BAB 2
LANDASAN TEORI

2.1. Peringkas Teks Otomatis

Ringkasan adalah sebuah teks yang dihasilkan dari sebuah dokumen atau lebih yang menyatakan informasi penting dari dokumen asli, dan sebuah ringkasan memiliki ukuran yang relatif lebih pendek dari dokumen asli [**7**]. Tujuan utama dari ringkasan adalah menghasilkan sebuah intisari atau informasi inti yang terdapat dalam dokumen asli dalam bentuk yang lebih kecil agar pembaca dapat mendapatkan informasi penting yang terdapat dalam dokumen dengan lebih cepat. Proses peringkasan teks secara otomatis sendiri merupakan proses peringkasan pada suatu dokumen dengan menggunakan bantuan komputer.

2.1.1. Pendekatan Peringkasan Teks Secara Otomatis

Terdapat dua buah pendekatan yang dilakukan untuk menghasilkan sebuah ringkasan secara otomatis yaitu :

a. Ekstraksi

   Pada teknik ekstraksi, sistem 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.

b. Abstraksi

   Teknik abstraksi 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.

Pada penelitian ini teknik atau pendekatan peringkasan teks secara otomatis yang digunakan adalah peringkasan secara ekstraksi dengan

menggunakan computer sebagai alat bantu untuk menghasilkan ringkasan dari dokumen asli.

2.2. Preprocessing

Preprocessing merupakan tahapan awal untuk menghasilkan sebuah ringkasan. Teks masukan yang akan di ringkas sebelumnya harus melalui tahap untuk membuang berbagai macam jenis *noise* atau kata – kata yang di anggap tidak penting dalam ringkasan yang masih terdapat pada teks masukan [8].

Dalam penerapan *preprocessing* terdapat beberapa tahap yang harus dilalui dimulai dari tahapan tokenisasi, penghilangan *stop words* sampai dengan tahapan *stemming*, selain tahapan itu juga biasanya di tambahkan beberapa tahapan lain untuk kasus tertentu seperi *case folding* dan penghilangan kata yang jarang dimunculkan atau kata dengan frekuensi kemunculan yang kecil [9].

Proses *preprocessing* yang diterapkan pada penelitian ini meliputi beberapa macam tahap yaitu :

a. Pemecahan Kalimat

   Pada tahapan ini teks masukan dipecah menjadi beberapa kalimat berdasarkan delimiter atau pemisah yang sudah ditetapkan. Pemisah tersebut adalah tanda titik(.),tanda seru(!) dan tanda Tanya(?).

b. *Case Folding*

   Pada tahapan ini dilakukan proses penyamaan *case* atau besar kecil dari setiap huruf yang terdapat pada teks masukan yang telah di pecah menjadi beberapa kalimat. Selain itu pada tahap ini juga di lakukakan proses pembuangan pada tanda baca,simbol dan angka yang terdapat pada data masukan.

c. *Tokenizing*

   Pada tahapan ini teks masukan hasil dari proses *case folding* dipecah kembali menjadi beberapa kalimat berdasarkan spasi yang terdapat pada kalimat hasil *case folding*.

d. Stop Words Removal

Pada tahapan ini dilakukan proses penghapusan *stop words* yang masih terdapat pada teks hasil dari proses *tokenizing*. *Stop words* merupakan sebuah kata yang dianggap tidak terlalu penting dalam proses peringkasan atau kata yang tidak mempunyai arti dalam sebuah dokumen [10]. Contoh dari *stop words* dalam bahasa Indonesia adalah : dan,atau,jika dan sebagainya. Tujuan utama dilakukan tahap ini adalah untuk mengurangi dimensi dari teks masukan sehingga proses peringkasan dapat berjalan dengan lebih mudah [8].

c. Stemming

Stemming merupakan proses yang menyediakan pemetaan varian morfologi yang berbeda dari suatu kataker akar katanya dengan cara [11]. proses ini melakukan pemetaan dari penguraian berbagai bentuk kata baik itu *prefix*, *sufix*, maupun gabungan antara *prefix* dan *sufix* (*confix*), menjadi bentuk kata dasarnya. Pada penelitian ini algoritma stemming yang digunakan adalah algoritma stemming Nazief dan Adriani

2.2.1. Algoritma Nazief dan Adriani

Algoritma Nazief dan Adriani merupakan algoritma stemming yang di gunakan untuk proses stemming terhadap teks berbahasa Indonesia [12]. Cara Kerja dari Algoritma ini adalah :

1. Di awal proses *stemming* dan setiap langkah yang selanjutnya dilakukan, lakukan pengecekan hasil proses stemming kata yang di-input-kan pada langkah tersebut ke kamus 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 *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a

a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.

4. Hilangkan *derivation prefixes*.
   a. Langkah 4 berhenti jika :
      i. Terjadi kombinasi awalan dan akhiran yang terlarang.
      ii. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      iii. Tiga awalan telah dihilangkan.
   b. Identifikasi tipe awalan dan hilangkan. Awalan terdiri dari dua tipe :
      i. Standar (“di-”, “ke-”, “se-”) yang dapat langsung dihilangkan dari kata.
      ii. Kompleks (“me-”, “be-”, “pe”, “te-”) adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2.1 untuk mendapatkan hasil pemenggalan yang tepat.
   c. Cari kata yang telah dihilangkan awalannya ini di dalam 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 Tabel 2.1. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Pada Tabel 2.1, karakter *recoding* adalah huruf kecil setelah tanda hubung („-“) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 13), 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 0.1 Aturan Pemenggalan Awalan Algortima Nazief dan Adriani

Aturan	Format Kata	Pemenggalan
1	berV …	ber-V … | be-rV …
2	BerCAP …	ber-CAP … dimana C!=”r” & P!=”er”
3	berCAerV …	ber-CaerV … dimana C!=”r”
4	Belajar	bel-ajar
5	BeC1erC2 …	be-C1erC2 … dimana C1!={“r”|”l”}
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{l|r|w|y}V …	Me-{l|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!=”r”
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} …

Aturan	Format Kata	Pemenggalan
26	Pen{c|d|j|z} …	Pen-{c|d|j|z} …
27	penV …	Pe-nV … | Pe-tV …
28	Peng{g|h|q} …	Peng-{g|h|q} …
29	pengV …	Peng-V … | Peng-kV …
30	penyV …	Peny-sV …
31	pelV …	Pe-lV … kecuali “pelajar” yang meghasilkan “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 Vokal

A : huruf vokal atau konsonan

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

2.3. Algoritma Term Frequency – Inverse Document Frequency (TF-IDF)

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) 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 kata (*t*) pada kalimat (*d*). *Document frequency* (DF) adalah banyaknya kalimat dimana suatu kata (*t*) muncul.

Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut [4]. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen. Pada aloritma TF-IDF

digunakan rumus untuk menghitung bobot (W) masing masing dokumen terhadap kata kunci dengan rumus yaitu :

$$W = tf * IDF \quad (2.1)$$

Keterangan:

d = dokumen ke-d

t = kata ke-t dari kata kunci

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

tf = banyaknya kata yang dicari pada sebuah dokumen

IDF = Inversed Document Frequency dengan rumus :

$$\text{IDF} = \quad log \frac{N}{df(t)} \quad (2.2)$$

D = total dokumen

df = banyak dokumen yang mengandung kata yang dicari

2.4. Latent Semantic Analysis

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 [8].

Metode ini terinspirasi dari penggunaan latent semantic indexing yang mengimplementaasikan metode singular value decomposition (SVD) untuk menghasilkan sebuah ringkasan. Terdapat 3 tahapan utama dalam proses menghasilkan ringkasan menggunakan metode ini, yaitu :

a. Pembuatan matriks
b. Singular value decomposition
c. Ekstraksi kalimat ringkasan

2.4.1. Pembuatan matriks

Pada tahap ini di buat sebuah matriks berdasarkan dengan teks masukan yang akan di ringkas. Matriks yang dibuat adalah kalimat dengan kata atau matriks m x

n matriks dengan m atau baris matriks di ambil dari kalimat dari teks yang akan di ringkas, sedangkan n atau kolom matriks merupakan kata yang terdapat dalam teks yang telah diberi bobot dengan algoritma tf-idf.

2.4.2. Singular Value Decomposition

Setelah matriks di ciptakan maka tahap selanjutnya adalah mengubah matriks tersebut menjadi lebih kecil dengan menggunakan metode *singular value decomposition* (SVD). Suatu proses dekomposisi akan memfaktorkan sebuah matriks menjadi lebih dari satu matriks. Demikian halnya dengan Dekomposisi Nilai Singular (*Singular Value Decomposition*) atau yang lebih dikenal sebagai SVD, adalah salah satu teknik dekomposisi berkaitan dengan nilai singular (*singular value*) suatu matriks yang merupakan salah satu karakteristik matriks tersebut.Dekomposisi nilai singular matriks riil A mxn adalah faktorisasi.

$$A = USV^T \tag{2.3}$$

Dengan *U* matriks orthogonal *m* x *m*, *V* matriks orthogonal *n* x *n* dan S matriks diagonal *m* x *n* bernilai riil tak negatif yang disebut nilai-nilai singular. Dengan kata lain S = diag (σ1, σ2, … , σn) terurut sehingga σ1 ≥ σ2 ≥ … ≥ σn .

Teorema tersebut juga menyatakan bahwa matriks Amxn dapat dinyatakan sebagai dekomposisi matriks yaitu matriks *U,* S dan *V* . Matriks S merupakan matriks diagonal dengan elemen diagonalnya berupa nilai-nilai singular matriks *A*, sedangkan matriks *U* dan *V* merupakan matriks-matriks yang kolom-kolomnya berupa vektor singular kiri dan vektor singular kanan dari matriks *A* untuk nilai singular yang bersesuaian.

Menentukan SVD meliputi langkah-langkah menentukan nilai eigen dan vektor eigen dari matriks AA^T atau A^TA. Vektor eigen dari A^TA membentuk kolom V, sedangkan vektor eigen dari AA^T membentuk kolom U. Nilai singular dalam S adalah akar pangkat dua dari nilai-nilai eigen matriks AA^T atau A^TA. Nilai singular adalah elemen-elemen diagonal dari S dan disusun dengan urutan menurun [9].

2.4.3. Ekstraksi Ringkasan

Pada proses ini dilakukan tahap pemilihan kalimat yang akan di jadikan sebagai ringkasan. Kalimat yang dipilih diambil dari kalimat yang terdapat pada matriks V^T. Kemudian dilakukan pemilihan kalimat yang akan di jadikan sebagai ringkasan berdasarkan kalimat yang mengandung bobot kata terbesar. Proses pemilihan diulang sebanyak jumlah kalimat yang terdapat dalam matriks.Contoh dari proses ekstraksi menggunakan metode ini adalah sebagai berikut.

2.5. Cross Method Latent Semantic Analysis

Cross method latent semantic analysis merupakan sebuah pegembangan dari metode *latent semantic analysis* yang telah ada sebelumnya. Metode ini dapat menghasilkan ringkasan dari teks masukan yang lebih akurat dibandingkan dengan metode *latent semantic analysis* yang sebelumnya [4]. Tahapan dari metode ini sebenarnya sama seperti metode *latent semantic analysis* sebelumnya yaitu dimulai dari tahap pembuatan matriks, *singular value decomposition* dan ekstraksi ringkasan.

Yang menjadi perbedaan metode ini dengan metode *latent semantic analysis* terdapat pada saat tahap ekstraksi ringkasan. Metode ini menggunakan nilai rata – rata (*average*) dan panjang(*length*) yang di ambil dari matriks V^T dan matriks S. Nilai *average* diambil dari nilai rata – rata dari setiap bobot kata yang terdapat baris matriks V^T. setelah ditemukan nilai rata – rata dari setiap kata yang terdapat pada matriks V^T , langkah selanjutnya adalah mencocokan nilai rata-rata yang didapatkan pada setiap baris dengan nilai dari setiap kata yang terdapat pada baris tersebut. Jika ternyata nilai dari kata tersebut lebih rendah dari nilai rata-rata yang didapat, maka nilai kata tersebut di ubah menjadi nol(0). Tetapi jika tidak maka nilai dari kata tersebut tetap atau tidak diubah.

Setelah tahapan pencarian dan pencocokan nilai rata – rata telah dilakukan, tahap selanjutnya adalah menghitung *length* dari setiap baris dari matriks V^T dengan rumus :

$$length = \sqrt{\sum_{j=1}^{n} V_{ij}^2 * S_{jj}^2} \quad (2.4)$$

Baris - baris pada matriks atau kalimat - kalimat yang mempunyai nilai *length* yang tinggi akan di jadikan sebagai ringkasan.

2.6. Metode Pengujian Hasil Ringkasan

Hasil ringkasan yang dihasilkan oleh peringkas teks otomatis selanjutnya harus melalui tahapan pengujian dan evaluasi untuk mengetahui tingkat akurasi dan ketepatan hasil ringkasan yang dihasilkan. Proses evaluasi hasil ringkasan dapat dikategorikan menjadi dua yaitu metode evaluasi intrinsik dan metode evaluasi ekstrinsik [10].

Pada proses evaluasi secara ekstrinsik, kualitas dari hasil ringkasan dilandaskan pada efek apakah hasil dari ringkasan dapat membantu pada kasus yang diberikan. Sedangkan pada proses evaluasi secara intrinsik, kualitas dari hasil ringkasan berdasarkan dari hasil analisis yang dilakukan pada ringkasan secara langsung. Pada kasus evaluasi intrinsik hasil ringkasan dibandingkan dengan dokumen asli, dari situ akan di analisis seberapa banyak ide utama pada dokumen asli yang terdapat pada hasil ringkasan dengan menyamakan hasil ringkasan dengan hasil ringkasan abstrak atau hasil ringkasan manual yang dilakukan oleh manusia [15].

Pada penelitian ini proses evaluasi yang akan digunakan adalah proses evaluasi secara intrinsik dengan menggunakan metode *precision*, *recall* dan *f - measure* . Nilai *f - measure* dihasikan berdasarkan nilai *precision* dan *recall*. Metode evaluasi ini merupakan metode evaluasi yang sering digunakan dalam proses evaluasi hasil ringkasan. Dalam metode intrinsik, *precision* dan *recall* digunakan untuk mengukur kualitas ringkasan sistem dengan cara membandingkan ringkasan sistem dengan ringkasan manual (buatan manusia). *Precision* adalah tingkat ketepatan hasil ringkasan yang dihasilkan peringkas teks otomatis sedangkan *recall* adalah tingkat keberhasilan ringkasan yang dihasilkan peringkas teks otomatis.Untuk menghitung nilai *precision* dan nilai *recall* digunakan persamaan berikut [15]:

$$Precision = \frac{\sum kalimat\ relevan}{\sum kalimat\ ringkasan\ sistem} \quad (2.5)$$

$$Recall = \frac{\sum kalimat\ relevan}{\sum kalimat\ ringkasan\ manual} \quad (2.6)$$

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

Kalimat relevan = $kalimat\ ringkasan\ sistem \cap kalimat\ ringkasan\ manual$ (2.7)

Setelah diketahui nilai *precision* dan nilai *recall*, tahapan selanjutnya adalah menghitung nilai dari *f – measure* yang merupakan nilai yang digunakan untuk mengukur nilai akurasi dari ringkasan yang dihasilkan dengan menggunakan nilai yang dihasilkan pada perhitungan *precision* dan *recall* sebelumnya, sehingga *f-measure* bisa juga disebut sebagai gabungan atau kombinasi nilai *precision* dan *recall. F-measure* dapat dicari dengan menggunakan persamaan berikut [12]:

$$F - Measure = 2 * \frac{Precision * Recal}{(Presicion + Recall)} \quad (2.8)$$

2.7. PHP : Hypertext Prepocessor (PHP)

PHP adalah bahasa pemrograman *script server-side* yang didesain untuk pengembangan *web.* PHP disebut bahasa pemrograman server side karena PHP yang artinya dijalankan di server, kemudian hasilnya atau *output*nya dikirim ke *client (browser)* . Hal ini berbeda dibandingkan dengan bahasa pemrograman client-side seperti JavaScript yang diproses pada web browser (client).

Pada awalnya PHP merupakan singkatan dari *Personal Home Page.* Sesuai dengan namanya, PHP digunakan untuk membuat website pribadi. Dalam beberapa tahun perkembangannya, PHP menjelma menjadi bahasa pemrograman web yang powerful dan tidak hanya digunakan untuk membuat halaman web sederhana, tetapi juga website populer yang digunakan oleh jutaan orang

Saat ini PHP adalah singkatan dari PHP: Hypertext Preprocessor, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya terdiri dari

singkatan itu sendiri: PHP: Hypertext Preprocessor.PHP dapat digunakan dengan gratis (free) dan bersifat *Open Source*. PHP dirilis dalam lisensi *PHP License*, sedikit berbeda dengan lisensi *GNU General Public License (GPL)* yang biasa digunakan untuk proyek *Open Source*. PHP sendiri sudah mendukung atau dapat berinteraksi dengan beberapa database seperti MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC dan lain – lain [16].