IMPLEMENTASI SUPPORT VECTOR MACHINE (SVM) DAN MAXIMAL MARGINAL IMPORTANCE(MMI) UNTUK PERINGKASAN MULTI DOKUMEN ARTIKEL BERBAHASA INDONESIA

Novriana

Teknik Informatika – Universitas Komputer Indonesia
Jalan Dipati Ukur 114 Bandung
E-mail : Novriana@hotmail.com

ABSTRAK

Peringkasan teks otomatis atau *automatic text summarization* merupakan bagian dari NLP (*natural language processing*) yang memungkinkan sebuah aplikasi dapat meringkas secara otomatis. Pada penelitian sebelumnya, metode *support vector machine* (SVM) dan *maximum marginal relevance* (MMR) digunakan untuk melakukan peringkasan teks otomatis pada multi dokumen, dimana metode SVM digunakan untuk proses klasifikasi kandidat kalimat ringkasan dan metode MMR dilakukan untuk melakukan seleksi kalimat hasil ringkasan. Hasil pengujian dari penelitian tersebut didapat nilai parameter terbaik klasifikasi SVM dengan kernel RBF yakni nilai C=4.0 dan γ=512.0. Nilai performansi f-measure didapat sebesar 0.771. Disamping itu dilakukan penelitian mengenai *maximal marginal importance* (MMI) yang merupakan modifikasi dari metode MMR untuk melakukan peringkasan teks otomatis pada single dokumen. Dari penelitian ini didapat nilai threshold yang optimal untuk metode MMI adalah 0.1. Oleh karena itu, dalam penelitian ini akan mengimplentasikan metode SVM dan MMI pada peringkasan teks otomatis untuk mengetahui seberapa baik penggunaan kedua metode tersebut.
Pada penelitian ini peringkasan teks otomatis dibagi menjadi empat tahap utama yaitu: *preprocessing*, ekstraksi fitur, klasifikasi kalimat menggunakan SVM, dan pembentukan ringkasan menggunakan MMI.
Dari pengujian nilai performansi, didapat nilai *recall* sebesar 55.2%, nilai *precision* sebesar 56% dan *f-measure* sebesar 53.65% .

Kata kunci : *automatic text summarization*, support vector machine, *maximal marginal importance*, ekstraksi fitur, multi dokumen

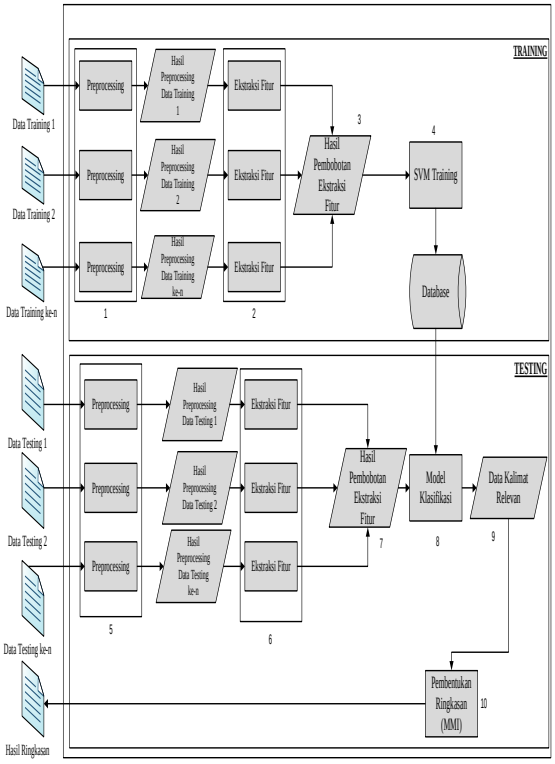
1. PENDAHULUAN

Peringkasan teks otomatis atau *automatic text summarization* merupakan bagian dari NLP (*natural language processing*) yang memungkinkan sebuah aplikasi dapat meringkas secara otomatis. Peringkasan teks otomatis dapat dilakukan dengan metode perangkingan yang dilakukan sebagai proses penyeleksian kalimat pada teks dengan cara memberikan bobot nilai pada kalimat-kalimat tersebut sehingga kalimat yang mengandung informasi penting akan menempati urutan teratas dengan nilai bobot tertinggi dan nilai dipilih untuk dimasukkan sebagai kandidat kalimat ringkasan [1]. Pada penelitian [2] dilakukan peringkasan teks otomatis untuk multi dokumen dengan cara mengklasifikasikan kalimat-kalimat dalam setiap artikel ke dalam kelas relevan (termasuk ke dalam ringkasan) dan kelas non-relevan (tidak termasuk ke dalam ringkasan) menggunakan SVM *classifier*. Selanjutnya dipilih n kalimat dari kelas positif berdasarkan nilai relevansinya yang dapat berupa nilai probabilitas kemunculan kelas relevan atau nilai *marginal relevance* yang dihasilkan melalui MMR. Penelitian tersebut berhasil dilakukan dengan nilai parameter terbaik pada parameter C = 4.0 dan γ = 512.0, dan nilai f-measure yang diperoleh adalah sebesar 0.771. Penelitian [1] menggunakan metode *maximal marginal importance* (MMI) yang merupakan modifikasi dari metode *maximum marginal relevance* (MMR), dalam penelitian tersebut mengatakan bahwa metode *maximal marginal importance* dipilih karena melihat kekurangan pada *maximum marginal relevance* yang merupakan metode *query based* dimana dalam menentukan tingkat pentingnya suatu kalimat adalah dengan melihat kemiripan antara kalimat tersebut dengan *query* yang diinputkan user. Dari penelitian ini didapat nilai threshold yang optimal untuk metode MMI adalah 0.1. Berdasarkan uraian di atas maka akan dilakukan peringkas teks otomatis menggunakan metode *support vector machine* (SVM) dan *maximal marginal importance* (MMI) untuk melihat sejauh mana metode tersebut bisa menghasilkan ringkasan yang baik.

2. MODEL ANALISA DAN DESAIN IMPLEMENTASI

Untuk dapat mengkaji implementasi metode yang digunakan, maka sistem yang akan dibangun akan dibagi ke dalam sub-sub sistem, dimana proses dalam menghasilkan rangkuman akan

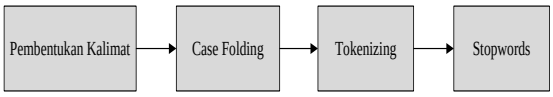
diuraikan dan diidentifikasi menjadi beberapa tahapan dan proses. Masukan dari sistem adalah artikel berita berbahasa Indonesia dengan topik sejenis. Keluaran dari sistem adalah ringkasan yang sudah dilakukan ekstraksi (*compression rate*). Sistem ini memiliki dua proses utama yaitu proses training dan proses testing. Dimana pada proses training metode SVM akan mencari model klasifikas yang nantinya akan digunakan pada proses testing. Adapun gambaran umum dari sistem yang akan dibuat bisa dilihat pada gambar 1. berikut:



Gambar 1 Gambaran Umum Sistem

2.1. Preprocessing

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. *Text preprocessing* yang digunakan pada penelitian ini menggunakan proses pembentukan kalimat, *case folding*, *tokenizing* dan *stopwords*. Tahap *preprocessing* dapat dilihat pada gambar 2.



Gambar 2 Tahap Preprocessing

Contoh teks yang dijadikan data masukan dapat dilihat pada tabel 1 berikut:

Tabel 1 Contoh Data Masukan

Jessica Wongso Divonis 20 Tahun Penjara

Majelis hakim Pengadilan Negeri Jakarta Pusat telah membacakan vonis atas kasus pembunuhan berencana Wayan Mirna Salihin. Terdakwa tunggal Jessica Kumala Wongso dijatuhi vonis 20 tahun penjara. "Menyatakan Jessica Kumala alias Jessica Kumala Wongso atau Jess telah terbukti sah dan meyakinkan bersalah melakukan pembunuhan berencana dan menjatuhkan pidana selama 20 tahun penjara dipotong masa tahanan dan menetapkan terdakwa tetap ditahan," ujar Ketua Majelis Hakim Kusworo di PN Jakarta Pusat, Kamis (27/10/2016). Vonis hakim ini sama dengan tuntutan jaksa penuntut umum (JPU).

2.1.1. Pembentukan Kalimat

Pembentukan kalimat yaitu proses membentuk string teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Dalam proses untuk mengubah dokumen menjadi kalimat-kalimat, digunakan fungsi split(), dengan tanda titik ".", tanda tanya "?", dan tanda seru "!" sebagai delimiter untuk memotong string dokumen [3]. Setelah dilakukan pembentukan kalimat, maka data masukan pada tabel 1 akan menjadi kalimat-kalimat yang bisa dilihat pada tabel 2.

Tabel 2 Hasil Pembentukan Kalimat

S	Kalimat
1	Majelis hakim Pengadilan Negeri Jakarta Pusat telah membacakan vonis atas kasus pembunuhan berencana Wayan Mirna Salihin.
2	Terdakwa tunggal Jessica Kumala Wongso dijatuhi vonis 20 tahun penjara.
3	"Menyatakan Jessica Kumala alias Jessica Kumala Wongso atau Jess telah terbukti sah dan meyakinkan bersalah melakukan pembunuhan berencana dan menjatuhkan pidana selama 20 tahun penjara dipotong masa tahanan dan menetapkan terdakwa tetap ditahan," ujar Ketua Majelis Hakim Kusworo di PN Jakarta Pusat, Kamis (27/10/2016).
4	Vonis hakim ini sama dengan tuntutan jaksa penuntut umum (JPU).

2.1.2. Case Folding

Case folding adalah proses pemrosesan teks dimana semua teks diubah ke dalam *case* yang sama. Pada kasus ini semua teks diubah representasinya ke dalam huruf kecil (*lowercase*) serta hanya menerima angka dan menghilangkan simbol-simbol. Setelah dilakukan *case folding* dari data hasil pembentukan kalimat pada tabel 2, maka hasil dari *case folding* dapat dilihat pada tabel 3 berikut:

Tabel 3 Hasil Case Folding

S	Kalimat
1	majelis hakim pengadilan negeri jakarta pusat telah membacakan vonis atas kasus pembunuhan berencana wayan mirna salihin
2	terdakwa tunggal jessica kumala wongso dijatuhi vonis 20 tahun penjara
3	menyatakan jessica kumala alias jessica kumala wongso atau jess telah terbukti sah dan meyakinkan bersalah melakukan pembunuhan berencana dan menjatuhkan pidana selama 20 tahun penjara dipotong masa tahanan dan menetapkan terdakwa tetap ditahan ujar ketua majelis hakim kusworo di pn jakarta pusat, kamis 27102016
4	vonis hakim ini sama dengan tuntutan jaksa penuntut umum jpu

2.1.3. Tokenizing

Pada proses *tokenizing* kalimat hasil *case folding* harus dipecah-pecah terlebih dahulu menjadi bagian-bagian yang lebih kecil berupa kata tunggal yang memiliki arti atau biasa disebut token. Setelah dilakukan tokenizing, maka data hasil case folding pada tabel 3 dapat dilihat pada tabel 4 berikut:

Tabel 4 Hasil Tokenizing

S	Kalimat	Token
1	majelis hakim pengadilan negeri jakarta pusat telah membacakan vonis atas kasus pembunuhan berencana wayan mirna salihin	“majelis”, “hakim”, “pengadilan”, “negeri”, “jakarta”, “pusat”, “telah”, “membacakan”, “vonis”, “atas”, “kasus”, “pembunuhan”, “berencana”, “wayan”, “mirna”, “salihin”.
2	terdakwa tunggal jessica kumala wongso dijatuhi vonis 20 tahun penjara	“terdakwa”, “tunggal”, “jessica”, “kumala”, “wongso”, “dijatuhi”, “vonis”, “20”, “tahun”, “penjara”
3	menyatakan jessica kumala alias jessica kumala wongso atau jess telah terbukti sah dan meyakinkan bersalah melakukan pembunuhan berencana dan menjatuhkan pidana selama 20 tahun penjara dipotong masa tahanan dan menetapkan terdakwa tetap ditahan ujar ketua majelis hakim kusworo di pn jakarta pusat, kamis 27102016	“menyatakan”, “jessica”, “kumala”, “alias”, “jessica”, “kumala”, “wongso”, “atau”, “jess”, “telah”, “terbukti”, “sah”, “dan”, “meyakinkan”, “bersalah”, ‘melakukan”, “pembunuhan”, “berencana”, “dan”, “menjatuhkan”, “pidana”, “selama”, “20”, “tahun”, “penjara”, “dipotong’, “masa”, ‘tahanan”, “dan”, “menetapkan”, “terdakwa”, “tetap”, “ditahan”, “ujar”, “ketua”, “majelis”, “hakim”, “kusworo”, “di”, ‘pn”, “jakarta”, “pusat”, “kamis”, “27102016”
4	vonis hakim ini sama dengan tuntutan jaksa penuntut umum jpu	“vonis”, “hakim”, “ini”, “sama”, “dengan”, “tuntutan”, “jaksa”, “penuntut”, “umum”, “jpu”

2.1.4. Stopwords

Text filtering bertujuan untuk mengambil kata-kata yang dapat merepresentasikan isi dokumen dengan cara membuang kata-kata yang dianggap tidak penting yang biasa disebut *stopword*. *Stopwords* dapat berupa kata sambung, kata depan, dan kata seru seperti “di”, “yang”, “dan”, “ke”, “wah”, “serta”, “wow” dan lain-lain [3]. Dalam penelitian ini digunakan daftar list stopwords untuk bahasa Indonesia [4]. Setelah dilakukan proses stopwords dari data pada tabel 4, maka hasil stopwords bisa dilihat pada tabel 5 berikut.

Tabel 5 Hasil Stopwords

S	Kalimat	Token
1	majelis hakim pengadilan negeri jakarta pusat telah membacakan vonis atas kasus pembunuhan berencana wayan	“majelis”, “hakim”, “pengadilan”, “negeri”, “jakarta”, “pusat”, “membacakan”, “vonis”,

	mirna salihin	“pembunuhan”, “berencana”, “wayan”, “mirna”, “salihin”.
2	terdakwa tunggal jessica kumala wongso dijatuhi vonis 20 tahun penjara	“terdakwa”, “tunggal”, “jessica”, “kumala”, “wongso”, “dijatuhi”, “vonis”, “20”, “tahun”, “penjara”
3	menyatakan jessica kumala alias jessica kumala wongso atau jess telah terbukti sah dan meyakinkan bersalah melakukan pembunuhan berencana dan menjatuhkan pidana selama 20 tahun penjara dipotong masa tahanan dan menetapkan terdakwa tetap ditahan ujar ketua majelis hakim kusworo di pn jakarta pusat, kamis 27102016	“jessica”, “kumala”, “alias”, “jessica”, “kumala”, “wongso”, “jess”, “terbukti”, “sah”, “meyakinkan”, “bersalah”, “pembunuhan”, “berencana”, “menjatuhkan”, “pidana”, “20”, “tahun”, “penjara”, “dipotong’, ‘tahanan”, “menetapkan”, “terdakwa”, “ditahan”, “ketua”, “majelis”, “hakim”, “kusworo”, ‘pn”, “jakarta”, “pusat”, “kamis”, “27102016”
4	vonis hakim ini sama dengan tuntutan jaksa penuntut umum jpu	“vonis”, “hakim”, “tuntutan”, “jaksa”, “penuntut”, “jpu”

2.2. Ekstraksi Fitur

Untuk menggunakan metode statistik, kalimat diwakili sebagai vektor fitur [5]. Fitur-fitur ini adalah atribut yang digunakan untuk mewakili data yang digunakan dalam tugasnya. Kata kunci digunakan untuk mengekstrak fitur, dimana fitur merupakan hasil ekstraksi yang diasumsikan memberikan informasi mengenai kalimat [2]. Pada penelitian ini digunakan 8 fitur, dimana fitur-fitur tersebut telah digunakan sebelumnya pada penelitian lain seperti pada penelitian [2], [5] dan [6]. Delapan fitur yang dijelaskan di bawah ini akan digunakan untuk proses klasifikasi menggunakan *support vector machine* (SVM).

2.2.1. Fitur Panjang Kalimat (F_1)

Fitur ini digunakan untuk menghitung panjang kalimat, dimana Fitur ini dihitung dengan membagi jumlah kata-kata dalam kalimat terhadap jumlah kata dari kalimat terpanjang pada setiap artikel[2]. Fitur panjang kalimat dihitung menggunakan persamaan berikut:

$$Scoref1(s_i) = \frac{jumlah\ kata\ yang\ terdapat\ pada\ kalimat}{jumlah\ kata\ yang\ terdapat\ pada\ kalimat\ terpanjang} \quad (1)$$

2.2.2. Fitur Posisi Kalimat (F_2)

Fitur ini merepresentasikan kalimat pertama dan terakhir pada setiap artikel hasil *preprocessing* merupakan kalimat yang penting [5], maka nilai fitur untuk setiap kalimat pertama dan terakhir pada setiap artikel akan bernilai 1 dan kalimat lain akan bernilai 0.

2.2.3. Fitur Data Numerik (F_3)

Biasanya kalimat yang mengandung data numerik merupakan kalimat penting dan biasanya kalimat tersebut masuk ke dalam ringkasan [2]. Fitur data numerik dihitung dengan menggunakan persamaan sebagai berikut :

$$Scoref3(s_i) = \frac{jumlah\ data\ numerik}{panjang\ kalimat\ (jumlah\ kata\ pada\ kalimat)} \quad (2)$$

2.2.4. Fitur Kata Thematik Dalam Kalimat (F_4)

Fitur ini menghitung kemunculan relatif kata kunci pada suatu kalimat, biasanya kalimat yang memiliki relatif kata kunci yang baik merupakan kalimat ringkasan [2]. Fitur kata-kata thematik dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref4(s_i) = \frac{jumlah\ kata\ kunci\ pada\ kalimat}{panjang\ kalimat\ (jumlah\ kata\ pada\ kalimat)} \quad (3)$$

2.2.5. F Fitur Kalimat Yang Menyerupai Judul (F_5)

Kalimat yang menyerupai judul adalah kalimat yang memiliki *vocabulary overlap* antara kalimat dengan judul [2]. Fitur kalimat yang menyerupai judul dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref5(s_i) = \frac{jumlah\ kata\ pada\ judul}{jumlah\ kata\ yang\ sama\ pada\ judul} \quad (4)$$

2.2.6. Fitur Kemiripan Kalimat Dengan Kumpulan Kalimat Lain (F_6)

Kemiripan kalimat dapat dilihat dari *vocabulary overlap* antara kalimat dengan kalimat yang lain, untuk mempermudah maka kata yang dilihat hanyalah kata kunci [2]. Fitur kemiripan kalimat dengan kumpulan kalimat lain dihitung dengan menggunakan persamaan sebagai berikut:

$$Scoref6(s_i) = \frac{kata\ kunci\ dalam\ kalimat \cap kata\ kunci\ dalam\ kalimat\ lain}{kata\ kunci\ dalam\ kalimat \cup kata\ kunci\ dalam\ kalimat\ lain} \quad (5)$$

2.2.7. Fitur Ikatan Leksikal Dengan Kalimat Sebelumnya (F_7)

Ikatan leksikal antara kalimat dengan kalimat sebelumnya didefinisikan sebagai kata (*stem*) yang muncul dalam kedua kalimat tersebut, nilai akan 1 apabila memiliki hubungan leksikal, akan bernilai 0 jika tidak punya.

2.2.8. Fitur Ikatan Leksikal Dengan Kalimat Sesudahnya (F_8)

Ikatan leksikal antara kalimat dengan kalimat sesudahnya didefinisikan sebagai kata (*stem*) yang muncul dalam kedua kalimat tersebut, nilai 1 apabila memiliki hubungan leksikal, dan akan bernilai 0 jika tidak punya.

2.3. Support Vector Machine (SVM)

Metode klasifikasi yang banyak dikembangkan dan diterapkan adalah *support vector machine* (SVM) [6]. Ide dasar SVM adalah memaksimalkan batas hyperplane (*maximal margin hyperplane*) [6]. Dalam teknik ini, SVM akan berusaha untuk menemukan fungsi pemisah (*classifier*) yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda [7]. Secara matematika, formulasi problem optimisasi SVM untuk kasus klasifikasi linier di dalam *primal space* adalah:

$$min_{w,b} \frac{1}{2}\|w\|^2 \quad (6)$$

Syarat : $y_i(wx_i + b) \geq 1, i = 1, \dots, N$

Dimana x_i adalah input, y_i adalah output yang nilainya = 1 atau -1. w dan b adalah parameter yang akan dicari nilainya. Dalam persamaan (2.11), minimalkan fungsi tujuan (obyektif function). $\frac{1}{2}\|w\|^2$ atau memaksimalkan kuantitas $\frac{1}{\|w\|}$ atau $w^T w$ dengan memperhatikan pambatas :

$$y_i(wx_i + b) \geq 1 \quad (7)$$

Bila output data $y_i = +1$ maka pembatas menjadi :

$$(wx_i + b) \geq 1 \quad (8)$$

Sebaliknya jika $y_i = -1$ maka pembatas menjadi:

$$(wx_i + b) \leq -1 \quad (9)$$

Di dalam kasus yang tidak *feasible* (*infeasible*) dimana beberapa data mungkin tidak bisa dikelompokkan secara benar, formulasi matematikanya menjadi:

$$min_{w,b,t} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi i \quad (10)$$

Syarat : $y_i(wx_i + b) \geq 1, i = 1,2, \dots, N$

Dalam formulasi ini SVM akan berusaha meminimalkan kesalahan klasifikasi yang dinyatakan dengan adanya variabel *slack*. Penggunaan variabel *slack* adalah untuk mengatasi kasus ketidaklayakan (*infeasibility*) dari pembatas (*constraints*) pada persamaan (7) dengan cara memberikan pinalti dengan menerapkan parameter C. Vektor w tegak lurus terhadap fungsi pemisah :

$$wx + b = 0 \quad (11)$$

Konstanta b menentukan lokasi fungsi pemisah relatif terhadap titik asal. Persamaan (10) adalah programa nonlinier. Ini bisa dilihat dari fungsi tujuan yang berbentuk kuadrat. Untuk membuat masalah ini lebih mudah dan efisien untuk diselesaikan, masalah ini bisa ditransformasikan ke dalam *dual space*. Untuk itu, persamaan (10) menjadi fungsi Lagrangian:

$$max \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i\, y_j {x_i}^T x_j \quad (12)$$

Syarat : $\sum_{i=1}^{N} \alpha_i y_i = 0, i = 1, .., n$

Persamaan (12) adalah *Quadratic Programming* (*QP*) dengan pembatas (*constraint*) linier.

Banyak teknik data mining atau *machine learning* yang dikembangkan dengan asumsi kelinieran. Karena itu bila suatu klasifikasi memperlihatkan ketidaklinieran, algoritma *perceptron* tidak bisa mengatasinya. Metoda kernel adalah salah satu untuk mengatasinya [7]. Dengan memetakan setiap data ke *feature space*, maka persamaan (12) akan menjadi:

$$max \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j) \quad (13)$$

Syarat : $\sum_{i=1}^{N} \alpha_i y_i = 0, i = 1, .., n$

Dalam persamaan (13), *dot product* antara $\varphi(x_i)^T \varphi(x_j)$ bisa diwakili oleh $K(x_i, x_j)$ dimana K adalah fungsi kernel. Selanjutnya setelah SVM dilatih dan ditemukan nilai α dan b maka akan didapatkan persamaan :

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i\, K(x_i, x) + b \quad (14)$$

Fungsi pemisah optimal adalah :

$$g(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i\, K(x_i, x) + b\right) \quad (15)$$

Dimana $\alpha_i = 1, .., N$ adalah solusi optimal dari persamaan (13) dan b dipilih sehingga $y_i f(x_i) = 1$ untuk sembarang i dengan $C > \alpha_i > 0$. Data x_i dimana $\alpha_i > 0$ dinamakan *support vector* dan menyatakan data *training* yang diperlukan untuk mewakili fungsi keputusan yang optimal. Untuk mengatasi masalah ketidaklinieran yang sering terjadi dalam kasus nyata, maka bisa diterapkan metode kernel. Metode kernel yang digunakan dalam penelitian ini yaitu Radial Basis Function (RBF) dengan persamaan:

$$K(x, y) = \exp\left(-\gamma\|x - x_i\|^2\right) \quad (16)$$

2.4. Maximal Marginal Importance (MMI)

Maximal marginal importance (MMI) adalah modifikasi metode *maximal marginal relevance* (MMR) [1].

2.4.1. Pembobotan Unit atau Fitur Kalimat

Untuk mendapatkan unit-unit teks yang paling informatif, biasanya dilakukan pemberian

skor atau pembobotan nilai terhdap unit-unit teks yang dianggap penting tersebut. Bobot kalimat adalah nilai sebuah kalimat yang dikalkulasi dari nilai unit-unit teks di dalamnya yang mengindikasikan seberapa penting arti kalimat tersebut [1]. Fitur kalimat yang digunakan untuk melakukan proses pembentukan kalimat menggunakan MMI adalah sebagai berikut:

2.4.1.1. Word Sentence Score (WSS)

WSS adalah formula pemberian bobot nilai yang menerapkan metode TF-ISF (*Term Frequency- Inverse Sentence Frequency*). Untuk menghitung nilai sutu kata (term) pada suatu kalimat teks sumber dapat dihitung menggunakan persamaan berikut:

$$w_{ij} = tf_{ij} \times isf = tf(t_{ij}, s_i)\left[1 - \frac{\log(sf(t_{ij})+1}{\log(n+1)}\right] \quad (17)$$

Keterangan :

w_{ij} = bobot nilai suatu kata /*term* t_{ij} dalam suatu kalimat teks sumber

t_{ij} = suatu *term* pada kalimat teks sumber

$tf(t_{ij}, s_i)$= frekuensi kemunculan *term* t_{ij} dalam kalimat s_i

$sf(t_{ij})$ = jumlah kalimat yang mengandung *term* t_{ij}

n = jumlah kalimat keseluruhan kalimat dalam teks sumber

Setelah didapat bobot nilai term t_{ij} pada kalimat s_i, maka kalkulasi bobot nilai kalimat (WSS) s_i tersebut dengan persamaan berikut:

$$WSS(s_i) = 0.1 + \frac{\sum tj \in s_i w_{ij}}{HTFS}$$, jumlah kalimat yang mengandung $t_j \geq \frac{1}{2}LS$ (18)

Keterangan :

$WSS(s_i)$= bobot nilai kalimat s_i dalam teks sumber berdasarkan frekuensi *term*.

LS = jumlah/panjang kalimat ringkasan

t_j = term pada kalimat s_i yang memenuhi syarat yaitu kalimat dalam teks sumber yang mengandung lebih dari 1 atau = LS.

w_{ij} = bobot nilai suatu term t_{ij} pada kalimat s_i

HTFS = nilai w_{ij} paling besar yang terpilih dari keseluruhan w_{ij} term yang terdapat pada kalimat s_i.

Bobot nilai 0,1 merupakan nilai minimum yang didapatkan sebuah kalimat jika term-term pada kalimat tersebut tidak penting.

2.4.1.2. Keyword Feature

Sepuluh *term* dengan w_{ij} yang paling besar dari seluruh kalimat teks sumber dipilih menjadi *keyword* [1]. Kalimat yang mengandung *keyword* tersebut akan diberi nilai 0,1 setiap *keyword*nya.

2.4.1.3. NFriends Feature

$Nfriends(s_i, s_j)$ mengukur keterhubungan antara kalimat s_i dan s_j. $s_i\, friends$ ditentukan dengan menghitung nilai *similarity* antar s_i dan kalimat lainnya yang terdapat pada teks sumber (s_w). Jika nilai similarity $(s_i, s_w) \geq threshold$, maka s_w merupakan s_i *friends.* Dan untuk menghitung nilai *Nfriends* antara kalimat s_i dan s_j digunakan persamaan berikut:

$$Nfriends(s_i, s_j) = \frac{s_i(friends) \cap s_j(friends)}{|s_i(friends) \cup s_j(friends)|}$$, dimana $i \neq j$ (19)

2.4.1.4. Ngrams Feature

Ngrams (s_i, s_j) merupakan suatu metode yang menghitung nilai kesamaan antara n kalimat s_i dan s_j. Ngrams melakukan pemisahan per n kata dari suatu kalimat. Untuk menghitung kesamaan *Ngrams* antara kalimat s_i dan s_j digunakan persamaan berikut:

$$Ngrams(s_i, s_j) = \frac{s_i(ngrams) \cap s_j(ngrams)}{|s_i(ngrams) \cup s_j(ngrams)|}$$, dimana $i \neq j$ (20)

2.4.1.5. The Similarity to First Sentence (sim_fsd)

Fitur ini memberikan nilai kepada kalimat berdasarkan nilai similarity kalimat tersebut dengan kalimat pertama pada teks sumber. Karena pada artikel berita kalimat pertama dianggap kalimat penting.

$$simFSD(s_i) = sim(s_i, s_0) \quad (21)$$

Dimana s_0 merupakan kalimat pertama pada teks sumber.

2.4.1.6. Sentence Centrality (SC)

Fitur ini dihitung berdasarkan 3 faktor, yaitu *similarity*, fitur *Nfriends*, dan fitur *Ngrams*. Nilai similarity dihitung menggunakan persamaan 22 berikut:

$$sim(s_i, s_j) = \frac{\sum w_i \in s_i, s_j\, tf(w_i, s_i) tf(w_i, s_j)\left[1 - \frac{\log(sf(w_i)+1)}{\log n+1}\right]^2}{\sqrt{\sum w_i \in s_i (tf(w_i, s_i).\left[1 - \frac{\log(sf(w_i)+1)}{\log n+1}\right]^2} \times \sqrt{\sum w_i \in s_j tf(w_i, s_j).\left[1 - \frac{\log(sf(w_i)+1)}{\log n+1}\right]^2}}$$

(22)

Sentence centrality dihitung menggunakan persamaan berikut:

$$SC(s_i) = \frac{\sum_{j=1}^{n-1} sim(s_i, s_j) + \sum_{j=1}^{n-1} nfriends(s_i, s_j) + \sum_{j=1}^{n-1} ngrams(s_i, s_j)}{n-1}$$

, dimana $i \neq j\ and\ sim(s_i, s_j) \geq 0$ (23)

Keterangan :

$SC(s_i)$ = bobot nilai SC kalimat s_i dalam teks sumber

$sim(s_i, s_j)$ = nilai *similarity* antara kalimat s_i dan s_j

$Nfriends(s_i, s_j)$ = nilai *nfriends* antara kalimat s_i dan s_j

$Ngrams(s_i, s_j)$ = nilai *ngrams* antara kalimat s_i dan s_j

0 = *threshold*

n = jumlah kalimat

2.4.1.7. Tittle Help Sentence (THS)

THS adalah suatu formula untuk menghitung nilai kesamaan ngrams antara kalimat s_i dengan judul teks. THS dihitung menggunakan persamaan berikut:

$$THS(s_i) = \frac{s_i(ngrams) \cap T(ngrams)}{|s_i(ngrams) \cup T(ngrams)|} \quad (24)$$

2.4.1.8. Tittle Help Sentence Relevance Sentence (THSRS)

Fitur ini digunakan untuk menghitung nilai kesamaan ngrams antara kalimat s_i dengan kalimat yang mempunyai THS paling besar yang telah dihitung sebelumnya menggunakan persamaan (24). THSRS dihitung menggunakan persamaan:

$$THSRS(s_i) = \frac{s_j(ngrams) \cap THS(s_j(ngrams))}{|s_i(ngrams) \cup THS(s_j(ngrams))|} \quad (25)$$

Kemudian dihitung juga nilai SS_NG suatu kalimat s_i yang merupakan nilai rata-rata dari nilai $THS(s_i)$ dan $THSRS(s_i)$ menggunakan persamaan berikut:

$$SS_NG_{(s_i)} = \frac{THS(s_i)+THSRS(s_i)}{2} \quad (26)$$

2.4.2. Bobot Nilai Importance dan Relevance Kalimat

Bobot nilai *importance* adalah nilai kepentingan suatu kalimat [1]. Nilai *importance* suatu kalimat dihitung menggunakan persamaan berikut:

$$IMPR(s_i) = avg(WSS(s_i) + SC(s_i) + SS_{NG(s_i)} + simFSD(s_i) + kwrd(s_i)$$

(27)

Keterangan:

$WSS(s_i)$= bobot nilai kalimat s_i dalam teks sumber berdasarkan frekuensi *term*.

$SC(s_i)$ = bobot nilai SC kalimat s_i dalam teks sumber

$SS_NG_{(s_i)}$= bobot nilai SS_NG kalimat s_i dalam teks sumber

$simFSD(s_i)$= bobot nilai simFSD kalimat s_i dalam teks sumber

$kwrd(s_i)$= bobot nilai keyword kalimat s_i dalam teks sumber

Bobot nilai *relevance* adalah nilai keterhubungan suatu kalimat dengan klaimat lainnya yang terdapat pada teks sumber. Dihitung dengan menggunakan persamaan berikut:

$$Rel(s_i, s_j) = avg(Nfriends(s_i, s_j) + Ngrams(s_i, s_j) + sim(s_i, s_j))$$

(28)

Keterangan:

$sim(s_i, s_j)$ = nilai *similarity* antara kalimat s_i dan s_j

$Nfriends(s_i, s_j)$ = nilai *nfriends* antara kalimat s_i dan s_j

$Ngrams(s_i, s_j)$ = nilai *ngrams* antara kalimat s_i dan s_j

Sebelum dilakukan perhitungan *maximal marginal importance*, terlebih dahulu dicari nilai $ScoreBT(s_i)$ dengan menggunakan persamaan berikut:

$$ScoreBT(s_i) = impr(s_i) + (1 - (1 - impr(s_i) \times friendsNo(s_i)))$$

(29)

Keterangan:

$impr(s_i)$ = nilai importance kalimat s_i

$friendsNo(s_i)$ =jumlah friends pada kalimat s_i

Setelah didapatkan nilai kalimat $impr(s_i)$ nilai MMI dihitung menggunakan persamaan berikut:

$$MMI(s_i) = Arg\ \max_{si \in cs \setminus ss}[(ScoreBT(s_i) - \beta(s_i)) - max_{sj \in ss}(Rel(s_i, s_j))]$$

(30)

Keterangan:

s_i = kalimat indeks ke-i yang merupakan anggota kalimat ringkasan

s_j = kalimat indeks ke-j yang juga merupakan anggota kalimat ringkasan.

CS = kalimat kompetitif yang berada dalam 1 pohon biner

SS = kalimat-kalimat yang terpilih menjadi kalimat ringkasan

$Rel(s_i, s_j)$= nilai relevance antara kalimat s_i dan s_j

$\beta(s_i)$ = nilai penalty yang dinisialisasi 0.01*level kalimat pada pohon biner

3. PENGUJIAN

Pada tahap pengujian di uji 5 dokumen yang digunakan sebagai sampel data masukan dengan topik yang sama. Kemudian dipilih sejumlah n kalimat oleh *abstractor* berdasarkan pemampatan 30%. Sampel data masukan yang akan diuji dapat dilihat pada tabel berikut:

Table 6 Skenario Detail Pengujian

D	Nama File	Sumber	*Link*
1	Rio2016	BBC Indonesia	http://www.bbc.com/indonesia/olahraga/2016/08/160817_live_badminton
2	Rio2016 1	Liputan6	http://bola.liputan6.com/read/2579629/tontowi-liliyana-rebut-emas-olimpiade-rio

3	Rio2016 2	Okezone	http://sports.okezone.com/read/2016/08/18/40/1466341/liliyana-natsir-tontowi-ahmad-raih-emas-bagi-indonesia-di-olimpiade-rio-2016
4	Rio2016 3	Liga Olahraga	http://www.ligaolahraga.com/detail/20655/berita-badminton-kado-71-tahun-indonesia-merdeka-tontowi-liliyana-raih-emas-olimpiade-rio-2016/
5	Rio2016 4	Badminton Indonesia	http://badmintonindonesia.org/app/information/newsDetail.aspx?/5297

Pengujian nilai performansi dilakukan dengan menghitung nilai recall, precision dan f-measure. Adapun nilai ferformansi tersebut dihitung menggunakan persamaan berikut:

$$recall = \frac{\sum jumlah\ kalimat\ yang\ sama}{\sum kalimat\ ringkasan\ manual} \quad (31)$$

$$precision = \frac{\sum jumlah\ kalimat\ yang\ sama}{\sum kalimat\ ringkasan\ sistem} \quad (32)$$

$$kalimat\ yang\ sama = kalimat\ ringkasan\ sistem \cap kalimat\ ringkasan\ manual \quad (33)$$

$$f - measure = 2 \times \frac{recall \times precision}{precision + recall} \quad (34)$$

Dari pengujian yang dilakukan, didapatkan hasil ringkasan sistem yang dapat dilihat pada tabel berikut:

Table 7 Hasil Ringkasan Sistem

D	Jumlah Kalimat	Jumlah Ringkasan Sistem	Kalimat Ke-
1	10	4	1,2,4,10
2	16	3	8,10,13
3	9	5	1,2,3,7,8
4	21	4	1,2,7,10
5	19	7	1,3,4,5,6,7,13
	75	23	

Hasil ringkasan sistem akan dibandingkan dengan hasil ringkasan manual yang dibuat oleh *abstractor* yang dapat dilihat pada tabel berikut:

Table 8 Hasil Ringkasan Manual

D	Jumlah Kalimat	Jumlah Ringkasan Manual	Kalimat Ke-
1	10	2	1,9
2	16	2	8,12
3	9	6	2,3,4,6,7,8
4	21	6	1,2,3,7,10,13
5	19	7	1,2,4,5,13,18,19
	75	23	

Sehingga jika dilakukan perbandingan pada hasil ringkasan sistem pada tabel 4.2 maka akan diperoleh nilai performansi sebagai berikut:

Table 9 Hasil Perhtiungan Nilai Performansi

D	Recall	Precision	F-Measure
1	50%	25%	33.33%
2	50%	33%	39.76%
3	67%	80%	72.92%
4	67%	100%	80.23%
5	42%	42%	42%
Rata-Rata	55.2%	56%	53.65%

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat diketahui bahwa rangkuman yang dihasilkan oleh peringkas teks otomatis ini belum bisa dikatakan baik. Namun ringkasan yang dihasilkan menunjukkan bahwa peringkas teks otomatis yang dibangun ini menghasilkan kalimat hasil ringkasan yang kurang sesuai dengan hasil ringkasan manusia. Parameter C yang dipilih dari penelitian ini yaitu 1 dan nilai γ sebesar 0.5. Nilai performansi yang diperoleh yaitu nilai *recall* sebesar 55.2%%, nilai *precision* sebesar 56% dan nilai *f-measure* sebesar 53.65%.

5. DAFTAR PUSTAKA

[1] I. N. Sari, P. K. Angelinna and L. H. Imam, "Analisis dan Implementasi Peringkas Teks Otomatis Menggunakan Metode Maximal Marginal Importance(MMI)," 2011.

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

[3] Z. A. R. K. Mustaqhfiri M, "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance," 2011.

[4] T. F. Z, A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.

[5] E. N. H. S. Jamhari Muhammad, "Pengklusteran Dokumen Teks Hasil Peringkas Dokumen Otomatis Yang Menggunakan Metode Seleksi Fitur Dan Latent Sematic Analysis (LSA)," *Jurnal Teknologi Informasi*, *Volume 10 Nomor 1,* 2014.

[6] P. Eko, Data Mining Konsep dan Aplikasi Menggunakan Matlab, Yogyakarta: Andi, 2007.

[7] S. B, Data Mining Teknik Pemanfaatan Data untuk Keperluan Bisnis, 2007: Graha Ilmu, Yogyakarta.

[8] Aristoteles, Pembobotan Fitur Pada Peringkasan Teks Bahasa Indonesia Menggunakan Algoritme Genetika, 2011.

[9] Z. B. d. E. Winarko, "Analisis Fitur Kalimat untuk Peringkas Teks Otomatis pada Bahasa Indonesia," *ournals of Computing and Cybernetics Systems 5 (2);60-68*, 2011.