ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

Text Message Classification using Multiclass Support Vector Machine on Information Service Chatbot in the Informatics Department UPN "Veteran" Yogyakarta

Klasifikasi Teks Pesan menggunakan *Multiclass Support Vector Machine* pada *Chatbot* Pelayanan Informasi Jurusan Informatika UPN "Veteran" Yogyakarta

Rafly Pradana Putra[1], Awang Hendrianto Pratomo[2], Rifki Indra Perwira[3]

[1,2,3] Informatika, Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia

[1*]123170028@student.upnyk.ac.id, [2]awang@upnyk.ac.id, [3]rifki@upnyk.ac.id
**: Penulis korenspondensi* (*corresponding author*)

Article's Information / Informasi Artikel
Received: July 2022
Revised: September 2022
Accepted: October 2022
Published: October 2022

Abstract

Purpose: Testing the performance of the Multiclass Support Vector Machine algorithm in terms of accuracy, precision, and recall to classify chatbot text messages
Design/methodology/approach: Using the Multiclass Support Vector Machine algorithm to classify non-binary datasets or datasets that have more than two classes
Findings/result: Based on the results of the research and discussion conducted, the Multiclass SVM Algorithm can classify the text of messages sent properly based on message categories related to information in the information department by showing an accuracy value of 87%, a precision value of 89% and a recall value by 87%. This study uses a dataset of 950 data with the distribution of training data as much as 75% of the total data and test data as much as 25% of the total data.
Originality/value/state of the art: This study has differences in terms of the data used, the methods used, the parameters used, and the results obtained when compared to previous studies.

Keywords:chatbot; nlp; svm
Kata kunci: chatbot; nlp; svm

Abstrak

Tujuan: Menguji performa dari algoritma *Multiclass Support Vector Machine* dalam dalam tingkat akurasi, presisi, dan *recall* untuk melakukan klasifikasi pada pesan teks *chatbot.*
Perancangan/metode/pendekatan: Menggunakan algoritma *Multiclass Support Vector Machine* untuk melakukan

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

klasifikasi terhadap dataset yang bersifat nonbiner atau dataset yang memiliki lebih dari dua kelas.
Hasil: Berdasarkan hasil penelitian dan pembahasan yang dilakukan, Algoritma Multiclass SVM dapat melakukan klasifikasi teks pesan yang dikirimkan dengan baik berdasarkan kategori pesan yang berkaitan dengan informasi yang ada di jurusan informasi, seperti administratif, dokumen, jadwal, kegiatan dan sapaan, dengan menunjukkan nilai akurasi sebesar 87%, nilai presisi sebesar 89% dan nilai recall sebesar 87%. Penelitian ini menggunakan dataset yang berjumlah sebanyak 950 data dengan pembagian data latih sebanyak 75% dari total keseluruhan data dan data uji sebanyak 25% dari total keseluruhan data.
Keaslian/ *state of the art*: Penelitian ini memiliki perbedaan dalam hal jenis data yang digunakan adalah data teks pesan yang terbagi ke dalam 5 kategori, nilai parameter C dan *gamma* yang digunakan, dan hasil yang diperoleh jika dibandingkan dengan penelitian sebelumnya.

1. Pendahuluan

Pada era teknologi yang sudah berkembang pesat seperti saat ini, pelayanan informasi secara konvensional dinilai kurang efektif dan interaktif. Seperti halnya pelayanan informasi pada Jurusan Informatika UPN "Veteran" Yogyakarta, yang pada saat ini masih dilakukan dengan cara-cara konvensional, seperti menggunakan *website*, papan pengumuman, surat edaran, dan sebagainya, menimbulkan beberapa permasalahan, yaitu sering kali mahasiswa menanyakan informasi secara berulang kepada pengurus jurusan yang sebenarnya informasi tersebut sudah disampaikan/disebarkan, waktu pelayanan administratif yang hanya terbuka pada saat jam kerja, dan menumpuknya pesan yang belum terbalas ketika mahasiswa mengirim dalam waktu yang bersamaan sehingga pesan yang dikirimkan oleh mahasiswa sering kali tidak terbalas. Oleh karena itu, otomatisasi pelayanan informasi seperti *chatbot* perlu dilakukan sebagai solusi dari permasalahan yang sudah disebutkan.

Chatbot merupakan sebuah program komputer yang dirancang untuk membalas pesan secara otomatis [1]. Pada awalnya, sistem *chatbot* dikembangkan dengan menggunakan metode *pattern matching*, seperti pada penelitian Paliwahet, Sukarsa & Putra yang menggunakan metode Fulltext Search Boolean Mode dari MySQL [2] dan penelitian Christianto, Siswanto & Chaniago yang menggunakan *Artificial Intelligence Markup Language* (AIML) dan *Named Entity Recognition* (NER) [3]. Pada penelitian tersebut, penggunaan *pattern matching* memiliki beberapa kekurangan, seperti respon yang dihasilkan oleh *chatbot* sangat berpengaruh terhadap pola-pola kalimat/pertanyaan yang sudah didaftarkan, sehingga pola kalimat yang didaftarkan harus beragam dan jika pertanyaan tidak terdaftar di *knowloedge-based* maka *chatbot* tidak dapat menjawab pertanyaan yang diajukan. Oleh karena itu, selanjutnya *chatbot* dikembangkan dan dikombinasikan menggunakan pendekatan *Natural Language Processing* (NLP), *text preprocessing*, dan beberapa algoritma *machine learning* lainnya [4]. Dengan menggunakan pendekatan NLP, *chatbot* menjadi lebih canggih dan dapat meningkatkan daya tanggap dari

chatbot tersebut [5]. Selain itu, NLP juga berperan penting untuk membantu mesin memahami query yang diinputkan oleh pengguna pada bahasa alami [6].

Dalam penggunaan pendekatan NLP, sebuah sistem harus dapat memahami informasi yang diberikan oleh pengguna pada bahasa alami. Salah satu cara untuk mengekstrak dari bahasa alami adalah dengan menentukan fungsi dari teks/kalimat dengan melakukan klasifikasi *intent* (maksud/keinginan) dari pengguna sehingga nantinya *chatbot* dapat merespon sesuai dengan pesan yang dikirimkan [7]. Pemilihan metode klasifikasi yang baik dan sesuai ketika melakukan klasifikasi *intent* sangat diperlukan agar sistem chatbot dapat berjalan dengan lancar. Beberapa metode dan algoritma klasifikasi machine learning sudah digunakan, di antaranya adalah *Naive Bayes Classifier* (NBC), *Neural Network* (NN), *K-Nearest Neighbor* (KNN), dan *Support Vector Machine* (SVM). Algoritma NBC memiliki kelebihan pada waktu pembuatan model yang relatif singkat dan mudah, tetapi algoritma ini sangat bergantung terhadap jumlah data pada setiap kelasnya. Sehingga jika dataset yang dimiliki terbatas, akurasi yang akan dihasilkan menjadi berkurang [8]. Penggunaan NN pada klasifikasi teks menghasilkan rata-rata hasil pengujian yang memuaskan dalam segi klasifikasi, tetapi NN cenderung memiliki waktu pembelajaran yang lama dan memerlukan banyak pengaturan pada parameternya [9]. KNN merupakan algoritma yang sederhana dan sangat mudah untuk dimengerti dan diimplementasikan pada klasifikasi, tetapi algoritma ini kesulitan untuk mencari nilai K yang sesuai [10].

Disisi lain, SVM merupakan salah satu algoritma klasifikasi pada machine learning yang kuat dan cenderung lebih akurat. Selain itu, SVM memiliki beberapa kelebihan, yaitu tidak rentan untuk terjadi overfitting ketika melakukan training [11] dan memberikan solusi yang bersifat global optimal atau hasil yang cenderung sama untuk setiap percobaan [12]. Penggunaan SVM sebagai algoritma klasifikasi sudah banyak diterapkan, seperti pada penelitian Luqyana, Cholissodin, & Perdana yang menerapkan SVM untuk analisis sentimen cyberbullying dan mendapatkan hasil pengujian yang memuaskan dengan nilai akurasi sebesar 90%, precision sebesar 94,44%, recall sebesar 85%, dan f-measure sebesar 89,47% ketika menggunakan Lexicon Based Features [13]. Selain itu, penelitian Tineges, Triayudi, & Sholihati juga menggunakan SVM untuk analisis sentimen dan mendapatkan akurasi sebesar 87%, precision sebesar 86%, recall sebesar 95%, dan f1-score sebesar 90% [14]. Selain untuk penerapan pada analisis sentimen, SVM juga dapat digunakan untuk melakukan klasifikasi intent, seperti yang dilakukan pada penelitian Maharani & Permatasari dengan membandingkan algoritma SVM dengan algoritma klasifikasi lainnya. Hasilnya adalah algoritma SVM memiliki nilai akurasi tertinggi di angka 97,5% dengan menggunakan ekstraksi fitur TF-IDF [15].

Dari beberapa penelitian sebelumnya, SVM dibuktikan sebagai algoritma yang lebih optimal dibandingkan dengan algoritma lainnya. Tetapi, SVM memiliki kelemahan yaitu hanya dapat berjalan dengan optimal ketika melakukan klasifikasi dua kelas (*binary classifications*) [16]. Pada penelitian yang berkaitan dengan chatbot, perlu dikembangkan metode SVM yang dapat menangani kasus klasifikasi nonbiner (klasifikasi lebih dari dua kelas), salah satunya yaitu dengan menggunakan pendekatan *Multiclass Support Vector Machine* (*Multiclass* SVM). Penggunaan pendekatan *multiclass* pada algoritma SVM sangat membantu untuk meningkatkan performa SVM ketika diterapkan untuk melakukan klasifikasi nonbiner. Penelitian tentang penggunaan algoritma *multiclass* SVM telah dilakukan pada penelitian Delimayanti, dkk. yang mendapatkan tingkat akurasi 87,03% dengan kernel RBF yang berbasis pada Gaussian dan

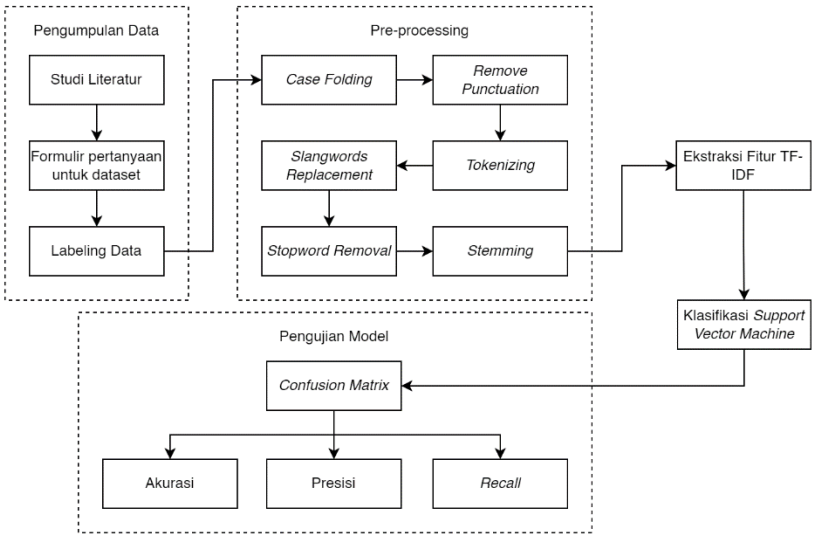
ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

menggunakan metode *One Versus All* (OVA) [17]. Penelitian lainnya yang menggunakan SVM dengan kernel yang sama dilakukan oleh Kusumahadi, Junaedi & Santoso untuk melakukan klasifikasi tiket helpdesk mendapatkan akurasi sebesar 81% dengan pendekatan *multiclass* OVA [12].

Berdasarkan uraian yang sudah dipaparkan di atas, penelitian ini menyediakan solusi dari permasalahan yang ada dengan melakukan otomatisasi terhadap layanan informasi berbasis NLP di jurusan Informatika UPN "Veteran" Yogyakarta pada aplikasi chat Telegram. Selain itu, penelitian ini juga menerapkan algoritma *Multiclass Support Vector Machine* (SVM) sebagai algoritma klasifikasi pesan pada chatbot dan menggunakan data yang berasal dari kumpulan pertanyaan terkait dengan informasi seputar akademik dan administrasi yang ditanyakan oleh mahasiswa jurusan Informatika UPN "Veteran" Yogyakarta. Penelitian ini diharapkan dapat mengetahui bagaimana performa dari algoritma *Multiclass Support Vector Machine* ketika diterapkan untuk melakukan klasifikasi pesan pada chatbot yang memiliki lebih dari dua kelas, membantu meringankan pekerjaan pengurus jurusan ketika ingin memberikan informasi kepada seluruh mahasiswa, serta dapat memudahkan mahasiswa dalam mengakses informasi yang ada di jurusan Informatika UPN "Veteran" Yogyakarta.

2. Metode Penelitian

Metode penelitian yang digunakan pada penelitian ini yaitu metode penelitian kuantitatif yang berawal dari pengumpulan data, preprocessing, ekstraksi fitur TF-IDF, klasifikasi Support Vector Machine, dan pengujian model. Adapun alur dari tahapan penelitian dapat dilihat pada **Gambar 1**.



Gambar 1. Metode Penelitian

2.1. Pengumpulan Data

Pengumpulan data pada penelitian ini menggunakan metode kuesioner. Metode ini dilakukan dengan membagikan sebuah formulir yang berisi pertanyaan-pertanyaan seputar hal yang

berkaitan dengan penelitian ini. Responden dari formulir ini adalah mahasiswa jurusan Informatika UPN "Veteran" Yogyakarta. Contoh pertanyaan pada formulir yang disebar dapat dilihat di **Tabel 1**.

Tabel 1. Pertanyaan pada formulir pengumpulan data

No	Pertanyaan
1	Tuliskan kalimat permohonan/pertanyaan yang ditujukan kepada Ketua Jurusan, contoh: "Permohonan permintaan tanda tangan untuk seminar" (boleh lebih dari satu kalimat)
2	Tuliskan kalimat permohonan/pertanyaan yang ditujukan kepada Koordinator Program Studi (boleh lebih dari satu kalimat)
3	Tuliskan kalimat permohonan/pertanyaan yang ditujukan kepada Dosen, termasuk dosen wali atau dosen pembimbing (boleh lebih dari satu kalimat)
4	Tuliskan kalimat permohonan/pertanyaan yang berkaitan dengan Administratif/Tata Usaha, contoh: "Permohonan transkrip nilai" (boleh lebih dari satu kalimat)

Hasil pengumpulan data dari formulir yang disebar tersebut selanjutnya akan kategorikan menjadi beberapa kategori yang dapat dilihat pada **Tabel 2** berikut.

Tabel 2. Kategori pesan mahasiswa

Kategori	Deskripsi
Administratif	Pesan yang berkaitan dengan pelayanan mahasiswa, seperti transkrip nilai, pengurusan cuti, dan permintaan tanda-tangan, dan sebagainya.
Dokumen	Pesan yang terkait dengan permintaan dokumen yang dibutuhkan mahasiswa, seperti form pendaftaran seminar, blanko cuti akademik, dll.
Kegiatan	Pesan yang berkaitan dengan kegiatan pada masa perkuliahan, seperti Tugas Akhir, Kerja Praktik, dan sebagainya
Jadwal	Pesan yang berkaitan dengan waktu pelaksanaan kegiatan perkuliahan dan akademik, seperti jadwal kuliah, kalender akademik, jadwal sidang, dan sebagainya
Sapaan	Pesan yang berkaitan dengan salam pembukaan maupun penutup, seperti Halo, selamat pagi, Assalamualaikum, terima kasih, dan sebagainya.

Berdasarkan kategori yang sudah dijelaskan pada **Tabel 2**, data yang sudah dikumpulkan dari hasil penyebaran formulir akan diperiksa kembali dan dilakukan pelabelan secara manual. Setelah itu, data tersebut digabungkan dengan dataset dari penelitian sebelumnya, namun tidak semua data dari dataset penelitian sebelumnya akan ditambahkan. Dataset dari penelitian sebelumnya akan diseleksi terlebih dahulu dan akan dipilih data-data yang relevan dengan data yang akan digunakan pada penelitian ini. Setelah semua data sudah diseleksi dan digabungkan, data yang terkumpul memiliki total jumlah 950 data. Contoh data yang sudah dikumpulkan dan labelnya dapat dilihat pada **Tabel 3** berikut.

Tabel 3. Contoh data yang telah dikumpulkan

No	Pesan	Kategori
D1	Permisi mau minta transkrip nilai NIM 123170028	Administratif
D2	Kapan jadwal sidang tugas akhir dilaksanakan?	Jadwal
D3	Permisi, bagaimana alur pelaksanaan kerja praktik?	Kegiatan
D4	Selamat pagi!	Sapaan

2.2. Proses *Text Preprocessing*

Proses *Text Preprocessing* merupakan sebuah tahapan yang dilakukan untuk membersihkan dan menyiapkan data pertanyaan yang sudah diperoleh sehingga nantinya akan siap ketika digunakan untuk klasifikasi. Tahapan preprocessing yang akan dilakukan pada penelitian ini meliputi *case folding*, *remove punctuation*, *tokenizing*, *slangwords replacement*, *stopword removal*, dan *stemming*. Proses preprocessing ini menerima input berupa dataset dan akan menghasilkan output hasil *preprocessing* nya.

2.2.1. Proses *Case Folding*

Proses *Case folding* bertujuan untuk mengubah semua huruf yang ada pada kalimat atau dokumen ke dalam satu jenis, baik menjadi huruf kapital maupun huruf nonkapital. Proses case folding sendiri, yaitu dengan melakukan pengecekan setiap huruf dari awal sampai akhir pada kalimat/dokumen menggunakan perulangan di setiap huruf dan mengubahnya menjadi huruf nonkapital dengan fungsi .lower() apabila huruf tersebut berjenis kapital.

2.2.2. Proses *Remove Punctuation*

Proses *Remove Punctuation* bertujuan untuk menghilangkan seluruh tanda baca, seperti titik (.), koma (,), tanda tanya (?), tanda seru (!), dan sebagainya. Data hasil *case folding* selanjutnya akan dilakukan pengecekan lebih lanjut untuk mengidentifikasi apakah terdapat tanda baca pada kalimat tersebut. Jika terdapat sebuah karakter tanda baca ataupun simbol, karakter tersebut akan dihapus.

2.2.3. Proses *Tokenizing*

Proses *Tokenizing* bertujuan untuk memisahkan kalimat berdasarkan kata penyusunnya menggunakan whitespace, seperti spasi (“ “). Data yang digunakan merupakan data hasil dari proses *remove punctuation* dan akan menghasilkan data yang berbentuk kata (token).

2.2.4. Proses *Slangwords Replacement*

Proses *Slangwords Replacement* bertujuan untuk mengubah kata-kata tidak baku di dalam teks menjadi kata-kata yang bersifat baku. Proses ini akan melakukan pengecekan dari data hasil proses *tokenizing* dan akan membandingkannya dengan kata-kata yang berada pada kamus slangwords. Jika kata yang akan dicek terdapat di dalam kamus, maka kata terebut akan diganti dengan kata bakunya yang terdapat di dalam kamus.

2.2.5. Proses *Stopword Removal*

Proses *Stopword Removal* bertujuan untuk membuang kata-kata yang tidak penting atau tidak memiliki arti tertentu, seperti kata penghubung dan kata ganti orang. Proses ini akan melakukan pengecekan kata-kata hasil dari proses *slangwords replacement* dan akan dibandingkan dengan kata-kata yang ada di kamus stopword. Jika kata hasil tokenizing terdapat di kamus, kata tersebut akan dibuang/dihapus.

2.2.6. Proses *Stemming*

Proses *Stemming* merupakan proses terakhir yang ada pada tahap preprocessing. Proses ini bertujuan untuk mengubah kata-kata hasil dari proses *stopword removal* ke dalam bentuk dasar dari kata tersebut dengan menghilangkan imbuhannya.

2.3. Proses Ekstraksi Fitur dengan TF-IDF

TF-IDF (*Term Frequency Inverse Document Frequency*) adalah sebuah cara yang digunakan untuk menghitung nilai frekuensi sebuah term atau kata pada sebuah dokumen atau banyak dokumen [12]. Dalam ruang vektor, bobot dari term direpresentasikan oleh term-frequency

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

matrix pada **Gambar 2**, dimana D merepresentasikan dokumen, T merepresentasikan term, dan w merepresentasikan bobot masing-masing term T yang ada di dokumen D

$$\begin{bmatrix} & T_1 & T_2 & \cdots & T_t \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} \end{bmatrix}$$

Gambar 2. Term-Frequency matrix [12]

Berikut merupakan contoh analisis ekstraksi fitur dengan TF-IDF terhadap contoh dokumen yang sudah dilakukan *text preprocessing* sebelumnya.

D1 : [‘permisi’, ‘transkrip’, ‘nilai’, ‘nim’]

D2 : [‘jadwal’, ‘sidang’, ‘tugas’, ‘laksana’]

D3 : [‘permisi’, ‘alur’, ‘laksana, ‘kerja’, ‘praktik’]

D4 : [‘selamat’, ‘pagi’]

Perhitungan *Term Frequency* (TF) bertujuan untuk mendapatkan jumlah setiap kata (*term*) pada suatu dokumen dengan menghitung frekuensi kemunculan *term* tersebut yang terdapat pada masing-masing dokumen. Sebagai contoh, kita akan menghitung TF dari *term* “transkrip”. *Term* “transkrip” muncul 1 kali pada dokumen 1 dan tidak muncul di 4 dokumen yang tersisa sehingga *term* “transkrip” bernilai 1 pada dokumen 1 dan bernilai 0 pada dokumen lainnya. Berikut penjabaran lebih lengkap kemunculan *term* “transkrip” pada setiap dokumen.

D1: *term*(“transkrip”) = 1

D2: *term*(“transkrip”) = 0

D3: *term*(“transkrip”) = 0

D4: *term*(“transkrip”) = 0

Document Frequency (DF) merupakan variabel yang bertujuan untuk menampung jumlah kemunculan suatu *term* pada keseluruhan dokumen (D). Sebagai contoh, *term* “transkrip” muncul 1 kali pada dokumen 1 dan tidak kembali muncul pada dokumen lainnya sehingga nilai *Document Frequency* (DF) adalah 1 pada dokumen (D). *Inverse Document Frequency* (IDF) merupakan jumlah dokumen yang ada pada korpus dibagi dengan jumlah dokumen yang mengandung *term*. IDF didapatkan dari **Persamaan 1**.

$$IDF_j = log\left(\frac{D}{df_j}\right) \quad (1)$$

Persamaan TF-IDF didapatkan dengan menggabungkan hasil TF dan **Persamaan 1** dengan perkalian, seperti pada **Persamaan 2**.

$$w_{ij} = TF\ x\ IDF \quad (2)$$

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

Hasil perhitungan nilai TF-IDF setiap *term* pada setiap dokumen lebih lengkapnya dapat dilihat pada **Tabel 4** berikut.

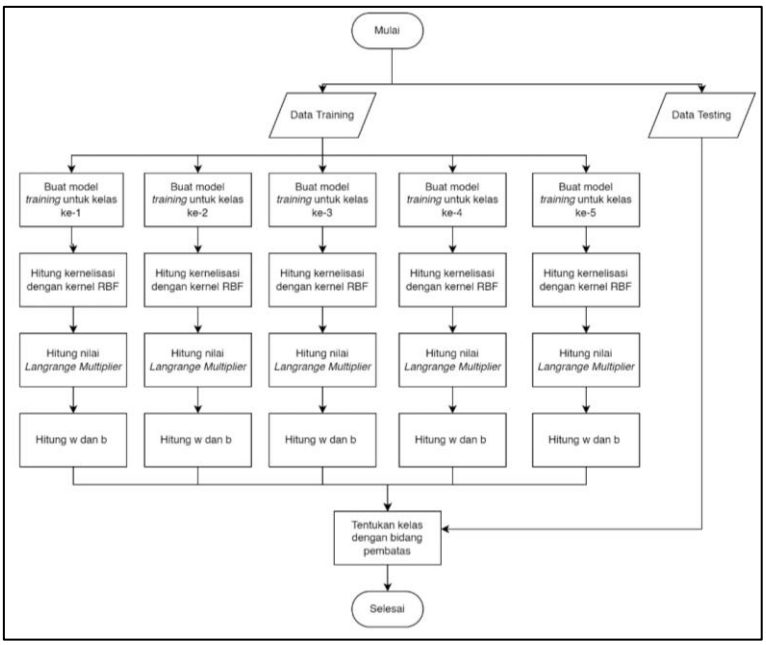
Tabel 4. Hasil perhitungan TF-IDF

Indeks	Term	TF-IDF			
		D1	D2	D3	D4
0	permisi	0.301	0	0.301	0
1	transkrip	0.602	0	0	0
2	nilai	0.602	0	0	0
3	nim	0.602	0	0	0
4	jadwal	0	0.602	0	0
5	sidang	0	0.602	0	0
6	tugas	0	0.602	0	0
7	laksana	0	0.301	0.301	0
8	alur	0	0	0.602	0
9	kerja	0	0	0.602	0
10	praktik	0	0	0.602	0
11	selamat	0	0	0	0.602
12	pagi	0	0	0	0.602

2.4. Proses Algortima Multiclass Support Vector Machine

Setelah mendapatkan hasil TF-IDF, langkah selanjutnya adalah mendapatkan hasil klasifikasi pesan chatbot menggunakan algoritma Multiclass Support Vector Machine (Multiclass SVM) dengan kernel Radial Basis Function (RBF). Selain itu, pendekatan multiclass yang akan digunakan di algoritma Multiclass SVM adalah pendekatan One Versus All (OVA). Pendekatan OVA dipilih karena menurut penelitian sebelumnya yang sudah pernah dilakukan, pendekatan OVA ini memiliki performa yang lebih baik daripada pendekatan OVO dan juga proses training pendekatan OVA dinilai lebih sedikit dan sederhana, yang hanya membagi jumlah proses training menjadi N model, dibandingkan dengan pendekatan OVO yang harus membagi jumlah proses training menjadi $\frac{N(N-1)}{2}$ model. Sebelum melakukan klasifikasi, dataset yang digunakan akan dibagi menjadi data train dan data test dengan perbandingan 80% untuk data train, dan 20% untuk data test. Gambaran alur klasifikasi Multiclass SVM dengan pendekatan OVA dapat dilihat pada **Gambar 3**.

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418



Gambar 3. *Flowchart* klasifikasi *multiclass* SVM

Langkah pertama dari proses pendekatan OVA adalah membagi permasalahan klasifikasi dan proses *training* menjadi N model sesuai dengan jumlah kategori dari dataset sampai didapatkan *hyperplane* dari masing-masing kategori. Selanjutnya, tahapan *training* pada setiap kelas yang akan dicontohkan dengan kategori administratif adalah sebagai berikut.

a. Membuat model *training* untuk kategori administratif dengan mengubah hasil perhitungan TF-IDF dari masing-masing dokumen menjadi notasi vektor, seperti berikut ini:
   D1: [0.301, 0.602, 0.602, 0.602, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   D2: [0, 0, 0, 0, 0.602, 0.602, 0.602, 0.301, 0, 0, 0, 0, 0]
   D3: [0.301, 0, 0, 0, 0, 0, 0, 0.301, 0.602, 0.602, 0.602, 0, 0]
   D4: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.602, 0.602]
   Setelah notasi vektor didapatkan, berikan label berupa $+1$ untuk dokumen yang termasuk ke dalam kategori administratif dan berikan label -1 untuk dokumen yang tidak termasuk ke dalam kategori administratif, seperti pada **Tabel 5** berikut:

Tabel 5. Data label masing-masing dokumen

Dokumen	Notasi vektor	y (kelas)
D1	[0.301, 0.602, 0.602, 0.602, 0, 0, 0, 0, 0, 0, 0, 0, 0]	+1
D2	[0, 0, 0, 0, 0.602, 0.602, 0.602, 0.301, 0, 0, 0, 0, 0]	-1
D3	[0.301, 0, 0, 0, 0, 0, 0, 0.301, 0.602, 0.602, 0.602, 0, 0]	-1
D4	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.602, 0.602]	-1

b. Langkah selanjutnya adalah menghitung nilai kernelisasi menggunakan kernel RBF dengan nilai parameter $\gamma = 0.01$. Perhitungan kernelisasi dilakukan mulai dari D1 sampai dengan D4 yang dimulai dengan $K(x_1.x_1)$ sampai $K(x_4.x_4)$. Misalkan dilakukan perhitungan $K(x_1.x_1)$:

$x_1 - x_1 = [0.301, 0.602, 0.602, 0.602, 0, 0, 0, 0, 0, 0, 0, 0, 0] -$

$[0.301, 0.602, 0.602, 0.602, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

$x_1 - x_1 = 0$

$K(x_1.x_1) = exp(-\gamma|x_1 - x_1|^2)$

$K(x_1.x_1) = exp(-0.01(0)^2)$

$K(x_1.x_1) = 1$

Setelah mendapat nilai dari $K(x_1.x_1)$, selanjutnya lakukan hal yang sama sampai nilai dari $K(x_4.x_4)$ ditemukan sehingga diperoleh matriks K seperti **Tabel 6** berikut.

Tabel 6. Matriks K

Dokumen	D1	D2	D3	D4
D1	1	0.834	0.840	0.864
D2	0.834	1	0.840	0.864
D3	0.840	0.840	1	0.858
D4	0.864	0.864	0.858	1

c. Setelah itu, dilakukan perhitungan untuk mencari nilai *Lagrange Multiplier* (a_i). Sebelum mencari nilai a_i, setiap pesan harus diubah menjadi ke dalam *support vector* menggunakan **Persamaan 3**.

$$S = \sqrt{x^2 + y^2 \leq 2} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} \quad (3)$$

$$S = \sqrt{1^2 + 1^2} = \sqrt{2} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Setelah mendapatkan nilai *support vector*, selanjutnya substitusikan nilai *support vector* ke dalam **Persamaan 4** dan setiap *support vector* diberikan nilai bias 1 untuk mendapatkan jarak prependikular terbaik dan membantu mendapatkan nilai dari *b* atau *hyperplane*. Sebagai contoh, dilakukan perhitungan untuk dokumen pertama seperti berikut.

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (4)$$

$$= a_1 \begin{bmatrix} x_i \\ y_i \\ bias \end{bmatrix}^T \times \begin{bmatrix} x_i \\ y_i \\ bias \end{bmatrix}$$

$$= a_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^T \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 3a_1$$

Setelah menghitung semua keseluruhan dokumen dari D1 sampai dengan D4, maka temukan a_i menggunakan **Persamaan 5**, sehingga bentuk persamaannya seperti berikut.

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (5)$$

$$3a_1 + 0.696a_2 + 0.706a_3 + 0.736a_4 = 1$$

$$0.696a_1 + 3a_2 + 2.491a_3 + 2.513a_4 = -1$$

$0.706a_1 + 2.491a_2 + 3a_3 + 2.520a_4 = -1$

$0.736a_1 + 2.513a_2 + 2.520a_3 + 3a_4 = -1$

Sehingga $a_1 = 0.451, a_2 = -0.154, a_3 = -0.160, a_4 = -0.181$

d. Langkah selanjutnya adalah menghitung nilai *w* dan *b* menggunakan **Persamaan 6** sebagai berikut.

$$w = \sum_{i=1}^{n} a_i S_i \quad (6)$$

$$w = a_1 S_1 + a_2 S_2 + a_3 S_3 + a_4 S_4$$

$$w = 0.451\begin{bmatrix}1\\1\\1\end{bmatrix} + \left(-0.154\begin{bmatrix}0.834\\-1\\1\end{bmatrix}\right) + \left(-0.160\begin{bmatrix}0.840\\-1\\1\end{bmatrix}\right) + \left(-0.181\begin{bmatrix}0.864\\-1\\1\end{bmatrix}\right)$$

$$w = \begin{bmatrix}0.451\\0.451\\0.451\end{bmatrix} + \begin{bmatrix}-0.128\\0.154\\-0.154\end{bmatrix} + \begin{bmatrix}-0.134\\0.160\\-0.160\end{bmatrix} + \begin{bmatrix}-0.156\\0.181\\-0.181\end{bmatrix}$$

$$w = \begin{bmatrix}0.033\\0.946\\-0.044\end{bmatrix}$$

$$w_1 = \begin{bmatrix}0.033\\0.946\end{bmatrix},\ b_1 = -0.044$$

e. Selanjutnya, ulangi tahap 1 – 4 untuk semua kelas pada dataset sampai masing-masing *hyperplane* kategori tersebut ditemukan. Hasil keseluruhan *hyperplane* dari setiap kategori dapat dilihat pada **Tabel 7.**

Tabel 7. Hasil *hyperplane* seluruh kelas

Kategori	Nilai *w*	Nilai *b*
Administratif	$\begin{bmatrix}0.033\\0.946\end{bmatrix}$	-0.044
Kegiatan	$\begin{bmatrix}-0.383\\5.588\end{bmatrix}$	-0.340
Jadwal	$\begin{bmatrix}2.081\\5.689\end{bmatrix}$	-0.353
Sapaan	$\begin{bmatrix}-1.822\\6.02\end{bmatrix}$	-0.396

2.5. Pengujian

Pengujian pada penelitian ini akan dilakukan menggunakan dataset yang sudah dikumpulkan yang berjumlah 950 data dengan pembagian data latih sebesar 75% dan data uji sebesar 25%. Metode pengujian yang dilakukan pada penelitian ini menggunakan *confusion matrix*. *Confusion matrix* merupakan salah satu metode untuk mengukur performa dari metode klasifikasi (Fitriana & Sibaroni, 2020). *Confusion matrix* akan membandingkan hasil klasifikasi yang dibuat oleh sistem dengan hasil klasifikasi yang sebenarnya menggunakan nilai *True Positive* (TP), dan *False Positive* (FP). Tabel *confusion matrix* untuk penelitian ini dapat dilihat pada **Tabel 8**.

Tabel 8. Rancangan *Confusion Matrix*

		Kelas Prediksi				**Total Kelas**
		Kelas 1	**Kelas 2**	**...**	**Kelas n**	
Kelas Aktual	Kelas 1	True Positive (TP)	False Positive (1,2)		False Positive (1,n)	Total(kelas-1)
	Kelas 2	False Positive (2,1)	True Positive (TP)	...	False Positive (2,n)	Total(kelas-2)
	....	...	...	...	...	...
	Kelas n	False Poisitve (n,1)	False Positive (n,2)	...	True Positive (TP)	Total(kelas-n)
Total Prediksi		Prediksi(kelas-1)	Prediksi(kelas-2)	...	Prediksi(kelas-n)	

Dari **Tabel 8**, dapat dihitung nilai akurasi, presisi, dan *recall* dengan perhitungan seperti berikut.

a. Akurasi

$$Akurasi = \frac{\sum_{i=1}^{n} TP(kelas-i)}{\sum_{i=1}^{n} Total(kelas-i)} \; X \; 100\% \quad (7)$$

b. Presisi

$$Presisi_{kelas-i} = \frac{TP(kelas-i)}{Prediksi(kelas-i)} \; X \; 100\% \quad (8)$$

c. *Recall*

$$Recall_{kelas-i} = \frac{TP(kelas-i)}{Total(kelas-i)} \; X \; 100\% \quad (9)$$

3. Hasil dan Pembahasan

3.1. Pencarian Kombinasi Nilai Parameter (C & γ) Terbaik

Proses pembuatan model algoritma *Multiclass Support Vector Machine* pada penelitian ini akan menggunakan kernel *Radial Basis Function* (RBF), dimana pada kernel tersebut membutuhkan dua parameter, yaitu parameter C dan parameter γ. Sedangkan untuk mendapatkan hasil yang optimal, perlu dilakukan pencarian kombinasi nilai yang terbaik dari masing-masing parameter tersebut. Untuk mencari kombinasi nilai parameter yang terbaik, dilakukan pembuatan *grid* parameter pada masing-masing kombinasi pasangan nilai parameter. Pasangan nilai dari kombinasi parameter C dan γ yang terbaik ditentukan dari kombinasi pasangan nilai parameter yang menghasilkan nilai skor tertinggi.

Skenario pencarian kombinasi nilai parameter dari parameter γ dan C terbaik dilakukan dengan melakukan proses *training* menggunakan data latih sebesar 75% dari jumlah keseluruhan data yang sudah dipisahkan serta menetapkan nilai dari masing-masing parameter dengan rentang nilai 10^{-2} sampai dengan 10^{2}. Pada skenario pertama, proses *training* dilakukan dengan menggunakan nilai parameter $C = 0.01$ dan nilai parameter $\gamma = 0.01, \gamma = 0.1, \gamma = 1, \gamma = 10,$ dan $\gamma = 100$. Setelah hasil skor dari skenario pertama ditemukan, kemudian dilanjutkan dengan mencari hasil skor dari skenario kedua, ketiga, dan seterusnya. Adapun masing-masing skenario pencarian beserta hasil skor dari semua skenario pencarian kombinasi parameter dengan *grid* dapat dilihat pada **Tabel 9**.

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v19i3.7418

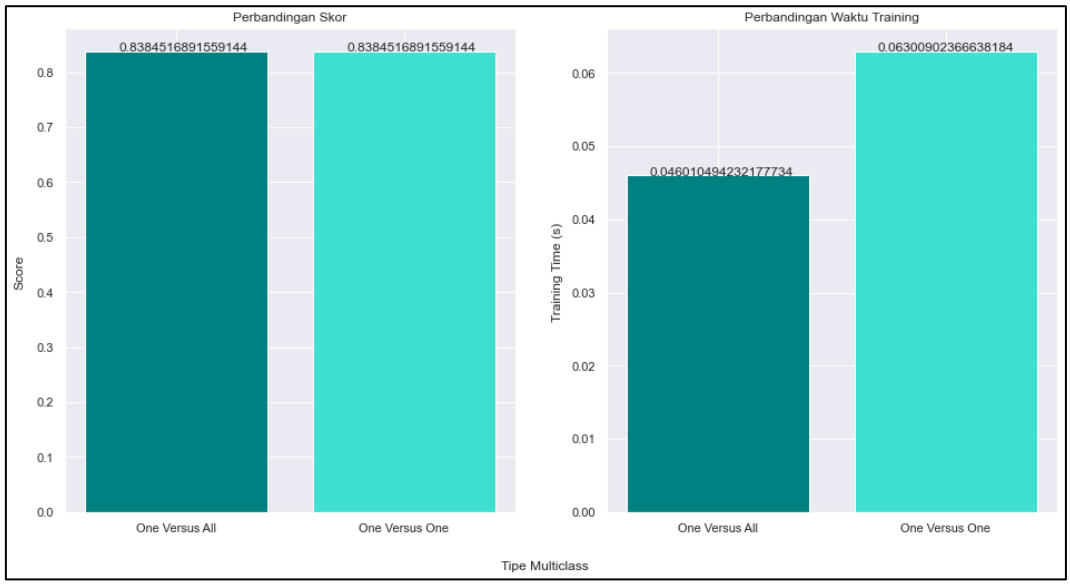
Tabel 9. Hasil skor tabel parameter *grid*

Skenario	Parameter C	Parameter Gamma				
		0.01	**0.1**	**1**	**10**	**100**
1	0.01	0.286	0.286	0.286	0.286	0.286
2	0.1	0.286	0.286	0.286	0.299	0.299
3	1	0.286	0.508	0.811	0.394	0.392
4	10	0.532	0.834	0.828	0.397	0.392
5	100	0.838	0.826	0.829	0.397	0.392

Dari **Tabel 9** dapat dilihat terdapat beberapa kombinasi pasangan parameter beserta skor yang dihasilkan oleh masing-masing kombinasi. Kombinasi pasangan parameter yang menghasilkan skor terbaik diperoleh oleh pasangan kombinasi parameter nilai $\gamma = 0.01$ dan nilai $C = 100$ dengan skor 0.838. Kombinasi parameter tersebut selanjutnya akan digunakan dalam melakukan pembuatan model algoritma yang akan digunakan pada tahap pengujian klasifikasi terhadap data uji.

3.2. Perbandingan Dua Pendekatan *Multiclass*

Selain menentukan kombinasi dari pasangan nilai parameter C dan γ yang cocok, penelitian ini akan melakukan perbandingan dari dua pendekatan *multiclass* yang ada terhadap proses *training* pembuatan model pada algoritma *Support Vector Machine*, yaitu pendekatan *One Versus All* (OVA) dan *One Versus One* (OVO). Proses *training* pada perbandingan ini juga akan menggunakan data latih yang sama dengan data latih pada pencarian pasangan kombinasi parameter terbaik dan menggunakan nilai parameter terbaik dari pencarian kombinasi parameter yang sebelumnya telah dilakukan, yaitu nilai parameter $\gamma = 0.01$ dan nilai parameter $C = 100$. Kedua pendekatan *multiclass* tersebut akan dibandingkan dari segi skor yang dihasilkan dan lamanya waktu untuk melakukan *training* model. Hasil perbandingan dari kedua pendekatan *multiclass* dapat dilihat pada **Gambar 4**.



Gambar 4. Perbandingan dua pendekatan *multiclass*

Pada **Gambar 4** dapat dilihat bahwa masing-masing dari pendekatan *multiclass* tidak menghasilkan perbedaan skor ketika melakukan proses *training* pembuatan model. Namun disisi lain, terdapat perbedaan dari segi lamanya waktu *training*. Pada **Gambar 4** terlihat bahwa pendekatan *multiclass* OVO menghasilkan waktu *training* yang lebih lama sekitar 0.02 detik dibandingkan dengan pendekatan *multiclass* OVA.

3.3. Pengujian *Confusion Matrix*

Setelah model algoritma *Multiclass Support Vector Machine* yang optimal selesai dibuat, model tersebut selanjutnya akan dilakukan pengujian. Teknik pengujian yang dilakukan pada penelitian ini adalah dengan menggunakan tabel *confusion matrix*. Data yang digunakan pada pengujian kali ini adalah data uji sebesar 25% dari dataset yang sudah didapatkan sebelumnya dengan total data sebanyak 238 baris data. Hasil pengujian *confusion matrix* dari prediksi model *Multiclass* SVM dapat dilihat pada **Tabel 10**.

Tabel 10. Tabel *confusion matrix*

		Kelas Prediksi					**Total Kelas**
		Administratif	**Dokumen**	**Jadwal**	**Kegiatan**	**Sapaan**	
Kelas Aktual	Administratif	53	2	2	2	2	61
	Dokumen	8	35	0	0	0	43
	Jadwal	4	0	35	0	1	40
	Kegiatan	7	1	1	46	0	55
	Sapaan	1	0	0	0	38	39
Total Prediksi		73	38	38	48	41	238

Tabel *confusion matrix* pada **Tabel 10** selanjutnya dapat digunakan sebagai acuan ketika melakukan perhitungan untuk mencari nilai akurasi, presisi dan *recall*. Dari hasil *confusion matrix* pada **Tabel 10**, nilai *True Positive* (TP) dan *False Positive* (FP) dari masing-masing kategori dapat diketahui. Hasil perhitungan nilai akurasi, pressisi, dan *recall* selengkapnya dapat dilihat pada **Tabel 11** berikut.

Tabel 11. Tabel hasil akurasi, presisi, dan *recall*

No	**Kategori**	**TP**	**FP**	**Akurasi**	**Presisi**	***Recall***
1	Administratif	53	20	-	0.726027	0.868852
2	Dokumen	35	3	-	0.921052	0.813953
3	Jadwal	35	3	-	0.921052	0.875000
4	Kegiatan	46	2	-	0.958333	0.836363
5	Sapaan	38	3	-	0.926829	0.974358
Keseluruhan		207	31	0.869747	0.890658	0.873705

Berdasarkan hasil pengujian yang sudah dilakukan, performa dari algoritma *Multiclass Support Vector Machine* (*Multiclass* SVM) dalam melakukan klasifikasi teks pesan pada *chatbot* jurusan Informatika UPN "Veteran" Yogyakarta dapat dikatakan sudah baik dan optimal. Hal tersebut dibuktikan dengan hasil pengujian menggunakan tabel *confusion matrix* yang mendapatkan nilai akurasi keseluruhan dari model algoritma *Multiclass Support Vector Machine* yang digunakan sebesar 0.869747 atau 87%, nilai presisi keseluruhan sebesar 0.890658 atau 89% dan nilai *recall* keseluruhan yang didapatkan sebesar 0.873705 atau 87%. Nilai-nilai tersebut didapatkan dari model *Multiclass* SVM dengan penggunaan kernel RBF, parameter nilai $C = 100$, parameter nilai $\gamma = 0.01$, dan penggunaan tipe *multiclass One Versus All* (OVA).

4. Kesimpulan dan Saran

Penelitian ini telah berhasil membangun sebuat chatbot layanan informasi Jurusan Informatika UPN "Veteran" Yogyakarta menggunakan pendekatan *Natural Language Prosessing* (NLP) dan algoritma *Multiclass Support Vector Machine* (*Multiclass* SVM) dalam melakukan klasifikasi teks pesan yang dikirimkan oleh *client* melalui aplikasi *chat* Telegram. Selain itu, algoritma *Multiclass* SVM dapat melakukan klasifikasi teks pesan yang dikirimkan dengan baik berdasarkan kategori-kategori pesan yang berkaitan dengan informasi yang ada di jurusan informasi dengan menunjukkan nilai akurasi sebesar 87%, nilai presisi sebesar 89% dan nilai *recall* sebesar 87%. Penelitian ini menggunakan dataset yang berjumlah sebanyak 950 data dengan pembagian data latih sebanyak 75% dari total keseluruhan data dan data uji sebanyak 25% dari total keseluruhan data

Adapun beberapa saran yang diusulkan sebagai pengembangan pada penelitian selanjutnya adalah algoritma *Multiclass Support Vector Machine* memerlukan beberapa kali percobaan dalam menentukan kombinasi pengaturan dari parameter yang cocok agar model yang dibuat dapat berjalan dengan optimal. Oleh karena itu, perlu dilakukan penentuan dari kombinasi parameter yang sesuai secara otomatis agar penelitian tidak terlalu lama dalam menemukan kombinasi parameter yang cocok untuk membangun model algoritma. Selain itu, pengembangan sistem chatbot selanjutnya perlu menambahkan jumlah dataset berupa berbagai variasi contoh kalimat pesan yang dikirimkan *client* agar nantinya *chatbot* dapat memberikan respon yang sesuai dengan pesan yang dikirimkan.

Daftar Pustaka

[1] D. G. S. Ruindungan and A. Jacobus, "Pengembangan Chatbot untuk Layanan Informasi Interaktif Akademik menggunakan Framework Rasa Open Source," vol. 10, no. 1, pp. 61–68, 2021.

[2] I. N. S. Paliwahet, I. M. Sukarsa, and I. K. Gede Darma Putra, "Pencarian Informasi Wisata Daerah Bali Menggunakan Teknologi Chatbot," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 8, no. 3, p. 144, 2017, doi: 10.24843/lkjiti.2017.v08.i03.p01.

[3] D. Christianto, E. Siswanto, and R. Chaniago, "Penggunaan Named Entity Recognition dan Artificial Intelligence Markup Language untuk Penerapan Chatbot Berbasis Teks," *J. Telemat.*, vol. 10, no. 2, p. 8, 2016.

[4] C. S. Rajender Kumar Surana, Shriya, Di. B. Gupta, and S. P. Shankar, "Intelligent Chatbot for Requirements Elicitation and Classification," *2019 4th IEEE Int. Conf. Recent Trends Electron. Information, Commun. Technol. RTEICT 2019 - Proc.*, pp. 866–870, 2019, doi: 10.1109/RTEICT46194.2019.9016907.

[5] S. S. Ranavare and R. S. Kamath, "Artificial Intelligence based Chatbot for Placement Activity at College Using DialogFlow," *Our Herit.*, vol. 68, no. 30, pp. 4806–4814, 2020.

[6] A. Elcholiqi and A. Musdholifah, "Chatbot in Bahasa Indonesia using NLP to Provide Banking Information," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 14, no. 1, p. 91, 2020, doi: 10.22146/ijccs.41289.

[7] J. Cahn, "CHATBOT : Architecture , Design , & Development," *Univ. Pennsylvania Sch.*

Eng. Appl. Sci. Dep. Comput. Inf. Sci., 2017.

[8] M. Y. H. Setyawan, R. M. Awangga, and S. R. Efendi, "Comparison Of Multinomial Naive Bayes Algorithm And Logistic Regression For Intent Classification In Chatbot," *Proc. 2018 Int. Conf. Appl. Eng. ICAE 2018*, pp. 1–5, 2018, doi: 10.1109/INCAE.2018.8579372.

[9] H. Bhavsar and A. Ganatra, "A Comparative Study of Training Algorithms for Supervised Machine Learning," *Int. J. Soft Comput. Eng.*, vol. 2, no. 4, pp. 74–81, 2012.

[10] K. A. Nugraha and Herlina, "Klasifikasi Pertanyaan Bidang Akademik Berdasarkan 5W1H menggunakan K-Nearest Neighbors," *JEPIN (Jurnal Edukasi dan Penelit. Inform.*, vol. 7, no. 1, pp. 44–51, 2021, doi: http://dx.doi.org/10.26418/jp.v7i1.45322.

[11] D. Mustafa Abdullah and A. Mohsin Abdulazeez, "Machine Learning Applications based on SVM Classification A Review," *Qubahan Acad. J.*, vol. 1, no. 2, pp. 81–90, 2021, doi: 10.48161/qaj.v1n2a50.

[12] S. Hilda Kusumahadi, H. Junaedi, and J. Santoso, "Klasifikasi Helpdesk Menggunakan Metode Support Vector Machine," *J. Inform. J. Pengemb. IT*, vol. 4, no. 1, pp. 54–60, 2019, doi: 10.30591/jpit.v4i1.1125.

[13] W. A. Luqyana, I. Cholissodin, and R. S. Perdana, "Analisis Sentimen Cyberbullying Pada Komentar Instagram dengan Metode Klasifikasi Support Vector Machine," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 11, pp. 4704–4713, 2018.

[14] R. Tineges, A. Triayudi, and I. D. Sholihati, "Analisis Sentimen Terhadap Layanan Indihome Berdasarkan Twitter Dengan Metode Klasifikasi Support Vector Machine (SVM)," *J. Media Inform. Budidarma*, vol. 4, no. 3, p. 650, 2020, doi: 10.30865/mib.v4i3.2181.

[15] D. A. Permatasari and D. A. Maharani, "Combination of Natural Language Understanding and Reinforcement Learning for Booking Bot," *J. Electr. Electron. Information, Commun. Technol.*, vol. 3, no. 1, p. 12, 2021, doi: 10.20961/jeeict.3.1.49818.

[16] Dhina Nur Fitriana and Yuliant Sibaroni, "Sentiment Analysis on KAI Twitter Post Using Multiclass Support Vector Machine (SVM)," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 5, pp. 846–853, 2020, doi: 10.29207/resti.v4i5.2231.

[17] M. K. Delimayanti, R. Sari, M. Laya, M. R. Faisal, and Pahrul, "Pengaruh Teknik Klasifikasi Pada Pesan Bencana Banjir Di Twitter Dengan Metode Multiclass-SVM," vol. 8, no. 1, p. 4, 2021, doi: https://doi.org/10.15294/edukomputika.v8i1.47858.