PERANCANGAN DAN PEMBUATAN PROTOTYPE MyBotS PADA MEDIA SOSIAL DISCORD SERTA KLASIFIKASI SVM

(Studi Kasus: Penjualan Game pada Platform Steam)

TUGAS AKHIR

Disusun Oleh :

Imam Al Maksur

16611064

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2020

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul : Perancangan Dan Pembuatan *Prototype* MyBotS Pada Media Sosial Discord Serta Klasifikasi SVM

Nama Mahasiswa : Imam Al Maksur

Nomor Mahasiswa : 16 611 064

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, 20 Agustus 2020

Pembimbing

(Muhammad Muhajir, S.Si., M.Sc.)

HALAMAN PENGESAHAN

TUGAS AKHIR

PERANCANGAN DAN PEMBUATAN PROTOTYPE MyBotS PADA MEDIA SOSIAL DISCORD SERTA KLASIFIKASI SVM

Nama Mahasiswa : Imam Al Maksur

Nomor Mahasiswa : 16 611 064

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL : 02 September 2020

Nama Penguji	**Tanda Tangan**
1. Dr.techn. Rohmatul Fajriyah, S.Si., M.Si.	:.....
2. Rahmadi Yotenka, S.Si.,M.Sc.	:.....
3. Muhammad Muhajir, S.Si.,M.Sc.	:.....

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Islam Indonesia

(Prof. Riyanto,M.Si., Ph.D.)

KATA PENGANTAR

Assalamu'alaikum Warahmatullaahi Wabarakaatuh

Alhamdulillaahirabbil'aalamiin, Puji Syukur kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah, dan inayahnya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **" Perancangan Dan Pembuatan Prototype MyBots Pada Media Sosial Discord Serta Klasifikasi SVM"** sebagai salah satu persyaratan yang harus dipenuhi dalam menyelesaikan jenjang strata satu di Program Studi Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Shalawat serta salam semoga selalu tercurah kepada junjungan nabi besar Muhammad SAW semoga mendapatkan safaatnya diakhir hayat nanti.

Proses penyelesaian tugas akhir ini tidak terlepas dari dukungan, bantuan, arahan, saran, dorongan dan bimbingan dari berbagai pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Kedua Orang tua, ayah Haji Efendi S.Ag dan ibu Zulfareni beserta ketiga saudaraku Hafizan, Fitri dan nur serta keluarga besar atas motivasi, semangat, dukungan dan doanya.
2. Bapak Fathul Wahid, S.T., M.Sc., Ph.D. selaku Rektor Universitas Islam Indonesia
3. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D. Selaku Dekan Fakultas dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia
4. Bapak Dr. Edy Widodo, S.Si., M.Si. selaku Ketua Jurusan Statistika, Fakultas dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia berserta jajarannya.

5. Bapak Muhammad Muhajir, S.Si.,M.Sc. selaku Dosen Pembimbing Tugas Akhir yang telah banyak bersabar dan meluangkan waktu serta berkorban untuk membimbing dan membantu dalam penyelesaian tugas akhir ini.
6. Ibu atina Ahdika, S.Si., M.Si. selaku dosen pembimbing akademik.
7. Seluruh staff, pengajar Program Studi Statistika Universitas Islam Indonesia yang telah memberikan bekal ilmu dan bantuan kepada saya.
8. Keluarga besar LASER-C (Islamic Science and Research Club), Kelompok Studi Fakultas (KSF) di Fakultas Matematika dan Ilmu Pengetahuan Alam.
9. Teman-teman satu bimbingan yang sudah sama-sama berjuang untuk menyelesaikan Tugas Akhir.
10. Teman-teman No Game No Life yang selalu mabar dari malam sampai subuh bahkan pernah bolos hanya untuk berimain game.
11. Teman-teman Statistika UII Angkatan 2016 yang bersama-sama menjadi pejuang gelar S.Stat.

Demikianlah yang dapat disampaikan, semoga Allah SWT senantiasa melimpahkan rahmat dan ridho-Nya kepada semua pihak yang telah membantu penulis. Semoga Tugas Akhir ini dapat memberikan manfaat bagi penulis khususnya dan umumnya bagi semua pihak yang membutuhkan. Akhir kata, semoga Allah SWT senantiasa melimpahkan rahmat serta hidayah-Nya kepada kita semua, Amin amin ya robbal ‘alamiin.

Wassalamu’alaikum Warahmatullaahi Wabarakaatuh

Yogyakarta, 20 Agustus 2020

Imam Al Maksur

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, Agustus 2020

Imam Al Maksur

INTISARI

PERANCANGAN DAN PEMBUATAN PROTOTYPE MyBotS PADA MEDIA SOSIAL DISCORD SERTA KLASIFIKASI SVM

(Studi Kasus: Penjualan Game pada Platform Steam)

Imam Al Maksur

Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Islam Indonesia

Teknologi terus berkembang pesat mengikuti perkembangan zaman, berbagai teknologi diciptakan untuk membantu manusia meringankan pekerjaan manusia. Pembeli atau pemain masih terlalu sedikit mendapatkan informasi terkait game di platform Steam, padahal informasi merupakan salah satu hal penting di era globalisasi saat ini. Semua kegiatan membutuhkan informasi, sehingga dapat dikatakan bahwa semua kegiatan yang dilakukan dituntut untuk menghasilkan informasi yang berguna bagi setiap pemain setia atau pembeli STEAM. Atas dasar permasalahan tersebut maka perlu dibuat suatu aplikasi yang lebih menarik dan interaktif yang memungkinkan pengguna berinteraksi dengan tanya jawab untuk mendapatkan informasi yaitu chatbot yang diimplementasikan di media sosial Discord. Kecerdasan Buatan adalah teknik yang memungkinkan mesin untuk berpikir dan dapat membuat keputusan sendiri. Prototipe chatbot Discord yang telah dibangun menyediakan berbagai layanan berdasarkan hasil pengujian klasifikasi menggunakan metode SVM dengan tiga kernel yaitu kernel Linear, Polynomial, dan RBF dengan data uji, prediksi nilai akurasi kategori SVM terbesar dengan kernel Linear, yang mana adalah 94% dan 6% kesalahan prediksi.

Kata Kunci : *Chatbot, Discord, Artificial Intelligence, NLP, SVM.*

ABSTRACT

DESIGN AND MANUFACTURE OF MYBOTS PROTOTYPE IN DISCORD SOCIAL MEDIA AND SVM CLASSIFICATION

(Case Study : Sales of Games on The Steam Platform)

Imam Al Maksur

Department of Statistics, Faculty of Mathematics and Natural Science
Universitas Islam Indonesia

Technology continues to develop rapidly following the times, various technologies are created to help humans ease human work. Buyers or players still get too little information related to games on the Steam platform, even though information is one of the important things in the current era of globalization. All activities require information, so it can be said that all activities carried out are required to produce useful information for every loyal player or buyer of STEAM. On the basis of this problem, it is necessary to make an application that is more attractive and interactive that allows users to interact with questions and answers to get information, namely a chatbot implemented on Discord social media. Artificial Intelligence is a technique that allows machines to think and be able to make their own decisions. The Discord chatbot prototype that has been built provides various services based on the results of classification testing using the SVM method with three kernels namely Linear, Polynomial, and RBF kernels with test data, category accuracy values prediction is the largest SVM with Linear kernel, which is 94% and 6% prediction error.

***Keywords**: Chatbot, Discord, Artificial Intelligence, NLP, SVM*

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Teknologi terus berkembang pesat mengikuti perkembangan zaman, berbagai teknologi tercipta guna membantu manusia meringankan perkerjaan manusia. Manusia dituntut untuk ikut berkembang dalam kemajuan teknologi tersebut. Kemajuan teknologi menciptakan sebuah industri baru yaitu industri video game yang juga tidak kalah menguntungkan dengan industri yang lainnya.

Video game pada saat ini sudah banyak tercipta dalam bentuk game digital. Game digital sangat diminati berbagai kalangan, baik dari anak anak sampai orang dewasa. Game digital bisa ditemukan pada media handphone maupun *personal computer* (PC). Video game terbagi menjadi dua yaitu game online dan offline. Game online membutuh jaringan internet sedangkan game ofline sebaliknya tidak butuh jaringan internet. Game online bisa dimainkan multiplayer atau bersama orang lain baik berjarak beda rumah, daerah maupun negara selama terdapat jaringan internet (KOMPASIANA, 2019).

Perkembangan industri video game yang menjanjikan membuat orang orang berlomba lomba menjadi developer atau biasa disebut pembuat game. Berbagai genre game dibuat oleh para developer baik genre horror, fps, rpg, galge, dll. Dengan berbagai konsep game yang menarik untuk menarik minat para player atau pembeli. Dengan adanya banyak game yang dibuat menjadi suatu masalah bagi para developer yaitu siapa yang akan menjual game-game tersebut.

Lalu munculah sebuah star-up yang membuat sebuah platform distributor game digital yang akan menjual game-game tersebut dengan sistem pajak pada penjualan melalui platform tersebut. Sehingga para developer tidak lagi bingung terkait penjulan game-game yang dibuatnya. Platform tersebut bernama STEAM. Kemunculan Steam

pertama adalah ketika salah satu developer game ternama bernama Valve Corporation yang memiliki masalah ketika ingin meng-update salah satu game onlinenya, Counter Strike, ketika ingin memberikan patch. Karena hal ini, Valve memutuskan untuk membuat sebuah platform yang memungkinkan game-game keluarannya di-update secara otomatis dan sekalian untuk mengimplimetasi kebijakan anti-bajakan dan juga anti-cheat.

Kemudian, Valve bekerja sama dengan beberapa perusahaan, di antaranya Acer, AT&T, dan juga GameSpy. Tidak hanya itu, Steam juga merilis mod. Mod pertama yang dirilis oleh Steam adalah mod untuk Half-Life pertama bernama Day of Defeat. Di tahun 2003, sekitar 80,000 sampai 300,000 pemain berpartisipasi versi betanya setelah client officialnya dirilis pada 12 September, 2003. Dua tahun kemudian, beberapa publisher besar seperti Capcom, Eidos Software, dan id Software mulai mendistribusikan game mereka di Steam di tahun 2007. Tidak disangka, 13 juta akun dibuat di Steam pada bulan Mei di tahun tersebut. Tidak hanya itu, juga 150 game dijual di platform tersebut. Penjualan game di platform ini semakin profitable, bahkan di tahun 2014 penjualan game terbaik di Steam diestimasikan sekitar $ 1.5 milyar. (Takur, 2019).

Menurut data dari situs *store.steampowered.com* tahun 2019, *game* dengan jumlah penjualan terbaik pada tahun 2019 terbagi menjadi 4 kategori yaitu *Bronze, silver, gold* dan *platinum*. Berikut peringkat aplikasi terpopuler berdasarkan kategorik

Tabel 1.1. *Daftar Penjualan Game Berdasarkan Kategorik pada Tahun 2019*

No	Bronze	Silver	Gold	Platinum
1	Cyberpun 2077	ARMA III	Rocket League	Dota 2
2	RAGE 2	Euro Truck Simulator 2	Assasin's Cread Odyssesy	Monster Hunter: World
3	AOE II: D.E	Team Fotress 2	Star wars jedi : Fallen order	PUBG

4	NBA 2K19	Planet Zoo	Devil May Cry 5	WARFRAME
5	American Truck Simulator	Borderlands2	Halo : The Master Chief Collection	Tom Clancy's R6 Siege
6	Insurgncy: Sandstrom	Rust	The Witcher 3	GTA V
7	theHunter: Call of the wild	ARK: Survival Evolved	Total War : WARHAMMER II	Total War: Three Kingdoms
8	Hitman 2	War Thunder	MORDHAU	CSGO
9	No Man's sky	CODE VEIN	Dead by Daylight	Sekiro: Shadows
10	Crusader kings II	Stellaris	RE:2/ BIOHAZARD	The Elder Scrolls

Sumber : (Steam, 2019)

Berdasarkan **tabel 1.1** dapat dilihat bahwa pengkategorian game dari *bronze*, *silver*, *gold* dan *platinum* dilakukan berdasarkan tahun rilis atau peluncurannya game tersebut. walaupun game yang dirilis sudah sangat lama tetapi tetap menarik minat para pemain karena platform steam selain menjadi tempat jual beli game tapi juga bisa menjadi media para developer untuk meng-*upgrade* atau meng-*update* game yang dibuatnya menjadi lebih baik dan dengan konsep-konsep terbaru.

Kemudian muncullah suatu masalah yaitu Informasi terkait game pada platform Steam masih terlalu minim untuk didapatkan oleh pembeli atau pemain yang jika ingin mendapatkan informasi baik game yang baru rilis atau game yang baru *update* harus search atau mencari informasi secara manual dengan cara membuka aplikasi STEAM terlebih dahulu yang mana membutuhkan laptop atau pc. Bagaimana jika pemain berada di luar rumah atau lagi melakukan perjalanan.

Padahal informasi merupakan salah satu hal penting di era globalisasi pada saat ini. Semua aktivitas memerlukan informasi, sehingga bisa dikatakan bahwa semua

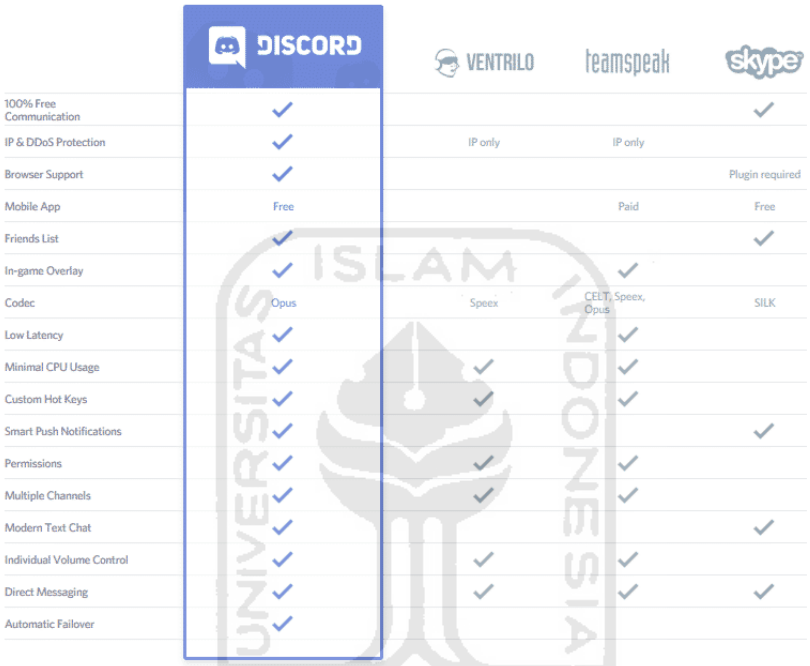
aktivitas yang dijalankan dituntut untuk menghasilkan informasi berguna bagi setiap pemain atau pembeli setia STEAM. Atas dasar masalah tersebut perlu dibuat sebuah aplikasi yang lebih menarik dan interaktif yang memungkinkan pengguna dapat melakukan interaksi tanya jawab untuk mendapatkan informasi seperti harga game, rating game, perilisan game baru dll mengenai *platform steam*, yaitu *chatbot* yang di implementasikan pada media sosial Discord.

Artificial Intelligence (*AI*) merupakan suatu teknik yang memungkinkan mesin untuk berpikir dan dapat mengambil keputusan sendiri. Dengan menggunakan kecerdasan buatan maka tidaklah mustahil akan ada mesin yang benar-benar mampu berpikir dan bertindak seperti manusia (Baiti, 2013) . Salah satu dari *AI* adalah *Natural Language Processing* (*NLP*). NLP adalah proses pembuatan model komputasi dari bahasa, sehingga dapat terjadi suatu interaksi antara manusia dan komputer dengan perantaraan bahasa alami (Sugianto,2005).

Bentuk aplikasi *Natural Language Processing* ini salah satunya adalah *chatbot* yang akan di implementasikan pada aplikasi Discord. Aplikasi Discord memiliki fitur-fitur keren untuk memudahkan para *gamer* saat saling berkomunikasi. Selain itu, aplikasi ini juga memiliki fakta menarik untuk diketahui. Sehingga para *gamer* bisa mengetahui apa saja kelebihan dan fakta aplikasi pendukung *gaming* selain dipakai untuk berkomunikasi dengan sesama pemain *game*. Layak nya Whatshapp, Telegram dan Skypea, Discord adalah aplikasi baru yang menawarkan fitur sejenis dengan desain yang simpel, praktis, mudah digunakan, menarik, dan dapat diakses dari berbagai gadget, laptop atau PC.

Discord adalah aplikasi gratis baik untuk pengguna android ataupun iphone. Pengguna bisa membuat server sendiri untuk voice chat dengan teman, saudara, keluarga atau bahkan pacar. Dan aplikasi ini aman karena menggunakan sistem Encryted. Jika pengguna ingin menjalankan aplikasi ini saat bermain game untuk memudahkan komunikasi sesama teman, aplikasi ini tidak akan berpengaruh terhadap

FPS game saat memainkannya. Karena pada dasarnya tujuan aplikasi dibuat adalah untuk bermain game sambil voice chat. Berikut sebuah gambar dari situs resminya https://discord.com/ yang membandingkan aplikasi voice chat ini dengan aplikasi voice chat lainnya seperti team speak, raidcall dan skype.



Gambar 1.1. Kelebihan aplikasi discord

Selain kelebihan yang ditampilkan pada **gambar 1.1** terdapat pembeda lain antara discord dengan aplikasi lainnya yaitu discord dapat dipasang bot seperti telegram atau Whatsapp, yang dimana bot pada discord bisa memutar musik, mencari gambar, mempunyai fitur level and credit dan masih banyak lagi sesuai fitur-fitur yang digunakan owner bot tersebut.

Data yang digunakan untuk klasifikasi berasal dari pertanyaan-pertanyaan *User* pada media sosial Discord, proses klasifikasi pada data pertanyaan berguna untuk melihat tingkat akurasi mesin, metode yang penulis gunakan dalam klasifikasi yaitu metode *Support Vector Machine* (SVM). Penulis memilih metode *Support Vector Machine* (SVM) karena merupakan salah satu metode yang cukup popular atau banyak

digunakan untuk melakukan klasifikasi data, terutama data teks (Asiyah & Fithriasari, 2016).

Berdasarkan latar belakang yang telah diuraikan, maka dalam penelitian ini akan dilakukan implementasi *Artificial Inteligence* berbentuk *prototype chatbot* yang dapat melakukan interaksi dengan manusia dan membantu manusia dengan menjawab pertanyaan yang diajukan seputar *Platform STEAM* pada aplikasi Discord. Sehingga akan dihasilkan percakapan antara pengguna dan program. Oleh karena itu penulis membuat penelitian yang berjudul "Perancangan dan Pembuatan Prototype MyBotS pada Media Sosial Discord Serta Klasifikasi SVM "

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka penulis merumuskan masalah yang akan dibahas yaitu:

1. Bagaimana gambaran umum data pertanyaan *Chatbot* dengan *User discord*?
2. Bagaimana cara memastikan program chatbot steam bisa menjawab pertanyaan yang diberikan *User discord* dibuktikan dengan hasil dari pengujian chatbot steam?
3. Bagaimana hasil dari klasifikasi interaksi user discord yang menggunakan bot steam menggunakan algorigma *Support Vector Machine*?

1.3 Batasan Masalah

Agar tujuan dalam penelitan ini tercapai tepat sasaran, maka penulis membatas masalah yang akan dibahas sebagai berikut:

1. Percapakan hanya akan membahas seputar informasi penjualan game pada platform steam, genre game, rating game, penjulan game terbaik, harga game, developer game, publisher game, informasi *event discount* akbar yang akan dilaksanakan steam, serta memainkan musik menggunakan bot didiscord.
2. Pertanyaan menggunakan Bahasa Inggris yang baik.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai penulis melalui penelitian dan penyusuan tugas akhir ini adalah:

1. Diketahuinya gambaran umum dari *prototype chatbot* steam
2. Mengetahui ketepatan jawaban pada pengujian program *bot* untuk memastikan program *chatbot* dapat menjawab pertanyaan yang diajukan oleh *user discord*.
3. Diketahuinya hasil klasifikasi *user discord* dengan menggunakan algoritma *Support Vector Machine*

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah dengan adanya rancangan prototype chatbot steam ini diharapkan dapat membantu *user* discord atau *player game* yang menggunakan discord mendapatkan informasi dengan mudah terkait penjualan game pada *platform* Steam sehingga tidak perlu lagi membuka *platform* steam yang mana membutuhkan waktu lebih lama dan *device* yang memadai hanya untuk mengetahui informasi tentang game lain-lainnya. Selain itu semoga dengan adanya chatbot ini bisa membantu industri game baik itu developer maupun publisher dan khusus nya platform penyedia jasa penjualan game.

BAB II
TINJAUAN PUSTAKA

Penelitian terdahulu menjadi acuan penting dalam penelitian, sebagai suatu kajian bagi peneliti dalam menulis untuk mengetahui hubungan antara penelitian sebelumnya dengan penelitian yang dilakukan saat ini dan menghindari unsur duplikasi dengan penelitian sebelumnya. Selain itu efek lainnya dari penelitian sebelumnya menyunjukan bahwa penelitian yang dilakukan mempunyai arti penting, sehingga dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan mampu memberikan solusi terhadap permasalahan pada aspek dengan teknologi. Berikut ini adalah beberapa penelitian terdahulu tentang implementasi *artificial intelligence* yang berkaitan dengan *Chatterbot*, Penggunaan *Natural Language Processing* dan klasifikasi SVM.

Penelitian mengenai implementasi *chatbot* pada bidang bisnis dilakukan oleh Eka Larasati dan Dimas Wahyu pada tahun 2019. Tujuan penelitiannya adalah Merancang *Chatbot* untuk meningkatkan performa Bisnis. Aplikasi *chatbot* yang dibangun menggunakan Bahasa sehari-hari yang merupakan media yang digunakan manusia untuk berkomunikasi. Penelitian ini memberikan hasil untuk meningkatkan usaha bisnis bahwa *chatbot* yang dibuat telah mampu membantu menjawab pertanyaan konsumen dengan cepat, pelacakan lokasi, penyimpanan data pesanan, pemrosesan, pencatatan pelanggan dan informasi lainnya.

Penelitian mengenai *Chatbot* pada tahun 2013 yang dilakukan oleh Evfi dan Yanti. Penelitian ini melakukan Analisa algoritma pemahaman kalimat pada *ALICE* (*Artificial Linguistic Internet Computer Entity*) pada Chatbot dengan menggunakan *Artificial Intelligence Markup Language* (AIML) sebagai penerjemah. AIML merupakan bahasa program yang diterapkan pada *ALICE ChatBot* dengan mengimplementasikan beberapa konsep penalaran yang ada pada *Artificial Intelligence.* Unsur-unsur AIML yang terdiri dari beberapa tag memiliki fungsi

masing-masing untuk kalimat dalam *knowledge base*-nya. Program *ALICE Chatbot* menggunakan bahasa pemrograman Java dalam pengembangannya dan menerapkannya dalam kumpulan *class – class* yang dikategorikan berdasarkan fungsionalitasnya. *Class* utama yang menjadi pusat *knowledge base*-nya adalah *class Graphmaster*. Dari penelitian ini akan menghasilkan beberapa algoritma seperti algoritma proses input normalization, sentence-splitting normalization dan pattern-fitting normalization, proses produksi jalur input, proses pemecah kalimat dan proses pencarian jawaban pada *knowledge base.*

Penelitian tentang *chatbot* dilakukan oleh Rico Savero pada tahun 2017, penelitian dilakukan untuk Merancang bangun Aplikasi *ChatBot* tentang penjualan mobil berbasis AIML menggunakan Algoritma Porter Stemmer. Program *ChatBot* penjualan mobil pada PT Permatahijau Automegah dibangun dengan *Artificial Intelligence Markup Language* (AIML) untuk mendefinisikan basis pengetahuan *chatbot* dan menggunakan algoritma Porter Stemmer untuk proses mengubah *input* kata kedalam bentuk kata dasar untuk menyederhanakan basis pengetahuan dari *chatbot*. Hasil dari penelitian ini adalah telah dibangun dan diuji dengan kata penerimaan (*acceptance*) mencapai 84,1%.

Penelitian mengenai penerapan Program *Chatbot* dilakukan oleh Fikri Haikal, Ucuk Darusalam, dan Andri pada tahun 2020. Penelitian ini menggunakan AIML yang merupakan bagian dari set XML. Dan juga sebagai Bahasa yang digunakan dalam Menyusun logika *chatbot* untuk memahami input yang telah diterima. Hasil dari penelitian penulis mengenai budidaya *chatbot* dalam memberikan motivasi belajar pada mahasiswa universita nasional beserta dengan diskusi yang telah dilakukan, dapat disimpulkan bahwa *chatbot* yang dibuat dapat membantu menjawab pertanyaan pengguna dengan cepat.

Penelitian mengenai perancangan Bot yang di terapkan pada aplikasi Discord dilakukan oleh Rizki, Sabriansyah, dan rizal pada tahun 2018. Penelitian dilakukan dengan merancang sebuah bot yang berjalan pada raspberry pi dan digunakan untuk melaukan pembacaan sensor yang ada pada raspberry pi dengan menggunakan sistem

learning yang dinamis. Pada penerapannya bot di berjalan di sosial media yang bernama Discord dan *user* dapat memberikan beberapa perintah pada kolom chat suatu server agar bot dapat berfungsi sesuai algoritma program. Pada implementasinya bot telah dilakukan pengujian fungsional dan setiap kasus yang diuji dapat berkerja dengan semestinya dan dinyatakan berhasil. Pada pengujian akurasi, didapatkan rata-rata persensetase error dengan menggunakan sensor LM35 yaitu 0,541% dan pada sensor HY-SRF05 yaitu 3,387%. Pada pengujian waktu pemrosesan dengan menggunakan sensor LM35 memiliki rata-rata waktu eksekusi perintah learn 545,54ms, perintah cmd 8,007ms dan perintah forgot 1,07ms. Pada pengujian waktu pemrosesan dengan menggunakan sensor ultrasonik HY-SRF05 memiliki rata-rata waktu eksekusi perintah learn 661,65ms, perintah cmd 5,42ms dan perintah forgot 1,24ms. Pada pengujian waktu pemrosesan dengan menggunakan sensor LDR memiliki rata-rata waktu eksekusi perintah learn 646,20ms, perintah cmd 5,43ms, dan perintah forgot 1,31ms.

Penelitian mengenai klasifikasi dokumen dilakukan oleh Niel dkk pada tahun 2018 menggunakan metode *Support Vector Machine*. Hasil uji Model SVM berbasis populasi sederhana dan estimasi area yang memberikan akurasi 57% dalam klasifikasi. Model berbasis variabel yang dibuat menggunakan hasil analisis kimia (jumlah pestisida yang terdeteksi di setiap sampel; jumlah total unit beracun, yang terkait dengan lebah dan mamalia; dan jumlah Analisis Mengenai Dampak Lingkungan, EIQ, lapangan dari setiap pestisida yang terdeteksi) memberikan akurasi inklasifikasi 100%.

Tabel 2.1. Penelitian terdahulu

Peneliti	Judul Penelitian	Metode Penelitian	Hasil
Omar Zahour, dkk. (2020)	*A system for educational and vocational guidance in Morocco: Chatbot E-Orientation*	*Chatbot* dan *Natural Language Processing*	*Chatbot* yang dapat melakukan percakapan sesuai dengan data set yang telah di buat

Eka Larasati Amalia dan Dimas Wahyu Wibowo	Rancangan Bangn Chatbot Untuk Meningkatkan Performa Bisnis	*Chatterbot*	meningkatkan usaha bisnis bahwa *chatbot* yang dibuat telah mampu membantu menjawab pertanyaan konsumen dengan cepat, pelacakan lokasi, penyimpanan data pesanan, pemrosesan, pencatatan pelanggan dan informasi lainnya.
Evfi Mahdiyah dan Yanti Andriyani	Analisa Algoritma Pemahaman Kalimat Pada *ALICE Chatbot* dengan menggunakan *Artificial Intelligence Markup Language (AIML)*	*Artificial Linguistic Internet Computer Entity* dan *Artificial Intelligence Markup Language*	menghasilkan beberapa algoritma seperti algoritma proses input normalization, sentence-splitting normalization dan pattern-fitting normalization, proses produksi jalur input, proses pemecah kalimat dan proses pencarian jawaban pada *knowledge base.*
Rico Savero (2017)	Rancangan Bangun Aplikasi *Chatbot* Penjualan Mobil Berbasis *Artificial Intelligence Markup Language* Menggunakan Algoritma Porter Stemmer	*Artificial Intelligence Markup Language*	Sistem telah dibangun dan diuji dengan angka penerimaan (*acceptance*) mencapai 84,1%
Muhammad Fikri Haikal Husein, dkk (2020)	*Aplication of the O-Chat Bot Program to Provide Learning*	*Chatbot* dan *Artificial Intelligence*	Hasil dari penelitian penulis mengenai budidaya *chatbot* dalam memberikan motivasi belajar pada

	Motivation to National Univertsity Students Using AIML	*Markup Language*	mahasiswa universita nasional beserta dengan diskusi yang telah dilakukan, dapat disimpulkan bahwa *chatbot* yang dibuat dapat membantu menjawab pertanyaan pengguna dengan cepat.
Rizki Septiansyah,dkk(2018)	Perancangan Bot Pada Discord Untuk Pembacaan Sensor Di Raspberry Pi Dengan Sistem Learning Yang Dinamis	*Chatterbot*	Pada pengujian waktu pemrosesan dengan menggunakan sensor LDR memiliki rata-rata waktu eksekusi perintah learn 646,20ms, perintah cmd 5,43ms, dan perintah forgot 1,31ms
S. Niell, dkk(2018)	*Beehives biomonitor pesticides in agroecosystems: Simple chemical and biological indicators evaluation using Support Vector Machines (SVM),*	*Support Vector Machines*	Model SVM berbasis populasi sederhana dan estimasi area yang memberikan akurasi 57% dalam klasifikasi. Model berbasis variabel yang dibuat menggunakan hasil analisis kimia (jumlah pestisida yang terdeteksi di setiap sampel; jumlah total unit beracun, yang terkait dengan lebah dan mamalia; dan jumlah Analisis Mengenai Dampak Lingkungan, EIQ, lapangan dari setiap pestisida yang terdeteksi) memberikan akurasi inklasifikasi 100%.

Berdasarkan tinjauan pustaka pada **tabel 2.1,** pada penelitian ini penulis menggunakan *NLP* untuk membuat program *chatbot* yang akan di implementasikan pada aplikasi media social *discord*. Kemudian program *chatbot* akan dilakukan klasifikasi menggunakan metode *support vector machine*. Serta menggunakan *AIML* sebagai metode pencocokan untuk mencari jawaban dari pertanyaan yang diajukan

BAB III
LANDASAN TEORI

3.1 Statistika Deskriptif

Menurut (Purwoto, 2007), analisis data deskriptif adalah cara mendeskripsikan atau menggambarkan data yang telah terkumpul sebagaimana adanya tanpa bermaksud membuat kesimpulan yang berlaku untuk umum/generalisasi. Ciri-ciri analisis data deskriptif yaitu penyajian data lebih ditekankan dalam bentuk tabel, grafik, dan ukuran-ukuran statistik seperti persentase, rata-rata, variansi, korelasi, dan angka indeks. Selain itu, analisis ini tidak menggunakan uji signifikansi dan taraf kesalahan karena tidak ada kesalahan generalisasi.

3.2 Kecerdasan Buatan (Artificial Intelligence)

Artifical Inteligence adalah kemampuan robot atau digital yang dikendalikan oleh komputer untuk melakukan tugas yang umumnya dikaitkan dengan mahkluk cerdas. *Artificial intelligence* berkaitan dengan sebuah mesin yang biasanya menyelesaikan tugas-tugas dengan melibatkan kecerdasan tertentu yang mana sebelumnya dianggap hanya dilakukan manusia. Simulasi proses kecerdasan buatan terutama sistem komputer meliputi pembelajaran, Analisa dan koreksi diri. (Chethan Kumar GN,2018)

Kecerdasan buatan atau *artificial intelligence* merupakan salah satu cabang disiplin ilmu yang berhubungan dengan pemanfaatan mesin untuk membantu manusia memecahkan persoalan dengan mudah. Hal ini biasanya dilakukan dengan mengikuti atau mencontohi karakteristik dan analogi berpikir dari kecerdasan atau *inteligensi* manusia dan menerapkannya sebagai algoritma yang dikenal oleh komputer. Dengan salah satu pendekatan yang lebih fleksibel dan efisien dapat diambil tergantung dari keperluan, yang mempengaruhi wujud dari perilaku kecerdasan buatan. AI biasanya diberkaitan dengan ilmu komputer, akan tetapi juga berkaitan erat dengan bidang ilmu lainnya seperti matematika, psikologi, ekonomi, industri dan lain-lainnya. Kemampuan

untuk mengkombinasikan pengetahuan pada akhirnya akan bermanfaat bagi kemajuan dalam upaya menciptakan suatu kecerdasan buatan. (Nugraha & Winiarti, 2014)

Kecerdasan buatan ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Bagian utama dari kecerdasan buatan adalah basis pengetahuan (*knowledge base*), yaitu suatu pengertian atau pemahaman tentang wilayah subjek yang diperoleh melalui pembelajaran dan pengalaman (Kristanto, 2003).

Kecerdasan buatan (*artificial inteligence*) adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berperilaku cerdas seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia. Aktivitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya (Simarmata, 2006).

3.3 Chatbot

Chatbot atau bisa disebut juga chatterbot atau bots adalah sebuah program komputer yang dirancang untuk menyimulasikan percakapan inteletual dengan manusia baik satu orang maupun lebih. Percakapan yang terjadi antara komputer dengan manusia adalah bentuk respon dari program pada database program pada komputer. Kemampuan komputer dalam menyimpan banyaknya data tanpa melupakan satu pun informasi yang disimpannya serta kemampuan learning yang dimilikinya membuat chatbot adalah customer service automatis yang bisa dihandalkan. (Raj & Suteja, 2019)

Chatbot adalah program komputer untuk mensimulasi percakapan yang didukung oleh aturan yang diprogram dan kecerdasan buatan (*Artificial Intelligent*). Chatbot bekerja dalam memberikan respon yang spesifik sesuai aturan dan tingkat kecanggihan yang diberikannya pada program. (Nugraha & Winiarti, 2014)

Pemanfaatan chatbot pada awalnya dilakukan di Tiongkok yang telah mulai sejak tahun 2013 yang difungsikan sebagi medium automasi pesan. Seiring berjalannya

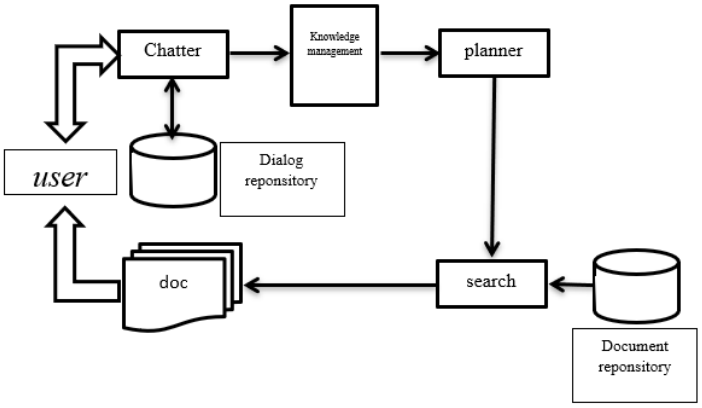
waktu, chatbot difungsikan oleh pelaku bisni dalam menjangkau konsumen dilingkup yang lebih personal. Sejak tahun 2017, teknologi chatbot Kembali marak dibicarakan di dunia teknologi dan startup digital hampir diseluruh dunia termasuk di Indonesia. Bahkan beberapa pelaku teknologi digital berbondong-bondong menerapkan chatbot kedalam platform mereka. Salah satu di antaranya adalah LINE. (Nugraha & Winiarti, 2014)

Para pelaku bisnis yang memanfaatkan teknologi chatbot bisa melakukan pelayanan kepada konsumen, melakukan poling, memberikan diskon atau potongan haraga, serta memberikan pilihan produk terbaik sesuai pilihan mereka. (Nugraha & Winiarti, 2014)

Kemampunan chatbot (Nugraha & Winiarti, 2014):

1. Pelayanan selama 24 jam
2. Respon yang cepat terhadap setiap pertanyaan
3. Kenyamanan pengguna
4. Kemudahan dalam berinteraksi
5. Pelayanan konsumen yang baik

Chatbot merupakan salah satu bentuk *bots*. *Chatbot* dapat memberikan jawaban sesuai dengan input pertanyaan atau keluhan dari user/pengguna. *Chatterbot* adalah sebutan untuk *robots chatting*. *Chatterbot* menggunakan kecerdasan buatan untuk mensimulasikan percakapan dengan penggunanya. *Chatterbot* dirancang untuk dapat mendekati sifat manusia. Konsep dasar *Chatterbot* yaitu: (*chatterbot.net*):



Gambar 3.1. Konsep dasar aplikasi *chatterbot* (sumber: www.chatterbot.net)

Salah satu *chat bot* yang terkenal adalah ELIZA (Dr. Eliza) yang dikembangkan oleh Joseph Weizenbaum di *MIT* (*Massachusetts Institute of Technology*). ELIZA mensimulasikan percakapan antara seorang *psikiater* dengan pasiennya dalam bahasa Inggris yang alami.

3.4 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) atau pengolahan bahasa alami merupakan salah satu bidang ilmu *Artificial Intelligence* (Kecerdasan Buatan) yang mempelajari komunikasi antara manusia dengan komputer melalui bahasa alami. *Natural Language Processing* (NLP) memodelkan pengetahuan terhadap bahasa, baik dari segi kata, bagaimana kata-kata bergabung menjadi suatu kalimat dan konteks kata dalam kalimat. Proses pembuatan model komputasi dari bahasa sehingga memungkikan terjadinya interaksi antara manusia dan komputer dengan perantaraan bahasa alami yang dipakai oleh manusia (Barakbah, 2005).

Bidang tersebut terutama berkaitan dengan membuat interaksi manusia dan komputer menjadi mudah tetapi efisien. Mesin mempelajari sintaks dan arti bahasa manusia, memprosesnya, dan memberikan keluaran kepada pengguna. Area NLP melibatkan pembuatan sistem komputer untuk melakukan tugas-tugas yang berarti dengan bahasa yang alami dan dapat dimengerti manusia. Alasan mengapa pemrosesan bahasa alami sangat penting di masa depan adalah membantu kita untuk membangun model dan proses yang mengambil potongan informasi sebagai input dan dalam bentuk suara atau teks atau keduanya dan memanipulasinya sesuai dengan algoritma di dalam komputer (Jain, Kulkarni, & Shah, 2018).

Natural Language Processing (NLP) tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk suara menjadi data digital dan atau sebaliknya pula, melainkan bertujuan untuk memahami arti dari ucapan yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu. Untuk mencapai tujuan

ini dibutuhkan tiga tahap proses. Proses yang pertama ialah parsing atau analisa sintaksis yang memeriksa kebenaran struktur kalimat berdasarkan suatu *grammar* (tata bahasa) dan *lexicon* (kosa kata) tertentu, *parsing* mempunyai 2 pendekatan yaitu *Top down parsing* dan *Bottom up parsing*. Proses kedua ialah interpretasi semantik yang bertujuan untuk merepresentasikan arti dari kalimat secara *context-independent* untuk keperluan lebih lanjut. Sedangkan proses ketiga ialah interpretasi kontekstual yang bertujuan untuk merepresentasikan arti secara *contextdependent* dan menentukan maksud dari penggunaan kalimat. (Suciadi, 2002)

Disiplin Ilmu *Natural Language Processing* (NLP) :

1. Fonetik/Fonologi

   Merupakan ilmu *Natural Language Processing* (NLP) yang berhubungan dengan suara yang menghasilkan kata untuk dapat dikenali. Bidang Fonetik/Fonologi dipakai dalam aplikasi-aplikasi yang menggunakan *Speech Based System.*

2. Morfologi

   Merupakan ilmu *Natural Language Processing* (NLP) yang memberikan pengetahuan tentang kata dan bentuknya sehingga bisa dibedakan antara yang satu dengan yang lain.

3. Sintaksis

   Merupakan ilmu *Natural Language Processing* (NLP) yang memberikan pengetahuan tentang urutan kata dalam pembentukan kalimat.

4. Semantik

   Merupakan ilmu *Natural Language Processing* (NLP) yang bertujuan untuk merepresentasikan arti dari kalimat atau suatu kata dan bagaimana arti kata tersebut membentuk suatu arti kata dalam kalimat.

5. Pragmatik

   Merupakan ilmu *Natural Language Processing* (NLP) yang berguna untuk memberi pengetahuan tentang konteks kata atau kalimat yang berhubungan erat keadaan atau situasi kata atau kalimat tersebut yang digunakan.

6. *Discourse Knowledge*

Merupakan ilmu *Natural Language Processing* (NLP) yang memberikan pengetahuan tentang hubungan antar kalimat serta melakukan pengenalan apakah suatu kalimat yang telah dikenali mempengaruhi kalimat selanjutnya. Penting untuk identifikasi kata ganti orang, keterangan tempat atau aspek sementara dari informasi.

7. *World Knowledge*

   Merupakan ilmu *Natural Language Processing* (NLP) yang berarti sebuah kata secara umum dan apakah arti khusus bagi suatu kata dalam suatu percakapan dengan konteks tertentu.

Aplikasi yang berbasis *Natural Language Processing* (NLP) terbagi menjadi 2 yaitu :

1. *Text-based application*

   Text-based application merupakan aplikasi *Natural Language Processing* (NLP) yang melakukan pemrosesan terhadap teks yang tertulis. Contoh :

   - Mencari topik tertentu dari buku di perpustakaan
   - Mencari isi dari suatu berita atau artikel
   - Mencari isi dari email
   - Menterjemahkan dikumen dari suatu bahasa ke bahasa lain
   - *Chatbot* berbasis *text*

2. *Speech-based application*

   Speech-based application merupakan aplikasi *Natural Language Processing* (NLP) yang melakukan pemrosesan dari bahasa lisan atau pengenalan suara. Contoh :

   - Sistem otomatis pelayanan melalui telepon
   - Kontrol suara pada peralatan elektronik
   - Aplikasi peningkatan kemampuan berbahasa
   - *Chatbot* berbasis *speech*

Aplikasi yang berbasis *Natural Language Processing* (NLP) yang sudah diciptakan yaitu :

ELIZA *Chatbot* (Aplikasi ini dibekali pengetahuan psikologi, sehingga beberapa orang terdorong untuk mampu merubah sikap dan perilakunya), Jupiter (Aplikasi ini mampu memberikan informasi cuaca melalui telepon, ALVIN *Chatbot* (Aplikasi yang mampu menjawab pertanyaan mengenai DOS), SEXPERT (Aplikasi yang dirancang untuk perbincangan mengenai pendidikan seksual), *Email translator*, *Web translator*, *World translator* (Barakbah, 2005).

3.5 *AIML*

Artificial Intelligence Markup Language merupakan turunan dari *markup language* XML berbasis *tag element* yang bertujuan untuk mempermudah permodelan dialog komunikasi berdasarkan paradigma *stimulus-response*. AIML mendefinisikan data objek yang disebut objek AIML yang digunakan untuk memodelkan pola input dan output dari suatu percakapan (Marientto dkk., 2013). AIML merepresentasikan *knowledge* yang digunakan oleh *chatbot* berdasarkan pengembangan teknologi *bot* ALICE(Kader, 2015)

```
<command> ListOfParameters </command>
```

Gambar 3.2. Sintaks Dasar AIML (Marietto dkk., 2013)

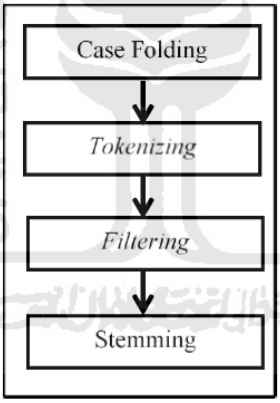
Sintaks dasar dari objek AIML ditunjukan pada Gambar 3.2 Menurut Marietto (2013) terdapat tiga tag utama dalam objek AIML yang perlu diperhatikan, yaitu *category* yang mendefinisikan suatu unit *knowledge* dari suatu percakapan, *pattern* yang mengidentifikasi input dari *user*, dan *template* yang digunakan untuk mencatat respon yang akan diberikan berdasarkan *input* tertentu dari *user*. *Pattern* ditulis dalam hutuf kapital standarisasi dan simplikasi proses *pattern matching* dari *input user*.

3.6 *Text Mining*

Dalam Damanik Ulina di tahun 2013, Menurut Ratna Maria *text mining* memiliki definisi menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, yang tujuannya adalah mencari kata-kata yang dapat

mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antara dokumen. *Text mining* meliputi ekstraksi dan penyimpanan berupa teks, preprosesing isi teks, pengumpulan data statistik serta pemberian indeks data dan analisa konten. *Text mining* mengekstrak indeks numerik yang bermakna dari teks dan kemudian informasi yang terkandung dalam teks akan diakses dengan menggunakan berbagai algoritma *data mining* (statistik dan *machine learning*) (Miner, 2012).

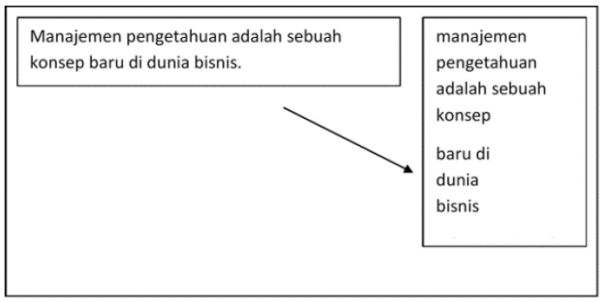
Pada umumnya proses *text mining* mengambil data berupa teks yang memiliki memiliki beberapa karakteristik diantaranya adalah berdimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Sebelum menentukan fitur-fitur yang mewakili, diperlukan tahap preprosesing yang dilakukan secara umum dalam *text mining* pada dokumen, yaitu *case folding*, *tokenizing*, *filtering*, dan *stemming* (Mooney, 2006).



Gambar 3.3. *Proses Text Mining*
(Mooney, 2006)

3.5.1. Case Folding

Case folding adalah proses mengubah semua huruf kapital dalam dokumen menjadi huruf kecil meliputi huruf "a" sampai dengan "z" dan menghilangkan karakter selain huruf "a" sampai dengan "z" karena dianggap sebagai delimeter.

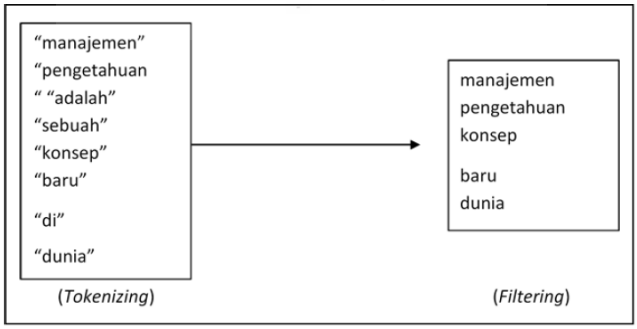


Gambar 3.4. Proses *Case Folding*

(Sumber: Ulina, 2014)

3.5.2. Tokenizing dan Filtering

Tokenizing adalah proses pemotngan *string input* di dalam teks berdasarkan kata-kata yang menyusunnya. *Tokenizing* menghasilkan seluruh kata-kata yangtelah dipisahkan dalam suatu kalimat. *Filtering* adalah proses pengambilan kata-kata yang penting atau yang dibutuhkan untuk dianalisis. Kemungkinan ada kata yang dihilangkan biasanya berupa kata penghubung dan kata-kata yang tidak diinginkan peneliti dalam teks tersebut. Hal ini bisa dilakukan menggunakan algoritma *stoplist*/*stopword* (membuang kata yang kurang penting sesuai dengan keinginan peneliti) atau *wordlist* (menyimpan kata-kata yang dianggap penting).



Gambar 3.5. Proses *Tokenizing* dan *Filtering*

(Sumber: Ulina, 2014)

3.7 *Data Training dan Data Testing*

Data umumnya dibagi menjadi data *training* dan data *testing*. Data *training* digunakan oleh algoritma klasifikasi (misalnya *Decision Tree*, bayesian, *Neural Network, SVM*) untuk membentuk sebuah model klasifikasi. Model ini merupakan representasi pengetahuan yang akan digunakan untuk mengukur sejauh mana klasifikasi berhasil melakukan klasifikasi dengan benar. Karena itu, data yang ada pada data *testing* seharusnya tidak boleh ada pada data *training* sehingga dapat diketahui apakah model klasifikasi dapat melakukan klasifikasi dengan baik. Proporsi antara data *training* dan data *testing* tidak mengikat tetapi agar variasi dalam model tidak terlalu besar maka disarankan data *training* lebih besar dibandingkan data *testing.* Biasanya 2/3 dari total data dijadikan data *training* sedangkan sisanya dijadikan data *testing.* Selain itu, ada pula penelitian yang menghasilkan keakuratan model klasifikasi optimum dengan proporsi 80:20 dan 90:10 untuk data *training* dan data *testing* (Paratu, 2003 dalam Estoatnowo, 2016).

3.8 Pembobotan Kata *Term Frequency–Inverse Document Frequency*

Pembobotan *term* diperlukan untuk mencari informasi dari koleksi dokumen yang bersifat beragam. *Term weight* perlu dilakukan pada setiap kata karena masing-masing kata memiliki tingkat kepentingan yang berbeda di dalam dokumen. Menurut Mandala (Fatah, 2016) dijelaskan bahwa *term weighting* atau pembobotan *term* sangat dipengaruhi oleh hal-hal berikut ini:

1. *Term Frequency* (*TF*) *factor*, yaitu faktor yang menentukan bobot *term* pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan suatu kata (*term frequency*) diperhitungkan dalam pemberian bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu *term* (*TF* tinggi) dalam dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuian yang semakin besar.
2. *Inverse Document Frequency* (*IDF*) *factor*, yaitu pengurangan dominansi *term* yang sering muncul di berbagai dokumen. Hal ini diperlukan karena *term* yang

banyak muncul di berbagai dokumen, dapat dianggap sebagai *term* umum (*common term*) sehingga tidak penting nilainya. Sebaliknya faktor kejarangmunculan kata (*term scarcity*) dalam koleksi dokumen harus diperhatikan dalam pemberian bobot. Menurut Mandala, kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon tems*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebalikan secara proposional dengan urutannya.

Menurut Defeng (Fattah, 2016), jenis formula yang akan digunakan untuk perhitungan *term frequency* (*TF*) yaitu *TF* murni (*raw TF*). *TF* menunjukkan seberapa sering sebuah kata muncul dalam suatu dokumen tertentu. Sedangkan *IDF* menjelaskan tentang seberapa penting arti sebuah kata dalam sebuah dokumen yang dirumuskan sebagai berikut:

$$IDF\ (A) = log\frac{D}{df_A} \tag{3.1}$$

Apabila nilai sama dengan maka akan didapatkan hasil nol untuk perhitungan *IDF* sehingga perlu ditambahkan nilai 1(+1) untuk rumus *IDF* menjadi

$$IDF\ (A) = log\frac{D}{df_A} + 1 \tag{3.2}$$

$$W(A) = TF(A)x\ IDF(A) \tag{3.3}$$

Keterangan :

W (A) = bobot *term* dari kata A

TF (A) = Jumlah Kata A dalam Dokumen

D = Jumlah Seluruh Dokumen

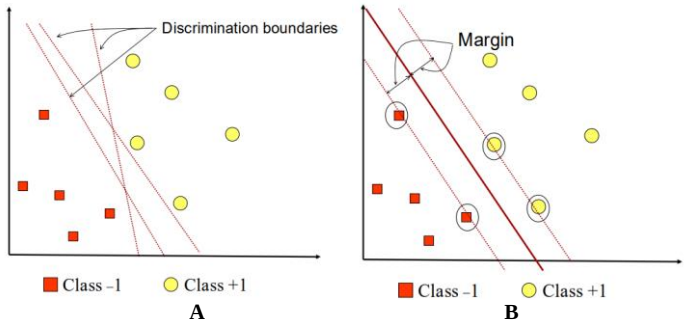
Df_A = Jumlah dokumen yang memuat kata A didalamnya

3.9 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) merupakan sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi – fungsi linier dalam sebuah fitur yang berdimensi tinggi dan dilatih dengan menggunakan algoritma pembelajaran yang didasarkan pada teori optimasi. SVM pertama kali diperkanalkan pada tahun 1992 oleh Vapnik sebagai rangkaian dari beberapa konsep-konsep unggulan dalam bidang *pattern recognition* (Susilowati dkk, 2015). Selain itu SVM juga bertujuan memberikan nilai dari banyaknya kata yang muncul atau dapat mengklasifikasi komen positif dan negatif.

SVM merupakan salah satu metode klasifikasi dalam data *mining*. *SVM* juga dapat melakukan prediksi baik pada klasifikasi maupun regresi (Santosa, 2007 dalam Pusphita, 2014). Pada dasarnya *SVM* memiliki prinsip linear, akan tetapi kini *SVM* telah berkembang sehingga dapat bekerja pada masalah *non*-linear. Cara kerja *SVM* pada masalah *non*-linear adalah dengan memasukkan konsep kernel pada ruang berdimensi tinggi. Pada ruang yang berdimensi ini, nantinya akan dicari pemisah atau yang sering disebut *hyperplane*. *Hyperplane* dapat memaksimalkan jarak atau *margin* antara kelas data. *Hyperplane* terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin* dan kemudian mencari titik maksimalnya. Usaha dalam mencari *hyperplane* yang terbaik sebagai pemisah kelas-kelas adalah inti dari proses pada metode *SVM* (Cristianai, 2000 dalam Assaffat, 2015).

3.8.1. SVM Data Terpisah Secara *Linear*



Gambar 3.6. Visualisasi SVM berusaha menemukan *hyperplane* terbaik

(Sumber: Nugroho, Witarto, & Handoko, 2003)

Menurut Nugroho, Witarto, & Handoko (2003) dari IlmuKomputer.Com Konsep SVM merupakan usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Pada gambar A memperlihatkan beberapa *pattern* yang merupakan anggota dari dua buah *class* +1 dan –1. *Pattern* yang tergabung pada *class* –1 disimbolkan dengan warna merah (kotak), sedangkan pattern pada *class* +1 disimbolkan dengan warna kuning (lingkaran). *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur margin *hyperplane* tersebut dan mencari titik maksimalnya. Margin adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing *class*. *Pattern* yang paling dekat disebut sebagai *support vector*. Garis *solid* pada gambar B menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua *class*, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM.

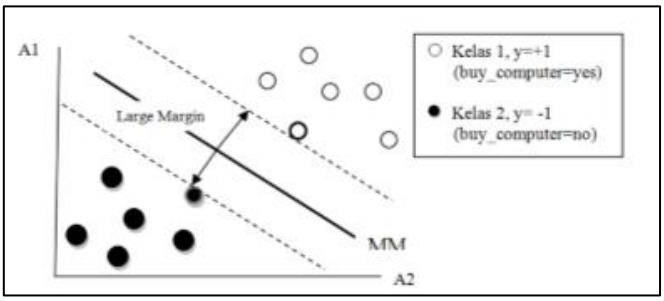
Langkah awal suatu algoritma SVM adalah pendefinisian persamaan suatu hyperplane pemisah yang dituliskan dengan:

$$W.X + b = 0 \tag{3.4}$$

W adalah suatu bobot vektor, yaitu $W = \{W1, W2, \ldots, Wn\}$; n adalah jumlah atribut dan b merupakan suatu skalar yang disebut dengan bias. Jika berdasarkan pada atribut A1, A2 dengan permisalan tupel pelatihan $X = (x_1, x_2)$, x_1 dan x_2 merupakan nilai dari atribut A1 dan A2, dan jika b dianggap sebagai suatu bobot tambahan w_0, maka persamaan suatu *hyperplane* pemisah dapat ditulis ulang sebagai berikut:

$$w_0 + w_1 x_1 + w_2 x_2 = 0 \tag{3.5}$$

Setelah persamaan dapat didefinisikan, nilai x_1 dan x_2 dapat dimasukkan ke dalam persamaan untuk mencari bobot w_1, w_2, dan w_0 atau b.



Gambar 3.7. Pemisahan Dua Kelas Data Dengan Margin Maksimum

(Kurniawan & Supriyanto, 2013)

Pada **Gambar 3.7**, SVM menemukan *hyperplane* pemisah maksimum, yaitu *hyperplane* yang mempunyai jarak maksimum antara tupel pelatihan terdekat. *Support vector* ditunjukkan dengan batasan tebal pada titik tupel. Dengan demikian, setiap titik yang terletak di atas hyperplane pemisah memenuhi rumus:

$$w_0 + w_1 x_1 + w_2 x_2 > 0 \qquad (3.6)$$

Sedangkan, titik yang terletak di bawah hyperplane pemisah memenuhi rumus:

$$w_0 + w_1 x_1 + w_2 x_2 < 0 \qquad (3.7)$$

Melihat dua kondisi di atas, maka didapatkan dua persamaan hyperplane yaitu:

$$H_1 : w_0 + w_1 x_1 + w_2 x_2 \geq 1 \; untuk \; y_1 = +1 \qquad (3.8)$$

$$H_2 : w_0 + w_1 x_1 + w_2 x_2 \leq -1 \; untuk \; y_1 = -1 \qquad (3.9)$$

Perumusan model SVM menggunakan trik matematika yaitu formula *Lagrangian.* Berdasarkan *Lagrangian formulation*, Maksimul Margin *Hyperplane* (MMH) dapat ditulis ulang sebagai suatu batas keputusan (*decision boundary*) yaitu:

$$d(X^T) = \sum_{i=1}^{l} \alpha_i X_i X^T + b_0 \qquad (3.10)$$

y_1 adalah label kelas dari *support vector* X_i. X^T merupakan suatu tupel *test*. α_i dan b_0 adalah parameter numerik yang ditentukan secara otomatis oleh optimalisasi algoritma SVM dan l adalah jumlah *vector support.*

Dong ddk, 2007 mengatakan dalam Kurniawan & Supriyanto, 2013 bahwa model pembelajaran SVM memperkenalkan istilah penalti untuk klasifikasi kesalahan dalam

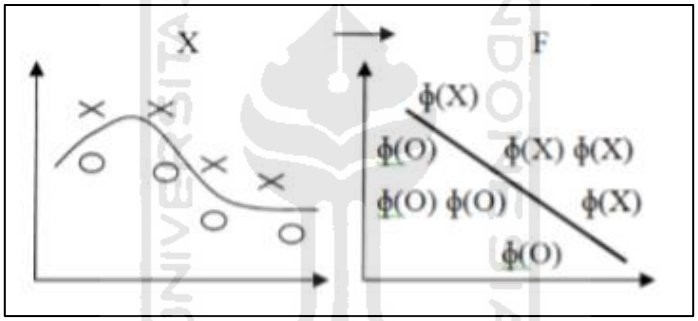
fungsi objektif dengan menggunakan parameter biaya [14]. Dengan adanya parameter biaya terhadap kesalahan, maka fungsi optimasi SVM menjadi:

$$\min \frac{1}{2}|W|^2 + C\sum_{i=1}^{m}\xi_i X_i X^T + b_0 \qquad (3.11)$$

Li-mei, 2009 mengatakan dalam Kurniawan & Supriyanto 2013, $\xi_i \geq 0,1 \leq m \leq l$ merupakan variabel slack untuk memungkinkan kesalahan beberapa klasifikasi dan *C* yang disebut sebagai parameter biaya untuk mengontrol keseimbangan antara margin dan kesalahan klasifikasi. Dengan demikian pembatas pada dua kelas diberi suatu tambahan berupa variable slack ξ_i sehingga margin pembatas menjadi:

$$x_i w + b \geq +1 - \xi_i \; untuk \; y_1 = +1 \qquad (3.12)$$

$$x_i w + b \leq -1 + \xi_i \; untuk \; y_1 = -1 \qquad (3.13)$$



Gambar 3.8. Kernel Mengubah Problem yang Tidak Liner Menjadi Linier

Pada **Gambar 3.8** memperlihatkan adanya permasalahan klasifikasi tidak dapat diselesaikan secara linier pada sampel data X. Pengubahan dari problem data non linier ke linier membutuhkan hitungan yang komplek. Maka diperlukan trik matematika lain yang dapat mempermudah perhitungan, dalam hal ini suatu penggunaan kernel mulai diterapkan. Terdapat 3 persamaan pada kernel SVM yang dapat digunakan yaitu polynomial, *radial basis function* (RBF) dan sigmoid.

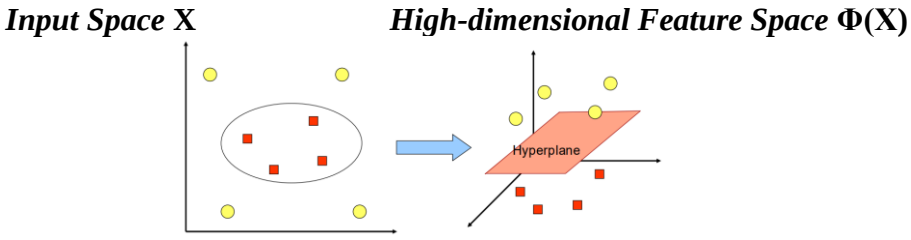
Klasifikasi dengan metode SVM melibatkan data *training* dan data *testing* dimana masing-masing terdiri dari beberapa masukan data. Masing-masing masukan dalam data *training* berisi satu nilai target dan beberapa atribut.

Tujuan lainnya dari SVM adalah memproduksi sebuah model yang dapat memprediksi nilai target dari data pengujian yang hanya diberikan nilai atributnya (Endah & KN, 2012).

3.8.2. Kernel Trick dan Non-Linear Classification pada SVM

Pada umumnya masalah dalam domain dunia nyata (*real world problem*) jarang yang bersifat *linear separable* dan kebanyakan bersifat non linear. Untuk menyelesaikan masalah non linear, SVM dimodifikasi dengan memasukkan fungsi *Kernel*. Dalam non linear SVM, pertama-tama data x dipetakan oleh fungsi Φ(x) ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua kelas tersebut dapat dikonstruksikan. Hal ini sejalan dengan teori *Cover* yang menyatakan *"*Jika suatu transformasi bersifat non linear dan dimensi dari *feature space* cukup tinggi, maka data pada *input space* dapat dipetakan ke *feature space* yang baru, dimana *pattern-pattern* tersebut pada probabilitas tinggi dapat dipisahkan secara linear" (Nugroho, Witarto, & Handoko, 2003).

Ilustrasi dari konsep tersebut dapat dilihat pada **Gambar 3.9** diperlihatkan data pada kelas kuning dan data pada kelas merah yang berada pada *input space* berdimensi dua tidak dapat dipisahkan secara linear. Selanjutnya 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*. Notasi matematika dari *mapping* ini adalah sebagai berikut.



Gambar 3.9. Visualisasi Fungsi Φ
(Sumber: Nugroho, Witarto, & Handoko, 2003)

Beberapa *kernel* yang umum dipakai pada SVM adalah:

1. Kernel Linear

   Kernel *trick* linear cocok digunakan untuk menyelesaikan masalah klasifikasi, dimana *dataset* pelatihan dalam bentuk linear. Kernel *trick* ini dinyatakan dalam persamaan 3.14 (A.Muis & Affandes, 2015).

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i})^p x \quad (3.14)$$

2. Kernel *Polynomial*

   Kernel *trick polynomial* cocok digunakan untuk menyelesaikan masalah klasifikasi, dimana *dataset* pelatihan sudah normal. Kernel *trick* ini dinyatakan dalam persamaan 3.15.

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i}, \overrightarrow{x_j} + 1)^p \quad (3.15)$$

3. Kernel *Radial Basis Function* (RBF) atau *Gaussian*

   Kernel *trick radial basis function* atau *gaussian* merupakan *kernel* yang paling banyak digunakan untuk menyelesaikan masalah klasifikasi untuk *dataset* yang tidak terpisah secara linear, dikarenakan akurasi pelatihan dan akurasi prediksi yang sangat baik pada kernel ini, dimana kernel *radial basis function* dinyatakan dalam persamaan 3.16.

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp\left(-\|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2\right) Y \quad (3.16)$$

4. Kernel *Sigmoid*

   Kernel *sigmoid* merupakan kernel *trick* SVM yang merupakan pengembangan dari jaringan saraf tiruan, dimana kernel ini dinyatakan dengan persamaan 3.17.

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \tanh(\overrightarrow{ax_i}. \overrightarrow{x_j} + \beta) \quad (3.17)$$

Kernel *trick* memberikan beberapa kemudahan, karena dalam proses pembelajaran SVM, untuk menentukan *support vector*, pengguna hanya cukup mengetahui fungsi kernel *trick* yang dipakai, tanpa perlu mengetahui wujud dari fungsi *non*-linier. Dari keseluruhan kernel *trick* tersebut, kernel *trick* radial basis function merupakan kernel *trick* yang memberikan hasil terbaik pada proses klasifikasi khususnya untuk data yang tidak bisa dipisahkan secara linear. Selanjutnya hasil klasifikasi dari data *x* diperoleh dari persamaan berikut :

$$f(x) = \sum_{i=1,\overrightarrow{x_i} \in SV}^{n} \alpha_i y_i K(\overrightarrow{x_i}, \overrightarrow{x_j}) + \mathrm{b} \qquad (3.18)$$

3.10 *Pattern Matching*

Pencocokan Pola atau *Pattern Matching* adalah suatu metode yang digunakan untuk mencocokkan suatu pola tertentu (kumpulan huruf) dengan suatu kumpulan kata (teks) atau *string*. Pada bidang sains komputer metode *pattern matching* sangat banyak digunakan antara lain *Editor Teks*, Mesin Pencari *Web*, Analisis Gambar dan lain-lain. *String* dapat kita asumsikan sebagai kumpulan dari beberapa karakter yang membentuk suatu kesatuan. (Budiasa, 2009).

3.11 *Graphmaster Pattern Matching*

Graphmaster adalah suatu metode untuk menyimpan kategori *stimulus response*. Untuk mencapai efisiensi dalam pencocokan pola dan penggunaan memori dapat menggunakan metode *Graphmaster*, dimana semua *tag* kategori akan disimpan dalam bentuk pohon bermula dari *node root* "*" sampai ke *path* tertentu dari suatu *pattern* (alicebot.org, 2008).

Graphmaster berbentuk sebuah pohon, saat klien dari *bot* (agen) memasukkan teks sebagai stimulus maka *Graphmaster* akan mencari kategori untuk mencocokkkannya ke dalam fungsi <*pattern*> sesuai dengan konteks kalimat, kemudian menghasilkan keluaran sebagai responnya.

Graphmaster matching (pencocokan graphmaster) adalah pencocokan yang bersifat *backtrack* yaitu menggunakan strategi pencarian mendalam *depthfirst search*. *Depth-first search* merupakan salah satu pencarian buta (*blind search*). Pencarian ini dilakukan dari *node* awal secara mendalam hingga yang paling akhir atau sampai ditemukan. Dengan kata lain, simpul cabang atau anak yang terlebih dahulu dikujungi. Kelebihan *Depth-first search* yaitu (Desiani dan Arhami, 2006):

1. Cepat mencapai kedalaman ruang pencarian.
2. Lebih efisien untuk ruang pencarian dengan banyak cabang karena tak perlu mengevaluasi semua simpul pada pada suatu level tertentu pada daftar open.

3. Memerlukan memori yang lebih kecil karena hanya node-node pada lintasan yang aktif saja yang akan disimpan.

Kelemahan *Depth-first search* yaitu (Desiani dan Arhami, 2006):

1. Memungkinkan tidak ditemukannya tujuan yang diharapkan.
2. Hanya akan mendapatkan satu solusi pada setiap pencarian.

3.2.1. Algoritma Graphmaster Pattern Matching

Algoritma *Graphmaster Pattern Matching* yaitu (www.alicebot.org):

1.2. Diberikan:

a. Sebuah masukan awal dengan kata "X" dan
b. sebuah *graph nodemapper*:

1.3. Apakah *nodemapper* memiliki kata kunci "*" ?Jika ya, cari *subgraph* di anak *node* yang berhubungan dengan "*". Coba semua kata yang tersisa dari masukan yang mengikuti "X" untuk melihat jika ada salah satu yang cocok. Jika tidak ditemukan yang cocok, coba:

1.4. Apakah *nodemapper* memiliki kata kunci "X" ? Jika ya cari *subgraph* di anak *node* yang berhubungan dengan "X", menggunakan masukan terakhir (akhiran dari masukan dengan "X" dihapus). Jika tidak ditemukan data yang cocok, coba:

1.5. Apakah *nodemapper* memiliki kata kunci "*" ? jika ya, cari *subgraph* di anak *node* yang berhubungan dengan "*". Coba semua sisa akhiran dari masukan yang mengikuti "X" untuk mencari jika ada yang cocok. Jika tidak ditemukan data yang cocok, kembali ke *graph* atas induk *node*, dan ambil "X" jadikan sebagai awal masukan.

1.6. Jika masukan adalah kosong (tidak ada lagi kata) dan *nodemapper* memiliki kata kunci jawaban, data yang cocok ditemukan. Pencarian berhenti dan kembalikan hasil data yang cocok.

Jika *nodemapper* memiliki kata kunci "*" dan menunjukkan ke sebuah *node*, algoritma digaransikan untuk menemukan satu yang cocok. Ketentuan *graphmaster pattern matching* antara lain:

1. Di setiap *node*, "*" memiliki prioritas utama, sebuah kata *atomic* yang cocok adalah prioritas kedua dan sebuah "*" yang cocok adalah prioritas terakhir.
2. *Pattern* tidak diurutkan secara alfabet atau berdasarkan sistem komplit yang lain, hanya sebagian diurutkan agar "*" sampai sebelum kata yang lain dan "*" setelah kata yang lain.
3. Pencocokan adalah kata per kata bukan kategori per kategori.
4. Algoritma pencocokan ini adalah sebuah versi dari DFS (*Depth First Search*), juga dikenal sebagai *backtracking*.

3.12 *Confusion Matrix*

Confusion Matrix (Kohavi dan Provost, 1998) berisi informasi tentang klasifikasi aktual dan yang telah diprediksi yang dilakukan oleh sistem klasifikasi. Kinerja sistem tersebut umumnya dievaluasi dengan menggunakan data dalam matriks. Tabel berikut menunjukkan *Confusion Matrix* untuk klasifikasi dua kelas.

Tabel 3.1. Tabel *Confusion Matrix*

		Predicted Class	
		Positif	*Negatif*
Actual Class	*Positif*	*True Positive*	*False Positive*
	Negatif	*False Negative*	*True Negative*

True positives adalah jumlah *record* positif yang diklasifikasikan sebagai positif, *false positives* adalah jumlah *record* positif yang diklasifikasikan sebagai negatif, *false negatives* adalah jumlah *record* negatif yang diklasifikasikan sebagai positif, *true negatives* adalah jumlah *record* negatif yang diklasifikasikan sebagai negatif. Setiap kolom dari *Confusion Matrix* merupakan contoh di kelas yang telah diprediksi, sedangkan setiap baris mewakili contoh di kelas yang sebenarnya. Setelah didapat *true positives, false positives, true negatives* dan *false negatives*, selanjutnya

hitung nilai *precision* dan akurasinya. *Precision* adalah ukuran terhadap suatu kelas yang telah diprediksi. Berikut Rumus dari *precision* dan akurasi.

$$\text{Akurasi} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \quad (3.19)$$

Keterangan :

TP : Jumlah *True Positive*

FP : Jumlah *False Positive*

TN : Jumlah *True Negative*

FN : Jumlah *False Positive*

3.13 *JavaScirpt*

JavaScript adalah bahasa pemrograman tingkat tinggi dan dinamis. JavaScript populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti Google Chrome, Internet Explorer (IE), Mozilla Firefox, Netscape dan Opera. Kode JavaScript dapat disisipkan dalam halaman web menggunakan tag SCRIPT. JavaScript merupakan salah satu teknologi inti World Wide Web selain HTML dan CSS. JavaScript membantu membuat halaman web interaktif dan merupakan bagian aplikasi web yang esensial. Awalnya hanya diimplementasi sebagai client-side dalam penjelajah web, kini engine JavaScript disisipkan ke dalam perangkat lunak lain seperti dalam server-side dalam server web dan basis data, dalam program non web seperti perangkat lunak pengolah kata dan pembaca PDF, dan sebagai runtime environment yang memungkinkan penggunaan JavaScript untuk membuat aplikasi desktop maupun mobile (Flanagan, 2011)

3.14 *Discord*

Discord adalah aplikasi baru yang menawarkan fitur sejenis dengan desain yang simpel, praktis, mudah digunakan, menarik, dan dapat diakses dari berbagai gadget. Di Discord tidak harus memiliki account untuk dapat bergabung ke main channel, tergantung dari apakah main channel tersebut apakah mengijinkan orang yang tidak memiliki account untuk bergabung atau tidak. Untuk dapat masuk ke dalam channel

cukup membuka link Instant Invite yang dapat digenerate oleh pemilik channel atau orang yang berada di channel tersebut.Kita juga bisa mengatur untuk berapa lama link tersebut berlaku, berapa kali link tersebut dapat dibuka, apakah link tersebut hanya untuk member sementara atau tidak, atau apakah link tersebut ingin dibuat lebih simpel atau tidak. (Raihan & Putri, 2018)

Menurut (Raihan & Putri, 2018) Aplikasi ini gratis untuk digunakan. Kita pun bisa membuat server sendiri untuk voice chat dengan teman, saudara atau bahkan pacar. Sebuah gambar dari situs resminya yang membandingkan aplikasi voice chat ini dengan aplikasi voice chat lainnya seperti team speak, raidcall dan skype. Bahkan aplikasi ini tersedia untuk Adroid dan Iphone secara free. Dan aplikasi ini aman karena menggunakan sistemEncryted. Jika ingin menggunakan aplikasi ini untuk bermain game, tidak berpengaruh terhadap FPS game saat memainkannya. Karena pada dasarnya pembuatan aplikasi ini digunakan untuk bermain game. Salah satu kendala yang dimiliki oleh Raid Call adalah server yang terlalu jauh menyebabkan delay pada suara, tetapi Discord memiliki server di Singapore yang dekat dengan Indonesia, dengan ping rendah, membuat Discord lancar untuk digunakan. Ada dua jenis channel di dalam Discord, yaitu Text Channels dan Voice Channels. Text Channels digunakan untuk text chat, kita bisa berpindah antar Text Channels dengan cepat. Jika ingin menggunakan halnya seperti LINE Messenger, Discord juga memiliki fitur yang serupa yang membuat berada di text terakhir baca (Semua unread messages akan berada dibawah ketika masuk ke Text Channel tersebut). Selain itu kita juga dapat mengedit atau menghapus chat yang sudah di tulis, tergantung dari settingan channel tersebut. Sedangkan Voice Channel, hanya dapat bergabung ke satu saja Voice Channel, dan tetap akan tersambung walaupun berada di Main Channel lain. Kita juga bisa melihat grafik ping untuk Voice Connection secara langsung, dan juga ping rata-rata. Di dalam Discord juga memiliki setting yang jika diaktifkan dapat memindahkan user yang AFK (user yang tidak aktif) di suatu Voice Channel ke Channel tertentu.

BAB IV
METODOLOGI PENELITIAN

4.1 Analisa Sistem

4.1.1. Deskripsi Sistem

Perancangan *ChatBot* menggunakan Metode *NLP* memakai Bahasa program *JavaScript* pada *platform* online STEAM berbasis media social discord. Sistem Chatbot ini dibuat untuk memudahkan para pemain game yang menggunakan social media discord mendapatkan informasi seputar penjualan game pada platform steam. Dalam hal ini chatbot dikhususkan untuk informasi penjualan game pada platform STEAM seperti nama, harga, genre, rating, jumlah pembeli/pemain, event, dan discount. Dengan adanya sistem *chatbot* ini maka informasi dapat diperoleh dalam 24 jam. Selain itu chatbot ini juga memberikan respon cepat, sehingga pengguna media social discord ini dapat dengan mudah memperoleh informasi dimanapun dan kapanpun, karena sosial media discord ini bisa digunakan di smarphone maupun PC

4.1.2. Tujuan Pembuatan Sistem

Beberapa permasalan yang dihadapi oleh para pemain game yang menggunakan *platform* steam yaitu informasi yang terkait penjualan game, event, update dan maintance suatu game. Seiring berkembangnya teknologi informasi dan elektronik maka menuntut *platform* steam untuk selalu hadir dan lebih cepat dalam memproses informasi dan akurat terutama di media social yang popular dikalangan para game terutama discord.

4.1.3. Kebutuhan Sistem

Berdasarkan tujuan dari sistem yang dibuat, maka berikut ini merupakan kebutuhan yang harus disediakan diantarnya yaitu :

a. Kebutuhan fungsional

    1. Sistem memiliki fitur informasi *Steam Store*

2. Sistem memiliki fitur *Steam Event*.
3. Sistem memiliki fitur *Clan Event*
4. Sistem memiliki fitur *Music Bot*
5. Sistem memiliki fitur game santai *Ball8 Game Bot*

b. Kebutuhan non-fungsional
1. Sistem chatbot ini bisa di akses pada media sosial Discord.
2. Sistem memiliki tampilan antarmuka yang mudah di pahami atau *user friendly*

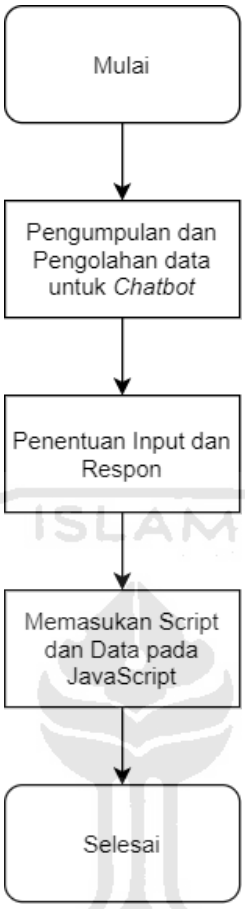
c. Kebutuhan perangkat lunak
1. Visual Studio Code
2. Sniping Tools
3. Node Java Script
4. Discord
5. Steam
6. Ffmpeg
7. *Browser*

d. Kebutuhan perangkat keras
1. CPU : Intel® Core™ i5-7300HQ CPU @ 2.50GHz
2. GPU : NVIDIA GeForce GTX 1050 4 GB
3. RAM : Team T-Force Vulcan SODIMM DDR4 16Gb
4. Hardisk : 2 TB
5. Laptop : LEGION Y520

4.2 Tahapan Perancangan Chatbot

4.2.1. Penyusunan Basis Pengetahuan Chatbot

Pada gambar 4.2 Merupakan alur dalam Menyusun basis pengetahuan basis pengetahuan untuk *Chatbot*. Basis pengetahuan tersebut untuk memberikan kecerdasan pada *Chatbot* yang dibangun untuk dapat melakukan percakapan dengan pengguna.



Gambar 4.1. Basis Pengetahuan *Chatbot*

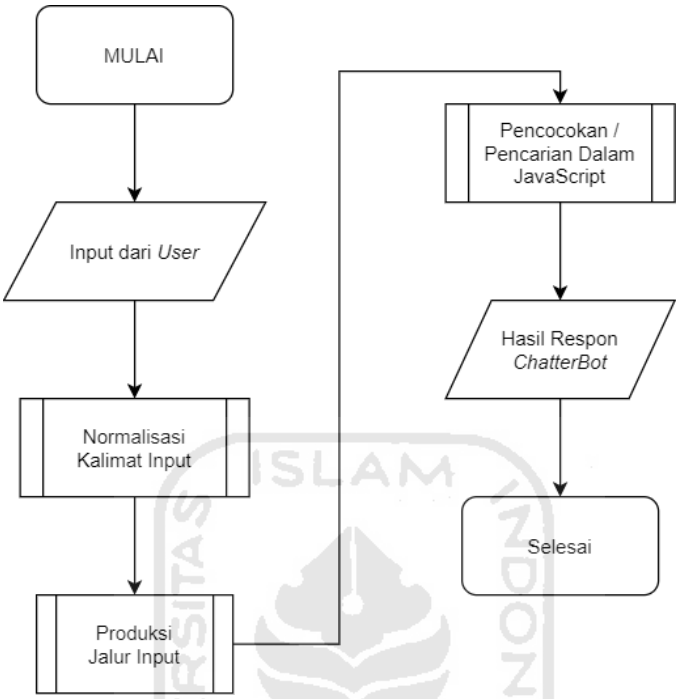
a. Pengumpulan dan pengolahan data *chatbot*

Data untuk *Chatbot* terdiri dari pengetahuan dasar dalam *JavaScript*. Penyerdahaan bentuk gramatikal kompleks menjadi bentuk yang lebih sederhana, penyerdahanaan kalimat menjadi sub kalimat, perbaikan ejaan, persamaan kata, dan tata Bahasa. Data *command* meliputi informasi seputar *Steam Store*, *Steam Event, Music Bot*, dan *Ball8 games*. Sedangkan data *Steam Store* didapat dari *platform* resmi milik STEAM.

b. Penentuan *input* dan *respon*

Dari data yang telah dikumpulkan sebelumnya maka akan dilakukan penentuan input serta respon *chatbot* dengan beberapa keterangan barang yang diambil seperti nama, genre, harga, *review user*, deskripsi game, developer dan publisher game

4.2.2. Algoritma *Chatbot*



Gambar 4.2. Alur Proses Input Output *Chatbot*

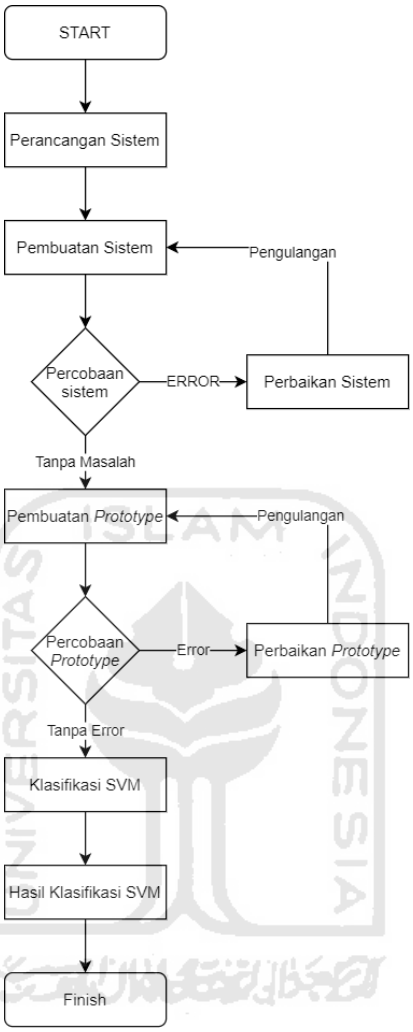
Gambar 4.2 merupakan bentuk proses *input output* yang dilakukan dalam sistem *Chatbot* digambarkan menggunakan *flowchat*

4.2.3. *Sistem Database*

Dalam perancangan sistem dibutuhkan basis data untuk menyimpan data barang dan data pengguna yang dibutuhkan. Basis data yang digunakan bertipe *json* yang mana disimpan lansung kedalam server developer discord pribadi

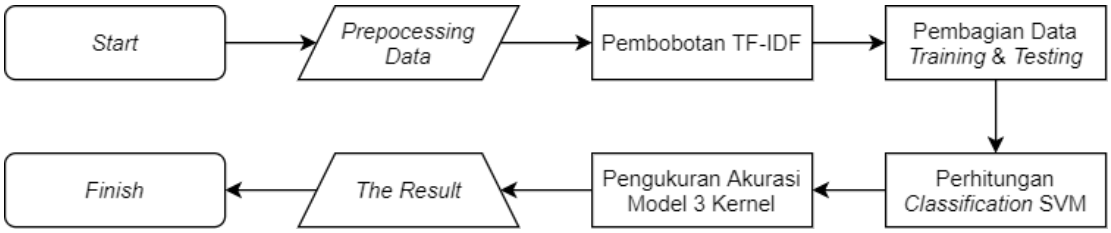
4.3 Langkah-Langkah Penelitian

Tahapan atau Langkah-langkah pada penelitian ini dapat dilihat dalam bentuk *flowchart* seperti berikut ini:



Gambar 4.3. Alur Tahapan Analisis

4.4 Metode Evaluasi SVM



Gambar 4.4. Alur Tahapan Evaluasi SVM

BAB V
HASIL DAN PEMBAHASAN

Pada bab ini berisikan tentang pembahasan hasil bagaimana perancangan prototype *chatbot*, interprestasi hasil, respon sistem dan analisis klasifikasi SVM.

5.1 Analisis Masalah

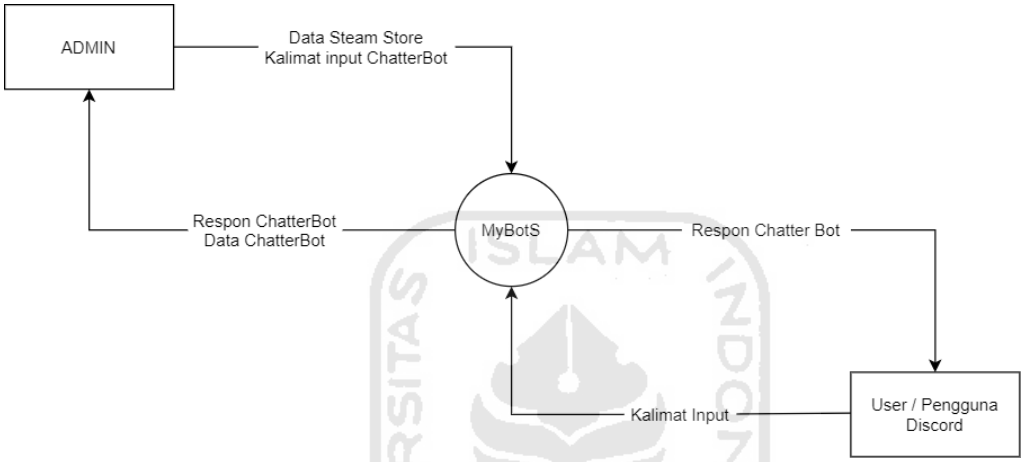
NLP merupakan salah astu cabung dari *AI* yang banyak diimplementasikan untuk menyelesaikan masalah dalam sebuah pembagian informasi atau *information Retrieval*(IR). Yaitu pekerjaan untuk menemukan dokumen yang relevan dengan kebutuhan informasi yang dibutuhkan oleh pengguna/*user*. Contoh sistem IR yang paling popular dikalangan umat manusia pengguna internet adalah search engine pada *world wide web.* Seorang user web bisa memasukan kata apapun kedalam sebuah mesin pencari atau *search engine* untuk melihat hasil dari pencarian yang relevan. Aplikasi *chatbot* juga membutuhkan IR untuk menentukan sebuah informasi yang dipertanyakan pengguna *chatbot* tersebut, *NLP* (*Natural Language Processing*) diharapkan menjadi metode untuk mengimplementasikan *chatbot* dalam menyelesaikan permasalahan informasi STEAM STORE pada Sosial media Discord.

5.2 Implementasi Sistem Chatbot

Sistem pada *chatbot* yang dirancang bangun ini menggunakan metode *graphmaster pattern matching* sebagai algoritma pencarian jawaban. Pada tahap pembuatan dibangun *chatbot* yang dapat memberikan informasi seputar STEAM STORE kepada *user* di media sosial Discord. Pada aplikasi *chatbot* yang dibangun tersebut, akan memberikan jawaban sesuai dengan representasi pengetahuan yang telah diberikan.

5.2.1. Diagram Konteks *Chatbot*

Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan *input* kedalam sistem atau *output* dari sistem yang memberikan gambaran tentang keseluruhan sistem. Diagram konteks sistem *Chatbot* digambarkan seperti pada gambar 5.1 berikut ini :



Gambar 5.1. Diagram Konteks *Chatbot*

a. Admin

Admin harus login ke sistem Discord developer terlebih dahulu agar bisa mengakses script chatterbot. Tugas admin yaitu memanage data, maintance dan perbaikan *chatterbot*.

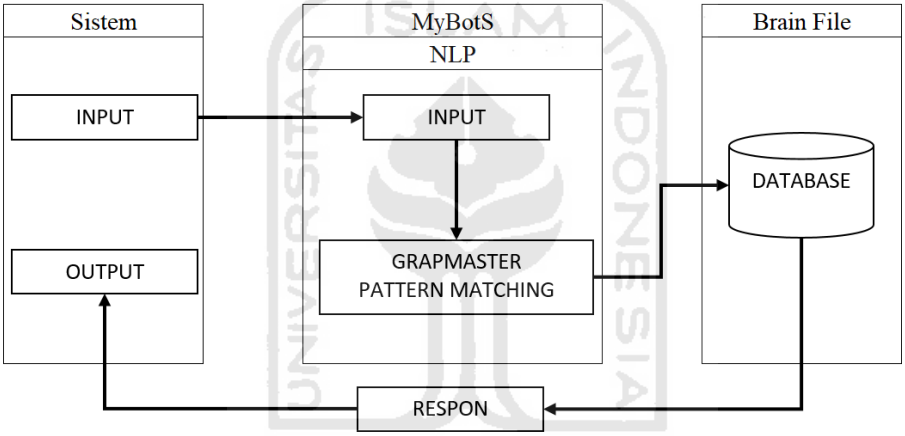
1. Admin dapat melalukan Kelola data *script*, dimana aktifitas yang dapat dilakukan diantara nya *Creat, Read, update, delete* (CRUD) serta pencarian.
2. Admin dapat melakukan melakukan uji coba *chatting bot* pada channel *test MyBotS* yang merupakan salah satu *channel* yang hanya bisa diakses oleh Admin atau Owner.

b. *User*/pengguna Discord

User atau pengguna Discord hanya dapat melakukan percakapan atau *Chatting* dengan cara memasukan kalimat atau pertanyaan mengenai seputar fitur fitur yang terdapat pada *chatterbot* MyBotS.

5.2.2. *Natural Language Processing*

Sistem NLP terdiri dari 3 bagian, yaitu input, bagian basis pengetahuan dan bagian output. Setelah chatbot di *invite* kedalam server discord *user* akan menyapa chatbot dengan kata hallo untuk memulai percakapan kemudian bot akan membalas untuk memberikan input selanjutnya yang bisa di gunakan oleh *user*. Kemudian input akan diolah dan dicari pila yang cocok dengan basis pengetahuan yang terdapat pada *script chatbot*. Jika pola sudah ditemukan dan sesuai, maka proses selanjutnya adalah bot akan mencari jawaban dari input *user*. Setelah jawaban ditemukan maka *chatbot* akan menampilkan jawaban dalam channel chatting sebagai *output*.



Gambar 5.2. Alur sistem *chatbot*

5.2.3. Basis Pengetahuan

Pada analisis sistem *chatbot* diberikan Analisa representasi pengetahuan yang akan digunakan sebagai basis pengetahuan. Peneliti mendeskripsikan representasi basis pengetahuan kedalam bentuk tabel sebagai berikut:

Tabel 5.1. Basis pengetahuan *chatbot*

No	Pattern	Template
1	switch (args[0]) { case "hallo": break;}	"hallo🙋‍♂️! My Name MyBotS 🧑‍💻 i will help you about information of STEAM STORE and another fiture.

		please check '!J help' to see anymore commands on this bot, My Owner 🧒🏫 Imam Al Maksur");}
2	(msg01.content =!JSteam Store')}	Steam Store, Top Seller Game, kumpulan penjulan game Terbaik di Platform Steam.
3	('message', msg =>{(msg.content !J Steam Game “nama game”)	Nama Game, deskripsi game, harga game terkini, Publisher, Developer, Genre, Release Date, dan *user review*
4	("message", async msg => { (msg.content === !J Event)	Informasi terkait event yang akan diadakan selanjutnya
5	("message", async msg => { (msg.content === !J Date)	Untuk mengetahui tanggal dan waktu pelaksaan event yang akan datang.
6	case 'play': function play(connection, msg){ (server.queue[0]) {play(connection, msg);}| {connection.disconnect();}	you need to provide a link | you must be in channel to play the bot! MyBotS akan memainkan musik req di Voice Channels
7	case 'skip': var server = servers[msg.guild.id]; server.dispatcher.end(); break;	skipping the song!
8	case 'stop': var server = servers[msg.guild.id]; (msg.guild.voiceConnection){ server.dispatcher.end();	Ending the queue leaving the voice channel!

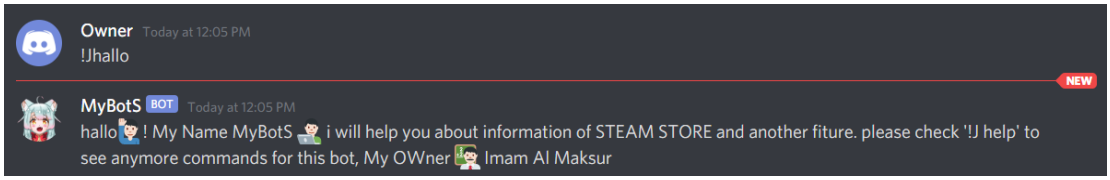
	msg.guild.voice Connection.disconnect(); break;	

9	("message", msg => {const msgToLower = msg.content.toLowerCase();
	const gameArray = ["rock", "paper", "scissors"];
	const multiply = 3;
	const inputArray = ["!rock", "!paper", "!scissors"];
		"It's a draw! 🥇", "The computer wins! 🥇", "You win! 🏆";
		"You've chosen rock!", "You've chosen paper!", "You've chosen scissors!";
		Pc has chosen:

5.3 Respon Sistem Chatbot

Tahap pengujian *sistem* ini melakukan Analisa berdasarkan hasil pengumpulan data sistem yang telah diperoleh. Pengujian dilakukan uncuk mencari seberapa akurat jawaban yang mampu diberikan oleh sistem pada *chatbot*. Adapun pengujian dilakukan dengan cara memberikan pola *commad* atau pertanyaan yang sesuai dengan basic chatbot, berikut hasil respresentasi sistem MyBotS

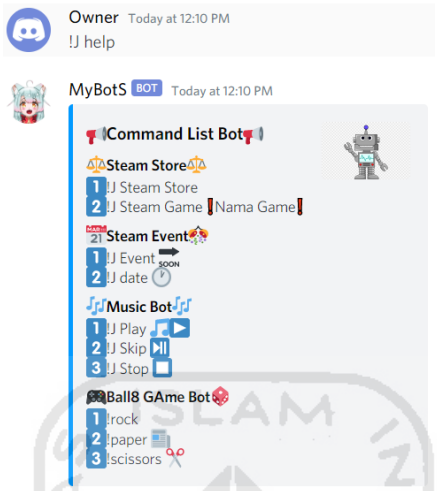
1. Salam pembuka



Gambar 5.3. Kalimat pembuka Chatting

Pada gambar 5.5 adalah hasil output dari pembukaan percakapan antar *user* dan MyBots. Dapat dilihat setelah *user* mengatakan !Jhallo, chatbot juga membalasnya kemudian sedikit mendeskripsikan dirinya dan kemudian memberi tahu commands selanjutnya yang harus diinput *user*.

2. List command MyBotS

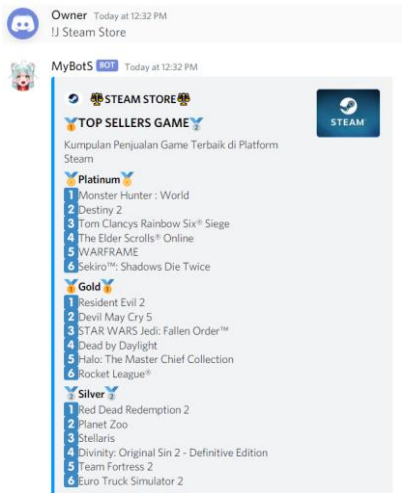


Gambar 5.4. List command MyBotS

Gambar 5.6 menunjukan List perintah MyBotS atau Command List Bot, dimana bisa dilihat terdapat 4 kategori command yang bisa digunaka oleh *user*, yang pertama kategori Steam Store, yang kedua kategori Steam Event, ke tiga kategori music bot dan yang keempat kategori Game Bot. user bisa meng-input command sesuai list dan kategori yang tersedia.

3. Kategori Steam Store

   a. Steam Store



Gambar 5.5. Kategori Steam Store

Pada gambar 5.7 menampilkan output Steam Store atau penjualan game pada steam yang mana bisa dilihat terdapat list-list game yang dijual pada *platform STEAM*.

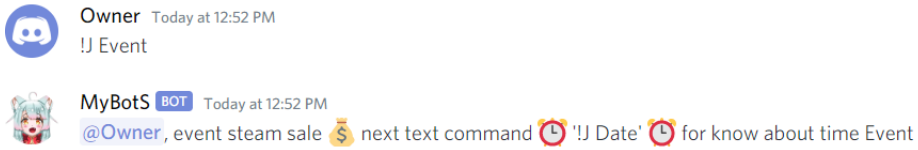
b. Steam Game (nama game)

Gambar 5.6. Steam Game Monster Hunter

Pada gambar 5.8 bisa dilihat ouput dari command "Steam Game Monster Hunter" yang mana adalah salah satu game yang terdapat pada list Steam Store, dapat dilihat berbagai informasi diantaranya deskripsi game, harga game, publisher game, developer game, Genre game, release date game, dan user views dari game tersebut.

4. Kategori steam event

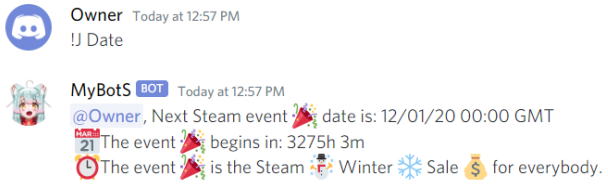
a. Steam Event



Gambar 5.7. Steam Event

Pada gambar 5.9 diperlihatkan output Event yang mana MyBotS memberitahukan bahwa akan ada steam sale, *user* diberitahukan untuk mengetik *command* "!J Date" untuk mengetahui waktu akan dilaksanakannya *Event*.

b. Event Date

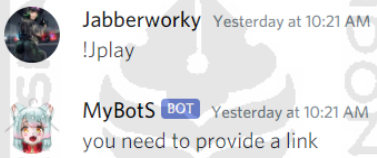


Gambar 5.8. Event Date

Pada gambar 5.10 menampilkan output tanggal dan waktu akan dilaksanakannya nya *event steam winter sale*, disitu juga ditampilkan waktu yang berjalan mundur menuju event yang ditampilkan dalam bentuk jam dan menit.

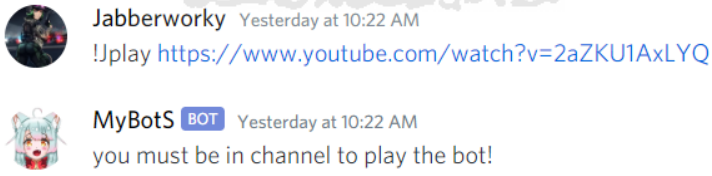
5. Kategori Music Bot

a. Play Music Bot

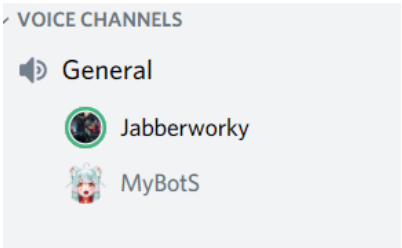


Gambar 5.9. *Command* memainkan musik

Pada gambar 5.11 *user* meminta MyBotS untuk memainkan music dengan memberikan *command* "!Jplay" kemudian MyBotS memberikan respon *you need to provide a link* yang arti nya jika *user* ingin memainkan music harus menyertakan link pada *command*.



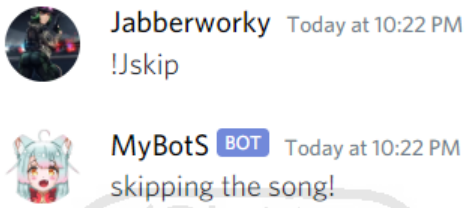
Gambar 5.10. *Command* memainkan music lanjutan 1



Gambar 5.11. bot musik memainkan musik di *voice channels*

Gambar 5.12 menunjukkan respon dari MyBotS dimana meminta User bergabung di Voice Channel untuk memainkan music bot. Gambar 5.13 adalah tampilan MyBotS dan user berada di dalam satu Voice Channel dimana MyBotS memainkan music sesuai *request* dari *user*.

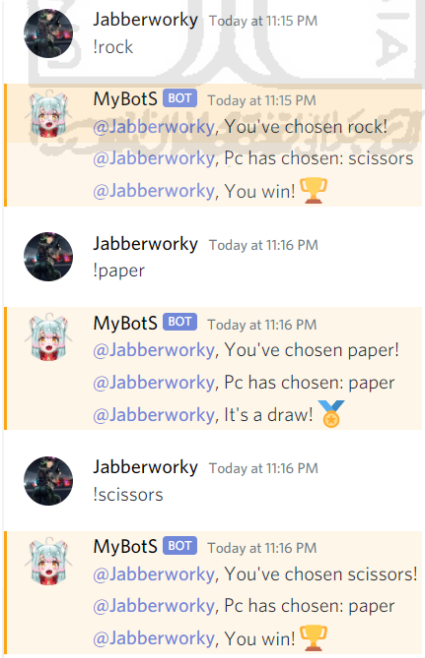
b. *Command skip music* bot



Gambar 5.12. *Command skip music*

Gambar 5.14 adalah *command skip music* jika *user* ingin memainkan music lain. Tetapi, sebelumnya user harus terlebih dahulu menggunakan *command play* untuk menginput musik kedalam list *request* musik bot. setelah itu *user* bisa menggunakan *command skip* untuk memainkan musik lain yang berada didalam list *request music*.

6. Kategori 8Ball Game Bot



Gambar 5.13. interaksi game antar user dan bot

Gambar 5.15 adalah hasil interaksi antara user dan MyBotS. Interaksi yang dilakukan antara user dan MyBotS adalah interaksi permainan gunting, batu dan kertas atau dikenal dengan 8ball games. Pengetahuan 8ball games menggunakan basic random yang mana setiap respon yang diberikan bot adalah random.

Sehingga dari Analisa sistem yang telah dibuat dengan menggunakan algoritma *graphmaster paftern matching* yang dikembangkan dapat berkomunikasi dengan baik dan mampu memberikan informasi seputar steam store maupun fitur-fitur lainnya

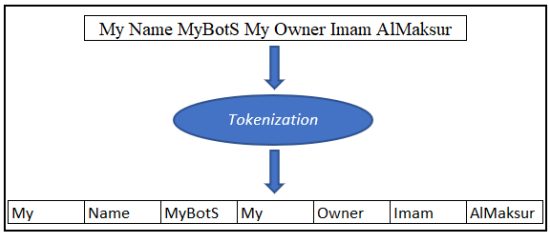
5.4 Topik Modeling

5.4.1. *Case Folding*

Proses *case folding* adalah tahapan yang mengubah semua huruf alphabet kapital dalam teks menjadi huruf kecil. Hanya huruf alphabet 'a' sampai dengan 'z' yang diterima. Selain itu *case folding* juga akan menghilangkan beberapa bagian yang kemungkinan terdapat di dalam teks seperti nomor, dan beberapa kata-kata yang tidak dibutuhkan dalam penelitian ini (*Remove Punctuation*).

5.4.2. *Tokenization*

Pada tahap *tokenizing*, antara kata yang satu dengan kata yang lain dalam masing-masing kalimat akan dipisahkan untuk dapat diidentifikasi. Umumnya setiap kata dalam suatu kalimat terpisahkan oleh karakter spasi, sehingga proses tokenisasi mengandalkan karakter spasi pada kalimat untuk melakukan pemisahan kata



Gambar 5.14. *Tokenization process*

Pada gambar 5.3 menjelaskan proses *tokenization* dimana kalimat "My Name MyBotS My Owner Imam Almaksur yang telah dilakukan proses pemisahan menjadi beberapa kata.

5.4.3. *Filtering*

Proses filtering merupakan proses pengambilan kata-kata yang dianggap penting atau memiliki arti dari hasil proses tokenisasi. Sedangkan kata-kata yang tidak dibutuhkan akan dihilangkan dari kalimat tersebut. Kata-kata yang akan dihilangkan adalah kata penghubung dan preposisi. Contoh kata yang dihilangkan adalah, which, what, that, for, dan, dll.

5.4.4. TF-IDF

After going through the case folding, tokenization, and filtering stages, the data will be further processed into the analyzing stage using TF-IDF weighting.

Tabel 5.2. TF-IDF Result

Class	age	aoe	Clancys	Clasic	Club	Warframe	Truck	Twice	…	Wars	World
SteamStore	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
SteamStore	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
SteamStore	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
SteamStore	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	5.430817
Event	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Event	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Event	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Event	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Music	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Music	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Music	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Music	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Game Bot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Game Bot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
Game Bot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0
……..	…..	…..	…..	…..	…..	…..	…..	…..	…	…..	…..
Game Bot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0

5.5 Klasifikasi SVM (Support Vector Machine)

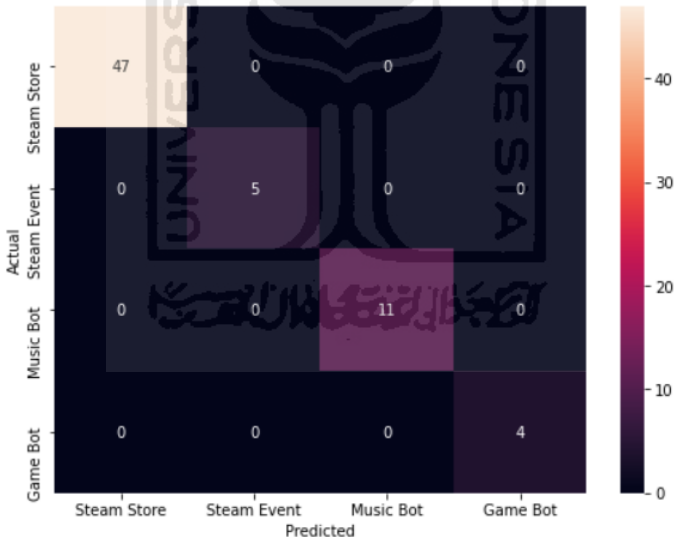
Sebelum dilakukannya pengujian klasifikasi, pertama-tama lebih dahulu data dibagi menjadi data latih / *training* dan data uji / *testing.* Data latih digunakan pada saat proses klasifikasi guna mempelajari pola data sedangkan data uji digunakan untuk melakukan pengujian klasifikasi terhadap sistem yang telah dibuat pada penelitian ini.

Pembagian data dilakukan secara acak yang akan di proses oleh sistem. Data *training* yang digunakan sebanyak 80% dari total data yaitu sebanyak 67 data dan data *testing* untuk prediksi sebanyak 20% dari total data yaitu sebanyak 17 data. Klasifikasi menggunakan Algoritma *Support Vector Machine kernel*. *Kernel* yang digunakan adlah *Linear*, *Poly* dan *RBF*. Sebelum dilakukan klasifikasi terlebih dahulu data dilakukan proses *preprocessing* dan pembobotan kata TF-IDF.

5.2.1. Hasil Klasifikasi *Support Vector Machine* (SVM)

Proses klasifikasi dilakukan pada data *training* bertujuan untuk membuat model, dan kemudian model tersebut digunakan untuk *testing*. Proses klasifikasi *Support Vector Machine* menggunakan 3 kenel diantaranya *Linear, Poly,* dan RBF.

5.5.1.1 *SVM Kernel Linear*



Gambar 5.15. *Confussion matrix SVM Kernel Linear* data *trainning*

Pengukuran kinerja klasifikasi dapat dilakukan menggunakan tabulasi silang (*Confussion matrix*). Cara membaca tabulasi silang yaitu akurasi terprediksi benar jika berada di poisi diagonal, sedangkan dinilai *error* jika tepat tidak berada di posisi diagonal. Berdasarkan gambar 5.15 dapat dihitung jumlah nilai akurasi dan errornya.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

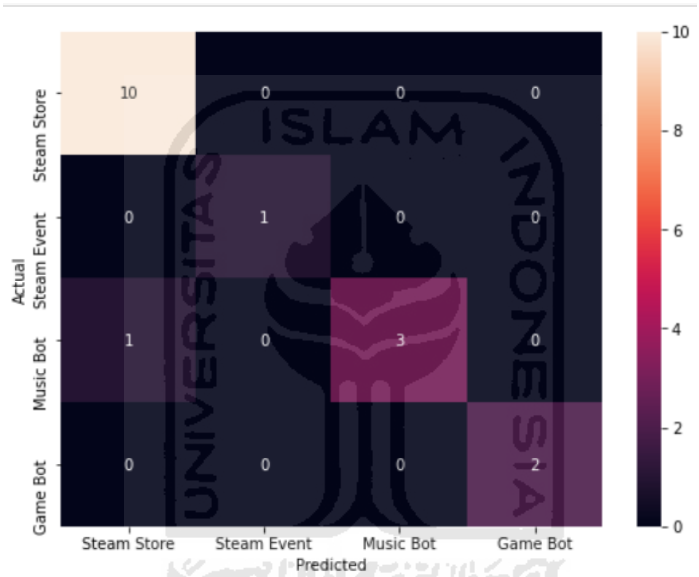
$$\text{Akurasi} = \frac{47 + 5 + 11 + 4}{67} \times 100\%$$

$$\text{Akurasi} = 100\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{0}{67} \times 100\%$$

$$Error = 0\%$$



Gambar 5.16. *Confussion matrix SVM Kernel Linear* data *testing*

Gambar 5.16 merupakan *Confussion matrix SVM Kernel Linear* data *testing*, dapat dilihat hasil prediksi berbeda dengan hasil *training*. Jika pada data *training* semua nilai berada tepat di dalam diagonal tapi pada hasil data *testing* ini terdapat nilai yang tidak tepat berada didalam diagonal. Didapatlah nilai akurasi sebagai berikut.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

$$\text{Akurasi} = \frac{10 + 1 + 3 + 2}{17} \times 100\%$$

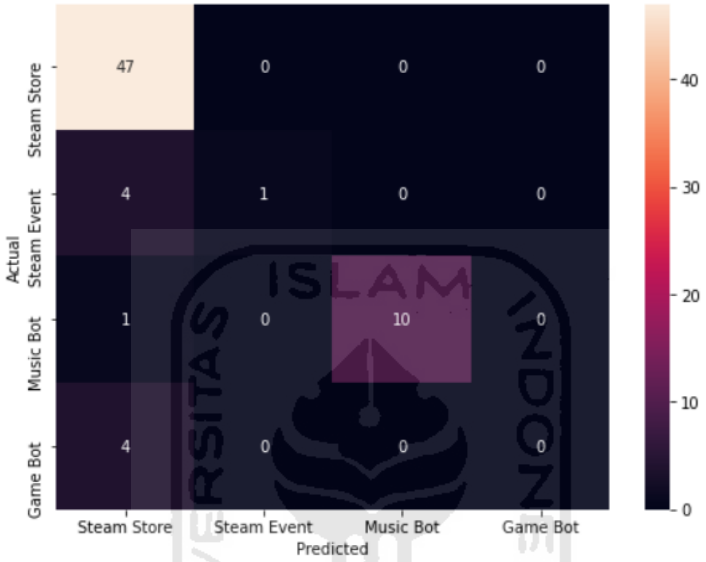
$$\text{Akurasi} = 94\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{1}{17} \times 100\%$$

$$Error = 6\%$$

5.5.1.2 *SVM Kernel Polynomial*



Gambar 5.17. *Confussion matrix SVM Kernel Polynomial* data *trainning*

Pada Gambar 5.17 dapat di lihat *Confussion matrix SVM Kernel Polynomial* data *training* menunjukan bahwa ada beberapa *misclassification* yang mana terdapat nilai yang tidak tepat berada didalam didiagonal. Berikut adalah nilai akurasi dan error-nya.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

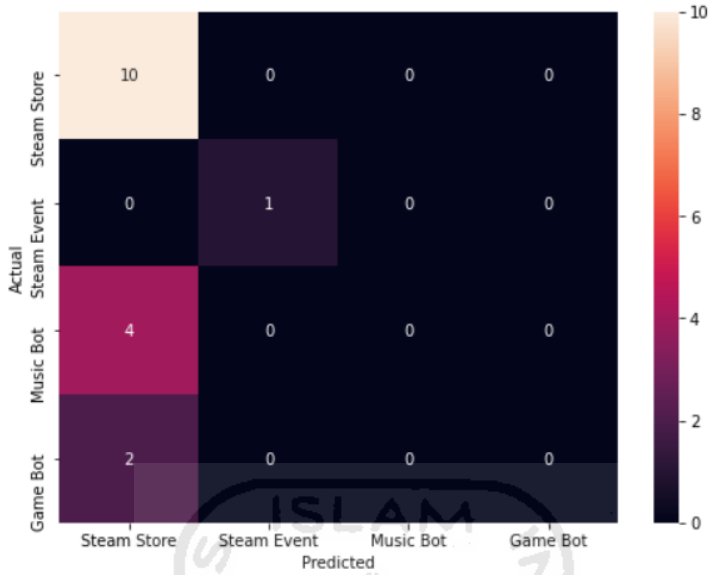
$$\text{Akurasi} = \frac{47 + 1 + 10 + 0}{67} \times 100\%$$

$$\text{Akurasi} = 86\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{4 + 1 + 4}{67} \times 100\%$$

$$Error = 14\%$$



Gambar 5.18. *Confussion matrix SVM Kernel Polynomial* data *testing*

Dari gambar 5.18 didapati hal menarik yaitu dari kategori music bot dan kategori game bot hanya terdapat nol data yang benar diklasifikasi oleh mesin. Perhitungan prediksinya sebagai berikut.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

$$\text{Akurasi} = \frac{10 + 1 + 0 + 0}{17} \times 100\%$$

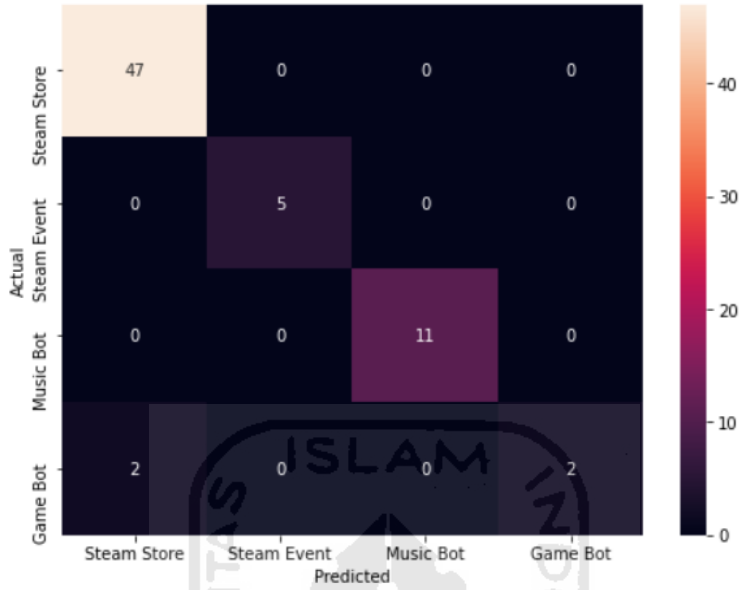
$$\text{Akurasi} = 64.7\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{4 + 2}{17} \times 100\%$$

$$Error = 35.3\%$$

5.5.1.3 *SVM Kernel RBF*



Gambar 5.19. *Confussion matrix SVM Kernel RBF* data *trainning*

Dari Gambar 5.19 dapat di lihat *Confussion matrix SVM Kernel RBF* data *training* menunjukan bahwa ada beberapa *misclassification* yang mana terdapat nilai yang tidak tepat berada didalam didiagonal. Berikut adalah nilai akurasi dan error-nya.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

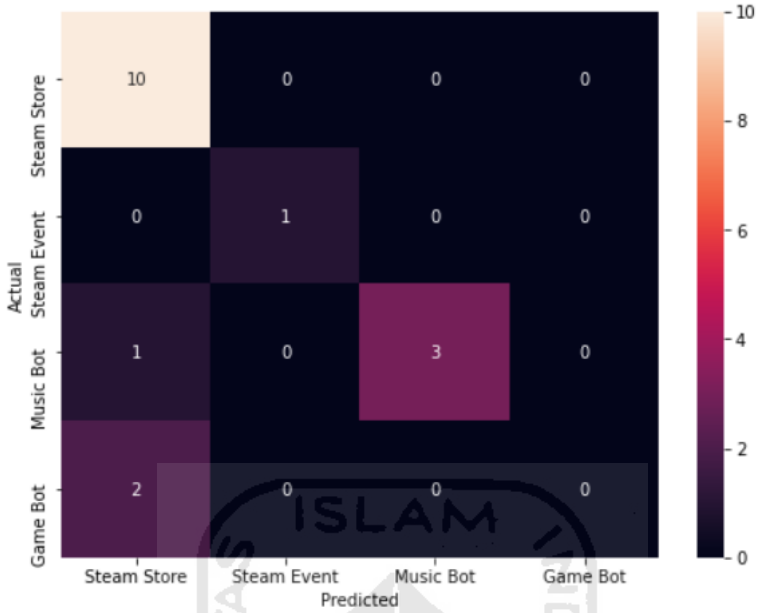
$$\text{Akurasi} = \frac{47 + 5 + 11 + 2}{67} \times 100\%$$

$$\text{Akurasi} = 97\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{2}{67} \times 100\%$$

$$Error = 3\%$$



Gambar 5.20. *Confussion matrix SVM Kernel RBF* data *testing*

Gambar 5.20 menunjukan hal menarik yang mana kategori kategori game bot hanya terdapat nol data yang benar diklasifikasi oleh mesin. Perhitungan prediksinya sebagai berikut.

$$\text{Akurasi} = \frac{\sum(\text{prediksi benar})}{\sum(\text{seluruh prediksi})}$$

$$\text{Akurasi} = \frac{10 + 1 + 3 + 0}{17} \times 100\%$$

$$\text{Akurasi} = 82.3\%$$

$$Error = \frac{\sum(\text{prediksi salah})}{\sum(\text{seluruh prediksi})}$$

$$Error = \frac{1 + 2}{17} \times 100\%$$

$$Error = 17.7\%$$

Hasil dari perbandingan klasifikasi SVM dengan tiga kernel akan dirangkum pada tabel 5.2 berikut ini

Tabel 5.3. Rangkuman hasil klasifikasi SVM dengan ketiga kernel

Kernel	Akurasi	Error	Keterangan
Linear	100%	0%	Trainning
Linear	94%	6%	Testing
Polynomial	86%	14%	Trainning
Polynomial	64.7%	35.3%	Testing
RBF	97%	3%	Trainning
RBF	82.3%	17.7%	Testing

Dari Tabel 5.3 dapat dilihat hasil perbandingan klasifikasi SVM yang menggunakan tiga kernel yang berbeda, didapati kernel *Linear* memiliki akurasi lebih tinggi dari kernel lainnya yaitu 100% untuk data training dan 94 untuk data testing. Sehingga dapat dipastikan bahwa kernel *Linear* akurasinya lebih baik dibandingkan kernel *Polynomial dan RBF* untuk penelitian ini.

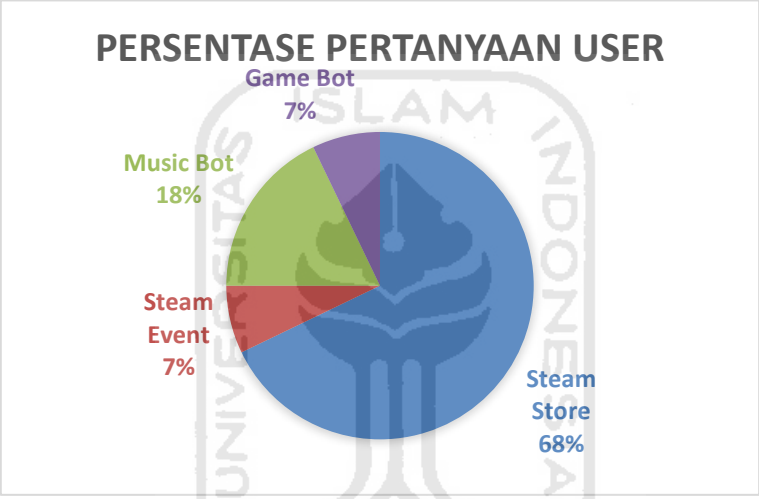
5.5.2 Prediksi Pertanyaan Baru

```
"1) !J Game Steam Fall Guys?"
  - Prediksi: 'Steam Store'

"2) !J Game terbaru Steam tahun 2020?"
  - Prediksi: 'Steam Store'

"3) !J Steam Store 2021?"
  - Prediksi: 'Steam Store'

"4) !J Game Steam GTA 6 ?"
  - Prediksi: 'Steam Store'

"5) !J Game Steam Action ?"
  - Prediksi: 'Steam Store'

"6) !J Play Music Syar'i?"
  - Prediksi: 'Music Bot'

"7) !J Event Of Years?"
  - Prediksi: 'Steam Event'

"8) !J Date Years Event?"
  - Prediksi: 'Steam Event'

"9) !rock pilihanku ?"
  - Prediksi: 'Game Bot'

"10) !scissors kesukaanku ?"
  - Prediksi: 'Game Bot'
```

Gambar 5.21. hasil prediksi data pertanyaan baru

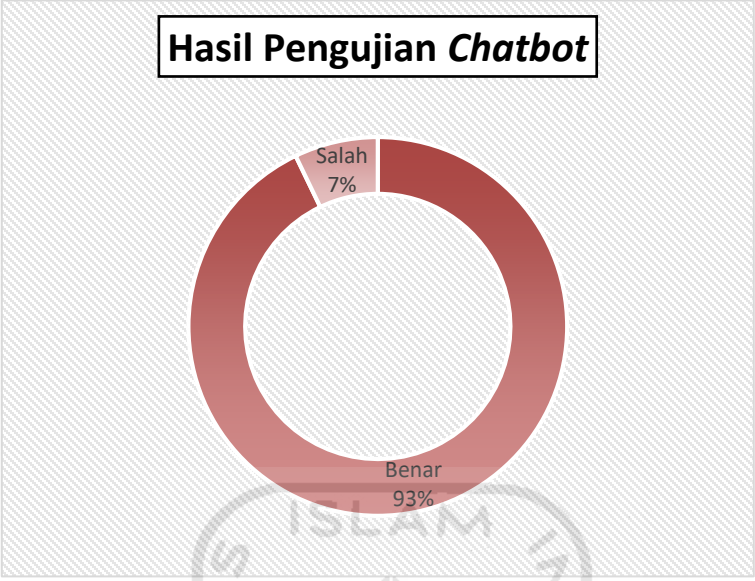
Setelah didapati hasil dari model klasifikasi SVM dengan Kernel *linear*, selanjutkan model akan dievaluasi dengan menggunakan data pertanyaan baru yang tidak terdapat didalam data latih. Dari gambar 5.21 dapat dilihat hasil evaluasi dari prediksi kategori yang mana menunjukan hasil yang memuaskan yaitu dari 10 pertanyaan baru tidak ada kesalahan yang dilakukan oleh mesin. Walaupun hasil klasifikasi mesin memuaskan, mesin masih perlu ditraining atau dilatih dengan data yang lebih banyak

5.6 Statistika Deskriptif



Gambar 5.22. Persentase pertanyaan *user*

Berdasarkan Visual pada **Gambar 5.22**, diketahui bahwa pertanyaan yang paling banyak banyak diajukan adalah pertanyaan kategori Steam Store dengan Presentase sebesar 68%. Kemudian kategori Music bot mendapatkan presentase 18%. Sedangkan pertanyaan yang paling sedikit diajukan adalah kategori Game Bot dan Steam Event dengan presentase 7%.



Gambar 5.23. Persentase Hasil Pengujian Chatbot

Dilakukan pengujian sebanyak 84 pertanyaan yang diajukan oleh *user* Discord, dari **Gambar 5.23** didapati bahwa Chatbot menjawab dengan benar pertanyaan yang diajukan oleh *user* mendapat presentase 93% sedangkan jawaban salah mendapatkan presentase 7%.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan sebagai berikut :

1. Gambaran umum data pertanyaan chatbot dengan user yaitu pertanyaan yang paling banyak banyak diajukan adalah pertanyaan kategori Steam Store dengan Presentase sebesar 68%. Kemudian kategori Music bot mendapatkan presentase 18%. Sedangkan pertanyaan yang paling sedikit diajukan adalah kategori Game Bot dan Steam Event dengan presentase 7%.
2. Prototype MyBotS telah berhasil dibangun dan chatbot hanya dapat menjawab pertanyaan yang sudah direpresentasikan didalam database. Didapatkan akurasi chatbot dari 84 pertanyaan yang diajukan adalah sebesar 93%.
3. Berdasarkan hasil analisis klasifikasi Support Vector Machine (SVM) dengan tiga kernel yaitu kernel Linear, Polynomial, dan RBF dengan data pengujian, nilai akurasi kategori prediksi adalah SVM terbesar dengan kernel Linear, yaitu 94 % dan kesalahan prediksi 6%. .Sistem program chatbot dapat memberikan respon jawaban dengan nilai akurasi 100% dan error 0% dari 10 pertanyaan baru.

6.2 Saran

Dari penelitian yang telah dilakukan peneliti memberikan saran kepada penelitian selanjutnya diantaranya adalah sebagai berikut :

1. Pada Penelitian ini topik yang di bahas di data pertanyaan kurang variatif, disarankan untuk lebih banyak mengumpulkan dataset pertanyaan lagi.
2. Mengembangankan prses klasifikasi dengan kategori yang lebih dari dua (*multiclass*)

3. Mengambangkan chatbot menjadi prototype yang lebih komplite dan sedikit kekurangan
4. Mengembangkan chatbot lebih bervariasi dan berbagai fitur yang menarik.

DAFTAR PUSTAKA

Amalia, Eka, L. dan Wibowo, Dimas, W. (2019). Rancang Bangun Chatbot Untuk Meningkatkan Performa Bisnis., *Jurnal Teknologi Informasi dan Ilmu Komputer*. Vol. 13, No. 2.

Abdul-Kader and Woods. (2015). Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications. Vol. 6,* 72-80.

Asiyah, S. N., & Fithriasari, K. (2016). Klasfikasi Berita Online Menggunakan Metode Support Vector Machine dan K- Nearest Neighbor. *Jurnal Sains dan Seni ITS Vol. 5 No. 2* , 317-322

Baiti, Z. N. (2013). *Aplikasi Chatbot "Mi3" Untuk Informasi Jurusan Teknik Informatika Berbasis Sistem Pakar Menggunakan Metode Forward Chaining*. Malang: UIN Maulana Malik Ibrahim .

Barakbah, A. R. (2005). *Natural Language Processing.* Surabaya: Institut Teknologi Sepuluh Nopember.

Budiasa. (2007). *Aplikasi Sederhana Pattern Matching dengan Algoritma Brute Force Pada Validasi suatu teks*. Institute Teknik Bandung: ITB.

Cristiani N., Taylor J.S.. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge Press University.

Desiani, dan Arhami. (2006). *Konsep Kecerdasan Buatan.* Yogyakarta: Penerbit ANDI.

Estoatnowo. (2016). *Klasifikasi Status Kelulusan Mahasiswa Menggunakan Metode CHAID dan Algoritma C4.5 (Studi Kasus : Mahasiswa Jurusan Statistika Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia).* Skripsi.

Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Islam Indonesia: Yogyakarta.

Fattah, F. 2014. *Twitter Text Mining Untuk Informasi Gempa Bumi Menggunakan TF-IDF di Indonesia*. Tugas Akhir. Program Studi Teknik Informatika Fakultas Teknologi Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim: Malang

Husein, M. F., Darusalam, U., dan Aningsih, A. (2020) Aplication of the O-Chat Bot Program to Provide Learning Motivation to National University Students Using AIML., *Jurnal Mantik*, Vol. 4, No. 1.

Kohavi, R., dan Provost, F., (1998), *On Applied Research in Machine Learning. In Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Columbia University, New York, Volume 30.

M. d. G. B. Marietto, et al.(2013), "Artificial Intelligence Markup Language: A Brief Tutorial," *arXiv Preprint* ArXiv:1307.3091.

Miner, G., Elder, J., Fast, A., Hill, T., Nisbet, R., dan Delen, D. (2012). *Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications*. Oxford: Elsevier.

Mooney, R. J. (2006), *Machine Learning Text Categorization*. Austin: University of Texas.

Mahdiyah, Evfi dan Andriyani.(2013). Analisa Algoritma Pemahaman Kalimat Pada ALICE ChatBot Dengan Menggunakan Artificial Intelligence Markup Language (AIML). *Conference: SEMIRATA 2013*, Vol. 1, No. 1.

Nugraha, D. dan Winiarti, S.(2014) PENGEMBANGAN MEDIA PEMBELAJARAN SISTEM PELACAKAN PADA MATA KULIAH KECERDASAN BUATAN BERBASIS MULTIMEDIA. *Jurnal Sarjana Teknik Informastika*, Vol. 2, No. 1.

Nugroho, A. S., Witarto, A. B., & Handoko, D. (2003). Support Vector Machine: Teori dan Aplikasinya dalam Bioinformatika. *Kuliah Umum IlmuKomputer.Com*.

Savero, Rico. (2017) Rancang Bangun Aplikasi Chatbot Penjualan Mobil Berbasis Artificial Intelligence Markup Language Menggunakan Algoritma Porter Stemmer. *Semantic Scholar*.

Sugianto, S.R. (2005). *Penerjemah Bahasa Jawa-Indonesia Menggunakan Natural Language Processing Dengan Metode Context-Free Recursive-Descent*. Skripsi. Universitas Islam Syarif Hidayatullah

Raihan, Jade, P., dan Putri, Yuliani, R. (2018). Pola Komunikasi Group Discord PUBG.INDO.FUN MELALUI APLIKASI DISCORD. *e-Proceeding of Management*, Vol. 5, No. 3, ISSN:2355-9357.

Septiansyah, R., Akbar, S., dan Maulana, R. (2018). Perancangan Bot Pada Discord Untuk Pembacaan Sensord Di Raspberry Dengan Sistem Learning Yang Dinamis. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, Vol 2, No. 10.

Niell, S., dkk. (2018). Beehives biomonitor pesticides in agroecosystems: Simple chemical and biological indicators evaluation using Support Vector Machines (SVM). *Ecological Indicators*, Vol 91, 149-154.

Tupac Takur. (2019), *Apa Itu Steam ? Pengertian dan Fungsi Untuk Gaming*. https://sobatgame.com/apa-itu-steam/. Diakses pada 1 mei 2020.

Zahour, O., Benlahmar, E. H., Eddaoui, A., Ouchra, H., dam Hourrane, O. (2020). A system for educational and vocational guidance in morocco:Chatbot E-Orentation. *Jurnal Procedia Computer Science*, Volume 175, Hal. 554-559,.

LAMPIRAN

Lampiran 1 Dataset Data Pertanyaan

Kategori	Pertanyaan
Steam Store	!J Steam Store
Steam Store	!J Steam Game MHW
Steam Store	!J Steam Game Monster Hunter
Steam Store	!J Steam Game Monster Hunter World
Steam Store	!J Steam Game D2
Steam Store	!J Steam Game Destiny 2
Steam Store	!J Steam Game R6
Steam Store	!J Steam Game Rainbow six
Steam Store	!J Steam Game RDR 2
Steam Store	!J Steam Game Red Dead Redemtion 2
Steam Store	!J Steam Game PZ
Steam Store	!J Steam Game Planet Zoo
Steam Store	!J Steam Game Stellaris
Steam Store	!J Steam Game Original Sin 2
Steam Store	!J Steam Game Divinity
Steam Store	!J Steam Game Divinity : Original Sin 2
⋮	⋮
⋮	⋮
Music Bot	!J Play music jazz
Music Bot	!J Skip
Music Bot	!J Stop
Game Bot	!rock
Game Bot	!paper
Game Bot	!scissors
Game Bot	!paper dong
Game Bot	!scissors dong
Game Bot	!rock dong

Lampiran 2 Script dan output analisis SVM

```
import numpy as np
import pandas as pd
data = pd.read_excel('Data_Bot2.xlsx')
data.head()
```

	Kategori	Pertanyaan
0	Steam Store	!J Steam Store
1	Steam Store	!J Steam Game MHW
2	Steam Store	!J Steam Game Monster Hunter
3	Steam Store	!J Steam Game Monster Hunter World
4	Steam Store	!J Steam Game D2

```
data.shape #melihat jumlah baris dan kolom
```

```
(84, 2)
```

```
col = ['Kategori', 'Pertanyaan']
data = data[col]
```

```
data.columns
```

```
Index(['Kategori', 'Pertanyaan'], dtype='object')
```

```
data['kategori_id'] = data['Kategori'].factorize()[0]
from io import StringIO
kategori_id_data = data[['Kategori', 'kategori_id']].drop_duplicates().sort_value
kategori_to_id = dict(kategori_id_data.values)
id_to_kategori = dict(kategori_id_data[['kategori_id', 'Kategori']].values)
```

```
data.tail()
```

	Kategori	Pertanyaan	kategori_id
79	Game Bot	!paper	3
80	Game Bot	!scissors	3
81	Game Bot	!rock dong	3
82	Game Bot	!paper dong	3
83	Game Bot	!scissors dong	3

```
countvec3 = pd.DataFrame(countvec, columns=kata_kata)
countvec3
```

	age	aoe	by	chief	china	clancys	clansys	clasic	club	collection	...	tournament	truck
0	0	0	0	0	0	0	0	0	0	0	...	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
79	0	0	0	0	0	0	0	0	0	0	...	0	0
80	0	0	0	0	0	0	0	0	0	0	...	0	0
81	0	0	0	0	0	0	0	0	0	0	...	0	0
82	0	0	0	0	0	0	0	0	0	0	...	0	0
83	0	0	0	0	0	0	0	0	0	0	...	0	0

84 rows × 106 columns

```
from xlsxwriter.utility import xl_rowcol_to_cell
saveresult = pd.ExcelWriter('saveresult22.xlsx', engine='xlsxwriter')
countvec3.to_excel(saveresult, index=False, sheet_name='report')
saveresult.save()
```

```
from sklearn.feature_extraction.text import TfidfTransformer
transformer = TfidfTransformer(norm=None, use_idf=True, smooth_idf=False, subline
tfidf = transformer.fit_transform(countvec)
tfidf
```

```
<84x106 sparse matrix of type '<class 'numpy.float64'>'
        with 292 stored elements in Compressed Sparse Row format>
```

```
tfidf1 = tfidf.toarray()
tfidf1
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
from xlsxwriter.utility import xl_rowcol_to_cell
savedf1 = pd.ExcelWriter('savdf1.xlsx', engine='xlsxwriter')
df1.to_excel(savedf1, index=False, sheet_name='report')
savedf1.save()
```

```
result = pd.concat([data['kategori_id'],df1], axis=1)
result
```

	kategori_id	age	aoe	by	chief	china	clancys	clansys	clasic	club	...	tournament	truck
0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
2	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
3	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
4	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
79	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
80	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
81	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
82	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
83	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

84 rows × 107 columns

```
#untuk menyimpan hasil cell output result menjadi format excel
#Link http://pbpython.com/improve-pandas-excel-output.html
from xlsxwriter.utility import xl_rowcol_to_cell
saveresult = pd.ExcelWriter('saveresult21.xlsx', engine='xlsxwriter') #nilai TF-i
result.to_excel(saveresult, index=False, sheet_name='report')
saveresult.save()
```

```
y = result['kategori_id']
y
```

```
0     0
1     0
2     0
3     0
4     0
     ..
79    3
80    3
81    3
82    3
83    3
Name: kategori_id, Length: 84, dtype: int64
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
 df1, y , test_size=0.2, random_state=123
)
```

```
x_test
```

	age	aoe	by	chief	china	clancys	clansys	clasic	club	collection	...	tournament	t
70	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
23	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
4	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
79	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
62	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	4.73767	
71	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
76	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
37	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
31	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
78	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
8	0.000000	0.0	0.0	0.0	0.0	5.430817	0.0	0.0	0.0	0.0	...	0.00000	
45	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
9	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
51	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
56	5.430817	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
65	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	
24	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	...	0.00000	

17 rows × 106 columns

```
from xlsxwriter.utility import xl_rowcol_to_cell
saveresult = pd.ExcelWriter('saveX_test.xlsx', engine='xlsxwriter')
x_test.to_excel(saveresult, index=False, sheet_name='report')
saveresult.save()
```

```
y_test
```

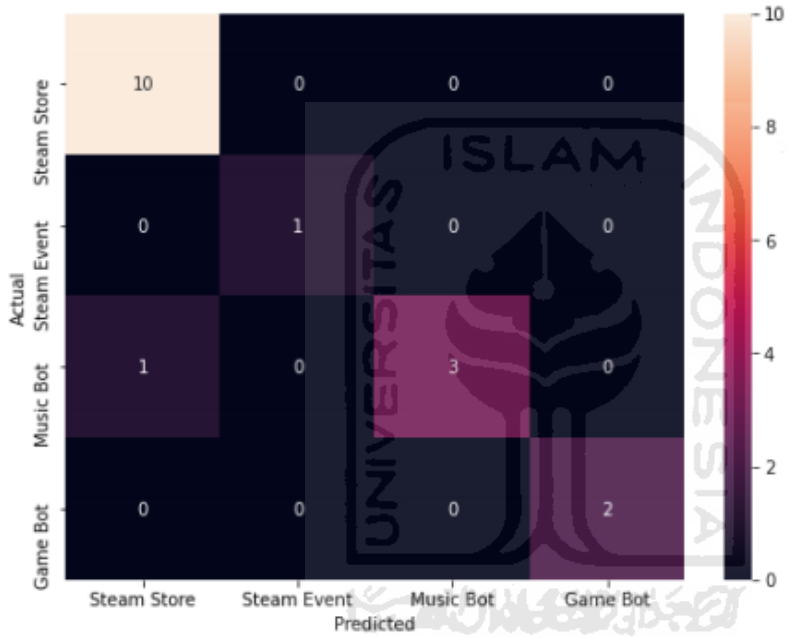
```
70    2
23    0
4     0
79    3
62    1
71    2
76    2
37    0
31    0
78    3
8     0
45    0
9     0
51    0
56    0
65    2
24    0
Name: kategori_id, dtype: int64
```

```
from xlsxwriter.utility import xl_rowcol_to_cell
saveresult = pd.ExcelWriter('savey_train.xlsx', engine='xlsxwriter')
y_train.to_excel(saveresult, index=False, sheet_name='report')
saveresult.save()
```

```
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC

def train_svm(x_train, y_train):
    """
    Create and train the Support Vector Machine.
    """
    svm = SVC(kernel='linear')
    svm.fit(x_train, y_train)
    return svm
# Create and train the Support Vector Machine

svm = train_svm(x_train, y_train)

# Make an array of predictions on the test set
pred = svm.predict(x_test)

# Output the hit-rate and the confusion matrix for each model
print(svm.score(x_test, y_test))
print(confusion_matrix(pred, y_test))
```

```
0.9411764705882353
[[10  0  1  0]
 [ 0  1  0  0]
 [ 0  0  3  0]
 [ 0  0  0  2]]
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=kategori_id_data.Kategori.values, yticklabels=kategori_id
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.savefig('gambarplot.png', transparent='True')
plt.show()
```




```
from IPython.display import display

for predicted in kategori_id_data.kategori_id:
  for actual in kategori_id_data.kategori_id:
    if predicted != actual and conf_mat[actual, predicted] >= 10:
      print("'{}' predicted as '{}' : {} examples.".format(id_to_kategori[actual]
      display(data.loc[indices_test[(y_test == actual) & (y_pred == predicted)]][
      print('')
```

```
model.fit(features, labels)
```

```
LinearSVC()
```

```
texts = ["1) !J Game Steam Fall Guys?",
         "2) !J Game terbaru Steam tahun 2020?",
         "3) !J Steam Store 2021?",
         "4) !J Game Steam GTA 6 ?",
         "5) !J Game Steam Action ?",
         "6) !J Play Music Syar'i?",
         "7) !J Event Of Years?",
         "8) !J Date Years Event?",
         "9) !rock pilihanku ?",
         "10) !scissors kesukaanku ?"]
text_features = tfidf.transform(texts)
predictions = model.predict(text_features)
for text, predicted in zip(texts, predictions):
  print('"{}"'.format(text))
  print("  - Prediksi: '{}'".format(id_to_kategori[predicted]))
  print("")
```

```
"1) !J Game Steam Fall Guys?"
  - Prediksi: 'Steam Store'

"2) !J Game terbaru Steam tahun 2020?"
  - Prediksi: 'Steam Store'

"3) !J Steam Store 2021?"
  - Prediksi: 'Steam Store'

"4) !J Game Steam GTA 6 ?"
  - Prediksi: 'Steam Store'

"5) !J Game Steam Action ?"
  - Prediksi: 'Steam Store'

"6) !J Play Music Syar'i?"
  - Prediksi: 'Music Bot'

"7) !J Event Of Years?"
  - Prediksi: 'Steam Event'

"8) !J Date Years Event?"
  - Prediksi: 'Steam Event'

"9) !rock pilihanku ?"
  - Prediksi: 'Game Bot'

"10) !scissors kesukaanku ?"
  - Prediksi: 'Game Bot'
```

Lampiran 3 Script Chatbot

```
const botconfig = require("./botconfig.json");
const { Client, Collection, MessageEmbed } = require("discord.js");
const bot = new Client();
const fs = require("fs");
bot.commands = new Collection();
==========================================================================
// array of commands
const commandsArray = [""];

fs.readdir("./commands/", (err, files) => {

    if(err) console.log(err)
    }
    jsfile.forEach((f, i) => {
    });
});
==========================================================================
bot.on("ready", async () => {
    console.log(`Logged in as ${bot.user.username} is online`)
    bot.user.setActivity("Work", {type: "WATCHING"});
});
==========================================================================
bot.on("message", async message =>{
    if(message.author.bot || message.channel.type === "dm") return;
});
==========================================================================
//Steam Store and Game
// script Steam bot
bot.on('message', msg01 => {
    if
});
            return;
    }
})
==========================================================================
ot.on("message", msg2 => {
    const msgToLower = msg2.content.toLowerCase();
    const gameArray = ["rock", "paper", "scissors"];
bot.login(botconfig.token);
```