BAB 2
LANDASAN TEORI

2.1 Emosi

Emosi dapat digambarkan sebagai keadaan yang pada umumnya disebabkan oleh suatu kejadian penting sebuah subyek yang meliputi (a) keadaan mental sadar yang dinyatakan dengan kemampuan mengenali, kualitas perasaan dan diarah untuk beberapa subyek, (b) gangguan jasmani pada beberapa organ tubuh, (c) pengenalan ekspresi pada wajah, suara dan isyarat tubuh, (d) kesiapan untuk melakukan tindakan tertentu. Karenanya emosi dalam sosiobiologi adalah kecenderungan mental (*conative* dan kognitif), keadaan, proses dan model komputasi harus spesifikasi semirip mungkin [15].

Sejumlah penelitian tentang emosi manusia telah dilakukan sehingga ada kesepakatan tentang emosi dasar [16]:

1. Takut sebagai ancaman fisik atau sosial untuk diri sendiri.
2. Marah sebagai ganjalan atau frustasi dari peran atau tujuan yang dirasakan orang lain.
3. Jijik menggambarkan penghapusan atau jarak dari seseorang, obyek, atau menolak ide untuk diri sendiri dan menghargai peran dan tujuan.
4. Sedih digambarkan sebagai kegagalan atau kerugian tentang peran dan tujuan.
5. Senang digambarkan sebagai berhasil atau bergerak menuju selesainya peran yang bernilai atau tujuan.

2.2 International Survey On Emotion Antecedents And Reaction (ISEAR)

ISEAR adalah sebuah data tentang emosi yang selama bertahun-tahun dimulai pada tahun 1990-an, sebuah kelompok besar psikolog di seluruh dunia mengumpulkan data dalam proyek ISEAR, dipimpin oleh Klaus R. Scherer dan Harald Wallbott. Mahasiswa responden, baik psikolog dan non-psikolog, diminta untuk melaporkan situasi di mana mereka mengalami semua tujuh emosi utama (senang, takut, marah, sedih, jijik, malu, dan rasa bersalah). Dalam setiap kasus, pertanyaan-pertanyaan meliputi cara mereka telah mengenali situasi dan bagaimana

mereka bereaksi. Data akhir ini memuat laporan tentang tujuh emosi masing-masing sekitar 3.000 responden di 37 negara di lima benua [7].

Dalam penelitian ini data latih yang digunakan diperoleh dari dataset ISEAR yang berbahasa Inggris kemudian diterjemahkan ke dalam bahasa Indonesia tanpa mengurangi maksud dari kalimat-kalimat dalam ISEAR. Data yang diambil sebanyak 1000 data dengan banyak masing-masing emosi adalah 200 data karena terdapat lima kategori emosi yang akan diklasifikasikan. Lima kategori emosi tersebut adalah senang, sedih, marah, takut dan bersalah.

2.3 Preprocessing

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Masukan awal pada proses ini adalah berupa dokumen teks. Teks yang akan dilakukan proses pada umumnya memiliki beberapa karakteristik, diantaranya adalah memiliki dimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Agar dapat dihasilkan fitur yang baik dan mewakili data dengan baik, perlu dilakukan tahapan *preprocessing* [17].

Adapun tahapan *preprocessing* yang akan dilakukan pada penelitian ini yaitu *case folding, convert negation, tokenizing, stopword removal dan stemming*. Berikut adalah gambaran tahap *preprocessing* yang dapat dilihat pada Gambar 2.1:



Gambar 2.1 Tahap Preprocessing

2.3.1 Case Folding

Case folding merupakan tahapan proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil, serta menghilangkan karakter selain a-z, kecuali karakter pemecah kalimat, seperti spasi, tab, dan *newline* (lompat baris) [17].

2.3.2 Convert Negation

Convert negation merupakan proses konversi kata-kata negasi yang terdapat pada suatu kalimat, karena kata negasi mempunyai pengaruh dalam merubah nilai emosi pada sebuah kalimat [11]. Di dalam klasifikasi teks emosi kata-kata negasi menjadi sangat berarti sehingga tidak boleh dihilangkan [18]. Perbedaan arti “tanpa

cinta" dan "tidak senang" dapat menempatkan dokumen dalam kelas yang berbeda. Kata-kata negasi tersebut meliputi kata "bukan", "tidak" dan "tanpa". Jika terdapat kata negasi maka akan digabungkan dengan kata setelahnya.

Langkah-langkah pada tahap *convert negation* adalah sebagai berikut:

a. Kata yang digunakan adalah hasil dari *case folding*.
b. Jika ditemukan data yang mengandung kata-kata negasi maka kata negasi tersebut akan digabungkan dengan kata setelah kata negasi tersebut.

2.3.3 Tokenizing

Tokenizing adalah proses pemotongan *string* input berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melihat pemisah seperti spasi, tab, dan *newline* (lompat baris) [17].

2.3.4 Stopwords Removal

Kebanyakan bahasa resmi di berbagai negara memiliki kata fungsi dan kata sambung seperti artikel dan preposisi yang hampir selalu muncul pada dokumen teks. Biasanya kata-kata ini tidak memiliki arti yang lebih di dalam memenuhi kebutuhan seorang *searcher* di dalam mencari informasi. Kata-kata tersebut (misalnya *a, an, the on* pada bahasa Inggris) disebut sebagai *stopwords* [19]. Sebuah sistem *text retrieval* biasanya disertai dengan sebuah *stoplist*. *Stoplist* berisi sekumpulan kata yang 'tidak relevan', namun sering sekali muncul dalam sebuah dokumen. Dengan kata lain *stoplist* berisi sekumpulan *stopwords* [20].

Stopwords removal adalah sebuah proses untuk menghilangkan kata yang 'tidak relevan' pada hasil *parsing* sebuah dokumen teks dengan cara membandingkannya dengan *stoplist* yang ada [21]. Tahapan ini bertujuan untuk menghilangkan kata yang dianggap tidak dapat memberikan pengaruh dalam menentukan suatu kategori tertentu dalam suatu dokumen. Proses ini dilakukan karena kata tersebut sering muncul hampir di setiap dokumen sehingga dianggap tidak dapat menjadi pembeda yang baik dalam membedakan kategori yang satu dengan kategori yang lain.

Setiap kata akan diperiksa apakah masuk ke dalam *stoplist* atau tidak, jika sebuah kata masuk ke dalam *stoplist* maka kata tersebut tidak akan diproses lebih

lanjut dan kata tersebut akan dihilangkan. Sebaliknya apabila sebuah kata tidak termasuk ke dalam *stoplist* maka kata tersebut akan masuk ke proses berikutnya.

Stoplist yang digunakan diambil dari penelitian Fadillah Z. Tala dengan judul *A study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Jumlah *stopword* yang terdapat pada penelitian tersebut adalah sebanyak 756 kata [17]. Kata-kata yang termasuk *stopword* tersebut biasanya berupa kata ganti orang, kata penghubung, pronominal penunjuk, dan lain sebagainya.

2.3.5 Stemming

Stemming merupakan suatu proses yang terdapat dalam sistem *information retrieval* yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh, kata bersama, kebersamaan, menyamai, akan di-*stem* ke *root word*-nya yaitu "sama". Kata dasar yang digunakan diambil dari situs http://bahtera.org. Bahtera adalah kamus Bahasa Indonesia yang menjadi rujukan sesuai Kamus Besar Bahasa Indonesia, kata dasar yang terdapat di bahtera ini sebanyak 28.526 kata. Proses *stemming* pada teks berbahasa Indonesia berbeda dengan *stemming* pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia, selain sufiks, prefiks, dan konfiks juga dihilangkan.

Algoritma yang digunakan pada proses ini adalah algoritma Nazief & Adriani. Berdasarkan hasil penelitian [12] menyimpulkan bahwa proses *stemming* dokumen teks berbahasa Indonesia menggunakan algoritma Porter memiliki prosentase keakuratan (presisi) lebih kecil dibandingkan dengan *stemming* menggunakan algoritma Nazief & Adriani. Oleh karena itu, pada penelitian ini algoritma stemming yang digunakan adalah algoritma Nazief & Adriani.

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tersebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa *particles* ("-lah", "-kah", "-tah" atau "-pun") maka langkah ini diulangi

lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.

3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a.
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan 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 pergi ke langkah 4b.
    a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
    b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan *Recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “none” maka awalan dapat dilihat pada Tabel 2.2 Hapus awalan jika ditemukan.

Tabel 2.1 Kombinasi Awalan Akhiran yang Tidak Diijinkan

Awalan	Akhiran yang tidak diijinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

Tabel 2.2 Cara Menentukan Tipe Awalan Kata yang Diawali dengan "te-"

Following Characters				**Tipe Awalan**
Set 1	**Set 2**	**Set 3**	**Set 4**	
"-r-"	"-r-"	-	-	none
"-r-"	vowel	-	-	ter-luluh
"-r-"	not (vowel or "-r-")	"-er-"	vowel	ter
"-r-"	not (vowel or "-r-")	"-er-"	not vowel	ter-
"-r-"	not (vowel or "-r-")	not "-er-"	-	ter
not (vowel or "-r-")	"-er-"	vowel	-	none
not (vowel or "-r-")	"-er-"	not vowel	-	te

Tabel 2.3 Jenis Awalan Berdasarkan Tipe Awalannya

Tipe Awalan	Awalan yang harus dihapus
di-	di-
ke-	ke-
se-	se-
te-	te-
ter-	ter-
ter-luluh	ter

Untuk mengatasi keterbatasan pada algoritma di atas, maka ditambahkan aturan-aturan dibawah ini:

1. Aturan untuk reduplikasi.
   - Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka *root word* adalah bentuk tunggalnya, contoh: "buku-buku" *root word*-nya adalah "buku".
   - Kata lain, misalnya "bolak-balik", "berbalas-balasan", dan "seolah-olah". Untuk mendapatkan *root word*-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki *root word* yang sama maka diubah menjadi bentuk tunggal, contoh: kata "berbalas-balasan", "berbalas" dan "balasan" memiliki *root word* yang sama yaitu "balas", maka *root word* "berbalas-balasan" adalah "balas". Sebaliknya, pada kata "bolak-balik", "bolak" dan

"balik" memiliki *root word* yang berbeda, maka *root word*-nya adalah "bolak-balik"

2. Tambahan bentuk awalan dan akhiran serta aturannya.
   - Untuk tipe awalan "mem-", kata yang diawali dengan awalan "memp-" memiliki tipe awalan "mem-".
   - Tipe awalan "meng-", kata yang diawali dengan awalan "mengk-" memiliki tipe awalan "meng-".

Berikut contoh-contoh aturan yang terdapat pada awalan sebagai pembentuk kata dasar.

1. Awalan SE-

   Se + semua konsonan dan vokal tetap tidak berubah.

   Contoh:
   - Se + bungkus = sebungkus
   - Se + nasib = senasib
   - Se + arah = searah
   - Se + ekor = seekor

2. Awalan ME-
   a. Me + vokal (a, i, u, e, o) menjadi sengau "meng".

      Contoh:
      - Me + inap = menginap
      - Me + asuh = mengasuh
      - Me + ubah = mengubah
      - Me + ekor = mengekor
      - Me + oplos = mengoplos

   b. Me + konsonan b menjadi "mem"

      Contoh:
      - Me + beri = memberi
      - Me + besuk = membesuk

   c. Me + konsonan (c, d, j) menjadi "men"

      Contoh:
      - Me + cinta = mencinta

- Me + cuci = mencuci
- Me + didik = mendidik
- Me + dengkur = mendengkur
- Me + jepit = menjepit
- Me + jemput = menjemput

d. Me + konsonan g dan h menjadi “meng”

Contoh:

- Me + gosok = menggosok
- Me + hokum = menghukum

e. Me + konsonan k menjadi “meng” (luluh)

Contoh:

- Me + kukus = mengukus
- Me + kupas = mengupas

f. Me + konsonan p menjadi “mem” (luluh)

Contoh:

- Me + pesona = mempesona
- Me + pukul = memukul

g. Me + konsonan s menjadi “meny” (luluh)

Contoh:

- Me + sapu = menyapu
- Me + satu = menyatu

h. Me + konsonan t menjadi “men” (luluh)

Contoh:

- Me + tanama = menanam
- Me + tukar = menukar

i. Me + konsonan (l, m, n, r, w) menjadi tetap “me”

Contoh:

- Me + lempar = melempar
- Me + masak = memasak
- Me + naik = menaik
- Me + rawat = merawat

- Me + warna = mewarna

3. Awalan KE-

   Ke + semua konsonan dan vokal tetap tidak berubah.

   Contoh:

   - Ke + bawa = kebawa
   - Ke + atas = keatas

4. Awalan PE-

   a. Pe + konsonan (g, h, k) dan vokal menjadi “per”

      Contoh:

      - Pe + gelar + an = pergelaran
      - Pe + hitung + an = perhitungan
      - Pe + kantor + an = perkantoran

   b. Pe + konsonan “t” menjadi “pen” (luluh)

      Contoh:

      - Pe + tukar = penukar
      - Pe + tikam = penikam

   c. Pe + konsonan c, d, j, z) menjadi “pen”

      Contoh:

      - Pe + cuci = pencuci
      - Pe + didik = pendidik
      - Pe + jahit = penjahit
      - Pe + zina = penzina

   d. Pe + konsonan (b, f, v) menjadi “pem”

      Contoh:

      - Pe + beri = pemberi
      - Pe + bunuh = pembunuh

   e. Pe + konsonan “p” menjadi “pem” (luluh)

      Contoh:

      - Pe + pikir = pemikir
      - Pe + potong = pemotong

f. Pe + konsonan “s” menjadi “peny” (luluh)

Contoh:

- Pe + siram = penyiram
- Pe + sabar = penyabar

g. Pe + konsonan (l, m, n, r, w, y) tetap tidak berubah.

Contoh:

- Pe + lamar = pelamar
- Pe + makan = pemakan
- Pe + nanti = penanti
- Pe + wangi = pewangi

Hasil proses *stemming* tersebut digunakan dalam melakukan pembobotan TF-IDF.

2.4 Pembobotan TF-IDF

Pembobotan dapat diperoleh berdasarkan jumlah kemunculan suatu *term* (kata) dalam sebuah dokumen *term frequency* (tf) dan jumlah kemunculan *term* dalam koleksi dokumen *inverse document frequency* (idf). Nilai idf sebuah *term* dapat dihitung menggunakan persamaan sebagai berikut [22]:

$$IDF = log\left(\frac{D}{dfi}\right) \quad (2.1)$$

D adalah jumlah dokumen dan *dfi* adalah jumlah kemunculan (frekuensi) *term* terhadap D.

Adapun persamaan yang digunakan untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci (*query*), yaitu:

$$W_{d,t} = tf_{d,t} * IDF_t \quad (2.2)$$

dengan keterangan sebagai berikut:

d = dokumen ke–d

t = *term* ke–t dari kata kunci

tf = *term* frekuensi/frekuensi kata

$W_{d,t}$ = bobot dokumen ke–d terhadap *term* ke–t.

2.5 Metode Support Vector Machine

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi linear dalam sebuah ruang fitur

(*feature space*) berdimensi tinggi, dilatih dengan algoritma pembelajaran yang didasarkan pada teori opimasi dengan mengimplementasikan *learning bias* yang berasal dari teori pembelajaran statistik [23]. SVM adalah salah satu teknik yang relatif baru dibandingkan dengan teknik lain, tetapi memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti *bioinformatics*, pengenalan tulisan tangan, klasifikasi teks dan lain sebagainya [24]. Proses pembelajaran pada SVM bertujuan untuk mendapatkan hipotesis berupa bidang pemisah (*hyperplane*) terbaik. *Hyperplane* terbaik tidak hanya dapat memisahkan data tetapi juga memiliki *margin* (jarak) yang paling besar. Data yang berada pada *hyperplane* disebut *support vector*.

Data pada ruang input (*input space*) berdimensi d dinotasikan dengan $x_i = \in \Re^d$ sedangkan label kelas dinotasikan dengan $y_i \in \{-1, +1\}$ untuk i = 1, 2, …n. Dimana n adalah banyaknya data. Diasumsikan kedua kelas -1 dan +1 dapat terpisah secara linear bidang pembatas [25], maka persamaan bidang pembatasnya didefinisikan pada persamaan (2.3) berikut:

$$w.x_i + b = 0 \tag{2.3}$$

Data x_i yang terbagi ke dalam dua kelas, yang termasuk kelas -1 (sampel negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan (2.4) berikut:

$$w.x_i + b < 0 \ \ untuk\ y_i = -1 \tag{2.4}$$

Sedangkan yang termasuk kelas +1 (sampel positif) memenuhi pertidaksamaan (2.5) berikut:

$$w.x_i + b > 0\ untuk\ y_i = +1 \tag{2.5}$$

Dimana:

x_i = data *input*

y_i = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relatif terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya. Bila label data y_i = -1, maka pembatas menjadi persamaan (2.6) berikut:

$$w.x_i + b \leq -1 \tag{2.6}$$

Bila label data y_i = +1, maka pembatas menjadi persamaan (2.7) berikut:

$$w.x_i + b \geq +1 \tag{2.7}$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan *quadratic programming* (QP) *problem* yaitu mencari titik minimal persamaan (2.8) dengan memperhatikan persamaan (2.9) berikut:

$$min\ \tau(w) = \frac{1}{2}||w||^2 \tag{2.8}$$

$$y_i(w * x_i + b) - 1 \geq 0, (i = 1, .., n) \tag{2.9}$$

Permasalahan ini ini dapat dipecahkan dengan berbagi teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan (2.8) ke dalam fungsi *Lagrangian* pada persamaan (2.10), dan menyederhanakannya menjadi persamaan (2.11) berikut:

$$L(w, b, a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i(y_i\big((w^T x_i + b) - 1\big)\big) \tag{2.10}$$

$$L(w, b, a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i(w^T x_i + b) + \sum_{i=1}^{n} a_i \tag{2.11}$$

Dimana a_i adalah *lagrange multiplier* yang bernilai nol atau positif ($a_i \geq 0$). Nilai optimal dari persamaan (2.11) dapat dihitung dengan meminimalkan L terhadap w, b dan a. Dapat dilihat pada persamaan (2.12) sampai (2.14) berikut:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} a_i\, y_i x_i = 0 \tag{2.12}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_i\, y_i = 0 \tag{2.13}$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{n} a_i y_i(w^T x_i + b) - \sum_{i=1}^{n} a_i = 0 \tag{2.14}$$

Maka masalah *Lagrange* untuk klasifikasi dapat dinyatakan pada persamaan (2.15) berikut:

$$Min\ L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i \quad (2.15)$$

Dengan memperhatikan persamaan (2.16) dan (2.17) berikut:

$$w - \sum_{i=1}^{n} a_i\, y_i x_i = 0 \quad (2.16)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.17)$$

Model persamaan (2.15) diatas merupakan model primal *Lagrange.* Sedangkan dengan memaksimalkan L terhadap a_i, persamannya menjadi persamaan (2.18) berikut:

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1,j=1}^{n} a_i a_j y_i y_j x_i x_j \quad (2.18)$$

Dengan memperhatikan persamaan (2.19) berikut:

$$\sum_{i=1}^{n} a_i y_i = 0,\ a_i \geq 0\ (i,j = 1,..,n) \quad (2.19)$$

Untuk mendapatkan nilai a_i, langkah pertama adalah mengubah setiap kalimat menjadi nilai vektor (*support vector*) = $\binom{x}{y}$. Kemudian nilai vektor dari setiap kalimat dimasukkan ke persamaan (2.20) berikut:

$$S_i = \phi\binom{x}{y} = \begin{cases} \sqrt{x^2 + y^2 > 2} \rightarrow \binom{4 - y + (x - y)}{4 - x + (x - y)} \\ \sqrt{x^2 + y^2 \leq 2} \rightarrow \binom{x}{y} \end{cases} \quad (2.20)$$

Nilai x didapatkan dari persamaan (2.21) kernel linear untuk x berikut:

$$\sum_{i=1,j=1}^{n} x_i x_j^T\ , (i,j = 1,..,n) \quad (2.21)$$

Nilai y didapatkan dari persamaan (2.22) kernel linear untuk y berikut:

$$\sum_{i=1,j=1}^{n} y_i y_j^T\ , (i,j = 1,..,n) \quad (2.22)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari substitusi nilai x dan nilai y ke persamaan (2.20) diberi nilai bias = 1 [25]. Kemudian cari parameter a_i, dengan terlebih dahulu mencari nilai fungsi setiap kalimat menggunakan persamaan (2.23), lalu mencari nilai a_i pada persamaan linear menggunakan persamaan (2.24) dengan memperhatikan $i,j = 1,..,n$ berikut:

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (2.23)$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (2.24)$$

Setelah parameter a_i didapatkan, kemudian masukkan ke persamaan (2.25) berikut:

$$\widetilde{W} = \sum_{i=1}^{n} a_i S_i \quad (2.25)$$

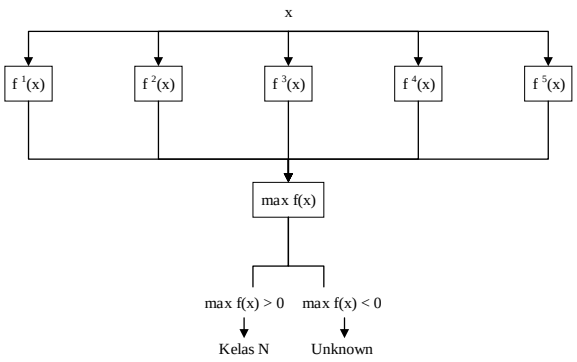
Selanjutnya digunakan persamaan (2.26) untuk mendapatkan nilai w dan b:

$$y = wx + b \quad (2.26)$$

Pada awalnya SVM dikembangkan untuk persoalan klasifikasi dua kelas, kemudian dikembangkan kembali untuk klasifikasi multikelas. Dalam klasifikasi kasus multikelas, *hyperplane* yang terbentuk adalah lebih dari satu. Salah satu metode pendekatan yang digunakan adalah *One Against All* (OAA). Metode OAA untuk kasus klasifikasi *k*-kelas, menemukan *k hyperplane* dimana *k* adalah banyak kelas dan ρ adalah *hyperplane*. Dalam metode ini $\rho^{(\ell)}$ diujikan dengan semua data dari kelas ℓ dengan label +1, dan semua data dari kelas lain dengan label -1 [26]. Berikut ini merupakan ilustrasi untuk persoalan klasifikasi dengan lima buah jumlah kelas, digunakan lima buah SVM biner pada Tabel 2.4 Dan penggunaannya pada pengklasifikasian data baru pada Gambar 2.2 [27].

Tabel 2.4 Contoh Metode One Against All

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Bukan kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan kelas 3	$f^3(x) = (w^3)x + b^3$
Kelas 4	Bukan kelas 4	$f^4(x) = (w^4)x + b^4$
Kelas 5	Bukan kelas 5	$f^5(x) = (w^5)x + b^5$



Gambar 2.2 Metode Klasifikasi SVM One Against All

Konsep pada OAA yaitu dimisalkan pada kasus lima kelas, kelas 1, 2, 3, 4 dan 5. Bila akan diujikan $\rho^{(1)}$, semua data dalam kelas 1 diberi label +1 dan data dari kelas 2, 3, 4 dan 5 diberi label -1. Pada $\rho^{(2)}$, semua data dalam kelas 2 diberi label +1 dan data dari kelas 1, 3, 4 dan 5 diberi label -1. Pada $\rho^{(3)}$, semua data dalam kelas 3 diberi label +1 dan data dari kelas 1, 2, 3 dan 4 diberi label -1. Pada $\rho^{(4)}$, semua data dalam kelas 4 diberi label +1 dan data dari kelas 1, 2, 3, 4 diberi label -1. Begitu juga untuk $\rho^{(5)}$, semua data dalam kelas 5 diberi label +1 dan data dari kelas 1, 2, 3 dan 4 diberi label -1. Kemudian dicari *hyperplane* dengan algoritma SVM dua kelas. Maka akan didapat *hyperplane* untuk masing-masing kelas di atas. Kemudian kelas dari suatu data baru x ditentukan berdasarkan nilai terbesar dari *hyperplane* [26]:

$$\text{kelas x} = \arg \max_{\ell=1\ldots k} \left(\left(w^{(\ell)}\right)^{T}.\phi(x) + b^{(\ell)}\right) \tag{2.27}$$

2.6 Pengukuran Kinerja Klasifikasi

Pengukuran kinerja klasifikasi pada data asli dan data hasil dari model klasifikasi dilakukan dengan menggunakan tabulasi silang (matriks konfusi) yang berisi informasi tentang kelas data asli yang direpresentasikan pada baris matriks dan kelas data hasil prediksi suatu algoritma direpresentasikan pada kolom klasifikasi [6]. Berikut merupakan contoh matriks konfusi:

Tabel 2.5 Matriks Konfusi Lima Kelas

F_{ij}		Kelas Prediksi (j)				
		Kelas 1	Kelas 2	Kelas 3	Kelas 4	Kelas 5
Kelas Asli (i)	Kelas 1	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
	Kelas 2	F_{21}	F_{22}	F_{23}	F_{24}	F_{25}
	Kelas 3	F_{31}	F_{32}	F_{33}	F_{34}	F_{35}
	Kelas 4	F_{41}	F_{42}	F_{43}	F_{44}	F_{45}
	Kelas 5	F_{51}	F_{52}	F_{53}	F_{54}	F_{55}

Menurut [6] ketepatan klasifikasi dapat dilihat dari akurasi klasifikasi. Akurasi klasifikasi menunjukkan performasi model klasifikasi secara keseluruhan, dimana semakin tinggi akurasi klasifikasi hal ini semakin baik performansi model klasifikasi.

$$\text{Akurasi} = \frac{\text{jumlah data yang diprediksi secara benar}}{\text{jumlah prediksi yang dilakukan}}$$

$$= \frac{F_{11}+F_{22}+F_{33}+F_{44}+F_{55}}{F_{12}+F_{13}+F_{14}+F_{15}+F_{21}+F_{23}+F_{24}+F_{25}+F_{31}+F_{32}+F_{34}+F_{35}+F_{41}+F_{42}+F_{43}+F_{45}+F_{51}+F_{52}+F_{53}+F_{54}+F_{15}+F_{11}+F_{22}+F_{33}+F_{44}+F_{55}} \quad (2.28)$$

2.7 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis, serta didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas yang meliputi, rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek. Konsep dasar berorientasi objek diantaranya [28]:

1. Kelas (*Class*) adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.
2. Objek (*Object*) adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3. Metode (*Method*) adalah operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.
4. Atribut (*Attribute*) dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*) merupakan prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapsulasi (*Encapsulation*) adalah pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.
7. Pewarisan (*Inheritance*) adalah mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
8. Antarmuka (*Interface*) sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan. Antarmuka biasanya digunakan agar kelas lain tidak langsung mengakses ke suatu kelas.
9. *Reusability* merupakan pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.
10. Generalisasi dan Spesialisasi menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil dan motor.
11. Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.
12. Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.
13. *Package* adalah sebuah kontainer atau kemasan yang daoat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.8 Unified Modeling Language

UML singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa

diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram*, *activity diagram*, *class diagram* dan *sequence diagram* [28].

2.8.1 Use Case Diagram

Use case diagram merupakan pemodelan untuk tingkah laku (*behavior*) pada sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang terdapat pada sistem. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case diagram* yaitu aktor dan *use case*.

1. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use Case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.8.2 Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis atau menu yang ada pada perangkat lunak. Setiap *use case* yang telah dibentuk digambarkan aktivitasnya dalam *activity diagram*, mulai dari peran aktor, peran sistem, dan *decision*. *Activity diagram* juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan tampilan antarmuka.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.8.3 Class Diagram

Class diagram menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang

dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private,* tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. *Protected,* hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public,* dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class diagram*:

1. *Association*

   Hubungan antar class yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.
2. *Aggregation*

   Relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.
3. *Composition*

   Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.
4. *Dependency*

   Hubungan antar *class* dimana *class* yang memiliki relasi *dependency* menggunakan *class* lain sebagai atribut pada *method.*
5. *Realization*

   Hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

2.8.4 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah

didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan iteraksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.