BAB 2
LANDASAN TEORI

2.1. Peringkasan Teks Otomatis

Peringkasan teks otomatis (*automatic text summarization*) adalah pembuatan bentuk yang lebih singkat dari suatu teks dengan memanfaatkan aplikasi yang dijalankan dan dioperasikan pada komputer. Sedangkan menurut [4], ringkasan adalah teks yang dihasilkan dari sebuah teks atau banyak teks, yang mengandung isi informasi dari teks asli dan panjangnya tidak lebih dari setengah panjang teks aslinya. Secara umum terdapat dua peringkasan yang banyak dikenali yaitu *single-document summarization* dan *multi-document summarization*. Ringkasan *multi-document* merupakan proses penyaringan informasi penting dari beberapa dokumen untuk menghasilkan ringkasan yang singkat. *Ringkasan multi-document* dapat dilihat sebagai perpanjangan dari ringkasan *single-document*. Faktor-faktor yang membuat *multi-document summarization* lebih rumit, antara lain [5]:

1. Setiap artikel yang digunakan ditulis oleh penulis yang berbeda yang mempunyai perbedaan gaya menulis dan struktur dokumen.
2. Beberapa artikel mungkin mempunyai pandangan yang berbeda tentang suatu topik yang sama.
3. Fakta dan sudut pandang dapat berubah kapanpun setiap dokumen dibuat sehingga dapat menimbulkan konflik dalam penyediaan informasi.

2.2. Pendekatan Peringkas Teks

Menurut [6], terdapat dua pendekatan peringkas teks yaitu:

1. Ekstraksi (*extractive summary*)

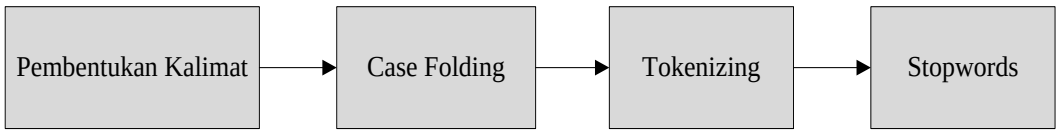
   Pada teknik ekstraksi, sistem menyalin unit-unit teks yang dianggap paling penting dari sebuah teks menjadi ringkasan. Unit-unit teks yang disalin dapat berupa klausa utama, kalimat utama, atau paragraph utama tanpa ada penambahan kalimat-kalimat baru yang terdapat pada dokumen aslinya.
2. Abstraksi (*abstractive summary*)

Teknik abstraksi menggunakan metode *linguistic* untuk memeriksa dan menafsirkan teks dokumen menjadi ringkasan. Ringkasan teks tersebut dihasilkan dengan cara menambahkan kalimat-kalimat baru yang merepresentasikan intisari teks sumber ke dalam bentuk yang berbeda dengan kalimat-kalimat yang ada pada teks sumber.

Pada penelitian ini, metode pendekatan yang digunakan untuk melakukan peringkasan teks otomatis adalah dengan menggunakan teknik ekstraksi (*extractive summary*) dimana hasil dari ringkasan merupakan kalimat asli yang terdapat pada dokumen yang dimasukkan ke dalam sistem dan tanpa mengalami penambahan kalimat.

2.3. *Preprocessing*

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Inputan awal pada proses ini adalah berupa dokumen [4]. *Text preprocessing* yang digunakan pada penelitian ini menggunakan proses pembentukan kalimat, *case folding*, *tokenizing* dan *stopwords*. Tahap *preprocessing* dapat dilihat pada gambar 2.1. berikut:



Gambar 2. 1 Tahap *Preprocessing*

2.3.1. Pembentukan Kalimat

Pembentukan kalimat yaitu proses membentuk string teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Dalam proses untuk mengubah dokumen menjadi kalimat-kalimat, digunakan fungsi split(), dengan tanda titik ".", tanda tanya "?", dan tanda seru "!" sebagai delimiter untuk memotong string dokumen [4].

2.3.2. *Case Folding*

Case folding adalah proses pemrosesan teks dimana semua teks diubah ke dalam case yang sama. Pada kasus ini semua teks diubah representasinya ke dalam huruf kecil (*lowercase*) [7].

2.3.3. *Tokenizing*

Pada proses *tokenizing*, kata-kata yang ada di dalam dokumen harus dipecah-pecah terlebih dahulu menjadi bagian-bagian yang lebih kecil berupa kata tunggal yang memiliki arti atau biasa disebut token. Sebelum dilakukan *tokenizing* terlebih dahulu dilakukan proses pengubahan huruf-huruf yang ada di dalam dokumen menjadi huruf kecil (*case folding*) serta dilakukan penghilangan tanda baca. Hal ini perlu dilakukan terlebih dahulu untuk mempermudah proses lebih lanjut [4].

2.3.4. *Stopwords*

Text filtering bertujuan untuk mengambil kata-kata yang dapat merepresentasikan isi dokumen dengan cara membuang kata-kata yang dianggap tidak penting yang biasa disebut *stopword*. *Stopwords* dapat berupa kata sambung, kata depan, dan kata seru seperti “di”, “yang”, “dan”, “ke”, “wah”, “serta”, “wow” dan lain-lain [4].

2.4. Ekstraksi Fitur

Untuk menggunakan metode statistik, kalimat diwakili sebagai vektor fitur [7]. Fitur-fitur ini adalah atribut yang digunakan untuk mewakili data yang digunakan dalam tugasnya. Kata kunci digunakan untuk mengekstrak fitur, dimana fitur merupakan hasil ekstraksi yang diasumsikan memberikan informasi mengenai kalimat [1]. Pada penelitian ini digunakan 8 fitur, dimana fitur-fitur tersebut telah digunakan sebelumnya pada penelitian lain seperti pada penelitian [2], [7] dan [8]. Delapan fitur yang dijelaskan di bawah ini akan digunakan untuk proses klasifikasi menggunakan *support vector machine* (SVM).

2.4.1. Fitur Panjang Kalimat (F_1)

Fitur ini digunakan untuk menghitung panjang kalimat, dimana Fitur ini dihitung dengan membagi jumlah kata-kata dalam kalimat terhadap jumlah kata dari kalimat terpanjang pada setiap artikel. Perhitungan dari fitur panjang kalimat bisa dilihat pada persamaan berikut [2]:

$$Scoref1(s_i) = \frac{Jumlah\ kata\ yang\ terdapat\ pada\ kalimat}{jumlah\ kata\ yang\ terdapat\ pada\ kalimat\ terpanjang} \quad (2.1)$$

2.4.2. Fitur Posisi Kalimat (F_2)

Fitur ini merepresentasikan kalimat pertama dan terakhir pada setiap artikel hasil *preprocessing* merupakan kalimat yang penting [7], maka nilai fitur untuk setiap kalimat pertama dan terakhir pada setiap artikel akan bernilai 1 dan kalimat lain akan bernilai 0.

2.4.3. Fitur Data Numerik (F_3)

Biasanya kalimat yang mengandung data numerik merupakan kalimat penting dan biasanya kalimat tersebut masuk ke dalam ringkasan. Fitur data numerik dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref3(s_i) = \frac{jumlah\ data\ numerik}{panjang\ kalimat(jumlah\ kata\ pada\ kalimat)} \quad (2.2)$$

2.4.4. Fitur Kata Thematik Dalam Kalimat (F_4)

Fitur ini menghitung kemunculan relatif kata kunci pada suatu kalimat, biasanya kalimat yang memiliki relatif kata kunci yang baik merupakan kalimat ringkasan [7]. Fitur kata-kata thematik dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref4(s_i) = \frac{jumlah\ kata\ kunci\ pada\ kalimat}{panjang\ kalimat\ (jumlah\ kata\ pada\ kalimat)} \quad (2.3)$$

2.4.5. Fitur Kalimat Yang Menyerupai Judul (F_5)

Kalimat yang menyerupai judul adalah kalimat yang memiliki *vocabulary overlap* antara kalimat dengan judul. Fitur kalimat yang menyerupai judul dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref5(s_i) = \frac{jumlah\ kata\ pada\ judul}{jumlah\ kata\ yang\ sama\ pada\ judul} \quad (2.4)$$

2.4.6. Fitur Kemiripan Kalimat Dengan Kumpulan Kalimat Lain (F_6)

Kemiripan kalimat dapat dilihat dari *vocabulary overlap* antara kalimat dengan kalimat yang lain, untuk mempermudah maka kata yang dilihat hanyalah kata kunci. Fitur kemiripan kalimat dengan kumpulan kalimat lain dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref6(s_i) = \frac{kata\ kunci\ dalam\ kalimat \cap kata\ kunci\ dalam\ kalimat\ lain}{kata\ kunci\ dalam\ kalimat \cup kata\ kunci\ dalam\ kalimat\ lain} \quad (2.5)$$

2.4.7. Fitur Ikatan Leksikal Dengan Kalimat Sebelumnya (F_7)

Ikatan leksikal antara kalimat dengan kalimat sebelumnya didefinisikan sebagai kata (*stem*) yang muncul dalam kedua kalimat tersebut, nilai akan 1 apabila memiliki hubungan leksikal, akan bernilai 0 jika tidak punya.

2.4.8. Fitur Ikatan Leksikal Dengan Kalimat Sesudahnya (F_8)

Ikatan leksikal antara kalimat dengan kalimat sesudahnya didefinisikan sebagai kata (*stem*) yang muncul dalam kedua kalimat tersebut, nilai 1 apabila memiliki hubungan leksikal, dan akan bernilai 0 jika tidak punya.

2.5. *Support Vector Machine* (SVM)

Metode klasifikasi yang banyak dikembangkan dan diterapkan adalah *support vector machine* (SVM) [9]. Metode ini berakar dari teori pembelajaran statistik yang hasilnya sangat menjanjikan untuk memberikan hasil yang lebih baik daripada metode yang lain. SVM juga bekerja dengan baik pada set data berdimensi tinggi, bahkan SVM yang menggunakan teknik kernel harus memetakan data asli dari dimensi asalnya menjadi dimensi lain yang relatif lebih tinggi. Hal ini menjadi kelebihan dari SVM karena tidak semua data latih akan dipandang untuk dilibatkan dalam setiap iterasi pelatihannya. Data-data yang berkontribusi tersebut disebut *support vector* sehingga metodenya disebut *support vector machine.*

2.5.1. Ide Dasar *Support Vector Machine*

Ide dasar SVM adalah memaksimalkan batas hyperplane (*maximal margin hyperplane*) [9]. Dalam teknik ini, SVM akan berusaha untuk menemukan fungsi

pemisah (*classifier*) yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda [10]. Dalam hal ini fungsi pemisah yang dicari adalah fungsi linier. Fungsi ini bisa didefinisikan sebagai:

$$g(x) \coloneqq sgn(f(x)) \tag{2.6}$$

Dengan $f(x) = w^T x + b,\ x,\ w \in R^n$ dan $b \in R$. Masalah klasifikasi ini bisa dirumuskan untuk menemukan set parameter (w, b) sehingga:

$$sgn\big(f(x_i)\big) = sgn(< w, x > +b) = y_i \tag{2.7}$$

Fungsi sgn digunakan untuk mengelompokkan semua nilai di atas 0 menjadi +1 dan dibawah 0 menjadi -1. Dalam teknik ini SVM berusaha menemukan fungsi pemisah (*classifier*/*hyperplane*) terbaik diantara fungsi yang tidak terbatas jumlahnya untuk memisahkan dua macam objek [10]. *Hyperplane* terbaik adalah *hyperplane* yang terletak ditengah-tengah antara dua set obyek dari dua kelas. Mencari *hyperplane* terbaik ekuivalen dengan memaksimalkan margin atau jarak antara dua set objek dari dua kelas berbeda. Secara spesifik, margin dihitung dengan cara sebagai berikut:

$$wx_1 + b = +1 \tag{2.8}$$

$$wx_2 + b = -1 \tag{2.9}$$

$$w(x_1 - x_2) = 2 \tag{2.10}$$

Untuk membuktikan bahwa memaksimalkan margin antara dua set obyek akan meningkatkan probabilitas pengelompokkan secara benar dari data *testing*. Pada dasarnya jumlah fungsi pemisah ini tidak terbatas banyaknya. Misalkan dari jumlah yang tidak terbatas ini diambil dua saja, yaitu $f_1(x)$ dan $f_2(x)$. Setelah menemukan dua fungsi ini, suatu data baru masuk dengan keluaran -1. Kemudian SVM akan mengelompokkan apakah data yang baru akan masuk ke dalam kelas -1 atau +1 menggunakan fungsi pemisah yang sudah ditemukan.

2.5.2. Formulasi Matematis

Secara matematika, formulasi problem optimisasi SVM untuk kasus klasifikasi linier di dalam *primal space* adalah:

$$min_{w,b} \frac{1}{2}\|w\|^2 \quad (2.11)$$

Syarat : $y_i(wx_i + b) \geq 1, i = 1, \dots, N$

Dimana x_i adalah input, y_i adalah output yang nilainya = 1 atau -1. w dan b adalah parameter yang akan dicari nilainya. Dalam persamaan (2.11), minimalkan fungsi tujuan (*obyektif function*). $\frac{1}{2}\|w\|^2$ atau memaksimalkan kuantitas $\frac{1}{\|w\|}$ atau $w^T w$ dengan memperhatikan pambatas :

$$y_i(wx_i + b) \geq 1 \quad (2.12)$$

Bila output data $y_i = +1$ maka pembatas menjadi :

$$(wx_i + b) \geq 1 \quad (2.13)$$

Sebaliknya jika $y_i = -1$ maka pembatas menjadi:

$$(wx_i + b) \leq -1 \quad (2.14)$$

Di dalam kasus yang tidak *feasible* (*infeasible*) dimana beberapa data mungkin tidak bisa dikelompokkan secara benar, formulasi matematikanya menjadi:

$$min_{w,b,t} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi i \quad (2.15)$$

Syarat : $y_i(wx_i + b) \geq 1, i = 1,2, \dots, N$

Dalam formulasi ini SVM akan berusaha meminimalkan kesalahan klasifikasi yang dinyatakan dengan adanya variabel *slack*. Penggunaan variabel *slack* adalah untuk mengatasi kasus ketidaklayakan (*infeasibility*) dari pembatas (*constraints*) pada persamaan (2.12) degan cara memberikan pinalti dengan menerapkan parameter C. Vektor w tegak lurus terhadap fungsi pemisah :

$$wx + b = 0 \quad (2.16)$$

Konstanta b menentukan lokasi fungsi pemisah relatif terhadap titik asal. Persamaan (2.15) adalah programa nonlinier. Ini bisa dilihat dari fungsi tujuan yang berbentuk kuadrat. Untuk membuat masalah ini lebih mudah dan efisien untuk diselesaikan, masalah ini bisa ditransformasikan ke dalam *dual space*. Untuk itu, persamaan (2.15) menjadi fungsi Lagrangian:

$$J(w,b,\alpha) = \frac{1}{2}w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1\right] \quad (2.17)$$

Dimana variabel α_i tidak boleh bernilai negatif. Fungsi ini harus diminimalkan terhadap variabel w dan b dan harus dimaksimalkan terhadap variabel α. Kemudian kita cari turunan pertama dari fungsi $J(w,b,\alpha)$ terhadap variabel w dan b dan disamakan dengan 0. Dengan melakukan proses ini, akan didapatkan dua kondisi optimalitas berikut:

Syarat 1:

$$\frac{\partial J(w,b,\alpha)}{\partial w} = 0 \quad (2.18)$$

Syarat 2:

$$\frac{\partial J(w,b,\alpha)}{\partial b} = 0 \quad (2.19)$$

Penerapan syarat 1 pada fungsi Lagrangian (2.17) akan menghasilkan:

$$w = \sum_{i=1}^{N} \alpha_i \, y_i x_i \quad (2.20)$$

Penerapan syarat 2 pada fungsi Lagrangian (2.17) akan menghasilkan:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (2.21)$$

Menurut [10] *duality theorem*:

1. Jika problem primal mempunyai solusi optimal, maka problem dual juga akan mempunyai solusi optimal yang nilainya sama.
2. Bila w_o adalah solusi optimal untuk problem primal dan α_0 untuk problem dual, maka perlu dan cukup bahwa w_o solusi layak untuk problem primal dan persamaan:

$$\Phi(w_0) = J(w_0, b_0, \alpha_0) = min_w J(w, b, \alpha) \quad (2.22)$$

Untuk mendapatkan problem dual, dijabarkan persamaan (2.17) sebagai berikut:

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b \sum_{i=1}^{N} \alpha_i y_i - \sum_{i=1}^{N} \alpha_i \quad (2.23)$$

Menurut syarat (2.21) term ketiga sisi sebelah kanan dalam persamaan tersebut =0. Dengan memakai nilai-nilai w pada persamaan (2.20) akan didapatkan:

$$w^T w = \sum_{i=1}^{N} \alpha_i y_i w^T x_i = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (2.24)$$

Maka persamaan (2.23) menjadi:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (2.25)$$

Selanjutnya akan didapat formulasi dual dari persamaan (2.15) menjadi:

$$max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (2.26)$$

Syarat:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \, , i = 1, .. , n \quad (2.27)$$

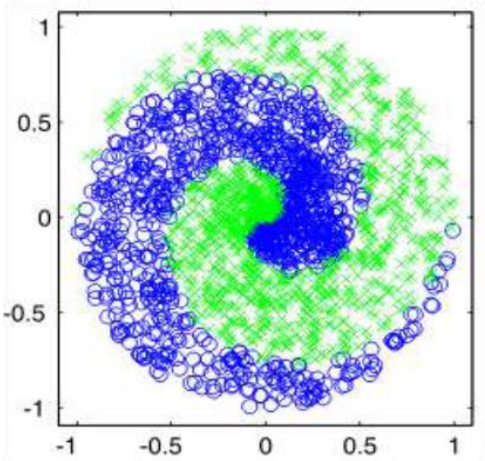
Persamaan (2.27) adalah *Quadratic Programming* (*QP*) dengan pembatas (*constraint*) linier. Melatih SVM ekuivalen dengan menyelesaikan problem *convex optimization*. Karena itu solusi dari SVM adalah unik dan global optimal. Hal ini berbeda dengan solusi *neural networks* yang ekuivalen dengan problem *nonconvex optimization* dengan akibat solusi yang ditemukan *local* optima [10].

2.5.3. Metoda Kernel

Banyak teknik data mining atau *machine learning* yang dikembangkan dengan asumsi kelinieran. Sehingga algoritma yang dihasilkan terbatas untuk kasus-kasus yang linier. Karena itu bila suatu klasifikasi memperlihatkan ketidaklinieran, algoritma *perceptron* tidak bisa mengatasinya. Metoda kernel adalah salah satu untuk mengatasinya [10]. Dengan metoda kernel, suatu data x pada input space di mapping ke *feature space* F dengan dimensi yang lebih tinggi melalui map φ sebagai berikut:

$$\varphi : x \rightarrow \varphi(x) \quad (2.28)$$

Karena itu data x diinput space menjadi $\varphi(x)$ di *feature space.* Data spiral yang menggambarkan ketidaklinieran dapat dilihat pada Gambar 2.2 berikut:



Gambar 2. 2 Data Spiral Yang Tidak Linier [10]

Seringkali fungsi φ(x) tidak tersedia atau tidak bisa dihitung, tetapi *dot product* dari dua vektor dapat dihitung baik di dalam *input space* maupun di *feature space*. Dengan kata lain, sementara φ(x) mungkin tidak diketahui, *dot product* $< \varphi(x_1), \varphi(x_2) >$ masih bisa dihitung di *feature space*. Untuk bisa memakai metoda kernel fungsi tujuan dari pembatas perlu diekspresikan dalam bentuk *dot product* dari vektor data x_i. Suatu fungsi kernel $K(x, x')$ bisa digunakan untuk mengganti dot product $< \varphi(x), \varphi(x)' >$. Kemudian di *feature space*,bisa menemukan fungsi pemisah yang linier yang mewakili fungsi nonlinier di *input space*. Gambar 2.2 mendeskripsikan suatu contoh *feature mapping* dari ruang dua dimensi ke *feature space* dua dimensi. Dalam *input space*, data tidak bisa dipisahkan secara linier, tetapi dalam *feature space* bisa. Karena itu dengan memetakan data ke *feature space* menjadikan tugas klasifikasi menjadi lebih mudah. Dengan memetakan setiap data ke *feature space*, maka persamaan (2.26) akan menjadi:

$$max \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j) \qquad (2.29)$$

Syarat:

$$\sum_{i=1}^{N} \alpha_i y_i = 0, i = 1, .., N$$

Dalam persamaan (2.29), *dot product* antara $\varphi(x_i)^T\varphi(x_j)$ bisa diwakili oleh $K(x_i,x_j)$ dimana K adalah fungsi kernel. Selanjutnya setelah SVM dilatih dan ditemukan nilai α dan b maka akan didapatkan persamaan :

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i,x) + b \quad (2.30)$$

Fungsi pemisah optimal adalah :

$$g(x) = sign(\sum_{i=1}^{N} \alpha_i y_i K(x_i,x) + b \quad (2.31)$$

Dimana $\alpha_i = 1,..,N$ adalah solusi optimal dari persamaan (2.29) dan b dipilih sehingga $y_i f(x_i) = 1$ untuk sembarang i dengan $C > \alpha_i > 0$.

Data x_i dimana $\alpha_i > 0$ dinamakan *support vector* dan menyatakan data training yang diperlukan untuk mewakili fungsi keputusan yang optimal. Secara detail nilai b dihitung dengan cara:

$$b = \frac{1}{nsv}\sum_{sv}(y_{sv} - K(x_{sig}, x_{sv}) * \alpha_{sv} * y_{sig} \quad (2.32)$$

Dimana:

nsv : banyaknya *support vector*

sv : indeks untuk *support vector*

sig : data yang berada dalam margin, dimana $0 < \alpha_i < C$

Untuk mengatasi masalah ketidaklinieran yang sering terjadi dalam kasus nyata, maka bisa diterapkan metode kernel. Fungsi kernel yang bisanya dipakai dalam SVM [9], yaitu:

Tabel 2. 1 Fungsi Kernel

No	Kernel	Fungsi Kernel
1	Linier	$K(x,y) = x.y$
2	Polynominal	$K(x,y) = (x \cdot y + c)^d$
3	Gaussian RBF	$K(x,y) = exp(-\gamma\|x - x_i\|^2$
4	Sigmoid	$K(x,y) = \tanh(\sigma(x \cdot y) + c)$

5	Invers Multikuadrik	$K(x,y) = \frac{1}{\sqrt{\|x-y\|^2 + c^2}}$

Dimana x dan y adalah pasangan dua data dari semua bagian data latih. Parameter $\sigma, c, d > 0$ merupakan konstanta. Dalam fungsi tersebut, ada beberapa parameter yang ditentukan oleh *user*. Misalnya dalam kernel RBF, nilai γ ditentukan oleh *user* dan bisa diberi nilai riil positif.

2.5.4. Klasifikasi Dengan *Support Vector Machine*

Operasi logika AND merupakan contoh pemetaan data yang pasti linier dengan distribusi data yang jelas dan sederhana [9]. Pertama, jangkauan set data dan kelas yang digunakan dikonversi agar sesuai dengan format SVM, karena SVM mensyaratkan bahwa kelas yang digunakan menggunakan nilai -1 dan +1. Berikut pada Tabel 2.2 merupakan contoh yang sesuai dengan notasi SVM :

Tabel 2. 2 Operasi Logika AND

x_1	x_2	Kelas (y)
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Contoh masalah operasi logika AND merupakan masalah linier sehingga tidak diperlukan fungsi kernel didalamnya. Pada tabel 2.2 karena ada dua fitur ($x_1$dan x_2), w juga akan memiliki dua fitur (w_1 dan w_2). Maka formulasi yang digunakan adalah sebagai berikut:

Minimalkan $\frac{1}{2}({w_1}^2 + {w_2}^2)$

Syarat : $y_i(w \cdot x_i) + b) \geq 1, i = 1, .., N$

1. $(w_1 + w_2 + b) \geq 1$, untuk $y_1 = 1;\ x_1 = 1;\ x_2 = 1$
2. $(-w_1 + w_2 - b) \geq 1$, untuk $y_2 = -1;\ x_1 = 1;\ x_2 = -1$

3. $(w_1 - w_2 - b) \geq 1$, untuk $y_3 = -1;\ x_1 = -1;\ x_2 = 1$
4. $(w_1 + w_2 - b) \geq 1$, untuk $y_4 = -1;\ x_1 = -1;\ x_2 = -1$

Dengan menjumlahkan persamaan pada syarat (1) dan (2) didapatkan $w_2 = 1$ dengan menjumlahkan persamaan pada syarat (1) dan (3) didapatkan $w_1 = 1$. Dengan menjumlahkan persamaan syarat (2) dan (3) didapatkan $b = -1$ sehingga persamaan hiperbolik yang didapatkan adalah:

$$f(x) = w_1 x_1 + w_2 x_2 + b$$

$$f(x) = x_1 + x_2 - 1$$

Operasi logika XOR merupakan contoh pemetaan data yang tidak linier dengan distribusi data yang jelas dan sederhana. Pertama, jangkauan set data dan kelas yang digunakan dikonversi agar sesuai dengan format SVM, karena SVM mensyaratkan bahwa kelas yang digunakan menggunakan nilai -1 dan +1. Berikut pada Tabel 2.3 merupakan contoh yang sesuai dengan notasi SVM:

Tabel 2. 3 Operasi Logika XOR

x_1	x_2	Kelas (y)
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

Formulasi yang digunakan adalah dualitas Lagrange Multiplier sehingga w harus menggunakan persamaan $w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)$. Kemudian perlu melakukan kernelisasi sehingga bisa didapatkan fungsi pemisah linier di dalam feature space. Untuk itu digunakan fungsi kernel polynominal pangkat 2 yang didefinisikan sebagai $K(x_i, x_i) = (x_i x'_i + 1)^2$. Dengan kernel ini perhitungan untuk matriks K (1,1) dan K(1,2) sebagai berikut:

$x^1 = [1 \quad 1]$ (nilai x pada pengamatan ke-1)

$x^2 = [-1 \quad 1]$ (nilai x pada pengamatan ke-2)

$$x^1x^1 = [1 \quad 1]\begin{bmatrix}1\\1\end{bmatrix} = 2; (x^1x^1 + 1)^2 = 9$$

$$x^1x^2 = [1 \quad 1]\begin{bmatrix}-1\\1\end{bmatrix} = 0; (x^1x^2 + 1)^2 = 1$$

Dengan prosedur yang sama untuk semua nilai x^1, akan didapatkan nilai matriks K sebagai berikut:

$$K = \begin{bmatrix}9 & 1 & 1 & 1\\1 & 9 & 1 & 1\\1 & 1 & 9 & 1\\1 & 1 & 1 & 9\end{bmatrix}$$

Dengan menggunakan matriks K sebagai pengganti dot product $\varphi(x_i)^T\varphi(x_j)$ maka akan didapatkan formulasi matematis sebagai berikut:

$$\max a_1 + a_2 + a_3 + a_4 - \frac{1}{2}(9{a_1}^2 - 2a_1a_2 - a_1a_3 + a_1a_4 + 9{a_2}^2 + 2a_2a_3 - a_2a_4 + 9{a_3}^2 - 2a_3a_4 + 9{a_4}^2)$$

Syarat :

$$-a_1 + a_2 + a_3 - a_4 = 0$$

$$a_1 \geq 0$$

Dalam fungsi tujuan diatas *term* kedua telah dikalikan dengan y_iy_j dan telah memenuhi persamaan *Quadratic Programming*. Penyelesaian problem tersebut memberikan hasil sebagai berikut:

$$a_1 = a_2 = a_3 = a_4 = 0.125$$

Hasil ini menunjukkan bahwa semua data merupakan *support vector* karena $a \neq 0$. Setelah proses SVM *training* dan nilai a telah didapatkan kita bisa menentukan label untuk data *testing* dengan menggunakan fungsi pemisah sebagai berikut:

$$f(x) = y\ a^T K(x_{test}.x_{train}) + b \qquad (2.33)$$

Dimana vektor a dan konstanta b diketahui dari hasil *training*.

2.6. *Maximal Marginal Importance* (MMI)

Maximal marginal importance (MMI) adalah modifikasi metode *maximal marginal relevance* (MMR) [1]. MMR merupakan suatu metode perangkingan dokumen teks yang bergantung pada kesamaan antar frasa kata dengan *query* yang dimasukkan oleh *user* ke dalam sistem ringkasan. Maka metode MMR ini bersifat *query based*. Sedangkan metode MMI bersifat general. Metode ini tidak memerlukan inputan *query* dari *user*.

2.6.1. Pemobotan Unit atau Fitur Kalimat

Untuk mendapatkan unit-unit teks yang paling informatif, biasanya dilakukan pemberian skor atau pembobotan nilai terhdap unit-unit teks yang dianggap penting tersebut. Bobot kalimat adalah nilai sebuah kalimat yang dikalkulasi dari nilai unit-unit teks di dalamnya yang mengindikasikan seberapa penting arti kalimat tersebut [1]. Fitur kalimat yang digunakan untuk melakukan proses pembentukan kalimat menggunakan MMI adalah sebagai berikut:

2.6.1.1 *Word Sentence Score* (WSS)

WSS adalah formula pemberian bobot nilai yang menerapkan metode TF-ISF (*Term Frequency- Inverse Sentence Frequency*). TF (*Term Frequency*) menyatakan frekuensi kemunculan kata dalam kalimat dan SF (*Sentence Frequency*) menyatakan jumlah kalimat yang mengandung kata tersebut [1]. Untuk menghitung nilai sutu kata (term) pada suatu kalimat teks sumber dapat dihitung menggunakan persamaan berikut:

$$w_{ij} = tf_{ij} \times isf = tf(t_{ij}, s_i)\left[1 - \frac{\log(sf(t_{ij})+1}{\log(n+1)}\right] \quad (2.34)$$

Keterangan :

w_{ij} = bobot nilai suatu kata /*term* t_{ij} dalam suatu kalimat teks sumber

t_{ij} = suatu *term* pada kalimat teks sumber

$tf(t_{ij}, s_i)$ = frekuensi kemunculan *term* t_{ij} dalam kalimat s_i

$sf(t_{ij})$ = jumlah kalimat yang mengandung *term* t_{ij}

n = jumlah kalimat keseluruhan kalimat dalam teks sumber

Setelah didapat bobot nilai term t_{ij} pada kalimat s_i, maka kalkulasi bobot nilai kalimat (WSS) s_i tersebut dengan persamaan berikut:

$WSS(s_i) = 0.1 + \frac{\sum tj \in s_i w_{ij}}{HTFS}$, jumlah kalimat yang mengandung $t_j \geq \frac{1}{2} LS$ (2.35)

Keterangan :

$WSS(s_i)$ = bobot nilai kalimat s_i dalam teks sumber berdasarkan frekuensi *term*.

LS = jumlah/panjang kalimat ringkasan

t_j = term pada kalimat s_i yang memenuhi syarat yaitu kalimat dalam teks sumber yang mengandung lebih dari 1 atau = LS.

w_{ij} = bobot nilai suatu term t_{ij} pada kalimat s_i

HTFS = nilai w_{ij} paling besar yang terpilih dari keseluruhan w_{ij} term yang terdapat pada kalimat s_i.

Bobot nilai 0,1 merupakan nilai minimum yang didapatkan sebuah kalimat jika term-term pada kalimat tersebut tidak penting.

2.6.1.2 *Keyword Feature*

Sepuluh term dengan w_{ij} yang paling besar dari seluruh kalimat teks sumber dipilih menjadi *keyword*. Kalimat yang mengandung keyword tersebut akan diberi nilai 0,1 setiap *keyword*nya.

2.6.1.3 *Nfriends Feature*

$Nfriends(s_i, s_j)$ mengukur keterhubungan antara kalimat s_i dan s_j. s_i $friends$ ditentukan dengan menghitung nilai *similarity* antar s_i dan kalimat lainnya yang terdapat pada teks sumber (s_w). Jika nilai similarity $(s_i, s_w) \geq threshold$, maka s_w merupakan s_i *friends.* Begitu pula dengan kalimat s_j, s_j friends ditentukan dengan menghitung nilai similarity antar s_j dengan kalimat lainnya yang terdapat pada teks sumber (s_w). Jika nilai similarity

$(s_i, s_w) \geq threshold$, maka s_w merupakan s_j friends. Dan untuk menghitung nilai *Nfriends* antara kalimat s_i dan s_j digunakan persamaan berikut:

$$Nfriends(s_i, s_j) = \frac{s_i(friends) \cap s_j(friends)}{|s_i(friends) \cup s_j(friends)|} \quad \text{, dimana } i \neq j \qquad (2.36)$$

2.6.1.4 *Ngrams Feature*

Ngrams (s_i, s_j) merupakan suatu metode yang menghitung nilai kesamaan antara n kalimat s_i dan s_j. Ngrams melakukan pemisahan per n kata dari suatu kalimat. *Ngrams* pada penelitian ini akan diinisialisasi 2 [1], sehingga persamaan (2.37) menghitung nilai kesamaan skip-bigram antara dua kalimat. Contoh *skip-bigram* suatu kalimat dapat dilihat pada tabel 2.4. berikut:

Kalimat : saya belajar bahasa Indonesia

Skip-bigram :

Tabel 2. 4 Contoh Skip-Bigram

saya belajar	saya bahasa	saya Indonesia
belajar bahasa	belajar Indonesia	bahasa Indonesia

Untuk menghitung kesamaan *Ngrams* antara kalimat s_i dan s_j digunakan persamaan berikut:

$$Ngrams(s_i, s_j) = \frac{s_i(ngrams) \cap s_j(ngrams)}{|s_i(ngrams) \cup s_j(ngrams)|} \quad \text{, dimana } i \neq j \qquad (2.37)$$

2.6.1.5 *The Simililarity to First Sentence* (sim_fsd)

Fitur ini memberikan nilai kepada kalimat berdasarkan nilai similarity kalimat tersebut dengan kalimat pertama pada teks sumber. Karena pada artikel berita kalimat pertama dianggap kalimat penting.

$$simFSD(s_i) = sim(s_i, s_0) \qquad (2.38)$$

Dimana s_0 merupakan kalimat pertama pada teks sumber.

2.6.1.6 *Sentence Centrality* (SC)

Fitur ini dihitung berdasarkan 3 faktor, yaitu *similarity*, fitur *Nfriends*, dan fitur *Ngrams*. Nilai *similarity* dihitung menggunakan persamaan 2.39 berikut:

$$sim(s_i,s_j) = \frac{\sum w_i \in s_i,s_j tf(w_i,s_i)tf(w_i,s_j)\left[1-\frac{\log(sf(w_i)+1)}{\log n+1}\right]^2}{\sqrt{\sum w_i \in s_i(tf(w_i,s_i).\left[1-\frac{\log(sf(w_i)+1)}{\log n+1}\right]^2} \times \sqrt{\sum w_i \in s_j tf(w_i,s_j).\left[1-\frac{\log(sf(w_i)+1)}{\log n+1}\right]^2}} \quad (2.39)$$

Sentence centrality dihitung menggunakan persamaan berikut:

$SC(s_i) = \frac{\sum_{j=1}^{n-1} sim(s_i,s_j)+\sum_{j=1}^{n-1} nfriends(s_i,s_j)+\sum_{j=1}^{n-1} ngrams(s_i,s_j)}{n-1}$ dimana

$$i \neq j \; and \; sim(s_i,s_j) \geq 0 \quad (2.40)$$

Keterangan :

$SC(s_i)$ = bobot nilai SC kalimat s_i dalam teks sumber

$sim(s_i,s_j)$ = nilai *similarity* antara kalimat s_i dan s_j

$Nfriends(s_i,s_j)$ = nilai *nfriends* antara kalimat s_i dan s_j

$Ngrams(s_i,s_j)$ = nilai *ngrams* antara kalimat s_i dan s_j

0 = *threshold*

n = jumlah kalimat

2.6.1.7 *Tittle Help Sentence* (THS)

THS adalah suatu formula untuk menghitung nilai kesamaan *ngrams* antara kalimat s_i dengan judul teks. THS dihitung menggunakan persamaan berikut:

$$THS(s_i) = \frac{s_i(ngrams) \cap T(ngrams)}{|s_i(ngrams) \cup T(ngrams)|} \quad (2.41)$$

2.6.1.8 *Tittle Help Sentence Relevance Sentence* (THSRS)

Fitur ini digunakan untuk menghitung nilai kesamaan ngrams antara kalimat s_i dengan kalimat yang mempunyai THS paling besar yang telah dihitung sebelumnya menggunakan persamaan (2.38). THSRS dihitung menggunakan persamaan:

$$THSRS(s_i) = \frac{s_j(ngrams) \cap THS(s_j(ngrams))}{|s_i(ngrams) \cup THS(s_j(ngrams))|} \quad (2.42)$$

Kemudian dihitung juga nilai SS_NG suatu kalimat s_i yang merupakan nilai rata-rata dari nilai $THS(s_i)$ dan $THSRS(s_i)$ menggunakan persamaan berikut:

$$SS_NG_{(s_i)} = \frac{THS(s_i)+THSRS(s_i)}{2} \quad (2.43)$$

2.6.2. Bobot Nilai *Importance* dan *Relevance* Kalimat

Bobot nilai importance adalah nilai kepentingan suatu kalimat [1]. Yang menjadi ide utama dari metode yang digunakan. Nilai importance suatu kalimat dihitung menggunakan persamaan berikut:

$$IMPR(s_i) = avg(WSS(s_i) + SC(s_i) + SS_{NG(s_i)} + simFSD(s_i) + kwrd(s_i)) \quad (2.44)$$

Keterangan:

$WSS(s_i)$ = bobot nilai kalimat s_i dalam teks sumber berdasarkan frekuensi *term*.

$SC(s_i)$ = bobot nilai SC kalimat s_i dalam teks sumber

$SS_NG_{(s_i)}$ = bobot nilai SS_NG kalimat s_i dalam teks sumber

$simFSD(s_i)$ = bobot nilai simFSD kalimat s_i dalam teks sumber

$kwrd(s_i)$ = bobot nilai *keyword* kalimat s_i dalam teks sumber

Bobot nilai *relevance* adalah nilai keterhubungan suatu kalimat dengan kalimat lainnya yang terdapat pada teks sumber. Dihitung dengan menggunakan persamaan berikut:

$$Rel(s_i, s_j) = avg(Nfriends(s_i, s_j) + Ngrams(s_i, s_j) + sim(s_i, s_j)) \quad (2.45)$$

Keterangan:

$sim(s_i, s_j)$ = nilai *similarity* antara kalimat s_i dan s_j

$Nfriends(s_i, s_j)$ = nilai *nfriends* antara kalimat s_i dan s_j

$Ngrams(s_i, s_j)$ = nilai *ngrams* antara kalimat s_i dan s_j

Sebelum dilakukan perhitungan *maximal marginal importance*, terlebih dahulu dicari nilai $ScoreBT(s_i)$ dengan menggunakan persamaan berikut:

$$ScoreBT(s_i) = impr(s_i) + (1 - (1 - impr(s_i) \times friendsNo(s_i))) \quad (2.46)$$

Keterangan:

$impr(s_i)$ = nilai importance kalimat s_i

$friendsNo(s_i)$ =jumlah friens pada kalimat s_i

Setelah didapatkan nilai kalimat $impr(s_i)$ nilai MMI dihitung menggunakan persamaan berikut:

$$MMI(s_i) = Arg\max_{si \in cs \setminus ss}[(ScoreBT(s_i) - \beta(s_i)) - max_{sj \in ss}(Rel(s_i, s_j))] \quad (2.47)$$

Keterangan:

s_i = kalimat indeks ke-i yang merupakan naggota kluster

s_j = kalimat indeks ke-j yang juga merupakan anggota kluster yang sama dengan s_i

CS = kalimat kompetitif yang berada dalam 1 pohon biner

SS = kalimat-kalimat yang terpilih dalam satu kluster

$Rel(s_i, s_j)$ = nilai relevance antara kalimat s_i dan s_j

$\beta(s_i)$ = nilai penalty yang dinisialisasi 0.01*level kalimat pada pohon biner

2.7. Evaluasi Ringkasan

Metode untuk mengevaluasi hasil ringkasan merupakan topik yang cukup sulit, baik evaluasi terhadap ringkasan yang dihasilkan dari mesin peringkas otomatis ataupun ringkasan manual dibuat oleh *abstractor* yang professional, dikarenakan tidak terdapat definisi ringkasan ideal. Terdapat dua klasifikasi metode evaluasi, yaitu:

a. *Ekstrinsik*, yaitu kualitas ringkasan diukur berdasarkan bagaimana ini membantu penyelesaian tugas *user*.
b. *Instrinsik*, yaitu hanya diukur dari kualitas hasil (*output*) ringkasan yang dihasilkan.

Evaluasi sistem peringkas yang ada saat ini adalah *intrinsik*. Pengevaluasi menciptakan sekumpulan ringkasan yang manual, masing-masing satu untuk menguji teks. Kemudian membandingkan hasil ringkasan sistem dengan ringkasan ideal. Yang diukur adalah *overlap* dari isi yang seringkali disebut dengan *recall* dan *precision* kalimat atau frase, tapi kadang-kadang dengan *overlap* kata tunggal. *Precision* dan *recall* digunakan untuk mengukur kualitas sebuah ringkasan. Pengukuran *precision* dan *recall* ini sangat dipengaruhi oleh panjang ringkasan manual dan juga panjang ringkasan yang dievaluasi. Akurasi menurun sejalan dengan bertambah panjangnya ringkasan. Sulit untuk mengambil kesimpulan terhadap *performance* sistem dari nilai *precision* dan *recall*. Masalah utama dari evaluasi ini adalah sangat nyata, yaitu tidak satupun ringkasan yang benar. Kombinasi antara nilai *recall* dan *precision* menghasilkan *f-measure*. Untuk menghitung nilai performansi digunakan persamaan berikut:

$$recall = \frac{\sum kalimat\ yang\ sama}{\sum kalimat\ ringkasan\ manual} \tag{2.48}$$

$$precision = \frac{\sum kalimat\ yang\ sama}{\sum kalimat\ ringkasan\ sistem} \tag{2.49}$$

Kalimat yang sama adalah kalimat-kalimat ringkasan yang dihasilkan leh peringkas teks otomatis yang sama dengan kalimat – kalimat ringkasan yang dihasilkan secara manual oleh manusia. Untuk mencari kalimat relevan digunakan persamaan berikut:

$$kalimat\ yang\ sama = kalimat\ ringkasan\ sistem\ \cap kalimat\ ringkasan\ manual \tag{2.50}$$

Setelah diketahui nilai *recall* dan nilai *precision*, tahapan selanjutnya adalah menghitung nilai dari *f – measure* yang merupakan nilai yang digunakan untuk mengukur nilai akurasi dari ringkasan yang dihasilkan dengan menggunakan nilai yang dihasilkan pada perhitungan *recall* dan *precision* sebelumnya, sehingga *f-measure* bisa juga disebut sebagai gabungan atau

kombinasi nilai *precision* dan *recall*. *F-measure* dapat dicari dengan menggunakan persamaan berikut:

$$f - measure = 2 \times \frac{recall * precision}{precision + recall} \quad (2.51)$$

2.8. Pemrograman Berorientasi Objek

Pemrograman berorientasi objek berdasarkan pemodelan dari dunia nyata yang disebut objek. Objek terdiri dari data dan perilaku yang dikemas dalam objek. Objek dapat mempunyai karakteristik dari objek lain yang menurunkannya, sehingga metode bisa dibuat hanya sekali untuk pekerjaan tertentu [11]. Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama [11], yaitu:

a. *Encapsulation* (pengkapsulan) meruakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada di dalam objek itu sendiri.
b. *Inheritance* (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data atau atribut dan metode dari induknya langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek digunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metode lagi pada anaknya.
c. *Polymorphism* (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

2.9. *Unified Modelling Language*

Unified modelling language merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Ada beberapa diagram yang digunakan pada proses pembuatan perangkat lunak

berorientasi objek, diantaranya *use case diagram, activity diagram, class diagram* dan *sequence diagram* [12].

2.9.1. *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.

2.9.2. *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 pengelompokkan 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.

2.9.3. *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. *Class diagram* mempunyai 3 macam hubungan, yaitu:

1. *Assosiation*

   Association adalah hubungan antara bagian dari dua kelas. *Association* terjadi antara dua kelas akan terjadi jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam *class diagram*, sebuah *association* adalah penghubung antara dua kelas.

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.

2.9.4. *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.

2.10. Software Pendukung

Pada pembangunan sebuah aplikasi dibutuhkan beberapa *software* pendukung yang membantu untuk membangun aplikasi. Adapun *software* pendukung yang digunakan adalah:

2.10.1. Netbeans IDE

NetBeans IDE (*Integrated Development Environment*), dapat juga berupa *Integrated Design Environment* dan *Integrated Debugging Environment*, yakni sebuah program atau alat bantu yang terdiri atas *Editor, Compiler, Debugger* dan *Design* yang terintegrasi dalam satu aplikasi. Netbeans IDE ditujukan untuk memudahkan dalam penggunaan bahasa pemrograman Java. Untuk memakai Netbeans IDE maka, sebelumnya harus terlebih dahulu mempunyai *driver* JDK yang akan mendukung pembuatan perangkat lunak dengan menggunakan Netbeans IDE, sehingga sebelum meng-*install* Netbeans, terlebih dahulu sebelumnya harus meng-*install driver* JDK.

2.10.2. Java Development Kit

Java Development Kit atau disingkat JDK adalah *software development kit* merupakan alat bantu yang digunakan untuk membuat, memanajemen dan membangun berbagai aplikasi dalam bahasa pemograman Java. Pada JDK, terdapat berbagai *tools* yang digunakan untuk membangun aplikasi Java. Bahasa pemrograman Java menyediakan *library-library* standar yang telah di-*compile* dan dapat langsung digunakan dalam implementasi pembuatan sebuah aplikasi. Pada *library*, terdapat berbagai macam *class* yang dapat digunakan dan telah dikelompokkan ke dalam *package*.