PERINGKASAN TEKS OTOMATIS PADA MULTI DOKUMEN MENGGUNAKAN *RELEVANCE VECTOR MACHINE*

Tendi Arifin[1,] Kania Evita Dewi[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
Email : tendiarifin0308@gmail.com[1] , kania.evita.dewi@email.unikom.ac.id[2]

ABSTRAK

Peringkasan teks otomatis pada multi dokumen adalah peringkasan banyak dokumen dengan tema yang sama menggunakan mesin. Penelitian sebelumnya, peringkasan teks otomatis pada multi dokumen dilakukan menggunakan metode *Support Vector Machine* dengan *recall* 0.773, *precission* 0.771, *f-measure* 0.771. Penelitian Christopher M. Bishop, Michael E. Tipping didapatkan tingkat error SVM lebih tinggi dari RVM yaitu SVM 10,6% dan RVM 9,3%.

Tahapan dari peringkasan teks multi dokumen diantaranya penerimaan input teks, *preprocessing*, pembobotan *ekstraksi fitur* dengan fitur yang dipakai adalah posisi kalimat, kemiripan antar kalimat, kalimat yang menyerupai judul dokumen, kalimat yang mengandung kata entiti, kalimat yang mengandung data numerik, panjang kalimat, koneksi antar kalimat dan penjumlahan bobot koneksi antar kalimat kemudian pengolahan hasil ekstraksi fitur dengan RVM, pembobotan hasil RVM dengan tf-idf dan disusun menggunakan metode *cosine similarity*.

Berdasarkan hasil pengujian disimpulkan bahwa Metode RVM dapat diterapkan pada Peringkasan Teks Otomatis Pada Multi Dokumen dengan performasi yang cukup kecil yaitu *recall* sebesar 47.83%, *precission* sebesar 35.48%, *f-measure* sebesar 40.74%, dan akurasi sebesar 67.35%.

Kata kunci : Peringkasan Teks, peringkasan teks Otomatis pada muti dokumen, *Relevance Vector Machine*, Metode klasifikasi, *machine learning* RVM.

1. PENDAHULUAN

Banyaknya situs yang memuat berita menimbulkan pembahasan berita dengan topik yang sama selalu berulang walaupun dengan judul yang berbeda. Oleh karena itu, dibutuhkan suatu peringkasan berita untuk mempermudah pembaca dalam mencari informasi yang lengkap tentang suatu topik berita tanpa harus membaca semua berita yang ada. Salah satu cara peringkasan adalah peringkasan manual yaitu peringkasan yang dilakukan secara manual oleh individu, tetapi dengan cara peringkasan manual akan memakan banyak waktu dan sumberdaya. Supaya mempermudah dalam peringkasan dibutuhkan suatu alat/sistem yang bisa otomatis meringkas isi dari beberapa dokumen [1].

Dari penelitian Agus Riyanto yaitu peringkasan teks menggunakan metode K-means mendapatkan rata-rata nilai precision adalah 45.5%, recall 57.74%, F-measures 46,13% dan pada penelitian ini disarankan untuk menggunakan algoritma lain untuk meningkatkan performasinya [2]. Dan pada penelitian multi dokumen telah di lakukan oleh Deni Fitriaman, Masayu Leylia Khodra, bambang Rianto Trilaksono. Dalam penelitian ini dilakukan peringkasan teks multi dokumen dengan menggunakan metode SVM (*Support Vektor Machine*). Adapun metode SVM pada penelitian tersebut digunakan untuk mengklasifikasikan kalimat yang masuk pada peringkasan. Selanjutnya kalimat hasil peringkasan disusun menggunakan metode MMR (*Maximum Marginal Relevance*). Dari penelitian ini didapatkan nilai rata-rata *recall* 0.773, *precission* 0.771, *f-measure* 0.771 [3].

Pengembangan metode SVM (*Support Vector Machine)* yaitu metode RVM (*Relevance Vector Machine*). Pada penelitian yang dilakukan oleh Christopher M. Bishop, Michael E. Tipping didapatkan kesimpulan tingkat error dari SVM lebih tinggi dari RVM yaitu SVM 10,6% dan RVM 9,3% [4].

Penelitian tentang pembobotan ekstraksi fitur telah dilakukan oleh Zulkifi, Agung Toto Wibowo, Gia Septiana. Pada penelitian tersebut didapatkan bahwa ekstraksi fitur yang baik adalah posisi kalimat, kemiripan antar kalimat, kalimat yang menyerupai judul dokumen, kalimat yang mengandung kata entiti, kalimat yang mengandung data numerik, panjang kalimat, koneksi antar kalimat dan penjumlahan bobot koneksi antar kalimat dengan akurasi 53% [5].

Berdasarkan uraian pada latar belakang masalah diatas, rumusan masalah dalam penelitian ini adalah apakah RVM dapat digunakan untuk peringkasan multi dokumen. Sehingga penelitian ini bermaksud untuk membuat sistem peringkasan teks otomatis pada multi dokumen yang bertujuan menguji /mengukur seberapa baik ringkasan yang dihasilkan dengan menggunakan metode RVM pada peringkasan teks multi dokumen dengan mengetahui nilai performasi (*recall, precision dan f-measure)* dan akurasi peringkasan pada multi dokumen dengan menggunakan metode RVM.

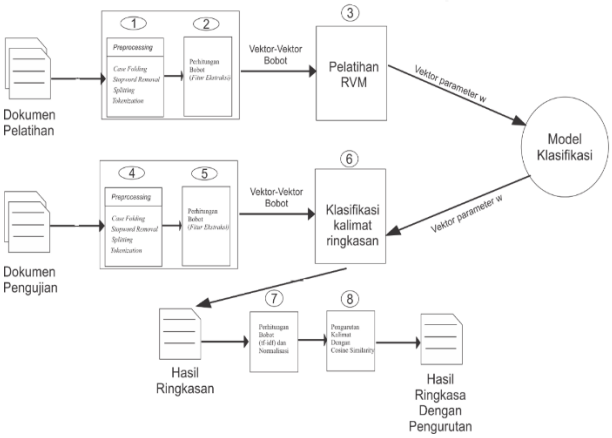
2. ISI PENELITIAN

Bagian ini terbagi menjadi beberapa bagian, yaitu penjelasan arsitektur sistem, *preprocessing*, pembobotan ekstraksi fitur, pelatihan, klasifikasi, *cosine similarity* dan pengujian ringkasan. Pada tahap ini akan di jelaskan tentang arsitektur sistem, *preprocessing*, pembobotnan ekstraksi fitur, pelatihan RVM, Klasifikasi RVM, *Cosine Similarity* dan pengjian hasil ringkasan.

2.1 Arsitektur Sistem

Arsitektur sistem menggambarkan komponen-komponen pendukung dari sistem yang akan dibangun. Adapun arsitektur sistem peringkasan teks yang akan dibangun dapat dilihat pada Gambar 1.

dsa



Gambar 1. Arsitektur Sistem

Sistem peringkas teks otomatis yang dibangun memiliki 5 proses utama yaitu preprocessing, perhitungan bobot (ekstraksi fitur), proses pelatihan untuk menghasilkan model klasifikasi, proses pengujian berupa klasifikasi kalimat, serta penyusunan hasil ringkasan.

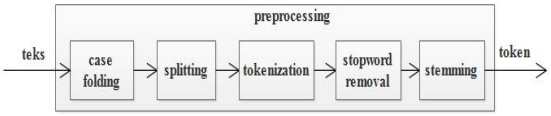
Untuk penjelasan setiap tahap pada proses peringkasan berdasarkan pada Gambar 1 dapat dilihat sebagai berikut:

1. Data latih dilakukan *preprocessing* yang merupakan tahap dimana data latih akan dibersihkan dari *noise* atau dari hal yang dianggap tidak memiliki pengaruh dalam ringkasan. *Preprocessing* yang dilakukan dalam tahap ini adalah *case folding*, *splitting*, *tokenization*, dan *stopword removal*.
2. Tahap berikutnya adalah melakukan perhitungan bobot dengan menggunakan metode ekstraksi fitur dimana fitur yang di pakai adalah posisi kalimat, kemiripan antar kalimat, kalimat yang menyerupai judul dokumen, kalimat yang mengandung kata entiti, kalimat yang mengandung data numerik, panjang kalimat, koneksi antar kalimat dan penjumlahan bobot koneksi antar kalimat.
3. Tahap berikutnya adalah melakukan proses pelatihan *Relevance Vector Machine* terhadap vektor-vektor yang didapatkan dari proses sebelumnya untuk mendapatkan model klasifikasi.
4. Tahap berikutnya adalah sama seperti tahap pertama, yaitu melakukan *preprocessing*, tetapi dilakukan untuk data uji. *Preprocessing* yang dilakukan dalam tahap ini pun sama yaitu *case folding*, *splitting*, *tokenization*, dan *stopword removal*.
5. Berikutnya adalah melakukan pembobotan dengan ekstraksi fitur pada dokumen uji dimana prosesnya sama dengan tahapan no 2.
6. Selanjutnya melakukan proses pengujian dengan menggunakan model klasifikasi yang telah didapatkan pada proses pelatihan.
7. Selanjutnya hasil dari ringkasan dilakukan perhitungan bobot setiap kata (*term*) dalam data hasil ringkasan dengan menggunakan metode *tf-idf* dengan menghitung *TF* (*Term Frequency*), *DF* (*Document Frequency*), dan dihitung bobot dari masing-masing *term* pada data hasil ringkasan. Setelah didapat nilai dari *tf-idf* masing-masing kalimat, langkah selanjutnya adalah melakukan normalisasi *tf-idf*.

Untuk penjelasan setiap proses-proses utama tersebut dijelaskan pada beberapa sub-bab berikutnya.

2.2 *Preprocessing*

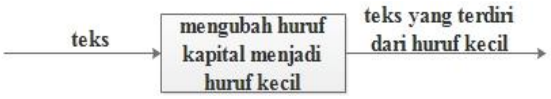
Preprocessing ini merupakan proses-proses yang dilakukan untuk mendapatkan fitur kata-kata yang bersih sehingga tidak akan mengandung *noise* yang akan menganggu hasil untuk proses selanjutnya. Tahapan-tahapan dari *preprocessing* dapat dilihat pada Gambar 2.



Gambar 2. Tahapan *Preprocessing*

1. *Case Folding*

   Case folding adalah proses yang mengubah semua huruf dalam teks dokumen menjadi huruf kecil. Proses dari *case folding* dapat dilihat pada Gambar 3. Sebagai contoh, untuk kata “Dalam” setelah melewati proses *case folding* berubah menjadi kata “dalam”, dimana huruf kapital ‘D’ diubah menjadi huruf ‘d’.



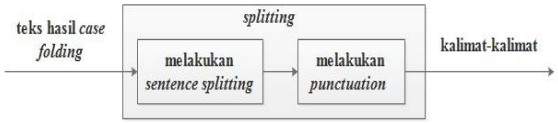
Gambar 3. Proses *Case Folding*

Pada proses *case folding* dilakukan proses penyatuan etitas dimana hasil dari proses penyatuan entitas akan digunakan pada tahapan selanjutya.

2. *Splitting*

   Splitting merupakan pemecahan sebuah teks utuh menjadi pecahan-pecahan kalimat.

Gambaran dari proses *splitting* dapat dilihat pada Gambar 4. Proses *splitting* melibatkan dua sub-proses, yaitu *split sentences* dan *punctuation*.



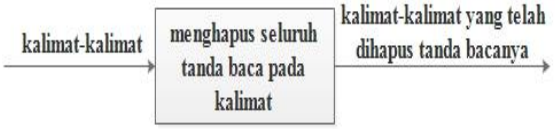
Gambar 4. Tahapan *Splitting*

Sentence splitting (pemecahan kalimat) adalah proses memecah *string* teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Pemecahan kalimat dalam dokumen menjadi kalimat-kalimat dapat dilakukan dengan cara mendeteksi keberadaan tanda titik (.), tanda tanya (?) dan tanda seru (!) sebagai *delimiter* untuk memotong *string* dokumen [5]. Langkah-langkah dari *sentence splitting* dapat dilihat pada Gambar 5.



Gambar 5. Proses *Sentence Splitting*

Punctuation merupakan proses penghilangan tanda baca ataupun simbol pada kalimat yang telah dipecah [4]. Proses dari *punctuation* dapat dilihat pada Gambar 6.



Gambar 6. Proses *Punctuation*

3. *Tokenization*

Tokenization adalah proses pemotongan *string* menjadi kumpulan *token-token*. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melakukan pemindaian kalimat dengan pemisah (*delimiter*) serta *white space* (spasi, *tab*, dan *newline*). Langkah-langkah dari tokenization dapat dilihat pada Gambar 7.

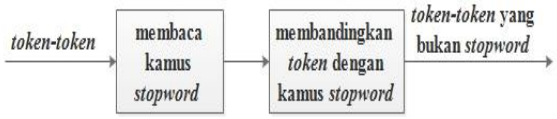


Gambar 7 tahapan *tokenization*

Sebagai contoh jika terdapat kalimat "dalam urusan pengembangan teknologi", setelah melewati tahapan *tokenization*, maka *token-token* yang terbentuk adalah "dalam", "urusan", "pengembangan" dan "teknologi".

4. *Stopword Removal*

Stopword removal merupakan proses penghilangan *stopword*. *Stopword* dapat berupa kata depan, kata penguhubung, dan kata pengganti. Pengecekan *stopword* dilakukan dengan cara memeriksa apakah kata yang bersangkutan terdaftar dalam kamus stopword. Jika terdaftar dalam kamus maka kata tersebut akan dihilangkan. Kamus *stopword* diambil dari rekomendasi Fadillah Z. Tala dalam penelitiannya yang berjudul "*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*". Proses dari *stopword removal* dapat dilihat pada Gambar 8.



Gambar 8. Proses *Stopword Removal*

Sebagai contoh pada *token-token* "dalam", "urusan", "pengembangan" dan "teknologi" setelah melewati tahapan *stopword removal*, *token* yang tersisa menjadi "urusan", "pengembangan", dan "teknologi". Hal tersebut dikarenakan *token* "dalam" terdapat pada kamus *stopword* sehingga *token* tersebut dihilangkan.

2.3 Pembobotann 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 :

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

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 (2.6)

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

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)

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

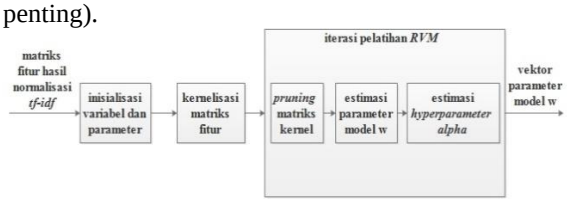
8. Penjumlahan Bobot Koneksi antar Kalimat (f8)

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)

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

2.3 Pelatihan *Relevance Vector Machine*

Pelatihan dari metode *Relevance Vector Machine* ini dilakukan untuk mendapatkan parameter dari persamaan model klasifikasi yang nantinya akan digunakan untuk proses pengujian klasifikasi. Adapun langkah-langkah pelatihan dalam *Relevance Vector Machine* ini dapat dilihat pada Gambar 9. Dalam proses pelatihan, data yang akan digunakan sebagai data latih terlebih dahulu harus diberi label yang menunjukan kelas dari elemen-elemen data tersebut. Dalam kasus ini, kalimat-kalimat dalam data latih akan diberi label kelas berupa angka 0 ataupun 1. Angka 0 merepresentasikan bahwa kalimat tersebut masuk ke dalam kelas negatif (kalimat tidak penting), sedangkan angka 1 merepresentasikan bahwa kalimat tersebut masuk ke dalam kelas positif (kalimat tidak penting).



Gambar 9. Proses Pelatihan *Relevance Vector Machine*

Berdasarkan Gambar 9, terdapat lima proses utama dalam pelatihan, yaitu inisialisasi parameter dan variabel, kernelisasi matriks fitur, *pruning* matriks kernel, estimasi parameter model **w**, dan estimasi *hyperparameter* model. Langkah-langkah detail pada kelima proses tersebut adalah sebagai berikut.

1. Inisialisasi Parameter
   Inisialisasi parameter dan variabel merupakan proses pemberian nilai awal pada parameter-parameter yang diperlukan untuk proses pelatihan. Parameter yang akan diinisialisasi adalah parameter model vektor **w**, vektor hyperpamater α, dan jumlah iterasi maksimum yang dilakukan. Pada penelitian ini, vektor model **w** akan diinisialisasi elemen-elemennya dengan nilai 0, vektor *hyperparameter* **α** diinisialisasi dengan nilai sembarang, iterasi maksimum, terdiri dari iterasi maksimum dan iterasi maksimum pada proses estimasi parameter model w dibatasi masing-masing sebanyak 250 dan 25 iterasi.
2. Kernelisasi Matriks Fitur
   Kernelisasi matriks fitur digunakan untuk merubah matriks data latih hasil ekstraksi fitur menjadi matriks kernel. Adapun fungsi kernel yang digunakan dalam penelitian ini adalah fungsi kernel *gauss*, dengan persamaan:

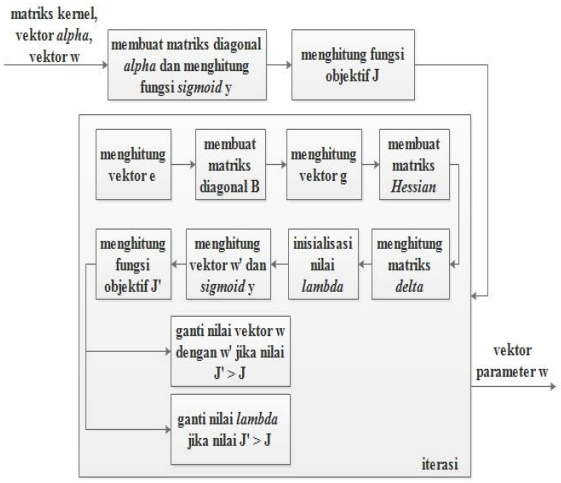
$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \quad (2.9)$$

   Dimana x dan x' merupakan data latih dan σ merupakan parameter bebas.

3. *Pruning* Matriks Kernel
   Pruning merupakan proses pemotongan elemen dari matriks kernel yang dianggap tidak relevan. Elemen matriks kernel yang tidak relevan ini nantinya tidak akan diproses pada iterasi-iterasi berikutnya. Indikator sebuah elemen matriks kernel relevan atau tidak adalah nilai *hyperparameter* α yang dihasilkan oleh elemen matriks tersebut. Apabila nilai *hyperparameter* α dirasa sudah sangat besar hingga mencapai nilai tak hingga, maka matriks yang berkoresponden dengan nilai *hyperparameter* tersebut akan dihilangkan. Untuk menentukan bahwa nilai *hyperparameter* tersebut termasuk pada kategori relevan atau tidak dibutuhkan sebuah nilai pembanding *alpha*, dimana pada penelitian ini nilai pembanding ditentukan bernilai 10^{12}.
4. Estimasi Parameter Model **w**
   Estimasi parameter model w dilakukan dengan cara melakukan maksimasi fungsi objektif pada persamaan:

$$\ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) = \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} \quad (2.10)$$

dimana N merupakan jumlah data latih, **w** merupakan vektor bobot, t merupakan vektor label kelas dari data latih, dan **A** merupakan matriks diagonal yang terdiri dari elemen vektor α. Selain maksimasi fungsi objektif ln p(**w**|**t**, **α**), estimasi parameter **w** juga dilakukan dengan cara melakukan minimasi nilai fungsi -ln p(**w**|**t**, **α**). Secara rinci langkah-langkah estimasi parameter model w dijabarkan dalam blok diagram pada Gambar 10.



Gambar 10. Proses Estimasi Parameter w

Langkah pertama adalah membuat matriks diagonal **A** yang elemen diagonalnya merupakan nilai dari vektor *hyperparameter* **α** yang telah diinisialisasi sebelumnya pada langkah pertama. Langkah berikutnya adalah mencari variabel vektor **y**, dengan menggunakan persamaan (2.11), dimana x merupakan data latih, w merupakan vektor parameter model yang sebelumnya telah diinisialisasi, **φ**(**x**) merupakan fungsi kernel.

$$y(\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-(\mathbf{w}.\varphi(\mathbf{x})))} \quad (2.11)$$

Setelah mendapatkan parameter-parameter yang dibutuhkan, maka persamaan (2.10) dapat dilakukan proses maksimasi. Adapun proses estimasi menggunakan metode *Iterative Reweighted Least Square* (*IRLS*), dengan terlebih dahulu mencari invers dari hasil persamaan (2.10). Setelah mendapatkan nilai invers dari persamaan (2.10), maka selanjutnya adalah melakukan proses *Iterative Reweighted Least Square*, dengan langkah-langkah sebagai berikut.

a. Pertama-tama carilah variabel vektor **e**, dengan persamaan:

$$\mathbf{e} = \mathbf{t} - \mathbf{y} \quad (2.12)$$

   Dimana **t** merupakan vektor kelas label dan **y** merupakan vektor **y**.
b. Langkah berikutnya adalah mencari vektor **β**, yang setiap nilai elemen vektornya dihitung dengan persamaan:

$$\beta_j = y_j\,(1\text{-}y_j) \quad (2.13)$$

   Dimana y merupakan elemen vektor **y**, dan j merupakan baris vektor.
c. Setelah mendapatkan vektor **β**, maka selanjutnya adalah membuat matriks diagonal **B**, elemen diagonalnya terdiri dari elemen vektor **β**.
d. Langkah berikutnya adalah mencari gradien negatif dari variabel J dengan menggunakan persamaan:

$$\mathbf{g} = \varphi_n^T e - \mathbf{A}w_n \quad (2.14)$$

   dimana **φ** merupakan matriks kernel, serta **B** dan **A** merupakan matriks diagonal dari elemen vektor **α** dan **β**.
e. Tahap berikutnya adalah membuat matriks *Hessian* **H** dari turunan kedua fungsi objektif persamaan (2.10), dengan menggunakan persamaan:

$$\mathbf{H} = \boldsymbol{\varphi}_n^T\mathbf{B}\boldsymbol{\varphi}_n + \mathbf{A} \quad (2.15)$$

   dimana φ merupakan matriks kernel, serta **B** dan **A** merupakan matriks diagonal dari elemen vektor **α** dan **β**.

f. Langkah berikutnya adalah menghitung vektor Δ dengan menggunakan persamaan:

$$\Delta = \mathbf{H^{-1}}.\mathbf{g} \qquad (2.16)$$

dimana $\mathbf{H^{-1}}$ merupakan invers dari matriks *Hessian*, dan **g** merupakan vektor gradient negatif dari fungsi objektif persamaan (2.10).

g. Selanjutnya adalah melakukan inisialisasi nilai λ yang akan digunakan untuk proses iterasi *update* vektor **y**, nilai fungsi objektif persamaan (2.10) dan vektor bobot **w**. Inisialiasi nilai λ dengan nilai 1. Adapun proses iterasi *update* vektor **y**, nilai fungsi objektif persamaan (2.10) dan vektor bobot **w** adalah sebagai berikut.

1) Hitung vektor **w'** dengan menggunakan persamaan:

$$\mathbf{w}' = \mathbf{w}_n + \lambda\Delta \qquad (2.17)$$

dimana $\mathbf{w_n}$ merupakan vektor parameter **w** yang sebelumnya diinisialisasi dengan 0 dan Δ merupakan vektor Δ.

2) Selanjutnya *update* vektor **y** dengan menggunakan persamaan:

$$\mathbf{y} = \frac{1}{1+\exp(-(\mathbf{w}'.\varphi(\mathbf{x})))} \qquad (2.18)$$

Dengan **w'** merupakan vektor bobot dan **φ(x)** merupakan fungsi kernel.

3) Selanjutnya adalah menghitung variabel J' dengan menggunakan persamaan berikut:

$$J' = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1 - y_n)\} + \frac{1}{2}\mathbf{w}'^{T}\mathbf{A}\mathbf{w}' \qquad (2.19)$$

Dimana N merupakan jumlah data latih, **w'** merupakan vektor bobot, **t** merupakan vektor label kelas dari data latih, dan **A** merupakan matriks diagonal yang terdiri dari elemen vektor **α**.

4) Selanjutnya bandingkan apakah hasil perhitungan J' dengan nilai fungsi objektif persamaan (2.10) sebelumnya. Apabila nilai J' lebih dari nilai sebelumnya, maka nilai tersebut dianggap minimum dan ganti vektor parameter **w** dengan nilai yang baru serta beri nilai *lambda* dengan 0 agar iterasi *update* variabel **y**, J' dan vektor bobot **w** berhenti. Sedangkan apabila nilai J' kurang dari nilai sebelumnya, maka nilai *lambda* dibagi dengan nilai 2, dan iterasi iterasi *update* variabel **y**, J' dan vektor bobot **w** dilanjutkan. Dari hasil perhitungan sebelumnya didapat bahwa nilai J' yang baru belum lebih besar dari nilai yang sebelumnya, oleh karena itu proses iterasi terus dilakukan.

h. Setelah langkah iterasi a hingga g dilakukan dan telah mencapai batas iterasi maksimum, maka vektor **w** yang baru telah didapatkan, dan *proses iterative reweighted least square* telah selesai.

5. Estimasi *Hyperparameter Alpha*

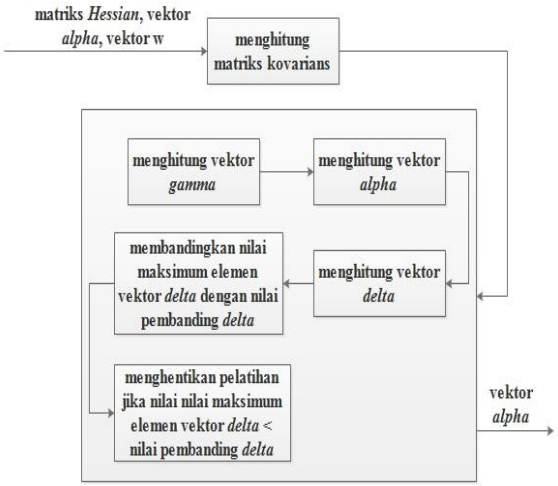
Setelah proses estimasi parameter vektor **w** telah dilakukan, langkah berikutnya adalah melakukan estimasi *hyperparameter alpha* α. Adapun langkah-langkah yang dilakukan dalam hal estimasi *hyperparameter* dapat dilihat pada Gambar 2.12.

Langkah pertama yang dilakukan adalah membuat matriks kovarians (*covariance*) **Σ**, dengan menggunakan persamaan:

$$\mathbf{\Sigma} = \mathbf{H}^{-1} \qquad (2.20)$$

dimana $\mathbf{H^{-1}}$ merupakan invers dari matriks *Hessian*.

Langkah berikutnya adalah proses iterasi untuk estimasi ulang vektor *hyperparameter* **α**, yang proses-prosesnya adalah sebagai berikut.



Gambar 11. Proses Estimasi *Hyperparameter Alpha*

a. Menghitung variabel vektor **γ** dengan menggunakan persamaan

$$\gamma_j = 1 - \alpha_j\Sigma_{jj} \qquad (2.21)$$

dimana α merupakan vektor *hyperparameter* dan Σ merupakan matriks kovarians.

b. Langkah selanjutnya adalah mengganti vektor *hyperparameter* sebelumnya dengan vektor *hyperparameter* yang baru. Vektor *hyperparameter* yang baru didapat dengan menggunakan persamaan:

$$\alpha_j = \frac{\gamma_j}{w_j^2}; \text{jika } i < \frac{i_{max}}{2}$$
$$\alpha_j = \gamma_j \left(\frac{w_j^2}{\gamma_j} - \Sigma_{jj} \right)^{-1}; \text{jika } i \geq \frac{i_{max}}{2} \quad (2.22)$$

dimana γ_i merupakan elemen vektor $\boldsymbol{\gamma}$, $\mathbf{w}_i$ merupakan elemen dari vektor bobot $\mathbf{w}$, dan i_{max} merupakan jumlah iterasi maksimum pelatihan yang telah diinisialisasi sebelumnya. Berikutnya vektor *hyperparamater* yang lama diganti dengan vektor *hyperparameter* yang baru.

c. Langkah berikutnya adalah menghitung variabel vektor δ, dengan menggunakan persamaan

$$\delta_j = \left| \ln(\alpha_j) - \ln(\alpha_j^{\text{lama}}) \right| \quad (2.23)$$

Dimana α merupakan vektor *hyperparameter*.

Setelah langkah iterasi a hingga c dilakukan, selanjutnya adalah membandingkan nilai elemen terbesar dari vektor *delta alpha* **δ** dengan nilai pembanding minimum *delta alpha*. Jika nilai elemen terbesar dari vektor *delta alpha* **δ** kurang dari nilai pembanding minimum *delta alpha*, maka proses pelatihan selesai dan parameter yang telah diestimasi dianggap telah optimum. Sedangkan jika nilai elemen terbesar dari vektor *delta alpha* **δ** lebih besar atau sama dengan dari nilai pembanding minimum *delta alpha*, maka proses pelatihan terus dilanjutkan. Untuk nilai pembanding minimum *delta alpha* pada penelitian ini adalah 10^{-6}.

Semua langkah-langkah yang telah dilakukan diatas terus-menerus diulang hingga mencapai nilai maksimum iterasi ataupun kondisi berhenti yang lainnya telah tercapai. Adapun hasil akhir dari proses pelatihan adalah berupa vektor parameter model **w** yang telah optimal.

2.4 Klasifikasi *Relevance Vector Machine*

Setelah mendapatkan vektor parameter model **w** dalam proses pelatihan yang telah dilakukan sebelumnya, berikutnya adalah melakukan klasifikasi menggunakan model yang ada. Proses klasifikasi dalam kasus peringkasan teks otomatis adalah melakukan klasifikasi kalimat dalam sebuah teks menjadi kalimat yang penting (yang probabilitas masuk kedalam kalimat pentingnya bernilai lebih dari 0.5), dan kalimat yang tidak penting (yang probabilitas masuk kedalam kalimat pentingnya bernilai kurang dari atau sama dengan 0.5). Adapun langkah-langkah pengklasifikasian dapat dilihat pada Gambar 13.



Gambar 12. Proses Klasifikasi *Relevance Vector Machine*

Sebelum dilakukan klasifikasi, terlebih dahulu dilakukan perhitungan nilai prediksi untuk setiap elemen vektor kalimat data uji menggunakan persamaan:

$$y(x; w) = \sum_{i=1}^{M} \mathbf{w}^T \varphi(\mathbf{x}) + w_0 \quad (2.24)$$

dimana w merupakan vektor parameter bobot yang telah didapatkan pada proses pelatihan, dan **x** merupakan data latih. Nilai prediksi kalimat tersebut selanjutnya diubah kedalam bentuk probabilitas, dengan menggunakan fungsi *logistic sigmoid*, menggunakan persamaan:

$$\sigma(y) = \frac{1}{1 + \exp(-(y))} \quad (2.25)$$

dimana y merupakan nilai prediksi yang sebelumnya telah dihitung. Nilai probabilitas tersebut akan dikonversikan kedalam label kelas (0 atau 1), yang akan menentukan apakah kalimat tersebut masuk kedalam kalimat penting atau tidak. Jika nilai probabilitas kalimat lebih dari 0.5, maka kalimat tersebut memiliki nilai klasifikasi 1 atau masuk ke dalam label kelas kalimat "Penting". Sedangkan jika nilai probabilitas kalimat kurang dari atau sama dengan 0.5, maka kalimat tersebut memiliki nilai klasifikasi 0 atau masuk ke dalam label kelas kalimat "Tidak Penting".

Setelah melakukan klasifikasi kalimat, langkah terakhir adalah menyusun hasil klasifikasi kalimat tersebut menjadi sebuah ringkasan.

2.5 Cosine Similarity

Kalimat hasil ringkasan akan disusun menggunakan metode *cosine* similarity dimana sebelum masuk pada metodenya terlebih dahulu dilakuakan pembobotan *term* dilakukan untuk menghitung bobot *term* pada kalimat, dimana *term* tersebut merupakan *token-token* hasil dari *preprocessing* yang sebelumnya telah dilakukan. Pembobotan *term* ini menggunakan metode *tf-idf*,

dimana metode ini menghitung frekuensi kemunculan *term* di dalam dokumen (*tf*) dan perhitungan jumlah dokumen yang mengandung sebuah *term* yang dicari dari kumpulan dokumen yang ada (*idf*). Adapun persamaan *tf-idf* yang akan digunakan adalah persamaan *tf-idf* yang telah dinormalisasi, yang persamaannya adalah sebagai berikut

$$w_{i,j} = \frac{tf_{i,j} \cdot \log_2 \frac{N}{n_j}}{\sqrt{\sum_{s=1}^{k} (tf_{i,s} \cdot \log_2 \frac{N}{n_j})^2}} \qquad (2.26)$$

dimana tf merupakan *term frequency*, N merupakan jumlah dokumen, dan n_j merupakan jumlah dokumen dimana *term* j muncul.
Setelah dilakukan pembobotan *term* selanjutnya akan dilakukan peritungan menggunakan metode *cosine similarity* dengan persamaan
$sim(K1, Ki) = \frac{\sum(K1.Ki)}{\sqrt{\sum K1^2}\sqrt{\sum Ki^2}}$ (2.27).
Proses ini akan menghasilkan ringkasan hasil dari pengurutan.

2.6 Pengujian Hasil Ringkasan

Pengujian hasil ringkasan dilakukan menggunakan metode evaluasi intrinstik yaitu hanya diukur dari kualitas hasil output ringkasan yang dihasilkan. Pengevaluasi menciptakan sebuah ringkasan manual dengan model ekstraktif, untuk menguji teks. Kemudian membandingkan hasil ringkasan sistem dengan ringkasan manual.

Adapun pengujian ringkasan yang akan dilakkan yaitu pengujian dengan cara menghitung performasi *recall, precission, f-measure* dan akurasi. Dimana perhitungannya mengugunakan rumus :

$recall = \frac{tp}{tp+fn}$

$precission = \frac{tp}{tp+fp}$

$f\text{-}measure = \frac{2*precission*recall}{recall+precission}$

$akurasi = \frac{tp+tn}{tp+fp+fn+tn}$

dimana :
tp : jumlah kalimat yang dihasilkan sistem dan juga ada dikalimat manual.
fp : jumlah kalimat ringkasan yang hanya ada di sistem tapi tidak ada di manual
fn : jumlah kalimat yang hanya ada di manua tidak ada di sistem.
tn : jumlah kalimat yang tidak termasuk pada ringkasan manual dikurangi fp

pada proses pengujian digunkan data sebagai berikut:

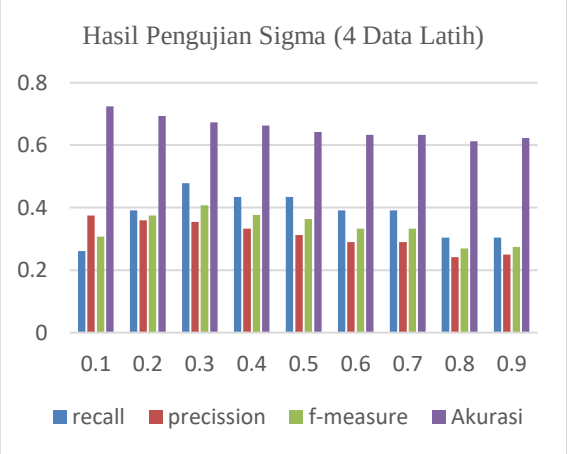
Proses	Jumlah Artikel Berita yang digunakan	Julah kaliat Penting	Jumlah kalimat tidak penting
Pelatian	4	16	28
Pengujian	7	23	75

Dari data tersebut didapatkan sebagai berikut :

Tabel 2.1 Hasil Pengujian Nilai Sigma (4 data latih)

Sigma	tp	fp	fn	tn	*R*	*P*	*F*	*A*
0.1	6	10	17	65	0.26	0.37	0.31	0.72
0.2	9	16	14	59	0.39	0.36	0.38	0.69
0.3	**20**	**12**	**12**	**55**	**0.48**	**0.35**	**0.41**	**0.67**
0.4	10	20	13	55	0.43	0.33	0.37	0.66
0.5	10	22	13	53	0.43	0.31	0.36	0.64
0.6	9	22	14	53	0.39	0.29	0.33	0.63
0.7	9	22	14	52	0.39	0.29	0.33	0.63
0.8	7	22	16	53	0.30	0.24	0.27	0.61
0.9	7	21	16	54	0.30	0.25	0.27	0.62

Dimana :
R = *Recall*
P = *Precission*
F = *F-measure*
A = *Akurasi*

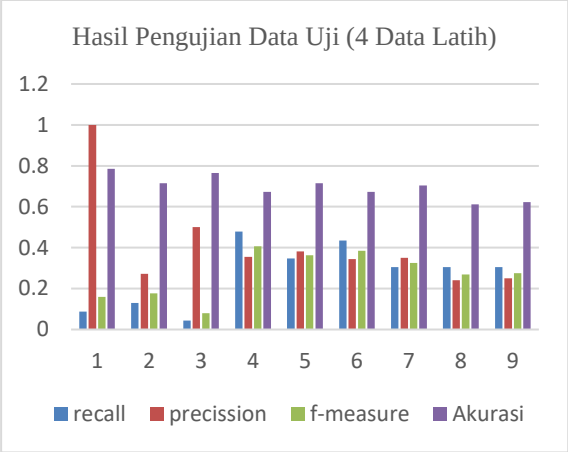


Gambar 13 Hasil Pengujian Sigma (4 Data Latih)
Dari pengujian sigma tersebut didapatkkan nilai sigma terbaik yaitu 0.3 emudian dilakukan pengujian data uji paling optimal sebagai berikut:

Tabel 2.2 Hasil ringkasan Data Uji (4 data latih)

Banyak Data latih	tp	fp	fn	tn	R	P	F	A
1	2	0	21	75	0.09	1	0.16	0.79
2	3	8	20	67	0.13	0.27	0.18	0.71
3	1	1	22	74	0.04	0.5	0.08	0.77
4	**20**	**20**	**12**	**55**	**0.48**	**0.35**	**0.41**	**0.67**
5	8	13	15	62	0.35	0.38	0.36	0.71
6	10	19	13	56	0.43	0.34	0.38	0.67
7	7	13	16	62	0.30	0.35	0.33	0.70

Dimana :
R = *Recall*
P = *Precission*
F = *F-measure*
A = *Akurasi*



Gambar 14 Hasil Pengujian Data Uji (4 Data Latih)
Hasil dari pengujian didapatkan *recall* 0.48, *precission* 0.35, *f-measure* 0.41 dan akurasi 0.67.

3 PENUTUP

Kesimpulan yang didapat dari penelitian ini adalah bahwa metode RVM (*Relevance Vector Machine)* dapat digunakan pada peringkasan multi dokumen. Adapun dari sisi akurasinya, sistem peringkasan diuji menggunakan metode intrinsik, untuk pencarian nilai *recall, precission, f-measure* dan akurasi-nya. Dari pengujian instrinsik tersebut didapatkan rata-rata *recall, precission dan f-measure* sebesar 41% dengan akurasi sebesar 67%.

Terdapat beberapa saran untuk perbaikan pda penelitian ini, diantaranya adalah :

1. Dibuat NER (*Named Entity Recognizer*) otomatis untuk mendeteksi entitas.
2. Setelah diperoleh f-measure ternyata metode RVM masih kurang baik performasinya maka diperlukan metode lain dengan ekstraksi fitur yang sama.

DAFTAR PUSTAKA

[1] Juan-Manuel dan T.-M. , Automatic Text Summarization, Great Britain: ISTE Ltd and John Wiley & Sons, Inc., 2014.

[2] A. Riyanto, Peringkasan teks menggunakan metode K-means, Bandung: unikom, 2016.

[3] M. L. K. B. R. T. Deni Fitriaman#1, "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Pada Multi-Document Menggunakan Metode Support Vector Machines (SVM)," *Peringkasan Teks otomatis.*

[4] C. M. Bishop, Variational Relevance Vector Machines, 7 J. J. Thompson Avenue, Cambridge CB3 0FB, U.K: http://research.microsoft.com/{~cmbishop,~mtipping}, 2000.

[5] A. T. W. 2. G. S. 3. Zulkifli 1, "Pembobotan Fitur Ekstraksi Pada Peringkasan Teks Bahasa Indonesia Menggunakan Algoritma Genetika," *Pembobotan Fitur Ekstraksi Pada Peringkasan Teks Bahasa Indonesia Menggunakan Algoritma Genetika*, pp. 2-3.

[6] M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Sparse Bayesian Learning and the Relevance Vector Machine*, 2001.

[7] R. Hidaya, "implementasi Support Vector Machines Untuk Pencarian Informasi Buku Di perpustakaan Daerah Bandug Provinsi Jawa Barat".

[8] M. Mustaqhfiri, Z. Abidin dan R. Kusumawati, "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance," *Matics Matics*, vol. 4, no. 4, pp. 134-147, 2011.

[9] Andita Dwiyoga Tahitoe - Diana Purwitasari, "IMPLEMENTASI MODIFIKASI ENHANCED CONFIX STRIPPING STEMMER UNTUK BAHASA INDONESIA DENGAN METODE CORPUS BASED STEMMING," *IMPLEMENTASI MODIFIKASI ENHANCED CONFIX STRIPPING STEMMER UNTUK BAHASA INDONESIA DENGAN METODE CORPUS BASED STEMMING* .

[10] A. budiman, Peringkasan Teks Otomatis menggunakan SVM, Bandung: UNIKOM, 2016.

[11] Jogianto, Analisis & Desain, Yogyakarta: Andi.

[12] A.-B. B. LADJAMUDIN, Analisis dan Desain Sistem Informasi, Yogyakarta: GRAHA ILMU, 2005.

[13] Z. A. d. R. K. Muchammad Mustaqhfiri1, "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance," *Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance* .

[14] A. D. T. -. D. Purwitasari, "IMPLEMENTASI MODIFIKASI ENHANCED CONFIX STRIPPING STEMMER UNTUK BAHASA INDONESIA DENGAN METODE CORPUS BASED STEMMING," *IMPLEMENTASI MODIFIKASI ENHANCED CONFIX STRIPPING STEMMER UNTUK BAHASA INDONESIA DENGAN METODE CORPUS BASED STEMMING* .

[15] P. Roger S. Pressman, Software Engineering A Practitioner' s Approach, Buston: Mc Graw Hill, 2001.

[16] I. Sommerville, Sofware engineriing edisi 6 Jilid 1(Rekayasa Perangkat Lunak).

[17] .. P. D. Sugiyono, Metode Penelitian Kombinasi, 2014.

[18] F. Z. Tala, “A Study of Steming Effect on Information Retrieval in Bahasa Indonesia”.