BAB 2
LANDASAN TEORI

2.1 Peringkasan Teks

2.1.1 Defenisi Peringkasan Teks Otomatis

Peringkasan teks otomatis (*Automatic Text Summarization*) adalah suatu proses pembentukan yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan dan dioperasikan pada komputer. Sedangkan menerut Hovy, ringkasan adalah teks yang dihasilkan dari sebuah teks atau banyak teks, yang mengandung isi informasi dari teks asli dan panjangnya tidak lebih dari setengah teks aslinya [5].

2.1.2 Tahapan Membuat Ringkasan

Tahapan dalam membuat ringkasan bacaan [6]:

1. Membaca bacaan atau karangan asli mengetahui kesan umum, maksud pengarang.
2. Mencatat gagasan utama atau pokok pikiran dalam tiap paragraph.
3. Menyusun pokok pikiran atau gagasan pokok bacaan menjadi suatu paragraph atau lebih.

Untuk membuat dan mendapatkan ringkasan yang baik, yaitu dengan cara membaca dengan seksama teks bacaan dan mencatat hal-hal pokok dari bacaan tersebut. Ciri-ciri sebuah ringkasan yang baik itu harus mudah dipahami, bentuknya lebih singkat atau lebih ringkas, dan terdiri dari gagasan-gagasan utama (*Expert Judgement*).

2.1.3 Tujuan Ringkasan

Tujuan dari peringkasan teks dapat dikategorikan berdasarkan maksud, fokus, dan cakupannya. Sebagai berikut :

a. *Informatif* tujuannya ringkasan ini adalah untuk menyatakan informasi–informasi penting yang terdapat pada dokumen asal.

b. *Indicatif* tujuannya ringkasan ini adalah untuk dijadikan referensi yang membantu pembaca mengetahui isi dari teks daripada membaca keseluruhan teks yang ada. Ringkasan ini meliputi topik dari teks asal.
c. *Evaluatif* yaitu melibatkan pembuatan sebuah pertimbangan pada teks asal, seperti suatu tinjauan ulang atau opini.
d. *User-Focused* yaitu ringkasan yang dibuat berdasarkan topik yang dipilih oleh user, sering yang merupakan jawaban dari query yang dimiliki oleh user.
e. *Generic* yaitu sifatnya lebih umum dan berdasarkan pada teks aslinya.
f. *Single Document* (dokumen tunggal) yaitu ringkasan yang dihasilkan merupakan ringkasan dari satu dokumen.
g. *Multi Document* pada peringkasan *multi document*, ringkasan merupakan hasil ringkasan dari banyak dokumen.

2.2 Proses Peringkasan Teks

Secara umum terdapat tiga tahapan dalam proses peringkasan teks, yaitu sebagai berikut :

1. *Topic Identification*

   Tahapan ini meliputi identifikasi faktor yang sangat penting tentang apa yang dibicarakan dala teks tersebut. Ada beberapa teknik untuk melakukannya, diantaranya:

   a. Dalam beberapa tipe teks, informasi penting terdapat dalam bagianbagian tertentu dalam teks tersebut, seperti dalam judul, kalimat pertama, kalimat terakhir dan lain sebagainya.
   b. Beberapa kata atau frase mengidentifikasi intisari dari suatu teks.
   c. Beberapa kata cenderung muncul lebih. Ini dijadikan faktor penentu topik dari suatu teks (*word frequency*).
   d. Ada juga topik diidentifikasi dari jumlah pengertiannya atau makna dibandingkan kata.

2. *Interpretetion*

   Interpretasi peringkas yang ekstraktif berdasarkan pada metode yang digunakan, sedangkan pada peringkas yang abstraktif interpretasi ditunjukkan

dengan cara penggabungan pengertian yang serupa menjadi satu, penghilangan redudansi dan lain sebagainya.

3. *Generating*

   Tahapan ini adalah pembangkitan atau pembentukan hasil akhir. Terdiri dari penggabungan frase, pencetakan kata atau frase dan pembangkitan kalimat. Terdapat beberapa metode yang dapat digunakan, diantaranya :

   a. *Extraction,* hasil akhir berisi kalimat atau frase yang dihasilkan setelah semua tahap proses pada peringkasan teks selesai dilakukan.
   b. *Topic List,* hasil akhir berisi kata-kata yang sering muncul atau penggabungan pengertian yang telah diinterpretasi.

2.3 Kata

Kamus Besar Bahasa Indonesia (KBBI 1997) memberikan beberapa definisi mengenai kata, sebagai berikut :

1. Elemen terkecil dalam sebuah bahasa yang diucapkan atau dituliskan dan merupakan realisasi kesatuan perasaan dan pikiran yang dapat digunakan dalam berbahasa.
2. Konversasi, bahasa.
3. Morfem atau kombinasi beberapa morfem yang dapat diujarkan sebagai bentuk yang bebas.
4. Unit bahasa yang dapat berdiri sendiri dan terdiri dari satu morfem (contoh kata) atau beberapa morfem gabungan (contoh perkataan).

Kategori kata berdasarkan sintaksisnya terdiri dari lima kelas kata yaitu :

1. Kata Benda (nomina)

   Adalah kata yang mengacu pada manusia, binatang, benda dan konsep atau pengertian.
2. Kata Kerja (verba)

   Adalah yang menyatakan tindakan.
3. Kata Sifat (adjektiva)

Adalah kata yang memberi keterangan yang lebih khusus tentang sesuatu yang dinyatakan oleh manusia dalam kalimat.

4. Kata Keterangan (adverbial)
   Adalah kategori yang mendampingi adjektiva, numerilia atau preposisi dalam kontruksi sintksis.
5. Kata Tugas
   Adalah kata yang hanya memiliki arti gramatikal dan tidak memiliki arti leksikal.

2.4 Kalimat

Kalimat adalah satuan bahasa berupa kata atau rangkaian kata yang dapat berdiri sendiri dan menyatakan makna yang lengkap. Kalimat adalah satuan bahasa terkecil yang mengungkapkan pikiran yang utuh, baik dengan cara lisan maupun tulisan. Dalam wujud lisan, kalimat diucapkan dengan suara naik turun, dan keras lembut, disela jeda, dan diakhiri dengan intonasi akhir. Sedangkan dalam wujud tulisan berhuruf latin, kalimat dimulai dengan huruf kapital dan diakhiri dengan tanda titik (.) untuk menyatakan kalimat berita atau yang bersifat informatif, tanda tanya (?) untuk menyatakan pertanyaan dan tanda seru (!) untuk menyatakan kalimat perintah. Sekurang-kurangnya kalimat dalam ragam resmi, baik lisan maupun tertulis, harus memiliki sebuah subjek (S) dan sebuah predikat (P). Jika tidak memiliki kedua unsur tersebut, pernyataan itu bukanlah kalimat melainkan hanya sebuah frasa [7].

2.5 Paragraf

Paragraf disebut juga alinea. Paragraf merupakan inti penuangan pikiran dalam sebuah karangan. Kalimat-kalimat dalam paragraf memperlihatkan kesatuan pikiran atau mempunya keterkaitan dalam membentuk gagasan atau topik. Paragraf dapat terdiri dari satu kalimat/kumpulan kalimat, akan tetapi yang berhubungan antara yang satu dengan yang lain dalam suatu rankaian yang membentuk suatu kalimat, dan dapat disebut juga dengan penuangan ide dari penulis melalui kalimat/kumpulan kalimat yang satu dengan yang lainnya, yang berkaitan dan juga

hanya memiliki satu tema. Paragraf juga dapat disebut sebagai karangan yang singkat. Terdapat dua syarat dalam membentuk paragraf.

Terdapat dua syarat dalam membentuk paragraf.

1. Menulis pernyataan (kalimat) tentang pokok bahasan dengan baik.
2. Menggali pola susunan rincian dengan patut.

Ciri-ciri paragraf.

1. Kalimat pertama bertekuk/letaknya agak dalam, ke dalam lima ketukan spasi untuk jenis karangan yang biasa.
2. Paragraf memakai pikiran utama yang dinyatakan dalam kalimat topik.
3. Setiap paragraf memakai sebuah kalimat topik dan juga selebihnya merupakan kalimat pengembang yang mempunyai fungsi menjelaskan, menguraikan atau pun menerangkan pikiran utama yang terdapat dalam kalimat topik.
4. Paragraf memakai pikiran penjelas yang dinyatakan dalam kalimat penjelas.

2.6 Berita

Berita adalah laporan mengenai fakta atau ide terbaru yang benar, dan penting bagi sebagian besar khalayak, melalui media berkala seperti surat kabar, radio, televisi, atau media internet.

Berita atau news adalah laporan mengenai suatu peristiwa atau kejadia yang terbaru (aktaul), laporan mengenai fakta-fakta yang actual, menarik perhatian, dinilai penting, atau luar biasa.

Susunan berita umumnya terdiri dari empat bagian, yaitu:

1. Headline, kepala berita atau judul berita.
2. Dateline, yaitu waktu dan nama tempat berita dibuat atau diperoleh.
3. Teras berita (*Lead*), Adalah bagian berita yang terletak dialinea atau paragraf pertama, setelah head dan dateline sebelum badan atau isi berita. Biasanya berisi fakta penting dengan mengedepankan unsur 5W+1H (*what*, *who*, *when*, *where*, *why*, dan *how*).
4. Tubuh berita (*News Body*), Berisi penjelasan atau uraian rinci unsur 5W+1H, baik yang sudah dikemukakan dalam teras berita maupun yang belum

diungkapkan. Penulisan tubuh berita untuk melanjutkan apa yang sudah tertuang dalam teras yang mencerminkan pokok-pokok terpenting isi berita. Biasanya berupa kutipan dari isi berita atau kutipan isi pembicaraan nara sumber yang paling menarik.

2.7 Algoritma Peringkasan Teks

2.7.1 *Preprocessing*

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah ditahapan berikutnya. Input awal proses ini adalah berupa dokumen. Pada umumnya *preprocessing* memiliki beberapa tahapan yaitu *case folding, tokenizing, stop word removal, stemming,* dan lain-lain. *Preprocessing* pada penelitian ini terdiri dari beberapa tahapan, yaitu proses pemecahan kalimat, proses *case folding,* proses *tokenizing* kata, dan proses *stop word removal.*

1. Pemecahan Kalimat

   Memecahkan dokumen menjadi kalimat-kalimat merupakan langkah awal tahapan *preprocessing.* Pemecahan kalimat yaitu proses memecahkan *string* teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Dalam memecahkan dokumen menjadi kalimat-kalimat menggunakan fungsi split(), dengan tanda titik ".", tanda tanya "?", dan tanda seru "!" pada dokumen [8].

Tabel 2. 1 Contoh Pemecahan Kalimat

Kalimat	Hasil pemecahan kalimat
Manajemen transaksi elektronik. Pengetahuan antar individu. Dalam manajemen pengetahuan, terdapat transfer pengetahuan elektronik. Tahun 2015 pembelian barang elektronik meningkat pesat.	-Manajemen transaksi elektronik -- -Pengetahuan antar individu -Dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik. -Tahun 2015 pembelian barang elektronik meningkat pesat.

2. *Case Foldin*

   Casa folding adalah tahapan pemrosesan teks dimana semua teks diubah kedalam *case* yang sama. Pada penelitian ini semua huruf dalam teks dokumen diubah representasinya menjadi huruf kecil semua [8].

Tabel 2. 2 Contoh *Case Folding*

Hasil pemecahan Kalimat	Hasil *case folding*
-Manajemen transaksi elektronik -Pengetahuan antar individu. -Dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik. -Tahun 2015 pembelian barang elektronik meningkat pesat.	-manajemen transaksi elektronik -pengetahuan antar individu -dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik -tahun 2015 pembelian barang elektronik meningkat pesat

3. *Filtering*

   Data teks dalam dokumen yang sebelumnya sudah diubah kedalam huruf kecil semua. Selanjutnya dilakukan proses *filtering* teks. *Filtering* adalah tahapan pemrosesan teks dimana semua teks selain karakter “a” sampai “z” akan dihilangkan dan hanya menerima spasi [8].

Tabel 2. 3 Contoh *Filtering*

Hasil *case folding*	Hasil *filtering*
-manajemen transaksi elektronik -pengetahuan antar individu -dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik -tahun 2015 pembelian barang elektronik meningkat pesat	-manajemen transaksi elektronik -pengetahuan antar individu -dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik -tahun pembelian barang elektronik meningkat pesat

4. *Tokenizing*

   Tokenizing adalah proses pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men-*scan* kalimat dengan pemisah (*delemiter*) *white space* (spasi, tab, dan *newline*) [8].

Tabel 2. 4 Contoh Tokenizing

Hasil *case folding*	**Hasil *tokenizing***
-manajemen transaksi elektronik -pengetahuan antar individu -dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik -tahun pembelian barang elektronik meningkat pesat	manajemen transaksi elektronik pengetahuan antar individu dalam manajemen pengetahuan terdapat transfer pengetahuan elektronik tahun pembelian barang elektronik meningkat pesat

5. *Stopword Removal*

   Penghapusan *stopword* merupakan proses penghilangan kata *stopword. Stopword* adalah kata-kata yang sering kali muncul dalam dekumen namun arti

dari kata-kata tersebut tidak desktriptif dan tidak memiliki keterkaitan dengan tema tertentu. Misal “di”, “oleh”, “sebuah”, “karena” dan lain sebagainya [8].

Tabel 2. 5 Contoh Stopword Removal

Kalimat hasil *tokenizing*	**Hasil *Stopword Removal***
manajemen	manajemen
transaksi	transaksi
elektronik	elektronik
pengetahuan	pengetahuan
antar	individu
individu	transfer
dalam	tahun
manajemen	pembeli
pengetahuan	meningkat
terdapat	pesat
transfer	
pengetahuan	
elektronik	
tahun	
pembelian	
barang	
elektronik	
meningkat	
pesat	

6. *Stemming*

Menurut Zaman B. dan E Winarko *stemming* adalah proses pemetaan dari penguraian berbagai bentuk kata baik itu *prefix, suffix,* maupun gabungan antara *prefix* dan *suffix (cofix),* menjadi bentuk kata dasarnya.

Algoritma *stemmer* yang diperkenalkan Nazief dan Adriani didefinisikan sebagai berikut :

1. Diawal proses *stemming* dan setiap langkah yang selanjutnya dilakukan, lakukan pengecekan hasil proses *stemming* kata yang di-*input*-kan pada langkah tersebut kakamus kata dasar. Jika kata ditemukan, berarti kata tersebut sudah berbentuk kata dasar dan proses *stemming* dihentikan. Jika tidak ditemukan, maka langkah selanjutnya dilakukan.
2. Hilangkan *Inflection Suffixes* (“-lah”,”-kah”,”-ku”,”-mu”, atau “-nya”). Jika berupa *particle* (“-lah”,”-kah”,”-tah”, atau ”-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”,”-mu”, atau “nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an”, atau “-kan”). Jika kita ditemukan dikamus, maka algoritma berhati. Jika tidak maka kelangkah 3a.
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
    b. Akhiran yang dihapus (“-i”, “-an”, atau “-kan”) dikembangkan, lanjut kelangkah 4.
4. Hilangkah *Derivation Prefixes.*
    a. Langkah 4 berhenti jika :
        1. Terjadi kombinasi awalan dan akhiran yang terlarang.
        2. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
        3. Tiga awalan telah dihilangkan.
    b. Identifikasikan ntipe awalan dan hilangkan. Awalan terdiri dari dua tipe :
        1. Standar (“di-”, “ke-”, “se-”) yang dapat langsung dihilangkan dari kata.
        2. Kompleks (“me-”, “be-”, “pe-”, “te-”) adalah tipe-tipe awalan yang dapat bermofologi sesuai kata dasar yang mengikutinya.

Oleh karena itu, gunakan aturan pada Tabel 2.5 untuk mendapatkan hasil pemenggalan yang tepat.

c. Cari kata yang telah dihilangkan awalannya ini didalam kamus kata dasar. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.

5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada table 2.5. *Recoding* dilakukan dengan menambahkan karekater *recoding* diawal kata yang dipenggal. Pada table 2.5 karakter *recoding* adalah huruf kecil setelah tanda hubung ("-") dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata "menangkap" (aturan 15), setelah dipenggal menjadi "nangkap". Karena tidak valid, maka *recoding* dilakukan dan menghasilkan kata "tangkap".
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Tabel 2. 6 Aturan Pemenggalan Awalan Stemmer Nazief dan Adriani

Aturan	Format Kata	Pemenggalan
1.	berV…	ber-V…| ber-rV…
2.	berCAP…	ber-CAP…dimana C!="r" & P!="er"
3.	berCAerV	ber-CaerV…dimana C!="r"
4.	Belajar	bel-ajar
5.	berC1erC2	be-C1erC2…dimana C1!={"r"|"1}
6.	terV…	ter-V…| te-rV…
7.	terCerV	ter-CerV…dimana C!="r"
8.	terCP…	ter-CP…dimana C!="r" dan P!="er"
9.	teC1erC2…	te-C1erC2…dimana C1!="r"
10.	me{1|r|w|y}V…	me-{1|r|w|y}V…
11.	mem{b|f|v}…	mem-{b|f|v}…
12.	mempe{r|l}	mem-pe…

13.	mem{rV|V}…	me-m{rV|V}…| me-p{rV|V}…
14.	men{c|d|j|z}…	men-{c|d|j|z}…
15.	menV…	me-nV…| me-tV
16.	meng{g|h|q}…	meng-{g|h|q}…
17.	mengV…	meng-V… | meng-kV…
18.	menyV…	meny-sV…
19.	mempV…	mem-pV… dimana V!=”e”
20.	pe{w|y}V…	pe-{w|y}V…
21.	perV…	per-V… | pe-rV…
22.	perCAP	per-CAP… dimana C!=”r” dan P!=”er”
23.	perCAerV…	per-CAerV… dimana C!=”r”
24.	pem{b|f|V}…	pem-{b|f|V}…
25.	pem{rV|V}…	pe-m{rV|V}… | pe-p{rV|V}…
26.	pen{c|d|j|z}…	pen-{c|d|j|z}…
27.	penV…	pe-nV… | pe-tV…
28.	peng{g|h|d}…	peng-{g|h|d}…
29.	pengV…	peng-V… | peng-kV…
30.	penyV…	peny-sV…
31.	pelV…	pe-lV… kecuali “pelajar” yang menghasilkan “ajar”
32.	peCerV…	per-erV… dimana C!={r|w|y|l|m|n}
33.	peCP…	pe-CP… dimana C!={r|w|y|l|m|n} dan P!=”er”

Keterangan simbol huruf :

C : huruf konsonan

V : huruf vocal

A : huruf vocal atau konsonan

P : partikel atau fragmen dari suatu kata, misalnya “er”

Tabel 2. 7 Contoh *Stemming*

Kalimat hasil *Stopword Removal*	Hasil *Stemming*
manajemen	manajemen
transaksi	transaksi
elektronik	elektronik
pengetahuan	tahu
individu	individu
transfer	transfer
tahun	tahun
pembeli	beli
meningkat	tingkat
pesat	pesat

2.8 *Extraction Feature*

Extraction feature merupakan proses pengolahan kalimat menggunakan beberapa fitur sehingga menghasilkan nilai-nilai yang nantinya akan digunakan untuk meringkas dokumen. *Extraction feature* akan dilakukan dari data hasil *stemming. Extraction feature* untuk penelitian ini meliputi kemunculan kata, bobot kata, dan *cosine similarity.*

a. Kemunculan Kata

Merupakan kumunculan sebuah kata yang terdapat didalam sebuah dokumen (kalimat). Jika kata yang dicari ada dokumen (kalimat) ada, maka akan diberikan nilai 1 jika tidak akan diberikan nilai 0.

b. Bobot Kata

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen (kalimat). Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini

menggunakan dua konsep untuk perhitungan bobot, yaitu *term frequency* (TF) merupakan frekuensi kemunculan kata (*t*) pada kalimat (*d*). *Document frequency* (DF) adalah benyaknya kalimat dimana suatu kata (*t*) muncul [1]. Frekuensi kemunculan kata didalam dokumen yang diberikan menunjukkan seberapa penting kata itu didalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen. Pada algoritma TF-IDF digunakan rumus untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci dengan rumus yaitu :

$$W = D/df * IDF \quad \textbf{(2.1)}$$

Dimana:

W = bobot dokumen ke-d terhadap kata ke-t

Tf = banyaknya kata yang dicari pada sebuah dokumen

IDF = log (D/df)

D = total dokumen

df = jumlah banyak kata yang muncul dari setiap dokumen (kalimat)

c. Perkalian Bobot Kata Pada Judul dengan Bobot Kata Pada Kalimat

Merupakan proses untuk mendapatkan nilai bobot antar judul dengan kalimat yang hasilnya merupakan nilai kelanjutan dari proses berikutnya.

$$bobot\ W = W(Q) * W(i..n) \quad \textbf{(2.2)}$$

Keterangan :

W (Q) = bobot judul

W (i..n) = bobot kalimat

d. Panjang kalimat

Merupakan proses untuk mengetahui rasio dalam jumlah kata yang terjadi dalam kalimat atas jumlah kata yang terjadi dalam kalimat dokumen terpanjang.

$$Panjang\ Kalimat = W(i..n)\text{^}2 \qquad \textbf{(2.3)}$$

Keterangan :

W(i..n) = bobot kalimat

e. *Cosine Similarity*

Cosine similarity digunakan untuk menghitung relevansi *query* dengan dokumen. Nilai relevansi didapat dengan mengukur kesamaan atara 2 vektor, yaitu vector *query* dan vector dokumen. Semakin besar nilai relevansi, maka *query* dan dokumen akan semakin relevan [1].

Menurut Patidar, ukuran kesamaan (*similarity measure*) adalah jarak antara berbagai titik data. *Similarity measure* juga digunakan dalam mengukur kesamaan antara set berdasarkan persimpangan dua set. *Similary measure* juga dikenal sebagai fungsi yang menghitung tingkat kesamaan antara sepasang objek teks. Singkatnya, *similarity* adalah jumlah yang mencerminkan kekuatan hubungan antara dua data [8].

Cosine similiarity merupakan dasar perhitungan untuk mendapatkan nilai relevansi antara *query* dengan dokumen dan relevansi antara dokumen. *Cosine similarity* adalah *cosin* sudut θ antara vector. *Cosine similarity* memiliki rumus seperti dibawah ini :

$$sim(S_1, S_2) = \frac{\sum_i t_{1i} t_{2i}}{\sqrt{\sum_i t_{1i}^2}\ X\ \sqrt{\sum_i t_{2i}^2}} \qquad \textbf{(2.4)}$$

Keterangan :

S_1 = vector bobot kata yang menjadi kandidat

S_2 = vector bobot kata selain kandidat

Dimana t_i merupakan bobot kata dari kata w_i.

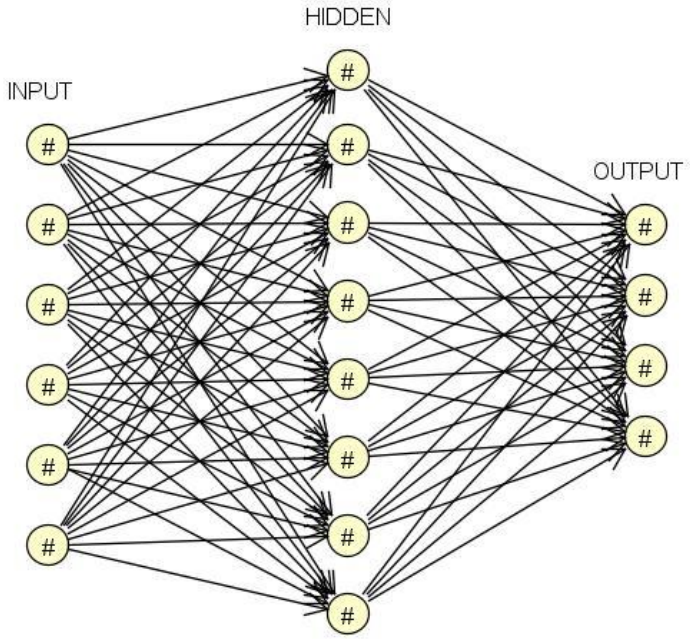
2.9 Hidden Markov Model

Hidden markov model adalah pengembangan model statistic dari model markov. Model ini dikembangkang pertama kali oleh Andreyevich Markov, seorang ilmuan Rusia pada awal abad 20, *Markov model* bisa disebut *Markov Chain* atau *Markov Process* dan merupakan bagian dari proses stokastik yang memiliki property *Markov.* Dengan demikian properti tersebut berarti apabila diberikan inputan keadaan saat ini, keadaan akan datang dapat diprediksi dan ia lepas dari keadaan dimasa lampau. Artinya, deskripsi kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem dimasa depan. Dengan kata lain, kondisi masa depan ditunjukkan dengan menggunakan probabilitas bukan dengan determinitas [9].

HMM adalah sebuah proses stokastik ganda dimana salah satu prosesnya tidak dapat diobservasi (*hidden*). Proses yang tidak dapat diobservasi ini hanya dapat diobservasi melalui proses yang dapat diobservasi. Sebuah HMM menggabungkan dua atau lebih rantai *markov* dengan hanya satu rantai yang terdiri dari *state* yang dapat diobserbasi dan rantai lainnya membentuk *state* yang tidak dapat diobservasi (*hidden*), yang memperngaruhi hasil dari *state* yang dapat diobservasi. Jika $x = \{x_{1,}\, x_{2,}\, x_{3,} \ldots\ldots\ x_{T,}\}$ adalah sebuah proses *markov,* $O = \{O_{1,}\, O_{2,}\, O_{3,} \ldots . O_{T,}\}$ adalah sebuah fungsi dari *X*, maka *X* adalah sebuah HMM yang dapat diobservasi melalui *O*, atau dapat ditulis *O = f(X)* untuk suatu fungsi *f.* Parameter *X* menyatakan *state process* yang tersembunyi (*hidden*), sementara parameter *O* menyatakan *observation process* yang dapat diobservasi. Ada dua parameter yang berkaitan dengan setiap *state* pada HMM sebagai berikut:

5. *Transition probabilities* : menelaskan peluang perpindahan dari *state* saat ini ke *state* berikutnya.
6. *Emision probabilities* : menjelaskan peluang dari *output* berbeda yang mungkin dari suatu *state.*

Untuk ilustrasi HMM dapat dilhat pada gambar 2.1 berikut :



Gambar 2. 1 Ilustrasi HMM

HMM dikenal dengan notasi persamaan (**2.5**) dengan A berukuran *N x N* dan B *N x M.*

$$\boldsymbol{\lambda = (A, B, \pi)} \qquad \textbf{(2.5)}$$

Dimana :

1. A merupakan matriks peluang transisi $A = (a_{ij})$ yang merupakan himpunan distribuasi kemungkinan perpindahan *state* (*transition probabiliy*).
2. B merupakan himpunan distribusi kemungkinan simbol pengamatan pada *state j*. Distribusi peluang observasi pada saat t, pada keadaan i, yang biasa dikenal dengan matrik emisi $B = \{b_i(k)\}$ dimana $b_i^k = b_i(k) = P(O_t = K|X_t = i), 1 \leq i \leq N, 1 \leq k \leq M$.
3. $\pi = \{x_i\}\, dimana\, x_i = P(q_1 = S_t)$ merupakan himpunan distribusi probabilitas awal dengan $\pi(i) = P(X_1 = i), 1 \leq i \leq N$.

2.10 Algoritma *Forward*

Algoritma *forward* adalah proses iterasi yang didasarkan pada perhitungan peluang bersyarat melalui sifat-sifat pada peluang. Algoritma *forward* digunakan untuk menghitung probabilitas barisan yang terobservasi. Bila diketahui sebuah model $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$ dan sebuah barisan observasi $O = \{ O_1, O_2, \dots . O_r \}$, kemudian akan dihitung $P(O|\lambda)$ yang dapat ditulis sebagai berikut [9]:

$$P(O|\lambda) = \sum_X P\,(O|X, \lambda)\; P(X|\lambda) \qquad \textbf{(2.6)}$$

Menurut Rabiner (1989), secara umum algoritma maju (*forward*) terdiri dari atas tiga bagian, yaitu :

1. Tahap inisialisasi

   $\alpha_1(i) = \pi(i) b_i(O_1) \;\; dimana \; 1 \le i \le N$ **(2.7)**

2. Tahap induksi

   $\alpha_{t+1} = \{\Sigma_{i=1}^{N} \alpha_t(i) \alpha_{ij}\} b_j(O_{t+1})$ **(2.8)**

   $dimana \;\; j = 1, \dots, N \;\; . \;\; t = 1, \dots, T-1$

3. Tahap terminasi

   $P(O|\lambda) = \Sigma_{i=1}^{N} \alpha_T(i)$ **(2.9)**

2.11 Algoritma *Viterbi*

Algoritma *viterbi* adalah algoritma *dynamic programming* untuk menemukan kemungkinan rangkaian *state* yang tersembunyi (bisa disebut *viterbi path*) yang dihasilkan pada rangkaian pengamatan kejadian dan menentukan barisan *state* yang mana yang optimal terutama dalam lingkungan HMM.

Didefinisasikan $\arg max_y \{z\}$ **(2.10)**

Yaitu argument *y* bersesuaian dengan nilai maksimum dari *z.* Algoritma *viterbi* memaksimalkan $P(X, O)$ dan probabilitas bersyarat $P(X|O)$. Algoritma *viterbi* terdiri atas empat tahap yaitu :

1. Tahap inisialisasi

   $\delta_1(i) = b_i(O_1)\pi(i)$ **(2.11)**

   pada tahap ini $\psi_1(i) = 0$ **(2.12)**

2. Tahap rekursi

   $\delta_t(j) = \max_{1 \le i \le N} \{a_{ij}\delta_{t-1}(i)\}$ **(2.13)**

3. Tahap terminasi

   $P^* = \max_{1 \le i \le N}\{\delta_r(i)\}$ **(2.14)**

   $X_t^* = \max_{1 \le i \le N}\{\delta_T(i)\}$ **(2.15)**

4. Tahap *backtracking*

   $X_t^* = \psi_{t+1}(X_{t+1}^*),\ t = T-1, T-2, \dots, 1$ **(2.16)**

2.12 Evaluasi Peringkasan Teks

Menurut Nedunchelian (2011:5), proses evaluasi hasil *text summarization* dilakukan menggunakan tiga parameter yaitu *Precision*, *recall*, dan *F-measure*. Cara mengevaluasi dengan membandingkan ringkasan otomatis hasil sistem dengan ringkasan manual [8].

1. *Precession*

   Merupakan perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi yang terambil oleh sistem baik yang relevan maupun tidak. Persamaan *precision* ditujukan pada persamaan berikut :

   $$P = \frac{correct}{(correct + wrong)} \quad \textbf{(2.17)}$$

2. *Recall*

   Merupakan perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi relevan yang ada dalam koleksi informasi (baik yang terambil atau tidak terambil oleh sistem).

   $$R = \frac{correct}{(correct + missed)} \quad \textbf{(2.18)}$$

Keterangan :

Correct : jumlah kalimat yang diekstrak oleh sistem dan manusia

Wrong : jumlah kalimat yang diekstrak oleh sistem tetapi tidak diekstrak oleh manusia

Missed : jumlah kalimat yang diekstrak oleh manusia tetapi tidak diekstrak oleh sistem

3. *F-measure*

   Merupakan hubungan antara *recall* dan *precision* yang mempresentasikan akurasi sistem. Persamaan *F-measure* seperti berikut :

$$F = \frac{2*R*P}{(R+P)} \qquad \textbf{(2.19)}$$

2.13 Pemrograman Berorientasi Objek

Metodologi beorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek [10].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk massalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan

Dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Sifat pewaris dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut ini adalah beberapa konsep dasar metodologi berorientasi objek:

1. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek. Kelas secara fisik adalah berkas atau file yang berisi kode program, di mana kode program merupakan semua hal yang terkait dengan nama kelas.

2. Objek (*objek*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, 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. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat dari segi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi.

3. Metode (*method*)

   Metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode dapat berasal dari *event*, aktivitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata. Contoh metode misalnya *read*, *write*, *move*, *copy*.

4. Atribut (attribute)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut sebaiknya bersifat privat untuk menjaga konsep enkapsulasi.

5. Abstraksi (abstraction)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek yang lain mengetahui cara kerjanya.

7. Pewaris (inheritance)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka atau interface sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama

persis dengan yang ada pada antarmuka. Antarmuka atau *interface* biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

2.14 *Unified Modelling Language* (UML)

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 [11].

2.14.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Diagram ini juga mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case*, digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu:

1. Aktor

   Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case*

   Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.14.2 *Activity Diagram*

Activity diagram merupakan diagram menggambarkan *workflow* (alur kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

2.14.3 *Sequence Diagram*

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

2.14.4 *Class Diagram*

Class diagram atau kelas diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode. Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak dapat sinkron.

Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main

   Kelas main merupakan kelas yang mempunyai fungsi awal yang dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*)

   Kelas ini mendefinisikan dan mengatur tampilan ke pengguna.
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas ini menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*. Kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data (model)

   Kelas ini digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas.

2.15 Perangkat Lunak Pendukung

a. JAVA

JAVA adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pemrogram Java banyak mengambil keuntungan dari kumpulan kelas di pustaka kelas Java, yang disebut dengan *Java Application Programming Interface* (API). Kelas-kelas ini diorganisasikan menjadi sekelompok yang disebut paket *(package).* Java API telah menyediakan fungsionalitas yang memadai untuk menciptakan *applet* dan aplikasi canggih. Jadi ada dua hal yang harus dipelajari dalam Java, yaitu mempelajari bahasa Java dan bagaimana mempergunakan kelas pada Java API. Kelas merupakan satu-satunya cara menyatakan bagian eksekusi program, tidak ada cara lain. Pada Java program javac untuk mengkompilasi file kode sumber Java menjadi kelas-kelas *bytecode*. File kode sumber mempunyai ekstensi *.java. Kompilator javac menghasilkan file *bytecode* kelas dengan ekstensi *.class. Interpreter merupakan modul utama sistem Java yang digunakan aplikasi Java dan menjalankan program *bytecode* Java.

Beberapa keunggulan java yaitu java merupakan bahasa yang sederhana. Java dirancang agar mudah dipelajari dan digunakan secara efektif. Java tidak menyediakan fitur-fitur rumit bahasa pemrograman tingkat tinggi, serta banyak pekerjaan pemrograman yang mulanya harus dilakukan manual, sekarang digantikan dikerjakan Java secara otomatis seperti dealokasi memori. Bagi pemrogram yang sudah mengenal bahasa C++ akan cepat belajar susunan bahasa Java namun harus waspada karena mungkin Java mengambil arah (semantiks) yang berbeda dibanding C++.

b. NetBeans IDE (Versi 81)

NetBeanss merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasiskan *Java* dari Sun Microsystems yang berjalan diatas *Swing. Swing* merupakan sebuah teknologi *java* untuk pengembangan aplikasi *desktop* yang dapat berjalan pada berbagai macam *plaform* seperti *windows, linux, Mac OS X,* dan *Solaris.* Sebuah IDE merupakan lingkup pemrograman yang diintegrasikan kedalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), suatu kode editor atau *text* , suatu *compiler* dan suatu *debugger.*

Netbeands juga digunakan oleh sang *programmer* untuk menulis, meng-*compile,* mencari kesalahan dan menyebarkan program netbeands yang ditulis dalam bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat *professional desktop, enterprise, web, and mobile applications* dengan *Java language, C/C++,* dan bahakan *dynamic languages* seperti PHP, *JavaScript, Groovy,* dan *Ruby* [12].

c. Visio (versi 2013)

Microsoft Visio (atau sering disebut Visio) adalah sebuah program aplikasi komputer yang sering digunakan untuk membuat diagram, diagram alir (*flowchart*), *brainstorm*, dan skema jaringan yang dirilis oleh Microsoft Corporation. Aplikasi ini menggunakan grafik vektor untuk membuat diagram-diagramnya.

Visio aslinya bukanlah buatan Microsoft Corporation, melainkan buatan Visio Corporation, yang diakusisisi oleh Microsoft pada tahun 2000. Versi yang telah menggunakan nama Microsoft Visio adalah Visio 2002, Visio 2003, Visio 2007, Visio 2010, Visio 2013, dan Visio 2016 yang merupakan versi terbaru. Visio 2013 Standard dan Professional menawarkan antarmuka pengguna yang sama, tapi seri Professional menawarkan lebih banyak pilihan template untuk pembuatan diagram yang lebih lanjut dan juga penataan letak (layout). Selain

itu, edisi Professional juga memudahkan pengguna untuk mengoneksikan diagram-diagram buatan mereka terhadap beberapa sumber data dan juga menampilkan informasi secara visual dengan menggunakan grafik [13].