# IMPELENTASI METODE *RELEVANCE VECTOR MACHINE* DALAM PERINGKASAN TEKS OTOMATIS

Arief Budiman Eka Putra
Universitas Komputer Indonesia
Jl. Dipatiukur No. 112-116, Bandung 40132
E-mail : ariefbud688@gmail.com

## ABSTRAK

*Support Vector Machine* (*SVM*) merupakan salah satu metode pembelajaran mesin bersifat *supervised* yang digunakan untuk proses regresi dan klasifikasi. *SVM* telah banyak diimplementasikan dalam berbagai permasalahan dunia nyata, salah satunya adalah untuk peringkasan teks otomatis. Berdasarkan hasil penelitian yang dilakukan, *SVM* memiliki beberapa kelemahan. Oleh karena itu dibuatlah sebuah metode klasifikasi dan regresi yang dapat mengatasi kekurangan-kekurangan yang dimiliki oleh *SVM*, yaitu *Relevance Vector Machine* (*RVM*). Dari hasil berbagai penelitian dan perbandingan yang dilakukan menunjukan bahwa *RVM* dalam melakukan regresi dan klasifikasi memiliki tingkat akurasi yang lebih baik dengan waktu komputasi lebih cepat dari *SVM*. Penelitian ini mencoba untuk mengimplementasikan metode *RVM* pada peringkasan teks otomatis pada dokumen tunggal. Metode *RVM* ini akan digunakan pada saat proses klasifikasi kalimat hasil *preprocessing* kalimat dalam dokumen. Kalimat dalam dokumen akan dibagi menjadi dua kelas, yaitu kelas positif dan kelas negatif. Kalimat positif berisi kalimat yang termasuk ke dalam ringkasan, sedangkan kalimat negatif tidak termasuk ke dalam ringkasan. Dari hasil penelitian ini, didapat bahwa metode *RVM* ini dapat diimplementasikan dalam kasus peringkasan teks otomatis. Adapun dari sisi akurasinya, sistem peringkasan diuji menggunakan metode evaluasi ringkasan intrinsik, dimana didapat nilai rata-rata *recall*, *precision*, dan *f-measure* sebesar 53%.

**Kata kunci :** *SVM*, *RVM*, Peringkasan, *Preprocessing*, Klasifikasi

## 1. PENDAHULUAN

*Support Vector Machine* (*SVM*) merupakan salah satu metode pembelajaran mesin bersifat *supervised* (*supervised machine learning*) yang digunakan untuk proses regresi dan klasifikasi [1]. Dari hasil beberapa penelitian yang mengkaji baik secara teori maupun implementasi, *Support Vector Machine* memiliki beberapa kelebihan, diantaranya adalah tingkat akurasi yang tinggi serta impelementasinya yang relatif mudah. *Support Vector Machine* telah banyak diimplementasikan dalam berbagai permasalahan dunia nyata (*real-world problems*), salah satunya adalah untuk peringkasan teks otomatis (*automatic text summarization*) [2].

Berdasarkan hasil penelitian yang dilakukan [1], *Support Vector Machine* memiliki beberapa kelemahan. Salah satu dari kelemahan tersebut antara lain proses prediksinya yang tidak bersifat probabilistik yang dapat mengurangi tingkat akurasi dari klasifikasi [1]. Oleh karena itu dibuatlah sebuah metode klasifikasi dan regresi yang dapat mengatasi kekurangan-kekurangan yang dimiliki oleh *Support Vector Machine*. Metode tersebut adalah *Relevance Vector Machine* (*RVM*).

*Relevance Vector Machine* adalah sebuah metode pembelajaran mesin yang diadaptasi dari *Bayesian Framework* dan memiliki bentuk model fungsi yang mirip dengan *Support Vector Machine*. Dari hasil berbagai penelitian dan perbandingan yang dilakukan menunjukkan bahwa *Relevance Vector Machine* dalam melakukan regresi dan klasifikasi memiliki tingkat akurasi yang lebih baik dengan waktu komputasi lebih cepat dari *Support Vector Machine* [1] [3], yang ditunjukkan dengan jumlah *relevant vector* (*RV*) yang dihasilkan oleh *Relevance Vector Machine* jauh lebih sedikit dari jumlah *support vector* (*SV*) yang dihasilkan oleh *Support Vector Machine*.

Penelitian kali ini mencoba untuk mengimplementasikan metode *Relevance Vector Machine* pada peringkasan teks otomatis pada dokumen tunggal (*single document*). Sama halnya dengan metode pembelajaran mesin lainnya, metode *Relevance Vector Machine* ini akan digunakan pada saat proses klasifikasi kalimat hasil *preprocessing* kalimat dalam dokumen. Kalimat dalam dokumen akan dibagi menjadi dua kelas, yaitu kelas positif dan kelas negatif. Kalimat positif berisi kalimat yang termasuk ke dalam ringkasan, sedangkan kalimat negatif tidak termasuk ke dalam ringkasan [2]. Adapun tujuan yang ingin dicapai dalam penelitian ini adalah melihat akurasi metode *Relevance Vector Machine* dalam membuat sebuah ringkasan, dengan menggunakan metode evaluasi ringkasan intrinsik, yaitu akurasi dari ringkasan hanya diukur dari kualitas hasil keluaran (*output*) ringkasan yang dihasilkan.

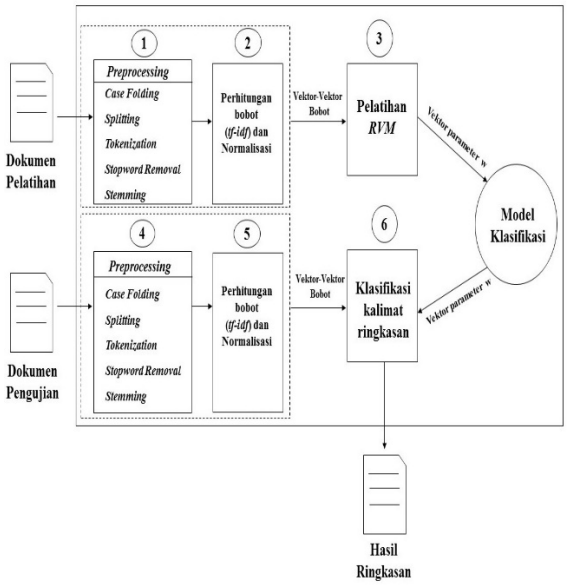
Dengan dilakukannya penelitian ini diharapkan bahwa hasil dari penelitian dapat dijadikan pendekatan baru dalam mengembangkan peringkasan teks otomastis yang sampai saat ini masih terus dikembangkan dan diteliti [4].

## 2. ISI PENELITIAN

Bagian ini terbagi menjadi beberapa bagian, yaitu penjelasan arsitektur sistem, *preprocessing*, pembobotan *term*, pelatihan, klasifikasi, dan pengujian 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.




**Gambar 1.** Arsitektur Sistem

Sistem peringkas teks otomatis yang dibangun memiliki 4 proses utama yaitu *preprocessing*, perhitungan bobot kata (term), proses pelatihan untuk menghasilkan model klasifikasi, serta proses pengujian berupa klasifikasi kalimat yang akan dijadikan ringkasan menggunakan model yang yang telah didapatkan pada proses pelatihan.

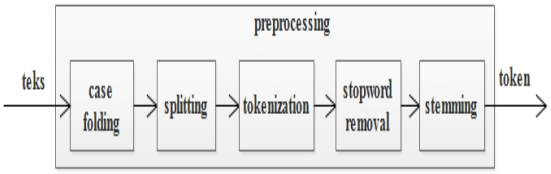
Untuk penjelasan setiap tahap pada proses peringkasan berdasarkan pada Gambar 1 dapat dilihat sebagai berikut:

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*, *stopword removal*, dan *stemming*.
2. Tahap berikutnya adalah melakukan perhitungan bobot setiap kata (*term*) dalam data latih dengan menggunakan metode *tf-idf* dengan menghitung *TF* (*Term Frequency*), *DF* (*Document Frequency*), dan dihitung bobot dari masing-masing *term* pada data latih. Setelah didapat nilai dari *tf-idf* masing-masing kalimat, langkah selanjutnya adalah melakukan normalisasi *tf-idf*.
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*, *stopword removal*, dan *stemming*.
5. Berikutnya adalah melakukan pembobotan bobot pada setiap *term* untuk data uji, dimana sama seperti tahap ketiga menggunakan metode *tf-idf* dan nantinya nilai *tf-idf* tersebut akan dilakukan normalisasi.
6. Selanjutnya melakukan proses pengujian dengan menggunakan model klasifikasi yang telah didapatkan pada proses pelatihan.

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

### 2.2 *Preprocessing*

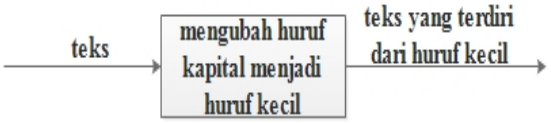
*Preprocessing* ini merupakan proses-proses yang dilakukan untuk mendapatkan fitur kata-kata yang bersih sehingga tidak akan mengandung *noise* yang akan menganggu hasil untuk proses selanjutnya. Tahapan-tahapan dari *preprocessing* dapat dilihat pada Gambar 2.




**Gambar 2.** Tahapan *Preprocessing*

1. *Case Folding*

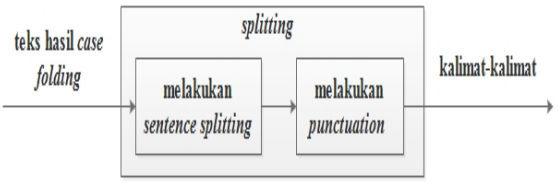
   *Case folding* adalah proses yang mengubah semua huruf dalam teks dokumen menjadi huruf kecil. Proses dari *case folding* dapat dilihat pada Gambar 3. Sebagai contoh, untuk kata “Dalam” setelah melewati proses *case folding* berubah menjadi kata “dalam”, dimana huruf kapital ‘D’ diubah menjadi huruf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




**Gambar 3.** Proses *Case Folding*

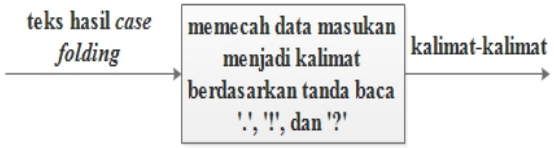
2. *Splitting*

   *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*.



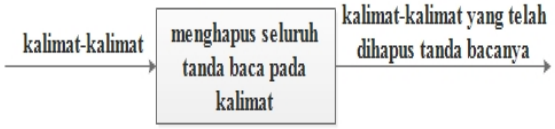

**Gambar 4.** Tahapan *Splitting*

*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.



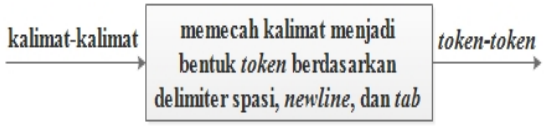

**Gambar 5.** Proses *Sentence Splitting*

*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*

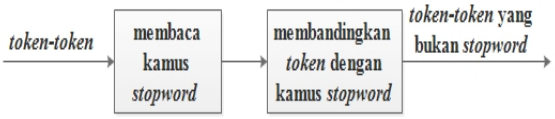
3. *Tokenization*
   *Tokenization* adalah proses pemotongan *string* menjadi kumpulan *token-token*. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melakukan pemindaian kalimat dengan pemisah (*delimiter*) serta *white space* (spasi, *tab*, dan *newline*). Langkah-langkah dari tokenization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




**Gambar 7.** Proses *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.

5. *Stemming*
   *Stemming* adalah proses pengembalian kata berimbuhan menjadi kata dasar. Algoritma yang digunakan dalam proses *stemming* pada penelitian ini adalah algoritma Nazief-Adriani. Adapun gambaran proses dari *stemming* dapat dilihat pada 9.




**Gambar 9.** Proses *Stemming*

Sebagai contoh, pada *token* "urusan", setelah melewati tahapan *stemming*, menjadi "urus".

### 2.3 Pembobotan *Term*

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

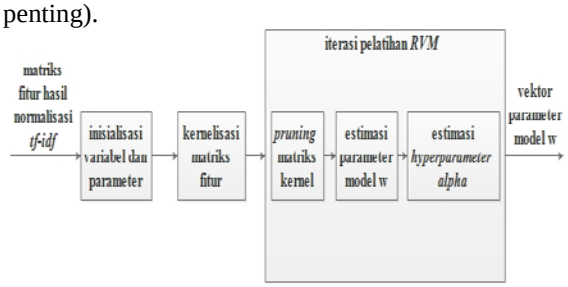
telah dinormalisasi, yang persamaannya adalah sebagai berikut

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

dimana tf merupakan *term frequency*, N merupakan jumlah dokumen, dan $n_j$ merupakan jumlah dokumen dimana *term* j muncul.

### 2.4 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 10. 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 10.** Proses Pelatihan *Relevance Vector Machine*

Berdasarkan Gambar 10, 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:

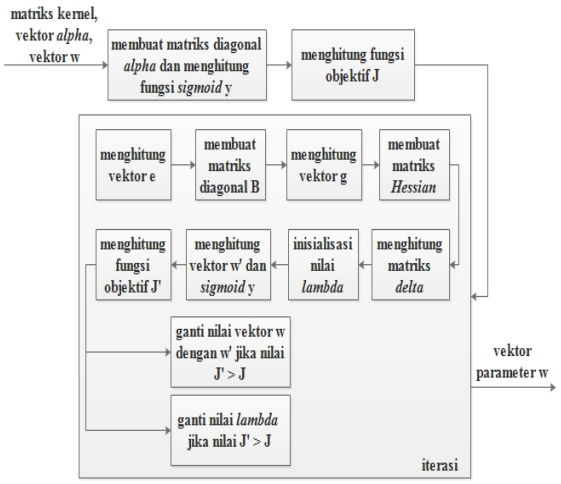
$$K(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2}) \quad (2)$$

   Dimana x dan x' merupakan data latih dan σ merupakan parameter bebas. Pada matriks hasil kernel juga akan ditambah dengan *bias*, dimana pada kolom pertama matriks hasil kernel akan diisi dengan angka 1, sehingga menghasilkan matriks dengan ukuran kolom M+1, dimana M merupakan jumlah kolom matriks data latih awal. Nilai *bias* nantinya akan digunakan pada saat proses klasifikasi.
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:

$$\ln p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) = \sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} + \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} \quad (3)$$

   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 11.




**Gambar 11.** 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 (4), dimana x merupakan data latih, w merupakan vektor parameter model yang sebelumnya telah diinisialisasi, **φ(x)** merupakan fungsi kernel.

$$y(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-(\mathbf{w}.\varphi(\mathbf{x})))} \qquad (4)$$

Setelah mendapatkan parameter-parameter yang dibutuhkan, maka persamaan (3) dapat dilakukan proses maksimasi. Adapun proses estimasi menggunakan metode *Iterative Reweighted Least Square* (*IRLS*), dengan terlebih dahulu mencari invers dari hasil persamaan (3). Setelah mendapatkan nilai invers dari persamaan (3), maka selanjutnya adalah melakukan proses *Iterative Reweighted Least Square*, dengan langkah-langkah sebagai berikut.

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

$$\mathbf{e} = \mathbf{t} - \mathbf{y} \qquad (5)$$

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:

$$\beta_j = y_j\,(1\text{-}y_j) \qquad (6)$$

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:

$$\mathbf{g} = \varphi_n^T \mathbf{e} - \mathbf{A}\mathbf{w}_n \qquad (7)$$

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 (3), dengan menggunakan persamaan:

$$\mathbf{H} = \boldsymbol{\varphi}_n^T \mathbf{B} \boldsymbol{\varphi}_n + \mathbf{A} \qquad (8)$$

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:

$$\Delta = \mathbf{H}^{-1}.\mathbf{g} \qquad (9)$$

dimana $\mathbf{H}^{-1}$ merupakan invers dari matriks *Hessian*, dan **g** merupakan vektor gradient negatif dari fungsi objektif persamaan (3).

g. Selanjutnya adalah melakukan inisialisasi nilai λ yang akan digunakan untuk proses iterasi *update* vektor **y**, nilai fungsi objektif persamaan (3) dan vektor bobot **w**. Inisialiasi nilai λ dengan nilai 1. Adapun proses iterasi *update* vektor **y**, nilai fungsi objektif persamaan (3) dan vektor bobot **w** adalah sebagai berikut.

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

$$\mathbf{w}' = \mathbf{w}_n + \lambda\Delta \qquad (10)$$

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:

$$\mathbf{y} = \frac{1}{1 + \exp(-(\mathbf{w}'.\varphi(\mathbf{x})))} \qquad (11)$$

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

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

$$J' = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \frac{1}{2}\mathbf{w'}^{T}\mathbf{A}\mathbf{w'} \quad (12)$$

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 (3) 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*

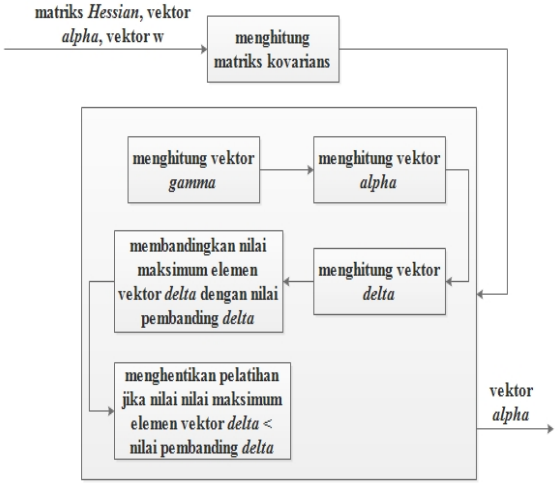
Setelah proses estimasi parameter vektor **w** telah dilakukan, langkah berikutnya adalah melakukan estimasi *hyperparameter alpha* α. Adapun langkah-langkah yang dilakukan dalam hal estimasi *hyperparameter* dapat dilihat pada Gambar 12.

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

$$\mathbf{\Sigma} = \mathbf{H}^{-1} \quad (13)$$

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 12.** Proses Estimasi *Hyperparameter Alpha*

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

$$\gamma_j = 1 - \alpha_j \Sigma_{jj} \quad (14)$$

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:

$$\alpha_j = \frac{\gamma_j}{w_j^2}; \text{jika } i < \frac{i_{max}}{2}$$
$$\alpha_j = \gamma_j \left(\frac{w_j^2}{\gamma_j} - \Sigma_{jj}\right)^{-1}; \text{jika } i \geq \frac{i_{max}}{2} \quad (15)$$

dimana $\gamma_i$ merupakan elemen vektor **γ**, $\mathbf{w}_i$ merupakan elemen dari vektor bobot **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

$$\delta_j = \left|\ln(\alpha_j) - \ln(\alpha_j^{lama})\right| \quad (16)$$

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.

### 2.5 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 13.** Proses Klasifikasi *Relevance Vector Machine*

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

$$y(x; w) = \sum_{i=1}^{M} \mathbf{w}^{T}\varphi(\mathbf{x}) + w_0 \qquad (17)$$

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:

$$\sigma(y) = \frac{1}{1 + \exp(-(y))} \qquad (18)$$

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. Ringkasan terdiri dari kumpulan kalimat-kalimat yang memiliki label kelas kalimat "Penting" atau yang bernilai 1. Jumlah kalimat yang diambil sebanyak 30% dari keseluruhan kalimat. Apabila dari hasil klasifikasi terdapat lebih dari 30% jumlah kalimat, maka yang diambil adalah kalimat yang memiliki nilai probabilitas yang paling tinggi.

### 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 data yang akan digunakan pada pengujian performansi adalah berita yang bertema "Penembakan Dubes Rusia di Ankara". Berita tersebut terbagi ke dalam dua bagian, yaitu data latih dan data uji. Data latih terdiri dari 20 artikel berita dan untuk data uji sebanyak 6 artikel berita yang diambil dari beberapa situs portal berita, seperti detik.com, kompas.com, viva.com, voa.com, dan okezone.com. Untuk data latih dan data uji, ringkasan manual akan dibuat oleh satu orang ahli bahasa, yaitu seorang guru Bahasa Indonesia. Data uji yang nantinya akan dihitung performansinya untuk mengukur akurasi dari sistem peringkasan teks otomatis yang telah dikembangkan. Adapun rincian dari data latih serta data uji yang akan digunakan dapat dilihat pada Tabel 1, dan perbandingan hasil ringkasan yang dihasilkan secara manual oleh ahli Bahasa dan oleh sistem dapat dilihat pada Tabel 2.

**Tabel 1.** Rincian Data Latih dan Data Uji dalam Pengujian

| Proses | Jumlah Artikel | Jumlah Kalimat Penting | Jumlah Kalimat Tidak Penting |
|---|---|---|---|
| Pelatihan | 20 | 98 | 230 |
| Pengujian | 6 | 35 | 81 |

**Tabel 2.** Perbandingan Hasil Ringkasan Manual dan Ringkasan Sistem

| Dokumen ke- | Jumlah Kalimat Ringkasan | Indeks Ringkasan (Manual) | Indeks Ringkasan (Sistem) |
|---|---|---|---|
| 1 | 5 | 1,2,4, 5,17 | 1,2,3, 4,15 |
| 2 | 3 | 1,2,5 | 1,2,4 |
| 3 | 5 | 1,2,3, 8,12 | 1,2,3, 5,6 |
| 4 | 12 | 1,2,3, 4,16,19, 20,21,22, 26,29,31 | 1,2,3, 4,5,6, 9,11,14, 15,16,17 |
| 5 | 5 | 1,2,4, 6,16 | 1,2,3, 13,14 |
| 6 | 4 | 1,2,4, 11 | 1,2,3, 13 |

Pengujian performansi dilakukan dengan cara menghitung *recall*, *precision*, dan *f-measure* untuk masing-masing dokumen, dengan persamaan:

$$r = \frac{\sum KR}{\sum KRM} \tag{19}$$

$$p = \frac{\sum KR}{\sum KRS} \tag{20}$$

$$f = \frac{2 * p * r}{p + r} \tag{21}$$

dimana ∑KR merupakan jumlah kalimat relevan, yaitu jumlah kalimat ringkasan yang terdapat pada sistem dan juga terdapat pada ringkasan manual, ∑KRM merupakan jumlah kalimat ringkasan yang dihasilkan secara manual, dan ∑KRS merupakan jumlah kalimat ringkasan yang dihasilkan oleh sistem.

Adapun hasil perhitungan dari recall, precision dan f-measure untuk masing-masing dokumen dapat dilihat pada Tabel 3.

**Tabel 3.** Hasil Perhitungan Performansi

| Dok | ΣKS | ΣKR | ΣKRS | *r* | *p* | *f* |
|---|---|---|---|---|---|---|
| 1 | 5 | 5 | 3 | 0.6 | 0.6 | 0.6 |
| 2 | 3 | 3 | 2 | 0.67 | 0.67 | 0.67 |
| 3 | 5 | 5 | 3 | 0.6 | 0.6 | 0.6 |
| 4 | 12 | 12 | 5 | 0.42 | 0.42 | 0.42 |
| 5 | 5 | 5 | 2 | 0.4 | 0.4 | 0.4 |
| 6 | 4 | 4 | 2 | 0.5 | 0.5 | 0.5 |

**Keterangan:**
ΣKRS: ΣKalimat Sistem
ΣKRM: ΣKalimat Ringkasan Manual
ΣKR: ΣKalimat Ringkasan Relevan
r: *recall*
p: *precision*
f: *f-measure*

The results of the calculations in Table 3 are then calculated their average score to obtain a value of recall, precision and F-measure, which resulted in a score of 0.53 or 53%.

## 3. PENUTUP

Kesimpulan yang didapat dari penelitian ini adalah bahwa metode *Relevance Vector Machine* ini dapat diimplementasikan dalam kasus peringkasan teks otomatis untuk dokumen tunggal (*single document*). Adapun dari sisi akurasinya, sistem peringkasan diuji menggunakan metode evaluasi ringkasan intrinsik, untuk dicari nilai *recall*, *precision*, dan *f-measure*-nya. Dari pengujian instrinsik tersebut, didapat nilai rata-rata *recall*, *precision*, dan *f-measure* sebesar 53%.

Terdapat beberapa saran untuk perbaikan pada penelitian ini, diantaranya adalah sebagai berikut.

1. Untuk *preprocessing*, khususnya pada *stopword removal*. Metode *stopword removal* selain memanfaatkan kamus *stopword*, dapat diuji coba, seperti pendekatan statistik.
2. Ekstraksi fitur yang lain dapat diuji coba, seperti posisi kalimat, ikatan leksikal antar kalimat, dan fitur-fitur lainnya yang telah diuji pada metode machine learning lain [2].

## DAFTAR PUSTAKA


[1] M. E. Tipping, "Relevance Vector Machine," *Advances in Neural Information Processing Systems,*, vol. NIPS 1999, no. 12, pp. 652-658, 2000.

[2] D. Fitriaman, M. L. Khodra and B. R. Trilaksono, "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Pada Multi-Document Menggunakan Metode Support Vector Machines (SVM)," Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, Bandung, 2012.

[3] B. Demir, "Hyperspectral Image Classification Using Relevance Vector Machines," *IEEE Geoscience And Remote Sensing Letters*, vol. 4, no. 4, pp. 586-589, 2007.

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

[5] A. D. Tahitoe and D. Purwitasari, "Implementasi Modifikasi Enchanced Confix Stripping Stemmer Untuk Bahasa Indonesia Dengan Metode Corpus Based Stemming," *Jurnal Informatika*, 2010.