Klasifikasi Informasi, Anjuran dan Larangan pada Hadits Shahih Bukhari menggunakan Metode *Support Vector Machine*.

Andina Kusumaningrum[1] , Said Al-Faraby[2] , Adiwijaya[3]

Prodi S1 Teknik Informatika, Telkom Informatics School, Universitas Telkom

[1]andinakusumaningrum@gmail.com, [2]saidalfaraby@telkomuniversity.ac.id, [3]adiwijaya@telkomuniversity.ac.id

Abstrak

Pertumbuhan informasi khususnya dalam bentuk data teks pada *media online* yang semakin pesat mendorong berbagai penelitian terkait dengan pengolahan data teks. Salah satu penelitian yang banyak dilakukan adalah klasifikasi teks dengan memanfaatkan *machine learning*. Namun, penelitian klasifikasi teks masih cukup jarang diterapkan untuk mengolah data teks kitab agama seperti hadits. Saat ini selain dibukukan, kumpulan data hadits dapat dengan mudah didapatkan baik dari internet maupun disajikan didalam aplikasi namun tidak disertai dengan informasi makna dari hadits tersebut. Sehingga, untuk masyarakat awam atau bahkan umat muslim yang sedang mempelajari hadits mungkin mengalami kesulitan dalam memahami makna dari hadits-hadits tersebut mengingat jumlah data hadits yang cukup banyak. Maka, pengolahan terhadap data teks hadits ini perlu dilakukan, karena didalam hadits terdapat berbagai pedoman yang bermanfaat untuk masyarakat khususnya umat muslim dalam berperilaku sesuai dengan *sunnah* Nabi Muhammad Shallallahu 'Alaihi Wasallam. Berdasarkan permasalahan tersebut, pada penelitian ini akan dilakukan analisis pada sebuah sistem klasifikasi teks secara otomatis yang dibangun terhadap salah satu kumpulan hadits terbaik yaitu Hadits Shahih Bukhari terjemahan Bahasa Indonesia ke dalam kategori Anjuran, Larangan, dan Informasi. Selain membangun *classifier* dengan menggunakan pendekatan *machine learning* yaitu metode *Support Vector Machine*, peneliti juga membandingkan dengan *classifier* yang dibangun dengan menggunakan pendekatan *simple rule-based system*. Hal itu dilakukan untuk membuktikan bahwa pendekatan *rule-based* memiliki performa yang kurang baik untuk mengklasifikasikan data teks hadits sehingga dibutuhkan pendekatan *machine learning*.

Kata Kunci : klasifikasi teks, *multiclass classification*, Hadits Shahih Bukhari, *Support Vector Machine*.

1. Pendahuluan

Kemajuan teknologi informasi yang terus berkembang semakin mmempermudah masyarakat dalam mendapatkan berbagai informasi khususnya di *media online*. Sebagian besar dari informasi yang didapatkan di internet adalah berupa data teks. Pertumbuhan data teks yang semakin pesat mendorong para peneliti membangun sebuah sistem untuk melakukan pengolahan terhadap kumpulan data teks tersebut. Salah satu teknik yang sering digunakan adalah klasifikasi teks. Teknik ini sudah banyak digunakan pada banyak penelitian dengan berbagai domain, namun masih sedikit yang melakukan penelitian dengan mengangkat domain teks kitab agama, salah satunya adalah Hadits. Hadits merupakan segala sabda, perbuatan, *taqrir*, dan hal-ihwal yang disandarkan kepada Nabi Muhammad Shallallahu 'Alaihi Wasallam, sehingga bagi umat muslim hadits dijadikan sebagai pedoman dan sumber hukum kedua setelah Al-Qur'an [1] [2]. Terdapat beberapa kumpulan hadits yang dianggap *shahih* dan dijadikan pedoman oleh umat muslim, salah satunya adalah Hadits Rasulullah yang diriwayatkan oleh Imam Bukhari. Bagi umat muslim, mengikuti dan mempraktekkan segala perbuatan Nabi Muhammad Shallallahu 'Alaihi Wasallam yang tertuang pada hadits dalam kehidupan sehari-hari merupakan salah satu *sunnah*. Namun, bagi orang awam ataupun orang muslim yang ingin belajar tentang hadits dan tidak memiliki waktu cukup, mungkin masih sedikit kesulitan untuk memahami satu persatu makna dari setiap hadits, mengingat jumlah hadits yang cukup banyak. Menurut aplikasi Lidwa jumlah data hadits Shahih Bukhari berjumlah kurang lebih 7.008 hadits termasuk hadits yang berulang [3]. Dalam pengklasifikasian hadits ini, penulis menemukan beberapa tantangan yaitu data teks hadits yang belum memiliki label dan masih mengandung *sanad*, sehingga proses awal yang dilakukan adalah memisahkan *sanad* secara manual dan melakukan *hand labeling* untuk menentukan kelas anjuran, larangan, dan informasi. Selain itu, tantangan lain yang dihadapi adalah adanya kalimat ambigu yang ada pada hadits, contohnya terdapat hadits yang merupakan kelas anjuran, namun juga memiliki fitur yang merepresentasikan kelas larangan. Selain itu, terdapat beberapa hadits yang ada di dalam kelas informasi mengandung fitur-fitur yang seharusnya ada pada kelas anjuran dan larangan. Hal itu mengakibatkan kesalahan pelabelan yang dilakukan oleh *classifier*. Sehingga, pada penelitian ini akan dilakukan analisis terhadap sistem klasifikasi yang dibangun untuk mengkategorikan data hadits ke dalam tiga kelas yaitu Anjuran, Larangan, dan Informasi. Oleh karena itu, *simple rule-based system* berdasarkan kemunculan fitur "-lah" dan beberapa kata penting lainnya tidak akan bekerja dengan maksimal. Untuk itu, penelitian ini memilih menggunakan teknik *machine learning* yaitu metode *Support Vector Machine*.

2. Tinjauan Pustaka

2.1 *Related Work*

Terdapat beberapa penelitian terkait dengan klasifikasi yang mengangkat konten data teks islami maupun domain teks lain dan memanfaatkan teknik *machine learning*. Salah satu penelitian terkait dengan klasifikasi yang pernah dilakukan adalah penelitian yang dilakukan oleh Untary N. Wisesty, M. Syahrul Mubarok, dan Adiwijaya pada tahun 2017 yaitu mengenai klasifikasi pelafalan huruf Hijaiyah menggunakan metode Hidden Markov Model yang

memanfaatkan *speech recognation*. Pemanfaatan *speech recognition* pada pengenalan huruf Hijaiyah ini adalah sistem akan mengolah sinyal suara yang menjadi inputan kemudian diubah menjadi data yang dapat dikenali oleh komputer. Sehingga sinyal suara yang diekstraksi kemudian dapat menghasilkan informasi yang dapat dianalisis untuk setiap variasi sinyal suara. Fitur ekstrasi digunakan untuk mengubah data menjadi nilai yang dapat digunakan untuk proses selanjutnya. Pada penelitian ini, metode fitur ekstraksi yang dibandingkan adalah LPC*(Linear Predictive Coding)* dan MFCC*(Mel Frequency Cepstral Coefficient).* Dari hasil penelitian tersebut, sistem yang menggunakan LPC menghasilkan akurasi performasi sebesar 96,1% sedangkan sistem yang menggunakan MFCC menghasilkan akurasi sebesar 94% [4].

Penelitian lain yang memanfaatkan teknik klasifikasi dengan metode *machine learning* adalah penelitian yang dilakukan oleh Rifqi Abdul Aziz, Mohamad Syahrul Mubarok, dan Adiwijaya pada tahun 2016 yang berjudul Klasifikasi Topik pada Lirik Lagu dengan Metode *Multinomial Naive Bayes*. Sistem mengklasifikasikan lirik lagu kedalam kelas topik lagu cinta, persahabatan, nasionalisme, keluarga, religi dan konten negatif. Akurasi performansi sistem yang dihasilkan dari penelitian ini adalah sebesar 96% dan 88,91% untuk nilai *f1-measure* [5].

Penelitian yang memanfaatkan teknik klasifikasi juga telah dilakukan oleh Mohamad Syahrul Mubarok, Adiwijaya, dan Muhammad Dwi Aldhi. Penelitian tersebut melakukan analisis sentimen berbasis aspek pada *review product* menggunakan metode *Naive Bayes*. Dari penelitian tersebut, akurasi *f1-measure* yang dihasilkan dari sistem adalah sebesar 78,12% [6].

2.2 *Text Classification*

Text Classification adalah salah satu teknik dari *text mining* yang bertujuan untuk menentukan kelas atau kategori dari suatu teks yang dapat berbentuk frase, kalimat, paragraph, atau bahkan dokumen teks [6]. Proses klasifikasi teks memiliki beberapa teknik salah satunya adalah klasifikasi dilakukan berdasarkan data teks yang sudah dilatih pada sistem atau bisa juga disebut sebagai *supervised learning* [5]. Dengan menggunakan *machine learning*, proses klasifikasi ini melibatkan algoritma *classifier* yang digunakan untuk melakukan proses *learning* dan *testing* pada dataset. Terdapat beberapa metode yang dapat digunakan untuk pada klasifikasi data teks yaitu *Support Vector Machine, Bayesian Network, Random Forest*, dan ANN [7] [8] .

2.3 *Pre-processing*

Preprocessing adalah tahap pemrosesan awal dengan tujuan untuk mengubah data ke dalam bentuk yang terstruktur sesuai kebutuhan, yang biasanya data hasil dari pemrosesan awal ini akan diubah kedalam nilai numerik [9] [10]. Setelah data melalui proses *preprocessing*, maka data disajikan sebagai sumber data yang dapat diolah lebih lanjut. Beberapa teknik *preprocessing* yang digunakan pada penelitian ini adalah *data preparation* berupa pemisahan hadits dengan sanadnya kemudian melakukan proses *hand labeling* secara manual, *remove punctuation, case folding, tokenizing, stemming* menggunakan algoritma sastrawi, dan *stopword removal* yang dibuat sendiri menyesuaikan data teks hadits.

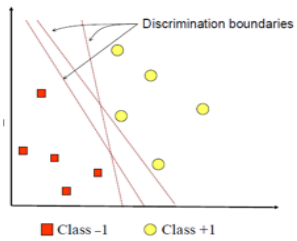
2.4 TF-IDF

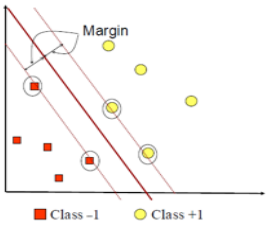
TF-IDF *(Term Frequency Inverse Document Frequency)* merupakan salah satu metode pembobotan *term*. Metode ini menghitung TF (*Term Frequency*) yang merupakan jumlah kemunculan tiap term pada setiap dokumen dan IDF (*Inverse Document Frequency*) yang merupakan jumlah dokumen terkait yang mengandung suatu *term* tertentu [11]. *Input* yang digunakan pada proses ektraksi fitur ini adalah hasil *output* dari proses *preprocessing* berupa *bag of words* .*Bag of words* didapatkan dari proses penggabungan semua hasil *preprocessing* data teks hadits, atau pada istilah matematika biasa di sebut sebagai *union* [12]. Sedangkan *output* proses ekstraksi fitur ini adalah berupa bobot dari setiap fitur berdasarkan dokumen terkait. Kemudian, akan diubah menjadi bentuk matriks berdimensi $(d\ x\ n)$ dengan *d* adalah koleksi dokumen dan *n* adalah koleksi fitur. Matriks adalah kumpulan dari bilangan yang yang terdiri dari baris dan kolom, dimana bilangan yang ada pada matriks disebut elemen matriks [13]. Elemen dari matriks *word vector* ini adalah berupa bobot dari setiap *term* atau kata. Selanjutnya, matriks tersebut akan menjadi *input* bagi SVM dalam membangun model *classifier*.

2.5 *Support Vector Machine*

SVM (*Support Vector Machine*) adalah salah satu teknik yang dikembangkan oleh Boser, Guyon, Vapnik dan pertama kali diperkenalkan di Annual Workshop on Computational Theory pada tahun 1992 [8]. SVM adalah salah satu teknik yang relatif baru dbandingkan dengan teknik lain, namun SVM memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti bioinformatika, pengenalan tulisan tangan, dan klasifikasi teks [8]. Konsep dasar dari metode SVM ini adalah menganalisis data dan mendefinisikan batas kelas dengan menemukan *hyperplane* terbaik pada *input space* [8] [14] [15]. *Hyperplane* pemisah terbaik dapat ditemukan dengan mengukur margin *hyperplane* tersebut dan mencari titik maksimalnya. Margin adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing kelas. Pattern yang paling dekat ini disebut sebagai *support vector*.

Pada prinsipnya SVM merupakan *linear classifier* yaitu dimana dataset dapat dipisahkan secara linear atau dengan garis lurus. Pada SVM, data yang akan diklasifikasikan dinotasikan dengan $\{x_1, \dots, x_n\}$ sedangkan label kelas dari data x_i dinotasikan dengan $y_i \in \{+1, -1\}$. Berikut adalah ilustrasi proses *learning* dari SVM yang mencari alternatif bidang pemisah sacara linear.





Gambar 2. Alternatif bidang pemisah [16]

Gambar 1. *Hyperplane* terbaik dengan margin terbesar [16]

Pada Gambar 2 diatas, menunjukkan bahwa terdapat dua kelas yang direpresentasikan dengan kotak merah yang merupakan kelas -1 dan bulat kuning yang merupakan kelas +1. *Discrimination boundaries* adalah alternatif garis pemisah pada proses *learning* SVM. Garis pada Gambar 3 menunjukkan *hyperplane* yang terbaik, yaitu terletak tepat pada tengah-tengah kedua kelas dan memiliki margin terbesar, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. [16] Untuk memenuhi kondisi seperti pada Gambar 3, maka harus memenuhi persamaan (2) berikut.

$$\vec{w}.\vec{x} + b = 0 \quad (2)$$

Sedangkan bidang pembatas yang membatasi kelas +1 dan kelas -1 didefinisikan dengan persamaan (3) berikut.

$$(w.x_i) + b \geq 1, untuk\ y_i = +1 \quad (3)$$
$$(w.x_i) + b \geq 1, untuk\ y_i = -1$$

Dengan *w* adalah normal bidang dan *b* adalah posisi bidang alternatif terhadap pusat koordinat. Kemudian, nilai *margin* terbesar dapat ditemukan dengan memaksimalkan jarak antara *hyperplane* dengan titik terdekatnya yaitu dapat dirumuskan dengan $\frac{1}{\|\vec{w}\|}$ yang dimaksimalkan dengan tetap memenuhi persamaan (3) [16]. Penentuan *margin* terbesar dapat diselesaikan dengan *Quadratic Programming* dengan menentukan solusi dari persamaan (4) dan memperhatikan batas dari persamaan (5) berikut. [16]

$$\min_{\vec{w}} \tau(w) = \frac{1}{2}\|\vec{w}\|^2 \quad (4)$$
$$y_i(\vec{x}_i.\vec{w} + b) - 1 \geq 0, \forall_i \quad (5)$$

Penyelesaian *QP problem* diatas dapat diselesaikan dengan menggunakan formula *Lagrangian* yaitu menambahkan λ (koefisien *Lagrange multiplier*) dan melakukan normalisasi parameter *w*, sehingga fungsi keputusan berubah menjadi persamaan (7) sebagai berikut. [16]

$$f(x) = sign\left(\sum_{i=1}^{l} y_i\, \lambda_i(x, x_i) + b\right) \quad (7)$$

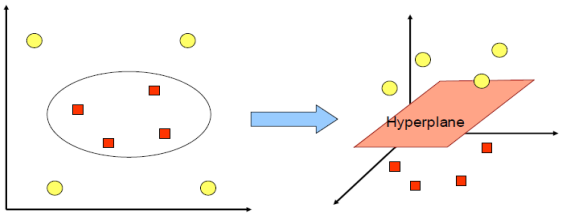
Solusi dari persamaan diatas akan menghasilkan α_i yang bernilai positif dan disebut sebagai *support vector*. Penyelesaian diatas adalah *hard margin* dimana diasumsikan bahwa semua data dapat diklasifikasikan 100% dengan benar, namun pada kenyataannya tidak semua data berhasil diklasifikasikan. Sehingga pada SVM terdapat *soft margin* yang berguna untuk melunakkan batas (*constraint*) dengan memberikan toleransi untuk data yang tidak terklasifikasikan dengan benar dengan menambahkan variabel *slack* pada solusi penyelesaian. Sehingga formulanya berubah menjadi persamaan (8) sebagai berikut.

$$\min \frac{1}{2}|w|^2 + C\left(\sum_{i=1}^{n} \xi_i\right)$$
$$y_i(\vec{x}_i.\vec{w} + b) \geq 1 - \xi_i \quad (8)$$
$$\xi_i \geq 0$$

Dengan C merupakan parameter yang menentukan besarnya penalti akibat kesalahan dari klasifikasi data dan nilainya ditentukan oleh pengguna. Selain dapat menyelesaikan problem klasifikasi secara *linear*, SVM telah dikembangkan juga untuk menangani klasifikasi data *non-linear* dengan memasukkan konsep *kernel trick* pada ruang berdimensi lebih tinggi [16]. Dalam *non-linear* SVM, data $\vec{x}$ dipetakan oleh fungsi $\phi(\vec{x})$ ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua kelas tersebut dapat dikonstruksikan. Setelah data ditransformasikan pada *feature space* yang berdimensi lebih tinggi, proses *learning* hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yaitu $\Phi\vec{X}_i.\Phi\vec{X}_j$. Pada umumnya transformasi dari Φ tidak dapat diketahui dan sulit dipahami, sehingga *dot product* dari data dapat digantikan dengan fungsi kernel yang mendefinisikan secara implisit transformasi dari Φ tersebut [16]. Selanjutnya, formula pada *non-linear separable* dengan *kernel trick* dapat dilihat pada persamaan (9) berikut.

$$f\left(\Phi(\vec{x})\right) = \vec{w}.\Phi(\vec{x}) + b$$
$$= \sum_{i=1,\vec{x}_i \in SV}^{n} \alpha_i y_i \Phi(\vec{x}).\Phi(\vec{x}_i) + b \quad (9)$$
$$= \sum_{i=1,\vec{x}_i \in SV}^{n} \alpha_i y_i K(\vec{x}, \vec{x}_i) + b$$

Ilustrasinya dapat dilihat pada Gambar 4 berikut. Pada Gambar 4 (kiri) diperlihatkan bahwa data pada kelas kuning dan data pada kelas merah berada pada *input space* berdimensi dua dan tidak dapat dipisahkan secara linear. Selanjutnya Gambar 4 (kanan) menunjukkan bahwa fungsi ϕ memetakan tiap data pada *input space* tersebut ke ruang vektor baru yang berdimensi lebih tinggi (dimensi 3) , dimana kedua kelas dapat dipisahkan secara linear oleh sebuah *hyperplane.* Selanjutnya, proses *learning* pada SVM dalam menemukan titik-titik *support vector* hanya bergantung pada *dot pruduct* dari data yang telah ditransformasikan ke dalam ruang baru [16].



Gambar 3. Ilustrasi Non-linier SVM [16]

Terdapat beberapa jenis fungsi kernel yang sering digunakan, yaitu *Polynomial*, Gaussian, dan Sigmoid [16]. Tabel 1 berikut adalah fungsi kernel yang ada pada SVM. *Kernel trick* memberikan berbagai kemudahan, karena dalam proses pembelajaran SVM, untuk menentukan *support vector*, kita hanya cukup mengetahui fungsi kernel yang dipakai, dan tidak perlu mengetahui wujud dari fungsi *non-linear* ϕ.

2.6 SVM *Multiclass Classification*

Pada dasarnya, *Support Vector Machine* hanya dapat digunakan untuk kasus klasifikasi biner atau mengklasifikasikan data ke dalam dua kelas. Namun, seiiring dengan penelitian lebih lanjut yang dikembangkan pada SVM , saat ini SVM dapat mengklasifikasikan data yang memiliki lebih dari dua kelas. Pendekatan yang dapat digunakan pada SVM untuk *multiclass classification* adalah metode *"one-against-all"* dan metode *"one-against-one"*. Namun, pada penelitian ini, peneliti memilih model "*one-againts-all*" untuk menyelesaikan klasifikasi data hadits ke dalam tiga kelas. Model yang digunakan pada metode ini adalah dengan membangun k (jumlah kelas) buah model SVM biner. Setiap model SVM biner yang dibangun terdiri dari sebuah *classifier* yang memisahkan salah satu *sample* kelas sebagai kelas positif dan *sample* lainnya sebagai kelas negatif. Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data untuk mencari solusi permasalahannya [17]. Hasil dari proses pelatihan, akan didapatkan *k* fungsi keputusan dari setiap *k* model *classifier* yang dibangun. Proses yang terjadi pada data baru *x* yang akan diujikan menggunakan pendekatan *one-versus-rest* ini, yaitu data *x* akan masuk ke dalam kelas tertentu yang memiliki nilai fungsi keputusan terbesar yaitu $\hat{y} = argmax\ f_k(x)$ dengan $k \in \{1 \ldots . . K\}$. Ilustrasi pemodelannya di gambarkan pada Tabel 2 dibawah ini.

Tabel 1. Ilustrasi *One-againts-all* [17]

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas informasi	Bukan kelas informasi	$f^1(x) = (w^1)x + b^1$
Kelas anjuran	Bukan kelas anjuran	$f^2(x) = (w^2)x + b^2$
Kelas larangan	Bukan kelas larangan	$f^3(x) = (w^3)x + b^3$

2.7 K-*Fold*

K-fold adalah salah satu teknik *cross-validation* yang digunakan untuk membagi data menjadi data latih dan data uji. *K-fold cross-validation* membagi data secara acak menjadi *k* bagian berukuran sama. Secara bertahap akan dilakukan pelatihan dan validasi sebanyak *k* ulangan. Sehingga dalam setiap perulangan *k-1* bagian akan menjadi data latih dan 1 bagian sisanya akan digunakan untuk validasi. [17] [18]

2.8 *F1-score*

Pada umumnya, *F1-score* digunakan untuk mengukur performansi sistem pada kasus *binary classification*, namun *F1-score* juga dapat digunakan untuk mengukur performansi sistem pada *multiclass classification* dengan menghitung *precision* dan *recall* pada setiap label/kelas yang kemudian dirata-rata. Pengukuran yang dilakukan adalah berdasarkan prediksi yang didapatkan dari *confusion matrix*. Parameter yang diukur adalah:

Tabel 2. Tabel Confusion Matrix

	Correct Tabel	
	Positive	**Negative**
Positive	TP (True Positive)	FP (False Positive)
Negative	FN (False Negative)	TN (True Negative)

- *Precision* adalah jumlah kelas anjuran,larangan, atau informasi yang diprediksi dengan benar oleh sistem dibagi total jumlah kelas anjuran, larangan, atau informasi pada dataset.

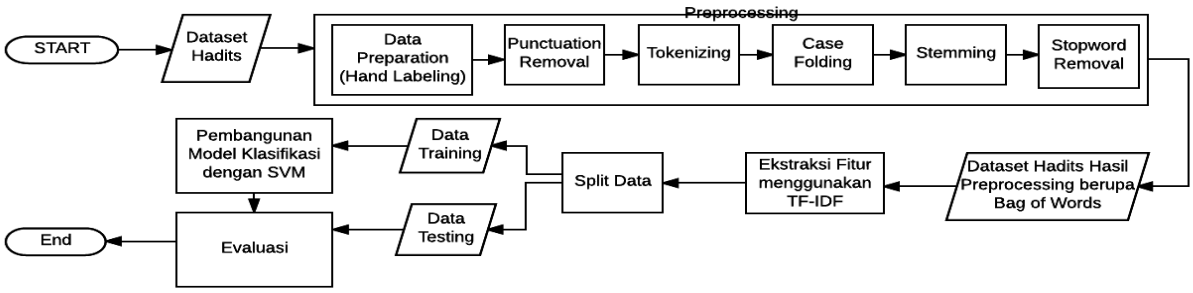
  $Precision = \frac{TP(Anjuran/Larangan/Informasi)}{TP(Anjuran/Larangan/Informasi)+FP(Anjuran/Larangan/Informasi)}$
- *Recall* adalah jumlah kelas anjuran,larangan,atau informasi yang diprediksi dengan benar oleh sistem dibagi dengan jumlah kelas yang diprediksi benar ditambah jumlah kelas yang relevan dari kelas yang diprediksi.

  $Recall = \frac{TP(Anjuran/Larangan/Informasi)}{TP(Anjuran/Larangan/Informasi)+FN(Anjuran/Larangan/Informasi)}$
- *F1-score* adalah bobot akurasi berdasarkan rata-rata dari *precision* dan *recall*.

  $F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision+Recall}$

3. Perancangan Sistem

Dalam penelitian ini, dibangun sebuah sistem yang dapat melakukan klasifikasi teks secara otomatis pada data teks Hadits Shahih Bukhari ke dalam kelas hadits anjuran, larangan, dan informasi. Gambaran sistem yang dibangun dapat dilihat pada Gambar 3 dibawah ini.



Gambar 4. Gambaran umum sistem

Dataset Hadits

Dataset yang digunakan pada penelitian ini diambil dari aplikasi lidwa "Kitab Hadits 9 Imam". Data asli hadits ini belum memiliki label dan masih mengandung *sanad*, sehingga perlu dilakukan proses data *preparation* untuk memisahkan hadits dengan *sanad* dan melakukan pelabelan pada data secara manual. Kemudian dilakukan proporsi data yang seimbang untuk setiap kelas. Berikut adalah jumlah data yang digunakan pada penelitian ini.

Tabel 3. Indeks dan jumlah data hadits yang digunakan pada penelitian

Indeks	Kelas	Jumlah Data
1-550	Informasi	550
551-1100	Anjuran	550
1101-1650	Larangan	550

Tabel 4. Penjelasan detail pada gambaran umum sistem klasifikasi data teks hadits

No.	Tahapan	Input	Proses	Output
1.	*Preprocessing*	*Dataset* Hadits	Tahapan pemrosesan awal yang dilakukan pada data teks hadits adalah terdiri dari *data preparation, remove punctuation, tokenizing, case folding, stemming* menggunakan algoritma sastrawi dan *stopword removal*.	*Bag of words*
2.	*Feature Extraction*	*Bag of words*	Mengubah *bag of words* dari proses *preprocessing* menjadi *word vector* dengan melakukan pembobotan kata menggunakan TF-IDF (*Term Frequency- Inverse Document Frequency*).	*Word vector* yang dinyatakan dengan matriks (*d x n*), *d* adalah koleksi dokumen, *n* adalah koleksi *term*, dan elemen matriks adalah bobot setiap *term*.
3.	*Split* Data	*Word vector*	Melakukan *splitting* data hadits berupa *word vector* menjadi data *training* dan data *testing* menggunakan 5-*fold cross validation* dengan perbandingan 90:10.	*Word vector* untuk data *training* dan *word vector* untuk data *testing*.
4.	Pembangunan Model Klasifikasi	*Word vector* dan kelas *training*	Menerapkan pendekatan *one-versus-rest* pada SVM *multiclass* dengan membangun tiga buah model klasifikasi biner yaitu *classifier* yang memisahkan kelas informasi dan non-informasi, kelas anjuran dan non-anjuran, kelas larangan dan non-larangan. Semua model dilatih dengan keseluruhan data *training*.	Tiga fungsi keputusan dari masing-masing model klasifikasi.
5.	Klasifikasi	*Word vector* data *testing*	Data *testing* diujikan pada ketiga model klasifikasi untuk menentukan kelas dari data tersebut.	Hasil klasifikasi kelas dari data uji berdasarkan nilai fungsi keputusan terbesar.

6.	Evaluasi	Hasil klasifikasi	Mengukur performansi dari *classifier* yang dibangun dan dinyatakan dengan *F1-score*	Nilai akurasi *F1-score* dari sistem klasifikasi

4. Pengujian dan Analisis

4.1 Tujuan Pengujian

Tujuan dari pengujian sistem klasifikasi hadits yang telah dibangun ini adalah sebagai berikut:

1. Membandingkan serta menganalisis performansi dari pendekatan *machine learning* menggunakan *Support Vector Machine* dan pendekatan *rule-based system* yaitu *string matching* dalam mengklasifikasikan data teks hadits.
2. Menganalisis pengaruh variasi parameter C pada fungsi kernel linear dalam peningkatan akurasi.
3. Menganalisis pengaruh variasi parameter *p* atau degree pada fungsi kernel polynomial dalam peningkatan akurasi.
4. Menganalisis pengaruh variasi paramater C dan γ pada fungsi kernel RBF dalam peningkatan akurasi.
5. Menganalisis pengaruh *split data* untuk setiap *classifier* paling optimal dari masing-masing fungsi kernel terhadap akurasi sistem.

4.2 Skenario Pengujian

Berdasarkan tujuan dari pengujian sistem yang telah dijelaskan diatas, maka skenario pengujian yang akan dilakukan adalah sebagai berikut.

1. Perbandingan akurasi dari performansi *rule-based system* dan *machine learning* menggunakan SVM.
   Pada skenario pengujian ini, akan dibandingkan performansi *classifier* yang dibangun dengan pendekatan *rule-based system* dan *machine learning* SVM. Untuk *classifier* SVM, data teks hadits akan melalui proses *preprocessing* yang terdiri dari *data preparation*, *punctuation removal*, *tokenizing*, *case folding*, *stemming*, dan *stopword removal* dan pembobotan *term* dengan TF-IDF sedangkan untuk *classifier rule-based system*, data teks hadits tidak melalui kedua proses tersebut. Kemudian, akan dujikan tiga fungsi kernel pada *classifier* SVM yaitu fungsi kernel *linear*, *polynomial*, dan RBF.
2. Pengaruh variasi paramater C (*cost*) pada akurasi performansi fungsi kernel *linear*.
   Untuk kernel *linear*, akan dicari parameter terbaik dari parameter yang paling berpengaruh pada performansi *classifier* yaitu C *(cost)*, dimana nilai C merupakan tingkat toleransi terhadap kesalahan pada *classifier*. Variasi parameter C yang akan diujikan adalah C = {0.001 ; 0.01 ; 0.1 ; 1 ; 10 ; 100}
3. Pengaruh variasi paramater *p* atau *degree* pada akurasi performansi fungsi kernel *polynomial*.
   Untuk kernel *polynomial*, skenario pengujian yang dilakukan adalah dengan mencari parameter terbaik dari nilai *p* atau *degree* yang merupakan derajat fungsi dari kernel *polynomial*. Variasi parameter *p* yang akan diujikan adalah *p* = {1 ; 2 ; 3 }
4. Pengaruh variasi paramater C (*cost*) dan γ (*gamma*) pada akurasi performansi fungsi kernel RBF.
   Untuk kernel RBF, skenario pengujian dilakukan dengan melakukan variasi terhadap nilai C dan *gamma*. Variasi parameter C dan *gamma* yang akan diujikan adalah sebagai berikut.

Tabel 5. Tabel variasi parameter C dan gamma pada fungsi kernel RBF

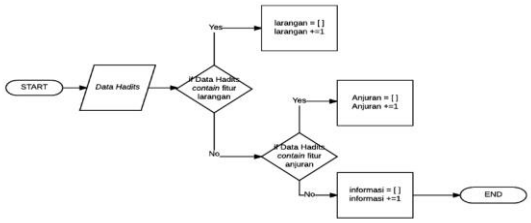
No.	Parameter C	Parameter γ
1.	0,1	γ = {0,01 ; 0,1 ; 1 ; 10}
2.	1	
3.	10	

5. Pengaruh variasi *split data* pada akurasi performansi setiap fungsi kernel.
   Sedangkan untuk sknario ini, peneliti ingin menganalisis pengaruh proporsi data pada akurasi performansi *classifier*, sehingga proporsi data *training* dan data *testing* yang akan diujicobakan adalah 90:10 ; 80:20 ; 70:30 ; 60:40 ; dan 50:50. Proporsi data tersebut akan diterapkan pada *classifier* optimal yang didapatkan dari skenario 2, 3, dan 4.

4.3 Hasil Pengujian dan Evaluasi

4.3.1 Hasil Pengujian perbandingan akurasi dari performansi *rule-based system* dan *machine learning* menggunakan SVM.

Berikut adalah konsep dasar classifier yang dibangun dengan menggunakan pendekatan *baseline String Matching*.



Gambar 5. Flowchart string matching

Tabel 6. Contoh fitur unik masing-masing kelas

Contoh fitur unik pada kelas Anjuran	Contoh fitur unik pada kelas Larangan	Contoh fitur unik pada kelas Informasi
Hendaklah	Janganlah	Untuk fitur yang berorientasi pada kelas informasi pada umunya adalah kata kerja seperti contohnya mengabarkan, memberikan, menunaikan dan lain sebagainya.
Shalatlah	Melarang	
Bersedekahlah	Tidak boleh	

Tujuan pengujian *classifier* yang dibangun dengan *string matching* adalah untuk memastikan bahwa masalah klasifikasi data teks hadits ini tidak cukup apabila hanya dengan menggunakan SVM karena terdapat fitur-fitur yang seharusnya merupakan fitur unik dari kelas anjuran namun ada didalam hadits larangan dan informasi. Apabila dilihat berdasarkan intuisi manusia itu akan mudah, namun untuk mesin hal itu akan menyebabkan kesalahan klasifikasi. Hal itu dapat dilihat dari hasil akurasi dari *classifier* yang dibangun dengan *string matching* yaitu hanya **66,30%.** Contoh dari data hadits tersebut adalah sebagai berikut.

Tabel 7. Contoh hadits yang memiliki fitur yang berorientasi pada kelas lain

No.	Dataset Hadits	Keterangan
1.	Jika salah seorang dari kalian masuk ke dalam WC untuk buang hajat, maka **janganlah** menghadap ke arah kiblat membelakanginya. **Hendaklah** ia menghadap ke arah timurnya atau baratnya.	Data hadits yang memiliki fitur unik anjuran dan larangan sekaligus yaitu "hendaklah" dan "janganlah"
2.	Barangsiapa berdusta terhadapku maka **hendaklah** ia persiapkan tempat duduknya di neraka.	Data hadits yang memiliki fitur unik anjuran, namun apabila dilihat secara intuisi merupakan hadits larangan.

Sedangkan akurasi performansi yang dihasilkan dari *classifier* SVM menghasilkan akurasi yang lebih baik dibandingkan *classifier* yang dibangun dengan *rule-based system*. Berikut adalah hasil pengujian dari perbandingan antara *classifier* SVM dan *string matching.*

Tabel 8. Perbandingan hasil akurasi dari classifier string matching dan SVM

No.	**Akurasi *String Matching***	**Akurasi *Kernel Linear***	**Akurasi *Kernel Polynomial***	**Akurasi *Kernel* RBF**
1.	66,30%	87,79%	87,79%	88,20%

4.3.2 Hasil Pengujian pengaruh variasi paramater C (*cost*) pada akurasi performansi fungsi kernel *linear*.

Pada skenario pengujian pertama, *classifier* dibangun dengan menggunakan pendekatan fungsi kernel linear dan mengatur parameter C *(cost)* pada enam titik yaitu $C = 0{,}001\ ;\ 0{,}01\ ;\ 0{,}1\ ;\ 1\ ;\ 10\ ;\ 100$. Kemudian setiap proses *running* dilakukan *split* data dengan 5-*fold cross validation*. Berikut adalah grafik hasil dari skenario pengujian pertama.

Tabel 9. Hasil Pengujian pengaruh variasi parameter C pada kernel linear

No	Parameter C	*Fold*-1	*Fold*-2	*Fold*-3	*Fold*-4	*Fold*-5	Avg
1.	0,001	85,08%	86,33%	83,14%	89,55%	88,72%	86,57%
2.	0,01	85,73%	86,33%	83,80%	89,55%	88,72%	86,83%
3.	0,1	84,54%	86,33%	83,19%	88,90%	89,42%	86,47%
4.	1	88,42%	86,93%	83,89%	88,30%	91,42%	**87,79%**
5.	10	85,20%	86,28%	85,82%	86,97%	87,48%	86,35%
6.	100	85,20%	86,28%	85,80%	86,97%	87,48%	86,35%

Berdasarkan Tabel 10, hasil pengujian diatas, dapat dilihat bahwa rata-rata akurasi *f1-score* yang dihasilkan dari fungsi kernel linear tidak menunjukkan peningkatan yang signifikan namun yang paling optimal dihasilkan dari *classifier* pada variasi parameter C dititik C = 1 yaitu sebesar **87,79%**. Hal itu memenuhi prinsip SRM (*Structural Risk Minimization*) yang berusaha untuk meminimumkan *error* pada data pelatihan dengan cara memilih *hyperplane* dengan margin terbesar. Parameter C mempengaruhi *margin hyperplane* pada *feature space* sehingga apabila semakin besar nilai C, maka semakin besar juga penalti yang disebabkan oleh *error* atau kesalahan pada klasifikasi data, sehingga sistem akan memperketat toleransi terhadap kesalahan pada klasifikasi data tersebut dan nilai α yang didapatkan juga optimal. Namun, apabila nilai C terlalu besar maka akan membuat data testing yang dipetakan berada pada area yang tidak diketahui kelasnya sehingga menyebabkan menurunnya akurasi.

4.3.3 Hasil Pengujian pengaruh variasi paramater *p* atau *degree* pada akurasi performansi fungsi *kernel polynomial.*

Pada skenario pengujian kedua, *classifier* dibangun dengan menggunakan pendekatan *non linear separable* dengan fungsi kernel *polynomial* dan melakukan variasi nilai *p* atau *degree* pada empat titik yaitu *p* = 1,2,3 sedangkan untuk parameter C dan *gamma* diatur dengan nilai 1. Berikut adalah hasil pengujian skenario kedua.

Tabel 10. Hasil pengujian pengaruh variasi parameter degree pada kernel polynomial

No	Parameter *p*	*Fold*-1	*Fold*-2	*Fold*-3	*Fold*-4	*Fold*-5	Avg
1.	1	88,42%	86,93%	83,89%	88,30%	91,42%	**87,79%**
2.	2	75,79%	81,25%	76,17%	78,37%	83,69%	79,05%
3.	3	74%	75,28%	74,11%	70,84%	76,05%	74,06%

Berdasarkan Tabel 11, hasil pengujian *classifier* menggunakan fungsi kernel *polynomial* mendapatkan hasil yang optimal saat parameter *degree* = 1 yaitu sebesar **87,79%.** Hal itu tidak sesuai dengan prinsip parameter *degree* yang pada dasarnya digunakan untuk memetakan data ke dalam *feature space* yang berdimensi lebih tinggi. Semakin besar nilai *degree*, maka semakin besar juga ruang dimensi *feature space* yang seharusnya semakin mudah dalam menemukan *hyperplane* yang optimal. Namun, kondisi ini bisa terjadi apabila kemungkinan data memang sudah dapat dipisahkan dengan baik secara linear, sehingga tidak membutuhkan pemetaan data pada *feature space* yang lebih tinggi. Hal itu ditunjukkan dengan akurasi *f1-score* yang cenderung turun saat parameter *degree* = 3. Nilai *degree* yang terlalu besar justru menyebabkan algoritma kesulitan dalam menemukan *classifier* yang optimal karena dimensi data yang terlalu besar sehingga mengakibatkan data semakin renggang dan mengakibatkan data kehilangan informasinya [18].

4.3.4 Hasil Pengujian variasi paramater C (*cost*) dan γ (*gamma*) pada akurasi performansi fungsi kernel RBF.

Pada skenario pengujian ketiga, *classifier* dibangun dengan menggunakan fungsi kernel RBF dengan melakukan variasi pada parameter *gamma* di empat titik yaitu *gamma* = 0,01 ; 0,1 ; 1 ; 10, sedangkan parameter C diatur pada variasi nilai yaitu 0,1; 1; dan 10. Hasil pengujian dijelaskan pada tabel berikut.

Tabel 11. Hasil pengujian pengaruh variasi parameter C dan gamma pada kernel RBF

No	C	γ	*Fold*-1	*Fold*-2	*Fold*-3	*Fold*-4	*Fold*-5	Avg
1.	0,1	0,01	85,72%	87,60%	86,35%	88,78%	90,06%	**87,72%**
2.		0,1	85,69%	87,66%	85,04%	88,88%	89,42%	87,34%
3.		1	79,32%	86,29%	81,42%	84,95%	86,63%	83,72%
4.		10	62,80%	65,92%	55,12%	56,82%	59,84%	60,10%
5.	1	0,01	85,72%	86,93%	85,03%	89,53%	90,70%	**87,58%**
6.		0,1	83,96%	88,31%	86,39%	88,91%	88,75%	87,26%
7.		1	79,44%	82,41%	76,27%	81,24%	86,07%	81,09%
8.		10	62,80%	65,92%	55,12%	55,93%	59,07%	59,77%
9.	10	0,01	84,60%	88,24%	85,84%	90,25%	91,40%	88,07%
10.		0,1	89,09%	86,31%	84,53%	89,61%	91,46%	**88,20%**
11.		1	81,48%	83,29%	77,58%	82,50%	86,76%	82,32%
12.		10	62,80%	65,92%	55,12%	56,82%	59,07%	59,95%

Berdasarkan tabel hasil pengujian *classifier* dengan menggunakan fungsi kernel RBF, akurasi *f1-score* yang paling optimal dihasilkan dari *classifier* dengan parameter *gamma* = 0,1 dan saat parameter C = 10 yaitu sebesar **88,20%.** Parameter C mempengaruhi jarak *margin* sedangkan *gamma* mempengaruhi tingkat kedekatan antara dua titik. Jika dilihat dari hasil pengujian diatas, akurasi optimal dihasilkan dari *classifier* dengan nilai *gamma* paling kecil dan nilai C paling besar. Untuk pengaruh parameter *gamma*, hal tersebut tidak memenuhi prinsipnya pada fungsi kernel RBF. Apabila nilai *gamma* semakin besar maka akan memudahkan dalam menemukan *hyperplane* yang optimal. Namun, untuk kasus klasifikasi data teks hadits ini menunjukkan bahwa *gamma* yang terlalu besar menyebabkan akurasi *f1-score* yang semakin menurun, hal itu karena apabila *gamma* terlalu besar maka akan menyebabkan algoritma kesulitan dalam menentukan *hyperplane* yang optimal pada *feature space* yang terlalu tinggi. Sedangkan untuk pengaruh parameter C, dengan nilai C = 10 didapatkan akurasi *classifier* yang paling optimal. Berdasarkan hal tersebut, dengan nilai C yang paling besar maka sistem akan memperketat *error* atau kesalahan klasifikasi sehingga didapatkan akurasi yang optimal.

4.3.5 Pengaruh variasi *split data* pada akurasi performansi setiap fungsi kernel.

Tujuan dari skenario pengujian keempat ini adalah menganalisis pengaruh dari proporsi *data training* dan *data testing* pada akurasi *f1-score*. Skenario proporsi data diimplementasikan pada *classifier* yang memiliki *f1-score* yang paling optimal pada skenario 1 , 2, dan 3. Hasil pengujian dijelaskan pada tabel dibawah ini.

Tabel 12. Hasil pengujian pengaruh variasi split data pada setiap fungsi kernel

Fungsi Kernel Linear C = 0,001 ; γ = 1						
Proporsi Data	***Fold*-1**	***Fold*-2**	***Fold*-3**	***Fold*-4**	***Fold*-5**	**Avg**
90:10	86,87%	87,39%	90,51%	87,27%	93,84%	89,18%

80:20	89,35%	89,43%	88,71%	89,88%	92,37%	**89,95%**
70:30	85,62%	87,87%	84,70%	87,70%	89,85%	87,14%
60:40	85,25%	86,29%	80,10%	90,86%	87,35%	85,97%
50:50	89,67%	80,01%	86,10%	93%	86,84%	87,13%
Fungsi Kernel *Polynomial* $p = 1$; $C = 1$; $\gamma = 1$						
90:10	90,77%	89,61%	89,90%	87,76%	94,42%	**90,49%**
80:20	92,02%	89,44%	88,08%	89,98%	91,84%	90,27%
70:30	88,01%	88,45%	84,93%	89,38%	92,06%	88,57%
60:40	86,89%	89,80%	81,01%	88,38%	89,06%	87,03%
50:50	88,59%	81,31%	85,08%	89,08%	88,94%	86,56
Fungsi Kernel RBF $C = 1$; $\gamma = 0{,}001$						
90:10	89,03%	86,33%	90,51%	87,83%	93,27%	89,39%
80:20	91,31%	90,05%	89,35%	90,50%	91,72%	**90,59%**
70:30	87,06%	87,86%	87,79%	90,66%	90,59%	88,79%
60:40	87,71%	87,10%	81,74%	93,28%	88,15%	87,60%
50:50	88,74%	80,93%	87,10%	92,01%	86,84%	87,13%

Berdasarkan tabel hasil pengujian pada skenario empat, dapat dilihat bahwa akurasi *f1-score* yang optimal didapatkan dari proporsi data 80:20 untuk fungsi kernel linear dan RBF, sedangkan akurasi *f1-score* yang optimal pada kernel *polynomial* didapatkan saat proporsi data 90:10. Hal itu disebabkan proporsi data yang tepat dapat mempengaruhi akurasi *f1-score* dari sistem. Apabila jumlah proporsi *data training* semakin dikurangi, *classifier* akan mengalami kesulitan dalam melakukan prediksi.

5. Kesimpulan dan Saran.

5.1 Kesimpulan

Berdasarkan analisis dan pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Klasifikasi yang dilakukan pada Data teks Hadits Shahih Bukhari dengan menggunakan pendekatan *machine learning* SVM menghasilkan akurasi performansi yang lebih baik dibandingkan dengan *classifier* yang dibangun dengan pendekatan *rule-based system* yaitu *string matching*.
2. Ketiga fungsi kernel yang diterapkan pada *classifier* SVM mempengaruhi akurasi dari performansi *classifier* walaupun tidak menunjukkan perbedaan yang jauh. Hal itu disebabkan karena untuk kasus data teks hadits ini dapat diklasifikasikan dengan baik secara *linear separable* maupun *non-linear separable.*
3. Variasi parameter C pada skenario pengujian *classifier* yang dibangun dengan menggunakan *kernel linear* menunjukkan akurasi yang optimal pada nilai $C \geq 1$, yaitu **88,39%** ketika nilai C terlalu kecil dan terlalu besar membuat akurasi cenderung turun.
4. Variasi parameter *p* atau *degree* pada skenario pengujian *classifier* yang dibangun dengan menggunakan *kernel polynomial* menunjukkan akurasi yang optimal pada nilai $p \geq 1$, yaitu **87,79%** ketika nilai *p* semakin diperbesar membuat akurasinya menurun.
5. Variasi parameter C dan γ pada skenario pengujian *classifier* yang dibangun dengan menggunakan *kernel RBF* menunjukkan akurasi yang optimal pada nilai $C \geq 10$ dan $\gamma \geq 0{,}1$, yaitu **88,20%**.
6. Proporsi perbandingan antara data *training* dan testing yang tepat dapat mempengaruhi peningkatan akurasi performansi pada *classifier*. Proporsi perbandingan *data training* dan *data testing* yang menghasilkan akurasi paling optimal adalah 90:10 untuk setiap *classifier* dengan ketiga fungsi kernel yang diterapkan.

5.2 Saran

Saran yang diperlukan untuk pengembangan sistem pengklasifikasian Hadits Shahih Bukhari ini adalah sebagai berikut:

1. Melakukan *multilabel classification* karena terdapat beberapa hadits yang berorientasi pada beberapa kelas.
2. Menambahkan jumlah data yang digunakan pada proses klasifikasi.
3. Melakukan penelitian terhadap *imbalance data* menggunakan SVM karena data dari kelas informasi lebih mendominasi dalam data Hadits Shahih Bukhari.

Daftar Pustaka

[1] H. A, "Hadith Nabi Sebagai Sumber Ajaran Islam : Dari Makna Lokal-Temporal Menuju Makna Universal," *Istinbath, Jurnal Hukum Islam,* vol. XII, pp. 2-10, 2013.

[2] S. D, T. D dan S. E, "Pengembangan Aplikasi Seratus Satu Hadits Tentang Budi Luhur Berbasis Multimedia," *Jurnal Algoritma Sekolah Tinggi Teknologi Garut,* vol. XIII, pp. 137-138, 2016.

[3] Y. N, Sunnah 9 Kitab Imam Hadits dalam Bahasa Indonesia, app.lidwa, 2009.

[4] M. S. Mubarok. Adiwijaya. Untari N. Wisesty, "A Classification of Marked Hijaiyah Letters's Pronunciation Using Hidden Markov Model," *AIP Conference roceedings 1867 (1),* no. 020036, 2017.

[5] R. A. Aziz, M. S. Mubarok dan Adiwijaya. , "Klasifikasi Topik pada Lirik Lagu dengan Metode Multinomial Naive Bayes," *In Indonesia Symposium on Computing (IndoSC),* 2016.

[6] M. R. M, "Perbandingan Naive Bayes Classifier dan Support Vector Machine untuk Klasifikasi Judul Artikel," *JISKa,* vol. I, pp. 90-93, 2016.

[7] M. S. Mubarok. Adiwijaya. Azmi Hafizha Rahman Z.A., "Learning Struktur Bayesian Networks menggunakan Novel Modified Binary Differential Evolution pada Klasifikasi Data," *Indonesian Symposium on Computing,* pp. 265-278, 2016.

[8] I. S. Karima, "Optimasi Parameter Pada Support Vector Machine Untuk Klasifikasi Fragmen Metagenome Menggunakan Algoritme Genetika," pp. 3-6, 2014.

[9] S. S dan W. M, "K-Nearest Neighbors sebagai Analisis Sentiment Review Produk Appstore for Android," *Konferensi Nasional Ilmu Pengetahuan dan Teknologi (KNIT),* pp. 82-87, 2015.

[10] M. S. Mubarok, Adiwijaya. dan M. D. Aldhi, "Aspect-based sentiment analysis to review products using Naïve Bayes," *AIP Conference Proceedings 1867,* vol. 020060, p. 2, 2017.

[11] A. Yusuf dan T. Priambadha, "Support Vector Machines yang Didukung K-Means Clustering dalam Klasifikasi Dokumen," *JUTI,* vol. XI, no. 1, pp. 13-16, 2013.

[12] Adiwijaya, Matematika Diskrit dan Aplikasinya, Bandung: Alfabeta, 2016.

[13] Adiwijaya, Aplikasi Matriks dan Ruang Vektor, Yogyakarta: Graha Ilmu, 2014.

[14] F. P. Shah dan V. Patel, "A Review on Feature Selection and Feature Extraction for Text Classification," *IEEE WiSPNET 2016 conference,* p. 2264, 2016.

[15] A. Tripathy, A. Agrawal dan S. K. Rath, "Classification of Sentiment Reviews using N-gram Machine Learning Approach," *Expert Systems With Applications,* 2016.

[16] A. S. Nugroho, A. B. Witarto dan D. Handoko, "Support Vector Machine (Teori dan Aplikasinya dalam Bioinformatika)," *Kuliah Umum Ilmu Komputer.com,* 2003.

[17] S. K, "Penerapan Teknik Support VEctor Machine untuk Pendeteksian Intrusi pada Jaringan," pp. 3-4, September 2007.

[18] S. Prangga, "Optimasi Parameter pada Support Vector Machine menggunakan Pendekatan Metode Taguchi untuk Data High-Dimensional," pp. 33-37, 2017.