BAB 2
TINJAUAN PUSTAKA

2.1. Text Mining

Text mining adalah bagian dari penelitian ilmu pengetahuan komputer yang mencoba memecahkan krisis dari muatan informasi dengan mengkombinasikan teknik dari *data mining*, *machine learning*, *natural language processing*, *information retrieval* dan *knowledge management*. Text mining mencari dan mengekstrak informasi berharga dari data sumber dengan mengidentifikasi dan mengeksplorasi pada pola yang menarik [5].

Proses kerja *Text Mining* hampir sama dengan *Data Mining* namun yang membedakan pada *Text Mining* data sumber yang digunakan sebagai *document collection* tidak memiliki pola yang terstruktur secara tekstual dan tidak diformulasikan di *database*, Sehingga dalam *text mining* diperlukan pengubahan data tidak terstruktur menjadi data yang terstruktur dan disajikan dalam bentuk numerik. Salah satu bentuk penggunaan *text mining* adalah untuk Analisis Sentimen.

2.2. Analisis Sentimen

Analisis Sentimen atau *Opinion Mining* mengacu pada Pengolahan Bahasa Alami, *Text Mining* dan Komputasi Linguistik yang bertujuan menganalisa, mengidentifikasi serta mengekstrak informasi dari kumpulan teks sebagai sumber[5]. Analisis Sentimen biasa digunakan dalam menganalisis pendapat, opini, penilaian, dan sentimen seseorang terhadap suatu topik atau bahasan tertentu.

Pada dasarnya Analisis Sentimen merupakan bagian dari klasifikasi, yang membedakan Analisis Sentimen melakukan klasifikasi tidak secara langsung, akan tetapi dipengaruhi oleh penggunaan bahasa. Analisis Sentimen bertujuan

untuk mencari dan mengelompokkan kecenderungan Positif atau Negatif dari pendapat yang berbentuk dokumen atau tulisan.

Ada beberapa tahapan dalam Analisis Sentimen pada penelitian ini antara lain :

1. *Preprocessing*
2. *Feature Selection*
3. *Naive Bayes Classifier*

2.2.1. Preprocessing

Sebelum melakukan klasifikasi, diperlukan proses *preprocessing* untuk mengolah data latih dan data uji yang tidak terstruktur menjadi terstruktur[6]. *Preprocessing* sangat dibutuhkan agar data latih dapat diklasifikasikan. Ada beberapa tahap dalam proses *preprocessing*, antara lain :

1. *Convert Emoticon*

Emoticon (*Emotional Icon*) adalah sebuah representasi perasaan dalam bentuk simbol tertentu yang menyerupai ekspresi wajah manusia. *Emoticon* biasanya digunakan untuk memperjelas perasaan seperti senang, sedih, marah, menangis, dll. dalam Analisis Sentimen penggunaan *emoticon* juga harus diperhitungkan karena memiliki arti tertentu.

Pada proses *Convert Emoticon*, *emoticon* yang terkandung dalam *tweet* akan diubah menjadi *string* yang bersesuaian, gunanya untuk memudahkan dalam proses pembobotan kata. Berikut adalah contoh deksripsi *emoticon* dalam jenis *emoticon western style* dapat dilihat pada Tabel 2.1 :

Tabel 2.1 Deskripsi Emoticon

Emoticon	Deskripsi
:-) :) :D :o) :] :3 :c) :> =] 8) =) :} :^) :ↄ) :))	Senang
>:[:-(:(:-c :c :-< :ↄC :< :-[:[:{ :'-(:'(	Sedih
:D :-D 8-D 8D x-D xD X-D XD =-D =D =-3 =3 B^D	Tertawa
:* <3	Suka

Berikut adalah contoh penggunaan *emoticon* dalam sebuah *tweet* serta hasil dari *Convert Emoticon* dapat dilihat pada Tabel 2.2 :

Tabel 2.2 Contoh Penerapan Convert Emoticon

Sebelum	Sesudah
@_sweetdhelis : @basuki_btp semangat pak <3 kami sekeluarga mendukung bapak.. :-) #teamahok	@_sweetdhelis : @basuki_btp semangat pak suka kami sekeluarga mendukung bapak.. senang #teamahok

2. *Cleansing*

Cleansing adalah proses menghilangkan karakter symbol yang dianggap sebagai *noise*. Karakter yang dimaksud seperti koma(,) titik(.) kutip(‘) kutip dua(“) tanda tanya(?) tanda seru(!) dsb. Proses *Cleansing* harus dilakukan setelah *Convert Emoticon* karena *emoticon* merupakan gabungan karakter simbol yang memiliki arti sedangkan simbol yang mengalami proses *cleansing* adalah karakter simbol yang tidak memiliki arti. Selain karakter symbol, angka, *Hastag*(#), *At*(@),URL dalam tweet pun dihilangkan karena tidak berhubungan dengan proses Analisis Sentimen. Berikut adalah contoh proses *Cleansing* pada suatu kalimat dapat dilihat pada Tabel 2.3 :

Tabel 2.3 Contoh Penerapan Cleansing

Sebelum	Sesudah
Semangat, Sehat, dan Sukses terus Pak Ahok... Tuhan Beserta Bapak...	Semangat Sehat dan Sukses terus Pak Ahok Tuhan Beserta Bapak

3. *Case Folding*

Case Folding adalah proses mengubah ukuran semua kata menjadi sama besar. Dalam proses ini semua kata dari *tweet* yang mengandung huruf besar akan diubah ke huruf kecil (*lower case*). Proses ini bertujuan untuk memudahkan pembandingan kata. Berikut adalah contoh proses *Case Folding* pada suatu kalimat dapat dilihat pada Tabel 2.4 :

Tabel 2.4 Contoh Penerapan Case Folding

Sebelum	Sesudah
Semangat Sehat dan Sukses terus Pak Ahok Tuhan Beserta Bapak	semangat sehat dan sukses terus pak ahok tuhan beserta bapak

4. *Remove Stopword*

Remove Stopword adalah proses menghilangkan kata yang sering muncul, bersifat umum dan dianggap sebagai kata yang tidak memiliki makna. Kata yang dihilangkan biasanya berperan hanya sebagai kata pelengkap dan tidak berpengaruh terhadap sentimen sebuah opini. Berikut adalah *Stopwords List* yang akan dihilangkan dari kalimat dapat dilihat pada Tabel 2.5 :

Tabel 2.5 Kata yang termasuk Stopwords List

Stopwords List				
ada	dgn	jadi	lain	saya
ah	di	juga	lg	seperti
aja	dia	kalau	lo	tapi
akan	dlm	kalo	mah	tentang
apa	dpt	ke	mau	tuh
bahwa	dulu	kita	mungkin	untuk
bilang	gue	kok	om	utk
dan	harus	krn	pak	wah
dari	ini	lagi	saat	yang
dengan	itu	lah	saja	yg

Berikut adalah contoh proses *Remove Stopwords* dapat dilihat pada Tabel 2.6 :

Tabel 2.6 Contoh Penerapan Remove Stopwords

Sebelum	Sesudah
Mentri Yohana dipolisikan tp tolong yg plng bnyk menyebar fitnah RatnaSpaet dipolisikan jg secepatnya meresahkan	Mentri Yohana dipolisikan tolong menyebar fitnah RatnaSpaet dipolisikan secepatnya meresahkan

5. *Negation Handling*

Negation Handling adalah proses mempertegas kata yang bermakna negasi. Proses ini biasanya dilakukan pada kata yang memiliki arti positif namun diawali dengan kata tertentu yang bermaksud menyatakan kebalikannya. Contoh kata yang digunakan sebagai negasi antara lain : “tidak”, ”tak”, ”ga”, ”gak”,” ngga” , ”engga”, ”enggak”, ”kaga”, ”kagak”, ”kurang”. Berikut adalah contoh proses *Negation Handling* pada suatu kalimat dapat dilihat pada Tabel 2.7.

Tabel 2.7 Contoh Penerapan Negation Handling

Sebelum	Sesudah
dishub dki ga punya nyali buat beresin term bayangan dpn poin square lebak bulus	dishub dki ga_punya nyali buat beresin term bayangan dpn poin square lebak bulus

6. *Tokenzing*

Tokenzing adalah proses memisahkan atau memotong semua kata yang ada pada suatu kalimat dan mengubahnya menjadi kumpulan token atau *term*. Token yang dihasilkan akan digunakan sebagai pembanding pada proses klasifikasi. Berikut adalah contoh proses *Tokenzing* dapat dilihat pada Tabel 2.8 :

Tabel 2.8 Contoh Penerapan *Tokenzing*

Sebelum	Sesudah
semangat sehat dan sukses terus pak ahok tuhan beserta bapak	String-1 : semangat String-2 : sehat String-3 : dan String-4 : sukses String-5 : terus String-6 : pak String-7 : ahok String-8 : tuhan String-9 : beserta String-10 : bapak

7. *Stemming*

Stemming adalah proses menghilangkan imbuhan pada kata yang berimbuhan dan mengembalikan kata tersebut kepada bentuk aslinya. Proses ini bertujuan untuk mengoptimasi pencocokan sehingga hasil yang dicari dapat lebih akurat. Tahapan dari algoritma Nazief dan Adriani adalah sebagai berikut :

1. Cek kata di kamus dasar, ketika ditemukan maka dianggap sebagai kata dasar dan algoritma dihentikan.
2. Lakukan pemotongan prefiks1 kemudian cek di database, ketika ditemukan maka dianggap kata dasar dan algoritma dihentikan.
3. Lakukan pemotongan prefiks2 kemudian cek di database, ketika ditemukan maka dianggap kata dasar dan algoritma dihentikan.
4. Lakukan pemotongan sufiks1 kemudian cek di database, ketika ditemukan maka dianggap kata dasar dan algoritma dihentikan.
5. Lakukan pemotongan sufiks2 kemudian cek di database, ketika ditemukan maka dianggap kata dasar dan algoritma dihentikan.
6. Lakukan pemotongan sufiks3 kemudian cek di database, ketika ditemukan maka dianggap kata dasar dan algoritma dihentikan.
7. Ketika seluruh tahapan proses telah dilakukan dan tidak ditemukan maka keluar dari algoritma.

Berikut adalah contoh proses *Stemming* dapat dilihat pada Tabel 2.9 :

Tabel 2.9 Contoh Penerapan *Stemming*

Sebelum	Sesudah
dishub dki ga_punya nyali buat beresin term bayangan dpn poin square lebak bulus	dishub dki ga_punya nyali buat beres term bayang dpn poin square lebak bulus

2.2.2. *Feature* Selection dengan metode *Chisquare*

Feature Selection adalah pemilihan ciri unik berupa kata atau *term* pada text mining. Dalam analisis sentiment data set berupa dokumen akan menjadi masalah ketika jumlahnya banyak. Feature Selection dilakukan untuk memilah kata atau *term* yang dianggap unik. Yang dimaksud dengan kata unik adalah kata yang hanya ada pada salah satu kategori. Kata tersebut dihitung sesuai dengan kemunculannya pada setiap kelas. Ada banyak metode dalam pembobotan kata. Metode yang digunakan dalam penelitian ini adalah *Chi Square*.

Chi Square adalah salah satu jenis uji komparatif non parametris yang dilakukan dua variabel, dimana skala data dua variabel adalah nominal. Ketika hasil perhitungan *Chi Square* bernilai tinggi, maka ada keterkaitan antara term (*t*) terhadap kelas (*c*) (Dependen) sedangkan ketika nilai *Chi* Square kecil maka tidak ada keterkaitan antara term (*t*) terhadap kelas (*c*). Berikut adalah persamaan dari *Chi Square* yang dapat dilihat pada Persamaan 2.1 :

$$\chi^2(t,c) = \frac{N(N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01})(N_{11} + N_{10})(N_{10} + N_{00})(N_{01} + N_{00})} \quad \textbf{(2.1)}$$

Keterangan :

N_{11} = Banyaknya *term* (t) yang muncul pada kelas (c)

N_{01} = Banyaknya *term* (t) yang tidak muncul di kelas (c)

N_{10} = Banyaknya dokumen kelas (c) yang tidak memiliki *term* (t)

N_{00} = Banyaknya dokumen bukan kelas (c) yang tidak memiliki *term* (t)

N = Jumlah kasus

2.2.3. Naive Bayes Classifier

Naive Bayes adalah teknik prediksi berbasis probabilistik sederhana pada penerapan teorema bayes dengan asumsi independensi (ketidaktergantungan) yang kuat (naif) [9]. *Naive Bayes Classifier* (NBC) mengasumsikan bahwa ada atau tidak suatu fitur tertentu pada suatu kelas tidak berhubungan dengan ada atau tidak fitur lainnya. Tergantung pada situasi yang tepat dari model probabilitas, *naive bayes classifier* dapat dilatih sangat efisien dalam *supervised learning*. naive bayes classifier hanya memerlukan sejumlah kecil data pelatihan untuk mengstimasi parameter yang diperlukan untuk klasifikasi. Karena variabel diasumsikan independen, hanya varian dari variabel-variabel untuk setiap kelas yang perlu ditentukan.

Prediksi Bayes didasarkan pada teorema Bayes dengan persamaan umum seperti pada Persamaan 2.2 :

$$P(H|E) = \frac{P(E|H) \text{ X } P(H)}{P(E)} \qquad \textbf{(2.2)}$$

Keterangan :

P(H|E) = Probabilitas akhir bersyarat (*conditional probability*) suatu hipotesis H terjadi jika diberikan bukti (*evidence*) E terjadi.

P(E|H) = Probabilitas sebuah bukti E akan mempengaruhi hipotesis H

P(H) = Probabilitas awal (*priors*) hipotesis H terjadi tanpa memandang bukti apapun

P(E) = Probabilitas awal (*priors*) bukti E terjadi tanpa memandang hipotesis/bukti yang lain

Adapun tahapan dalam melakukan klasifikasi Naive Bayes dengan pendekatan *Bernoulli Model* menurut Manning [2] dapat dilihat pada gambar 2.1 :

```
TRAINBERNOULLINB(ℂ, 𝔻)
1  V ← EXTRACTVOCABULARY(𝔻)
2  N ← COUNTDOCS(𝔻)
3  for each c ∈ ℂ
4  do N_c ← COUNTDOCSINCLASS(𝔻, c)
5     prior[c] ← N_c/N
6     for each t ∈ V
7     do N_ct ← COUNTDOCSINCLASSCONTAININGTERM(𝔻, c, t)
8        condprob[t][c] ← (N_ct + 1)/(N_c + 2)
9  return V, prior, condprob
```

Gambar 2.1 Algoritma NB (Bernoulli Model) : Data Training

Keterangan :

V = Kumpulan *term* yang ada pada keseluruhan dokumen training.

N = Banyaknya dokumen.

C = Kelas.

Nc = Jumlah dokumen pada kelas.

Nct = Banyaknya *term* pada dokumen yang ada pada suatu kelas.

t = *term*.

Langkah pertama tuliskan semua *term* di *vocabulary*. V = {t_1,t_2,t_3,t_4,..,t_n} kemudian hitung N (banyaknya dokumen). Setelah itu hitung *Prior* untuk setiap kelas dengan persamaan yang ada pada Persamaan 2.3 :

$$prior[c] = \frac{N_c}{N} \quad \textbf{(2.3)}$$

atau dapat dituliskan juga sebagai berikut (Persamaan 2.4)

$$\hat{P}(c) = \frac{N_c}{N} \quad \textbf{(2.4)}$$

Kemudian untuk setiap *term* pada *vocabulary*, Hitung Nct (banyaknya *term* pada dokumen di suatu kelas). Nct setiap *term* digunakan untuk menghitung *Conditional Probabilities.* Persamaan dari *conditional probabilities* dapat dilihat pada Persamaan 2.5 :

$$condprob[t][c] = \frac{(N_{ct} + 1)}{(N_c + 2)} \quad \textbf{(2.5)}$$

Dari perhitungan yang telah dilakukan maka akan diperoleh nilai *conditional probabilities* tiap *term* terhadap kelas yang nantinya akan digunakan untuk perhitungan nilai *term* pada klasifikasi data testing.

Setelah *Bernoulli Model* dilakukan pada data training, selanjutnya proses klasifikasi NB dengan *Bernoulli Model* di data testing dapat dilakukan. Dalam *Multivariate Bernoulli* dokumen diwakili oleh atribut biner yang menunjukan ada dan tidak ada *term* dalam dokumen. Frekuensi kemunculan *term* dalam dokumen tidak ikut diperhitungkan. Ketika menghitung peluang dari sebuah dokumen, semua nilai atribut dikalikan termasuk kemungkinan ada dan tidak ada *term* dalam dokumen. Pada model ini dokumen akan direpresentasikan ke dalam angka biner. *e* kan bernilai 1 jika terdapat dalam dokumen atau 0 jika tidak terdapat dalam dokumen dapat dilihat pada Persamaan 2.6.

$$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle, \; e_i \in \{0, 1\}. \quad \textbf{(2.6)}$$

Peluang dokumen (*d*) dalam kelas (*c*) dihitung menggunakan cara yang dapat dilihat pada Persamaan 2.7

$$P(c/d) \propto P(c) \prod_{t_i \in V} P(U_i = e_i/c), \quad \textbf{(2.7)}$$

dengan $P(U_i = e_i \,|c)$ adalah rasio dokumen dari kelas (*c*) yang mengandung term U_i, $P(c)$ adalah peluang dokumen pada kelas *c*. Tahapan algoritma klasifikasi bernoulli model menurut manning[2] dijelaskan pada gambar 2.2.

```
APPLYBERNOULLINB(ℂ, V, prior, condprob, d)
1  V_d ← EXTRACTTERMSFROMDOC(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ V
5     do if t ∈ V_d
6           then score[c] += log condprob[t][c]
7           else score[c] += log(1 − condprob[t][c])
8  return arg max_{c∈ℂ} score[c]
```

Gambar 2.2 Algoritma NB (Bernoulli Model) : Data Testing

Keterangan :

V = Kumpulan term yang ada pada keseluruhan dokumen training.

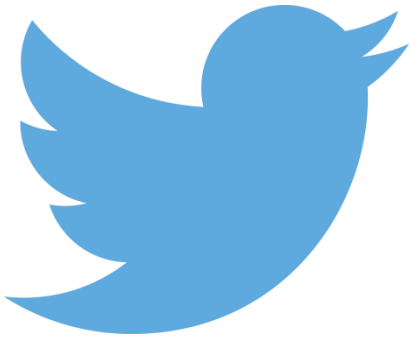
V_d = Kumpulan term yang ada pada dokumen uji.

C = Kelas.

t = term.

2.3. Twitter

Twitter adalah sebuah sosial media yang menawarkan layanan microblogging yang memungkinkan pengguna mengirimkan tweet atau kicauan dibatasi sebanyak 140 karakter dan membaca tweet pengguna lain. Twitter banyak dimanfaatkan oleh penggunanya sebagai media mengutarakan pendapat, berjualan produk, mengkampanyekan seuatu, dsb. Logo dari Twitter dapat dilihat pada gambar 2.3.



Gambar 2.3 Logo Twitter

Twitter didirikan pada bulan Maret 2006 oleh Jack Dorsey, Evan Williams, Biz Stone dan Noah Glass, dan diluncurkan pada bulan Juli 2006. Setelah hampir sepuluh tahun sekarang pengguna twitter berjumlah 305 juta pengguna pada tahun 2016. Berikut adalah fitur dan istilah yang ada pada Twitter diantaranya :

1. *Tweet*

Pesan singkat yang berisi informasi yang dituliskan oleh pengguna maksimal 140 karakter yang akan muncul di timeline pengguna tersebut dan timeline pengguna lain yang mengikutinya.

2. *Retweet*

Retweet dapat diartikan sebagai “mengirim tweet kembali” yaitu seorang pengguna akan mengirim kembali sebuah *tweet* yang mengacu pada *tweet* pengguna lain sebelumnya.

3. *Reply*

Reply adalah sebuah fitur dimana pengguna dapat menuliskan balasan kepada *tweet* pengguna lain.

4. *Like*

Like adalah sebuah fitur dimana pengguna dapat memberikan tanda hati sebagai tanda menyukai pada *tweet* pengguna lain.

5. *Follow*

Follow dapat diartikan sebagai mengikuti, yaitu pengguna akan memunculkan *tweet* pengguna lain yang ia ikuti yang akan muncul pada timeline.

6. *Follower*

Sedangkan *Follower* adalah istilah untuk menyebut pengguna lain yang mengikuti suatu pengguna.

7. *Unfollow*

Unfollow adalah suatu fitur untuk tidak mengikuti suatu pengguna yang telah di-*follow* sebelumnya.

8. # (*Hashtag*)

atau disebut juga tanda *hashtag* adalah cara untuk menjadikan kata pada suatu *tweet* agar menjadi kata kunci pada *tweet* tersebut. Ditulis dengan # kemudian diiktui oleh suatu kata, contohnya : #bandung, #kuliner, #jajan.

9. @ (*at*)

@ adalah cara penulisan *username* pada pengguna twitter yang bersifat unik yang membedakan satu pengguna dengan pengguna lain. Ditulis dengan @ kemudian diikuti langsung oleh nama pengguna, contoh : @basuki_btp, @Yusrilihza_Mhd, @sandiuno.

10. *Trending Topic*

Trending Topic adalah istilah yang mengacu kepada suatu topik yang sedang ramai diperbincangkan, ditandai dengan tanda # (*Hashtag*) diikuti oleh kata atau kalimat tanpa spasi. *Trending Topic* berdasarkan *keyword* atau kata yang paling banyak disertakan pada *Tweet* kebanyakan pengguna pada suatu periode tertentu.

2.3.1 API Twitter

API (*Application Programming Interface*) adalah sebuah penghubung yang disediakan oleh sistem operasi atau bahasa pemrograman yang berisi perintah, fungsi, komponen dan protokol yang dapat digunakan oleh pihak pengembang aplikasi. API mengubah perintah atau protocol pada *system calls* menjadi lebih terstruktur sehingga mudah diterapkan oleh Programmer dari luar.

Dalam API Twitter sendiri terdapat 3 bagian penting, yaitu :

1. OAuth Authenticaion

OAuth merupakan standarisasi kebijakan beberapa website besar. OAuth ini berperan sebagai autentifikasi antara aplikasi yang dibuat dan akun pengguna pada website tersebut. Dengan adanya OAuth Twitter, pengguna yang membangun aplikasi yang memanfaatkan API Twitter diwajibkan untuk memiliki akun Twitter terlebih dahulu dan melakukan *sign in*. OAuth memungkinkan pengguna mengakses akun dari aplikasi yang dibuat. Setelah memiliki akun dan *sign in*, pengguna akan diarahkan untuk melakukan persetujuan tentang penggunaan API Twitter, setelah pengguna melakukan prosedur yang berlaku

maka pengguna akan mendapatkan Token yang berfungsi untuk mengakses akun dari aplikasi yang dibuat. Token yang diberikan bersifat unik.

2. Rest API

Rest API adalah salah satu fitur yang disediakan oleh API Twitter. Fitur ini adalah fitur yang paling umum digunakan, berfungsi untuk melakukan *request* kepada Twitter untuk mengambil data spesifik, misalkan data pribadi pengguna, *tweet* pengguna, dsb. Rest API biasa digunakan untuk melakukan *tweet collecting* dalam jumlah banyak dengan memanfaatkan Search API, kebijakan twitter pengguna Rest API hanya dapat melakukan *request data* yang hanya bersifat *public*. Adapun periode waktu yang dapat dikumpulkan dengan Search API adalah satu minggu atau 7 hari kebelakang.

3. Streaming API

Streaming API adalah fitur yang memungkinkan pengguna untuk mendapatkan informasi dari Twitter secara *real-time.* Namun penggunaan Streaming API memerlukan sumber daya lebih banyak karena penggunanya melakukan koneksi secara terus menerus.

Streaming API dibedakan menjadi tiga jenis, yaitu :

a. Public Stream

Pengguna mengakses data public pada twitter seperti *tweet public, hashtag,dll.*

b. User Stream

Pengguna mengakses data ataupun *tweet* akun tertentu.

c. *Site Stream*

Pengguna memonitor *real-time* Twitter feeds untuk sejumlah besar pengguna.

2.4 Object Oriented Programming (OOP)

Object Oriented Programming (OOP) adalah sebuah pendekatan dalam pengembangan perangkat lunak dimana dalam struktur perangkat lunak tersebut didasarkan kepada objek yang saling berinteraksi satu sama lain untuk menyelesaikan suatu tugas. Semua data dan fungsi di dalam OOP dasatukan kedalam class atau objek. OOP berbeda dengan pemrograman prosedural, kode pada OOP dipecah ke dalam beberapa segmen yang biasa disebut prosedur atau fungsi sehingga dapat melakukan fungsionalitas lebih kompleks dari pada pemrograman prosedural.

Berikut adalah konsep-konsep pemrograman OOP antara lain :

1. Klass (*Class*)

Kelas adalah rancangan atau template dari objek, berisikan struktur yang digunakan oleh objek, atribut-atribut yang mendefinisikan data yang akan diolah, juga berisikan method yang dapat berbentuk prosedur ataupun fungsi untuk dapat mengelola data. Atribut adalah semua field identitas yang ada pada kelas tertentu, contohnya kelas mahasiswa mempunyai atribut nim, nama, alamat, no telepon, dsb. Sedangkan method adalah prosedur atau fungsi dari kelas itu sendiri atau disebut prilaku (*behavior*) dari kelas.

2. Objek (*Object*)

Objek adalah struktur yang menggabungkan data dan prosedur agar dapat bekerja. Apa yang dapat dilakukan oleh objek tergantung dengan isi yang telah didefinisikan dari kelasnya.

3. Abtraksi (*Abstraction*)

Dalam aplikasi OOP, Objek hanya berisi informasi yang relevan dengan konteks aplikasi, maka penting untuk menggunakan konsep dari abstraksi untuk menampung informasi diluar konteks aplikasi.

4. Enkapsulasi (*Encapsulaion*)

Enkapsulasi adalah proses penutupan akses langsung kepada data tertentu. Biasanya untuk mengakses data yang dienkapsulasi diharuskan melewati objek yang bertanggung jawab atas data tersebut.

5. Polimorisme (*Polymorphism*)

Polimorisme adalah kemampuan untuk merespon pesan yang sama dengan cara berbeda. Pada polimorisme dua objek memiliki method yang sama namun memiliki cara yang berbeda. Biasanya diterapkan pada proses yang disebut overloading.

6. Pewarisan (*Inheritance*)

Pewarisan mengklasifikasikan objek dalam bentuk program berdasarkan karakteristik umum. Pewarisan membuat pemrograman lebih mudah karena memungkinkan untuk mengkombinasikan karakteristik umum ke dalam kelas induk kemudian diturunkan/diwariskan kepada kelas anaknya.

2.5 Unified Modelling Language (UML)

Unified Modelling Language adalah bahasa pemodelan standar dalam pembangunan sistem atau perangkat lunak dengan pendekatan orientasi objek. Model adalah abstraksi dari bentuk sesungguhnya. Pemodelan dibuat dengan tujuan untuk menyederhanakan permasalahan yang kompleks agar mudah dipahami. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sistem menggunakan diagram dan teks-teks pendukung[7 R]. UML digunakan agar proses analisis dan perancangan perangkat lunak lebih fokus.

Ada beberapa Digram yang biasa digunakan dalam UML untuk melakukan pemodelan perangkat lunak berorientasi objek namun yang akan dipakai pada penulisan skripsi ini hanya 4 antara lain :

1. *Use Case Diagram*

Use Case Diagram digunakan untuk memodelkan interaksi antara pengguna dengan sistem yang berjalan. Penggunaan *Use Case* bertujuan untuk mengetahui fungsi yang ada dalam sistem serta siapa saja yang berhak mengakses fungsi tersebut. Pengguna sistem digambarkan sebagai *actor*, fungsi digambarkan dengan bentuk elips, dan relasi antara keduanya digambarkan dengan garis lurus.

2. *Activity Diagram*

Activity Diagram digunakan untuk memodelkan alur kerja tahapan aktivitas yang ada pada sistem. Diagram ini hanya memodelkan fungsi-fungsi serta aktivitas komponen yang ada pada sistem, tidak berkaitan dengan aktor.

3. *Sequence Diagram*

Sequence Diagram digunakan untuk memodelkan detail urutan dari proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case. Diagram ini menggambarkan interaksi antara object dan operasi yang terlibat beserta urutan antara operasi dan informasi yang diperlukan oleh masing-masing operasi.

4. *Class Diagram*

Class Diagram digunakan untuk memodelkan hubungan antar class yang ada pada sistem. Diagram ini menggambarkan himpunan class, kolaborasi dan relasi antar objek.