BAB 2
LANDASAN TEORI

2.1 Ringkasan

Dalam Kamus Besar Bahasa Indonesia, ringkasan merupakan ikhtisar, singkatan cerita, meringkaskan dan kependekan. Tujuan utama dari sebuah ringkasan adalah membantu seseorang memahami dan mengetahui isi sebuah bacaan seperti dokumen. Dengan membuat ringkasan, pembaca dibantu untuk membaca dengan cermat dan cepat, sehingga waktu untuk pemahaman menjadi lebih sedikit[5].

2.1.1 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[8].

2.1.2 Bentuk ringkasan

Bentuk ringkasan dihasilkan melalui 2 pendekatan peringksan teks menurut Zaman B. dan E Winarko[9]. Berikut adalah penjelasan bentuk ringkasan melalui pendekatan secara abstraktif dan ekstraktif:

1. **Ringkasan dalam bentuk Abstraktif**

   Teknik abstraktif (*abstractive summary*) menggunakan metode linguistik 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.

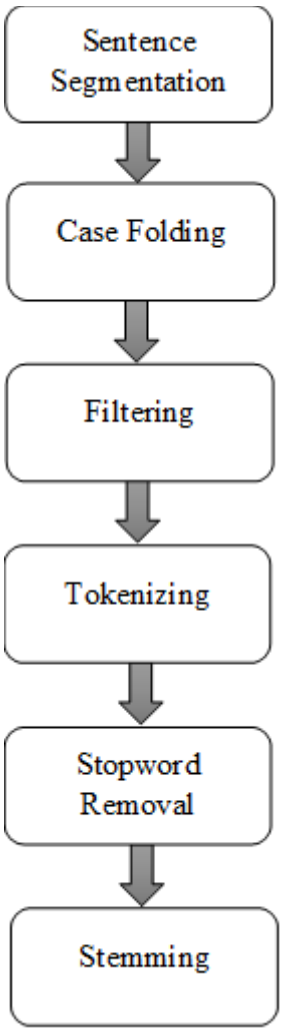
2. **Ringkasan dalam bentuk Ekstraksi**

   Ringkasan Ekstraktif (*extractive summary*) Pada teknik ekstraktif bekerja dengan cara menyalin unit-unit teks yang dianggap paling penting dari teks sumber menjadi ringkasan. Unit-unit teks yang disalin dapat berupa klausa utama,

kalimat utama, atau paragraf utama tanpa ada penambahkan kalimat-kalimat baru yang tidak terdapat pada dokumen aslinya.

2.2 *Preprocessing*

Preprocessing adalah merupakan tahapan awal dalam mengolah data input sebelum memasuki tahapan utama. Tujuan dari tahap ini adalah untuk penyeragaman dan untuk memudahkan dalam membaca. *Preprocessing* yang dilakukan adalah *sentence segmentation, Case Folding, filtering, tokenizing, stopword removal* dan *proses stemming*[10]. Untuk tahapan proses *preprocessing* datap dilihat pada gambar 2.1 berikut:



Gambar 2.1 Tahapan proses *preprocessing*

2.2.1 *Sentence Segmentation*

Sentence Segmentation adalah proses pemisahan teks tertulis menjadi unit makna kalimat[1]. Proses ini dilakukan untuk memisahkan kalimat di dalam sebuah teks, sehingga didapatkannya sekumpulan kalimat. Pada tahapan ini pemisahan kalimat dilakukan jika menemukan tanda titik (.).

2.2.2 *Case Folding*

Case Folding adalah proses merubah semua huruf kapital menjadi bukan kapital atau merubah semua huruf yang bukan kapital menjadi huruf kapital [1]. *Case Folding* dilakukan karena dokumen mengandung berbagai variasi dari bentuk huruf. Huruf harus diseragamkan untuk menghilangkan *noise* pada saat pengambilan informasi.

2.2.3 *Filtering*

Filtering adalah proses penghapusan simbol dan angka pada kalimat sehingga kalimat bersih dari noise[11]. *Filtering* dilakukan karena kalimat sering kali mengandung berbagai simbol dan angka yang dapat menjadi *noise* pada saat pengambilan informasi.

2.2.4 *Tokenizing*

Tokenizing adalah proses pemotongan string masukan berdasarkan tiap kata yang menyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen. Pada umumnnya setiap kata terpisahkan dengan kata yang lain oleh delimeter *spasi* dan *dash*, sehingga proses *tokenizing* mengandalkan delimeter tersebut untuk melakukan pemisahan kata[5].

2.2.5 *Stopword Removal*

Stopword removal merupakan proses menghilangkan kata-kata yang sering kali muncul dalam dokumen namun arti dari kata-kata tersebut tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. *Stopword* dapat berupa kata depan, kata penguhubung, dan kata pengganti contohnya adalah dan, yang, atau, ini, itu dan lain-lain[9].

2.2.6 *Stemming*

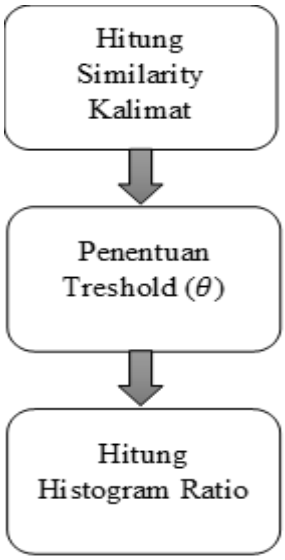
Stemming merupakan suatu proses untuk menemukan kata dasar dari sebuah kata. Dengan menghilangkan *prefixes, particle, possesive pronouns, suffixes* pada kata turunan[12]. *Stemming* digunakan untuk mengganti bentuk dari suatu kata menjadi kata dasar dari kata tersebut yang sesuai dengan struktur penulisan Bahasa Indonesia yang baik dan benar.

2.3 Clustering Kalimat

Clustering kalimat adalah proses pengelompokan berdasarkan similaritas pada setiap kalimat, semakin tinggi similaritas antar kalimat maka semakin baik juga cluster yang dihasilkan[5].

2.3.1 *Similarity-Based Histogram Clustering*

Similarity-based Histogram Clustering adalah metode representasi statistik yang berasal dari himpunan pasangan kalimat dalam suatu cluster dimana pengelompokan kalimat berdasarkan simlaritas pada kalimat[10]. Metode ini sangat baik untuk clustering kalimat karena mampu menjaga setiap cluster sebisa mungkin berada dalam kondisi koheren dan mempertahankan derajat koherensi dalam suatu cluster[1]. Untuk tahapan proses clustering kalimat dapat dilihat pada gambar 2.2 berikut:



Gambar 2.2 Tahapan proses *similarity-based histogram clustering*

2.3.2 *Similarity* Kalimat

Similarity kalimat adalah proses menentukan *similarity* dan *dissimilarity* setiap pasangan kalimat[5]. Metode yang digunakan untuk pengukuran *similarity* antar kalimat adalah *uni-gram matching-based similarity measure*, Metode ini dipilih mengingat kalimat-kalimat adalah unit yang sangat pendek dan *similarity* akan bernilai sangat kecil ketika dihitung dengan pengukuran *cosine similarity*[8]. Untuk perhitungan *similarity* antar kalimat menggunakan rumus 2.1 berikut:

$$\text{Sim}(S_i,S_j) = \frac{(2 \times |S_i| \cap |S_j|)}{|S_i| + |S_j|} \quad (2.1)$$

Keterangan:

$\text{Sim}(S_I,S_j)$ = Nilai *similarity* kalimat ke i dan ke j

$|S_i| \cap |S_j|$ = Irisan kalimat ke i dan ke j

$|S_i|$ = Jumlah kata unik pada kalimat i

$|S_j|$ = Jumlah kata unik pada kalimat j

Untuk menghitung jumlah *similarity* antar kalimat pada sebuah cluster menggunakan rumus 2.2 berikut:

$$m = n \times \frac{n+1}{2} \quad (2.2)$$

Keterangan:

m = Jumlah pasangan *similarity* antar kalimat

n = Jumlah kalimat

2.3.3 Penentuan *Treshold*

Sebuah nilai *threshold* digunakan untuk menentukan tingkat toleransi sebuah cluster menerima anggota cluster baru. Setiap kalimat yang menjadi kandidat anggota cluster harus mampu meningkatkan koherensi cluster tersebut atau memenuhi nilai *threshold*[5].

2.3.4 *Histogram Ratio*

Histogram ratio adalah proses perhitungan kualitas dari suatu cluster, semakin tinggi nilai *histogram ratio* semakin baik juga kualitas cluster yang dihasilkan[1]. Koherensi yang baik pada setiap cluster sangat penting untuk dijaga. Hal tersebut untuk mencegah adanya kalimat-kalimat yang redundan pada pemilihan kalimat penyusun ringkasan. Kualitas dari suatu *similarity histogram* yang merepresentasikan koherensi cluster ditentukan dengan menghitung *rasio similarity* yang berada diatas *threshold* dengan total jumlah *similarity* yang ada. Rasio dari histogram yang tinggi mencerminkan koherensi yang tinggi pula. Untuk perhitungan *Histogram ratio* menggunakan rumus rumus 2.3 berikut:

$$HR_c = \frac{\sum_{i=T}^{N_b} h_i}{\sum_{j=1}^{N_b} h_j} \tag{2.3}$$

Keterangan:

HR_c = *Histogram ratio* cluster c
N_b = Jumlah *bin* histogram
T = *Bin* yang dijadikan batasan *treshold*
h_i = Jumlah pasangan kalimat yang memenuhi *treshold*
h_j = Jumlah pasangan kalimat pada seluruh *bin*

Untuk menentukan *bin* yang menjadi *treshold* pada sebuah *histogram* menggunakan rumus 2.4 berikut:

$$T = S_t \times N_b \tag{2.4}$$

Keterangan:

T = *Bin* yang dijadikan batasan *threshold*
S_t = *Similarity Treshold*
N_b = Jumlah *bin* pada histogram

2.4 *Cluster Importance*

Cluster importance adalah sebuah metode yang melakukan pengurutan cluster berdasarkan nilai penjumlahan bobot dari kata-kata yang merupakan kata *frequent* (sering muncul) yang terkandung dalam cluster. Sebuah *threshold* (θ) ditetapkan untuk menentukan apakah suatu kata tersebut termasuk kata *frequent* atau tidak terhadap seluruh dokumen input. Jika frekuensi suatu kata memenuhi *threshold* θ maka kata tersebut dianggap sebagai kata yang memiliki bobot. Pendekatan *cluster importance* bertujuan mengukur pentingnya suatu cluster berdasarkan jumlah kata-kata *frequent* yang ada pada suatu cluster[11]. Untuk perhitungan bobot cluster menggunakan metode *cluster importance* menggunakan rumus 2.5 berikut:

$$\text{Weight}(C_i) = \sum_{t \epsilon C_i} \log(1+\text{count}(t)) \tag{2.5}$$

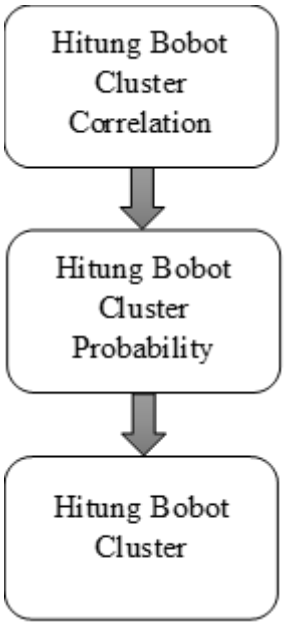
Keterangan:

Weight(C_i) = Bobot dari cluster i

count(t) = Jumlah dari kata t yang memenuhi *treshold*

2.5 *Cluster Correlation* dan *Probability*

Pengurutan cluster dilakukan karena pada proses clustering menggunakan metode *similarity-based Histogram Clustering* tidak pernah ada pengetahuan khusus berapa jumlah cluster yang akan terbentuk. Sehingga sangat penting untuk mengetahui urutan cluster yang nantiya akan menjadi urutan ringkasan akhir[8]. Untuk tahapan pembobotan *cluster correlation and probability* dapat dilihat pada gambar 2.3 berikut:



Gambar 2.3 Pengurutan cluster dengan *cluster correlation* dan *probability*

2.5.1 Bobot *Cluster Correlation*

Cluster correlation adalah proses menentukan hubungan pada setiap cluster, dimana apabila setiap cluster memiliki hubungan dengan cluster yang lain maka nilai dari korelasinya diantara -1 sampai dengan 1 dan apabila nilai korelasi pada cluster = 0 maka cluster tersebut tidak memiliki hubungan dengan cluster lain[5]. Untuk perhitungan bobot *term* frekuensi dari setiap korelasi cluster menggunakan rumus 2.6 berikut:

$$W_{C_jC_k} = n \times \sum_{i=1}^{n} W_{C_jt_i} \times W_{C_kt_i} \quad (2.6)$$

Keterangan:

$W_{C_jC_k}$ = Bobot *term* frekuensi dari setiap korelasi cluster

$W_{C_jt_i}$ = *Term* frekuensi t_i pada cluster C_j

$W_{C_kt_i}$ = *Term* frekuensi t_i pada cluster C_k

n = jumlah kata yang frekuensinya memenuhi *treshold*

Untuk menentukan jumlah frekuensi kata penting pada setiap cluster menggunakan rumus 2.7 berikut:

$$TF_{C_j} = \sum_{i=1}^{n} W_{C_j t_i} \quad (2.7)$$

Keterangan:

TF_{C_j} = Jumlah Frekuensi kata penting pada setiap cluster C_j

$W_{C_j t_i}$ = *Term* frekuensi t_i pada cluster C_j

n = Jumlah kata yang frekuensinya memenuhi *treshold*

Untuk menghitung *term* frekuensi pada setiap korelasi cluster menggunakan rumus 2.8 berikut:

$$TF_{C_j C_k} = TF_{C_j} \times TF_{C_k} \quad (2.8)$$

Keterangan:

$TF_{C_j C_k}$ = *Term* frekuensi pada setiap korelasi cluster

TF_{C_j} = *Term* frekuensi pada cluster C_j

TF_{C_k} = *Term* frekuensi pada cluster C_k

Untuk menghitung bobot *cluster correlation* pada setiap pasangan cluster menggunakan rumus 2.9 berikut:

$$correlation_{(C_j,C_k)} = \frac{W_{C_j C_k} - TF_{C_j c_k}}{\sqrt{\left[n \times \sum_{i=1}^{n} W_{C_j t_i}{}^2 - TF_{C_j}{}^2\right] \times \left[n \times \sum_{i=1}^{n} W_{C_k t_i}{}^2 - TF_{C_k}{}^2\right]}} \quad (2.9)$$

Keterangan:

$correlation_{(C_j,C_k)}$ = Nilai korelasi antara cluster C_j dan C_k

$W_{C_j C_k}$ = Bobot *term* frekuensi dari setiap korelasi cluster

$TF_{C_jC_k}$ = *Term* frekuensi pada setiap korelasi cluster

$W_{C_jt_i}$ = *Term* frekuensi t_i pada cluster C_j

$W_{C_kt_i}$ = *Term* frekuensi t_i pada cluster C_k

TF_{C_j} = *Term* frekuensi pada cluster C_j

TF_{C_k} = *Term* frekuensi pada cluster C_k

n = Jumlah kata yang frekuensinya memenuhi *treshold*

2.5.2 Bobot *Cluster Probability*

Probability adalah peluang muculnya suatu kejadian, dimana kejadian disini adalah peluang munculnya kata-kata yang dianggap penting pada sebuah kalimat di dalam cluster[5]. Proses perhitungan *cluster probability* menggunakan rumus 2.10 berikut:

$$\text{Probability}_{C_j} = \frac{\text{FrekuensiImportanceSentence}_{C_j}}{\text{FrekuensiImportanceSentenceAllDocument}} \quad (2.10)$$

Keterangan:

$\text{FrekuensiImprtanceSentence}_{C_j}$ = jumlah kalimat penting pada cluster C_j

FrekuensiImportanceSentenceAllDocument = jumlah kalimat penting pada seluruh cluster

2.5.3 Bobot Cluster

Weight cluster adalah proses pemberian bobot pada cluster dengan cara mengkalikan nilai korelasi suatu cluster dengan dengan nilai *cluster probability* tersebut. Cluster akan diurutkan dari nilai bobot cluster yang paling tinggi ke nilai bobot cluster terendah[5]. Untuk perhitungan bobot cluster menggunakan rumus 2.11 berikut:

$$\text{Weight}_{C_j} = \sum_{k=1}^{n} \text{correlation}_{(C_j,C_k)} \times \text{Probability}_{C_j} \quad (2.11)$$

Keterangan:

$Weight_{C_j}$ = Bobot Cluster ke C_j

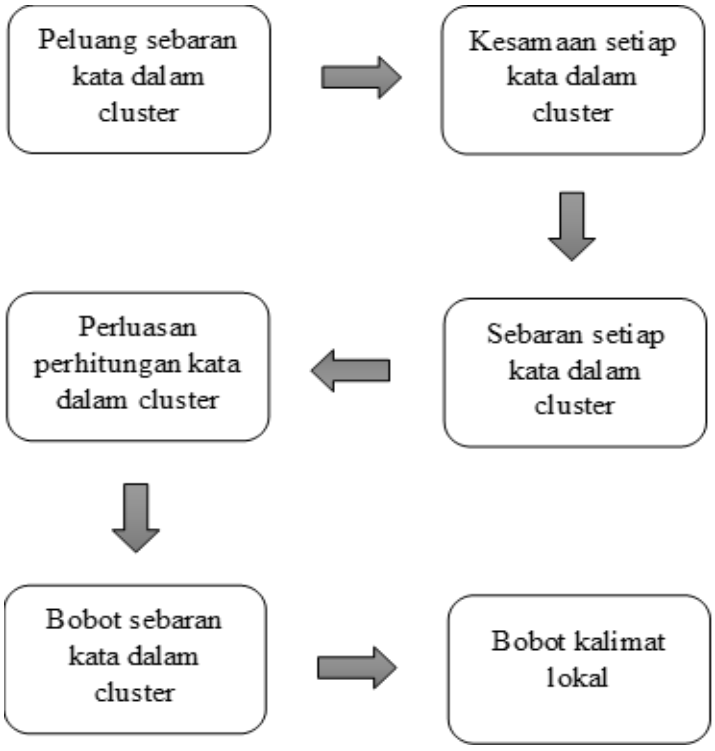
$Correlation_{(C_j,C_k)}$ = Bobot korelasi antara cluster C_j dan C_k

$Probability_{C_j}$ = Bobot *Probability* cluster C_j

n = Jumlah korelasi C_k

2.6 Ekstraksi Kalimat

Ekstraksi kalimat adalah proses penentuan kalimat yang merepresentasikan suatu cluster. Metode Sebaran kalimat lokal digunakan untuk menentukan kedudukan setiap kalimat dalam sebuah cluster. Dengan asumsi kalimat yang memiliki unsur-unsur kata paling tersebar dalam sebuah cluster akan memiliki kedudukan paling tinggi dalam cluster tersebut[13]. Untuk tahapan proses ekstraksi kalimat dapat dilihat pada gambar 2.4 berikut:



Gambar 2.4 Tahapan Proses *Distribution Local Sentence*

2.6.1 Peluang Sebaran kata dalam cluster

Proses awal dalam pembobotan sebaran kalimat lokal adalah menghitung peluang sebaran kata dalam sebuah cluster. Untuk menghitung peluang sebaran kata menggunakan rumus 2.12 berikut:

$$r_{ij} = \frac{|S_{ik}|_{dt}}{|C_k|_{dt}} \tag{2.12}$$

Keterangan:

r_{ij} = peluang sebaran seragam kata ke-j dalam kalimat ke-i

$|S_{ik}|_{dt}$ = jumlah kata dari kalimat ke-i pada cluster ke-k

$|C_k|_{dt}$ = jumlah kata pada cluster ke-k

2.6.2 Kesamaan setiap kata dalam cluster

Proses kedua dalam pembobotan sebaran kalimat lokal adalah menghitung kesamaan kata dalam sebuah cluster. Untuk menghitung kesamaan kata dalam cluster menggunakan rumus 2.13 berikut:

$$X^2_{jk} = \sum_{j=1}^{|C_k|_{dt}} \frac{(v_{ij} - n_{jk} \times r_{ij})^2}{(n_{jk} \times r_{ij})} \tag{2.13}$$

Keterangan:

X^2_{jk} = Sebaran seragam kata kej dalam cluster ke-k

n_{jk} = Frekuensi kata ke-j dalam cluster ke-k

v_{ij} = Frekuensi kata ke-j dalam kalimat ke-i

$|C_k|_{dt}$ = Jumlah kata dalam cluster ke-k

2.6.3 Sebaran setiap kata dalam cluster

Proses ketiga dalam pembobotan sebaran kalimat lokal adalah menghitung sebaran kata dalam sebuah cluster. Untuk menghitung sebaran kata dalam cluster menggunakan rumus 2.14 berikut:

$$U_{jk} = \frac{1}{1+ X^2_{jk}} \tag{2.14}$$

Keterangan:

U_{jk} = Nilai(tingkat) sebaran kata ke-j seragam dalam cluster ke-k

X^2_{jk} = Sebaran seragam kata ke-j dalam cluster ke-k

2.6.4 Perluasan perhitungan sebaran kata dalam cluster

Proses keempat dalam pembobotan sebaran kalimat lokal adalah menghitung perluasan sebaran kata dalam sebuah cluster. Untuk menghitung perluasan sebaran kata dalam cluster menggunakan rumus 2.15 berikut:

$$St_{jk} = \log_2 \left(1 + \frac{P_{jk}}{P_k}\right) \tag{2.15}$$

Keterangan:

St_{jk} = Nilai sebaran kata ke-j dalam cluster secara optimal

P_{jk} = Jumlah kalimat yang mengandung kata ke-j pada cluster ke-k

P_k = Jumlah total kalimat dalam cluster ke-k

2.6.5 Bobot sebaran kata dalam cluster

Proses kedua dalam pembobotan sebaran kalimat lokal adalah menghitung bobot sebaran kata dalam sebuah cluster. Bobot sebaran kata adalah proses menghitung nilai sebaran kata cluster[13]. Untuk menghitung bobot sebaran kata dalam cluster menggunakan rumus 2.16 berikut:

$$Wt_{jk} = \log_2\left(1 + U_{jk} \times St_{jk}\right) \tag{2.16}$$

Keterangan:

Wt_{jk} = Bobot sebaran kata lokal ke-j dalam cluster ke-k

U_{jk} = Nilai(tingkat) sebaran kata ke-j seragam dalam cluster ke-k

St_{jk} = Nilai sebaran kata ke-j dalam cluster secara optimal

2.6.6 Bobot sebaran kalimat lokal

Proses kedua dalam pembobotan sebaran kalimat lokal adalah menghitung bobot sebaran kalimat lokal dalam sebuah cluster semakin kata yang terkandung dalam kalimat tersebar maka nilai bobot pada kalimat akan semakin besar[13]. Untuk menghitung bobot sebaran kalimat lokal dalam cluster menggunakan rumus 2.17 berikut:

$$W_{(S_{ik})} = \frac{1}{S_{ik}} \times \sum_{Wt_{jk} \epsilon S_{ik}} Wt_{jk} \tag{2.17}$$

Keterangan:

$W_{(S_{ik})}$ = Bobot sebaran kalimat lokal kalimat ke-i pada cluster ke-k

S_{ik} = Jumlah kata penyusun kalimat ke-i pada cluster ke-k

Wt_{jk} = Bobot sebaran kata lokal ke-j dalam cluster ke-k

2.7 Evaluasi Ringkasan

RecallOriented Understudy for Gisting Evaluation (ROUGE) adalah metode evaluasi hasil ringkasan yang mengukur kualitas hasil ringkasan berdasarkan kesesuaian antara unit-unit ringkasan hasil sistem dengan unit-unit ringkasan referensi yang dibuat secara manual[11]. Untuk menghitung evaluasi ringkasan menggunakan rumus 2.18 berikut:

$$\text{ROUGE-N} = \frac{\sum \text{S} \epsilon \text{Sum}_{\text{ref}} \times \sum \text{N-gram} \in \text{S}^{\text{Count}_{\text{match(N-gram)}}}}{\sum \text{S} \epsilon \text{Sum}_{\text{ref}} \times \sum \text{N-gram} \in \text{S}^{\text{Count}_{\text{(N-gram)}}}} \quad (2.18)$$

Keterangan:

ROUGE-N = Hasil evaluasi ringkasan

$\sum \text{S} \epsilon \text{Sum}_{\text{ref}}$ = Jumlah kalimat referensi

$\sum \text{N-gram} \in \text{S}^{\text{Count}_{\text{match(N-gram)}}}$ = Jumlah *N-gram* yang sama pada ringkasan

$\sum \text{N-gram} \in \text{S}^{\text{Count}_{\text{(N-gram)}}}$ = Jumlah *N-gram* keseluruhan pada ringkasan

2.8 Pemrograman Berorientasi Objek

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Semua data dan fungsi ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya, Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan[14].

Salah satu keuntungan utama dari teknik pemrograman berorientasi obyek atas teknik pemrograman prosedural adalah bahwa memungkinkan programmer untuk membuat modul yang tidak perlu diubah ketika sebuah jenis baru objek ditambahkan. Seorang pemrogram hanya dapat membuat objek baru yang mewarisi banyak fitur dari objek yang sudah ada. Hal ini membuat program *object-oriented* lebih mudah untuk memodifikasi. Konsep dasar berorientasi objek diantaranya [14]:

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 atau 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 olehobjek dalam kelas objek. Attribut 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. Enkapulasi (*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. 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.
10. Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.
11. Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.
12. *Package* adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.9 *Unified Modeling Language* (UML)

Unified Modeling Language adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual dan juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. 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 adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh *Grady Booch, Jim Rumbaugh,* dan *Ivar Jacobson*. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem[14].

2.9.1 *Use Case Diagram*

Use case diagram adalah gambaran umum sistem dari sudut pandang pengguna sistem. Tujuan dari *use case* adalah untuk menggambarkan apa yang sistem dapat lakukan. *Use case* diagram dibentuk dari scenario tentang kegunaan sistem yang dinotasikan dengan sebuah *use case*. Selanjutnya, *use case diagram* tidak hanya sangat penting pada analisis, tetapi juga sangat penting untuk perancangan (*design*), untuk mencari (mencoba menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian (*testing*). Membuat *use case diagram* yang komprehensif merupakan hal yang sangat penting dilakukan pada tahap analisis. Dengan menggunakan *use case diagram*, akan mendapatkan banyak informasi yang sangat penting yang berkaitan dengan aturan-aturan bisnis. Dalam hal ini, setiap objek yang berinteraksi dengan sistem atau perangka lunak (misalnya, orang, suatu perangkat keras, sistem lain, dan sebagainya) merupakan aktor untuk sistem atau perangkat lunak yang dibangun, sementara *use case* merupakan deskripsi lengkap tentang bagaimana sistem atau perangkat lunak berprilaku untuk para aktor nya. Dengan demikian, *use case program* merupakan deskripsi lengkap tentang interaksi yang terjadi antara para aktor dengan perangkat lunak yang sedang dibangun. Ketika mengembangkan *use case diagram*, hal yang pertama dilakukan adalah mengenali aktor untuk aplikasi yang sedang dikembangkan. Dalam hal ini, ada beberapa karakteristik untuk para aktor, yaitu actor ada diluar sistem yang sedang dikembangkan dan aktor verinteraksi dengan sistem yang sedang dikembangkan [14].

2.9.2 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state* diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem)

secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. *Use case* diagram merupakan gambaran menyeluruh dan pada umumnya sangatlah tidak terperinci. Oleh karena itu, harus diperinci perilaku sistem untuk masingmasing *use case* yang ada. . Penggunaan *activity diagram* dapat memberikan gambaran secara meyeluruh terhadap perangkat lunak yang dibangun [14].

2.9.3 *Sequence Diagram*

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu. Berikut adalah tujuan utama dari *sequence diagram*[14]:

1. Memperlihatkan interaksi antar objek dalam perintah yang berurut.
2. Mendefinisikan urutan kejadian yang dapat menghasilkan output yang diinginkan.
3. Menggambarkan alur kejadian sebuah aktivitas.
4. Lebih detail dalam menggambarkan aliran data, termasuk data atau behaviour yang dikirimkan atau diterima.

2.9.4 *Class Diagram*

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. classs 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. Berikut adalah sifat-sifat yang dimiliki oleh atribut dan metode[14]:

1. *Private* sifatnya tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected* sifatnya hanya dapat dipanggil oleh *class* yang bersangkutan dan *class* yang mewarisinya.
3. *Public* sifatnya dapat dipanggil oleh semua *class*.

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

1. *Assiciation* merupakan hubungan antar *class* yang statis. *Class* yang mempunyai relasi asosiasi menggunakan *class* lain sebagai atribut pada dirinya.
2. *Aggregation* merupakan relasi yang membuat *class* yang saling terikat satu sama lain namun tidak terlalu berkegantungan.
3. *Composition* merupakn relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.
4. *Dependency* merupakan hubungan antar *class* dimana *class* yang memiliki relasi *dependency* menggunakan *class* lain sebagai atribut pada *method*.
5. *Realization* merupakan hubungan antar *class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya.

2.10 Perangkat Lunak Pendukung

Pada bagian ini akan dijelaskan tentang beberapa perangkat lunak dan bahasa pemrograman yang digunakan untuk mendukung dalam pengembangan aplikasi yang dibuat. Bahasa pemrograman yang digunakan diantaranya adalah *javascript*. Selain itu terdapat perangkat lunak yang digunakan yaitu mysql dan *sublime text*.

2.10.1 *Javascript*

Javascript adalah bahasa pemrograman yang berbentuk kumpulan skrip, yang pada fungsinya berjalan pada suatu dokumen HTML, sepanjang sejarah internet bahasa ini adalah bahasa skrip pertama untuk web. Bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan terhadap bahasa

HTML dengan mengijinkan pengeksekusian perintah perintah di sisi user, yang artinya di sisi *browser* bukan di sisi *server web*[15].

Javascript diperkenalkan pertama kali oleh Netscape pada tahun 1995. Pada awalnya bahasa ini dinamakan *LiveScript* yang berfungsi sebagai bahasa sederhana untuk browser *Netscape Navigator 2*. Pada masa itu bahasa ini banyak di kritik karena kurang aman, pengembangannya yang terkesan buru buru dan tidak ada pesan kesalahan yang di tampilkan setiap kali kita membuat kesalahan pada saat menyusun suatu program. Kemudian sejalan dengan sedang giatnya kerjasama antara *Netscape* dan *Sun* (pengembang bahasa pemrograman *Java*) pada masa itu, maka *Netscape* memberikan nama *JavaScript* kepada bahasa tersebut pada tanggal 4 desember 1995. Pada saat yang bersamaan *Microsoft* sendiri mencoba untuk mengadaptasikan teknologi ini yang mereka sebut sebagai *Jscript* di *browser Internet Explorer 3*[15].

2.10.2 *Mysql*

MySQL adalah sebuah basis data yang mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau sejumlah tabel. Tipe data *MySQL* adalah data yang terdapat dalam sebuah tabel berupa *field-field* yang berisi nilai dari data tersebut. Nilai data dalam *field* memiliki tipe sendiri-sendiri. *MySQL* merupakan *database server open source* yang cukup popular keberadaannya. Dengan berbagai keunggulan yang dimiliki, membuat *software database* ini banyak digunakan oleh praktisi untuk membangun suatu project. Adanya fasilitas *Application Programming Interfac*e (API) yang dimiliki oleh *MySQL*, memungkinkan bermacam - macam aplikasi komputer yang ditulis dengan berbagai bahasa pemrograman dapat mengakses basis data *MySQL*[16].

2.10.3 *Sublime Text*

Sublime text adalah teks editor berbasis *Python*, sebuah teks editor yang elegan, kaya fitur, *cross platform*, mudah dan simpel yang cukup terkenal di kalangan *developer* (pengembang), penulis dan *desainer*. Para *programmer* biasanya menggunakan sublime text untuk menulis *source code* bahasa pemrograman[17].