Adapun tujuan dari nomor surat adalah agar suatu instansi mengetahui jumlah surat yang telah dikeluarkan.

4. Lampiran

   Lampiran adalah berkas pendukung dari isi surat yang akan dilampirkan.

5. Perihal atau Hal

   Perihal bertujuan untuk menjelaskan tujuan atau apa yang diinformasikan surat tersebut.

6. Alamat Surat

   Alamat surat terdiri dari alamat tujuan pengiriman surat yang diikuti nama orang yang dituju beserta jabatannya dan nama instansinya.

7. Salam Pembuka

   Salam Pembuka bertujuan sebagai kalimat pembuka untuk menunjukkan rasa hormat dan sopan santun

8. Isi Surat

   a. Alinea pembuka

      Pada alinea pembuka biasanya menjelaskan dari alasan mengirim surat tersebut, dan dapat juga langsung menyampaikan maksud dari surat tersebut.

   b. Alinea isi

      Pada alinea isi menjelaskan isi pokok dari surat tersebut, biasanya berisi informasi yang dimaksud agar informasi utama tersampaikan.

   c. Alinea penutup

      Pada alinea penutup berisi ucapan terimakasih, harapan, atau penggabungan ucapan terimakasi dan harapan.

9. Jabatan Penandatangan Surat

Bagian ini berfungsi untuk mengetahui jabatan penandatangan surat atau pihak yang bertanggungjawab terhadap isi surat.

10. Tanda tangan dan cap surat

    Tanda tangan dan cap surat berfungsi untuk memberikan kesan resmi dari suatu organisasi.

11. Tembusan

    Tembusan berfungsi untuk mengetahui kepada siapa saja surat disampaikan, jika surat ditembuskan kepada banyak pihak maka diurutkan sesuai tingkatannya atau jabatannya.

2.3 Ekstraksi Informasi

Ekstraksi informasi adalah suatu proses untuk mengubah informasi tidak terstruktur yang terdapat dalam teks ke dalam data terstruktur [1]. Ekstraksi Informasi merujuk pada ekstraksi otomatis dari informasi terstruktur seperti entitas, hubungan antar entitas, dan atribut deskripsi entitas dari sumber yang tidak terstruktur [14]. Ekstraksi informasi kegiatan yang berkaitan dengan pemrosesan teks bahasa manusia melalui pemrosesan bahasa alami (NLP). Ekstraksi informasi berperan sebagai sistem yang akan mengenali data yang tidak terstruktur yang memiliki informasi yang belum dikategorikan dan belum memiliki arti yang spesifik. Tidak terstruktur bukan berarti secara data yang membingungkan atau tidak jelas dengan artian bahwa informasi yang terkandung didalamnya tidak dapat langsung diterjemahkan oleh komputer sehingga membutuhkan sistem yang dinamakan ekstraksi informasi.

2.4 Tokenisasi

Tokenisasi adalah proses pemisahan suatu karakter atau kalimat berdasarkan spasi, dan mungkin pada waktu yang bersamaan dilakukan juga proses penghapusan karakter tertentu, seperti tanda baca. [15]

2.5 Ekstraksi Fitur

Ekstraksi fitur merupakan proses untuk mencari nilai-nilai fitur yang terkandung dalam dokumen [16]. 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 [17]. Berikut ini contoh ekstraksi fitur dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh Ekstraksi Fitur

No	Ekstraksi Fitur	Keterangan
1	*INITCAPS*	Mengenali setiap token yang hurufnya diawali dengan kapital.
2	*EMAIL*	Mengenali token yang memiliki email
3	*CONTAINDASH*	Mengenali token yang memiliki sedikitnya 1 strip
4	*CONTAINMORESLASH*	Mengenali token yang memiliki 1 garis miring
5	*CONTAINDATE*	Mengenali token yang memiliki tanggal
6	*ALLCAPS*	Mengenali setiap token yang semua hurufnya kapital.
7	*CONTAINSDIGIT*	Mengenali setiap token yang mengandung digit.
8	*ALLDIGIT*	Mengenali setiap token yang semuanya digit.
9	*CONTAINSDOTS*	Mengenali setiap token yang mengandung titik.
10	*LOWERCASE*	Mengenali setiap token yang semuanya huruf kecil.
11	*PUNTUATION*	Mengenali setiap token yang mengandung tanda tertentu seperti titik, koma, titik dua, titik koma, tanda kurung, dan tanda seru.

2.6 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 [18].

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*

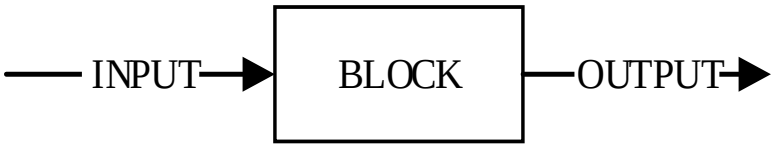
   Algoritma yang digambarkan dalam bentuk diagram alir. Diagram alir merupakan aliran yang berlangsung ketika suatu bagian (prosedur) dalam program dieksekusi.

2.7 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.*

2.7.1 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 *output* [19].



Gambar 2.3 Blok Diagram

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

2.7.2 Diagram Konteks

Diagram konteks merupakan data *flow* diagram yang menggambarkan garis besar operasional sistem. Konteks diagram menggambarkan hubungan sistem dengan entitas-entitas di luar sistem. Diagram konteks memperlihatkan sistem sebuah proses. Tujuannya adalah memberikan pandangan umum sistem. Diagram Konteks memperlihatkan sebuah proses yang berinteraksi dengan lingkungan luarnya. Ada pihak luar yang memberikan masukan dan pihak yang menerima keluaran sistem [20].

2.7.3 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. Terdapat 4 simbol utama pada *DFD* yang terdiri dari [21]:

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 *Learning Vector Quantization*.

2.7.4 *Flow Chart*

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 [22]. *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.

2.8 *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*" [23].

2.9 *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 [24].

2.10 *I Love PDF*

I Love PDF adalah *tools* untuk membantu pengeditan *PDF* menjadi lebih mudah. Berdiri pada tahun 2010 dan terletak di Barcelona, *tools* ini memiliki layanan yang gratis, mudah diakses dan memiliki kualitas yang baik dalam mengelola *file PDF*. *I Love PDF* menyediakan aplikasi berbasis *mobile* dan *desktop*. *I Love PDF Mobile App* berfungsi untuk memodifikasi *file PDF* di kapan saja dan dimana saja, sedangkan *i Love PDF Desktop* berfungsi untuk memproses *file* secara *offline*.

2.11 *ABBY Finereaderonline*

ABBYY FineReader adalah sebuah *software* untuk merubah hasil *scan* gambar menjadi *text* agar dokumen dapat diedit kembali.

Dengan *ABBYY FineReader* dengan mudah:

1. Konversi dokumen hanya dengan menekan satu tombol.
2. Merubah hasil *scan* dokumen cetak dan gambar kedalam format *Microsoft word, Excel, Searchable PDF*, & Format Lainya.
3. Mengenali tulisan lebih dari 183 bahasa.

2.12 *CSV*

Comma Separated Values (CSV) adalah format dasar yang terdiri dari file teks yang berisi *line* dan *value* [25]. 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* sebagai tempat penyimpanan data. Data *training* yang disimpan pada file *CSV,*

memiliki tujuan agar data dapat tersusun dengan rapih terhadap banyaknya data yang digunakan.

2.13 Metode Pengujian *Black Box*

Pengujian adalah satu set aktivitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan [26]. *Black Box Testing* berfokus pada spesifikasi fungsional dari perangkat lunak. Pengujian Black Box cenderung untuk menemukan hal-hal berikut :

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (*interface errors*).
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi (*performance errors*). Kesalahan inisialisasi dan terminasi.

2.14 Akurasi

Berikut rumus perhitungan Nilai akurasi [27].

$$Akurasi\ (\%) = \frac{Jumlah\ data\ terklasifikasi\ \ benar}{Keseluruhan\ testing\ data}\ x\ 100\%$$

$$Error\ (\%) = \frac{jumlah\ data\ terklasifikasi\ \ salah}{Keseluruhan\ testing\ data}\ x\ 100\%$$

Nilai masukan untuk uji akurasi dan error adalah data testing yang telah terklasifikasi benar untuk akurasi, sendangkan salah untuk perhitungan error.