BAB 2
TINJAUAN PUSTAKA

1.1 Analisis Sentimen

Analisis sentimen atau disebut juga opinion mining adalah bidang studi untuk menganalisis pendapat, sentimen, evaluasi, penilaian sikap dan emosi terhadap entitas seperti produk, jasa, organisasi, individu, peristiwa, topik dan atribut lainnya. Analisis sentimen dan opinion mining berfokus kepada opini yang mengekspresikan sentimen positif dan negatif [8].

Analisis sentimen salah satu masalah penelitian yang populer baik dalam aplikasi kehidupan nyata atau dalam bidang akademik. Secara umum, analisis sentimen dapat dilakukan penelitian pada tiga level, yaitu :

1) Document Level
   Pada level ini adalah untuk mengklasifikasi apakah seluruh pendapat dalam satu dokumen mengekspresikan sentimen positif atau negatif.
2) Sentence Level
   Pada level ini adalah untuk menentukan apakah setiap kalimat menyatakan pendapat positif, negatif, atau netral. Netral biasanya tidak ada pendapat.
3) Entity dan Aspect Level
   Pada level ini adalah menentukan sentimen (positif atau negatif) berdasarkan target (aspek) dalam suatu kalimat. Analisis sentimen pada level ini cenderung lebih baik, karena seringkali analisis level dokumen atau level kalimat tidak menemukan apa yang sebenernya disukai dan tidak disukai pembuat opini.

1.1.1 Analisis Sentimen Berdasarkan Aspek

Analisis sentimen berdasarkan aspek adalah salah satu domain kasus *opinion mining*yang bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Klasifikasi teks opini di level dokumen atau kalimat seringkali tidak cukup karena level tersebut hanya bisa mengindentifikasi sentimen secara umum.Jika

diasumsikan sebuah dokumen atau kalimat memiliki sebuah sentimen positif, tidak berarti semua aspek yang berada di dokumen atau kalimat tersebut semuanya positif begitupun untuk dokumen atau kalimat yang mengandung sentimen negatif.Untuk analisis lebih lengkap perlu ditemukan aspek dan menentukan sentimen positif atau negatif pada setiap aspek

Ada 2 tugas utama untuk dalam melakukan analisis sentimen berdasarkan aspek antara lain:

1. *Aspect Extraction*

   Pada tahap ini mengekstrak aspek yang sudah ditentukan sebelumnya. Misalnya, dalam kalimat “kualitas suara ponsel ini luar biasa”, aspeknya adalah “kualitas suara” entitasnya “ponsel ini” . Pada kata “ponsel ini” tidak menunjukan aspek umum karena evaluasi bukan tentang ponsel secara kesuluruhan tetapi hanya tentang “kualitas suara”.
2. *Aspect Sentiment Classification*

   Pada tahap ini menentukan apakah pendapat dari berbagai aspek kedalam sentimen positif, negatif atau netral. Pada contoh kalimat “kualitas suara ponsel ini luar biasa” sentimen dari aspek “kualitas suara” adalah positif.

Tujuan analisis sentimen berdasarkan aspek adalah untuk mengidentifikasi aspek dari suatu entitas, dan sentimen yang diungkapkan oleh penulis komentar tentang aspek tersebut[9]. Misalnya, dari kalimat “makanan di restoran ini sangat enak tapi pelayanannya kurang bagus” pada kalimat tersebut kata “makanan” dan “pelayanan” diidentifikasikan sebagai aspek yang kemudian ditentukan sentimennya dengan kata “enak”, untuk aspek “makanan” dan mengandung sentimen positif sedangkan kata “kurang bagus”, untuk aspek “pelayanan” dan mengandung sentimen negatif. Pada tahap mengidentifikasikan aspek ada beberapa pendekatan yaitu, *frequency based*, *relation based*, *supervised learning*, dan *topic modelling* dan untuk penentuan sentimen terhadap aspek mempunyai 2 pendekatan yaitu, *supervised learning* dan *lexicon based* [8].

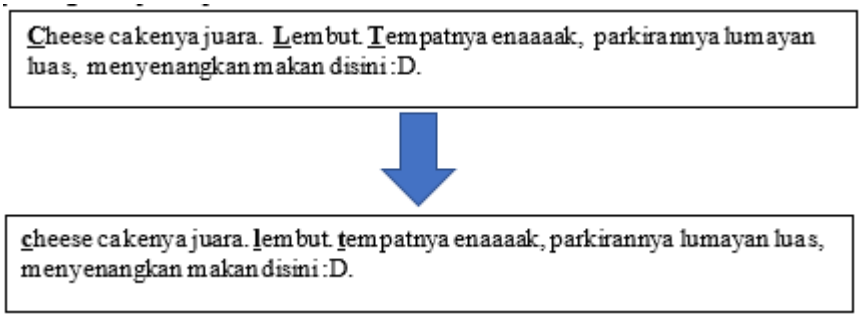
Pada penelitian ini algoritma yang digunakan untuk klasifikasi sentimen adalah *supervised learning* sehingga untuk mengembangkan sistem ABSA, dataset yang sudah dianotasi sangat penting untuk melakukan proses pelatihan. Dataset yang digunakan pada penelitian ini diambil dari penelitian [3]

1.1 Preprocessing

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Masukan awal pada proses ini adalah berupa dokumen teks. Dokumen teks yang akan dilakukan proses pada umumnya memiliki beberapa karakteristik, diantaranya adalah memiliki dimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Adapun tahapan dari *preprocessing* pada penelitian ini antara lain, *case folding, filtering, tokenization, word normalization* dan *stopword removal*.

1.1.1 Case Folding

Case folding merupakan proses mengubah semua huruf dalam dokumen teks menjadi huruf kecil. Contohnya adalah sebagai berikut.

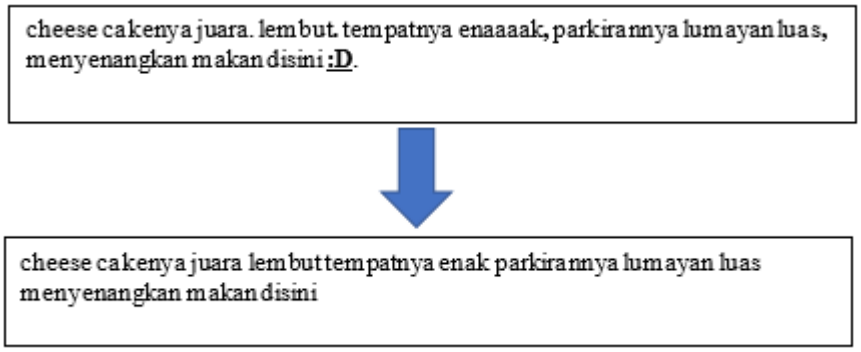


Gambar 2. 1 *Case Folding*

Setiap kata yang mengandung huruf kapital pada Gambar 2.1 diubah menjadi huruf kecil yaitu kata “Cheese”, “Lembut”, dan “Tempatnya” menjadi “cheese”, “lembut” dan “tempatnya”.

1.1.1 Filtering

Filtering merupakan proses menghilangkan karakter selain a-z, kecuali karakter pemecah kalimat, seperti spasi, tab, dan *newline* (lompat baris). Berikut adalah contoh dari *Filtering*.

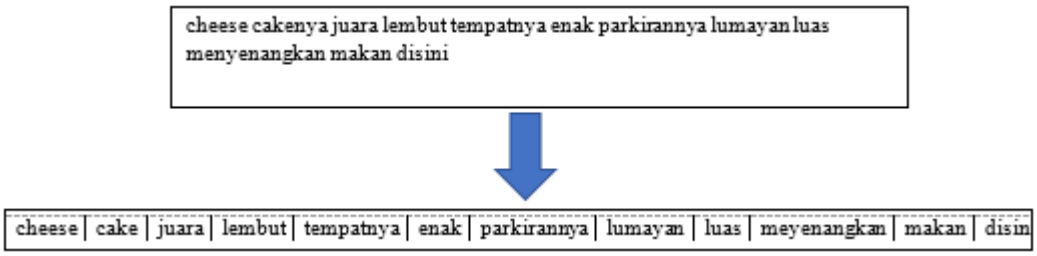


Gambar 2. 2 Filtering

Pada Gambar 2.2 semua tanda baca titik (.), koma (,) dan emoticon ":D" dihilangkan.

1.1.2 Tokenization

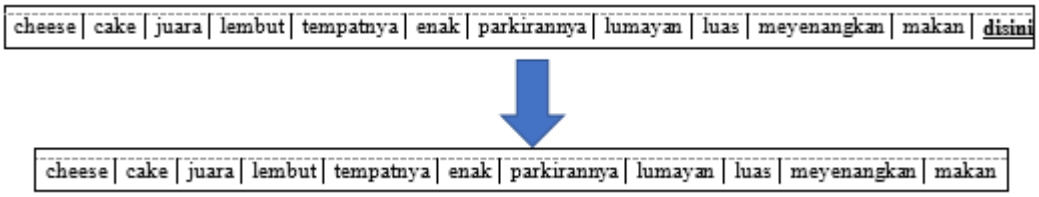
Tokenization adalah proses pemotongan string input berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan melihat pemisah seperti spasi, tab, dan newline (lompat baris). Berikut merupakan contoh dari Tokenizing.



Gambar 2. 3 Tokenization

1.1.1 Stopword Removal

Kebanyakan bahasa resmi di berbagai negara memiliki kata fungsi dan kata sambung seperti artikel dan preposisi yang hampir selalu muncul pada dokumen teks. Biasanya kata-kata ini tidak memiliki arti yang lebih di dalam memenuhi kebutuhan informasi. Kata-kata tersebut (contoh : *a*, *an*, *the*, dan *on* pada bahasa Inggris) disebut sebagai stopwords. Sebuah sistem text retrieval biasanya disertai dengan sebuah stoplist. Stoplist berisi sekumpulan kata yang 'tidak relevan', namun sering sekali muncul dalam sebuah dokumen. Dengan kata lain stoplist berisi sekumpulan stopwords. Berikut contoh dari *Stopword Removal*.



Gambar 2.4 Stopword Removal

1.2 *Term Frequency-Inverst Document Frequency (TF-IDF)*

Metode *Term Frequency-Inverse Document Frequency* (*tf-idf*) adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *term frequency* (*tf*) merupakan frekuensi kemunculan *termi* pada dokumen *j*.

$$tf(i) = \left(\frac{freq_i\,(d_j)}{\sum_{i=1}^{k} freq_i\,(d_j)}\right) \qquad (2.1)$$

Inverst Document frequency (df) adalah *invers* dari banyaknya kalimat dimana suatu *term i* muncul.

$$idf(i) = log_2\left(\frac{N}{n}\right) \qquad (2.2)$$

Dimana $tf_{i,j}$merupakan frekuensi kemunculan *termi* dalam dokumen *j*,*N*merupakan jumlah keseluruhan dokumen; dan *n* merupakan jumlah dokumen dimana *termj* muncul.

Frekuensi kemunculan *term* di dalam dokumen yang diberikan menunjukkan seberapa penting *term* itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung *term* tersebut menunjukkan seberapa umum *term* tersebut. Bobot *term* semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen. Pada algoritma *tf-idf* digunakan rumus untuk menghitung bobot *term* masing masing dokumen dengan rumus sebagai berikut[10]

$$tf.idf = tf(i) * idf(i) \tag{2.3}$$

Selain versi *tf-idf* di atas, terdapat pula versi *tf-idf* yang telah dinormalisasi. Versi *tf-idf* yang telah dinormalisasi ini sudah termasuk menghitung kesamaan (*similiarities*) dari dokumen. Adapun persamaan dari tf-idf yang telah dinormalisasi adalah sebagai berikut.

$$w_{i,j} = \frac{\mathrm{tf}_{i,j} \cdot \mathrm{idf}_j}{\| \vec{D}_i \|} = \frac{\mathrm{tf}_{i,j} \cdot \log_2 \frac{N}{n_j}}{\sqrt{\sum_{s=1}^{k} (\mathrm{tf}_{i,s} \cdot \log_2 \frac{N}{n_j})^2}} \tag{2.4}$$

Dimana $tf_{i,j}$merupakan frekuensi kemunculan *term i* dalam dokumen *j*; dan n_i merupakan jumlah dokumen dimana *termi* muncul.

1.1 *Relevance Vector Machine*

Dalam *supervised learning*, metode *machine learning* diberikan contoh sekumpulan vektor masukan $\{\boldsymbol{x_n}\}_{n=1}^{N}$ digunakan bersamaan dengan target yang bersesuaian $\{\boldsymbol{t_n}\}_{n=1}^{N}$, dimana pada dasarnya target merupakan *nilai sebenarnya* dalam hal regresi dan

label kelas sebenarnya pada prosesklasifikasi. Dari pelatihan memungkinkan ketergantungan pada input untuk tujuan prediksi yang akurat. Secara umum prediksi tersebut dapat dihitung dengan menggunakan persamaan y(x), yaitu:

$$y(x; w) = \sum_{i=1}^{M} w_i \phi_i(x) + w_0 = w^T \varphi(x) + w_0 \qquad 2.5)$$

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

Proses belajar atau pelatihan pada dasarnya merupakan sebuah proses untuk menentukan parameter dari fungsi y(x), dalam konteks ini menentukan parameter bobot $\mathbf{w} = (w_1, w_2, \dots, w_m)$. Untuk himpunan dari pasangan data latih $\{x_n, t_n\}_{n=1}^{N}$, tugas dari proses pelatihan ini adalah untuk mencari nilai dari bobot $\mathbf{w} = (w_1, w_2, \dots, w_m)^T$, sehingga fungsi y(x) dapat digeneralisasi secara cukup baik terhadap data yang baru, dan bobot yang dihasilkan memiliki elemen nilai bukan nol (*nonzero*) yang sedikit. Memiliki beberapa elemen nilai yang bukan nol dapat menghasilkan representasi vektor yang *sparse*, dengan keuntungan dapat menyediakan implementasi yang cepat [11].

Relevance Vector Machine adalah sebuah metode pembelajaran mesin yang diperkenalkan oleh Mike Tipping [11] pada tahun 2001, yang diadaptasi dari *Bayesian Framework* dan memiliki bentuk model fungsi yang mirip dengan *Support Vector Machine*. Sama halnya dengan *Support Vector Machine*, Relevance Vector Machine digunakan dalam proses klasifikasi dan regresi. *Relevance Vector Machine* diperkenalkan untuk menutupi beberapa kelemahan yang dimiliki oleh *Support Vector Machine*. Kelemahan tersebut antara lain :

1) Walaupun *Support Vector Machine* menghasilkan vektor yang relatif *sparse*, namun jumlah *Support Vector* yang ada akan secara linear bertambah seiring dengan ukuran himpunan data latih.
2) Prediksi yang tidak bersifat probabilistik, dan *Support Vector Machine* tidak cocok untuk proses klasifikasi dimana peluang *posterior* dari keanggotaan kelas dianggap penting.

1) *Support Vector Machine* memerlukan proses estimasi parameter *error/margin*C, yang secara umum memerlukan prosedur *cross-validation* yang dapat memperlambat proses komputasi.
2) Fungsi *kernel* yang digunakan dalam *Support Vector Machine* harus memenuhi kondisi *Mercer*.

Pada klasifikasi untuk dua label kelas (*binary classification*), semua target dapat diklasifikasikan kedalam dua kelas, yang dapat direpresentasikan dengan nilai 0 dan 1, seperti $\boldsymbol{t_n} \in \{\mathbf{0}, \mathbf{1}\}$. Distribusi *bernoulli* dapat diadopsi untuk peluang p(t|x) dalam *framework* probabilistik karena hanya dua nilai kelas yang mungkin, yaitu 0 dan 1. Dalam kasus klasifikasi, model prediksi mengambil bentuk kombinasi linear dari fungsi basis/kernel yang diubah oleh fungsi *logistic sigmoid*.

$$\mathrm{y}(\mathbf{x}, \mathbf{w}) = \sigma(\mathrm{x}; \mathrm{w}) = \sigma(\mathbf{w}^{\mathrm{T}}\varphi(\mathrm{x})) \tag{2.6}$$

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

$$\sigma(\mathrm{y}) = \frac{1}{1 + \exp(-\mathrm{y})} \tag{2.7}$$

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

$$\mathrm{p}(\mathrm{t}|\mathrm{w}) = \prod_{\mathrm{t}=1}^{\mathrm{N}} \sigma\{\mathrm{y}(\mathrm{x_n}; \mathbf{w})\}^{\mathrm{t_n}}[1 - \sigma\{\mathrm{y}(\mathrm{x_n}; \mathbf{w})\}^{1-\mathrm{t_n}} \tag{2.8}$$

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

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

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{t=1}^{N} \frac{\sqrt{\alpha_i}}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_i {w_i}^2}{2}\right) \quad (2.9)$$

dimana $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ merupakan *hyperparameter* yang diperkenalkan untuk mengontrol kekuatan dari *prior* terhadap parameter bobot yang diasosiasikannya. Untuk nilai α tertentu, distribusi *posterior* bobot terhadap data dapat dihitung menggunakan aturan *Bayes*, dengan persamaan

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) = \frac{p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha})} \quad (2.10)$$

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

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

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) &= \ln\{p(t|w)p(w|\alpha)\} + \ln p(\mathbf{t}|\alpha) \\ &= \sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} + \frac{1}{2} w^T A w \end{aligned} \quad (2.11)$$

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

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

$$\mathbf{H} = \nabla\nabla \ln p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) = \boldsymbol{\varphi}_n^T \mathbf{B} \boldsymbol{\varphi}_n + \mathbf{A} \quad (2.12)$$

dimana $\boldsymbol{\varphi}$merupakan matriks kernel, dan $\mathbf{B}$ = diag(β_1, β_2, …, β_n) merupakan matriks diagonal yang elemen matriksnya diperoleh dengan persamaan

$$\beta_n = \sigma\{y(x_n; \boldsymbol{w}')\}[1 - \sigma\{y(x_n; \boldsymbol{w}')\} \quad (2.13)$$

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

$$\mathbf{w}' = \mathbf{w}_n + \lambda\Delta \quad (2.14)$$

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

$$\Delta = \mathbf{H}^{-1}.\mathbf{g} \quad (2.15)$$

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

$$\mathbf{g} = -\nabla \ln p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}) = \varphi_n^T e - \mathbf{A} w_n \quad (2.16)$$

Setelah estimasi parameter bobot **w** dilakukan, selanjutnya *hyperparameter*α_j dilakukan reestimasi dengan menggunakan persamaan

$$\alpha_j^{new} = \frac{\gamma_j}{w_j^2} \quad (2.17)$$

dimana γ_j didefinisikan dengan persamaan

$$\gamma_j = 1 - \alpha_j \Sigma_{jj} \quad (2.18)$$

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

$$\boldsymbol{\Sigma} = -\left(\nabla\nabla \ln p\left(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}\right)\right)^{-1} = \mathbf{H}^{-1} \quad (2.19)$$

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

$$\alpha_j = \frac{\gamma_j}{w_j^2}; \text{jika } i < \frac{i_{max}}{2}$$
$$\alpha_j = \gamma_j \left(\frac{w_j^2}{\gamma_j} - \Sigma_{jj}\right)^{-1}; \text{jika } i \geq \frac{i_{max}}{2} \quad (2.20)$$

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

Pada saat proses optimasi, banyak *hyperparameter*$\alpha_j$yang akan memiliki nilai yang sangat besar, dan parameter bobot model yang berkoresponden dengan *hyperparameter* tersebut akan dipangkas (*pruned*), karena telah dalam keadaan *sparsity*. Proses optimasi terus berlanjut hingga perubahan maksimum dari $\alpha_j$dibawah koefisien tertentu atau jumlah maksimum iterasi telah dicapai.

1.1 Kernel

Kernel pada metode *Relevance Vector Machine* adalah pemisah antara kelas satu dengan kelas lainnya. Ada beberapa kernel yang dapat digunakan, diantaranya *Linear*, *Polinomial*, dan *Radial Basis Function(RBF)*. Dalam penelitian-penelitian sebelumnya, kernel *Linear* dan *RBF*lebih sering digunakan.

1. Linear

Linear, seperti namanya yang berarti garis lurus. Kernel linear menggunakan garis lurus sebagai pembatas/hyperplane antar kelas.

$$\mathrm{k}(x_i, x\ ) = x_i^T x \qquad 2.21$$

2. Polinomial

Kernel Polinomial memiliki dua parameter berbeda dari kernel lain. Parameter r adalah parameter bebas yang jika diisi r=0 maka disebut homogen. Sedangkan parameter d adalah derajat/quadrat yang umumnya diisi dengan d=2

$$\mathrm{k}(x_i, x\ ) = (y.\ \ x_i^T\ x\ \ + r)^{\ d} \qquad (2.22)$$

3. RBF

Dalam penyelesaiannya RBF membutuhkan parameter gamma dan C. Gamma berfungsi sebagai batas keputusan dan wilayah keputusan, sebagai contoh jika gamma bernilai kecil maka batas keputusan akan kecil namun wilayah keputusan akan

menjadi luas dan begitupun sebaliknya. C berfungsi sebagai penalti terhadap kesalahan dalam klasifikasi.

$$\exp(-\gamma||x_i - x||^2), \gamma > 0 \quad (2.23)$$

Pada Penelitian ini, kernel yang digunakan adalah kernel linear yang merupakan dasar dari kernel SVM.

1.1 Multi Class

Terdapat dua teknik multi class yang sering digunakan pada SVM yang dapat diterapkan juga pada RVM, yaitu *One Versus One* (OVO) dan *One Versus All* (OVA). OVA membandingkan satu dengan semua selain dirinya yang dianggap menjadi satu kesatuan. *Multiclass* ini digunakan karena sejatinya RVM adalah machine learning yang hanya mengklasifikan dua kelas saja secara linear.

Misalnya, terdapat 4 buah kelas dalam permasalahan tersebut maka model yang terbentuk mengacu pada Tabel 2.

Tabel 0. RVM biner metode *One Versus All*

$h_i = 1$	$h_1 = -1$	Hipotesis
Kelas 1	Bukan Kelas 1	$f^1(g) = (w^1)g + b^1$
Kelas 2	Bukan Kelas 2	$f^2(g) = (w^2)g + b^2$
Kelas 3	Bukan Kelas 3	$f^3(g) = (w^3)g + b^3$
Kelas 4	Bukan Kelas 4	$f^4(g) = (w^4)g + b^4$
Kelas 5	Bukan Kelas 5	$f^5(g) = (w^5)g + b^5$

Pada penelitian-penelitian sebelumnya, diketahui bahwa akurasi OVA lebih baik dibanding OVO walaupun dari segi kecepatan OVO lebih cepat. Maka dari itu, pada penelitian yang akan dilakukan mengenai penerapan RVM terhadap Analisis Sentimen Berdasarkan Aspek bahasa Indonesia ini menggunakan OVA. Dengan metode tersebut diharapkan dapat memberikan akurasi yang tinggi.

Dalam menerapkan metode OVA akan dibangun *z* buah model RVM biner, zadalah jumlah kelas. Dalam mengklasifikasikan hal tersebut dapat dilihat pada persamaan berikut.

$$Kelas\ g = arg\ \underset{r=1..z}{max}(\left(w^{(r)}\right)^{T}.\varphi(g) + b^{(r)}) \qquad (2.\ 20)$$

1.1 Pengujian Performa

Untuk mengukur seberapa bagus model agoritma yang telah dibuat maka dibutuhkan metode untuk mengukur performa algoritma . Berikut adalah penjelasan dari metode untuk mengukur performa algoritma dalam penelitian ini.

2.7.1. Akurasi

Akurasi salah satu metode yang digunakan untuk mengukur performa algoritma. Metode ini memberikan informasi mengenai jumlah persentase keberhasilan sebuah algoritma melakukan prediksi secara benar [20]. Adapun persamaan dari akurasi dapat dilihat pada persamaan (2.16)

2.7.2. F1 Score

Pada kasus klasifikasi *f1 score* bertujuan untuk mengetahui keseimbangan kinerja pada kelas minoritas dan mayoritas, oleh karena itu *f1 score* cocok untuk mengukur performa algoritma dalam hal data yang tidak seimbang [21]. Selain *f1 score* ada juga *precision* dan *recall*, precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem sedangkan *recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Pada penelitian ini aspek yang berlabel bukan sentimen lebih banyak dari label positif dan label negatif. Adapun persamaan *f1 score* dapat dilihat pada persamaan

$$recall = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Kelas\ X}$$

$$precision = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Prediksi\ terhadap\ Kelas\ X}$$

$$F_1\ score(X) = (2 \times \frac{(precision\ \times recall)}{(precision + recall)})$$

$$weighted\ F_1\ score = \frac{1}{\sum_i |X_i|} \times \sum_i |X|\ _1\ score(X_i)$$

Dimana:

$|X_i|$: jumlah anggota kelas X_i

Pengukuran kinerja klasifikasi pada data asli dan data hasil dari model klasifikasi ($F_{i,j}$) dilakukan dengan menggunakan tabulasi silang (matriks konfusi) yangberisi informasi tentang kelas data asli yang direpresentasikan pada baris matriksdan kelas data hasil prediksi suatu algoritma direpresentasikan pada kolom klasifikasi [6]. Berikut merupakan contoh matriks konfusi.

Tabel 2. 1 Matriks Konfusi Lima Kelas

Fij		Kelas Prediksi (j)				
		Kelas 1	Kelas 2	Kelas 3	Kelas 4	Kelas 5
Kelas Asli (i)	Kelas 1	F11	F12	F13	F14	F15
	Kelas 2	F21	F22	F23	F24	F25
	Kelas 3	F31	F32	F33	F34	F35
	Kelas 4	F41	F42	F43	F44	F45
	Kelas 5	F51	F52	F53	F54	F55

Menurut [10] ketepatan klasifikasi dapat dilihat dari akurasi klasifikasi. Akurasi klasifikasi menunjukkan performasi model klasifikasi secara keseluruhan, dimana semakin tinggi akurasi klasifikasi hal ini semakin baik performansi model klasifikasi.

$$Akurasi = \frac{Jumlahdatayangdiprediksisecarabenar}{Jumlahprediksiyangdilakukan}$$

$$= \frac{F_{11} + F_{22} + F_{33} + F_{44} + F_{11} + F_{55}}{F_{12} + F_{13} + F_{14} + F_{15} + F_{21} + F_{23} + F_{24} + F_{25} + F_{31} + F_{32} + F_{34} + F_{35} + F_{41} + F_{42} + F_{43} +}$$

1.1 Python

Python adalah bahasa pemrograman[15] yang menyediakan struktur tingkat tinggi seperti daftar dan susunan asosiatif(kamus), pengetikan dinamis dan pengikatan dinamis. Bahasa ini dirancang oleh Guidon Van Rossum pada tahun 1990, memiliki sintaks yang sederhana dan dapat dijalankan secara praktis. Programnya dikompilasi dengan cara menafsirkan kode byte setelah menginterpreter ke dalam platform independen.