BAB 2
TINJAUAN PUSTAKA

2.1 Analisis sentimen

Analisis sentimen atau *opinion mining* merupakan proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam suatu kalimat opini. Analisis sentimen dilakukan untuk melihat pendapat atau kecenderungan opini terhadap sebuah masalah atau objek oleh seseorang, apakah cenderung berpandangan atau beropini negatif atau positif. Salah satu contoh penggunaan analisis sentimen dalam dunia nyata adalah identifikasi kecenderungan pasar dan opini pasar terhadap suatu objek barang. Besarnya pengaruh dan manfaat dari analisis sentimen menyebabkan penelitian semakin berkembang pesat[2].

Salah satu metode dari *text mining* yang bisa digunakan untuk menyelesaikan masalah *opinion mining* adalah *Naïve Bayes* (NB). *Naïve Bayes* bisa digunakan untuk mengklasifikasikan opini ke dalam opini positif dan negatif. *Naïve Bayes* bisa berfungsi dengan baik sebagai metode pengklasifikasi teks.

Tahapan yang dilakukan analisis sentimen yaitu mengumpulkan *tweets* positif, negatif dan netral lalu memisahkan menjadi tiga kategori yaitu positif, negatif dan netral tahap selanjutnya mencari kata kunci yang terkait dengan sekumpulan *tweet* positif, negatif dan netral. Kata kunci ini terutama untuk menggambarkan aspek apa yang mendapat opini.

2.2 *Review* Literatur

Samodra Joko et al [3] Melakukan Klasifikasi untuk teks berbahasa indonesia menggunakan data dari situs www.tempointeraktif.com dengan jumlah dokumen sebanyak 2400 yang dibagi menjadi 4 kategori. Pada tahap awal dengan melakukan *preprocessing* dan file HTML diedit secara manual untuk diambil teksnya saja lalu dokumen dipisahkan kedalam 4 kategori. Pengujian yang dilakukan menggunakan pendekatan *stopwords* dan tanpa *stopwords*. Hasil yang

didapat menunjukkan penghilangan kata-kata yang tidak penting tidak berpengaruh besar terhadap hasil klasifikasi.

Yudi Wibisono [4] dalam Klasifikasi Berita Berbahasa Indonesia menggunakan Naïve Bayes telah melakukan klasifikasi pada 582 dokumen berbahasa Indonesia menggunakan metode NB. Hal yang menarik adalah akurasi tidak menunjukkan peningkatan yang signifikan walaupun dokumen contoh telah meningkat banyak dari 70% menjadi 90% serta akurasi masih relative tinggi walaupun dokumen contoh secara ekstrim dikurangi hanya 58 dokumen (10%).

Amir Hamzah [5] melakukan pengelompokkan teks berita dan abstrak akademis. Pada setiap koleksi dokumen diformat dalam bentuk <DOC></DOC> untuk membedakan dokumen yang satu dengan dokumen yang lain, tag <DOCNO>..</DOCNO> untuk identifikasi nomor dokumen dan tag <CATNO>..</CATNO> untuk identifikasi dari nomor kategori dokumen.

2.3 Dokumen Pelatihan

Satu set pelatihan adalah seperangkat dokumen yang digunakan dalam berbagai bidang ilmu informasi untuk menemukan hubungan yang berpotensi prediktif. Set pelatihan yang digunakan dalam kecerdasan buatan, pembelajaran mesin, pemrograman genetik, sistem cerdas, dan statistik. Dalam semua bidang tersebut, satu set pelatihan memiliki banyak peran yang sama dan sering digunakan dalam hubungannya dengan satu set [7].

2.4 *Preprocessing text*

Data uji diproses terlebih dahulu sebelum digunakan dalam program. Proses ini disebut *preprocessing text*. *preprocessing text* bertujuan untuk mengurangi volume kosakata, menyeragamkan kata dan menghilangkan *noise*. *preprocessing text* meliputi *lexical analysis of the text* yang terdiri dari *case folding* dan *tokenizing* kemudian dilanjutkan dengan penghapusan *stopwords* dan *stemming* menggunakan algoritma Adriani dan Nazief

2.4.1 *Lexical Analysis of The Text*

Lexical analysis merupakan proses mengubah kumpulan karakter menjadi kumpulan kata [6]. Tujuan utama dari fase ini adalah mengidentifikasi kata-kata pada dokumen. Fase ini terdiri dari dua langkah yaitu *case folding* yaitu penyeragaman bentuk huruf serta penghilangan tanda baca dan parsing yang bertujuan untuk memecah kalimat menjadi kata-kata yang independen.

2.4.1.1 *Case folding*

Dokumen mengandung beragam variasi dari bentuk huruf sampai tanda baca. Variasi huruf harus diseragamkan (menjadi huruf besar saja atau huruf kecil saja) dan tanda baca harus dihilangkan untuk menghilangkan noise pada saat informasi. Hal ini dapat dilakukan dengan case folding. *Case folding* adalah proses mengubah huruf dalam dokumen menjadi huruf kecil [8]. Hanya huruf 'a' sampai dengan 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap sebagai delimiter.

Karakter-karakter simbol yang akan dihapus atau dianggap sebagai pemisah kata, dapat dilihat di tabel 2.1 dibawah ini.

Tabel 2.1 Karakter Simbol

Karakter Simbol		
^	,	:
@	(	;
%	)	|
$	-	]
#	<	}
„	>	{
~	+	\
&	=	/
.	!	“

2.4.1.2 *Tokenizing*

Dalam tahap ini dilakukan pemecahan atau pemotongan dokumen menurut tiap-tiap kata yang menyusun dokumen tersebut setelah mengalami proses

filtering. Hasil pemotongan (*tokenizing*) terhadap kata-kata tunggal tersebut dijadikan kumpulan *token* dan membentuknya menjadi sebuah daftar atau *list*[6].

2.4.2 *Stemming*

Stemming berfungsi mengubah kata menjadi kata dasar. Pada umumnya kata dalam dokumen memiliki variasi kombinasi imbuhan kata yang beragam. Variasi imbuhan dapat berupa *prefix* (awalan), *suffix* (akhiran), *infix* (sisipan) dan *confix* (kombinasi antara awalan dan akhiran). Dengan menggunakan stemming dapat mengurangi variasi kata yang sebenarnya memiliki kata dasar yang sama. Salah satu algoritma *stemming* yaitu Algoritma Nazief dan Adriani.

Algoritma stemming Nazief dan Adriani (1996) dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (prefix), sisipan (infix), akhiran (suffix) dan gabungan awalan-akhiran (confixes). Algoritma ini menggunakan kamus kata dasar dan mendukung recoding, yakni penyusunan kembali kata-kata yang mengalami proses stemming berlebih.

Algoritma *stemming* Nazief dan Adriani menggunakan morfologi imbuhan sebagai berikut :

1. *Inflection suffixes* merupakan kumpulan akhiran (*suffixes*) yang tidak merubah kata dasar. Misalnya kata ‘duduk’+’-lah’-> ’duduklah’. *Inflection suffixes* terbagi menjadi :
   a. *Particles* (P) termasuk ‘-lah’ dan ‘-kah’. Contoh : pergilah, pernahkah.
   b. *Possesive pronouns* (PP) termasuk ‘-ku’,’-mu’ dan ‘nya’. Contoh : diriku, dirimu, dirinya.
2. *Derivation suffixes* (DS) merupakan kumpulan akhiran (*suffixes*) yang langsung menempel pada kata dasar. Misalnya kata dasar ‘lapor’ ditambah derivation *suffix* ‘-kan’ menjadi ‘laporkan’. Setelah itu ditambah *inflection suffix* ‘-lah’ menjadi ‘laporkanlah’.
3. *Derivation prefixes* (DP) merupakan himpunan awalan (*prefixes*) yang menempel langsung pada kata dasar maupun terhadap kata yang telah

mempunyai sampai dua *derivation prefixes*. Misalnya *derivation prefixes* ‘mem-’ dan ‘per-’ + ‘-perindahkannya’->’memperindahkannya.

Dibawah ini adalah urutan pemakaian imbuhan sebagai *inflections* dan *derivations*:

[DP+[DP+[DP+]]] kata dasar [[+DS][+PP][+P]]

Kurung siku menandakan bahwa imbuhan bersifat optional [14]. Pada penelitian ini, algoritma yang digunakan untuk stemming bahasa Indonesia adalah algoritma Adriani dan Nazief. Algoritma dimulai dengan pembacaan tiap kata dari *file input* sehingga *input* berupa sebuah kata. Algoritma ini menggunakan kamus berisi kata-kata dasar sebagai pedoman pengecekan jika proses stemming telah mencapai kata dasar. Selain itu, pengecekan juga dilakukan pada panjang kata. Jika panjang kata kurang dari dua karakter, maka proses *stemming* tidak dilanjutkan. Setiap kata yang akan di-stemming akan menjalankan langkah-langkah sebagai berikut :

1. Kata yang belum di-*stemming* dicari di kamus kata dasar. Jika ditemukan, maka kata tersebut dianggap sebagai kata dasar. Algoritma mengembalikan kata tersebut dan berhenti.
2. *Inflection suffixes* (‘-lah’,’-ku’,’-kah’,’-mu’ atau ‘-nya’) dihapus. Awalnya penghapusan dilakukan pada akhiran *particles* (‘-lah’ atau ‘-kah’) kemudian disusul dengan penghapusan akhiran *possessive pronoun* (‘-ku’,’-mu’ atau ‘-nya’).
3. *Derivation suffixes* (‘-i’ atau ‘-an’) dihapus. Jika sukses, langkah 4 dijalankan. Jika langkah 4 tidak sukses
   a. Jika ‘-an’ dihapus dan huruf terakhir dari kata tersebut ‘-k’, maka ‘-k’ juga dihapus dan langkah 4 kembali dijalankan. Jika gagal, maka langkah 3b dijalankan.
   b. Hapus akhiran (‘-i’,‘-an’ atau ‘-kan’) kemudian simpan.
4. *Derivation previx* dihapus. Proses ini memiliki beberapa langkah yaitu :
   a. Jika akhiran dihapus pada langkah 3, maka cek kombinasi awalan akhiran berdasarkan Tabel 2.1. Jika ada yang cocok, maka algoritma mengembalikan kata tersebut dan berhenti.

b. Jika awalan sekarang sama dengan awalan sebelumnya, maka algoritma mengembalikan kata tersebut dan berhenti.
c. Jika tiga awalan telah dihapus sebelumnya, maka algoritma mengembalikan kata tersebut dan berhenti.
d. Jenis awalan ditentukan dengan langkah berikut.
    i. Jika awalan merupakan ‘di-‘,’ke-‘ atau ‘se-‘, maka jenis awalan tetap ‘di-‘,’ke-‘ atau ‘se-‘.
    ii. Jika awalan adalah ‘te-’,’be-‘,’me-‘ atau ‘pe-‘, maka diperlukan proses tambahan untuk mengekstrak karakter yang berawalan ‘te-‘ berdasarkan Tabel 2.2. Misalnya, kata ‘terlambat’, setelah menghapus ‘te-‘ menghasilkan ‘rlambat’. Ekstraksi pertama dilakukan menurut aturan ‘Set 1’ dan hal yang serupa terjadi pada lima baris pertama. Setelah itu, terdapat karakter ‘-l-‘ (Set 2), hal yang serupa terjadi pada baris tiga sampai lima. Sehingga kata menjadi ‘ambat’ sesuai dengan ‘Set 3’ dan buang kemungkinan baris tiga dan empat. Hasilnya ditunjukan pada kolom terakhir bahwa akhiran dari kata ‘terlambat’ berjenis awalan ‘ter-‘.
    iii. Jika dua karakter pertama tidak sama dengan ‘di-‘,’ke-‘,’se- ‘,’te-‘, ‘be-‘,’me-‘ atau ‘pe-‘, maka algoritma mengembalikan kata tersebut dan berhenti.
e. Jika jenis awalan ‘none’ atau tidak ada, maka algoritma mengembalikan kata tersebut dan berhenti. Jika awalan tidak ‘none’, maka jenis awalan ditemukan pada Tabel 2.3. Awalan yang ditemukan akan dihapus.

Jika kata dasar tidak ditemukan pada kamus, langkah 4 secara rekursif dijalankan untuk penghapusan lebih lanjut. Jika kata dasar ditemukan pada kamus, maka algoritma mengembalikan kata tersebut dan berhenti.

5. Setelah semua langkah dijalankan dan gagal, algoritma mengembalikan kata awal [9].

Tabel 2.2 Kombinasi awalan dan akhiran yang tidak diperbolehkan

Prefiks	*Disallowed Suffixes*
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

Pada Tabel 2.1 terdapat satu pengecualian, kata dasar ‘tahu’ diperbolehkan untuk kombinasi awalan ‘ke-‘ dan akhiran ‘i-’.

Tabel 2.3 Penentuan tipe awalan untuk kata dengan awalan ‘te-‘.

Following characters				*Prefix type*
Set 1	Set 2	Set 3	Set 4	
‘-r’	‘-r’	-	-	None
‘-r’	vowel	-	-	ter-luluh
‘-r’	Not (‘-r’ or vowel)	‘-er-’	vowel	Ter
‘-r’	Not (‘-r’ or vowel)	‘-er-’	Not vowel	None
‘-r’	Not (‘-r’ or vowel)	Not ‘-er-‘	-	Ter
Not (vowel or ‘-r’)	‘-er-’	vowel	-	None
Not (vowel or ‘-r’)	‘-er-’	Not vowel	-	Te

Keterangan : Jika awalan ‘te-‘ tidak sesuai dengan salah satu dari aturan pada tabel, maka dikembalikan ke ‘none’. Aturan yang sama berlaku untuk ‘be-‘, ‘me-‘ dan ‘pe-‘.

Tabel 2.4 Penentuan awalan dari jenis awalan

Prefix type	*Prefix to be removed*
di	di-
ke	ke-
se	se-
te	te-
ter	ter-

ter-luluh	ter-

2.4.3 Penghapusan *Stopwords*

Kata yang terlalu sering muncul pada setiap dokumen tidak terlalu baik digunakan sebagai kata kunci. Faktanya, kata yang muncul sampai 80% pada setiap dokumen tidak berguna untuk pengambilan informasi [6]. Kata-kata yang sering muncul tersebut dikenal sebagai *stopwords*. *Stopwords* adalah kata-kata yang sering muncul dan tidak memiliki arti. Daftar kata *stopword* yaitu kumpulan seluruh kata-kata yang sering muncul dan tidak memiliki arti yang disimpan dalam bank kata khusus *stopword* yang terdiri dari 810 *stopword* bahasa indonesia Contoh dari *stopwords* untuk bahasa indonesia yaitu adalah, adanya, akhirnya, bahkan, bagaimanapun. Dengan dibuangnya *stopwords*, ukuran kosakata menjadi berkurang sehingga hanya kata-kata penting yang terdapat dalam dokumen dan diharapkan akan menjadi kata-kata yang memiliki bobot yang tinggi.

2.5 Klasifikasi Dokumen

Klasifikasi merupakan proses mengidentifikasi objek ke dalam kelas, grup atau kategori berdasarkan prosedur, karakteristik dan definisinya [3]. Salah satu bentuk klasifikasi yang diterapkan dalam penelitian ini adalah klasifikasi *Tweets*.

Pada penelitian ini, setiap *tweets* dipetakan tepat satu ke kategori yang paling sesuai dengan isinya. Klasifikasi dokumen terbagi menjadi dua salah satunya yaitu *supervised document classification* yaitu klasifikasi yang melibatkan mekanisme external (seperti human *feedback*) guna menyediakan informasi klasifikasi yang benar. Klasifikasi tipe ini biasanya menggunakan dokumen pembelajaran. Salah satu contoh *supervised document classification* adalah *Naive Bayes* yang akan diulas pada penelitian ini.

2.6 *Naive Bayes*

Naive Bayes tergolong *supervised classification document*. Metode ini memiliki tingkat keakuratan tinggi dengan penghitungan yang sederhana [7]. Metode ini mengasumsikan kemunculan suatu kata tidak mempengaruhi kemunculan kata yang lain. Walaupun asumsi ini melanggar aturan disetiap

bahasa, asumsi ini tidak berpengaruh besar terhadap tingkat keakuratan metode ini [1].

Teorema *Bayes* berawal dari persamaan 2.1, yaitu:

$$P(A|B) = \frac{P(B \cap A)}{P(B)} \quad (2.1)$$

Di mana P(A|B) artinya peluang A(Kata) pada data uji jika diketahui keadaan B(Kelas) positif, negatif dan netral. Kemudian dari persamaan 2.1 didapatkan persamaan 2.2

$$P(B \cap A) = P(B|A).P(A) \quad (2.2)$$

Sehingga didapatkan teorema *Bayes* seperti persamaan yang ditunjukkan pada persamaan 2.3

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \quad (2.3)$$

Algoritma pembelajaran bayes menghitung probabilitas eksplisit untuk menggambarkan hipotesa yang dicari. Sistem dilatih menggunakan data latih lengkap berupa pasangan nilai-nilai atribut dan nilai target kemudian sistem akan diberikan sebuah data baru dalam bentuk $<a_1, a_2, a_3,....a_n>$ dan sistem diberi tugas untuk menebak nilai fungsi target dari data tersebut [7].

Naive Bayes memberi nilai target kepada data baru menggunakan nilai V_{map}, yaitu nilai kemungkinan tertinggi dari seluruh anggota himpunan set domain V yang ditunjukkan pada persamaan 2.4.

$$Vmap = \underset{V_j \in V}{\operatorname{argmax}} \; P(V_j|a_1, a_2, a_3 \ldots . a_n) \quad (2.4)$$

a = kata dalam dokumen

V_j = Kategori ke j

j = 1, 2, 3

V_1 = Kategori Positif

V_2 = Kategori Negatif

V_3 = Kategori Netral

Teorema *Bayes* kemudian digunakan untuk menulis ulang persamaan 2.4 menjadi persamaan 2.5

$$Vmap = \underset{V_j \in V}{\text{argmax}} \frac{P(a_1,a_2,a_3,\ldots,a_n|V_j)P(V_j)}{P(a_1,a_2,a_3\ldots a_n)} \quad (2.5)$$

Karena $P(a_1,a_2,a_3,\ldots,a_n)$ nilainya konstan untuk semua V_j sehingga persamaan 2.5 dapat ditulis dengan persamaan 2.6.

$$V_{map} = \underset{V_j \in V}{\text{argmax}} P(a_1,a_2,a_3\ldots..a_n |V_j)\, P(V_j) \quad (2.6)$$

Tingkat kesulitan menghitung $P(a_1,a_2,a_3,\ldots.a_n|V_j)\, P(V_j)$ menjadi tinggi karena jumlah kata $P(a_1,a_2,a_3,\ldots.a_n|V_j)\, P(V_j)$ bisa menjadi sangat besar. Ini disebabkan jumlah kata tersebut sama dengan jumlah kombinasi posisi kata dikali dengan jumlah kategori. Metode klasifikasi *Naive Bayes* menyederhanakan hal ini dengan bekerja dengan asumsi bahwa atribut-atribut yang digunakan bersifat *conditionally independent* antara satu dan yang lainnya, dengan kata lain dalam setiap kategori, setiap kata *independent* satu sama lain. Sehingga menjadi persamaan 2.7.

$$V_{map} = \underset{V_j \in V}{argmax}\, P(V_j) \prod_{i=1}^{n} P(a_i\,|V_j) \quad (2.7)$$

Subtitusi persamaan 2.7 dengan persamaan 2.6 menjadi persamaan 2.8.

$$V_{map} = \underset{V_j \in V}{argmax}\, P(V_j) \prod_{i=1}^{n} P(a_i\,|V_j) \quad (2.8)$$

V_{Map} adalah nilai probabilitas hasil perhitungan *Naive Bayes*. Untuk nilai fungsi target yang bersangkutan. Frekuensi kemunculan kata menjadi dasar perhitungan nilai dari $P(V_j)$ dan $P(a_1|v_j)$. Himpunan *set* dari nilai-nilai probabilitas

ini berkorespondensi dengan hipotesa yang ingin dipelajari. Hipotesa kemudian digunakan untuk mengklasifikasikan data-data baru. Pada pengklasifikasikan teks, perhitungan persamaan 2.7 dapat didefinisikan :

$$P(v_j) = \frac{docs_j}{|D|} \quad (2.9)$$

$$P(w_k|v_j) = \frac{n_k+1}{n+|kata|} \quad (2.10)$$

Keterangan :

1. $Docs_j$: Kumpulan dokumen yang memiliki kategori v_j.
2. |D| : Jumlah dokumen yang digunakan dalam pelatihan (kumpulan data latih).
3. n : jumlah total kata yang terdapat di dalam kata tekstual yang memiliki nilai fungsi target yang sesuai.
4. n_k : Jumlah kemunculan kata w_k pada semua data tekstual yang memilliki nilai fungsi target yang sesuai.
5. |kata| : Jumlah kata yang berbeda yang muncul dalam seluruh data tekstual yang digunakan.

2.7 Evaluasi

Hasil klasifikasi teks dengan menggunakan metode Naive Bayes akan dievaluasi keakuratannya menggunakan pendekatan *Accuracy and Error rate*.

2.7.1 *Accuracy and Error Rate*

Untuk mengevaluasi kinerja klasifikasi teks, *Accuracy and Error rate* secara luas digunakan. Definisi akurasi dan kesalahan tingkat untuk masing-masing kategori adalah sebagai berikut [10]:

$$\text{Accuracy } \frac{TP+TN}{TP+FP+FN+TN} \quad (2.11)$$

$$\text{Err} \frac{FP+FN}{TP+FP+FN+TN} \quad (2.12)$$

TP = True Positive, jumlah data uji yang benar diklasifikasi ke kategori yang benar

TN = True Negative, jumlah data uji yang benar klasifikasikan ke kategori yang salah

FP = False Positive, jumlah data uji yang salah diklasifikasikan ke kategori yang benar

FN = False Negative, jumlah data uji yang salah diklasifikasikan ke kategori yang salah

Mirip dengan penyaringan spam, klasifikasi yang salah dari dokumen ke kategori memiliki konsekuensi berat. Kedua langkah sederhana cukup intuitif dan menyatakan nilai persentase untuk hasil kategorisasi baik benar dan salah. Namun, dalam situasi di mana TN lebih besar dari TP untuk beberapa kategori, mereka bisa menyesatkan. Dalam hal ini, akurasi yang tinggi dapat diperoleh, sebagai dokumen yang umumnya tidak relevan dengan kategori tertentu. Namun, dua kesalahan set FB dan FN dapat dibandingkan dengan set yang relevan TP.

2.8 Konsep Dasar Pemrograman Objek

Pemrograman berorientasi objek berarti sebuah teknik pemrograman yang dalam proses pengembangannya menggunakan terminologi objek, di mana setiap objek memiliki atribut beserta dengan fungsi yang dapat saling berinteraksi satu dengan yang lain. Contohnya objek makhluk hidup – baik manusia ataupun binatang maka akan mengenalinya dari bentuk, ukuran, beratnya kemudian dari perilakunya dapat dilihat bahwa makhluk hidup dapat melihat, makan, berjalan, berlari dan selain itu terdapat fungsi lainnya seperti fungsi peredaran darah yang berhubungan dengan fungsi pernapasan atau fungsi pencernaan.

Ciri-ciri inilah yang menjadi ide dasar bagaimana mengembangkan sebuah perangkat lunak yang kompleks dengan menggunakan model objek tersebut. Dengan demikian, proses pengembangan perangkat lunak akan lebih mudah karena hanya akan menyesuaikan model pemrograman dengan objek yang kita buat. PBO memiliki tujuan untuk memberikan pemahaman sistem kepada user atau client. Hal ini karena *user/client* dapat lebih mudah memahami alur pemrograman dengan kasus yang dihadapi, hal ini berbeda dengan pemrograman terstruktur karena lebih berorientasi kepada programmer untuk menyelesaikan sebuah kasus.

Pengertian UML (*Unified Modeling Language*)

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar. Ketika akan membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang dibuat berhubungan satu dengan lainnya dan harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya [14].

UML adalah himpunan struktur danteknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat Lunak
2. Sarana Komunikasi antara perangkat lunak dengan proses bisnis
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Beberapa literature menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan sepuluh karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram perwaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokkan menjadi 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.

2. Diagram Paket (*Package Diagram*). Bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram Use-Case. Bersifat statis. Diagram ini memperlihatkan himpunan use-case dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengordinasikan dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan sequence (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram*). Bersifat dinamis. Diagram sebagai pengganti diagramkolaborasi UML 1.4 yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram Statechart (*Statechart Diagram*). Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (state), transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (interface), kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram Aktivitas (*Activity Diagram*). Bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*). Bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antaramuka-antarmuka serta kolaborasi-kolaborasi.

Diagram Deployment (*Deployment Diagram*). Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*). Memuat simpul-

simpul beserta komponen-komponen yang ada di dalamnya. Diagram deployment berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*)