BAB 2
LANDASAN TEORI

2.1 Twitter

Twitter adalah jejaring sosial dan *microblogging service* yang memungkinkan penggunanya untuk mengirim 140 karakter pesan teks yang disebut "*tweet".* Pengguna yang terdaftar dapat membaca dan mengirim *tweet.* Tetapi pengguna yang tidak terdaftar hanya dapat membaca *tweet.* Pengguna mengakses twitter melalui antarmuka *website*, SMS, atau aplikasi perangkat mobile. Twitter memiliki popularitas yang mendunia dengan 29,5 juta pengguna yang terdaftar pada tahun 2012 yang mengirim sebanyak 340 juta *tweet* per hari [5]

2.2 Partai Politik

Pemilihan umum 2014 dibagi menjadi dua putaran yaitu PEMILU legislatif yang dilaksanakan tanggal 9 April 2014 dan PEMILU Presiden yang diadakan tanggal 9 juli 2014. PEMILU 2014 diikuti oleh 12 peserta partai politik yang disaring dari 46 partai politik yang mendaftar. 12 partai politik peserta pemilu yaitu Partai Nasional Demokrat, Partai Kebangkitan Bangsa, Partai Keadilan Sejahtera, Partai Demokrasi Indonesia Perjuangan, Partai Golongan Karya, Partai Gerakan Indonesia Raya, Partai Demokrat, Partai Amanat Nasional, Partai Persatuan Pembangunan, Partai Hati Nurani Rakyat, Partai Bulan Bintang, dan Partai Keadilan dan Persatuan Indonesia.

2.3 Text Mining

Teks Mining merupakan ilmu yang mengkombinasikan teknik dari *data mining, machine learning, natural learning processing*, dan information retrieval. Salah satu contoh implementasi text mining termasuk mengklasifikasikan dokumen ke dalam topik yang spesifik atau *supervised learning*, mengelompokan dokumen yang mempunyai kemiripan atau unsupervised learning, dan diimplementasikan ke dalam sistem perncari dokumen atau *information retirieval*.

Text data mining fokus terhadap metode *data mining* yang diaplikasikan pada tekstual data. Beberapa bagian dari *text mining* adalah *term, corpus, lexicon*, dan dokumen. *Term* adalah kata atau kombinasi kata atau prasa. Dokumen adalah sekumpulan kata dan tanda baca, dokumen dapat terdiri dari beberapa segmen teks dan memiliki panjang yang tidak ditentukan. Contoh dari dokumen adalah kalimat, paragraf, bagian sebuah buku atau bab, buku, halaman web, email dan lain-lain. *Corpus* adalah koleksi dokumen. *Lexicon* adalah sekumpulan kata yang bersifat unik yang terdapat pada *corpus*[6].

2.4 Text Pre-Processing

Text Pre-Processing adalah serangkaian proses yang dilakukan untuk mempersiapkan data sebelum proses lebih lanjut. Tahapan ini memiliki tujuan untuk mengubah data agar dapat dilanjukan ke proses klasifikasi.

Terdapat beberapa proses yang harus dilakukan dalam tahap *text pre-processing* atau persiapan dokumen teks, yaitu *tokenization, casefolding, stemming.*

2.4.1 Tokenization

Tokenization merupakan proses memecah suatu kalimat dalam seluruh isi dokumen menjadi sekumpulan kata atau token yang dipisahkan oleh spasi. Hasil dari proses *tokenization* selanjutnya digunakakan pada proses selanjutnya seperti *casefolding* dan *stemming*.



Gambar 2.1 Tokenization

2.4.2 Casefolding

Case Folding, merupakan proses mengubah semua huruf yang ada pada seluruh dokumen menjadi huruf kecil. Huruf "a" sampai dengan "z"yang akan diproses, karakter selain huruf akan dianggap *delimiter*.



Gambar 2.2 Casefolding

2.4.3 Stemming

Stemming adalah salah satu cara untuk meningkatkan performa dalam *text processing* dengan cara mentransformasi kata-kata dalam sebuah dokumen teks ke kata dasarnya. Algoritma *stemming* untuk masing-masing bahasa berbeda-beda, sebagai contoh bahasa Inggris memiliki morfologi yang berbeda dengan bahasa Indonesia sehingga algoritma *stemming* untuk kedua bahasa tersebut juga berbeda [7].

Stemming menggunakan aturan-aturan tertentu sebagai contoh kata bersama, kebersamaan dan menyamai mempunyai *root word* yang sama yaitu "sama". Pada teks berbahasa Inggris, proses yang perlu dilakukan hanya proses menghilangkan suffiks, sedangkan pada teks berbahasa Indonesia, selain suffiks, prefiks dan konfiks juga dihilangkan. Salah satu algoritma untuk *stemming* pada teks berbahasa Indonesia adalah algoritma nazief & adriani yang dibuat oleh Bobby Nazief dan Mirna Adriani. Algoritma tersebut memiliki tahap-tahap sebagai berikut:

1. Cari kata yang akan di-stem dalam kamus, jika ditemukan maka diasumsikan bahwa kata tersebut adalah *root word.*

2. Penghapusan *Inflexion Suffix*, (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Cek kata dalam kamus, jika kata ditemukan maka diasumsikan bahwa kata tersebut adalah *root word.*
3. Penghapusan *derivation suffiks* (“-i”, “-an”, “kan”). Jika kata ditemukan di kamus, maka algoritma berhenti, jika tidak maka lanjutkan ke langkah 3a.
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k” , maka “-k” juga dihapus. Jika kata ditemukan dalam kamus maka algoritma berhenti. Jika tidak maka lanjutkan ke langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an”, atau “-kan”). Dikembalikan, lanjut ke langkah 4.
4. Hapus *derivation prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak maka lanjut ke langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak lanjutkan ke langkah 4b.
   b. Tentukan tipe awalan kemudian hapus awalan. Jika root word belum juga ditemukan lakukan langkah 5.
5. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Pada tahap ke empat langkah-langkah yang dilakukan proses *stemming*, terdapat pemeriksaan kombinasi awalan dan akhiran yang tidak dizinkan. Tabel II.1 menunjukan kombinasi awalan akhiran yang tidak diizinkan.

Tabel 2.1 Kombinasi Awalan Akhiran

Awalan	Akhiran
be-	-i
di-	-an
ke-	-i,-kan
me-	-an
se-	-i,-kan
te-	-an

Beberapa contoh kata yang kata berimbuhan yang akan dilakukan proses *stemming:*

1. Awalan Se-
    a. Sebungkus = se + bungkus
    b. Searah = se + arah
2. Awalan Me-
    a. Mengasuh = me + asuh
    b. Memberi = me + beri
    c. Mencuci = me + cuci
    d. Mendidik = me + didik
    e. Menghukum = me + hukum
3. Awalan Pe-
    a. Perhitungan = Pe + Hitung + An
    b. Pergelaran = Pe + Gelar + An

2.5 Klasifikasi

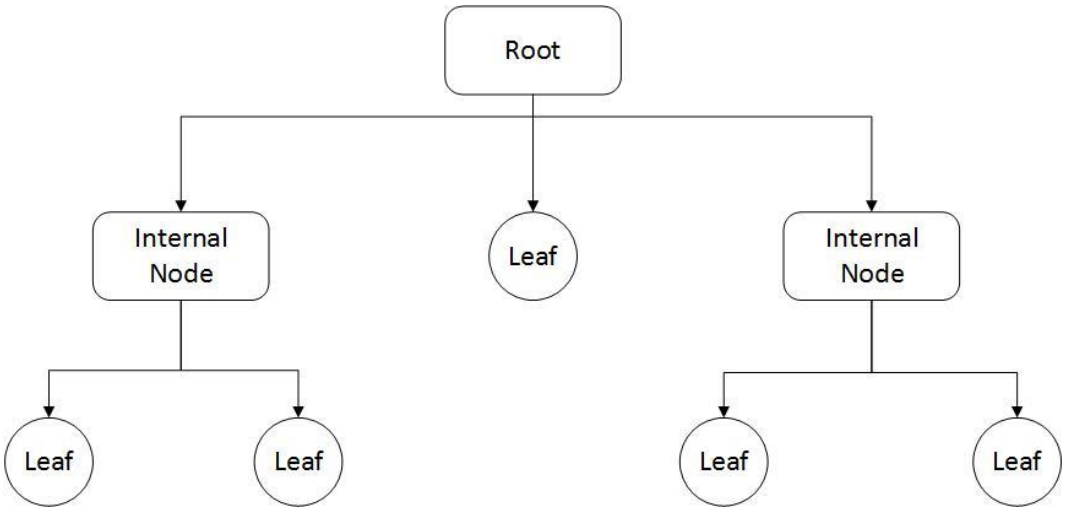
Salah satu aplikasi dari data mining adalah klasifikasi. Menggunakan sampel data sebelumnya yang telah memiliki kategori untuk dijadikan data training, tujuannya adalah untuk memberikan kategori pada data baru yang belum dikategorikan. Klasifikasi pada teks melibatkan banyak teknik termasuk *indexing* pada *information retieval* dan teknik dari *machine learning* untuk menentukan kategori secara otomatis dari data training yang sudah diberi label. Pada proses klasifikasi sebuah dokumen, tidak digunakan informasi lain selain konten dari dokumen tersebut.

Banyak teknik *machine learning* yang digunakan untuk klasifikasi diantaranya adalah *Naïve Bayesian*, *Decision Tree*, k-*Nearest Neighbor* (k-NN), *Neural Network*(NN), dan *Support Vector Machines*(SVM). Teknik teknik tersebut berbeda dalam mekanisme pembelajaran dan representasi model yang dipelajari [8].

2.6 Decision Tree

Decision Tree atau pohon keputusan adalah model yang dikembangkan untuk membantu mencari dan membuat keputusan untuk masalah yang akan dipecahkan dengan memperhitungkan berbagai macam faktor yang ada di dalam lingkup masalah tersebut. Mekanisme dari pohon keputusan seperti struktur pohon, dimana tiap *internal node* menunjukan sebuah test pada sebuah attribut, setiap cabang menunjukan hasil dari test, dan *leaf node* menunjukan kelas atau kategori. Pada *decision tree* terapat terdapat tiga jenis *node*, yaitu:

a) *Root Node*, merupakan *node* teratas atau akar, pada *node* ini tidak memiliki *parent* dan mempunyai *child* lebih dari satu.
b) *Internal Node*, merupakan node percabangan, memiliki *parent* dan minimal dua *child*.
c) *Leaf Node*, merupakan *node* akhir, pada *node* ini memiliki *parent* dan tidak memiliki *child.*



Gambar 2.3 Model Pohon Keputusan

Setiap percabangan atau internal *node* menyatakan kondisi yang harus dipenuhi dan tiap ujung pohon atau *leaf node* menyatakan kelas atau kategori dari setiap sampel pada training set.

2.7 Algoritma C4.5

Algoritma C4.5 dibangun oleh Ross Quinlan merupakan ekstensi atau pengembangan dari algortima ID3. Algoritma ini merupakan kelompok algoritma

pohon keputusan. Mempunyai masukan berupa data training. Data training berupa data contoh yang telah mempunyai label kelas positif atau negatif, akan digunakan untuk membangun sebuah *tree*. Sedangkan atribut-atribut data digunakan sebagai parameter dalam proses klasifikasi data.

2.7.1 Perhitungan Information Gain

Perhitungan *information gain* dilakukan untuk *attribute selection measure* yang digunakan untuk memilih atribut pada setiap simpul pada pohon keputusan. Atribut dengan *information gain* tertinggi atau nilai pengurangan *entropy* yang terbesar dipilih sebagai tes atribut pada simpul. Untuk menghitung *gain* digunakan rumus seperti tertera dalam persamaan (1)

$$Gain(S,A) = Entropy(S) - \left(\frac{S_0}{S} * Entropy(S_0) + \frac{S_1}{S} * Entropy(S_1) \right) \quad (1)$$

Keterangan :

S : Jumlah seluruh kasus

S_0 : Jumlah kasus dengan nilai 0

S_1 : Jumlah kasus dengan nilai 1

A : atribut

Perhitungan nilai *entropy* dapat dilihat pada persamaan (2).

$$Entropy(S) = -\frac{Pos(S)}{S} \log_2 \frac{Pos(S)}{S} - \frac{Neg(S)}{S} \log_2 \frac{Neg(S)}{S} \quad (2)$$

Keterangan :

S : Jumlah seluruh kasus

Neg(S) : Jumlah kasus dengan kelas negatif

Pos(S) : Jumlah kasus dengan kelas positif

2.7.2 Pembentukan Pohon Keputusan

Algoritma C4.5 membangun pohon keputusan dari training set yang telah ditentukan menggunakan konsep information *entropy*. Secara umum langkah-langkah algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut:

1. Terdapat masukan berupa training set yang setiap sampelnya telah diberi kelas atau kategori.
2. Jika seluruh sampel pada training set memiliki kelas yang sama maka pohon keputusan akan memiliki satu *node* berupa *leaf node* yang diberi label kelas yang terdapat pada semua sampel dalam training set.
3. Jika seluruh sampel tidak dalam satu kelas yang sama maka akan dicari *gain* tertinggi dari seluruh atribut untuk memilih atribut yang paling berpengaruh pada *training set*, dan akan dijadikan atribut penguji pada *node* tersebut.
4. Jumlah cabang setiap *node* dibangun berdasarkan partisi nilai dari atribut pengujian. Jika ada partisi nilai yang memiliki nilai entropy nol, maka cabang dari partisi nilai tersebut menjadi *leaf node* yang diisi kelas yang memiliki jumlah kemunculan tertinggi pada training data.
5. Jika ada cabang dari *node* yang dibentuk pada langkah sebelumnya belum mencapai *leaf node,* maka akan dicari nilai gain seperti pada langkah nomor 3 dimulai dari cabang paling kiri yang belum mencapai *leaf node*.
6. Jika seluruh cabang dari *node* yang dibentuk pada langkah sebelumnya sebelumnya telah mencapai *leaf node,* maka akan dicek cabang dari *node* diatas dari *node* yang dibentuk pada langkah sebelumnya, jika cabang tersebut belum mencapai *leaf node* maka akan dicari nilai *gain* seperti pada langkah nomor 3.
7. Proses yang sama akan dilakukan secara rekursif untuk membentuk pohon keputusan dari setiap sampel.
8. Proses rekursif akan berhenti jika semua sampel pada *node* memiliki kelas yang sama, semua simpul sudah mencapai *leaf node* atau semua atribut telah digunakan untuk mempartisi sampel.

   Berikut ini adalah algoritma pembentuk pohon.

Algoritma : Generate_decision_tree
Narasi : Membuat pohon keputusan
Masukan : Sampel data pelatihan, *samples*, yang dipresentasikan dengan atribut bernilai diskret, himpunan kandidat atribut, atribut list
Keluaran : Pohon Keputusan
Metode :
(1) Buat simpul *N*;
(2) if *samples* semua memiliki kelas yang sama, C, then
(3) return *N* sebagai simpul daun dengan label kelas C;
(4) if *attribute-list* kosong then
(5) return *N* sebagai simpul daun dengan label kelas terbanyak di *samples*
(6) pilih *test-atribut,* yaitu salah satu atribut dari *atribute-list* dengan *gain-ratio* terbesar;
(7) beri label pada simpul *N* dengan *test-attribute*;
(8) for setiap nilai ?? pada *test-atribut;*
(9) tambahkan cabang pada simpul N untuk kondisi *test-attribut* = a;
(10) buat partisi sampel ?? dari *samples* dimana *test-attribut* = a;
(11) if ?? kosong then
(12) tempelkan daun yang diberi label dengan kelas terbanyak di *samples*;
(13) else tempelkan simpul yang dibuat oleh Generate_decision_tree (??, *attribut-list- test-attribute*);

Gambar 2.4 Algoritma Pembentukan Pohon

Berikut ini adalah contoh proses pembentukan pohon keputusan menggunakan algoritma C4.5 menggunakan empat kalimat opini yang diberi kelas negatif dan positif mengenai PEMILU.

Tabel 2.2 Contoh Training Data

No.	**Kalimat Opini**	**Kelas**
1	Pemilu gagal	Negatif
2	Pemilu gagal dicurangi	Positif
3	Pemilu sukses	Positif
4	Pemilu dicurangi	Negatif

Proses perhitungan gain masing-masing atribut terhadap training data dilakukan untuk menentukan atribut penguji pada setiap simpul. Didapatkan empat kata yang akan dijadikan atribut penguji setelah dilakukan proses *text preprocessing* yaitu “pemilu”, “curang”, “sukses”, dan “gagal”. Selanjutnya setiap sampel dimasukan kedalam sebuah tabel dengan keempat atribut penguji, atribut penguji bernilai 1 jika sampel mengadung kata pada atribut penguji dan bernilai 0 jika tidak mengandung kata pada atribut penguji. Berikut ini tabel yang dihasilkan

dari training data dan proses perhitungan information gain terhadap semua atribut penguji.

Tabel 2.3 Nilai Training Data

No	curang	pemilu	gagal	sukses	kelas
1	0	1	1	0	Negatif
2	1	1	1	0	Negatif
3	0	1	0	1	Positif
4	1	1	0	0	Negatif

Perhitungan entropy untuk seluruh training data :

$Entropy(Total) = -\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4} = 1$

Perhitungan information gain pada setiap atribut penguji untuk menentukan simpul akar :

1. curang

   Perhitungan entropy atribut "curang" dengan nilai 0 dan 1:

   $Entropy(0) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$

   $Entropy(1) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$

   $Gain(Total, curang) = 1 - ((2/4)Entropy(0) + (2/4)\ Entropy(1))$

   $= 0$

2. pemilu

   Perhitungan entropy atribut "pemilu" dengan nilai 0 dan 1:

   $Entropy(0) = -0 - 0 = 0$

   $Entropy(1) = -\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4} = 1$

   $Gain(Total, pemilu) = 1 - ((0/4)Entropy(0) + (4/4)\ Entropy(1))$

   $= 0$

3. gagal

   Perhitungan entropy atribut "gagal" dengan nilai 0 dan 1:

   $Entropy(0) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$

$$Entropy(1) = -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2} = 1$$

$$Gain(Total, gagal) = 1 - ((2/4)Entropy(0) + (2/4)\ Entropy(1))$$

$$= 0$$

4. sukses

Perhitungan entropy atribut "sukses" dengan nilai 0 dan 1:

$$Entropy(0) = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3} = 0.918$$

$$Entropy(1) = -\frac{1}{1}log_2\frac{1}{1} - 0 = 0$$

$$Gain(Total, sukses) = 1 - ((3/4)Entropy(0) + (1/4)\ Entropy(1))$$

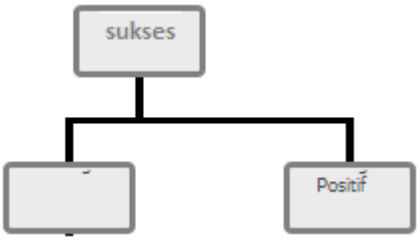
$$= 0.312$$

Proses perhitungan dilakukan sampai ditentukan simpul daun. Seluruh hasil perhitungan nilai gain untuk membentuk simpul akar pohon keputusan terdapat pada Tabel 2.4,

Tabel 2.4 Perhitungan Information Gain Untuk Simpul Akar

Perhitungan Node Akar Total Entropy : 1							
Attribut	Neg(0)	Pos(0)	Entropy(0)	Neg(1)	Pos(1)	Entropy(1)	Gain
pemilu	0	0	0	2	2	1	0
gagal	1	1	1	1	1	1	0
curang	1	1	1	1	1	1	0
sukses	2	1	0.918	0	1	0	0.312
Max Gain : 0.312 Node : sukses							

Hasil perhitungan nilai gain untuk menentukan simpul akar akan membentuk *rule tree* seperti pada Gambar II.4.



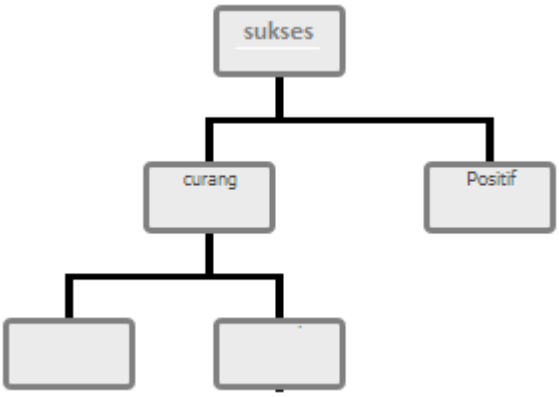
Gambar 2.5 Hasil Perhitungan Nilai Gain Simpul Akar

Karena nilai entropy untuk cabang simpul “sukses” dengan nilai 0 masih tidak sama dengan 0, maka dilakukan perhitungan kembali untuk menentukan simpul berikutnya.

Tabel 2.5 Perhitungan Information Gain Untuk Cabang Simpul Sukses Nilai 0

Perhitungan Cabang Node 'sukses' Value '0' Total Entropy : 0.918							
Attribut	Neg(0)	Pos(0)	Entropy(0)	Neg(1)	Pos(1)	Entropy(1)	Gain
pemilu	0	0	0	2	1	0.918	0
gagal	1	0	0	1	1	1	0.251
curang	1	0	0	1	1	1	0.251
Max Gain : 0.251 Node : curang							

Hasil perhitungan nilai gain untuk menentukan cabang simpul sukses dengan nilai 0 akan membentuk *rule tree* seperti pada Gambar II.5.



Gambar 2.6 Hasil Perhitungan Nilai Gain Cabang Simpul Sukses Nilai 0

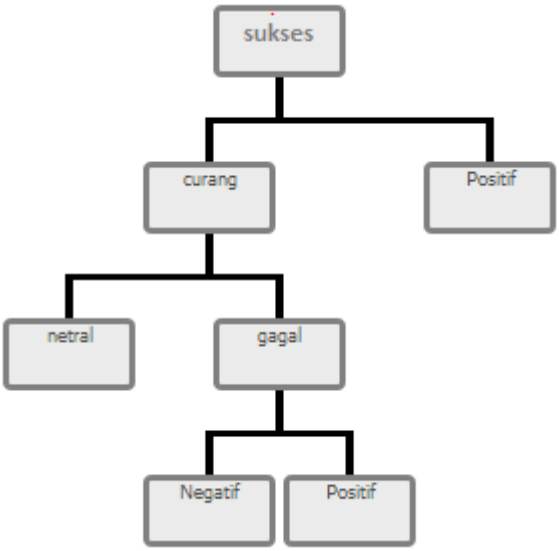
Karena nilai entropy untuk cabang simpul “curang” dengan nilai 1 masih tidak sama dengan 0, maka dilakukan perhitungan kembali untuk menentukan simpul berikutnya.

Tabel 2.6 Perhitungan Information Gain Untuk Cabang Simpul Curang Nilai 1

Perhitungan Cabang Node 'curang' Value '1' Total Entropy : 1							
Attribut	Neg(0)	Pos(0)	Entropy(0)	Neg(1)	Pos(1)	Entropy(1)	Gain
pemilu	0	0	0	1	1	1	0
gagal	1	0	0	0	1	0	1

Max Gain : 1 Node : gagal

Hasil perhitungan nilai gain untuk menentukan cabang simpul ‘curang’ dengan nilai 1 akan membentuk *rule tree* seperti pada Gambar II.6.



Gambar 2.7 Hasil Perhitungan Nilai Gain Cabang Simpul Curang Nilai 1

Kedua nilai entropy untuk cabang simpul “gagal” bernilai 0, maka kedua cabang atribut “gagal” *berupa leaf node* dan tidak dilakukan perhitungan nilai gain untuk menentukan simpul berikutnya.

Pohon keputusan yang dibangun oleh algoritma C4.5 ini bergantung pada training data yang disiapkan. *Tweet* yang akan diklasifikasikan akan diuji dimulai dari simpul teratas atau simpul akar. Jika sebuah *tweet* mengandung kata yang menjadi atribut penguji pada simpul akar yaitu ‘sukses’ maka kata atau simpul selanjutnya yang akan diuji adalah simpul ruas kanan, jika *tweet* tidak mengandung kata ‘sukses’ maka simpul yang diuji selanjutnya adalah simpul ruas kiri, proses ini akan terus dilakukan sampai ditemukan simpul daun yang berisi kelas sentimen negatif, positif dan netral. Jika sebuah *tweet* mengandung kata yang tidak diuji pada jalur yang sedang ditelusuri, maka kata tersebut tidak akan diuji.

2.7.3. Klasifikasi Menggunakan Pohon Keputusan

Model pohon keputusan yang telah dibentuk menggunakan algoritma C4.5 selanjutnya digunakan untuk mengklasifikasikan sebuah kalimat opini. Kata-kata

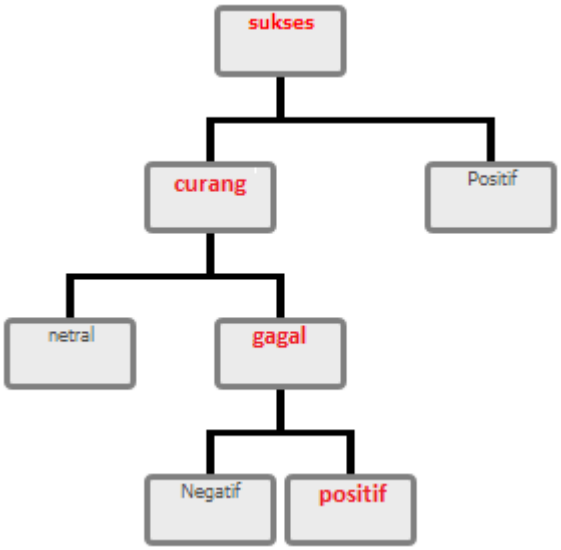
yang akan diuji adalah kata-kata yang menjadi atribut penguji pada pohon keputusan, jika pada kalimat opini terdapat kata yang tidak menjadi atribut penguji pada pohon keputusan maka kata tersebut tidak akan diuji. Jika pada kalimat terdapat kata yang menjadi atribut penguji simpul yang sedang diuji maka simpul berikutnya yang diuji adalah simpul ruas kanan dan seluruh simpul ruas kiri tidak akan diuji begitu juga sebaliknya. Berikut adalah langkah-langkah klasifikasi menggunakan pohon keputusan :

1. Simpul akar akan menjadi simpul pertama yang akan diuji,
2. jika pada kalimat terdapat kata yang menjadi atribut penguji pada simpul maka selanjutnya akan diuji simpul pada ruas kanan yang merupakan cabang dari simpul yang diuji sebelumnya. Jika pada kalimat tidak terdapat kata yang menjadi atribut penguji pada simpul tersebut maka selanjutnya akan diuji simpul pada ruas kiri yang merupakan cabang dari simpul yang diuji sebelumnya.
3. Jika pada simpul yang diuji bukan merupakan *leaf node* maka akan dilakukan langkah sebelumnya yaitu langkah 2 pada simpul yang sedang diuji.
4. Proses akan berhenti jika simpul yang diuji merupakan *leaf node*.

Selanjutnya adalah contoh klasifikasi sebuah kalimat menggunakan pohon keputusan yang telah dibentuk.

Kalimat Opini : *"Pemilu gagal dicurangi"*

Simpul pertama yang akan dilakukan pengecekan adalah simpul kata "sukses" yang tidak terdapat pada kalimat maka simpul selanjutnya yang dilakukan pengecekan adalah simpul pada ruas kiri yaitu curang. Kata "curang" terdapat pada kalimat sehingga pengecekan selanjutnya dilakukan pada simpul ruas kanan yaitu "gagal". Kata "gagal" terdapat pada kalimat sehingga pengecekan selanjutnya dilakukan pada simpul ruas kanan yaitu "Positif". Karena pengecekan simpul telah mencapai simpul daun, maka pengecekan dihentikan dan menghasilkan "Positif" sebagai kelas dari kalimat yang diuji. Berikut ini adalah gambar pengecekan simpul pada proses klasifikasi.



Gambar 2.8 Proses Pengecekan Kalimat Opini

2.8 Cross Validation

K-fold *cross-validation* adalah sebuah pengujian dimana *dataset* X dibagi secara acak kedalam K bagian dengan ukuran yang sama, Xi, i=1,...,K. Untuk memproses tiap pasangnya, diambil terlebih dahulu salah satu bagian dari K bagian sebagai data tes, dan mengkombinasikan sisanya sebanyak K-1 bagian sebagai data latih. Proses pengujian dilakukan sebanyak K kali [13].

Dari pengujian yang dilakukan, dievaluasi dengan menggunakan *confusion* matrix untuk mengetahui tingkat kebenaran dari penelitian.

Tabel 2.7 *Confusion Matrix*

	Tested Class		
Predicted Class	Positive	Negative	Total
Positive Negative	True Positive False Positive	False Negative True Negative	P *n*
Total	*p'*	*n'*	N

Keterangan

True Positive : Hasil diprediksi bernilai *True*, dan pengujian menghasilkan *True*

False Positive : Hasil diprediksi bernilai *False*, dan pengujian menghasilkan *True*

False Negative : Hasil diprediksi bernilai *True*, dan pengujian menghasilkan *False*

True Negative : Hasil diprediksi bernilai *False*, dan pengujian menghasilkan *False*.

Setelah hasil pengujian menggunakan k-fold cross validation dievaluasi menggunakan confusion matrix, selanjutnya dapat dihitung nilai error dari keseluruhan iterasi dengan persamaan berikut.

$$CV(a) = \frac{1}{K}\sum_{i=1}^{K} Error_i(f(x,a))$$

Keterangan

$CV(a)$: Nilai *error* pada pengujian

K : jumlah *fold*

$f(x,a)$: Jumlah error pada masing-masing iterasi

2.9 Basis Data

Basis data adalah kumpulan data yang saling berkaitan, berhubungan yang disimpan secara bersama-sama sedemikian rupa tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan. Data ini mengandung semua informasi untuk mendukung semua kebutuhan sistem.

Basis data merupakan salah satu komponen yang penting dalam sebuah sistem, karena merupakan basis dalam menyediakan informasi. Basis data menjadi penting karena munculnya beberapa masalah bila tidak menggunakan data terpusat, seperti adanya duplikasi data, hubungan antar data tidak jelas, organisasi data dan perbaharuan data menjadi rumit. Bahasa yang digunakan dalam basis data yaitu [9] :

1. DDL (*Data Definition Language*)

   Merupakan bahasa definisi data yang digunakan untuk membuat dan mengelola objek *database* seperti *database*, tabel, *view*. Adapun perintah yang dimiliki adalah :

   a. CREATE : Perintah untuk pembuatan tabel dan *database.*
   b. DROP : Perintah untuk penghapusan tabel dan *database.*

c. ALTER : Perintah untuk pengubahan struktur tabel yang telah dibuat, seperti menambahkan atribut, mengganti nama atribut ataupun merubah nama tabel.

2. DML (*Data Manipulation Language*)

Merupakan bahasa manipulasi data yang dilakukan setelah DDL, digunakan untuk memanipulasi data pada objek *database* seperti tabel yang telah dibuat sebelumnya, adapun perintah yang dimiliki DDL adalah :

a. INSERT : untuk melakukan penginputan atau pemasukan data pada tabel *database*.
b. UPDATE : : Untuk melakukan perubahan/peremajaan terhadap data yang ada pada tabel.
c. DELETE : Untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa *recordset*.

3. DCL (*Data Control Language*)

Merupakan bahasa yang digunakan untuk mengendalikan pengaksesan data pada *database*, adapun perintah yang dimiliki DCL adalah:

a. GRANT : Untuk memberikan hak akses atau izin oleh *administrator server* kepada pengguna.
b. REVOKE : Untuk menghilangkan atau mencabut hak akses yang telah diberikan kepada *user* oleh *administrator*.

Pengolahan basis data meliputi proses ketiga bahasa yang digunakan diatur dengan menggunakan perangkat sistem manajemen basis data (*Database Management Sistem* / DBMS).

2.10 Pemodelan Analisis

Model analisis merupakan representasi teknis yang pertama dari sistem. Pertama analisis terstruktur adalah metode pemodelan klasik, dan analisis berorientasi objek.

Analisis terstruktur merupakan salah satu teknis analisis yang menggunakan pendekatan berorientasi fungsi. Teknik ini mempunyai sekumpulan perangkat

yang memungkinkan analisis sistem mendefinisikan spesifikasi fungsional perangkat lunak secara terstruktur, yang dimaksud dengan perangkat analisis terstruktur adalah alat bantu pemodelan yang digunakan untuk menggambarkan hasil pelaksanaan analisis.

Entity-relational diagram (ERD) adalah notasi yang digunakan untuk melakukan aktivitas pemodelan data, sedangkan *data flow diagram* (DFD) memberikan informasi sebagai dasar bagi pemodelan fungsi

2.11 ERD

Entity-relational diagram (ERD) adalah model konseptual untuk mendeskripsikan data dari sebuah sebuah *domain*. Komponen utama dari ERD adalah entitas dan relasi yang menghubungkan entitas. ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Dengan ERD, model dapat diuji dengan mengabaikan proses yang dilakukan. *Entity-relational diagram* pertama kali dikembangkan oleh Peter Chen dalam sebuah penelitiannya pada tahun 1976. Untuk menggambarkan ERD digunakan beberapa notasi dan simbol. Pada dasarnya ada tiga simbol yang digunakan , yaitu [10] :

a. *Entity*

   Entiti merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dengan objek yang lain. Simbol dari entiti ini biasanya digambarkan dengan persegi panjang.

b. Atribut

   Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar atribut diwakili oleh simbol elips.

c. Hubugan / Relasi

   Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Relasi dapat digambarkan sebagai berikut :

   1. Satu ke satu (*One to one*)

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

2. Satu ke banyak (*One to many*)
   Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitas A.
3. Banyak ke banyak (*Many to many*)
   Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

2.12 DFD

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana 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.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan. Diagram aliran data disefinisikan sebagai berikut [10] :

"Model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil".

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan alir data, yaitu :

1. Kejelasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Menghubungkan sistem dengan pengguna melalui diagram alir data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah data dan proses yang diperlukan sudah ditetapkan.

DFD terdiri dari *context diagram* dan diagram rinci (DFD *leveled*), *context diagram* berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. DFD *leveled* menggambarkan sistem jaringan kerja antara fungsi yang berhubungan satu sama lain dengan aliran data penyimpanan data, model ini hanya memodelkan sistem dari sudut pandang fungsi.

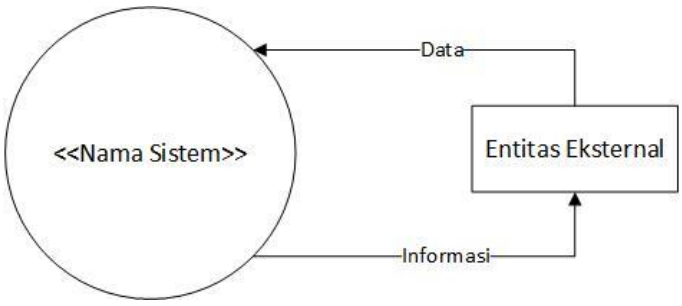
a. Penggambaran

Secara garis besar penggambaran DFD adalah sebagai berikut :

1. Buat Diagram Konteks

   Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Cara penggambarannya adalah :

   a. Tentukan nama sistemnya.
   b. Tentukan batasan sistemnya.
   c. Tentukan terminator apa saja yang ada dalam sistem.
   d. Tentukan apa yang diterima/diberikan terminator dari/pada sistem.
   e. Gambarkan diagram konteks.



Gambar 2.9 Contoh Diagram Konteks

2. Buat diagram level satu

   Diagram ini adalah dekomposisi dari diagram konteks. Cara penggambarannya adalah :

   a. Tentukan proses utama yang ada pada sistem.

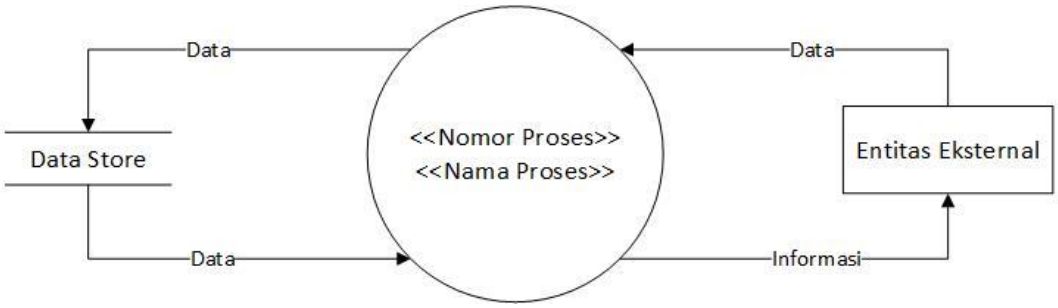
b. Tentukan apa yang diberikan/diterima masing-masing proses pada/dari sistem sambil memperhatikan konsep keseimbangan (alur data yang keluar/masuk dari suatu level harus sama dengan alur data yang masuk/keluar pada level berikutnya).

c. Apabila diperlukan, munculkan data store (master) sebagai sumber maupun tujuan alur data.

d. Gambarkan diagram level satu.

e. Hindari perpotongan arus data.

f. Beri nomor pada proses utama (nomor tidak menunjukkan urutan proses).



Gambar 2.10 Contoh DFD Level 1

3. Buat diagram level dua

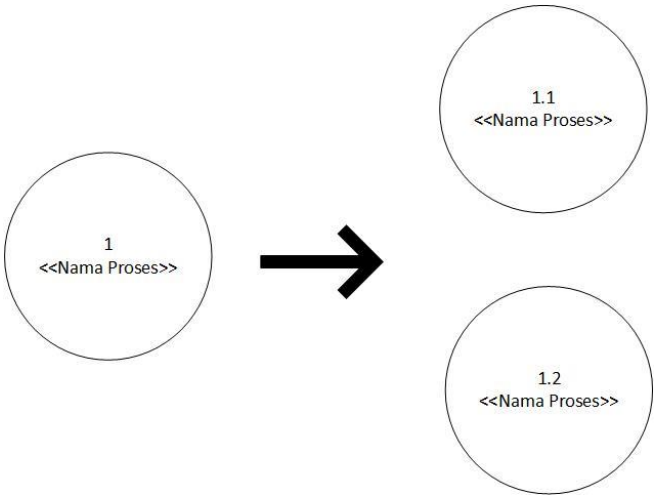
Diagram ini merupakan dekomposisi dari diagram level satu. Cara penggambarannya adalah :

a. Tentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level satu.

b. Tentukan apa yang diberikan/diterima masing-masing sub-proses pada/dari sistem dan perhatikan konsep keseimbangan.

c. Apabila diperlukan, munculkan data *store* (transaksi) sebagai sumber maupun tujuan alur data.

d. Hindari perpotongan arus data dan beri nomor pada masing-masing sub-proses yang menunjukkan dekomposisi dari proses sebelumnya. Contoh : 1.1, 1.2, 2.1.



Gambar 2.11 Penomoran DFD Level Berikutnya

4. DFD level tiga, empat, ..

   Diagram ini merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.

b. Elemen Dasar

1. Entitas Luar (*External Entity*)

   Sesuatu yang berada diluar sistem, tetapi memberikan data kedalam sistem atau memberikan data dari sistem, disimbolkan dengan suatu kotak notasi. *External Entity* tidak termasuk bagian dari sistem. Bila sistem informasi dirancang untuk satu bagian maka bagian lain yang masih terkait menjadi *external entity*.

2. Arus Data (*Data Flow*)

   Arus data merupakan tempat mengalirnya informasi dan digambarkan dengan garis yang menghubungkan komponen dari sistem. Arus data ditunjukan dengan arah panah dan garis diberi nama atas arus data yang mengalir.

3. Proses (*Process*)

   Proses merupakan apa yang dikerjakan oleh sistem. Proses dapat mengolah data atau aliran data masuk menjadi aliran data keluar. Setiap

proses memiliki satu atau beberapa masukan serta menghasilkan satu atau beberapa data keluar.

4. Simpanan Data (*Data Store*)

   Simpanan data merupakan tempat penyimpanan data yang ada dalam sistem. Data *store* dapat disimbolkan dengan dua garis sejajar atau dua garis dengan salah satu sisi samping terbuka. Proses dapat mengambil data dari atau memberikan data ke simpanan data (*database*).

2.13MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Berbeda dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Perintah-perintah yang ada di SQL antara lain [11] :

a. Menambah data (*Insert*)

   Perintah SQL yang digunakan untuk menambah data ke *database* adalah perintah *INSERT*. Perintah ini mempunyai bentuk umum sebagai berikut:

   INSERT [INTO] nama_tabel [(daftar *field*/kolom)] VALUES (daftar_data).

b. Mengakses data (*Select*)

   Perintah SQL yang digunakan untuk memilih data adalah perintah *SELECT*. Bentuk umum perintah ini adalah sebagai berikut:

   SELECT daftar_select *FROM* daftar_tabel

   [WHERE kondisi_pencarian]

[GROUP BY daftar_group_by]
[HAVING kondisi_pencarian]
[ORDER BY daftar_order[ASC │ DESC]]

c. Mengedit/Mengubah data (*Update*)

Perintah SQL yang digunakan untuk mengubah data adalah perintah *UPDATE SET*. Perintah ini mempunyai bentuk umum sebagai berikut:
UPDATE nama_table SET field1=databaru1 [,data2=databaru2]
[WHERE kondisi_update].

d. Menghapus data (*Delete*)

Perintah SQL yang digunakan untuk menghapus data adalah perintah *DELETE FROM*. Perintah ini mempunyai bentuk umum sebagai berikut:
DELETE FROM tabel_hapus WHERE kondisi_hapus.

MySQL memiliki beberapa kelebihan diantaranya [11] :

1. MySQL sebagai *Data Management System* (DBMS) dan *Relation Databese Management System* (RDBMS).
2. MySQL adalah *software database* yang *OpenSource*, artinya program ini bersifat *free* atau bebas digunakan oleh siapa saja tanpa harus membeli dan membayar lisensi kepada pembuatnya.
3. MySQL merupakan *database server*, jadi dengan menggunakan *database* ini dapat menghubungkan ke media *internet* sehingga dapat diakses jarak jauh.
4. MySQL merupakan sebuah *database client*. Selain menjadi *server* yang melayani permintaan, MySQL juga dapat melakukan *query* yang mengakses *database* pada *server*, jadi MySQL dapat juga berperan sebagai *Client*.
5. MySQL mampu menerima *query* yang bertumpuk dalam satu permintaan atau yang disebut *Multi-Threading*.
6. MySQL merupakan sebuah *database* yang mampu menyimpan data berkapasitas sangat besar hingga berukuran *Gigabyte* sekalipun.
7. MySQL didukung oleh driver ODBC, artinya *database* MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti Delphi maupun Visual Basic.

8. MySQL adalah *database* menggunakan enkripsi *password*. Jadi *database* ini cukup aman karena memiliki *password* untuk mengaksesnya.
9. MySQL dapat menciptakan lebih dari 16 kunci per tabel dan dalam satu kunci memungkinkan berisi belasan *field*. Serta MySQL mendukung *field* yang dijadikan sebagai kunci *primer* dan kunci unik atau (*Unique*).
10. MySQL memiliki kecepatan dalam pembuatan tabel maupun peng-*update*-an tabel.

2.14 PHP

PHP (*Hypertext Processor*) adalah bahasa pemrograman *scripting* sisi server (*server-side*), bahasa pemrograman yang digunakan oleh *server web* untuk menghasilkan dokumen HTML secara *on-the-fly*. PHP merupakan *interpreter* yang dapat dieksekusi sebagai program CGI untuk *server web*. PHP merupakan bahasa *script*, selain paling popular di lingkungan programer, pengembang web, di lingkungan server web apache [12].

Kelebihan script program menggunakan PHP antara lain :

a. *Web server* pendukung PHP banyak ditemukan di mana-mana, mulai dari IIS sampai dengan *apache*, dan konfigurasinya pun relatif mudah.
b. Pengembangannya mudah karena banyaknya milis-milis dan *developer* pengembang.
c. Referensi yang merujuk PHP banyak ditemukan, sehingga mudah untuk dipahami.
d. Merupakan bahasa *open source* yang dapat dioperasikan di berbagai sistem operasi, juga dapat dijalankan secara *run time* melalui *console* dan dapat menjalankan perintah-perintah sistem.

Selanjutnya akan dijelaskan mengenai dasar-dasar PHP yaitu sintak-sintak sederhana pada PHP. Sintaksis standar PHP ini diawali dengan <?php , dan diakhiri dengan ?>, setiap akhir baris kode PHP harus diakhiri dengan titik koma. Titik koma adalah pemisah dan digunakan untuk membedakan satu set instruksi dari yang lain.

1. Pernyataan print()

Pernyataan print() hanya akan membalikan satu nilai keluaran saja. Printf memiliki aturan seperti dibawah ini.

```
<?php
print("<p>Baju saya berwarna kuning.</p>");
?>
```

Gambar 2.12 Pernyataan *print*

2. pernyataan echo()

echo memiliki tujuan yang sama dengan print() meskipun ada perbedaan teknis antara echo() dan print(). Echo memiliki struktur dibawah ini.

```
<?php
$hello = "Selamat Pagi";
echo $hello." dunia";
?>
```

Gambar 2.13 Pernyataan *echo*

3. Variabel

Variabel adalah sebuah simbol yang dapat menyimpan nilai yang berbeda pada waktu yang berbeda. Dalam mendeklarasikan sebuah variabel, dimulai dengan menulis tanda dolar $, yang kemudian diikuti oleh nama variabel. Nama variabel dapat dimulai dengan huruf atau garis bawah, karakter ASCII 127 sampai 255 karakter. Contoh penulisan variabel yang valid adalah $color, $model_, $operating_system, atau $_schema.

4. Ekspresi

Ekspresi merupakan sebuah frasa yang mengekspresikan keterangan aksi dalam suatu program. Ekspresi terdiri dari *operand* dan *operator*. *Operand* merupakan masukan dari sebuah ekspresi.

```
$sum = $var1 + $var2; //$sum, $var1, $var2 adalah operand
$a++; //$a adalah operand
```

Gambar 2.14 Ekspresi

5. Struktur Kontrol

Struktur kontrol menerangkan alur kode yang terdapat dalam sebuah aplikasi. Mendefinisikan pengeksekusian karakteristik seperti bagaimana dan berapa kali kode tersebut di eksekusi.

a. Pernyataan Kondisi

Pernyataan kondisi memungkinkan sebuah program untuk melakukan berbagai macam input, menggunakan bermacam kondisi yang didasari dengan sebuah nilai masukan.

a) Pernyataan if
b) Pernyataan else
c) Pernyataan else-if

b. Pernyataan Perulangan

Pernyataan perulangan memungkinkan sebuah program untuk melakukan sebuah instruksi yang berulang sampai kondisi yang diinginkan tercapai.

a) Pernyataan while
b) Pernyataan do-while
c) Pernyataan for
d) Pernyataan foreach