BAB 2
LANDASAN TEORI

2.1. Ringkasan

Ringkasan dalam Kamus Besar Bahasa Indonesia adalah memendekkan (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 [7], pengertian yang lainnya ringkasan adalah representasi akurat dari isi dokumen [7].

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.

Ringkasan dapat dikategorikan dengan set yang berbeda, seperti fungsi, jenis dan sumber dokumen, dll [7].

1) Ringkasan dilihat dari fungsinya antara lain :
    1. Ringkasan indikasi : ringkasan indikatif memberikan informasi tetang topik yang dibahas dalam dokumen sumber. Ringkasan ini menyerupai daftar isi.
    2. Ringkasan informatif : ringkasan informatif bertujuan untuk mencerminkan isi sumber teks, bisa menjelaskan argumen. Ringkasan ini versi pendek dari dokumen.
2) Ringkasan dilihat dari jumlah dokumen antara lain :
    1. Dokumen tunggal : ringkasan satu dokumen.
    2. 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 :
    1. Ringkasan berita : ringkasan artikel berita.

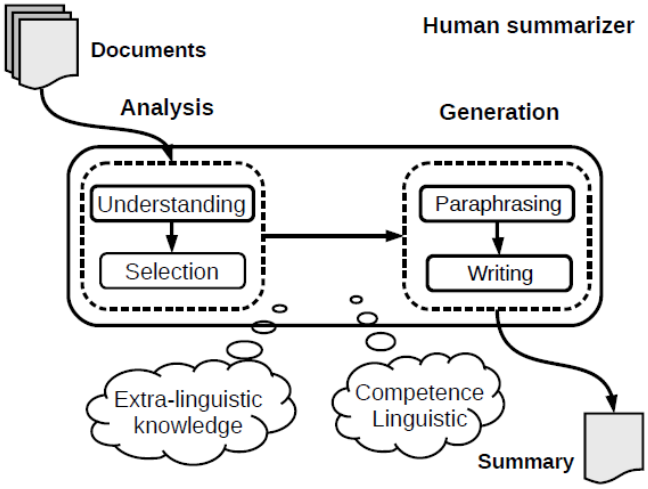
2. Ringkasan khusus : ringkasan dokumen yang mempunyai domain khusus. Contohnya ilmu pengerahuan dan teknologi dll.
3. Ringkasan sastra : ringkasan dokumen narasi, teks-teks sastra dll.
4. Ringkasan ensiklopedis : ringkasan dokumen ensiklopedis, sepeti wikipedia dll.
5. Ringkasan jaringan sosial : ringkasan blog dan dokumen singkat.

4) Ringkasan dilihat dari tipe antara lain :
    1. Ekstrak : Perakitan fragmen dari dokumen asli.
    2. Absrak : meringkas dengan merumuskan, ringkasan diproduksi dengan menulis ulang dan/atau mengutip teks.
    3. Kompresi kalimat : ringkasan yang berisi jumlahh yang sama dari kalimat sebagai teks asli, tapi dengan panjang yang berbeda.
5) Ringkasan dilihat dari jenis ringkasan antara lain :
    1. Ringkasan penulis : ringkasan yang ditulis oleh penulis dari dokumen menurut sudut pandangnya.
    2. Ringkasan ahli : ringkasan yang ditulis oleh selain penulis, seorang yang mengkhususkan diri dalam domain tapi mungkin tidak mengkhususkan diri dalam memproduksi ringkasan.
    3. Ringkasan profesional : ringkasan yang ditulis oleh seorang peringkas ahli.
6) Ringkasan berdasarkan konteks antara lain :
    1. Ringkasan genetik : ringkasan dokumen yang mengabaikan kebutuhan informasi pengguna.
    2. Ringkasan permintaan-dipadu : ringkasan yang dipadukan dengan kebutuhan informasi pengguna.
    3. 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 :
    1. Tanpa profil : ringkasan yang independen dari kebutuhan profil pengguna.
    2. Berdasarkan profil : ringkasan yang ditargetkan pada pengguna yang tertarik dalam domain khusus(kimia, politik internasional, olahraga, ekonomi).

Berdasarkan jenis-jenis ringkasan yang telah di paparkan maka peringkasan multi dokumen adalah peringkasan dari beberapa dokumen dengan sumber yang berbeda dengan tema yang sama (tidak harus relevan) [7]. Peringkasan multi dokumen dapat dilihat sebagai perpanjangan beberapa peringkasan *single* dokumen yang keluarannya menyatu.

Pembentukan ringkasan yang dilakukan oleh manusia dapat dilihat pada Gambar 2.1. Pembentukan ringkasan secara abstraksi yang dilakukan secara manual oleh manusia terbagi ke dalam dua fase, yaitu memahami teks sumber (*analysis*) dan menuliskan kembali hasil pemahaman teks sumber tersebut dalam bentuk sebuah ringkasan (*generation*). Dalam fase *analysis*, 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 [7].



Gambar 2.1 Proses Pembentukan Ringkasan Hasil Buatan Manusia

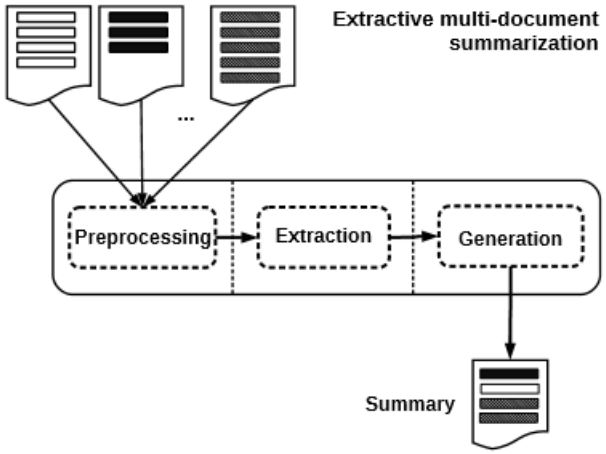
2.2. Peringkasan Teks Otomatis

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 [7]. 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 untuk normalisasi kata-kata dalam teks serta pengurangan kosa kata yang akan diproses untuk memudahkan proses peringkasan.

Setelah proses *preprocessing* dilakukan, langkah selanjutnya adalah merepresentasikan hasil *preprocessing* tersebut menjadi bentuk yang dapat diproses dan dibaca oleh algoritma peringkasan. Salah satu bentuk representasi teks yang sering digunakan adalah representasi dalam bentuk vektor [7]. Vektor tersebut biasanya mengandung nilai penting yang merepresentasikan kata-kata kunci yang penting dalam sebuah teks atau dokumen yang selanjutnya akan diolah oleh

algoritma peringkasan. Skema peringkasan pada multi dokumen bisa dilihat pada gambar 2.2.



Gambar 2.2 Skema Peringkasan Multi Dokumen

Penelitian tentang peringkasan teks otomatis pertama kali dimulai pada tahun 1950-an oleh Hans Peter Luhn untuk meringkas dokumen-dokumen riset dan teknis kimia [7]. Setelah itu pada tahun 1950 hingga 1970-an beberapa riset peringkasan teks otomatis pun mulai dibuat, termasuk karya Edmundson dan Wyllys, Rush, Gerald Francis DeJong, Sparck-Jones, serta Julian Kupiec.

Perkembangan peringkasan teks terus melaju, salah satu risetnya adalah peringkasan teks otomatis menggunakan *machine learning.* Joel Larocca Neto dkk adalah pengembang pertama dari peringkasan teks menggunakan pendekatan *machine learning* dengan menggunakan algoritma *naive bayes* dan C45. Penelitian menggunakan metode *machine learning* untuk peringkasan teks telah banyak dilakukan. Salah satunya Deni Fitriaman, Masayu Leylia Khodra, Bambang Rianto Trilaksono menggunakan algoritma *Support Vector Machine* untuk penelitian peringkasan teks.

2.3. *Preprocessing*

Preprocessing adalah sebuah tahapan yang bisa membuat teks menjadi data yang bisa diolah pada proses berikutnya. *Preprocessing* dalam peringkasan teks otomatis memiliki dua tujuan, yaitu untuk normalisasi kata-kata dalam teks serta

pengurangan kosa kata yang akan diproses untuk memudahkan proses peringkasan [7].

Proses-proses yang akan digunakan pada tahap *preprocessing* pada peringkasan teks otomatis meliputi *case folding, splitting, tokenization* dan *stopword removal.*

2.3.1. Case Folding

Case folding/capitalization adalah proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil atau huruf kapital [8]. *Case folding* dilakukan karena dokumen mengandung berbagai variasi dari bentuk huruf. Variasi huruf harus diseragamkan untuk menghilangkan *noise* pada saat pengambilan informasi. Pada penelitian ini, *case folding* merupakan tahapan awal dari *preprocessing*.

Setelah melakukan *case folding*, dilakukan penyatuan kata entitas yang mempunyai >= 2 kata pada entitas tersebut. Hal ini dilakukan supaya mepermudah untuk proses ekstraksi fitur kalimat pada urutan 4.

2.3.2. *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 lainnya [9]. Cara penentuan kata yang akan di hapus yaitu dengan membawa daftar kata (*stoplist*) dari penelitian yang berkaitan dengan kata pada bahasa Indonesia yang telah dilakukan [4].

2.3.3. Splitting

Splitting merupakan pemecahan teks untuk menjadi pecahan-pecahan paragraf/kalimat [7]. Dalam proses *splitting* melibatkan dua proses, yaitu *sentences splitting* dan *punctuation*. Perbedaan *split sentences* dan *punctuation* terletak pada fungsinya. *Sentences splitting* memecah teks berdasarkan *delimiter*(pembatas yang akan digunakan untuk pemisah), sedangkan *punctuation* menghapus *delimiter* yang tersisa dan tanda baca serta simbol lainnya dari hasil *sentence splitting*.

Sentences Splitting adalah 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].

Setelah melakukan proses *split sentences,* langkah selanjutnya adalah melakukan *punctuation,* yaitu proses menghilangkan tanda baca atau simbol pada kalimat yang telah dipecah [7].

2.3.4. Tokenization

Tokenization adalah proses pemotongan teks berupa kalimat untuk mendapatkan bagian-bagian atau *token-token* tertentu. Secara umum pemecahan kalimat menjadi kumpulan *token* dilakukan dengan melakukan pemindaian kalimat dengan pemisah (*delimiter*) serta *white space* (spasi, *tab,* dan *newline*) [6].

2.4. Ekstraksi Fitur

Fitur ekstraksi adalah cara untuk menghitung skor tiap-tiap kalimat dalam dokumen [5]. Untuk setiap kalimat dalam dokumen, skor kalimat dihitung berdasarkan *fitur ekstraksi* dimana nilai dari tiap-tiap *fitur* dinormalisasikan sehingga berada dalam range[0,1]. Normalisasi ini dilakukan agar nilai dari tiap-tiap *fitur ekstraksi* tidak memiliki gap atau selisih yang besar.

Fitur ekstraksi yang digunakan dalam penelitian ini berdasarkan :

1. Posisi Kalimat (f1)

   Posisi kalimat adalah letak kalimat dalam sebuah paragraf. Kalimat pertama dalam sebuah paragraf adalah yang paling penting [5]. Fitur ini dapat dihitung menggunakan rumus 2.1 :

$$Score_{f1}(s_i) = \frac{N-i}{N} \qquad (2.1)$$

   Dimana :

   N adalah jumlah kalimat dan i adalah posisi kalimat (dimulai dari 0).

2. Kemiripan Antar Kalimat (f2)

   Kemiripan antar kalimat adalah daftar kata-kata yang dapat dicocokkan antara kalimat yang satu dengan kalimat yang lainnya dalam dokumen atau dengan kata lain merupakan kata yang muncul dalam kalimat sama dengan kata

yang muncul dalam kalimat lain [5]. *Fitur* ini dihitung menggunakan rumus (2.2)

$$\text{Score}_{f2}\ (s) = \left| \frac{keyword\ dalam \cap\ keyword\ dalam\ antar\ kalimat}{keyword\ dalam\ s \cup keyword\ dalam\ antar\ kalimat} \right| \quad (2.2)$$

3. Kalimat yang Menyerupai Judul Dokumen (f3)

Kalimat yang menyerupai judul dokumen adalah kumpulan kata yang dapat dicocokkan antara kalimat satu dengan judul kalimat atau dengan kata lain merupakan kata yang muncul dalam kalimat sama dengan kata yang ada dalam judul dokumen [5]. *Fitur* ini dapat dihitung menggunakan rumus (2.3)

$$\text{Score}_{f3}\ (s) = \left| \frac{keyword\ dalam\ s \cap\ keyword\ dalam\ judul}{keyword\ dalam\ s \cup keyword\ dalam\ judul} \right| \quad (2.3)$$

4. Kalimat yang Mengandung Entiti (f4)

Nama entiti adalah sebuah kumpulan kata yang memiliki makna atau membentuk nama sebuah intitusi. Misalnya adalah Real Madrid merupakan kumpulan kata yang memiliki makna sebuah klub sepak bola. Fitur ini dihitung menggunakan rumus (2.4)

$$\text{Score}_{f4}\ (s) = \left| \frac{\#(nama\ entiti\ dalam\ (s))}{length(s)} \right| \quad (2.4)$$

5. Kalimat yang Mengandung Data Numerik (f5)

Pada peringkasan teks mempertimbangkan data numerik karena kalimat yang memiliki angka numerik biasanya penting dan sangat mungkin berada pada ringkasan dokumen [5]. Fitur ini dapat dihitung menggunakan rumus (2.5)

$$\text{Score}_{f5}\ (s) = \left| \frac{\#(data\ angka\ dalam\ kalimat\ (s))}{length(s)} \right| \quad (2.5)$$

6. Panjang Kalimat (f6)

Fitur ini bertujuan untuk menghilangkan kalimat-kalimat yang terlalu pendek. Fitur ini dihitung berdasarkan jumlah kata dalam kalimat dibagi jumlah kata unik dalam paragraf [5]. Fitur ini dihitung menggunakan rumus (2.5)

$$\text{Score}_{f6}\ (s) = \left| \frac{\#(banyak\ kata\ dalam\ kalimats)}{\#banyak\ kata\ dalam\ dokumen} \right| \quad (2.6)$$

7. Koneksi antar Kalimat (f7)

Koneksi antar kalimat adalah banyaknya link dari suatu kalimat yang terhubung dengan kalimat yang lain. Atau dengan kata lain merupakan jumlah kalimat yang memiliki kata yang sama dengan kalimat lain dalam satu dokumen [5]. Fitur ini dapat dihitung menggunakan rumus (2.7)

$$\text{Score}_{f7}\,(s) = \left| \frac{\#jumlah\ koneksi\ antar\ kalimat\ s}{\sum_{i=0}^{n} \#jumlah\ koneksi\ antar\ kalimat\ s} \right| \quad (2.7)$$

8. Penjumlahan Bobot Koneksi antar Kalimat (f8)

Fitur ini berfungsi menjumlahkan kata dalam suatu kalimat dengan kata yang sama dari kalimat lain dalam suatu dokumen [5]. Fitur ini dihitung menggunakan rumus (2.8)

$$\text{Score}_{f8}\,(s) = \left| \frac{\#jumlah\ kata\ sama\ pada\ kalimat\ s}{\sum_{i=0}^{n} \#jumlah\ kata\ sama\ pada\ kalimat\ s} \right| \quad (2.8)$$

2.5. Relevance Vector Machine

Sekumpulan data latih yang yang terdiri dari himpunan vector $\{x_n\}_{n=1}^{N}$ digunakan bersamaan dengan target yang sesuai $\{t_n\}_{n=1}^{N}$, dimana pada dasarnya target merupakan nilai sebenarnya dalam hal regresi dan label kelas pada proses klasifikasi hal itulah yang dilakukan oleh *supervised learning*. Dari pelatihan memungkinkan ketergantungan pada input untuk tujuan prediksi yang akurat [2]. Secara umum prediksi tersebut dapat dihitung dengan menggunakan persamaan y(x), yaitu:

$$\mathbf{y}(\mathbf{x};\mathbf{w}) = \sum_{\mathbf{i=1}}^{\mathbf{M}} \mathbf{w_i}\boldsymbol{\phi}_\mathbf{i}(\mathbf{x}) + \mathbf{w_0} = \mathrm{w}^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}) + \mathbf{w_0} \quad (2.9)$$

dimana $w = (w_1, w_2, \ldots, w_M)$ merupakan vektor bobot (*weight vector*) dan $\phi_i(x)$ merupakan fungsi kernel terhadap data x, dan w_0 merupakan *bias*.

Relevance Vector Machine adalah sebuah metode pembelajaran mesin yang diperkenalkan oleh Mike Tipping pada tahun 2000, yang diadaptasi dari *Bayesian Framework* dan memiliki bentuk model fungsi yang mirip dengan *Support Vector Machine* [2]. Sama halnya dengan *Support Vector Machine*, Relevance Vector Machine digunakan dalam proses klasifikasi dan regresi. *Relevance Vector Machine* diperkenalkan untuk menutupi beberapa kelemahan yang dimiliki oleh *Support Vector Machine*. Kelemahan tersebut antara lain [2]

1. Prediksi yang tidak *probabilistik*. Dalam regresi SVM keluaranya adalah titik dan dalam klasifikasinya mempunyai keputusan biner. Hal itu membuat support vector machine tidak cocok untuk proses klasifikasi dimana peluang posterior dari keanggotaan kelas dianggap penting
2. Walaupun *Support Vector Machine* menghasilkan vektor yang relatif *sparse*, namun jumlah *Support Vector* yang ada akan secara linear bertambah seiring dengan ukuran himpunan data latih.
3. *Support Vector Machine* memerlukan proses estimasi parameter *error/margin* C, yang secara umum memerlukan prosedur *cross-validation* yang dapat memperlambat proses komputasi.
4. Fungsi *kernel* yang digunakan dalam *Support Vector Machine* harus memenuhi kondisi *Mercer*.

Dalam kasus klasifikasi, model prediksi mengambil bentuk kombinasi linear dari fungsi basis/kernel yang diubah oleh fungsi *logistic sigmoid* [2].

$$\mathbf{y}(\mathrm{x},\mathrm{w}) = \boldsymbol{\sigma}(\mathbf{x};\mathbf{w}) = \boldsymbol{\sigma}(\mathrm{w}^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x})) \tag{2.10}$$

Dimana $\sigma(.)$ Merupakan fungsi *logistic sigmoid* yang didefinisikan dengan persamaaan

$$\boldsymbol{\sigma}(-\mathbf{y}) = \frac{\mathbf{1}}{\mathbf{1}+\mathbf{exp}(-\mathbf{y})} \tag{2.11}$$

Berdasarkan definisi dari distribusi *Bernoulli*, *likelihood* terdefinisikan sebagai berikut

$$\mathbf{p}(\mathbf{t}|\mathbf{w}) = \prod_{\mathbf{t=1}}^{\mathbf{N}} \boldsymbol{\sigma}\{\mathbf{y}(\mathbf{x_n};\mathrm{w})\}^{\mathbf{t_n}}[\mathbf{1}-\boldsymbol{\sigma}\{\mathbf{y}(\mathbf{x_n};\mathrm{w})\}^{\mathbf{1-t_n}} \tag{2.12}$$

untuk target $t_n \in \{0,1\}$.

Persamaan *likelihood* dilengkapi dengan sebuah *prior* terhadap parameter (bobot) dalam bentuk

$$\mathbf{p}(\mathrm{w}|\alpha) = \prod_{\mathbf{t=1}}^{\mathbf{N}} \frac{\sqrt{\boldsymbol{\alpha}_\mathbf{i}}}{\sqrt{\mathbf{2\pi}}} \mathbf{exp}\left(-\frac{\boldsymbol{\alpha}_\mathbf{i}\mathbf{w}_\mathbf{i}^{\mathbf{2}}}{\mathbf{2}}\right) \tag{2.13}$$

dimana $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ merupakan *hyperparameter* yang diperkenalkan untuk mengontrol kekuatan dari *prior* terhadap parameter bobot yang diasosiasikannya, N merupakan banyak kalimat , dan w merupakan parameter bobot.

Untuk nilai α tertentu, distribusi *posterior* bobot terhadap data dapat dihitung menggunakan aturan *Bayes*, dengan persamaan

$$\mathbf{p}(\mathrm{w}|\mathrm{t}, \alpha) = \frac{\mathbf{p}(\mathrm{t}|\mathrm{w})\mathbf{p}(\mathrm{w}|\alpha)}{\mathbf{p}(\mathrm{t}|\alpha)} \tag{2.14}$$

dimana p(**t**|**w**) adalah *likelihood*, p(**w**|**α**) adalah *prior*, dan p(**t**|**α**) adalah *evidence*.

Parameter bobot model tidak dapat diperoleh dengan cara analitik, sehingga aproksimasi *Laplacian* digunakan. Sejak p(**w**|**t**,**α**) secara linear proporsional untuk p(**t**|**w**) x p(**w**|**α**), dapat dimungkinkan untuk mencari maksimum dari persamaan

$$\begin{aligned} \ln \mathrm{p}(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) &= \ln \{\mathrm{p}(\mathrm{t}|\mathrm{w})\mathrm{p}(\mathrm{w}|\alpha)\} + \ln \mathrm{p}(\mathbf{t}|\alpha) \\ &= \sum_{\mathrm{n}=1}^{\mathrm{N}} \{\mathrm{t_n} \ln \mathrm{y_n} + (1 - \mathrm{t_n}) \ln(1 - \mathrm{y_n})\} + \frac{1}{2}\mathrm{w^T A w} \end{aligned} \tag{2.15}$$

untuk parameter bobot yang paling mungkin **w**', dengan $y_n = \sigma\{y(x_n; \boldsymbol{w})\}^{t_n}$ dan **A** = diag($\alpha_1, \alpha_2, \ldots, \alpha_n$) yang terdiri dari nilai α yang diinisialisasi. Untuk mencari parameter **w'** pada proses reestimasi, metode *Iteratively Reweight Least-Square* dapat digunakan.

Fungsi *logistic log-likelihood* dapat diturunkan sebanyak dua kali untuk memperoleh matriks *Hessian* dengan persamaan

$$\mathrm{H} = \boldsymbol{\nabla\nabla} \ln \mathbf{p}(\mathrm{w}|\mathrm{t}, \alpha) = \varphi_\mathbf{n}^\mathbf{T} \mathrm{B} \varphi_\mathbf{n} + \mathrm{A} \tag{2.16}$$

dimana **φ** merupakan matriks kernel, dan **B** = diag(β_1, β_2, …, β_n) merupakan matriks diagonal yang elemen matriksnya diperoleh dengan persamaan

$$\boldsymbol{\beta_n} = \boldsymbol{\sigma}\{\boldsymbol{y}(\boldsymbol{x_n}; w')\}[\mathbf{1} - \boldsymbol{\sigma}\{\boldsymbol{y}(\boldsymbol{x_n}; w')\} \quad (2.17)$$

dimana **w'** merupakan parameter bobot yang akan direestimasi. Proses reestimasi parameter bobot **w** didapat dengan persamaan

$$\mathrm{w}' = \mathrm{w}_{\mathbf{n}} + \lambda\Delta \quad (2.18)$$

dimana λ merupakan koefisien estimasi, dan Δ merupakan vektor yang diperoleh dari persamaan

$$\Delta = \mathrm{H}^{-1}.\mathrm{g} \quad (2.19)$$

dimana $\mathbf{H^{-1}}$ adalah invers dari matriks hessian dan **g** adalah gradien negatif yang diperoleh dari penurunan fungsi *logistic log-likelihood* sebanyak satu kali dengan persamaan

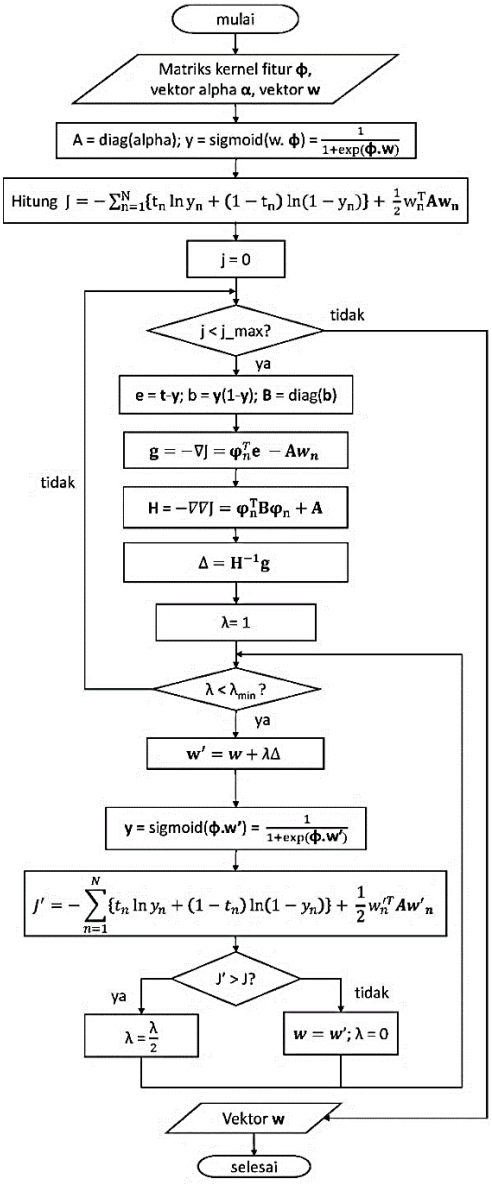
$$\mathrm{g} = -\boldsymbol{\nabla} \mathbf{ln}\, \boldsymbol{p}\,(w|t, \alpha) = \boldsymbol{\varphi}_{\mathbf{n}}^{\mathbf{T}}\mathbf{e} - \mathrm{A}\mathbf{w}_{\mathbf{n}} \quad (2.20)$$

Penjelasan proses estimasi parameter bobot **w** secara rinci dalam bentuk diagram *flowchart* dapat dilihat pada gambar 2.3. Setelah estimasi parameter bobot **w** dilakukan, selanjutnya *hyperparameter* α_j dilakukan reestimasi dengan menggunakan persamaan

$$\boldsymbol{\alpha}_{\mathbf{j}}^{\boldsymbol{new}} = \frac{\boldsymbol{\gamma}_{\mathbf{j}}}{\mathbf{w}_{\mathbf{j}}^{\mathbf{2}}} \quad (2.21)$$

dimana γ_j didefinisikan dengan persamaan

$$\boldsymbol{\gamma_j} = \mathbf{1} - \boldsymbol{\alpha}_{\mathbf{j}}\boldsymbol{\Sigma}_{\mathbf{jj}} \quad (2.22)$$



Gambar 2.3 *Flowchart* Estimasi Parameter Bobot w

dimana Σ_{jj} merupakan elemen diagonal matriks kovarians yang didapat dari persamaan

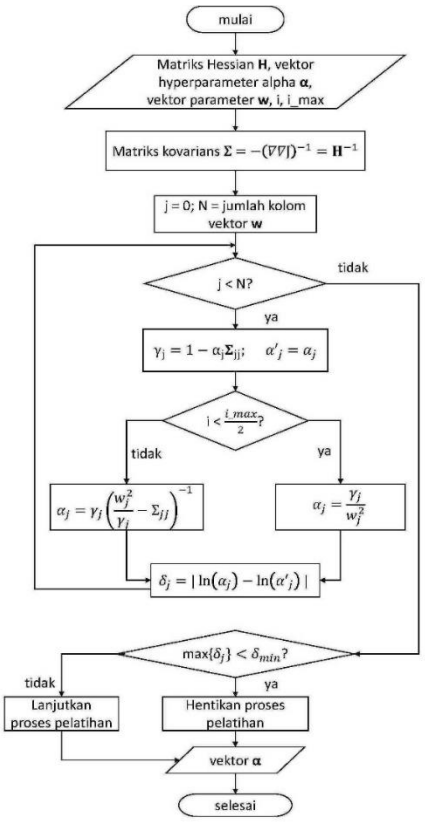
$$\Sigma = -\left(\nabla\nabla \ln \boldsymbol{p}\,(w|t,\alpha)\right)^{-1} = \mathrm{H}^{-1} \tag{2.23}$$

dimana $\mathbf{H}^{-1}$ merupakan invers dari matriks *Hessian*. Selain dengan cara menggunakan persamaan (2.15), reestimasi *hyperparameter* α_j juga dapat dilakukan dengan cara "*hybrid update*" [8] dengan menggunakan persamaan

$$\alpha_j = \frac{\gamma_j}{w_j^2}; \text{jika } i < \frac{i_{max}}{2}$$
$$\alpha_j = \gamma_j \left(\frac{w_j^2}{\gamma_j} - \Sigma_{jj} \right)^{-1}; \text{jika } i \geq \frac{i_{max}}{2} \quad (2.24)$$

dimana i_{max} merupakan jumlah iterasi maksimum pada pelatihan.

Penjelasan proses estimasi parameter *hyperparameter alpha* secara rinci dalam bentuk *flowchart* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Flowchart Estimasi Hyperparameter Alpha

Proses optimasi terus berlanjut hingga perubahan maksimum dari α_j dibawah koefisien tertentu atau jumlah maksimum iterasi telah dicapai.

2.6. Evaluasi Ringkasan

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 kegiatan yang subjektif oleh karena itu evaluasi sebuah ringkasan merupakan hal yang sulit.

Sebuah ringkasan menurut sebagian orang sudah tepat, tetapi menurut sebagian orang lagi tidak demikian [8].

Secara umum terdapat dua jenis metode evaluasi ringkasan, yaitu [8]:

a. Ekstrinsik yaitu kualitas ringkasan diukur berdasarkan bagaimana ringkasan tersebut dapat membantu penyelesaian tugas orang yang membaca ringkasan tersebut.
b. Intrinstik yaitu hanya diukur dari kualitas hasil keluaran (*output*) ringkasan yang dihasilkan.

Evaluasi sistem peringkas yang akan digunakan dalam penelitian ini adalah evaluasi intrinsik. Pengevaluasi menciptakan sebuah ringkasan yang 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 dieksplorasi. Kombinasi antara nilai *recall* dan *precision* menghasilkan *f-measure* [7].

2.6.1. *Recall*

Recall (r) merupakan istilah yang digunakan untuk kalimat terpanggil yang relevan dengan pernyataan atau vektor *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 [9]. Hal ini berarti *recall* adalah bagian dari proses evaluasi yang dapat digunakan sebagai alat ukur relevannya sebuah

ringkasan. Adapun rumus untuk menghitung *recall* dapat dilihat pada persamaan berikut.

$$\boldsymbol{recall} = \frac{tp}{tp + fn} \tag{2.25}$$

Dimana

Recall : Tingkat keberhasilan

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

fn : jumlah kalimat yang di ekstrak manusia tetapi tidak terddaan dalam kalimat yang diekstrak sistem.

2.6.2. *Precision*

Precision (*p*) biasanya menjadi salah satu ukuran yang digunakan untuk menilai efektivitas kalimat yang di hasilkan 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.

$$\boldsymbol{precision} = \frac{tp}{tp + fp} \tag{2.26}$$

Dimana :

Precission : 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.

2.6.3 *F-Measure*

F-Measure (F_1) merupakan salah satu perhitungan evaluasi yang mengkombinasikan *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* [9]. Perhitungan *f-measure* dapat dirumuskan pada persamaan berikut.

$$\boldsymbol{F - measure = \frac{2 * precision * recall}{precision + recall}} \quad (2.27)$$

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

2.6.3. Akurasi

Akurasi merupakan salah satu perhitungan evaluasi yang mengkombinasikan antara kalimat yang terpilih oleh manusia ,kalimat yang tidak terpilih oleh manusia dan kalimat hasil pengambilan sistem.

Perhitungan akurasi adalah sebagai berikut :

$$\boldsymbol{akurasi = \frac{tp+tn}{tp+fp+fn+tn}} \quad (2.28)$$

Dimana :

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 terddaan dalam kalimat yang diekstrak sistem.

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

2.7. Cosine Similarity

Cosine similarity digunakan untuk menghitung pendekatan relevansi query terhadap dokumen. Semakin besar nilai kesamaan vektor query dengan vektor dokumen maka query tersebut dipandang semakin relevan dengan dokumen. Cosinus dari 0^0 adalah 1, dan kurang dari 1 untuk setiap sudut lainnya. Dengan demikian dua vektor dengan orientasi yang sama memiliki kesamaan cosinus dari 1, dua vektor pada 90^0 memiliki kesamaan 0. *Cosine similarity* terutama digunakan dalam ruang positif, dimana hasilnya dibatasi (0,1). Cosine similarity kemudian memberi tolak ukur seberapa mirip dua dokumen .

Cosinus dari dua vektor dapat diturunkan dengan menggunakan *dot product Euclidean* yaitu 1.b = ||a|| ||b|| cos Θ dimana vektor atribut, A dan B kesamaan dan besarnya ditukan dengan persamaan (2.29).

$$CS(b1,b2) = \frac{\sum_{t=1}^{n} W_{t.b1} W_{t.b2}}{\sqrt{\sum_{t=1}^{n} W_{t.b1}^2} \sqrt{\sum_{t=1}^{n} W_{t.b2}^2}} \qquad (2.29)$$

Dimana :

t : term dalam Kalimat

$W_{t.b1}$: bobot term t dalam blok b1

$W_{t.b1}$: bobot term t dalam blok b2

2.7.1. Metode TF-IDF (Term Frequency – Inversed Document Frequency)

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 *term j* pada dokumen *i*. *Document frequency* (*df*) adalah banyaknya kalimat dimana suatu *term j* muncul [7].

Dalam *tf* frekuensi *term* pilihan paling sederhana adalah dengan menggunakan frequensi baku dalam dokumen, yaitu berapa kali *term* (t) terjadi dalam dokumen (d) . Nilai idf sebuah term (kata) dapat dihitung menggunakan dalam persamaan (2.30).

$$IDF = \log(\frac{N}{idf}) \qquad (2.30)$$

N adalah jumlah dokumen yang berisi *term* (*t*) dan *idf* adalah jumlah kemunculan (frekuensi) *term* terhadap N. Adapun algoritma yang digunakan untuk menghitung bobot (w) masing-masing dokumen terhadap kata kunci (*query*) menggunakan persamaan (2.31).

$$W_{d.t} = tf_{d.t} * IDF_t \qquad (2.31)$$

Dimana :

D : dokumen ke-*d*

t : *term* ke-t dari kata kunci

tf : term frekuensi/ frekuensi kata

w : bobot dokumen ke-d terhadap term ke-t

2.8. DFD (Data Flow Diagram)

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil [10] . Diagram aliran data dipakai untuk menunjukkan bagaimana data mengalir melalui serangkaian langkah pemrosesan. Data ditransformasikan pada setiap langkah sebelum ke tahap berikutnya. Langkah-langkah pemrosesan atau transformasi ini merupakan fungsi program ketika diagram aliran data digunakan untuk mendokumentasikan desain perangkat lunak. Namun demikian, pada model analisis, pemrosesan dapat dilakukan oleh orang atau komputer [6].

Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai atau user yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan [10].

Beberapa komponen yang digunakan di DFD adalah:

Simbol bisa dilihat pada daftar simbol Tabel 1. Simbol DFD.

a. Entitas Luar (Boundary) Entitas luar merupakan kesatuan (entity) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem [11].
b. Arus Data (Data Flow) Arus data diberi simbol suatu panah. Arus data mengalir diantara proses (process), simpanan data (data store) dan kesatuan luar (external entity) [11].
c. Proses (Process) Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk kek dalam proses untuk menghasilkan arus data yang akan keluar dari proses [11].
d. Penyimpanan Data (Data Store) Simpanan data merupakan simpanan dari data yang dapat berupa file atau database di sistem komputer, arsip atau catatan manual, agenda atau buku.

   Simpanan data DFD dapat di simbolkan dengan sepasang garis horizontal [11].

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. Untuk memudahkan analisa dimulai dengan [10]:

a. Diagram Konteks (*Context Diagram*) Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau output dari sistem [10].
b. Diagram Rinci (*Level Diagram*) Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram zero atau diagram level atasnya [10].

2.9.Pengujian Black Box

Pengujian *black box* disebut juga dengan pengujian perilaku, merupakan metode pengujian perangkat lunak yang berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black box* berupaya untuk menemukan kesalahan dalam kategori seperti berikut [12]:

1. Fungsi yang salah atau hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam struktur data atau akses basis data eksternal.
4. Kesalahan perilaku atau kinerja.
5. Kesalahan inisialisasi dan penghentian.