RELEVANCE VECTOR MACHINE DALAM PERINGKASAN MULTIDOKUMEN

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

[1,2] Universitas Komputer Indonesia
Jl. Dipatiukur No. 112-116 Bandung
E-mail : tendiarifin0308@gmail.com[1], kania.evita.dewi@email.unikom.ac.id[2]

ABSTRAK

Penelitian ini bertujuan untuk menunjukkan efektifitas algoritma RVM dalam peringkasan otomatis. Setiap dokumen berita yang memiliki tema yang sama, dipraproses, diektrak fitur, kemudian dipilah oleh RVM mana yang merupakan kalimat ini dari kumpulan dokumen tersebut. Hasil dari penelitian ini diperoleh yaitu *recall* sebesar 47.83%, *precission* sebesar 35.48%, *f-measure* sebesar 40.74%, dan akurasi sebesar 67.35%. Hasil dari penelitian ini dirasa masih kurang, dikarenakan dibandingkan dengan algortima SVM. Tetapi algoritma RVM masih dapat digunakan untuk peringkasan teks multidokumen.

Kata kunci: Peringkasan teks Otomatis, Multidokumen, Ektraksi Fitur, RVM

1. PENDAHULUAN

Situs berita sudah sangat banyak sekarang ini, menurut https://www.it-jurnal.com/situs-berita-terpopuler-di-indonesia/ situs buatan CV.Indotekno tujuh situs berita yang terpopuler adalah detik.com, kompas.com, tribunnews.com, Republika.co.id, Okezone.com, Tempo.co, dan Antaranews.com. Berita yang dimuat seringkali sama walaupun dengan judul yang berbeda. Pembaca terkadang merasa kecewa ketika membuka berita dengan judul yang berbeda tetapi isinya sama. Salah satu cara agar pembaca tidak perlu membuka berita satu per satu, maka diperlukan suatu proses peringkasan berita. Tetapi jika proses peringkasan dilakukan secara manual maka ini pun akan membutuhkan waktu yang lama.

Peringkasan teks otomatis pada multidokumen adalah peringkasan banyak dokumen dengan tema yang sama menggunakan mesin. Peringkasan dapat dilakukan dengan mudah dengan bantuan alat/sistem [1]. Banyak penelitian yang sudah dilakukan menegnai peringkasan teks otomatis. Penelitian yang dilakukan oleh Agus Riyanto [2] mendapatkan dengan menggunakan algoritma K-means proses peringkasan teks otomatis diperoleh nilai precision, recall, f-measure secara berturut-turut adalah 45,5%, 57,74%, dan 46,13%. Sedangkan pada penelitian yang dilakukan oleh Deni Fitriaman [3] diperoleh bahwa peringkasan teks otomatis multidokumen dengan algoritma SVM (*Support Vector Machine*) yang kemudian disusun kalimatnya menggunakan algoritma MMR diperoleh nilai recall, precission, dan f-measure berturut-turut adalah 0,773, 0,771, dan 0,771.

Metode SVM telah dikembangkan oleh M. E.Tipping menjadi RVM (Relevance Vector Machine). Pengembangan yang dilakukan oleh E.Tipping adalah dalam pemilihan fungsi kernel didalam SVM, diubah dengan mengikuti prinsip kerja *Sparse Bayesian Learning* [4]. Pada penelitian yang dilakukan oleh Christoper M. Bishop, Michael E. Tipping diperoleh tingkat eror dari SVM lebih tinggi disbanding RVM, dimana nilai error SVM sebesar 10,6% sedangkan nilai error RVM 9,3% [5].

Masukan dari sebuah algoritma untuk proses peringkasan didalam machine learning adalah fitur-fitur yang menggambarkan perbedaan antar kalimat 1 dengan yang lainnya. Penelitian Zulkifi diperlihatkan 8 fitur yang menghasilkan akurasi yang tertinggi dalam proses peringkasan [6]. Pada penelitian ini diperoleh bahwa fitur ektraksi yang baik adalah posisi kalimat, kemiipan antar kalimat, kalimat yang menyerupai judul dokumen, kalimat yang mengandung kata entity, kalimat yang mengandung data numerik, panjang kalimat, koneksi antar kalimat dan penjumlahan bobot koneksi antar kalimat.

Berdasarkan penelitian-penelitian tersebut, maka akan dilakukan penelitian peringkasan teks otomatis dengan menggunakan algoritma RVM dengan menggunakan fitur ektraksi yang diperoleh dari penelitian Zulkifi. Dalam upaya memperoleh hasil ringkasan teks yang lebih menyerupai hasil manusia.

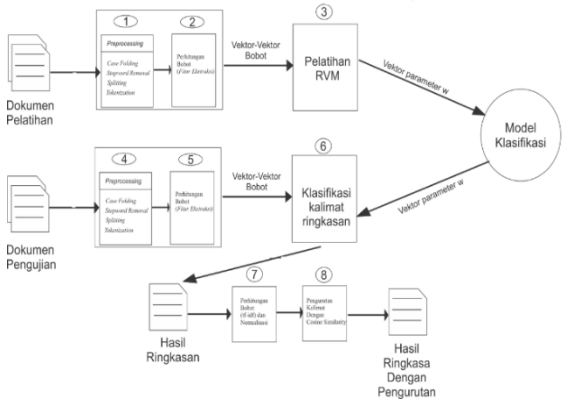
2. ISI PENELITIAN

Pada bab ini akan dijelaskan tentang arsitektur system, pengujian dan pembahasan hasil pengujian.

2. 1. Arsitektur Sistem

Pada penelitian ini setiap dokumen yang memiliki topik/tema yang sama akan disimpan dalam 1 file, kemudian akan masuk kedalam proses

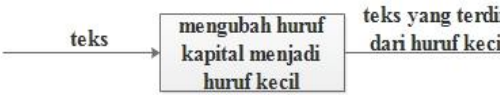
praproses, ektraksi fitur, pemilihan kalimat inti, kemudian kalimat yang sudah terpilih disusun sehingga menjadi sebuah paragraph. Gambaran system tersebut dapat dilihat pada gambar 1.



Gambar 1. Arsitektur sistem peringkasan

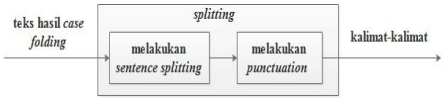
Proses yang terjadi dalam Gambar 1. Adalah sebagai berikut:

1. Dokumen/data latih berupa data berita yang diambil dari situs berita www.liputan6.com dan www.kompas.com, dalam hal ini hanya diambil yang memiliki topik olahraga, didalam topik olahraga diambil beberapa tema dalam sepakbola yang sedang banyak diperbincangkan. Berita yang memiliki tema yang sama disimpan dalam sebuah file. Data latih ini sudah divalidasi/diringkas oleh ahli.
2. Setiap data latih masuk kedalam praproses. Dalam tahap ini setiap data latih akan masuk ke dalam tahap *case folding*, *splitting*, *tokenization*, dan *stopword removal*. Berikut adalah penjelasan-penjelasan setiap bagian proses:
   a. Case folding
      Case folding adalah proses yang mengubah semua huruf dalam teks dokumen menjadi huruf kecil. Proses dari *case folding* dapat dilihat pada Gambar 2. Sebagai contoh, untuk kata "Dalam" setelah melewati proses *case folding* berubah menjadi kata "dalam", dimana huruf kapital 'D' diubah menjadi huruf 'd'.



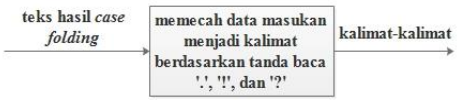
Gambar 2. Proses *Case Folding*

      Selain mengubah menjadi huruf kecil, pada proses *case folding* dilakukan proses penyatuan entitas dimana hasil dari proses penyatuan entitas akan digunakan pada tahapan fitur ektraksi.
   b. Splitting
      Splitting merupakan pemecahan sebuah teks utuh menjadi pecahan-pecahan kalimat. Gambaran dari proses *splitting* dapat dilihat pada Gambar 3. Proses *splitting* melibatkan dua sub-proses, yaitu *split sentences* dan *punctuation*.



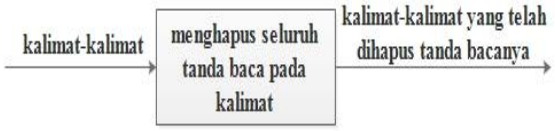
Gambar 3. Tahapan *Splitting*

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 [6]. Langkah-langkah dari *sentence splitting* dapat dilihat pada Gambar 4.



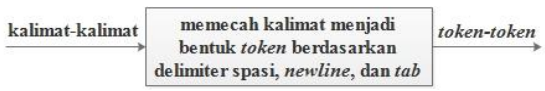
Gambar 4. Proses *Sentence Splitting*

Punctuation merupakan proses penghilangan tanda baca ataupun simbol pada kalimat yang telah dipecah [5]. Proses dari *punctuation* dapat dilihat pada Gambar 5.



Gambar 5. Proses *Punctuation*

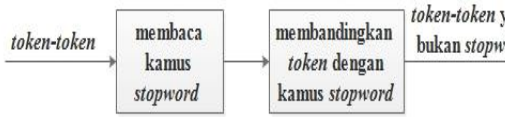
   c. Tokenization
      Tokenization adalah proses pemotongan *string* menjadi kumpulan *token-token*. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melakukan pemindaian kalimat dengan pemisah (*delimiter*) serta *white space* (spasi, *tab*, dan *newline*). Langkah-langkah dari tokenization dapat dilihat pada Gambar 6.



Gambar 6. 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".
   d. 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*" [7]. Proses dari *stopword removal* dapat dilihat pada Gambar 7.



Gambar 7. 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.

3. Tahap berikutnya adalah melakukan fitur ektraksi. Dalam penelitian ini akan menggunakan fitur ektraksi yang dihasilkan oleh penelitian zulkifi. Fitur ekstraksi yang akan digunakan adalah:
   a. Posisi kalimat (f1)
      Posisi kalimat adalah letak dalam sebuah kalimat. Kalimat inti biasanya terletak pada kalimat pertama pada setiap paragraf [6]. Jika sebuah dokumen memiliki N kalimat dan j adalah indeks kalimat (dimulai dari 0), maka fitur posisi kalimat ke-j dapat dihitung dengan cara:

$$f_{1j} = \frac{N - j}{N} \quad (1)$$

   b. Kemiripan Antar Kalimat (f2)
      Kemiripan antar kalimat adalah daftar kata-kata yang dimiliki oleh kalimat ke-j dengan kalimat lain [6]. Jika $A = \{a_1, a_n, \dots, a_n\}$ token-token pada kalimat ke-j dan $B = \{b_1, b_2, \dots, b_m\}$ token-token dalam kalimat ke-k, dimana $j \neq k$, dan M adalah jumlah kata yang ada didalam dokumen tersebut, maka fitur kemiripan antar kalimat ke-j dapat dihitung dengan cara:

$$f_{2j} = \frac{n(A \cap B)}{M} \quad (2)$$

   c. Kalimat yang Menyerupai Judul Dokumen (f3)
      Kalimat yang menyerupai judul dokumen adalah kumpulan kata sama dengan kata-kata yang ada pada judul kalimat [6]. Jika $A = \{a_1, a_n, \dots, a_n\}$ token-token pada kalimat ke-j dan $C = \{c_1, c_2, \dots, c_k\}$ token-token dalam judul. Maka fitur kalimat yang menyerupai judul dokumen untuk kalimat ke-j dapat dihitung dengan cara:

$$f_{3j} = \frac{n(A \cap C)}{n(A \cup C)} \quad (3)$$

   d. Kalimat yang mengandung Entiti (f4)
      Nama entiti adalah sebuah kumpulan kata yang memiliki makna atau membentuk nama sebuah intitusi. Misalnya adalah AC Milan merupakan kumpulan kata yang memiliki makna sebuah klub sepak bola. Jika $n(E_j)$ adalah jumlah kata entity dalam kalimat ke-j dan l_j adalah banyak kata dalam kalimat ke-j, maka fitur kalimat yang mengandung entity kalimat ke-j dapat dihitung dengan cara:

$$f_{4j} = \frac{n(E_j)}{l_j} \quad (4)$$

   e. 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 [6]. Jika n(Nj) adalah jumlah data numerik pada kalimat ke-j dan l_j adalah banyak kata dalam kalimat ke-j, maka fitur kalimat yang mengandung data numerik kalimat ke-j dapat dihitung dengan cara:

$$f_{5j} = \frac{n(N_j)}{l_j} \quad (5)$$

   f. Panjang Kalimat (f6)
      Fitur ini bertujuan untuk menghilangkan kalimat-kalimat yang terlalu pendek [6]. Jika N adalah banyak kata dalam kalimat dan M adalah banyaknya kata dalam dokumen, maka fitur Panjang kalimat untuk kalimat ke-j dapat dihitung dengan cara

$$f_{6j} = \frac{N}{M} \quad (6)$$

   g. Koneksi antar kalimat (f7)
      Koneksi antar kalimat adalah banyaknya link dari suatu kalimat yang terhubung dengan kalimat yang lain. Jika n(Oj) adalah banyaknya link kalimat ke-j dengan kalimat yang lain dalam dokumen n(P) adalah jumlah koneksi antar kalimat dalam

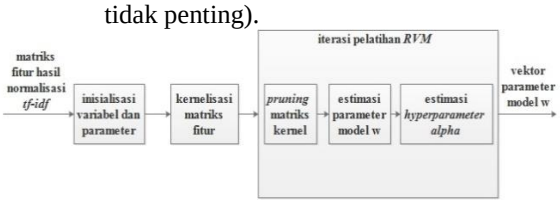
dokumen maka fitur koneksi antar kalimat ke-j dapat dihitung dengan cara:

$$f_{7j} = \frac{n(O_j)}{n(P)} \quad (7)$$

h. Penjumlahan bobot koneksi antar kalimat (f8)
Fitur ini berfungsi menjumlahkan kata yang sama antara kalimat ke-j dengan kalimat yang lain. Jika n(Ok_j) adalah banyaknya kata yang sama antara kalimat ke-j dengan kalimat lain didalam dokumen dan n(Pk) adalah jumlah semua kata yang sama antar kalimat, maka fitur penjumlahan bobot koneksi antar kalimat ke-j dapat dihitung dengan cara:

$$f_{8j} = \frac{n(Ok_j)}{n(Pk)} \quad (8)$$

4. Tahap berikutnya pelatihan RVM terhadap vector-vektor fitur ektraksi yang diperoleh dari tahap berikutnya. Pada tahap ini ditentukan model untuk digunakan dalam tahap pengujian. 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 8. 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 8. Proses Pelatihan *Relevance Vector Machine*

Berdasarkan Gambar 8, 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 [8]. 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* [9], dengan persamaan:

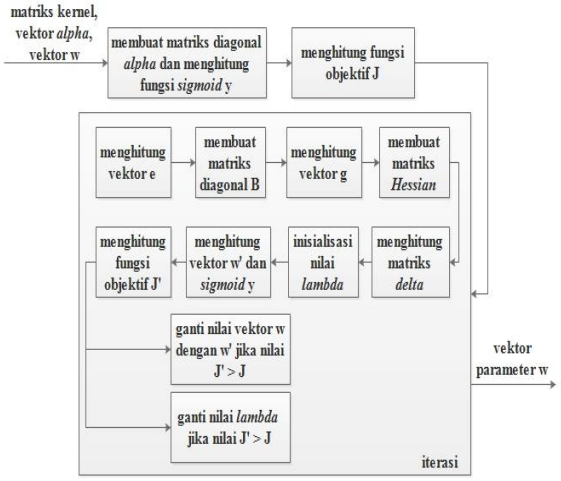
$$\mathrm{K}(\mathrm{x}, \mathrm{x}') = \exp(-\frac{||\mathrm{x} - \mathrm{x}'||^2}{2\sigma^2}) \quad (9)$$

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:

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

(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 9.



Gambar 9. 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.

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

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:

$$\mathbf{e} = \mathbf{t} - \mathbf{y} \qquad (12)$$

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:

$$\beta_j = y_j\,(1-y_j) \qquad (13)$$

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:

$$\mathbf{g} = \varphi_n^T e - \mathbf{A}w_n \qquad (14)$$

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:

$$\mathbf{H} = \boldsymbol{\varphi}_n^T\mathbf{B}\boldsymbol{\varphi}_n + \mathbf{A} \qquad (15)$$

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:

$$\Delta = \mathbf{H}^{-1}.\mathbf{g} \qquad (16)$$

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:

$$\mathbf{w}' = \mathbf{w}_n + \lambda\Delta \quad (17)$$

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:

$$\mathbf{y} = \frac{1}{1 + \exp(-(\mathbf{w}'.\varphi(\mathbf{x})))} \quad (18)$$

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

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

$$J' = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} + \frac{1}{2}\mathbf{w}'^{T}\mathbf{A}\mathbf{w}' \quad (19)$$

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**.

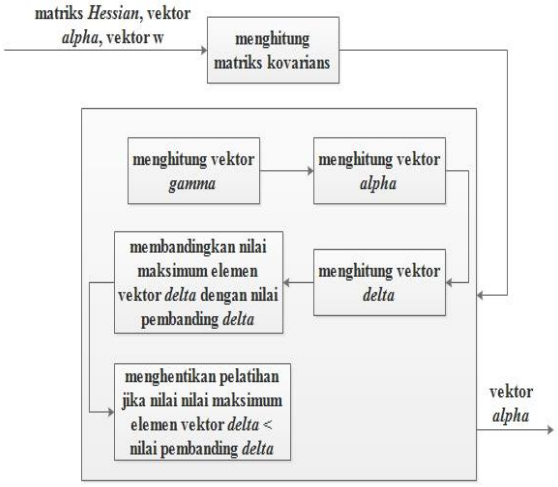
4) Selanjutnya bandingkan apakah hasil perhitungan J' dengan nilai fungsi objektif persamaan (10) sebelumnya. Apabila nilai J' lebih dari nilai sebelumnya, maka nilai tersebut dianggap minimum dan ganti vektor parameter **w** dengan nilai yang baru serta beri nilai *lambda* dengan 0 agar iterasi *update* 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 9.
Langkah pertama yang dilakukan adalah membuat matriks kovarians (*covariance*) **Σ**, dengan menggunakan persamaan:

$$\mathbf{\Sigma} = \mathbf{H}^{-1} \quad (20)$$

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 10. Proses Estimasi Hyperparameter Alpha

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

$$\gamma_j = 1 - \alpha_j\Sigma_{jj} \quad (21)$$

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:

$$\alpha_j = \begin{cases} \dfrac{\gamma_j}{w_j^2} & , i < \dfrac{i_{max}}{2} \\ \gamma_j\left(\dfrac{w_j^2}{\gamma_j} - \Sigma_{jj}\right)^{-1} & i \geq \dfrac{i_{max}}{2} \end{cases} \quad (22)$$

dimana γ_i merupakan elemen vektor $\boldsymbol{\gamma}$, $\mathbf{w}_i$ 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

$$\delta_j = \left|\ln(\alpha_j) - \ln(\alpha_j^{lama})\right| \qquad (23)$$

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.

5. Dokumen/data uji akan masuk ke proses yang sama yang dilalui oleh data latih, tetapi pada tahap pengujian, hanya menggunakan model hasil proses dari pelatihan RVM. 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 12.

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:

$$y(x; w) = \sum_{i=1}^{M} \mathbf{w}^T \varphi(\mathbf{x}) + w_0 \qquad (24)$$

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:

$$\sigma(y) = \frac{1}{1 + \exp(-(y))} \qquad (25)$$

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.

6. Kalimat-kalimat per tema dihitung bobot kemunculan katanya menggunakan TF-IDF kemudian metode *cosine* similarity untuk menyusunnya. Pertama dilakuakn dulu pembobotan token pada kalimat. Tahap pertama menghitung frekuensi kemunculan *term* di dalam dokumen (*tf*) dan perhitungan inverse 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* [10], kemudian bobot term dinormalisasi, sehingga persamaan yang digunakan adalah

$$w_{i,j} = \frac{tf_{i,j} \cdot \log \frac{N}{n_j}}{\sqrt{\sum_{s=1}^{k} (tf_{i,s} \cdot \log \frac{N}{n_j})^2}} \quad (26)$$

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* [11] dengan persamaan

$$sim(K1, Ki) = \frac{\sum(K1.Ki)}{\sqrt{\sum K1^2}\sqrt{\sum Ki^2}} \quad (27)$$

Proses ini akan menghasilkan ringkasan hasil dari pengurutan.

2.2 Hasil dan Pembahasan

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 [6]. Dimana perhitungannya mengugunakan rumus:

$$recall = \frac{tp}{tp + fn} \quad (28)$$

$$precission = \frac{tp}{tp + fp} \quad (29)$$

$$f - measure = \frac{2 * \text{precission} * \text{recall}}{recall + precission} \quad (30)$$

$$akurasi = \frac{tp + tn}{tp + fp + fn + tn} \quad (31)$$

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
Data yang digunakan untuk pelatihan sebanyak 4, sedangkan untuk pengujian digunakan sebanyak 7, berikut adalah perincian dari data latih dan data uji yang dapat dilihat pada tabel 1 berikut

Tabel 1. Data yang digunaan dalam penelitian

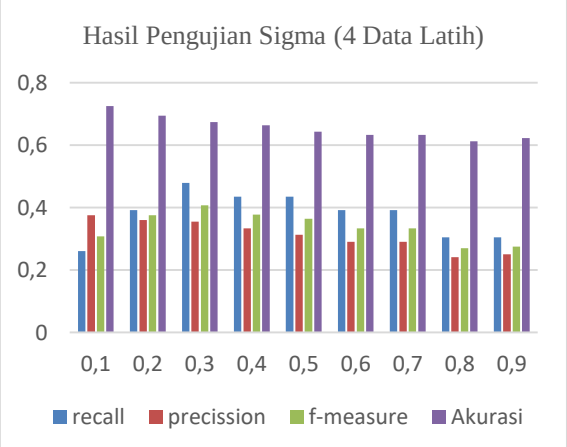
Proses	Jumlah	Jumlah	Jumlah
	Artikel Berita yang digunakan	**kalimat Penting**	**kalimat tidak penting**
Pelatian	4	16	28
Pengujian	7	23	75

Didalam RVM terdapat beberapa parameter yang dapat mempengaruhi hasil dari proses klasifikasi. Pada tabel 2 diperlihatkan pengaruh sigma terhadap peringkasan dan dapat dilihat dengan menggunakan sigma 0,3 diperoleh nilai *recall* terbesar.

Tabel 2. 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*
Jika data tersebut digambarkan dalam grafik, maka gambar 13 memperlihatkan hasil perhitungan recall, precission, f-measure dan akurasi.



Gambar 13 Hasil Pengujian Sigma (4 Data Latih)

Dengan menggunakan hasil pengujian yang telah dilakukan digunakan nilai sigma sebesar 0,3 kedalam pengujian, menurut M.E. Tipping, RVM dapat bekerja dengan data latih yang lebih kecil dibanding SVM [4]. Pengaruh data latih terhadap pengujian dapat dilihat pada tabel 2. Pada table 2 tersebut dapat dilihat dengan datalatih sebanyak 4 diperoleh recall terbesar.

Tabel 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*

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] J. Manuel dan T. Moreno, Automatic Text Summarization, Great Britain: ISTE Ltd and John Wiley & Sons, Inc., 2014.

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

[3] D. Fitriaman dan M. K. R. B. T. Leylia, "docplayer," [Online]. Available: https://docplayer.info/36453812-Peringkasan-teks-otomatis-berita-berbahasa-indonesia-pada-multi-document-menggunakan-metode-support-vector-machines-svm.html. [Diakses 9 Januari 2016].

[4] M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Mach. Learn. Res.*, vol. I, pp. 211-244, 2001.

[5] C. Bishop dan M. E. Tipping, "Variational Relevance Vector Machine," dalam *Morgan Kaufmann Publisher Inc.*, San Francisco, 2000.

[6] Z. Zulkifli, A. Wibowo dan G. Septiana, "Pembobotan Fitur Ektraksi Pada Peringkasan Teks Bahasa Indonesia Menggunakan Algoritma Genetika," dalam *eProceeding of Engineering 2.2*, Bandung, 2015.

[7] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indoensia," Universiteit van Amsterdam, Netherlands, 2003.

[8] X. Jianxiong, "jmlr.org," [Online]. Available: www.jmlr.org/papers/volume1/tipping01a/tipping01a.ps. [Diakses 12 1 2016].

[9] W. X. Z. L. W. &. Z. X. Wang, "Determination of the Spread Parameter in the Gaussian Kernel for Classification and Regression," *Neurocomputing*, vol. 55, no. 3-4, pp. 643-663, 2003.

[10] K. Ghag dan K. Shah, "SentiTFIDF-Sentiment Classification Using Relative Term Frequency Inverse Document Frequency," *International Journal of Advance Computer Science and Applications*, vol. 5, no. 2, pp. 36-43, 2014.

[11] K. Sarkar, K. Saraf dan G. A, "Improving Graph Based Multidocument Text Summarization Using an Enhanced Sentence Similarity Measure," dalam *IEEE 2nd International Conference on Recent Trends in Information System (ReTIS)*, Kolkata, 2015.