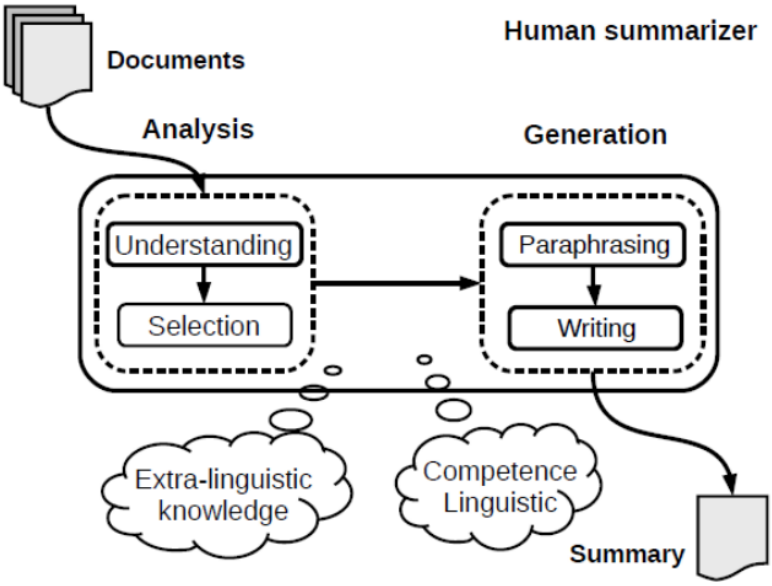
BAB 2
LANDASAN TEORI

2.1 Ringkasan

Ringkasan dalam Kamus Besar Bahasa Indonesia adalah memendekan (cerita, pembicaraan), mengikhtisarkan, mengambil inti sarinya saja. Menurut Van Dijk, ringkasan adalah sebuah teks yang fungsi utamanya adalah untuk mengindikasi dan memprediksi struktur dan isi dari suatu teks. Pengertian lainnya ringkasan adalah representasi akurat dari isi dokumen [4].

Tujuan utama dari sebuah ringkasan adalah membantu seseorang memahami dan mengetahui isi sebuah buku atau karangan. Dengan membuat ringkasan, pembaca dibantu untuk membaca dengan cermat dan cepat, sehingga waktu untuk pemahaman menjadi lebih sedikit.

Pembentukan ringkasan yang dilakukan oleh manusia dapat dilihat pada Gambar 2.1. Pembentukan ringkasan secara abstraksi yang dilakukan manual oleh manusia terbagi ke dalam dua fase, yaitu memahami teks sumber (*analysis*) dan menuliskan kembali hasil pemahaman teks sumber tersebut kedalam bentuk sebuah ringkasan (*generation*). Dalam fase analisis, seorang peringkas (*summarizer*) terlebih dahulu memahami teks yang akan diringkas (*understanding*), setelah itu melakukan pemilihan kalimat-kalimat mana saja yang akan digunakan dalam ringkasan yang akan dibuat (*selection*). Sedangkan untuk fase *generation*, peringkas akan menuliskan kembali hasil seleksi teks dalam bentuk sebuah ringkasan. Penulisan kembali teks sumber dapat dilakukan dengan cara menuliskan kalimat pilihan secara utuh ataupun dengan menuliskan kembali kalimat dengan menggunakan kata-kata sendiri (*paraphrasing*). Kedua fase tersebut membutuhkan kemampuan dan pengetahuan linguistik sehingga ringkasan yang dibuat akan memiliki hasil yang baik [4].



Gambar 2.1 Proses Pembentukan Ringkasan Hasil Buatan Manusia [4]

Ringkasan dapat dikategorikan dengan set yang berbeda, seperti fungsi, jenis dan sumber dokumen, dan lain-lainnya [4].

1. Ringkasan dilihat dari fungsinya antara lain:
   a. Ringkasan indikasi: ringkasan indikatif memberikan informasi tentang topik yang dibahas dalam dokumen sumber. Ringkasan ini menyerupai daftar isi.
   b. Ringkasan informatif: ringkasan informatif bertujuan untuk mencerminkan isi sumber teks, bisa menjelaskan *argument*. Ringkasan ini versi pendi dari dokumen.
2. Ringkasan dilihat dari jumlah dokumen antara lain:
   a. Dokumen Tunggal: ringkasan satu dokumen
   b. Multi dokumen: ringkasan dari beberapa dokumen yang belum tentu heterogen, dokumen yang diringkas biasanya tentang topik spesifik.
3. Ringkasan dilihat dari genre dokumen antara lain:
   a. Ringkasan berita: ringkasan artikel berita.
   b. Ringkasan khusus: ringkasan dokumen yang mempunyai domain khusus. Contohnya ilmu pengetahuan dan teknologi, dan lain lain.

c. Ringkasan ensiklopedia: ringkasan dokumen ensiklopedia, seperti Wikipedia dan lain lain.
d. Ringkasan jaringan sosial: ringkasan blog dan dokumen singkat.

4. Ringkasan dilihat dari tipe antara lain:
   a. Ekstrak: perakitan fragmen dari dokumen asli.
   b. Abstrak: meringkas dengan merumuskan, ringkasan diproduksi dengan menulis ulang dan/atau mengutip teks.
   c. Kompresi kalimat: ringkasan yang berisi jumlah yang sama dari kalimat sebagai teks asli, tapi dengan panjang yang berbeda.
5. Ringkasan dilihat dari jenis ringkasan antara lain:
   a. Ringkasan penulis: ringkasan yang ditulis oleh penulis dari dokumen menurut sudut pandangnya.
   b. Ringkasan ahli: ringkasan yang ditulis oleh selain penulis, seorang yang mengkhususkan diri dalam domain tapi mungkin tidak mengkhususkan diri dari dalam memproduksi ringkasan.
   c. Ringkasan professional: ringkasan yang ditulis oleh seorang peringkas ahli.
6. Ringkasan berdasarkan konteks antara lain:
   a. Ringkasan genetik: ringkasan dokumen yang mengabaikan kebutuhan informasi pengguna.
   b. Ringkasan permintaan-dipadu: ringkasan yang dipadukan dengan kebutuhan informasi pengguna.
   c. Pembaharuan ringkasan: ketika pengguna sudah familiar dengan topik tertentu dianggap bahwa mereka telah membaca dokumen dan ringkasan mereka yang berkaitan dengan tema tertentu. Oleh karena itu ringkasan ini hanya menampilkan informasi baru yang penting dan menghindari pengulangan informasi.
7. Ringkasan dilihat dari target pengguna antara lain:
   a. Tanpa profil: ringkasan yang independen dari kebutuhan profil pengguna.

Berdasarkan profil: ringkasan yang ditargetkan pada pengguna yang tertarik dalam domain khusus (kimia, politik internasional, olah raga, ekonomi).

2.2 Peringkasan Teks Otomatis

Menurut Juan-Manuel dan Torres-Moreno, ringkasan otomatis adalah sebuah ringkasan yang dibentuk oleh sebuah perangkat lunak, yang mengandung jumlah informasi relevan yang signifikan terhadap teks sumber [4]. Dengan peringkasan teks otomatis pembaca tidak perlu membuat sebuah ringkasan secara manual, tetapi dapat dilakukan dengan menggunakan bantuan komputer.

Secara umum pembentukan ringkasan oleh peringkasan teks otomatis didahului oleh proses *Preprocessing*, dimana proses tersebut memecah isi dari dokumen atau teks menjadi bagian-bagian tertentu, seperti kalimat atau kata. *Preprocessing* dalam peringkasan teks otomatis memiliki dua tujuan yaitu menormalisasi kata-kata dalam teks serta pengurangan kosa kata yang akan diproses untuk memudahkan proses peringkasan.

Perkembangan peringkasan teks terus melaju, salah satu risetnya adalah peringkasan teks otomatis menggunakan *machine learning*. Joel Larocca Neto dan kawan-kawannya adalah pengembang pertama dari peringkasan teks menggunakan pendekatan *machine learning* dengan menggunakan algoritma *naïve bayes* dan C45 [4].

2.3 *Preprocessing*

Preprocessing atau praproses adalah tahapan awal untuk menghasilkan sebuah ringkasan. Teks masukan yang akan diringkas sebelumnya harus melewati tahap untuk membuang berbagai macam jenis *noise* atau kata-kata yang dianggap tidak penting dalam ringkasan yang masih terdapat pada teks masukan [5].

Dalam penerapan *Preprocessing* terdapat beberapa tahapan yang harus dilalui dimulai dari tahapan tokenisasi, penghilangan *stopword* sampai dengan *stemming*. Selain tahapan-tahapan tersebut biasanya ditambahkan tahapan lain untuk kasus tertentu seperti *Case Folding* dan penghilangan kata yang jarang dimunculkan atau kata dengan frekuensi kemunculan yang kecil [6].

2.3.1 *Case Folding*

Case Folding adalah proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil atau huruf kapital *Case Folding* dilakukan karena dokumen memiliki berbagai variasi dari bentuk huruf. Variasi huruf harus diseragamkan untuk menghilangkan *noise* pada saat pengambilan informasi. Pada penelitian ini proses *Case Folding* dilakukan dengan menyeragamkan semua teks menjadi huruf kecil [7].

2.3.2 *Parsing*

Parsing merupakan proses memecah *string* teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Pemecahan kalimat dalam dokumen bisa dilakukan dengan mendeteksi keberadaan titik ".", tanda tanya "?", dan tanda seru "!" yang digunakan sebagai *delimiter* untuk memotong *string* pada dokumen [7].

2.3.3 *Filtering*

Filtering merupakan proses menghilangkan tanda baca atau simbol pada kalimat-kalimat yang telah dipecah dari hasil *Parsing* [7].

2.3.4 *Tokenization*

Tokenization merupakan proses pemecahan teks berupa kalimat menjadi kata-kata atau token-token. Pemecahan menjadi kata dalam *Tokenization* dilakukan dengan mendeteksi keberadaan *whitespace* spasi, tab, dan *newline* sebagai *delimiter* untuk memecah kalimat menjadi token-token [7].

2.3.5 *Stopword Removal*

Stopword Removal merupakan proses menghilangkan kata-kata yang sering muncul dalam dokumen tetapi arti dari kata tersebut tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. *Stopword* dapat berupa kata depan, kata penghubung, dan kata pengganti contohnya adalah "dan", "yang", "atau", "ini", "itu", dan lain lain. Daftar kata yang merupakan *stopword* yaitu berdasarkan *library python sastrawi* [7].

2.4 *Term Frequency Inverse Document Frequency* (*TF-IDF*)

Metode *Term Frequency-Inverse Document Frequency* adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *term frequency* (tf) merupakan frekuensi kemunculan term t pada dokumen i. Dan *document frequency* (df) merupakan frekuensi banyaknya dokumen i dimana term t muncul [7].

Dalam tf frekuensi *term* pilihan paling sederhana adalah dengan menggunakan frekuensi baku dalam dokumen, yaitu berapa kali *term* (t) terjadi dalam dokumen (d). nilai idf sebuah *term* dapat dihitung menggunakan persamaan berikut:

$$IDF = \log(\frac{n}{df}) \quad (2.1)$$

n = Jumlah dokumen/kalimat yang terdapat *term* t

df = kemunculan frekuensi *term* terhadap n

Adapun algoritma yang digunakan dalam menghitung bobot (w) masing-masing dokumen terhadap kata kunci atau *query* menggunakan persamaan berikut:

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

$W_{d.t}$ = Bobot dokumen ke-d terhadap *term* ke-t

d = Dokumen ke-d

t = *Term* ke-t

$tf_{d.t}$ = Frekuensi *term* dokumen ke-d terhadap *term* ke-t

IDF_t = nilai IDF *term* ke-t

2.5 *Latent Semantic Analysis (LSA)*

Latent Semantic Analysis (LSA) adalah suatu metode untuk mengekstrak sebuah tulisan dalam suatu dokumen dan kemudian mengaplikasikannya dalam perhitungan matematis. Penilaian dengan metode *LSA* lebih kepada kata-kata yang ada dalam tulisan tanpa memperhatikan urutan kata dan tata bahasa dalam tulisan tersebut, sehingga suatu kalimat yang dinilai adalah berdasarkan kata-kata kunci yang ada pada kalimat tersebut [5].

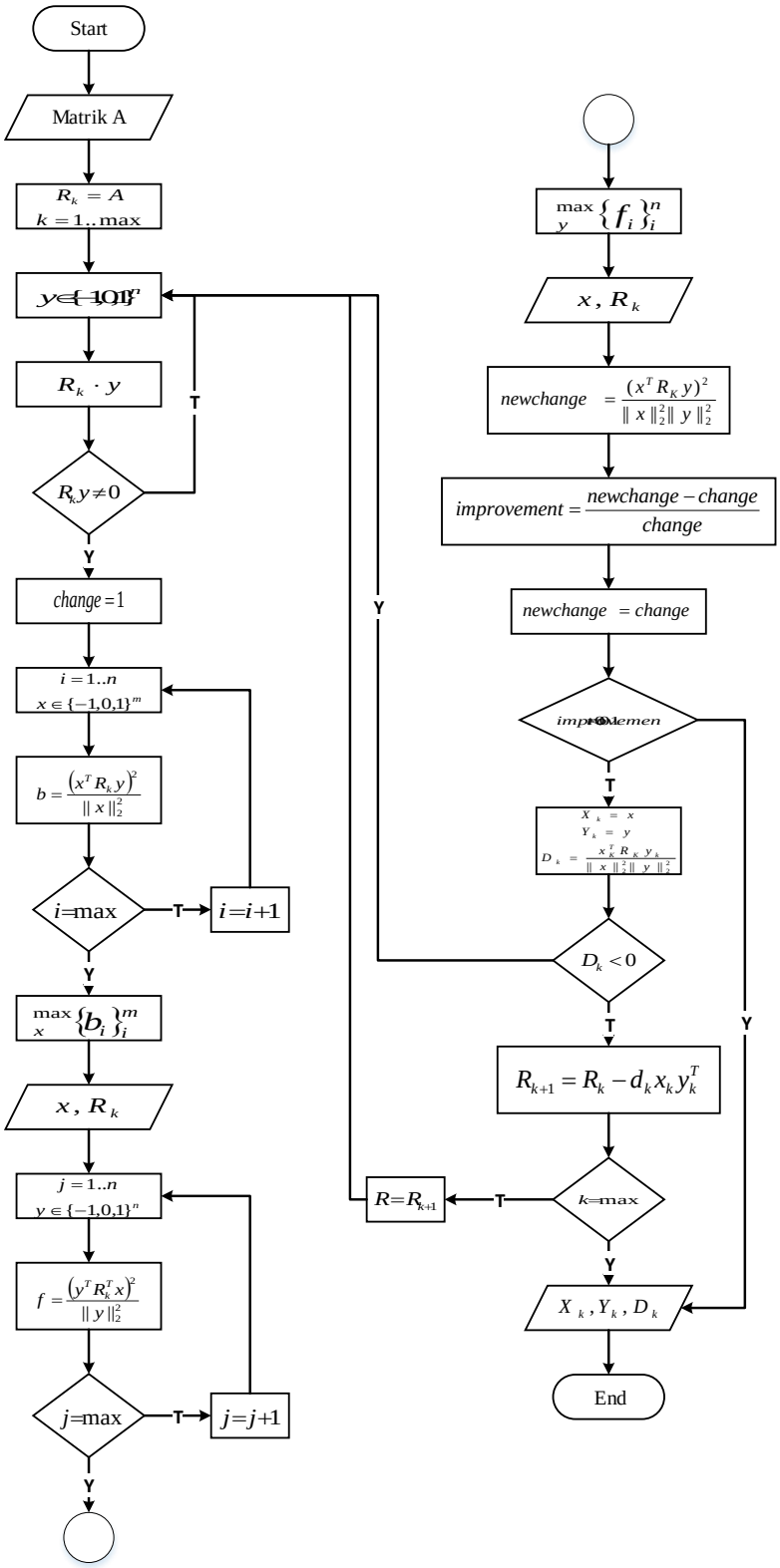
2.6 *Semi Discrete Decomposition (SDD)*

Semi Discrete Decomposition (SDD) adalah suatu proses dekomposisi yang memfaktorkan sebuah matriks menjadi lebih dari satu matriks, misalkan terdapat sebuah matriks hasil pembobotan berukuran $m \times n$ yang akan didekomposisi. Matriks tersebut direpresentasikan menjadi

$$A : A_k = X_K D_K Y_K^T \tag{2.3}$$

A = matriks hasil pembobotan berukuran $m \times n$
X = matrik orthogonal berukuran $m \times m$
D = matrik diagonal berukuran $m \times n$
Y = matrik orthogonal berukuran $n \times n$
k = bilangan bulat positif

Langkah-langkah pada proses *Semi Discrete Decomposition* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Tahapan-tahapan Semi Discrete Decomposition

Berikut adalah tahapan-tahapan dalam *Semi Discrete Decomposition*:

1. Inisialisasikan matriks awal berukuran $m \times n$
2. Inisialisasi nilai matrik awal R, nilai maksimum iterasi K, nilai $change$, nilai $improvement$.
3. Bangkitkan sebuah vektor y awal sehingga menghasilkan nilai $R_K y$ tidak sama dengan nol, jika sama hasil perkalian nol, maka bangkitkan vektor y yang lainya sampai menghasilkan nilai bukan nol.

$$R_k y \neq 0 \tag{2.4}$$

R_k = Matrik berukuran $m \times n$ hasil inisiaslisasi
y = Vektor y berukuran n dengan entri terdiri dari {-1, 0, 1}
k = Bilangan bulat positif

4. Bangkitkan vektor x secara random berukuran sebanyak m matrik R_k dengan entrinya hanya terdiri dari {-1, 0, 1}.
5. Cari nilai dari vektor x yang menghasilkan nilai terbesar untuk persamaan b (2.4).

$$b = \frac{(x^T R_k y)^2}{||x||_2^2} \tag{2.5}$$

b = Nilai maksimum
x^T = Vektor x berukuran m dengan entri terdiri dari {-1, 0, 1}
R_k = Matrik berukuran $m \times n$ hasil inisiaslisasi
k = Bilangan bulat positif
y = Vektor y berukuran n dengan entri terdiri dari {-1, 0, 1}
$||x||_2^2$ = Nilai norm dari x

6. Bangkitkan vektor y secara random berukuran sebanyak n matrik R_k dengan entrinya hanya terdiri dari {-1, 0, 1}

7. Cari nilai dari vektor y yang menghasilkan nilai terbesar untuk persamaan f(2.5)

$$f = \frac{(y^T R_k^T x)^2}{||y||_2^2} \tag{2.6}$$

f = Nilai maksimum

y^T = Vektor y berukuran n dengan entri terdiri dari {-1, 0, 1}

R_k^T = Matrik berukuran $m \times n$ hasil inisiaslisasi

k = bilangan bulat positif

x = Vektor x yang menghasilkan nilai terbesar persamaan b (2.5)

$||y||_2^2$ = Nilai norm dari vektor y

8. Cari nilai $newchange$ menggunakan persamaan berikut:

$$newchange = \frac{(x^T R_K y)^2}{||x||_2^2 ||y||_2^2} \tag{2.7}$$

$newchage$ = nilai *change* untuk iterasi berikutnya

x^T = Vektor x yang menghasilkan nilai terbesar persamaan b (2.5)

R_k = Matrik berukuran $m \times n$ hasil inisiaslisasi

k = bilangan bulat positif

y = Vektor y yang menghasilkan nilai terbesar persamaan f (2.6)

$||x||_2^2$ = Nilai norm dari vektor x

$||y||_2^2$ = Nilai norm dari vektor y

9. Hitung nilai $improvement$ menggunakan persamaan berikut:

$$Improvement = \frac{|newchange - change|}{change} \tag{2.8}$$

$Improvement$ = nilai error

$newchange$ = nilai *newchange* didapatkan dari persamaan (2.7)

$change$ = nilai *change* hasil inisialisasi

10. Cari nilai d dengan menggunakan persamaan:

$$\begin{aligned} X_k &= x \\ Y_k &= y \\ d_k &= \frac{(x_k^T R_k y_k)^2}{||x_k||_2^2 ||y_k||_2^2} \end{aligned} \tag{2.9}$$

X_k = Nilai matrik X baris ke-k

x = Vektor x yang menghasilkan nilai terbesar persamaan b (2.4)

Y_k = Nilai matrik Y baris ke-k

y = Vektor x yang menghasilkan nilai terbesar persamaan b (2.4)

k = bilangan bulat positif

$||x_k||_2^2$ = nilai norm x

$||y_k||_2^2$ = nilai norm y

$d_k = \frac{(x_k^T R_k y_k)^2}{||x_k||_2^2 ||y_k||_2^2}$ = Mencari nilai diagonal matrik d

11. Tentukan matriks untuk iterasi berikutnya menggunakan persamaan:

$$R_{k+1} = R_k - d_k x_k y_k^T \quad (2.10)$$

R_{k+1} = Matrik iterasi berikutnya berukuran $m \times n$
R_k = Matrik berukuran $m \times n$ hasil inisiaslisasi
d_k = Dilai diagonal ke-k matrik D
x_k = Vektor x yang menghasilkan nilai terbesar persamaan b (2.4)
y_k^T = Vektor y yang menghasilkan nilai terbesar persamaan f (2.5)
k = bilangan bulat positif

2.7 *Cross Method Latent Semantic Analysis* (*CMLSA*)

Cross Method Latent Semantic Analysis (CMLSA) merupakan sebuah pengembangan dari metode *LSA* yang telah ada sebelumnya. Metode ini dapat menghasilkan ringkasan dari teks masukan yang lebih akurat dibandingkan dengan metode *LSA* yang sebelumnya [8]. Yang menjadi perbedaan metode ini dengan metode *LSA* terdapat pada saat tahap ekstraksi ringkasan. Metode ini menggunakan nilai rata-rata (*average*) dan panjang (*length*) yang diambil dari matriks Y^T dan matriks D. Nilai *average* diambil dari nilai rata-rata dari setiap bobot kata yang terdapat pada baris matriks Y^T. Setelah ditemukan nilai rata-rata dari setiap kata yang terdapat pada matriks Y^T, lalu akan dilakukan pencocokan nilai rata-rata yang didapatkan pada setiap baris dengan nilai dari setiap kata yang terdapat pada baris tersebut. Jika nilai dari kata tersebut lebih rendah dari nilai rata-rata yang didapat, maka nilai kata tersebut diubah menjadi nol (0). Jika tidak, maka nilai dari kata tersebut tetap.

Tahapan selanjutnya adalah menghitung $length$ dari setiap baris dari matriks Y^Tdengan menggunakan persamaan (2.11). Baris-baris pada kalimat yang mempunyai nilai *length* yang tinggi akan dijadikan sebagai ringkasan.

$$length\sqrt{\sum_{j=1}^{n} Y_{ij}^2 D_{jj}^2} \quad (2.11)$$

$length$ = panjang nilai setiap baris

Y_{ij}^2 = nilai matrik Y posisi ke-ij

D_{jj}^2 = nilai diagonal matrik D posisi ke-jj

i = bilangan bulat positif

j = bilangan bulat positif

2.8 Evaluasi Ringaksan

Evaluasi ringkasan adalah cara untuk mengetahui hasil ringkasan itu baik oleh peringkasan secara manual ataupun memiliki tingkat akurasi yang baik atau tidak untuk peringkasan teks otomatis. Mengevaluasi sebuah ringkasan merupakan kegitaan yang subjektif karena itu evaluasi ringkasan merupakan hal yang sulit.

Sebuah ringkasan menurut sebagian orang sudah tepat, tapi menurut sebagian orang lagi belum tentu. Secara umum terdapat dua jenis metode evaluasi ringkasan yaitu ekstrinsik dan intrinsik. Ekstrinsik diukur kualitas ringkasan berdasarkan bagaimana ringkasan tersebut dapat membantu penyelesaian tugas orang yang membaca ringkasan tersebut. Sedangkan intrinsik diukur dari kualitas hasil keluaran ringkasan yang dihasilkan [9].

Evaluasi sistem peringkasan yang akan digunakan dalam penelitian ini adalah evaluasi intrinsik. Pengevaluasi menciptakan sebuah ringkasan manual untuk menguji teks. Kemudian membandingkan hasil ringkasan sistem dengan ringkasan ideal. Evaluasi yang diukur adalah *overlap* dari isi, seringkali disebut dengan *recall* dan *precision* kalimat. Pengukuran *precision* dan *recall* ini sangat dipengaruhi oleh panjang ringkasan manual dan juga panjang ringkasan yang dievaluasi. Akurasi menurun sejalan dengan bertambahnya panjang ringkasan. Sulit untuk mengambil kesimpulan terhadap *performance* sistem dari nilai *precision* dan *recall*. Untuk standarisasi proses evaluasi ringkasan sama sekali belum dieksprolasi. Kombinasi antara nilai *recall* dan *precision* menghasilkan *f-measure* [10].

2.8.1 *Recall*

Recall (r) merupakan istilah yang digunakan untuk kalimat terpanggil yang *relevan* dengan pernyataan atau *vector query*. Perbandingan jumlah informasi *relevan* yang didapatkan sistem dengan jumlah seluruh informasi *relevan* yang ada dalam koleksi informasi. *Recall* berhubungan dengan kemampuan dalam menemukan kalimat yang relevan. Hal ini berarti *recall* adalah bagian proses evaluasi yang dapat digunakan sebagai alat ukur relevannya sebuah ringkasan. Adapun rumus untuk menghitung *recall* dapat dilihat pada persamaan berikut:

$$Recall = \frac{tp}{tp + fn} \tag{2.12}$$

$Recall$: tingkat keberhasilan

tp : jumlah kalimat yang berhasil diekstrak sistem sesuai dengan kalimat yang diekstrak manusia.

fn : jumlah kalimat yang diekstrak manusia tapi tidak terdapat dalam kalimat yang diekstrak sistem.

2.8.2 *Precision*

Precision (p) biasanya menjadi salah satu ukuran yang digunakan untuk menilai efektifitas kalimat yang dihasilkan dalam sebuah rangkuman. *Precision* adalah tingkat ketepatan hasil ringkasan. Perbandingan jumlah informasi *relevan* yang didapatkan sistem dengan jumlah seluruh informasi yang terambil oleh sistem baik yang *relevan* maupun tidak. Adapun, pengukuran tingkat ketepatan *precision* dirumuskan pada persamaan berikut:

$$precision = \frac{tp}{tp + fp} \quad (2.13)$$

$precision$: Tingkat ketepatan.

$tp.$: Jumlah kalimat yang berhasil diekstrak sistem sesuai dengan kalimat yang diekstrak manusia.

fp : jumlah kalimat yang diekstrak sistem tetapi tidak terdapat dalam kalimat yang diekstrak manusia.

2.8.3 *F-Measure*

f-measure (f_1) merupakan salah satu perhitungan evaluasi yang mengkombinasi *recall* dan *precision*. Hubungan antara *recall* dan *precision* yang mempresentasikan akurasi sistem. Nilai *recall* dan *precision* pada suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *recall* dan *precision* adalah *f-measure* yang merupakan bobot *harmonic mean* dari *recall* dan *precision*. Perhitungan *f-measure* dapat dirumuskan pada persamaan berikut:

$$f - measure = \frac{2 * precision * recall}{precision + recall} \quad (2.14)$$

Dimana parameter *precision* merupakan hasil dari perhitungan *precision* dan parameter *recall* merupakan hasil dari perhitungan *recall*.

2.8.4 *Akurasi*

Akurasi merupakan salah satu perhitungan evaluasi yang mengkombinasikan antara kalimat yang terpilih oleh manusia, kalimat yang tidak terpilih oleh manusia, dan kalimat hasil pengambil sistem. Perhitungan akurasi adalah sebagai berikut:

$$akurasi = \frac{tp + tn}{tp + fp + fn + tn} \quad (2.15)$$

tp : jumlah kalimat yang berhasil diekstrak sistem sesuai dengan kalimat yang diekstrak manusia

fp : jumlah kalimat yang diekstrak sistem tetapi tidak terdapat dalam kalimat yang diekstrak manusia

fn : jumlah kalimat yang diekstrak manusia tetapi tidak terdapat dalam kalimat yang diekstrak sistem.

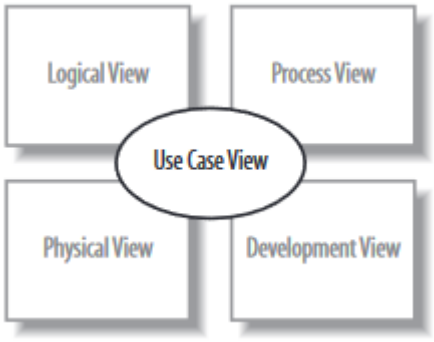
tn : jumlah kalimat yang tidak diekstrak manusia dan tidak diekstrak sistem.

2.9 *Unified* Modeling Language (*UML*)

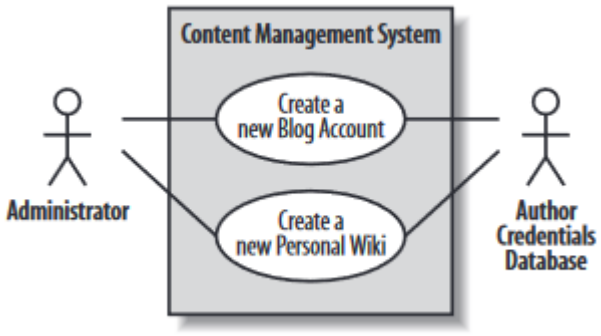
Unified Modeling Language (*UML*) adalah sebuah teknik atau bahasa pembuatan model untuk pengembangan atau pembangunan perangkat lunak dan sistem. Secara umum *modeling language* dapat juga dibuat dengan menggunakan *pseudo-code, actual code*, gambar, maupun *diagram* [11].

2.9.1 *Use Case Diagram*

Use Case adalah sebuah situasi dimana sistem digunakan untuk memenuhi satu atau seluruh kebutuhan user. *Use Case* sendiri adalah bagian terpenting atau bagian utama dari model pembangunan. Berdasarkan *UML*, *Use Case* sendiri dapat diumpamakan bagian-bagian dari *logical, process, physical*, dan *development* seperti pada Gambar 2.3 dan contoh sederhana dari *Use Case* sudah tergambarkan pada Gambar 2.4.



Gambar 2.3 Posisi *Use Case* Berdasarkan *UML*



Gambar 2.4 Contoh *Use Case*

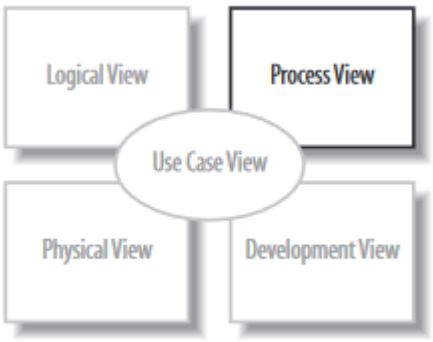
Setelah pembangunan *Use Case* akan dibangun pula sebuah skenario dari masing-masing *Use Case* yang ada. Kegunaan dari skenario *Use Case* sendiri adalah untuk memperjelas proses yang ada di dalam masing-masing *Use Case* tersebut secara tekstual. Contoh dari skenario *Use Case* dari "*Create a new Personal Wiki*" dari *Use Case* sebelumnya dan telah tergambarkan pada Gambar 2.5.

Use case name	**Create a new Personal Wiki**	
Related Requirements	Requirement A.2.	
Goal In Context	A new or existing author requests a new personal Wiki from the Administrator.	
Preconditions	The author has appropriate proof of identity.	
Successful End Condition	A new personal Wiki is created for the author.	
Failed End Condition	The application for a new personal Wiki is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the CMS to create a new personal Wiki.	
Main Flow	**Step**	**Action**
	1	The Administrator asks the system to create a new personal Wiki.
	2	The Administrator enters the author's details.
	3	The author's details are verified using the Author Credentials Database.
	4	The new personal Wiki is created.
	5	A summary of the new personal Wiki's details are emailed to the author.
Extensions	**Step**	**Branching Action**
	3.1	The Author Credentials Database does not verify the author's details.
	3.2	The author's new personal Wiki application is rejected.

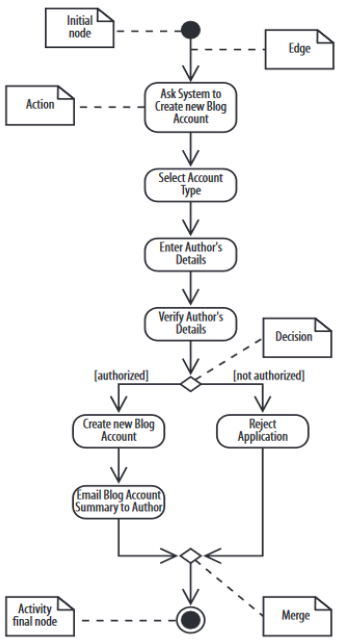
Gambar 2.5 Contoh Skenario *Use Case*

2.9.2 *Activity Diagram*

Activity diagram digunakan untuk menjelaskan alur atau cara dari sebuah sistem mencapai tujuannya dengan *diagram*. *Activity diagram* sendiri menjelaskan alurnya dengan cara *actions chained* atau aksi-aksi yang saling berhubungan pada sistem tersebut. Jika kita melihat Gambar 2.5 kembali, maka berdasarkan *UML*, *Activity diagram* adalah bagian dari *process view* seperti pada Gambar 2.6. Contoh dari *Activity diagram* sendiri telah tergambarkan pada Gambar 2.7.



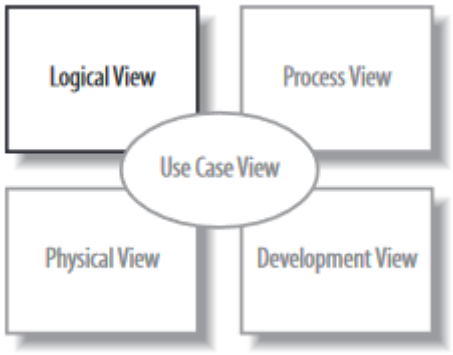
Gambar 2.6 Posisi *Activity Diagram* berdasarkan *UML*



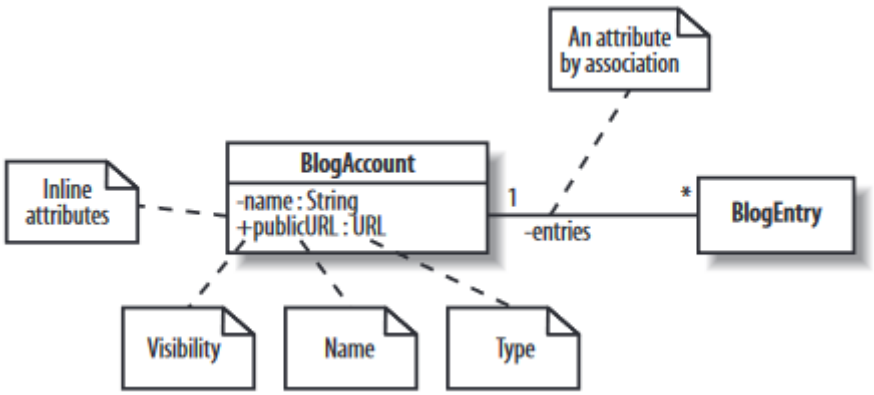
Gambar 2.7 Contoh *Activity Diagram*

2.9.3 *Class Diagram*

Class diagram digunakan untuk menjelaskan bagian objek-objek yang berbeda yang dibutuhkan pada sistem. *Class diagram* adalah bagian terpenting dari *object-oriented system* maka dari itu *Class diagram* adalah yang paling populer dari *UML*. Bagian dari *Class diagram* pada *UML* sendiri telah tergambarkan pada Gambar 2.8, dan contoh dari *Class diagram* telah tergambarkan pada Gambar 2.9.



Gambar 2.8 Posisi *Class Diagram* Pada *UML*

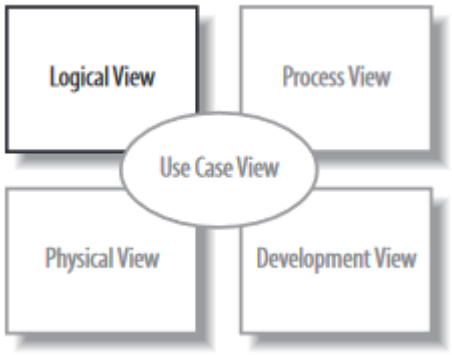


Gambar 2.9 Contoh *Class Diagram*

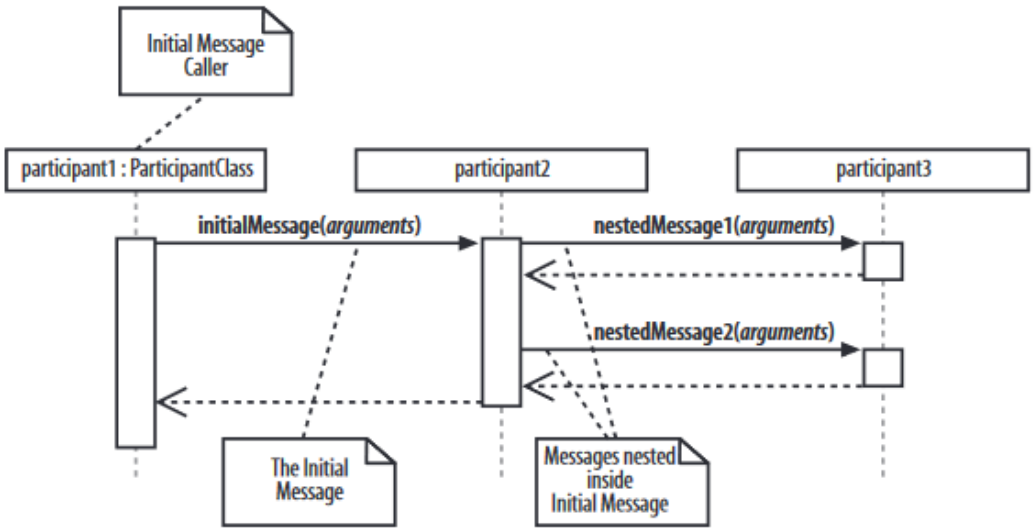
2.9.4 *Sequence Diagram*

Sequence diagram digunakan untuk menjelaskan urutan hubungan atau interaksi terhadap bagian-bagian dari sistem. Dengan *Sequence diagram* setiap interaksi dapat dijelaskan akan berjalan ketika sesuatu menjalankannya atau melakukan *trigger* terhadap interaksi tersebut. Bagian dari *Sequence diagram* pada

UML sendiri serupa dengan *Class diagram* dan telah tergambarkan pada Gambar 2.10, contoh untuk *Sequence diagram* sendiri telah tergambarkan pada Gambar 2.11.



Gambar 2.10 Posisi *Sequence Diagram* pada *UML*



Gambar 2.11 Contoh *Sequence Diagram*

2.10 *Python*

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. *Python* adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang

berfokus pada tingkat keterbacaan kode. *Python* merupakan salah satu bahasa pemrograman yang populer di dunia kerja Indonesia.

Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan dengan sitaks kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Pada umumnya *Python* mendukung multi paradigma pemrograman, seperti pemrograman berorientasi objek, pemrograman imperatif dan pemrograman fungsional [12].