Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning

Yurio Windiatmoko
19917020

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer
Konsentrasi Sains Data
Program Studi Informatika Program Magister
Fakultas Teknologi Industri
Universitas Islam Indonesia
2023

Lembar Pengesahan Pembimbing

Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning

Yurio Windiatmoko

19917020

Yogyakarta, 30 Januari 2023

Pembimbing I

Dhomas Hatta Fudholi, S.T., M.Eng.,

Ph.D.

Lembar Pengesahan Penguji

Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning

Yurio Windiatmoko

19917020

Yogyakarta, 30 Januari 2023

Tim Penguji,

Dhomas Hatta Fudholi, ST., M.Eng., Ph.D

Ketua

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D

Anggota I

Ahmad M. Raf'ie Pratama, Ph.D.

Anggota II

Mengetahui,

Ketua Program Studi Informatika Program Magister

Universitas Islam Indonesia

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.

Abstrak

Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning

Sistem cerdas untuk universitas yang didukung oleh kecerdasan buatan telah dikembangkan dalam skala besar untuk membantu orang dengan berbagai tugas. Konsep chatbot bukanlah hal baru di masyarakat saat ini yang berkembang dengan teknologi terkini. Mahasiswa atau calon mahasiswa seringkali membutuhkan informasi yang aktual, seperti menanyakan kepada customer service tentang universitas, terutama di masa pandemi saat itu, ketika sulit untuk mengadakan pertemuan pribadi secara langsung. Chatbot dimanfaatkan secara fungsional sebagai informasi jadwal perkuliahan, informasi nilai mahasiswa. Berbagai data sebelumnya dipertimbangkan dari bermacam skema skenario percakapan. Data percakapan disimulasikan dan di-*generate* sebagai landasan chatbot dalam memberikan respons. Chatbot melakukan proses pembelajaran melalui proses *training* data dari percakapan tersebut. Chatbot ini dikembangkan dengan model *Deep Learning* yang diadopsi oleh *Artificial Intelligence* yang mereplikasi kecerdasan manusia dengan skema pelatihan tertentu. *Deep Learning* yang diterapkan berbasis *RNN* yang memiliki skema penyimpanan memori khusus untuk model, khususnya pada bot percakapan ini menggunakan *GRU* yang terintegrasi dengan framework chatbot *RASA*. *GRU* juga dikenal sebagai *Gated Recurrent Unit*, yang secara efektif menyimpan sebagian memori yang diperlukan, tetapi menghapus bagian yang tidak diperlukan. Hasil penelitian dan pengembangan yang didapat, yaitu model *GRU* memiliki skor presisi rata-rata 0,99 dalam ketepatan memberikan respons chat. Penyajian chatbot ini dilakukan oleh pengembangan platform aplikasi web pada halaman mi-gateway yang dibuat oleh framework *React JavaScript*.

Kata kunci:

Artificial Intelligence, Chatbot, Deep Learning, Chatbot Framework, RNN, RASA, GRU

Abstract

Mi-Botway: a Deep Learning-based Intelligent University Enquiries Chatbot

Intelligent systems for universities that are powered by artificial intelligence have been developed on a large scale to help people with various tasks. The chatbot concept is nothing new in today's society, which is developing with the latest technology. Students or prospective students often need actual information, such as asking customer service about the university, especially during that pandemic, when it is difficult to hold a personal meeting in person. Chatbots utilized functionally as lecture schedule information, student grades information. Various data previously considered from various conversation schemes. Conversation data is simulated and generated as a basis for the chatbot to respond. The chatbot carries out the learning process through the process of training data from the conversation. This conversation bot was developed with a deep learning model adopted by an artificial intelligence model that replicates human intelligence with a specific training scheme. The deep learning implemented is based on RNN which has a special memory storage scheme for models, in this particular conversation bot is using GRU which integrated into RASA chatbot framework. GRU is also known as Gated Recurrent Unit, which effectively stores a portion of the memory that is needed, but removes the part that is not necessary. The research and development results obtained, namely the GRU model has an Average Precision Score of 0.99 in the accuracy of providing chat responses. The representation of this chatbot is done by the web application platform development on the mi-gateway page created by the React JavaScript framework.

Keywords:

Artifial Intelligence, Chatbot, Deep Learning, Chatbot Framework, RNN, RASA, GRU

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, 30 Januari 2023

Yurio Windiatmoko, S.Si

Daftar Publikasi

Publikasi yang menjadi bagian dari tesis

Windiatmoko, Yurio., Fathan, Ahmad., Hatta, Dhomas., Rahmadi, Ridho. (2022). "Mi-Botway: a Deep Learning-based Intelligent University Enquiries Chatbot." *IJAIR (International Journal Of Artificial Intelligence Research.* Vol 6, No 1 (2022) [1].

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Yurio Windiatmoko	Melakukan komputasi dan analisis model Menulis *paper*
Ahmad Fathan Hidayatullah	Melakukan analisis model Menulis *paper*
Dhomas Hatta Fudholi	Melakukan analisis model Menulis *paper*
Ridho Rahmadi	Memberikan saran analisis model

[1] IJAIR (2022). Vol 6, No 1 (2022).. https://ijair.id/index.php/ijair/article/view/247

Halaman Kontribusi

Dalam penulisan tesis ini pembimbing I dan II memberikan beberapa masukkan sebagai perbaikan dari cara penulisan tesis serta memberikan saran tentang data yang akan diolah dan dianalisis.

Halaman Persembahan

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Kupersembahkan Tesis ku ini dengan setulus hati & sepenuh jiwa untuk :

Istriku Mira tercinta, dan kedua Orang tuaku Ibuku Sri Rahayu, Ayahku Winarno yang selalu mengiringi langkah kakiku dengan do'a, yang tak pernah putus asa untuk mengurus, mendidik, menasehati, menyemangati, membimbing serta memberikan motivasi kepadaku.

Kakakku Yurito yang sudah mendahuluiku meninggal dunia, Insya Allah amal ibadahnya diterima di sisi-Nya, sebelumnya selalu mensupport doa dan dukungan saat memulai kuliah magister ini.

Dan terspesial untuk

Diri sendiri, terimakasih karena telah berjuang sejauh ini dengan melawan ego dan mood yang tidak tentu selama penulisan tesis ini.

Kata Pengantar

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Alhamdulillahirabbil'alamin, segala puji syukur kehadirat Allah SWT senantiasa penulis panjatkan karena atas limpahan rahmat dan hidayah-Nya, tesis yang berjudul "Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning" dapat berjalan dengan lancar dalam penyelesaiannya. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Dalam penyelesaian Tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak, untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
2. Ibu Izzati Muhimmah, S.T., M.Sc., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia sebelumnya.
3. Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia.
4. Bapak Dr. Ing. Ridho Rahmadi, S.Kom., M.Sc., selaku dosen pembimbing satu awal mulanya, yang telah banyak membantu penulis dalam memberikan ide, saran dan kritiknya.
5. Bapak Ahmad Fathan Hidayatullah, S.T., M.Cs., selaku dosen pembimbing dua yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini
6. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku dosen pembimbing satu yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
7. Bapak Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D., selaku dewan penguji tesis 1 yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
8. Bapak Ahmad M. Raf'ie Pratama, Ph.D., selaku dewan penguji tesis 2 yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.

9. Dosen Program Studi Magister Teknik Informatika yang telah memberikan bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal jariyah di dunia maupun akhirat.
10. Staff Akademik Program Pascasarjana Fakultas Teknologi Universitas Islam Indonesia, yang telah membantu dalam segala urusan administrasi di kampus.
11. Sahabat seperjuangan Sains Data (Eko, Malik, Fahmi, Yopi, Satya, dan Rifai) yang telah turut serta membantu, mendukung, menyemangati serta memberikan masukan kepada penulis selama menjalani studi di Program Pascasarjana Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.
12. Teman-teman Magister Teknik Informatika Universitas Islam Indonesia Yogyakarta se-angkatan.

Penulis menyadari bahwa dalam penulisan tesis ini masih banyak kelemahan dan kekurangan, untuk itu kritik dan saran yang sifatnya membangun sangat penulis harapkan agar tesis ini dapat menjadi lebih baik.

Daftar Isi

Daftar Tabel

Daftar Gambar

Glosarium

RNN	- Recurrent Neural Network
GRU	- Gated Recurrent Unit
LSTM	- Long Short-Term Memory
API	- Application Programming Interface
AP	- Average Precision
AR	- Average Recall
AF	- Average F1 Score
NLP	- Natural Language Processing
NLU	- Natural Language Understanding
AI	- Artificial Intelligence
FAQ	- Frequently Ask Question

BAB 1
Pendahuluan

1.1 Latar Belakang

Chatbot adalah sistem yang paling mudah diakses untuk setiap pengguna, ini adalah teknologi kecerdasan buatan untuk berkomunikasi melalui bahasa alami dengan manusia. Chatbot memungkinkan untuk memahami bahasa manusia melalui *natural language processing*. Bahasa adalah protokol timbal balik sosial dan budaya secara alami. Sudah saatnya mesin komputer menambahkan fitur ini untuk dapat memahami manusia seiring dengan meningkatnya kinerja perangkat keras mesin komputer saat ini (Yoo & Jeong, 2020).

Mahasiswa atau calon mahasiswa seringkali membutuhkan informasi yang *up-to-date*, misalnya dalam menanyakan sesuatu ke *customer service* apapun perihalnya itu mengenai kampus, apalagi di masa pandemi saat itu sulit untuk melakukan pertemuan tatap muka. Oleh karena itu diperlukan chatbot untuk menjawab pertanyaan dengan cepat, benar, dan tersedia selama 24 jam (Al-fakhri dkk., 2019). Secara fungsional, chatbot ini membantu dalam beberapa hal seperti info jadwal kuliah, informasi nilai mahasiswa.

Terdapat pertimbangan utama dalam memilih fitur utama chatbot, yaitu memberikan informasi jadwal kuliah dikarenakan adanya kesulitan mahasiswa saat menginginkan info jadwal terupdate dari akademik, harus membuka email dan melihat kiriman email update jadwal terbaru, maka dengan adanya chatbot yang sudah terintegrasi dengan sistem jadwal akademik mahasiswa cukup menanyakan jadwal pada chatbot ini. Selain itu pemilihan fitur utama lainya yaitu informasi nilai mahasiswa, ini didasari oleh adanya kesulitan mahasiswa dalam melihat update nilai pribadi yang harus login di sistem berulang yang terkesan cukup ribet, maka dengan adanya chatbot yang sudah terintegrasi dengan sistem nilai akademik mahasiswa, hal itu cukup ditanyakan langsung melalui chatbot ini. Untuk autentikasi saat menanyakan nilai, chatbot diharuskan memverifikasikan mahasiswa terkait melalui link SSO yang dikirimkan sebelum mengeluarkan daftar nilai. Namun pada penelitian chatbot ini belum dilakukan pengembangan sistem autentikasi tersebut, dikarenakan batasan pengembangan disini masih berupa tahap prototyping saja.

Tren pengembangan chatbot sendiri dalam implementasinya untuk *customer service* perguruan tinggi atau kampus dikembangkan (Al-fakhri, S. et al., 2019) dengan menggunakan aplikasi LINE dan masih menggunakan *rule-based* mapping untuk respons

chatbot. Perkembangan chatbots menggunakan *deep learning* untuk *customer service* perguruan tinggi atau kampus belum ditemukan, terutama dalam penggunaan bahasa Indonesia.

Sehingga untuk membantu kontribusi lingkup akademik dalam hal, pengembangan chatbot yang menggunakan bahasa indonesia, dan tidak lagi menggunakan *rule-based mapping* respon dari chatbot, penelitian ini mengusulkan sebuah chatbot berbahasa indonesia dengan menggunakan model *deep learning*. Supaya menambah khasanah chatbot *deep learning* berbahasa indonesia yang lebih mutakhir dan fleksibel juga saat berinteraksi, dikarenakan sudah tidak lagi menggunakan *rule-based mapping* yang terkesan kaku dengan syarat tertentu saat mengirimkan pesan pada chatbot.

Beberapa kelemahan *rule-based mapping* chatbot adalah tidak dapat menangkap kesalahan ketik atau *typos* yang berarti bahwa dalam beberapa kasus chatbot tidak akan memahami user, yang dapat menyebabkan frustrasi pihak user. Interaksi dengan chatbot *rule-based* terasa terlalu sederhana dan kaku seperti bercakap dengan robot dibandingkan dengan sebuah percakapan yang lebih natural. Untuk pengembangan berkala atau berkelanjutan pada *rule-based* chatbot dalam setiap perbaikannya perlu dilakukan secara manual dengan mengubah logic kode sedangkan, pada *AI / deep learning* chatbot perbaikan cukup dengan menambah data yang akan digunakan untuk men-training ulang model *deep learning* chatbot itu sendiri. Adapun beberapa *library parser* pada sebuah bahasa pemrograman yang diharapkan dapat mendukung penerapan *rule based* chatbot, masih belum mumpuni untuk mengekstrak entitas-entitas dalam sebuah tata bahasa tertentu terutama Bahasa Indonesia.

Awal mula deep learning dimulai pada tahun 2006, yaitu setelah Geoffrey Hinton mempublikasikan paper yang memperkenalkan salah satu varian *neural network* yang disebut *deep belief nets*. Paper ini merupakan awal kemunculan istilah *deep learning*, untuk membedakan arsitektur *neural network* konvensional (*single* layer) dengan arsitektur *neural network* multi-layer. *Deep learning* adalah salah satu cabang *machine learning* yang menggunakan *deep neural network* untuk menyelesaikan permasalahan pada domain *machine learning* (Muhamad Alfarisi, 2020). *Deep learning* terdiri dari algoritma pemodelan abstrak tingkat tinggi pada data, menggunakan transformasi fungsi non-linier yang terdiri dari beberapa lapisan dan secara mendalam (Dadang, 2018). *Deep learning* juga merupakan cabang dari *machine learning*, yang terinspirasi oleh neuron otak manusia diwakili oleh lapisan pada jaringan saraf (Santoso & Ariyanto, 2018). Metode *deep learning* yang digunakan dalam penelitian ini berbasis RNN dengan tipe khusus yang disebut *GRU*

(Gated Recurrent Unit), model ini memiliki beberapa skema penyimpanan memori khusus dalam deep learning. *GRU* mampu menghemat sebagian memori yang diperlukan secara efisien dan juga menghapus sebagian memori yang tidak perlu (Phi, 2018). Penelitian ini juga menggunakan framework bernama RASA untuk membangun aplikasi chatbot ini, yang akan diintegrasikan dengan *GRU* sebagai model *deep learning* di dalamnya. Implementasi *deep learning* pada chatbot membutuhkan peran *framework* chatbot, yaitu untuk menyatukan dan menyederhanakan proses *pipelining* model *deep learning* dan manajemen alur percakapan. RASA bekerja pada dua prosedur utama yaitu *RASA NLU* dan *RASA Core* (RASA, 2022j). *RASA* adalah library bahasa pemrograman *python* yang bersifat *open source* untuk membangun piranti lunak chatbot. Tujuan dan filosofi dari penggagas *framework* chatbot RASA adalah untuk menciptakan manajemen dialog berbasis *machine learning* yang memberikan kemudahan pengguna dalam hal implementasi, serta *bootstraping* yaitu memulai dari sumber yang ada untuk membuat hal-hal yang lebih kompleks dengan cara yang lebih efisien, bahkan dimulai dengan data yang lebih sedikit atau dengan data pelatihan yang minimum (Bocklisch dkk., 2017).

Berdasarkan uraian sebelumnya terutama dikarenakan chatbot universitas yang menggunakan Bahasa Indonesia masih sedikit sekali, sejauh ini baru ditemukan satu yaitu yang dikembangkan (Al-fakhri, S. et al., 2019) dengan menggunakan aplikasi LINE dan belum mengimplementasikan deep learning didalamnya. Maka Penulis tertarik untuk melakukan pengembangan chatbot universitas berbahasa Indonesia yang mengimplementasikan deep learning dalam bentuk tesis yang berjudul "Mi-Botway: Chatbot FAQs Universitas Berbasis Deep Learning". Judul ini bernama Mi-Botway dikarenakan implementasi akan memanfaat kan situs web magister TI kampus yaitu Migateway untuk menyematkan chatbot didalamnya. Model yang diteliti dan dikembangkan akan melalui evaluasi dan perbandingan antar model yaitu *Standard RNN*, *LSTM* dan *GRU*, dimana sebelumnya penelitian ini pernah dilakukan pada acara konferensi ICITDA, dalam penelitian tersebut hanya menggunakan model *default framework RASA* yaitu model *RNN LSTM*. Pertimbangan dalam memilih arsitektur *neural network* yaitu *RNN* bukan arsitektur *neural network* lainnya itu dikarenakan model *RNN* sangat tepat untuk mempelajari pola teks bahasa terutama di konteks chatbot ini yang membutuhkan ketepatan respons chatbot terhadap user. Kemudian representasi implementasi chatbot penelitian saat ini menggunakan aplikasi web React yang disematkan di beranda mi-gateway UII, sedangkan sebelumnya hanya menggunakan integrasi facebook messenger. Penelitian ini diharapkan dapat menjadi pemantik awal untuk membangun sistim *smart campus* di universitas.

1.2 Rumusan Masalah

Sistem *smart campus* yang akan membantu otomasi berbagai kegiatan untuk saat ini sangat diperlukan untuk mengurangi pekerjaan yang berulang. Chatbot merupakan salah satu gerbang menuju system smart campus. Chatbot fungsional yang dibuat selama ini didominasi oleh sistem *rule-based* yang berkesan sangat kaku. Melalui pengembangan model deep learning, ini sangat memungkinkan chatbot dibangun dengan respon dan penggunaan yang lebih fleksibel, sehingga bagaimanakah proses membangun, merancang lalu mengevaluasi model *deep learning* pada chatbot universitas?

1.3 Tujuan Penelitian

Dengan mengetahui bagaimana cara membangun, merancang dan mengevaluasi model *deep learning* pada chatbot secara efisien, maka chatbot yang dapat mensupport sistem *smart campus* dapat dikembangkan dan diimplementasikan. Sehingga sentralisasi sistem otomasi dan informasi pada kampus juga akan menjadi lebih fleksibel melalui chatbot.

1.4 Batasan Masalah

1. Penelitian model deep learning chatbot ini hanya dibatasi dengan *scope* percakapan informasi jadwal perkuliahan, informasi nilai mahasiswa.
2. Penelitian ini menggunakan sistem kerangka kerja *open-source* RASA.
3. Platform chatbot yang diimplementasikan pada web tidak sampai integrasi dengan halaman Mi-Gateway, namun hanya dummy pages halaman Mi-Gateway saja yang disematkan chatbot dan hanya merupakan *prototyping.*
4. Platform chatbot yang diimplementasikan hanya merupakan *prototyping* tidak menggunakan system *authentication* pengguna atau sistem login.
5. Siklus untuk implementasi platform pada *Production Level* tidak dilakukan pada penelitian ini, seperti penyediaan *web admin* untuk meng-update ataupun cara lainnya dalam mengupdate data.

1.5 Manfaat Penelitian

1. Manfaat Akademis

a. Dapat menambah wawasan tentang implementasi deep learning pada chatbot secara efisien.
b. Sebagai acuan atau rujukan bagi peneliti-peneliti selanjutnya yang ingin meneliti tentang implementasi NLP dan *deep learning* pada chatbot universitas.

2. Manfaat Praktis

   Dapat menjadi sebuah referensi ilmiah yang berguna bagi universitas maupun praktisi industry lainnya dalam membangun *smart system* dengan sentralisasi AI atau chatbot. Dewasa ini masyarakat sering dibingungkan dengan beragamnya aplikasi bertebaran dimana-mana dan bagaimana cara menggunakannya. Dengan adanya sentralisasi system pada chatbot seperti ini, memungkinkan hal itu dapat diminimalisir, dikarenakan justru aplikasi yaitu chatbot yang akan berusaha memahami masyarakat melalui pemahaman intent atau maksud pengguna.

1.6 Sistematika Penulisan

1. Abstrak berisi kesimpulan laporan secara umum.
2. Pernyataan Keaslian Tulisan.
3. Daftar Publikasi berisi laporan ini yang dipublish pada Jurnal Penelitian.
4. Halaman Kontribusi berisi dosen yang terlibat dalam penelitian.
5. Halaman Persembahan berisi ucapan terhadap keluarga.
6. Kata Pengantar berisi ucapan terimakasih kepada berbagai pihak yang terlibat dalam penelitian.
7. Daftar Isi berisi daftar judul bab, dan sub bab laporan.
8. Daftar Tabel berisi daftar nama tabel.
9. Daftar Gambar berisi daftar nama Gambar.
10. BAB 1 Pendahuluan
    a. 1.1 Latar Belakang berisi alasan penulis memilih tema tersebut.
    b. 1.2 Rumusan Masalah berisi pertanyaan yang harus dijawab oleh penelitian.
    c. 1.3 Tujuan Penelitian berisi tujuan utama dari penelitian.
    d. 1.4 Batasan Masalah berisi batasan apa saja yang akan dibatasi tidak dilakukan dalam penelitian.
    e. 1.5 Manfaat Penelitian berisi manfaat yang didapatkan sesudah melakukan penelitian.
    f. 1.6 Sistematika Penulisan berisi daftar kisi-kisi penulisan dari Bab awal sampai akhir.

11. BAB 2 Tinjauan Pustaka
    a. Penelitian Terdahulu berisi penelitian yang dilakukan sebelumnya.
    b. Konsep Pengetahuan berisi wawasan keilmuan apa saja yang akan dipakai dalam penelitian.
12. BAB 3 Metodologi
    a. 3.1 Alur Rancang Bangun Mi-botway berisi alur tahapan yang dilakukan dalam penelitian secara berurutan yang merupakan tahapan dalam membangun chatbot Mi-Botway.
    b. 3.2 Data berisi tentang tahapan penelitian dalam mensimulasikan data serta keberagamannya untuk membangun chatbot Mi-Botway.
    c. 3.3 Modeling berisi rancangan dan skenario untuk membangun model chatbot yang diimplementasikan dengan *training* data.
    d. 3.4 Implementasi Rancang Bangun Chatbot berisi rancangan platform chatbot Mi-Botway yang diimplementasikan pada halaman web.
    e. 3.5 Deployment berisi rancangan saat *deploying* platform chatbot pada server.
    f. 3.6 Penjelasan Struktur dan Kode berisi tentang penjelasan struktur dan kode platform Mi-Botway secara menyeluruh.
13. BAB 4 Hasil Dan Pembahasan
    a. 4.1 Analisis Hasil berisi tentang analisa hasil evaluasi model yang digunakan pada penelitian.
    b. 4.2 Implementasi Interface berisi hasil implementasi chatbot Mi-Botway pada platform web.
    c. 4.3 Uji Validasi Kegunaan Pada User berisi hasil uji validasi penggunaan chatbot Mi-Botway pada beberapa user.
14. BAB 5 Kesimpulan Dan Saran
    a. 5.1 Kesimpulan berisi kesimpulan yang didapatkan setelah melakukan penelitian.
    b. 5.2 Saran berisi saran yang dapat dilakukan pada penelitian selanjutnya.
15. Daftar Pustaka berisi rujukan yang dikutip dalam laporan.
16. Lampiran berisi lampiran training data yang digunakan pada chatbot Mi-Botway dan juga alamat github yang melampirkan kode hasil penelitian yang telah dilakukan.

BAB 2
Tinjauan Pustaka

2.1 Penelitian Terdahulu

Pada bagian ini, dibahas beberapa penelitian sebelumnya mengenai implementasi chatbot yang telah dilakukan oleh peneliti lain. Terdapat dua pendekatan utama untuk menghasilkan respons chatbot. Jika chatbot menggunakan pendekatan tradisional, maka itu merupakan pendekatan dengan berbagai *template rule-based* untuk memproses tanggapan. Namun, saat ini banyak pendekatan baru yang memungkinkan keterlibatan pada *deep learning*. Model *neural network* atau jaringan saraf tiruan, model yang dilatih dengan beberapa data untuk mempelajari pola dan proses dalam menghasilkan respons yang relevan secara tata bahasa, dalam hal memahami ucapan pengguna (Csaky, 2019).

Berikut ini adalah beberapa karya chatbot yang pernah dikembangkan tentang sistem manajemen percakapan atau chatbot di universitas. Sistem obrolan online untuk pertanyaan perguruan tinggi menggunakan *database* oleh (Prashant dkk., 2017). Dalam penelitian ini, mereka melakukan *pattern matching* untuk mencari informasi dan memberi respons pada chatbot. Belum menggunakan pendekatan *deep learning*, sehingga aturannya masih terlalu kaku bagi pengguna untuk bertanya. Namun detail langkah kerja pada penelitian ini sangat baik, terdapat desain UML, dan berbagai macam diagram proses (Prashant dkk., 2017). Selanjutnya yaitu sebuah chatbot bernama Erasmus yang merupakan AI chatbot. Erasmus adalah chatbot yang disematkan pada Facebook Messenger digunakan untuk menjawab pertanyaan terkait informasi kuliah. Merancang sistem *end-to-end* menggunakan layanan cloud, mulai dari api.ai (Dialogflow), Mlab (MongoDB cloud), IBM Bluemix (webhook API), scraper import.io untuk meminimalkan proses coding atau scripting, namun yang terjadi disini adalah, dikarenakan terlalu banyaknya layanan cloud yang beragam, hal ini mempengaruhi *latency* yang cukup lama antar layanan cloud (Thakkar dkk., 2018).

Adapun sebuah chatbot *LINE messenger* yang dikembangkan (Al-fakhri, S. et al., 2019), merupakan chatbot universitas Polban dengan sistem tanya jawab chatbot berbasis *Sentence Similarity Measurement (SSM)*, dengan melakukan *spelling correction* dan *context word matching* pada proses awalannya. Alur komunikasi hanya bisa satu pertanyaan satu jawaban saja. *Datastore* untuk *knowledge* chatbot ada sekitar empat buah yaitu untuk *spellcorrection,* konteks percakapan, template percakapan, dan template jawaban. Pada

chatbot ini entitas belum bisa diekstraksi, dimana entitas tertentu berguna untuk mendapatkan informasi yang lebih spesifik, sebagai lanjutan konteks dalam berkomunikasi.

Eaglebot adalah sistem tanya jawab chatbot berbasis *Multi-Tier*, untuk mengambil jawaban dari sumber yang heterogen menggunakan arsitektur BERT. Sistem chatbot yang *scalable* dengan menggunakan 3 metode pemilihan rute respons chatbot, *mainframe* menggunakan Dialogflow, kemudian ditambah beberapa metode *completion message* dan *document reader* (Rana, 2019). Eaglebot berfungsi untuk menjawab berbagai pertanyaan yang sering ditanyakan oleh mahasiswa di lingkungan universitas. Namun, aplikasi Eaglebot masih memiliki beberapa limitasi dikarenakan oleh kerangka chatbot utama yaitu Dialogflow, yang memiliki batasan *request* terutama untuk *retrieve* respons chatbot, jika menginginkan *request* yang tanpa batas untuk pengembang chatbot, diperlukan syarat untuk berlangganan hak istimewa, atau yang biasa dikenal oleh istilah *subscribe* yaitu dengan skema pembayaran tertentu.

Analisa dialog chatbot dengan ekstraksi entitas berbasis sistem *smart chatbot* menggunakan *RASA NLU* dan *Neural Network* oleh (Jiao, A., 2020), membandingkan kinerja antara kerangka kerja *RASA NLU* dengan *Neural Network Classifier* yang dibangun dari awal oleh penulis atau biasa disebut dengan istilah *neural network model built from scratch*. Penelitian ini menjelaskan bahwa metode kerangka kerja RASA NLU masih lebih unggul untuk mengekstraksi semua entitas dan mengklasifikasikan *intent* atau maksud pengguna. Dalam penelitian ini juga dilakukan penelitian yang cukup komprehensif dengan menjabarkan secara detail proses analisa model, namun pada implementasi chatbotnya penelitian ini hanya bergantung pada *API* gratis dalam respon penyelesaian chatbot dan tidak menggunakan sistem penyelesaian respon yang dibuat sendiri seperti men-setup database atau lainnya. Rangkuman secara keseluruhan penelitian sebelumnya dikemas kedalam sebuah tabel yaitu Tabel 2.1.

Jadi, dibandingkan dengan semua pengembangan sebelumnya, penelitian ini akan lebih andal dikarenakan menggunakan *deep learning* untuk pemahaman bahasa alami dan menggabungkan *API* gratis dengan sistem database sendiri untuk penyelesaian respons dari chatbot. Jadi tidak perlu *rule based mapping* atau *pattern matching*, tidak menggunakan layanan berbasis *cloud provider*, dan tidak hanya mengandalkan layanan *API* gratis, sehingga kustomisasi chatbot akan lebih mudah, terukur dan tidak ada batasan *request* interaksi pengguna.

Tabel 2.1 *State of the Art penelitian*

Peneliti	**Bidang/Tema**	**Alat Analisis**
(Csaky, 2019)	*Deep Learning Chatbot*	Chatbot berbasis *Deep Learning* untuk melakukan respons balasan
(Prashant dkk., 2017)	*Chatbot* Universitas	*Pattern matching* untuk *information retrieval* pada chatbot
(Thakkar dkk., 2018)	*Chatbot* Universitas	API respons *Chatbot* berbayar, berbasis layanan cloud
(Al-fakhri, S. et al., 2019)	*Chatbot* Universitas	Sistem tanya jawab chatbot berbasis *Sentence Similarity Measurement (SSM)*, *spelling correction* dan *context word matching*
(Rana, 2019)	*Chatbot* Universitas	API respons *Chatbot* berbayar, berbasis layanan cloud Dialogflow untuk chatbot universitas
(Jiao, 2020)	*Deep Learning Chatbot*	Chatbot berbasis *Deep Learning* untuk melakukan respons balasan dan membandingkan performa framework Rasa
Penelitian yang akan dikembangkan	*Chatbot* Universitas *Deep Learning*	Chatbot FAQs Universitas Berbasis Deep Learning menggunakan framework Rasa

Pada Table 2.1 menjabarkan berbagai penelitian sebelumnya mengenai chatbot dan deep learning. Pengelompokkan tema penelitian Deep Learning dan Chatbot Universitas menjadi acuan dan penelusuran untuk melakukan penelitian ini. Sedangkan alat analisis merupakan intisari dari analisa pengembangan riset yang dilakukan pada setiap penelitian tersebut.

2.2 Konsep Pengetahuan

Pada bagian ini, dibahas beberapa konsep pengetahuan dan istilah mengenai rancangan dalam membangun chatbot. Konsep disini akan lebih ditekankan pada istilah-istilah yang digunakan oleh *framework* chatbot *RASA*. Penjabaran istilah dirangkum dan

didapat dari sumber dokumentasi resmi RASA, (2022c) dan *manuscript paper* dari tim pengembang utama *open-source RASA* (Bocklisch dkk., 2017).

2.2.1 Framework Chatbot

Framework, kerangka kerja atau sering disebut juga *platform* dirancang sedemikian rupa untuk membantu membangun chatbot (Lutkevich, 2022). *Framework* chatbot merupakan sekumpulan *tools* dan *library,* yang digunakan untuk membantu proses pengerjaan, pengembangan, serta kustomisasi chatbot yang sesuai dengan kebutuhan. Disebut juga *bootstrapping* yaitu teknik pembuatan aplikasi dengan memulai tahapannya dari sumber yang sudah ada, untuk menciptakan sesuatu yang lebih kompleks dan efisien dalam alur pengembangan chatbot.

Banyak pakar menyebut 2016 sebagai tahunnya chatbot. Di dunia *AI*, otomatisasi upaya manusia ke dalam komputer semakin banyak berkembang. Chatbot adalah aplikasi perangkat lunak untuk sistem percakapan. Ini menjadi sangat populer di dunia industri, digunakan untuk mengotomatisasi tugas-tugas yang tidak memerlukan bakat berbasis keterampilan. Relevansi ini dibuktikan dengan penyebaran besar-besaran chatbot. Memang saat ini chatbot digunakan untuk menyelesaikan sejumlah pekerjaan bisnis di banyak industri seperti *e-commerce*, asuransi, perbankan, kesehatan, keuangan, legal, telekomunikasi, ritel, dan masih banyak lagi lainnya. Gartner Summits memproyeksikan bahwa dari 85% interaksi pelanggan akan dikelola tanpa manusia pada tahun 2020. Chatbot diharapkan menjadi aplikasi konsumen nomor satu *AI* selama lima tahun ke depan menurut Tech Emergence (Arora dkk., 2020).

Beberapa *platform* yang cukup terkenal dalam konteks membangun chatbot seperti; IBM Watson, Microsoft Bot Framework, Dialogflow, RASA, LUIS, Wit.ai, dan Blender (ParlAI) yang baru-baru ini dikembangkan oleh Facebook AI Research (FAIR). Serta masih banyak lagi beberapa framework lainnya yang masih dalam penelitian. Logo *icon* dari beberapa *platform* chatbot yang terkenal ditampilkan oleh gambar 2.1.

Gambar 2.1 Framework Chatbot

2.2.2 Intent

Intent merupakan istilah yang sangat khas dalam konteks pengembangan chatbot, berdasarkan artian secara bahasa, *intent* memiliki arti maksud atau tujuan untuk melakukan sesuatu yang secara eksplisit ataupun implisit terdapat pada suatu kalimat (Dictionary, 2022). Adapun tata letak kedudukan *intent* pada chatbot yang didasari oleh alur percakapan dengan *user*. Alur dasar percakapan chatbot tersebut melibatkan tiga poin:

1. Menerima input permintaan dari *user*
2. Menerjemahkan input permintaan *user*, dan
3. Mengembalikan respons kepada *user*

Untuk setiap *intent* atau maksud, direpresentasikan sebagai ucapan *user* atau istilahnya *user utterance*, apa saja kata yang harus difahami dan diekstrak oleh sistem chatbot dari ucapan tersebut, lalu bagaimana menanggapinya. Seperti contoh “saya ingin pizza”, “saya pesan pizza”, atau “saya butuh pizza” semuanya merujuk arti bahwa mereka menunjukkan maksud tujuan yang sama yaitu “maksud untuk memesan pizza” (RASA, 2022e).

2.2.3 Entity

Entity atau entitas secara arti bahasa merupakan suatu nomina satuan yang berwujud, sedangkan dalam istilah pada framework chatbot adalah suatu kata kunci yang dapat diekstraksi dari pesan *user* (Setiawan, 2022). Misalnya: nomor telepon, nama orang, lokasi, atau nama produk (RASA, 2022f). Pada mekanisme *RASA* proses itu digunakan untuk mengidentifikasi dan mengekstraksi data yang berguna sebagai informasi lanjutan dari input pesan *user*. Meskipun *intent* memungkinkan agen chatbot untuk memahami apapun motivasi dibalik input *user* tertentu, *entity* digunakan untuk memilih informasi tertentu yang telah

disebutkan oleh *user*. Misalnya, jika terdapat input frasa dari *user* seperti "Tolong saya, ingin memesan 12 pizza" *RASA* mencocokkan "12" sebagai *entity* kuantitas pizza.

2.2.4 Konteks

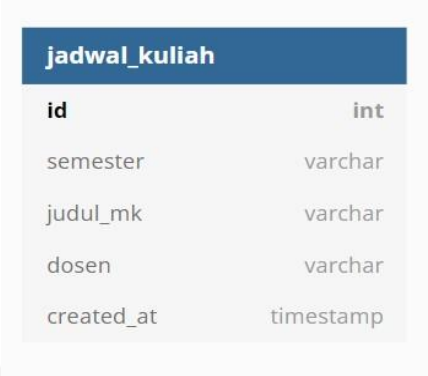
Konteks disebut juga *tracker* yang merupakan komponen *Rasa Open Source* yang mengatur dan mempertahankan status dialog chatbot dan *user*, yang direpresentasikan sebagai objek yang mencantumkan peristiwa dari setiap sesi (RASA, 2022g). Bersamaan dengan konteks dan *tracker* adapula istilah *slot,* merupakan objek yang digunakan *Rasa* untuk melacak informasi selama percakapan (RASA, 2022h). Konteks mewakili kondisi atau keadaan terbaru dari percakapan input *user* dan memungkinkan agen chatbot untuk membawa informasi dari satu maksud *intent* ke maksud lainnya. Dalam contoh percakapan pizza, jika pengguna bertanya: "Pesankan saya pizza", chatbot perlu mengetahui detail lebih lanjut tentang pesanan seperti: kuantitas dan ukuran pizza, topping, saus tambahan dan beberapa spesifikasi lainnya. Untuk mengumpulkan semua informasi ini pada percakapan *multi-route* dan tetap pada fase yang sama kita membutuhkan konteks dan ini diatur pada modul *tracker*. *Tracker* menjaga keadaan dialog antara chatbot dan *user* dalam bentuk sesi percakapan.

Seluruh proses pengambilan jawaban dari *user* akan diklarifikasikan seperti contoh dibawah ini:

1. Pertanyaan: Apakah Dr. X mengajar Algoritma Pemrograman ?
2. *Intent*: Pencarian terkait "kursus atau mata kuliah" dan "pengajar"
3. *Entity*: "nama pengajar": "Dr. X", "mata kuliah": "Algoritma Pemrograman"

Misal, dari pertanyaan diatas, sistem kami memetakan pertanyaan tersebut dengan jenis *intent* pencarian terkait pengajar dan mata kuliah lalu mengekstrak dua nama *entity* pengajar dan mata kuliah. Kemudian, kode pemrograman melanjutkan informasi pertanyaan ini menjadi kueri penelusuran ke *database*, dan yang digunakan secara spesifik adalah jenis *database MySQL*. Penelusuran tersebut bertujuan untuk mengekstrak hasil yang diinginkan dari *database,* tempat menyimpan banyak data tabular yang sering dicari dan dibutuhkan. Seperti yang dijelaskan melalui contoh struktur *table database jadwal_kuliah* di gambar 2.2, akan dilakukan kueri pada *table* tersebut, dengan *syntax* kueri sebagai berikut;

a. select judul_mk, dosen from jadwal_kuliah where judul_mk like "% mata_kuliah %" and dosen like "% nama_pengajar %" (query pada *MySQL*)



Gambar 2.2 *Table SQL jadwal_kuliah*

Pada akhirnya dari hasil kueri yang dikembalikan, memberikan pesan respons yang lengkap dengan *completion* untuk menunjukkan hasilnya kepada *user*, dalam konteks bahasan ini yaitu chatbot mengafirmasi benar bahwa Dr. X mengajar Algoritma Pemrograman saat *return* dari kueri memberikan hasil, dan chatbot akan mengafirmasi salah bahwa Dr. X tidak mengajar Algoritma Pemrograman saat *return* dari kueri itu kosong.

2.2.5 Policy

Policy merupakan komponen *Rasa Open-Source* yang memprediksi tindakan selanjutnya dari sistem dialog, membuat kebijakan atau keputusan tentang bagaimana alur percakapan harus dilanjutkan. Konfigurasi tipikal mencakup beberapa kebijakan, dan kebijakan dengan nilai *confidence* tertinggi akan memutuskan tindakan selanjutnya yang diambil dalam percakapan (RASA, 2022i). *Policy* atau kebijakan memiliki tugas untuk memilih tindakan selanjutnya yang akan dilakukan, melalui objek *tracker*. *Policy* digunakan bersama *featurizer*, yang membuat representasi vektor dari *state* dialog terkini yang didapat dari objek *tracker*. *Standard featurizer* melakukan proses *concatenate* yang mewakili :

a. tindakan *action* terakhir
b. *intent* dan *entity* dalam pesan pengguna terbaru
c. *slot* mana yang saat ini ditentukan oleh *Rasa*

Featurizer dari *slot* dapat bervariasi. Dalam kasus paling sederhana, sebuah slot diwakili oleh biner tunggal elemen vektor yang menunjukkan apakah itu diisi. *Slot* yang merupakan variabel kategori dikodekan sebagai sebuah vektor biner satu-dari-k, yang mengambil nilai kontinu dapat ditentukan dengan ambang batas tertentu yang mempengaruhi fitur, atau hanya diteruskan ke fitur tersebut sebagai sebuah nilai *float*. Terdapat parameter *max_history* yang menentukan jumlah *state* sebelumnya untuk dimasukan dalam fitur tersebut. Secara

default, keadaan *state* ditumpuk untuk membentuk *array* dua dimensi, yang bisa diproses oleh jaringan saraf berulang *RNN* atau model sekuensial sejenisnya. Secara praktikal para peneliti dari *RASA team* menemukan bahwa nilai *max_history* antara 3 dan 6 sudah bekerja dengan baik (RASA, 2022d).

2.2.6 Action

Action merupakan suatu langkah, tindakan yang dilakukan chatbot dalam percakapan (mis. memanggil *API* atau mengirim respons kembali ke *user*). *Action* akan dijalankan oleh service terpisah dari main server *Rasa* chatbot nantinya. *Action* dijalankan oleh *action server* yang menjalankan kode atau logic tindakan yang dibuat secara kustom, terpisah dari *Rasa Open Source* (RASA, 2022j). Action termasuk ke dalam sistem manajemen dialog. Di setiap iterasi percakapan, *Rasa Core* atau manajemen dialog *Rasa* memprediksi *action* mana yang akan diambil dari daftar yang telah ditentukan. Suatu *action* dapat berupa ucapan sederhana, misal :

a. mengirim pesan ke pengguna, atau
b. merupakan fungsi logic kustom keputusan tertentu untuk dieksekusi

Ketika suatu *action* dieksekusi chatbot, sebelumnya akan melalui modul *tracker* terlebih dahulu, sehingga dari sanalah terkumpul informasi yang relevan atas sejarah dialog. Informasi tersebut dapat berupa; *slot*, ucapan sebelumnya, dan hasil *action* sebelumnya. Tindakan atau *action* tidak dapat secara langsung merubah pelacak *tracker*, tetapi merubah ketika dieksekusi melalui pengembalian daftar *event*. *Tracker* mengkonsumsi *event* ini untuk memperbaharui kondisinya atau disebut juga *state*. Ada sejumlah jenis *event* yang berbeda seperti; *SlotSet*, *AllSlotsReset*, *Restart*, dll (RASA, 2022a).

2.2.7 Web Platform Channel Connector

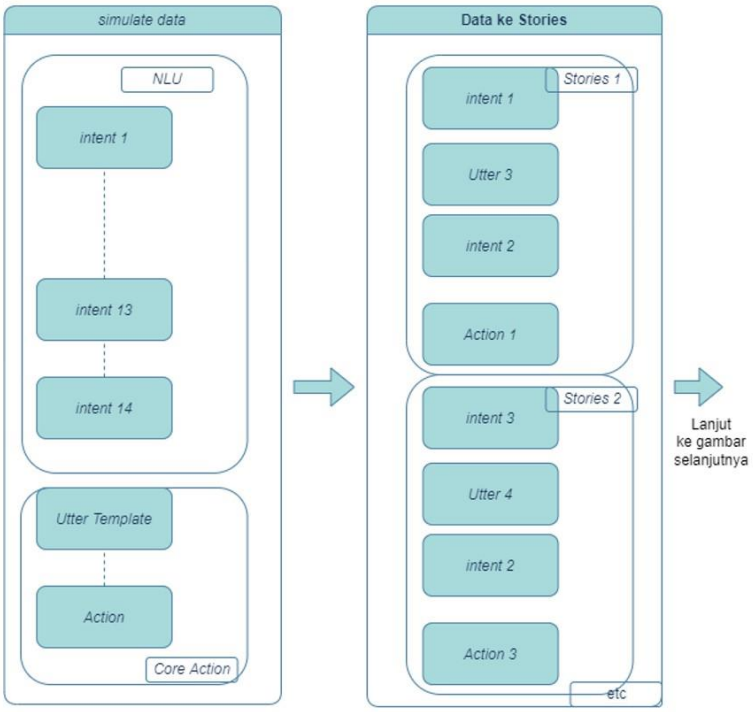
Web Platform adalah kumpulan teknologi yang dikembangkan sebagai standar terbuka oleh *World Wide Web Consortium* dan badan standardisasi lainnya seperti Kelompok Kerja *Teknologi Aplikasi Hypertext Web*, Konsorsium *Unicode*, *Internet Engineering Task Force*, dan *Ecma International*. Ini adalah istilah umum yang diperkenalkan oleh *World Wide Web Consortium*, dan pada tahun 2011 didefinisikan sebagai "platform untuk inovasi, konsolidasi, dan efisiensi biaya" oleh *CEO W3C* Jeff Jaffe. Selain itu, pengembangannya dalam membangun konten yang dapat dioperasikan pada platform yang kohesif. Platform web mencakup teknologi bahasa komputer dan *API* yang awalnya dibuat terkait dengan

publikasi halaman web. Ini termasuk *HTML*, *CSS*, *SVG*, *MathML*, *WAI-ARIA*, *ECMAScript*, *WebGL*, penyimpanan web, *API Database* terindeks, komponen web, *WebAssembly*, *WebGPU*, *DOM*, *XMLHttpRequest*, *HTTP*, *TLS 1.2*, dan *IRI* (wikipedia, 2022). *Channel Connector* sendiri adalah media yang akan digunakan untuk penyajian chatbot. Halaman web merupakan salah satu dari *Channel Connector* pada *RASA* yang sangat *customizable* untuk implementasinya, yaitu hanya dengan cara menambahkan *widget* chatbot ke dalam *HTML* halaman web (RASA, 2022k).

BAB 3
Metodologi

3.1 Alur Rancang Bangun Mi-botway

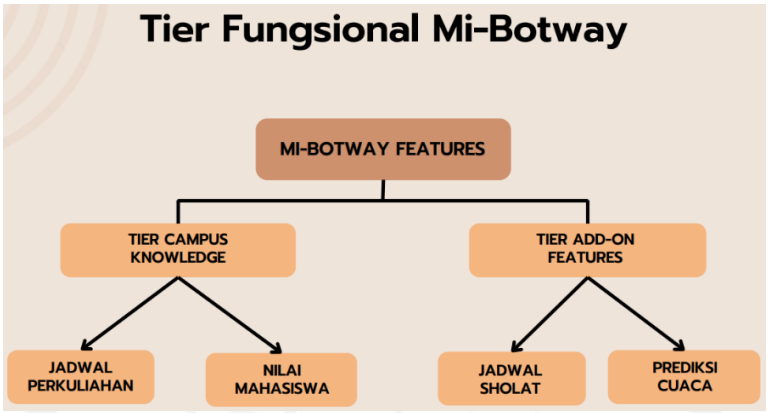
Mi-botway dirancang dan dibangun berdasarkan alur yang dibagi kedalam 3 flow gambar, yang pertama diilustrasikan oleh Gambar 3.1.



Gambar 3.1 Alur rancang bangun Mi-botway bagian-1

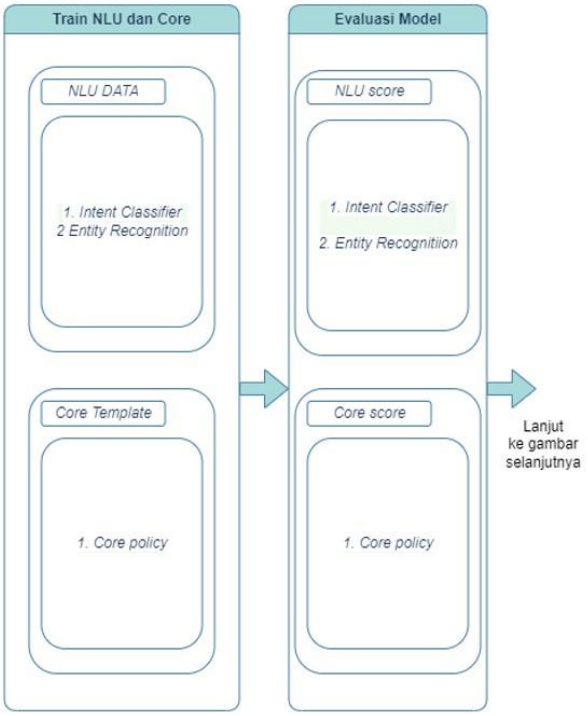
Alur pada Gambar 3.1 dibagi kedalam proses *simulate* atau *generate* data serta proses pengelompokan data menjadi *stories.* Proses ini akan dijelaskan secara mendetail pada subbab 3.2. Lalu alur yang kedua diilustrasikan oleh Gambar 3.3.

Namun pada pengembangan mi-botway disini perlu digarisbawahi bahwasanya, fitur mi-botway dibagi menjadi 2 kelompok besar yaitu fitur fungsional dan fitur tambahan diilustrasikan oleh Gambar 3.2.



Gambar 3.2 Tier fungsional Mi-botway

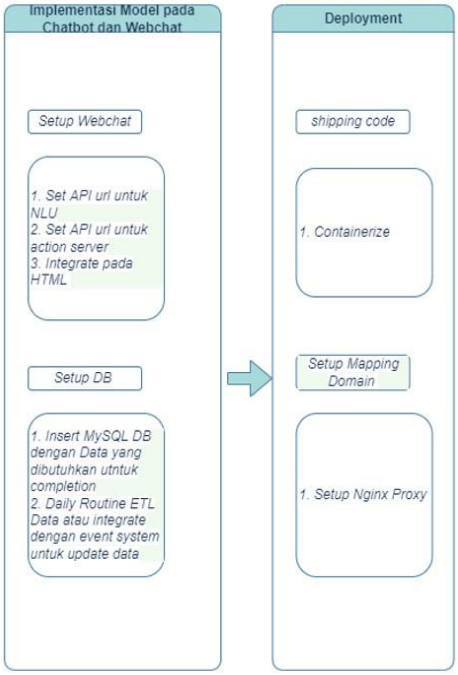
Kedua tier ini membagi menjadi fitur utama yaitu “Jadwal Perkuliahan” dan “Nilai Mahasiswa” dan fitur tambahan yaitu “Jadwal Sholat” dan “Prediksi Cuaca”. Maka penjelasan pada proses pengembangan chatbot pun lebih ditekankan hanya untuk fungsi pada fitur utama. Sedangkan untuk fitur tambahan tidak menjadi fokus bahasan penelitian dan hanya untuk menambah kebermanfaatan chatbot saja.



Gambar 3.3 Alur rancang bangun Mi-botway bagian-2

Alur pada Gambar 3.3 dibagi ke dalam tahapan *training* model *NLU* dan *Core* chatbot serta tahapan evaluasi score model agar didapat model terbaik untuk implementasi pada platform

chatbot. Proses ini akan dijelaskan juga secara mendetail pada subbab 3.3. Terakhir Alur ketiga diilustrasikan oleh Gambar 3.4.



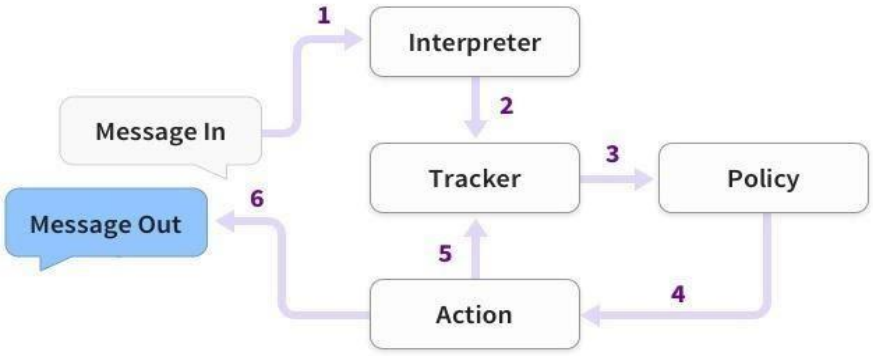
Gambar 3.4 Alur rancang bangun Mi-botway bagian-3

Alur terakhir pada Gambar 3.4 merupakan proses pengimplementasian chatbot pada web platform serta proses *deployment* chatbot pada server. Proses ini akan dijelaskan juga secara mendetail pada subbab 3.4 dan subbab 3.5.

Pada tahap ini semua alur tahapan dan metode proses penelitian dilakukan. Dimulai dengan simulasi data berdasarkan interaksi skema dialog pertanyaan mahasiswa, dilanjutkan dengan mengelompokkan setiap kalimat ke dalam topik *intent* tertentu, *utter template* chatbot RASA, (2022k), dan *action*. Khususnya pada topik *intent* yang telah diusulkan sebagai fitur chatbot, diperlukan juga untuk mendefinisikan *entity* pada topik intent tersebut. Perbedaan antara *utter* dan *action* yang utama adalah, pada *utter* hanya terdapat sebuah kalimat dengan ekspresi dari respon umum, sedangkan *action* digunakan ketika respon membutuhkan informasi data dari proses *retrieval database* atau *API*, *action* juga perlu dibungkus sebagai class action di RASA pada saat implementasinya.

Alur rancang bangun chatbot ini semuanya bersumber pada aturan *framework Rasa. Framework* chatbot sendiri merupakan suatu kerangka kerja yang digunakan untuk melakukan *bootstrap* pada semua modul chatbot. Khusus untuk implementasi *machine learning*, di bawah ini adalah arsitektur atau alur dari *framework RASA* yang digunakan. Arsitektur atau alur manajemen dialog terlihat pada Gambar 3.5 yang merupakan arsitektur

dari *framework RASA stack*. Pada tahap pertama skema pesan diterima dari user dan diteruskan ke interpreter yaitu *RASA NLU* untuk mengekstrak *intent*, *entity*, dan informasi terstruktur lainnya. Baik interpreter maupun *tracker* akan melakukan proses pelacakan, mendeteksi, dan menjaga status konteks percakapan melalui notifikasi pesan yang diterimanya. *Policy* menerima status konteks dari *tracker*. Lalu *policy* membuat kebijakan dalam memilih tindakan mana yang akan diambil selanjutnya. *Action* atau tindakan direkam oleh *Tracker*. Hingga akhirnya tindakan atau *action* ini dijalankan dengan mengirimkan pesan ke *user*. Jika tindakan atau *action* yang telah dieksekusi diabaikan oleh *user* pada waktu tertentu, proses kembali ke langkah pelacakan oleh *tracker* hingga penentuan keputusan oleh *policy*.



Gambar 3.5 *Rasa framework stack* (Bocklisch dkk., 2017).

3.2 Data

Berbagai data sebelumnya dilihat dan dipertimbangkan dari bermacam skema referensi umum atau skenario percakapan. Data percakapan untuk setiap fitur chatbot, seperti alur komunikasi pertanyaan mahasiswa mengenai jadwal kuliah, dan nilai mahasiswa. Kemudian ditambah dengan beberapa *intent* dialog umum seperti salam dan sebagainya. Sehingga, *intent*, *utter*, dan *action* diatur sedemikian rupa agar sesuai dengan *story* dan *entity* dalam setiap kalimat percakapan.

3.2.1 Simulate Data

Proses *generate* atau *simulate* data dimulai dengan memahami pola percakapan umum yang biasa dilakukan dalam konteks fitur-fitur utama mi-botway. Setelah memahami pola percakapannya selanjutnya disusun dan dibuat data percakapan tersebut berdasarkan format

data yang diperlukan untuk *NLU* data, *utter template,* dan *action* apa saja yang diperlukan. Beberapa sampel data *NLU* yang telah di-*generate* untuk konteks chatbot mi-botway terlihat pada Gambar 3.6.

```
...
...
 30   ## intent:intent_minta_daftar_nilai
 31   - mau tau nilai saya
 32   - nilai saya dong
 33   - nilai saya
 34   - cek nilai
 35   - nilai [19917003](NIM)
 36   - nilai [19917020](NIM)
 37   - nilai [19917012](NIM)
 38   - nilai [fahmi nurrahim](nama)
 39   - nilai [yurio windiatmoko](nama)
 40   - nilai [prastyo eko susanto](nama)
 41   - nilai [malik abdul aziz](nama)
 42
 43   ## intent:intent_confirm_nama_atau_nim
 44   - [19917003](NIM)
 45   - [19917020](NIM)
 46   - [19917012](NIM)
….
 59   - Tolong jadwalnya untuk [Data Sains](konsentrasi)
 60   - Jadwal [Data Sains](konsentrasi)
 61   - Minta jadwal dong untuk [Data Sains](konsentrasi)
....
692   - [yahukimo](kota)
693   - [yalimo](kota)
694   - [yapen waropen](kota)
695
696   ## intent:intent_setuju
697   - oke
698   - oks
699   - Iya
700   - Benar
701   - Tentunya
702   - Sepertinya oke
703
704   ## intent:intent_menolak
705   - tidak
706   - tidak pernah
707   - Aku rasa tidak
...
```

Gambar 3.6 Sampel potongan data NLU mi-botway (Windiatmoko, 2022e)

Dari Gambar 3.6 terdapat berbagai sampel simulasi data *intent* misalnya, untuk baris 30-41 merupakan kelompok data kalimat permintaan untuk menanyakan daftar nilai, hingga baris seterusnya yang merupakan kalimat dengan kelompok intent lainnya. Data *NLU* ini pada akhirnya akan di-*training* untuk menjadi model klasifikasi *intent* dan *entity extractor*.

Berbagai data sebelumnya dipertimbangkan dari bermacam skema skenario percakapan sesuai dengan fitur utama chatbot. Selain itu proses *simulate* melibatkan pendapat orang-orang terutama rekan satu kelas dengan mengekspresikannya langsung pada sebuah kalimat percakapan yang dikumpulkan sebuah pada *shared-spreadsheet*.

Total data *intent* yang berhasil terkumpul adalah 693 kalimat dengan 16 macam *intent* yaitu 'mood tidak senang', 'intent minta daftar nilai', 'menantang bot', 'intent prediksi cuaca', 'intent minta jadwal', 'intent salam', 'intent confirm kota', 'intent berpisah', 'intent minta jadwal saja', 'intent menyapa', 'terima kasih', 'intent confirm nama atau nim', 'mood baik', 'intent menolak', 'intent jadwal sholat', 'intent setuju'. Adapun total data *entity* yang berhasil terkumpul adalah 645 kata dengan 4 macam *entity* yaitu 'NIM', 'kota', 'nama', 'konsentrasi'

3.2.2 Data ke Stories

Proses dilanjutkan dengan mengelompokkan setiap data *intent*, *utter template*, dan *action* yang akan digunakan chatbot ke file bernama *domain* (RASA, 2022b). Domain mendefinisikan data semesta chatbot. Domain ini menspesifikasikan *intent*, *entity*, *slot*, *utter template*, dan *action* yang harus diketahui chatbot. Domain juga mendefinisikan konfigurasi untuk sesi percakapan. Berikut merupakan file domain yang digunakan pada chatbot mi-botway terlihat pada Gambar 3.7.

```
 1    intents:
 2     - intent_berpisah
 3     - intent_confirm_kota
 4     - intent_confirm_nama_atau_nim
 5     - intent_jadwal_sholat
 6     - intent_menolak
 7     - intent_menyapa
 8     - intent_minta_daftar_nilai
 9     - intent_minta_jadwal
10     - intent_minta_jadwal_saja
11     - intent_prediksi_cuaca
12     - intent_salam
13     - intent_setuju
14     - menantang_bot
15     - mood_baik
16     - mood_tidak_senang
17     - terima_kasih
18    entities:
19     - NIM
20     - konsentrasi
21     - kota
22     - nama
23    slots:
24      NIM:
25        type: text
26      konsentrasi:
27        type: text
….
86     - utter_tanya_nama_atau_nim
87     - utter_terima_kasih
```

Gambar 3.7 File *domain* mi-botway (Windiatmoko, 2022f)

Representasi *stories* merupakan bentuk percakapan antara *user* dan chatbot, ini disusun berdasarkan data *NLU* dan data pengelompokkan *domain* sebelumnya. Diubah menjadi format tertentu di mana input *user* dinyatakan sebagai *intent* (dan *entity* bila

diperlukan), sedangkan respons *utter* dan *action* chatbot dinyatakan sebagai nama *action*. *Stories* adalah jenis data training yang digunakan untuk melatih model manajemen dialog chatbot. *Stories* dapat digunakan untuk melatih model yang mampu menggeneralisasi ke jalur percakapan yang tidak terlihat. Berikut merupakan data *stories* yang digunakan pada chatbot mi-botway terlihat pada Gambar 3.8.

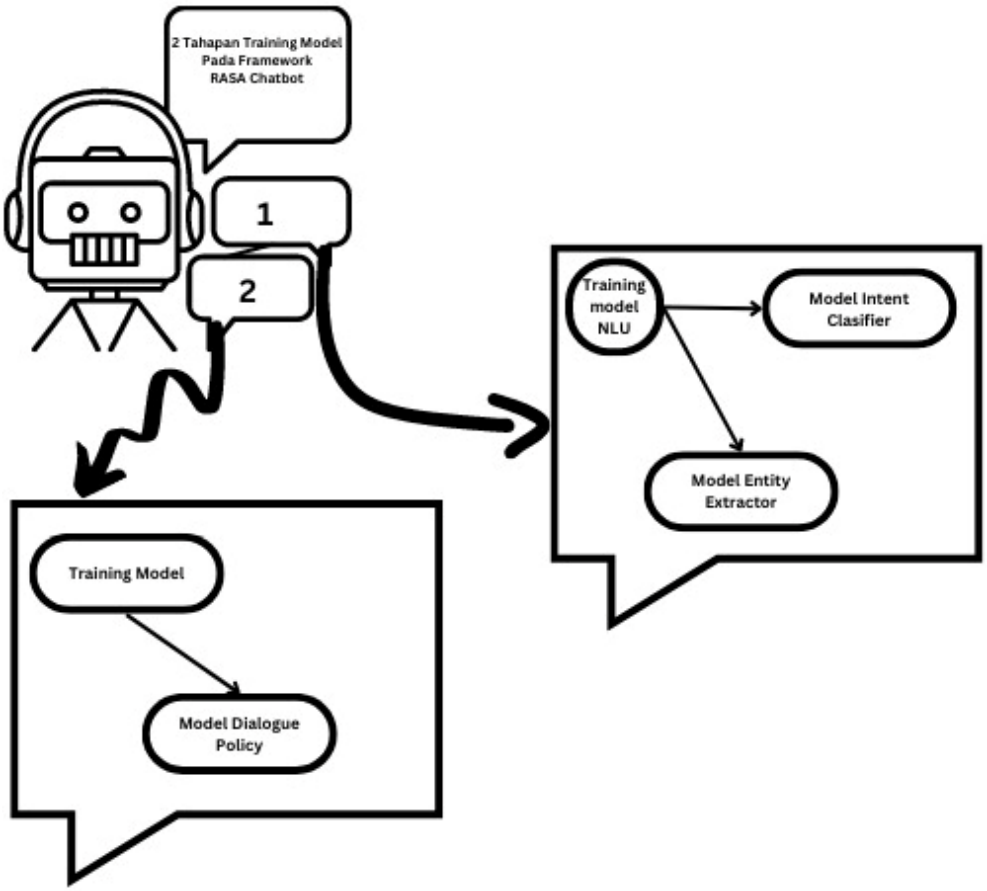
```
….
 48
 49   ## story minta jadwal
 50   * intent_menyapa
 51     - utter_menyapa
 52   * intent_minta_jadwal
 53     - action_daftar_jadwal
 54   * terima_kasih
 55     - utter_terima_kasih
 56
 57   ## story minta jadwal 1
 58   * intent_salam
 59     - utter_membalas_salam
 60   * intent_minta_jadwal
 61     - action_daftar_jadwal
 62   * terima_kasih
 63     - utter_terima_kasih
 64
 65   ## story minta jadwal 2
 66   * intent_menyapa
 67     - utter_menyapa
 68   * intent_minta_jadwal_saja
 69     - utter_tanya_konsentrasi
 70   * intent_minta_jadwal
 71     - action_daftar_jadwal
 72   * terima_kasih
 73     - utter_terima_kasih
 74
 75   ## story minta jadwal 3
 76   * intent_salam
 77     - utter_membalas_salam
 78   * intent_minta_jadwal_saja
 79     - utter_tanya_konsentrasi
 80   * intent_minta_jadwal
 81     - action_daftar_jadwal
 82   * terima_kasih
 …..
105     - action_pred_cuaca
```

Gambar 3.8 Data *stories* mi-botway (Windiatmoko, 2022g)

Dari Gambar 3.8 terdapat berbagai sampel kelompok skenario *stories*. Data *stories* ini pada akhirnya akan di-*training* untuk menjadi model *dialogue policy* yang digunakan chatbot untuk dapat merespons dengan tepat.

3.3 Modeling

Pada tahap modeling, *RASA framework* membaginya ke dalam dua bagian training ditujukan oleh Gambar3.9.

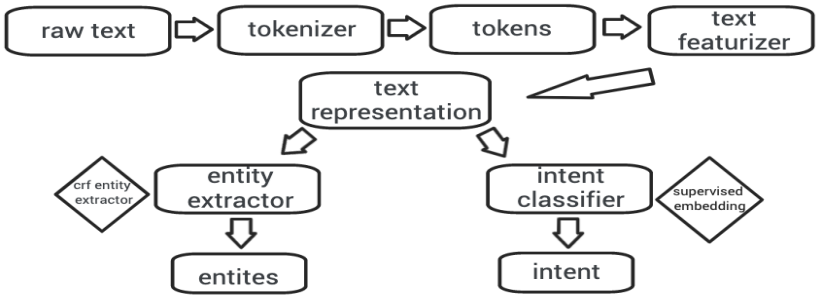


Gambar 3.9 Alur Training Model pada *RASA* Framework

Pada Gambar 3.9 ditunjukkan 2 tahapan tersebut. Pertama *Training model NLU* yang akan menghasilkan model *intent classification* dan model *entity extractor* dengan menggunakan data utamanya yaitu data NLU. Lalu yang kedua *Train model CORE* yang akan menghasilkan model *Dialogue Policy* dengan menggunakan data utamanya berupa data *stories*.

3.3.1 Training NLU dan Core

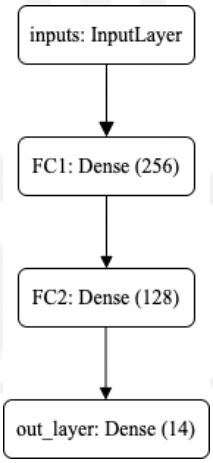
Tahap pertama dalam melakukan proses *training* adalah *training NLU* data. *NLU* atau interpreter terdiri dari dua komponen yaitu *Intent Classifier* dan *Entity Extractor*. Proses yang terjadi selama pelatihan dimulai dari ekstrak kalimat oleh tokenizer menjadi token, kemudian token diubah oleh text featurizer menjadi representasi vektor untuk dilatih menjadi model entity extractor oleh CRF entity extractor dan model Intent classifier dengan supervised embedding. Gambar 3.10 menunjukkan aliran proses training *intent* dan entity.



Gambar 3.10 Pre-prosesing Teks dan Modeling Klasifikasi Intent Entitas

Secara khusus, chatbot ini menggunakan *supervised embedded intent classifier*, yang juga dikenal sebagai *embedded intent classifier*. *Intent Classifier* meng-*embed* input user, dan memberi label *intent* ke dalam ruang dimensi yang sama. *Training supervised embedding* dilakukan dengan memaksimalkan kesamaan antara kedua label dan input yang sudah memiliki nilai *embedding*. Algoritma ini didasarkan pada StarSpace (Wu dkk., 2018). Namun, dalam implementasi ini, fungsi *loss* sedikit berbeda, dan *hidden layer* juga ditambahkan bersama dengan *dropout* untuk membantu proses *training*. Algoritma ini juga memberikan peringkat kesamaan untuk label.

Konfigurasi pelatihan untuk membuat model klasifikasi *intent* ini diset ke 300 untuk nilai *epoch*, kemudian *hidden layer* di *feed forward neural network* adalah 256 di h1 (hidden layer pertama), 128 di h2 (hidden layer kedua), dan 14 di kelas label *intent* pada *output layer neural network* seperti yang terllihat pada Gambar 3.11.



Gambar 3.11 Model NLU untuk klasifikasi *Intent*

Dimensi *embedding* yang digunakan untuk pelatihan adalah 20, ini adalah beberapa lapisan *embedding* dalam arsitektur model. Misalnya, vektor token '__CLS__' dan *intent* diteruskan ke lapisan embedding sebelum dibandingkan dan nilai *loss* dihitung. Konfigurasi untuk ukuran *batch* adalah (64, 256) yang merupakan nilai awal dan akhir untuk ukuran batch.

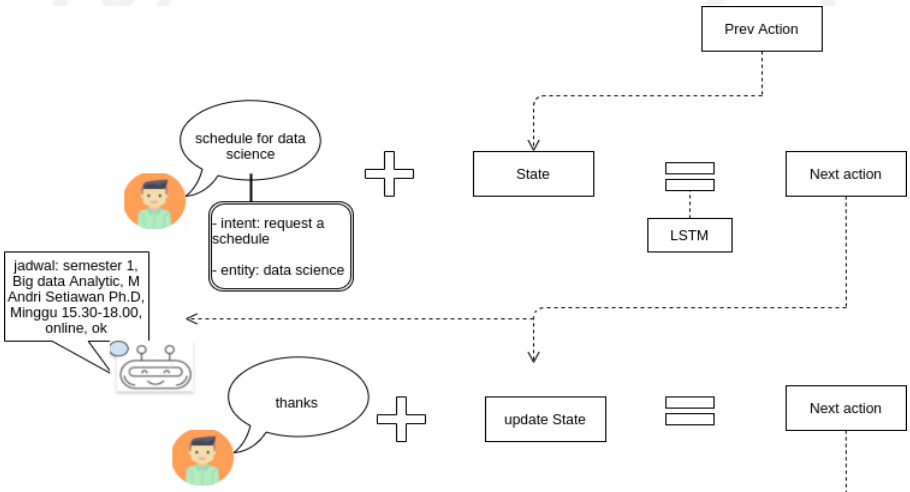
Ukuran batch akan meningkat secara linier untuk setiap *epoch*. *Learning rate* diatur dengan nilai 0,001 untuk konfigurasi *training*.

Entity Extractor yang digunakan untuk chatbot ini adalah *CRF entity extractor* (Lafferty, J. and Mccallum, A., 2001) yang merupakan metode yang cocok untuk *custom entity extractor*. Komponennya mengimplementasikan bidang acak bersyarat (*CRF*) untuk melakukan pengenalan *entity*. *CRF* dapat dianggap sebagai rantai markov tidak terarah di mana langkah waktu adalah kata-kata dan statusnya adalah kelas *entity*. Fitur berasal dari kata-kata seperti kapitalisasi, penandaan *POS*, dll. Fitur tersebut memberikan kemungkinan untuk kelas *entity* tertentu, seperti transisi antara *entity tag* dan sekumpulan *tag* yang kemungkinan besar akan dihitung dan dikembalikan. Pelatihan konfigurasi untuk *CRF entity extractor* memiliki daftar fitur untuk digunakan. Namun, itu dapat diatur dan dipilih dalam kondisi apa pun. Secara khusus, chatbot ini menggunakan proses deteksi daftar fitur *token* seperti:

a) "Huruf kecil" untuk memeriksa apakah token adalah huruf kecil.
b) "Huruf Besar" untuk memeriksa apakah token adalah huruf besar.
c) "Judul" untuk memeriksa apakah token dimulai dengan karakter huruf besar dan semua karakter yang tersisa adalah huruf kecil.
d) "Digit" untuk memeriksa apakah token hanya berisi angka.
e) "Awalan5" untuk mengambil lima karakter pertama dari token.
f) "Awalan2" untuk mengambil dua karakter pertama dari token.
g) "Suffix5" untuk mengambil lima token karakter terakhir.
h) "Suffix3" untuk mengambil tiga karakter terakhir dari token.
i) "Suffix2" untuk mengambil dua karakter terakhir dari token.
j) "Suffix1" untuk mengambil karakter terakhir dari token.
k) "Bias" untuk menambahkan fitur bias tambahan ke daftar fitur.

Ketika detektor bergerak pada *token* dalam pesan user dengan proses jendela geser, fitur ini mendefinisikan fitur untuk sebelumnya, *token* saat ini, dan token berikutnya dalam proses jendela geser. Jadi deskripsi fiturnya seperti *[before, token, after]* yang dianggap sebagai *array token*.

Tahap selanjutnya dalam melakukan proses *training* adalah *training dialogue policy*, proses *training* dilakukan dengan training menggunakan *stories* data. *Stories* tampak seperti beberapa skenario atau skema topik umum dalam percakapan apa pun sesuai chatbot yang dirancang atau diharapkan. Respons *chat* bekerja dengan *input dan state embedding* untuk selanjutnya model memprediksi tindakan selanjutnya seperti yang dijelaskan pada Gambar 3.12. *Dialogue policy* memutuskan tindakan mana yang harus diambil pada setiap langkah percakapan. Ada berbagai kebijakan untuk dipilih, dan ini termasuk beberapa kebijakan dalam satu agen atau chatbot. Pada setiap tahap, kebijakan yang memprediksi tindakan selanjutnya dengan keyakinan probabilitas tertinggi akan digunakan.



Gambar 3.12 Alur komponen manajemen dialog

Chatbot ini dikonfigurasi dengan kebijakan memori dan kebijakan *neural network*. Implementasi kebijakan akan didasarkan pada prioritas; kebijakan prioritas tertinggi akan dilaksanakan terlebih dahulu dan seterusnya. Prioritas dihitung berdasarkan nilai probabilitas keyakinan setiap kebijakan, yang memiliki skor lebih tinggi diberikan prioritas yang lebih tinggi. Kebijakan menghafal hanya mengingat percakapan dalam data pelatihan. Ini memprediksi tindakan selanjutnya dengan keyakinan 1,0 jika percakapan yang tepat ini ada dalam data pelatihan, jika tidak, ia memprediksi tidak ada dengan keyakinan 0,0. Kebijakan jaringan saraf kemudian diimplementasikan untuk memilih respons tindakan selanjutnya. Arsitektur default didasarkan pada LSTM (Hochreiter & Schmidhuber, 1997). Ini dikonfigurasi dengan *input layer* (input, 5, 32), unit *layer LSTM* (32), *dropout* (0.2), *output layer* (19), kemudian 100 *training epoch*. Konfigurasi awal ini didasari dengan hasil terbaik model chatbot pada *framework* RASA yang sebelumnya diteliti oleh (Jiao, A., 2020).

Melalui penelitian ini model *dialogue policy* akan dibandingkan dengan berbagai jenis arsitektur model *RNN* sesuai dengan skenario pada Tabel 3.1.

Tabel 3.1 Skenario pengukuran model RNN

model	dimensi filter
SimpleRNN	32
SimpleRNN	64
SimpleRNN	128
LSTM	32
LSTM	64
LSTM	128
GRU	32
GRU	64
GRU	128

3.3.2 Evaluasi Model

Penelitian pengembangan chatbot menggunakan model *deep learning* diharapkan dapat memberikan kinerja terbaik bagi sistem chatbot melalui model *deep learning* tersebut. Selanjutnya representasi hasil kinerja evaluasi model *deep learning* pada chatbot yaitu *intent classifier* dan *entity extractor* pada *NLU* serta *dialogue policy* pada system dialog management chatbot, diukur dan dievaluasi dengan nilai presisi, skor *recall*, dan skor *F1* (Goutte & Gaussier, 2005). Namun kemudian karena poin utama dari penelitian ini lebih menekankan pada model deep learning yang diimplementasikan pada *Dialogue Policy*, maka pada penelitian ini dilakukan perbandingan performansi antara arsitektur *RNN* yaitu Standard *RNN*, *LSTM*, dan *GRU*.

Confusion Matrix merupakan pengukuran kinerja untuk masalah klasifikasi machine learning dimana outputnya dapat berupa dua kelas atau lebih (Goutte & Gaussier, 2005). Biasanya *Confusion Matrix* adalah tabel dengan 4 kombinasi nilai prediksi dan nilai aktual yang berbeda. Ada empat istilah yang mewakili hasil proses klasifikasi dalam *confusion matrix*, yaitu *True Positive, True Negative, False Positive,* dan *False Negative*. *Confusion Matrix* untuk klasifikasi multi-kelas ini sedikit berbeda, anggap saja masalah klasifikasi multi-kelas menjadi masalah klasifikasi multi-kelas. Tidak seperti klasifikasi biner, tidak ada kelas positif atau negatif di sini. Yang harus kita lakukan di sini adalah mencari *TP, TN, FP* dan *FN* untuk masing-masing kelas individu. Disini hanya akan menampilkan data yang memiliki nilai *misclassification* karena tabel *confusion matrix* secara keseluruhan akan kurang efektif dan memakan *space* halaman penelitian.

Precision terlihat pada persamaan (3.1) dapat didefinisikan sebagai probabilitas bahwa suatu objek relevan karena dikembalikan oleh sistem sebagai model yang diprediksi, sedangkan *recall* terlihat pada persamaan (3.2) adalah probabilitas bahwa objek yang relevan dikembalikan berdasarkan kebenaran dasar dari setiap label data. Kemudian, skor *F1* terlihat pada persamaan (3.3) diperlukan untuk menemukan keseimbangan antara presisi dan daya ingat.

$$Precision = \frac{True\ Class}{True\ Class + False\ Predicted\ Class} = \frac{True\ Class}{Total\ Predicted\ Class} \quad (3.1)$$

$$Recall = \frac{True\ Class}{True\ Class + False\ Predicted\ Other\ Class} = \frac{True\ Class}{Total\ Actual\ Class} \quad (3.2)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.3)$$

Singkatnya, presisi itu menunjukkan seberapa tepat atau akurat model di luar kelas label yang diprediksi, berapa banyak di antaranya yang merupakan kelas label sebenarnya. Sementara itu, *recall* menghitung bagaimana model menangkap banyak kelas sebenarnya dengan melabelinya sebagai kelas itu. Akhirnya, skor *F1* digunakan untuk menemukan keseimbangan antara presisi dan daya ingat.

3.4 Implementasi Rancang Bangun Chatbot

Pembahasan implementasi rancang bangun chatbot ini terdiri dari penjelasan tentang konektor modul platform untuk chatbot yang merupakan alur komunikasi *webhook protocol http* antara *chatbot service* dengan halaman *web html*. Dilanjut dengan penjelasan mengenai proses perancangan halaman chatbot pada *web*, dan perancangan *table database* yang digunakan pada service chatbot.

3.4.1 Konektor Modul Platform

Pengaturan platform didasari oleh komunikasi antar pesan *webhook* dan *postback*. Untuk membuat komunikasi panggilan balik antar platform baik *chatbot service* ataupun halaman *web* menggunakan modul yang disebut dengan *REST channel* dan *websocket channel* RASA, (2022m) dengan cara menset *URL rasa main service* menjadi https://<HOST>/webhooks/webhooks pada file chat widget yang di-*embed* pada halaman web. Sebelumnya file *credentials.yml* dan *endpoints.yml* diset untuk meng-*expose webhook*

rasa service dan aktivasi *REST websocket channel*. File *credentials.yml* dan *endpoints.yml* diset pada mi-botway seperti Gambar 3.13 dan Gambar 3.14.

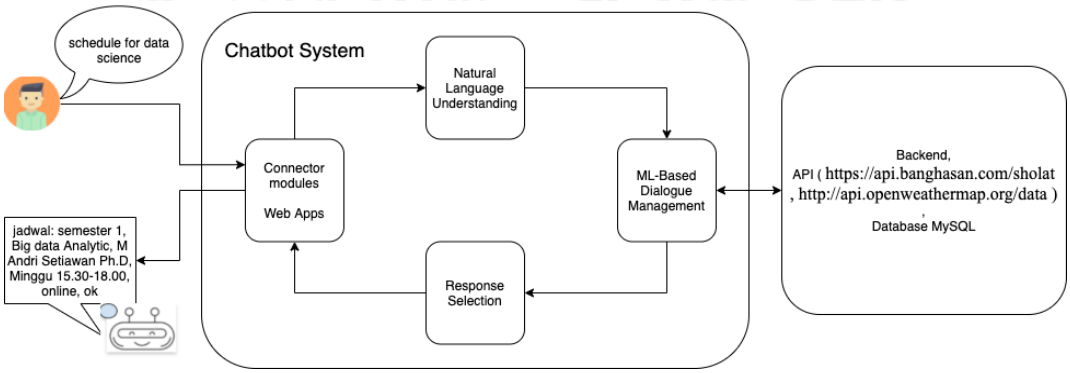
```
....
5   rest:
6   #  # you don't need to provide anything here - this channel doesn't
7   #  # require any credentials
....
20   socketio:
21     user_message_evt: user_uttered
22     bot_message_evt: bot_uttered
23     session_persistence: false
....
```

Gambar 3.13 File *credentials.yml* mi-botway (Windiatmoko, 2022h)

```
....
17                  action_endpoint:
18                   url: "http://action_server:5055/webhook"
....
```

Gambar 3.14 File *endpoints.yml* mi-botway (Windiatmoko, 2022i)

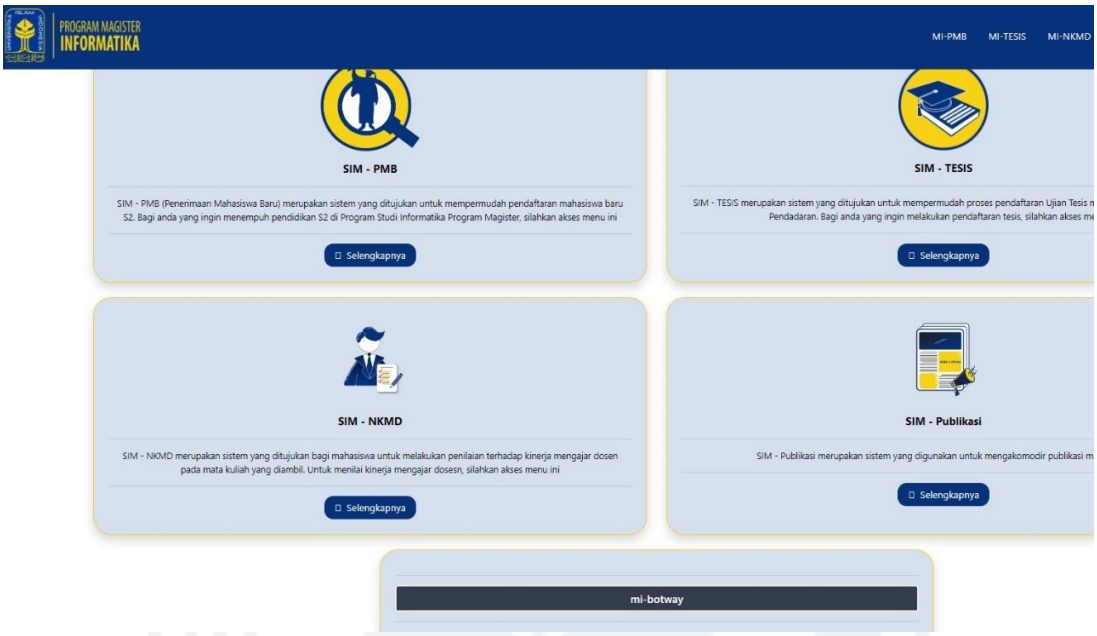
Hal ini menjadikan *webhook* memungkinkan chatbots untuk menerima pemberitahuan *HTTP* secara *real-time* tentang perubahan pada objek tertentu dan mengirimkannya kembali ke halaman antar pesan di platform web. Gambar 3.15 adalah ilustrasi aplikasi *web* sebagai modul konektor platform.



Gambar 3.15 Alur sistem chat

3.4.2 Perancangan Halaman ChatBot

Perancangan halaman chatbot didasari dengan mengembangkan web mi-gateway yang sudah ada pada STUDI INFORMATIKA PROGRAM MAGISTER, (2022), hanya menambahkan komponen semacam *popup* atau *dropdown* sebagai layer interface berkomunikasi dengan chatbot.



Gambar 3.16 Perancangan Halaman Awal

Pada tahapan ini kode dibangun berdasarkan template *bootstrap frontend* chatroom (Rasa Repository Chatbot Widget, 2022). Selanjutnya memodifikasi kode tersebut sesuai kebutuhan pengembangan mi-botway. Terutama menambahkan *script javascript* pada kode *html* mi-gateway. File kode *html*-nya sesuai Gambar 3.17.

```
  1   <head>
  2     <link rel="stylesheet"
href="http://127.0.0.1:8081/dist/Chatroom.css" />
  3   </head>
....
.... < -- ->
.... < - - - Mi-gateway html content - - - >
.... < -- ->
....
317     <script src="http://127.0.0.1:8081/dist/Chatroom.js"/></script>
318     <script type="text/javascript">
319       var chatroom = new window.Chatroom({
320         host: "http://127.0.0.1:5005",
321         title: "mi-botway",
322         container: document.querySelector(".chat-container"),
323   welcomeMessage: "assalamu alaikum, perkenalkan saya mi-botway
\nsaya bisa bantu untuk\n1.cek jadwal \n2.cek nilai \n3.cek cuaca
\n4.cek jadwal sholat \nformat pertanyaan simply tanya; cek jadwal, cek
nilai, cek cuaca, atau cek jadwal sholat"
324         // speechRecognition: "en-US",
325         // voiceLang: "en-US"
326       });
327       chatroom.closeChat();
328     </script>
329     <!-- FOOTER COPYRIGHT -->
330   <div class="row footer-bawah align-items-center">
331       <span class="copyright text-center">© Hak Cipta 2021 -
PROGRAM STUDI INFORMATIKA PROGRAM MAGISTER</span>
332   </div>
333   </body>
```

Gambar 3.17 File *index html* mi-botway (Windiatmoko, 2022j)

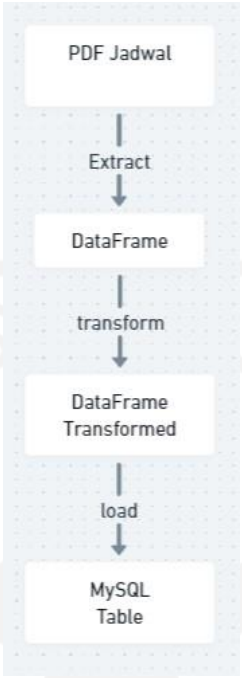
Penjelasan kode tersebut adalah sebagai berikut, pertama-tama *html* mengimport kode *javascript* pada *dist/Chatroom.js*, selanjutnya inisiasi *object class window.Chatroom* dengan parameter sesuai dengan kode backend terutama *endpoint API Rasa* di http://127.0.0.1:5005 , lalu yang lainnya merupakan parameter tambahan untuk memberikan keterangan chatbot seperti judul bot, dan welcome message. Sedangkan parameter container merupakan *DOM* untuk *retrieve response return* dari *backend API Rasa*.

3.4.3 Perancangan Table Database

Design database dilakukan berdasarkan keperluan data *completion response* chatbot, terutama untuk fitur daftar nilai dan daftar jadwal mahasiswa. Awal mula data didapat dari *pdf* jadwal mingguan yang biasa dikirim lewat emai oleh kampus. Dari file *pdf* tersebut di ekstrak menggunakan *library python tabula-py* yang dapat membaca tabel dalam *pdf* project package index, (2022b), lalu mengubahnya menjadi *pandas dataframe* (Docs Stable, 2022). *Tabula-py* juga memungkinkan untuk konversi file *pdf* menjadi file *csv*, *tsv*, atau *json*.

Selanjutnya setelah data tersebut menjadi *pandas dataframe* kemudian ditransformasikan dalam bentuk *dataframe* yang akan disesuaikan dengan *field table* yang

dibutuhkan saja. Bentuk transformasi terakhir *dataframe* kemudian di *load* ke *table mysql* menggunakan *library petl* project package index, (2022a) seperti ilustrasi Gambar 3.16.



Gambar 3.18 Ekstraksi transformasi *load table* jadwal kuliah

Selanjutnya file kode untuk ekstraksi *pdf* jadwal dan *load* ke *table mysql* akan ditunjukkan oleh Gambar 3.19.

```
import pymysql
import petl as etl
import tabula, os


def main():
    workdir = os.getcwd()

    # Step Extraction
    df = tabula.read_pdf(workdir+"/sample-jadwal.pdf", pages='all')

    table_name = "jadwal_kuliah"

    conn = pymysql.connect(
            host='127.0.0.1',
            user='root',
            database='chatbot_uii',
            port=3306,
            connect_timeout=5
            )

    conn.cursor().execute('SET SQL_MODE=ANSI_QUOTES')

    # Step Transformasi
    df.columns= ['semester','konsentrasi','matakuliah','pengajar','hari','jam','status','catatan']
        ks={'DS':'DATA SCIENCE', 'FD':'FORENSIKA DIGITAL', 'IM':'INFORMATIKA MEDIS', 'SIE':'SISTEM INFORMASI ENTERPRISE', 'Umum':'SEMUA'}
    df['ks']=df["konsentrasi"].map(ks)

    # Step Load DF to Table MySQL
    table = etl.fromdataframe(df)
    etl.todb(table, conn, table_name, create=True, drop=True)

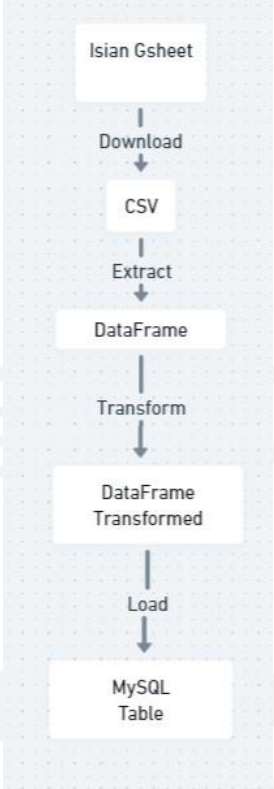
    conn.close()



if __name__ == "__main__":
    main()
```

Gambar 3.19 Ekstraksi *pdf* jadwal dan *load* ke *table mysql* (Windiatmoko, 2022a)

Sedangkan untuk nilai mahasiswa tahapan alur pembuatan *table*-nya sedikit mirip dengan *table* jadwal kuliah, hanya saja disini menggunakan data yang bersumber dari google spreadsheet yang diisi oleh rekan-rekan angkatan pada Sheets, (2022) , dari file tersebut di *download* data-nya menjadi csv, lalu csv tersebut di extract menjadi DataFrame ditransformasikan sesuai kebutuhan table dan di load ke table MySQL. Sesuai yang diilustrasikan Gambar 3.20.



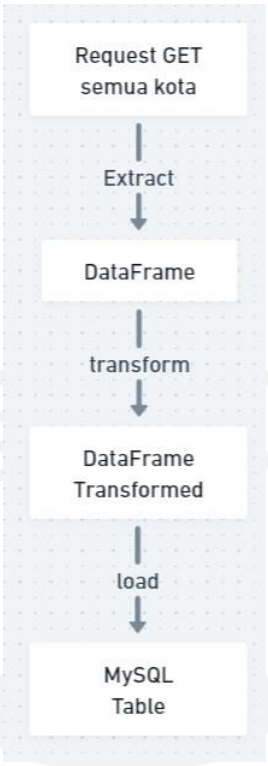
Gambar 3.20 Ekstraksi transformasi load table nilai mata kuliah

Selanjutnya file kode untuk ekstraksi nilai mata kuliah dan *load* ke *table mysql* akan ditunjukkan oleh Gambar 3.21.

```
 1    import pymysql
 2    import petl as etl
 3    import pandas as pd
 4    import os
 5
 6
 7    def main():
 8        workdir = os.getcwd()
 9
10        # Step Extraction
11        df      =      pd.read_csv(workdir+"/sample-nilai-kuliah.csv",
pages='all')
12
13        table_name = "nilai_mahasiswa"
14
15        conn = pymysql.connect(
16                host='127.0.0.1',
17                user='root',
18                database='chatbot_uii',
19                port=3306,
20                connect_timeout=5
21                )
22
23        conn.cursor().execute('SET SQL_MODE=ANSI_QUOTES')
24
25        # Step Transformasi
26        df.columns                                                    =
['no','kode','mata_kuliah','SKS','nilai','NIM','nama']
27
28        # Step Load DF to Table MySQL
29        table = etl.fromdataframe(df)
30        etl.todb(table, conn, table_name, create=True, drop=True)
31
32        conn.close()
33
34
35
36    if __name__ == "__main__":
37        main()
```

Gambar 3.21 Ekstraksi *pdf* jadwal dan *load* ke *table mysql* (Windiatmoko, 2022b)

Adapun untuk *table js_kota* atau disebut jadwal sholat kota, berisikan id kota dan nama kota yang digunakan untuk melakukan proses *request GET API*, agar mendapatkan daftar jadwal sholat di kota tertentu pada saat jadwal hari itu (API banghasan, 2022). *Table* kota untuk jadwal sholat ini didapat dari proses *request GET API* daftar semua kota, kemudian daftar semua kota itu diubah menjadi *dataframe* yang selanjutnya di *load* ke *table MySQL*. Sesuai yang diilustrasikan Gambar 3.22.



Gambar 3.22 Ekstraksi transformasi load table kota jadwal sholat

File kode untuk ekstraksi kota jadwal sholat dan *load* ke *table mysql* akan ditunjukkan oleh Gambar 3.23.

```
import pymysql
import petl as etl
import pandas as pd
import requests


def main():

    # Step Extraction
    url = 'https://api.banghasan.com/sholat/kota/semua'
    header = {'Content-Type': 'application/json'}
    try:
        response = requests.get(url, headers=header)
    except Exception as e:
        raise e
    results = response.json()
    df = pd.DataFrame(results)

    table_name = "js_kota"
    conn = pymysql.connect(
            host='127.0.0.1',
            user='root',
            database='chatbot_uii',
            port=3306,
            connect_timeout=5
            )

    conn.cursor().execute('SET SQL_MODE=ANSI_QUOTES')

    # Step Transformasi
    df.columns = ['id','nama']

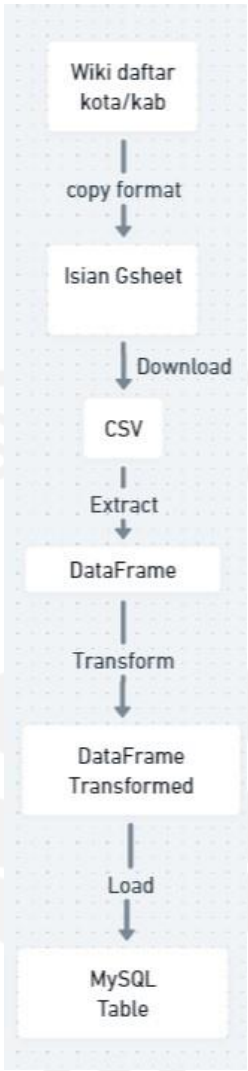
    # Step Load DF to Table MySQL
    table = etl.fromdataframe(df)
    etl.todb(table, conn, table_name, create=True, drop=True)

    conn.close()


if __name__ == "__main__":
    main()
```

Gambar 3.23 Ekstraksi kota jadwal sholat dan *load* ke *table mysql* (Windiatmoko, 2022c)

Sedangkan untuk *completion* prediksi cuaca dibutuhkan juga daftar kawasan kota atau kabupaten Indonesia, maka dibuatlah *table kota_indo* yang didapat dari wikipedia daftar kota dan kabupaten Indonesia (Wikipedia bahasa Indonesia, 2022). Tahapan alur pembuatan *table*-nya sedikit mirip dengan *table* nilai mahasiswa, hanya saja disini menggunakan data yang bersumber dari web kemudian dikopikan ke *google spreadsheet* dan disusun sesuai kebutuhan, dari file tersebut di *download* datanya menjadi *csv*, lalu *csv* tersebut di ekstrak menjadi *dataframe* ditransformasikan sesuai kebutuhan *table* dan di *load* ke *table MySQL* (sheets, 2022). Sesuai yang diilustrasikan Gambar 3.24.



Gambar 3.24 Ekstraksi transformasi load table Kawasan kota Indonesia

File kode untuk ekstraksi kawasan kota kabupaten dan *load* ke *table mysql* akan ditunjukkan oleh Gambar 3.25.

```
import pymysql
import petl as etl
import pandas as pd
import os


def main():
    workdir = os.getcwd()

    # Step Extraction
    df = pd.read_csv(workdir+"/kota-kab-indo.csv", pages='all')

    table_name = "kota_indo"

    conn = pymysql.connect(
            host='127.0.0.1',
            user='root',
            database='chatbot_uii',
            port=3306,
            connect_timeout=5
            )

    conn.cursor().execute('SET SQL_MODE=ANSI_QUOTES')

    # Step Transformasi
    df.columns = ['no','nama_kota','provinsi']

    # Step Load DF to Table MySQL
    table = etl.fromdataframe(df)
    etl.todb(table, conn, table_name, create=True, drop=True)

    conn.close()


if __name__ == "__main__":
    main()
```

Gambar 3.25 Ekstraksi kawasan kota kabupaten Indonesia dan *load* ke *table mysql* (Windiatmoko, 2022d)

Dikarenakan antara entitas satu table dengan table lainnya tidak memiliki hubungan atau keterkaitan apapun maka diantara table tersebut tidak dibuat skema relasi. Sehingga pada akhirnya diagram semua table yang ada pada *database* mi-botway adalah sebagai berikut.

Gambar 3.26 Diagram database mi-botway semua table

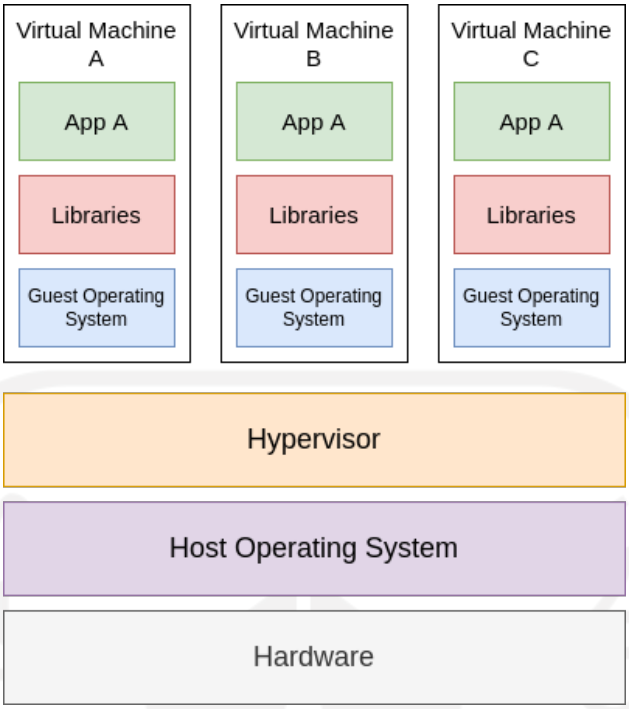
3.5 Deployment

Pembahasan *deployment* mi-botway ini terdiri dari penjelasan tentang *containerization apps*. Disebut *dockerisasi* semua komponen *service* mi-botway agar mempermudah proses *shipping* dan *running code* pada server. Dilanjut dengan penjelasan mengenai *setup mapping domain* untuk mi-botway *apps*.

3.5.1 Wrapping Code Backend dengan Container

Sedikit pengenalan tentang *docker* yang merupakan seperangkat platform layanan yang menggunakan virtualisasi tingkat *OS* untuk mengirimkan perangkat lunak dalam sebuah paket yang disebut *container*. Perangkat lunak yang menghosting container disebut *Docker Engine*. Ini pertama kali dimulai pada tahun 2013 dan dikembangkan oleh Docker, Inc (Wikipedia, 2022a).

Pendekatan awal yang dilakukan sebelumnya untuk implementasi berbagai *apps* dalam suatu komputer / server adalah pendekatan *VM* mungkin lebih akrab dengan brand-nya yaitu *Virtualbox* yang biasa dipakai untuk bermain-main dengan *linux OS*, dll. Pada *VM* setiap mesin akan memiliki *OS* mereka dengan *library* yang dibutuhkan tiap *apps*, ini membuat ukuran aplikasi cukup besar karena berisi *OS*. Diagramnya kurang lebih seperti di gambar 3.27 bawah ini.



Gambar 3.27 Penerapan *Virtual Machine* sebelum adanya *Containerization*

Hal ini sebenarnya bertujuan untuk mengisolir aplikasi di dalam *VM* agar *apps* satu sama lainnya tidak membuat komputer / server, *crash dependency* karena interdependensi *library*. Konsep dasar *container* mirip dengan *container* dalam kehidupan nyata, yaitu cara mengemas semuanya menjadi satu.

Seringkali terjadi pertanyaan sebagai berikut pada khalayak umum saat berkutat dengan *setup* aplikasi:

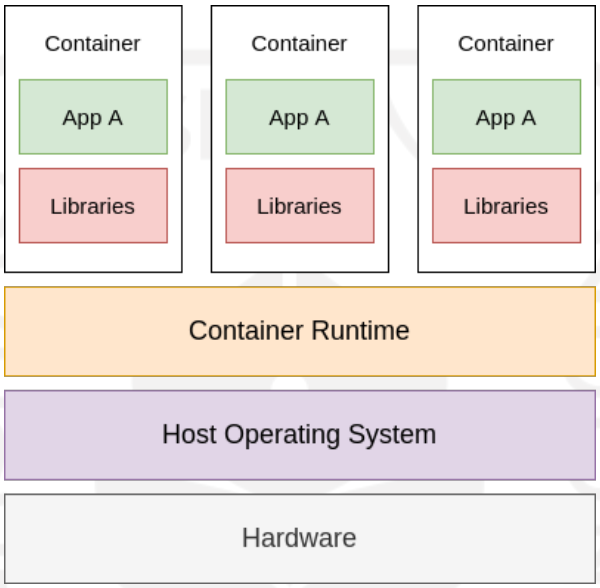
1. Aplikasi bekerja dengan lancar di laptop pribadi, tetapi mengapa tidak bekerja di komputer rekan lainnya?

Kemungkinan besar hal ini dikarenakan belum menerapkan *containerization* pada aplikasi tersebut yang memaksa mereka untuk melakukan instalasi langkah demi langkah setiap kali mereka ingin menguji aplikasi di mesin lokal. Apalagi instalasi yang menggunakan dependensi *library* berbeda pada masing-masing *Sistem Operasi*. Katakanlah ketika menginstal *NodeJS* di Windows memiliki langkah yang berbeda saat menginstalnya di *Linux*.

Tapi dengan *container*, kita tidak akan direpotkan dengan melakukan hal-hal yang berulang-ulang yang akan menghemat banyak waktu dan bisa fokus pada pengembangan. Lebih khusus lagi, wadah membungkus dan mengemas kode, dependensi, konfigurasi dalam satu file dan membuatnya terisolasi satu sama lain dengan *OS*. Meskipun mereka terisolasi

tetapi mereka dapat berkomunikasi melalui jaringan yang merupakan bagian dari *containerization*.

Seperti yang kita lihat sebelumnya bahwa *VM* berjalan di atas *Hypervisor* sementara *container* berjalan di atasnya, disebut sebagai *runtime container*. Ada beberapa *runtime container* tetapi yang paling populer adalah *Docker*. Dapat dilihat pada gambar 3.28 skema penerapan *container* pada Komputer / *server*.



Gambar 3.28 Penerapan *Containerization* pada computer / server

Seperti itulah *docker*, sesuai yang sudah dijelaskan sebelumnya, untuk penerapannya pada aplikasi chatbot mi-botway ini adalah seperti yang tertera pada gambar 3.29 sebagai berikut.

```
version: "2"

services:
  rasa:
    build:
      context: ./backend
      dockerfile: Dockerfile
    entrypoint: ./entrypoint.sh
    ports:
      - "5002:5002"
      - "5005:5005"
      # http://localhost:5002/api
    depends_on:
      - action_server
    env_file:
      - ./backend/.env
    container_name: rasa-wrapper

  action_server:
    build:
      context: ./backend
      dockerfile: Dockerfile
    entrypoint: ./action_entrypoint.sh
    container_name: action_server
    environment:
      - MYSQL_HOST=some-mariadb
    ports:
      - "5055:5055"
    depends_on:
      - some-mariadb

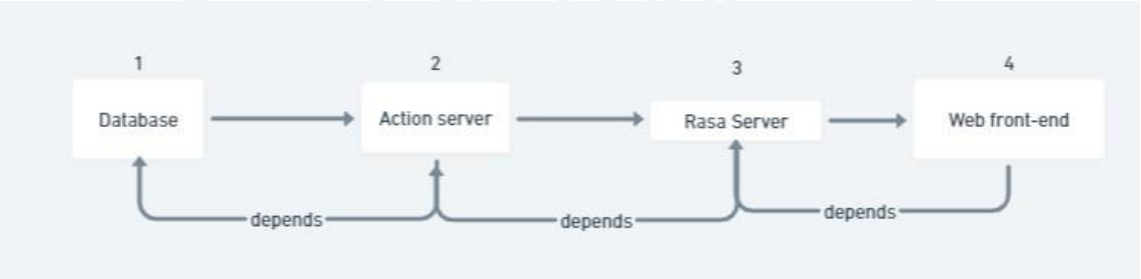
  some-mariadb:
    image: mariadb:10.6
    environment:
      - MYSQL_ALLOW_EMPTY_PASSWORD=yes
    volumes:
      - ./backend/mariadb:/var/lib/mysql
    ports:
      - "3306:3306"

  web:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "8081:8080"
    stdin_open: true
    tty: true
    depends_on:
      - rasa
```

Gambar 3.29 File *docker-compose.yml* untuk *running* aplikasi mi-botway (Windiatmoko, 2022k)

Terdapat 4 container utama pertama yaitu ***rasa*** sebagai *main server* bertugas untuk *serving model platform* atau *service* utama chatbot yang langsung berkomunikasi dengan *web front-end*, dan ***action_server*** *rasa* untuk *completion response* mi-botway. Selanjutnya

action_server ini bertugas juga dalam berkomunikasi dengan pihak *service* eksternal seperti *database* dan *API* untuk dapat memberikan *completion response* sesuai yang diharapkan. Lalu komponen *container* ***some-mariadb*** yaitu *MariaDB* sebagai *MySQL database* penyimpanan *table data tabular*. Dan terakhir ***web*** sebagai *interface user front-end* untuk melakukan komunikasi dengan chatbot di halaman mi-gateway. Dependensi setiap komponen dengan yang lainnya dapat dilihat melalui gambar 3.30.



Gambar 3.30 Dependensi setiap komponen *container*

Urutan *container running* adalah seperti gambar 3.28 diatas, dari mulai *database* lanjut *action server*, *Rasa server* hingga akhir *web front-end running* terakhir. Ini berurutan berdasarkan kebergantungan suatu *services* yaitu *web front-end* tidak akan bisa berjalan sebelum *API service* utama *rasa server* berjalan, lalu *rasa server* tidak bisa memberikan respons yang sempurna jika tidak ada *action server*, dan *action server* memanfaatkan *database* untuk *retrieve data* yang dibutuhkan untuk *completion*.

3.5.2 Setup Mapping Domain

Untuk melakukan *setup domain* suatu *service web* dibutuhkan suatu *web server* terlebih dahulu untuk menampung atau mensentralisasi *service*. Istilah *web server* itu dikenal sebagai perangkat lunak komputer yang menerima permintaan melalui *HTTP* (protokol jaringan yang dibuat untuk mendistribusikan konten *web*) atau varian amannya *HTTPS*. Agen pengguna, biasanya *browser web*, memulai komunikasi dengan membuat permintaan untuk halaman web atau sumber daya lain menggunakan *HTTP*, dan *server* merespons dengan konten sumber daya itu ataupun mengembalikan *error message*. *web server* juga dapat menerima dan menyimpan sumber daya yang dikirim dari agen pengguna jika dikonfigurasi untuk melakukannya disebut juga dengan istilah *caching* (Wikipedia, 2022b). Pada mi-botway *web server* yang digunakan adalah *nginx* (Official Docs, 2022). Proses

setup nginx oleh Glass, (2022) dan *secure https* oleh Heidi, (2022) berdasarkan tutorial pada *digital ocean*. Setelah semua itu tersetup maka selanjutnya mengkonfigurasi antar service pada nginx dengan mengedit file *default* pada *system path nginx*, untuk penerapannya pada aplikasi mi-botway ini adalah seperti yang tertera pada Gambar 3.31.

```
 1  map $http_upgrade $connection_upgrade {
 2    default upgrade;
 3    ''      close;
 4  }
 5
 6  server {
 7
 8          root /home/yurio/app/mi-botway-monorepo/frontend;
 9
10          index index.html index.htm index.nginx-debian.html;
11
12          server_name yuriowindi.space www.yuriowindi.space;
13
14  # location / {
15  #       return 301 /web/;
16  # }
17
18          location /mi-botway/ {
19                  proxy_pass http://127.0.0.1:8080/;
20                  proxy_http_version 1.1;
21                  proxy_set_header Upgrade $http_upgrade;
22                  proxy_set_header Connection $connection_upgrade;
23                  rewrite ^(/web/[^/]+)$ $1/ permanent;
24          }
25
26
27          location /api/ {
28                  proxy_pass http://127.0.0.1:5005/;
29                  proxy_http_version 1.1;
30                  proxy_set_header Upgrade $http_upgrade;
31                  proxy_set_header Connection $connection_upgrade;
32          }
33
34      listen [::]:443 ssl ipv6only=on; # managed by Certbot
35      listen 443 ssl; # managed by Certbot
36                                                      ssl_certificate
/etc/letsencrypt/live/yuriowindi.space/fullchain.pem;   #   managed   by
Certbot
37                                                    ssl_certificate_key
/etc/letsencrypt/live/yuriowindi.space/privkey.pem; # managed by Certbot
38       include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
39         ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
Certbot
40  }
41
42  server {
43      if ($host = www.yuriowindi.space) {
44          return 301 https://$host$request_uri;
45      } # managed by Certbot
46
47
48      if ($host = yuriowindi.space) {
49          return 301 https://$host$request_uri;
….
```

Gambar 3.31 File *config nginx* untuk *web server* mi-botway

3.6 Penjelasan Struktur dan Kode

Pembahasan dan penjelasan struktur kode mi-botway ini dibagi menjadi dua bagian, pertama *overview* struktur pembagian *folder* untuk mi-botway *apps*. Dilanjut dengan penjelasan *logic* dibalik kode *action* mi-botway.

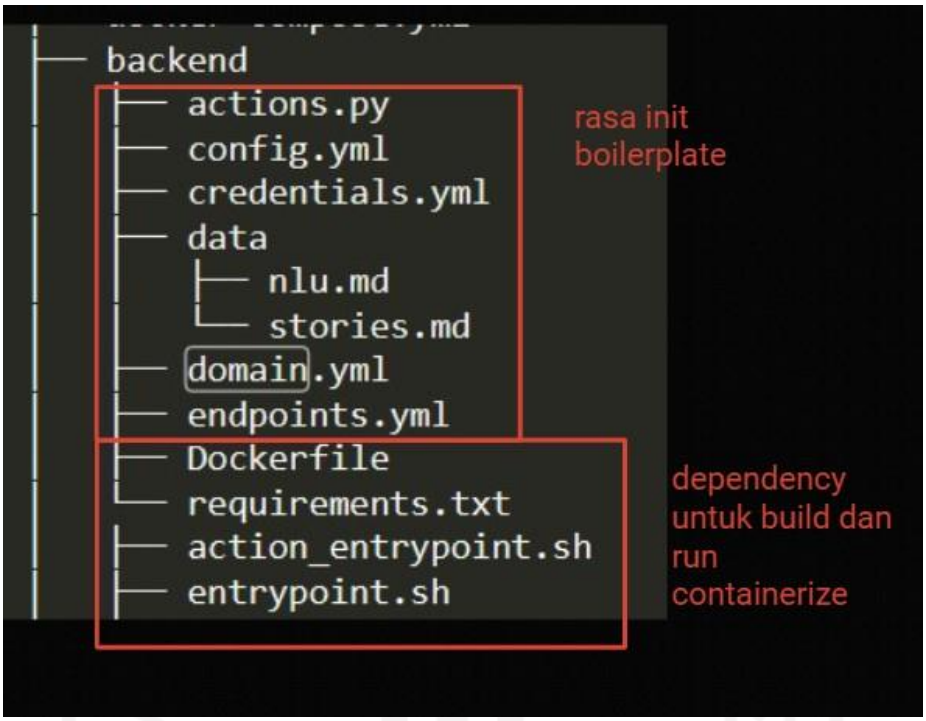
3.6.1 Struktur Kode Chatbot

Struktur kode utama terbagi menjadi dua bagian utama yaitu *frontend* dan *backend* dan pada *root project* hanya ada *docker-compose* yang merupakan orkestrator *docker container* dan *readme* untuk penjelasan dalam mengeksekusi keseluruhan kode (Windiatmoko, 2022l). Berikut merupakan penjabaran struktur kode adalah seperti pada Gambar 3.32:

```
➜  mi-botway-monorepo git:(master) tree
.
├── docker-compose.yml
├── backend
│   ├── actions.py
│   ├── config.yml
│   ├── credentials.yml
│   ├── data
│   │   ├── nlu.md
│   │   └── stories.md
│   ├── domain.yml
│   ├── endpoints.yml
│   ├── Dockerfile
│   ├── requirements.txt
│   ├── action_entrypoint.sh
│   ├── entrypoint.sh
│   ├── mariadb
│   │   ├── chatbot_uii
│   │   │   ├── jadwal_kuliah.frm
│   │   │   ├── jadwal_kuliah.ibd
│   │   │   ├── js_kota.frm
│   │   │   ├── js_kota.ibd
│   │   │   ├── kota_indo.frm
│   │   │   ├── kota_indo.ibd
│   │   │   ├── nilai_mahasiswa.frm
│   │   │   └── nilai_mahasiswa.ibd
│   ├── model_eval_results
│   │   ├── GRU_128_rnn_size
│   │   │   └── GRU_128_rnn_size.tar.gz
│   │   ├── GRU_32_rnn_size
│   │   │   └── GRU_32_rnn_size.tar.gz
│   │   ├── GRU_64_rnn_size
│   │   │   └── GRU_64_rnn_size.tar.gz
│   │   ├── LSTM_128_rnn_size
│   │   │   └── LSTM_128_rnn_size.tar.gz
│   │   ├── LSTM_32_rnn_size
│   │   │   └── LSTM_32_rnn_size.tar.gz
│   │   ├── LSTM_64_rnn_size
│   │   │   └── LSTM_64_rnn_size.tar.gz
│   │   ├── SimpleRNN_128_rnn_size
│   │   │   └── SimpleRNN_128_rnn_size.tar.gz
│   │   ├── SimpleRNN_32_rnn_size
│   │   │   └── SimpleRNN_32_rnn_size.tar.gz
│   │   └── SimpleRNN_64_rnn_size
│   │       └── SimpleRNN_64_rnn_size.tar.gz
└── frontend
    ├── src
    │   ├── Chatroom.js
    │   ├── Chatroom.scss
    │   ├── ConnectedChatroom.js
    │   ├── DebuggerView.js
    │   ├── Message.js
    │   ├── SpeechInput.js
    │   ├── index.js
    │   ├── messages.json
    │   └── utils.js
    ├── dist
    │   ├── Chatroom.css
….
```

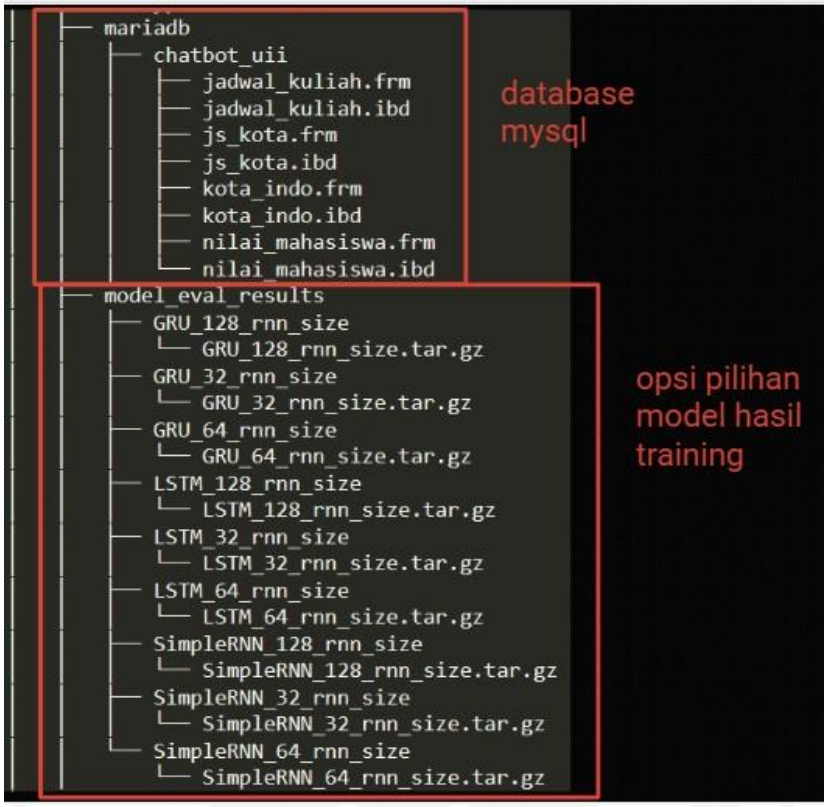
Gambar 3.32 Struktur *project* aplikasi mi-botway

Penjelasan, pertama pada bagian *backend* yaitu komponen utama saat meng-initialize *rasa project* atau biasa disebut komponen *boilerplate* yaitu *actions, config, credentials, data, domain* dan *endpoints* (Docs of RASA, 2022). Lalu komponen untuk *container rasa backend* yang terbagi menjadi dua bagian yaitu *rasa action server* dan *rasa main server*. Hal itu terlihat pada Gambar 3.33.



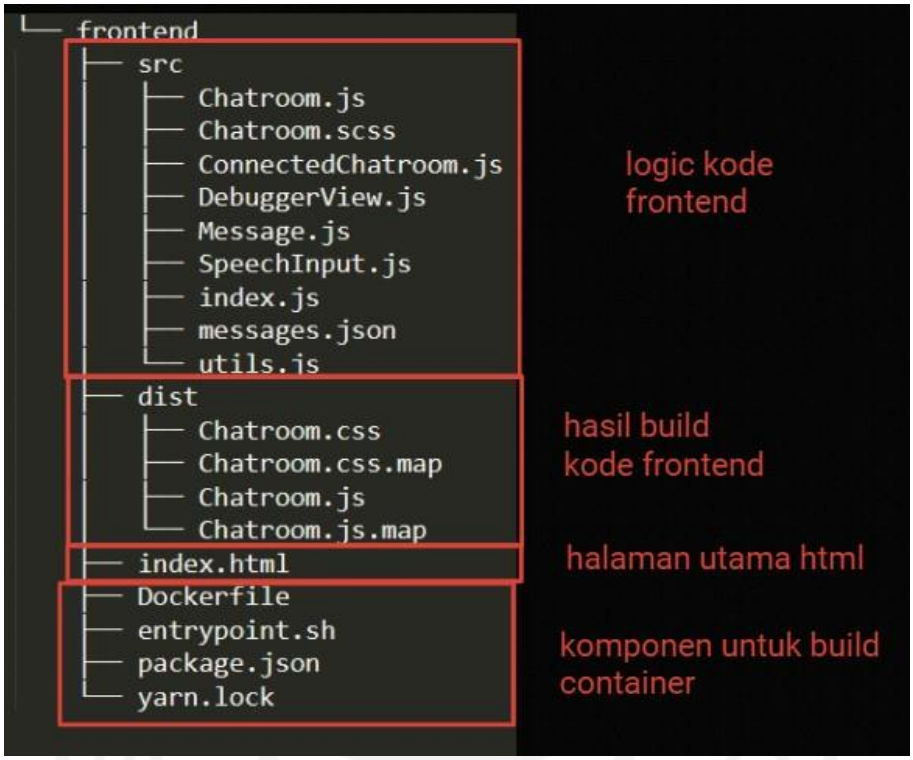
Gambar 3.33 Penjabaran struktur kode *backend rasa boilerplate* dan *container*

Penjelasan selanjutnya, pada bagian *database* yaitu komponen data tabular *mariadb* yang sudah terisi data, dimana hal ini selanjutnya akan digunakan sebagai *volume container* sebagai *persistent data* (Docker, 2022). Lalu komponen opsi model hasil *training* yang dapat digunakan oleh chatbot sebagai model utama saat *running*. Hal itu terlihat pada Gambar 3.34.



Gambar 3.34 Penjabaran struktur kode database dan opsi model

Selanjutnya, bagian *frontend* yaitu komponen chatbot widget rasa (Rasa Repository Chatbot Widget, 2022) yang terdiri dari *logic code frontend* pada *src,* dan sisanya merupakan *index.html* mi-botway, hasil *build* kode *javascript* pada *dist*, serta komponen *frontend* untuk *build container*. Struktur tersebut dapat terlihat pada Gambar 3.35.



Gambar 3.35 Penjabaran struktur kode frontend

3.6.2 Source Code Komputasi

Komputasi lebih banyak difokuskan pada *logic completion response* chatbot, yaitu untuk melengkapi response yang sesuai dan diharapkan user terutama menampilkan daftar jadwal nilai, jadwal sholat ataupun prediksi cuaca, pada bagian ini disebut *actions* dalam komponen *framework* chatbot rasa. Pertama adalah *logic completion* untuk *response* daftar jadwal adalah seperti yang tertera pada Gambar 3.36.

```
…      .
 36    class ActionDaftarJadwal(Action):
 37        def name(self) -> Text:
 38            return "action_daftar_jadwal"
 39
 40        def run(
 41            self,
 42            dispatcher: CollectingDispatcher,
 43            tracker: Tracker,
 44            domain: Dict[Text, Any],
 45        ) -> List[Dict[Text, Any]]:
 46
 47            conn = pymysql.connect(
 48                host=os.environ.get("MYSQL_HOST"),
 49                user="root",
 50                database="chatbot_uii",
 51                port=3306,
 52                connect_timeout=5,
 53            )
 54            mycursor = conn.cursor()
 55
 56            konsentrasi = tracker.get_slot("konsentrasi")
 57
 58            if konsentrasi is None:
 59                dispatcher.utter_message(text="mohon maaf prodi blm
terdaftar di system kami..")
 60                return []
 61
 62            konsentrasi = konsentrasi.upper()
 63
 64            if konsentrasi == "DATA SAINS" or konsentrasi == "SAINS
DATA":
 65                konsentrasi = "DATA SCIENCE"
 66
 67            # message = '''semester | matakuliah | pengajar | hari |
jam | status | catatan''' + '''\n'''
 68            message = "jadwal" + """\n"""
 69            mycursor.execute(
 70                "select * from jadwal_kuliah where konsentrasi='"
 71                + konsentrasi
 72                + "' or ks='"
 73                + konsentrasi
 74                + "'"
 75            )
 76            results = mycursor.fetchall()
 77
 78            for result in results:
 79                message += (
 ….
```

Gambar 3.36 *Logic completion* untuk *response* daftar jadwal (Windiatmoko, 2022m)

Penjelasan kode tersebut yaitu, pertama mendefinisikan nama *action* yang akan diregistrasikan pada *rasa framework* chatbot yaitu *action_daftar_jadwal*. Selanjutnya mendefinisikan *method* utama untuk melakukan *completion response*, parameter nya adalah *dispatcher* untuk memberikan *return completion chatbot*, *tracker* untuk menangkap *entity* yang akan digunakan sebagai *key* pada *completion*, dan *domain* untuk menjaga konteks chatbot. Disini yang dilakukan adalah adalah inisiasi *connection cursor MySQL*, lalu

menangkap *entity konsentrasi* dari *message user*, *konsentrasi* yang didapat dilanjutkan dengan *query* pada *table jadwal_kuliah* dengan filter *konsentrasi*. Menyusun *message response* dari *return query* dengan *text* sebagai berikut;

1. “semester | matakuliah | pengajar | hari | jam | status | catatan”
2. text tersebut akan bertambah sesuai return dari query yang didapat.

Kedua adalah *logic completion* untuk *response* daftar nilai adalah seperti Gambar 3.37 berikut.

```
102    class ActionDaftarNilai(Action):
103        def name(self) -> Text:
104            return "action_daftar_nilai"
105
106        def run(
107            self,
108            dispatcher: CollectingDispatcher,
109            tracker: Tracker,
110            domain: Dict[Text, Any],
111        ) -> List[Dict[Text, Any]]:
112
113            conn = pymysql.connect(
114                host=os.environ.get("MYSQL_HOST"),
115                user="root",
116                database="chatbot_uii",
117                port=3306,
118                connect_timeout=5,
119            )
120            mycursor = conn.cursor()
121
122            try:
123                nim = tracker.get_slot("NIM")
124            except:
125                nim = None
127            try:
128                nama = tracker.get_slot("nama")
129            except:
130                nama = None
132            if nim is None and nama is None:
133                dispatcher.utter_message(text="mohon maaf nim atau
nama blm terdaftar di system kami..")
134                return []
135
136            nama = nama.upper()
137
138            message = "daftar nilai" + """\n"""
139
140            if nim is not None:
141                mycursor.execute(
142                    "select mata_kuliah, SKS, nilai from
nilai_mahasiswa where NIM='"
143                    + nim
144                    + "'"
```

Gambar 3.37 *Logic completion* untuk *response* daftar nilai (Windiatmoko, 2022n)

Penjelasan kode tersebut adalah, pertama mendefinisikan nama *action* yang akan diregistrasikan pada *rasa framework chatbot* yaitu *action_daftar_nilai*. Selanjutnya definisikan *method* utama untuk melakukan *completion response*, parameternya sama seperti *object class action_daftar_jadwal* sebelumnya. Disini yang dilakukan adalah adalah inisiasi *connection cursor MySQL*, lalu mencoba menangkap *entity NIM* atau *nama* sesuai dari *input message user*, apakah *user* menggunakan *nama* ataupun *NIM*. *Nama* atau *NIM* yang didapat dilanjutkan dengan *query* pada *table nilai_mahasiswa* dengan filter *nama* ataupun *NIM*. Menyusun message *response* dari *return query* dengan *text* sebagai berikut;

1. "Mata kuliah | sks | nilai"
2. text tersebut akan bertambah sesuai return dari query yang didapat.

Ketiga adalah *logic completion* untuk *response jadwal sholat* adalah seperti Gambar 3.38 berikut.

```
....
171    class ActionJadwalSholat(Action):
172        def name(self) -> Text:
173            return "action_jadwal_sholat"
174
175        def run(
176            self,
177            dispatcher: CollectingDispatcher,
178            tracker: Tracker,
179            domain: Dict[Text, Any],
180        ) -> List[Dict[Text, Any]]:
181
182            conn = pymysql.connect(
183                host=os.environ.get("MYSQL_HOST"),
184                user="root",
185                database="chatbot_uii",
186                port=3306,
187                connect_timeout=5,
188            )
189            mycursor = conn.cursor()
190
191            kawasan = tracker.get_slot("kota")
192
193            if kawasan is None:
194                dispatcher.utter_message(text="mohon maaf kota atau
kawasan blm terdaftar di system kami..")
195                return []
196
197            kawasan = kawasan.lower()
198
199            mycursor.execute(
200                "select id from js_kota where nama_kota like '%" +
kawasan + "%'"
201            )
202
203            results = mycursor.fetchall()
204
205            results_id = results[0][0]
206
207            url                                                    =
"https://api.banghasan.com/sholat/format/json/jadwal/kota/{kota_id}/tan
ggal/{date_now}".format(
208                kota_id=results_id,
date_now=datetime.strftime(datetime.now(), "%Y-%m-%d")
209            )
210
211            header = {"Content-Type": "application/json"}
212
213            response = requests.get(url, headers=header)
214
215            results = response.json()
216
....
```

Gambar 3.38 *Logic completion* untuk *response* jadwal sholat (Windiatmoko, 2022o)

Penjelasan alur *logic* kode yaitu, pertama mendefinisikan nama *action* yang akan diregistrasikan pada *rasa framework chatbot* yaitu *action_jadwal_sholat*. Selanjutnya mendefinisikan *method* utama untuk melakukan *completion response*, parameternya sama seperti *object class action_daftar_jadwal* sebelumnya. Disini yang dilakukan adalah adalah inisiasi *connection cursor MySQL*, lalu mencoba menangkap *entity kota* dari *input message user*, *entity kota* yang didapat dilanjutkan dengan *query* pada *table js_kota* dengan filter *kota*. Selanjutnya *id kota* yang didapat dari *return query* akan dilanjutkan dengan *request API banghasan jadwal sholat* hingga akhirnya mendapat *return message jadwal sholat* berdasarkan area kota tadi. Menyusun message response dari *return API* dengan *text* sebagai berikut;

1. replace json response API dari tanda baca yang tidak digunakan
2. text jadwal sholat berdasarkan kota menjadi *completion message*

Terakhir adalah logic completion untuk *response* prediksi cuaca adalah seperti Gambar 3.39 berikut.

```
....
230    class ActionPredCuaca(Action):
231        def name(self) -> Text:
232            return "action_pred_cuaca"
233
234        def run(
235            self,
236            dispatcher: CollectingDispatcher,
237            tracker: Tracker,
238            domain: Dict[Text, Any],
239        ) -> List[Dict[Text, Any]]:
240
241            conn = pymysql.connect(
242                host=os.environ.get("MYSQL_HOST"),
243                user="root",
244                database="chatbot_uii",
245                port=3306,
246                connect_timeout=5,
247            )
248            mycursor = conn.cursor()
249
250            kawasan = tracker.get_slot("kota")
251
252            if kawasan is None:
253                dispatcher.utter_message(text="mohon  maaf  kota  atau
kawasan blm terdaftar di system kami..")
254                return []
255
256            kawasan = kawasan.lower()
257
258            mycursor.execute(
259                "select nama_kota from kota_indo where nama_kota like
'%" + kawasan + "%'"
260            )
261
262            results = mycursor.fetchall()
263
264            results_kawasan = results[0][0]
265
266            if len(results_kawasan.split()) > 1:
267                kawasan_req = "%20".join(results_kawasan.split())
268            else:
269                kawasan_req = results_kawasan
270
271            url                                                    =
"http://api.openweathermap.org/data/2.5/weather?q={kawasan},id&APPID={a
pp_id}&units=metric".format(
272                kawasan=kawasan_req,
app_id="9926258e5fbe65eb74b1eb220416b8d2"
…..
```

Gambar 3.39 *Logic completion* untuk *response* prediksi cuaca (Windiatmoko, 2022p)

Penjelasan kode tersebut, pertama mendefinisikan nama *action* yang akan diregistrasikan pada *rasa framework chatbot* yaitu *action_pred_cuaca*. Selanjutnya mendefinisikan *method* utama untuk melakukan *completion response*, parameternya sama seperti *object class action_daftar_jadwal* sebelumnya. Disini yang dilakukan adalah adalah inisiasi *connection cursor MySQL*, lalu mencoba menangkap *entity kota* dari *input message user*, *entity* kota

yang didapat dilanjutkan dengan *query* pada *table kota_indo* dengan filter *kota*. Selanjutnya nama kota yang lebih generic dan sesuai dengan daftar nama kota untuk *request API openweathermap* didapat dari *return query* akan dilanjutkan dengan *request API openweathermap* tersebut dengan *key parameter nama kota* dan jika nama kota memiliki lebih dari 1 kata maka akan di *join split* dengan *separator text "%20"* hingga akhirnya mendapat *return message prediksi cuaca* berdasarkan area kota tadi. Dikarenakan *return* prediksi tersebut berbahasa inggris maka *return message* tersebut diterjemahkan dulu menggunakan *google translate library*. Menyusun *message response* dari *return API* dengan *text* sebagai berikut;

1. ambil deskripsi prediksi cuaca area kota tersebut
2. text deskripsi prediksi cuaca yang telah diterjemahkan ke Bahasa Indonesia menjadi *completion message*

3.7 Metode Survey Evaluasi

Evaluasi tambahan yang digunakan dalam penelitian ini adalah metode penelitian survei dan metode penelitian deskritif verifikatif dengan pendekatan kuantitatif. Tujuan dalam evaluasi ini adalah untuk membuktikan kemudahan penggunaan chatbot mi-botway. Metode verifikatif menurut (Sugiyono, 2012) diartikan "sebagai penelitian yang dilakukan terhadap populasi atau sampel tertentu dengan tujuan untuk menguji hipotesis yang telah ditetapkan". Selanjutnya metode survey dan verifikatif adalah "metode survey yang digunakan untuk mendapatkan data dari tempat tertentu yang alamiah (bukan buatan), tetapi peneliti melakukan perlakuan dalam pengumpulan data, misalnya dengan mengedarkan kuesioner". Survey dilakukan dengan metode verifikatif karena melakukan pengujian dan memverifiikasi kebenaran untuk kemudahan penggunaan chatbot mi-botway.

BAB 4
Hasil dan Pembahasan

4.1 Analisis Hasil

Pada bagian ini, analisis hasil yang disajikan merupakan nilai-nilai *presisi*, *recall*, dan *f1-skor* yang sebelumnya telah dibahas secara konsep pada subbab 3.3.2 yang didapat dari evaluasi model *NLU* dan *dialogue policy* mi-botway. Tabel 4.1 adalah hasil evaluasi metrik chatbot mi-botway pada model *intent classifier*. Seperti yang tertera pada Gambar 3.5 sebelumnya terdapat simulasi data *intent* (Windiatmoko, 2022e). Untuk setiap kelompok data kalimat intent tersebut akan dibagi 15 persennya untuk menjadi data evaluasi, sebagai acuan perhitungan metriks evaluasi sesuai yang disajikan pada setiap tabel.

Tabel 4.1 Metrik evaluasi *Intent Classifier*

	presisi	recall	f1 -skor
intent selamat tinggal	1	1	1
intent konfirmasi kota	1	1	1
intent konfirmasi nama atau nim	1	1	1
intent jadwal ibadah	1	1	1
intent menolak	1	1	1
intent sapaan	1	0.85	0.93
menantang bot	1	1	1
suasana hati yang baik	1	1	1
intent minta daftar nilai	1	1	1
intent jadwal permintaan	1	1	1
intent permintaan jadwal saja	1	1	1
intent prediksi cuaca	1	1	1
intent sapaan muslim	0.75	1	0.8
intent setuju	1	1	1
bad mood	1	1	1
terima kasih	1	1	1

Tabel 4.1 menunjukkan evaluasi *intent classifier* metrik untuk proses klasifikasi. Perlu diperhatikan kelas *intent sapaan* dimana model dapat secara akurat memprediksi jumlah data kebenaran untuk kelas tersebut dengan nilai *recall* 1, namun model masih memprediksi *intent sapaan* di kelas *intent* lainnya dengan akurasi skor 0,75 yaitu kelas *intent sapaan muslim* seperti yang ditunjukkan pada Tabel 4.1.

Pada Tabel 4.2 merupakan metrik evaluasi untuk menjelaskan kebingungan model *intent classifier*. Pada table tersebut *intent sapaan* sendiri mendapat skor presisi 1 dimana *intent sapaan* terkadang tidak terdeteksi oleh model, sebaliknya *intent sapaan muslim* terdeteksi karena menggunakan frasa kata yang sama yaitu *'assalamu alaikum'* namun hal ini tidak menjadi masalah. karena *template chatbot utter salam* akan menjawab hal yang sama dengan

utter Greeting. Pada tabel 4.2 memberikan gambaran yang lebih rinci penjelasan presisi dan perhitungan *confusion matrix* mengingat dua kelas label yang sering salah diklasifikasikan. Dapat dilihat dari tabel 4.2 bahwa presisi *intent sapaan* bernilai 1, hal ini menunjukkan bahwa sistem chatbot mi-botway mampu mendeteksi *intent sapaan* dengan benar. Sedangkan untuk *intent sapaan muslim*, ketepatan skornya adalah 0,75.

Tabel 4.2 Matriks Kebingungan antara kelas label yang salah klasifikasi

Predict VS Actual	Data Hitung			
	Intent sapaan Muslim	*Intent* sapaan		*Recall*
Intent sapaan Muslim	3	0	3	1
Intent sapaan	1	6	7	0.85
Total Prediksi	4	6	-	-
Presisi	0.75	1	-	-

Pada tabel 4.3, metrik evaluasi kelas *entity* telah menunjukkan kinerja terbaik dari model. Model *Entity* sudah memiliki kemampuan untuk mengekstrak masing-masing label *entity* dari setiap kalimat yang diharapkan.

Tabel 4.3 Metrik Evaluasi Pengenalan Entitas

	Nama	Program Studi	Kota	NIM	Rata-rata	Total
Presisi	1	1	1	1	1	4
Recall	1	1	1	1	1	4
F1-Score	1	1	1	1	1	4

Proses alur percakapan dibagi menjadi berbagai alur atau cerita, berdasarkan kemampuan atau fitur chatbot. Pertama fitur jadwal, mahasiswa yang meminta jadwal kuliah dari chatbot memiliki proses sebagai berikut. Obrolan dimulai dengan siswa memberi salam kepada chatbot, dan chatbot mendeteksi *intent* dari pengguna dan membalas dengan balasan *utter* salam. Kemudian siswa langsung ke intinya untuk bertanya tentang jadwal kelas, dan chatbot mendeteksinya sebagai *intent* untuk meminta penjadwalan, lalu chatbot menjawab dengan menanyakan secara tuntas tentang konsentrasi program studi mahasiswa. Kemudian siswa memberitahu chatbot tentang konsentrasi program studinya yaitu *'fd'* atau *digital forensik*, dan chatbot menganggap ini sebagai *entity* konsentrasi program studi untuk

melanjutkan *entity* tersebut melalui *query MySQL DB* sebagai penyelesaian untuk mendapatkan *data jadwal*.

Proses request jadwal juga bisa dilalui dengan menanyakan langsung jadwal konsentrasi program studi tertentu, misalnya *'jadwal ds'* dan chatbot akan mendeteksi maksud untuk meminta jadwal serta mendeteksi konsentrasi entitas studi program *data science*. Kemudian chatbot dilanjutkan melalui *query* pada *MySQL DB* sebagai pelengkap untuk mendapatkan data jadwal perkuliahan.

Misalnya percakapan mahasiswa yang menanyakan daftar nilai. Alur percakapan ini tentang meminta daftar nilai siswa, hampir sama dengan alur meminta jadwal kelas, tetapi hanya melalui *intent* dan *entity* yang berbeda, serta tindakan yang berbeda untuk menampilkan daftar nilai, *entity nama siswa* atau *NIM* adalah *key* yang dilanjutkan ke *query* penyelesaian melalui *MySQL DB* sebagai pelengkap untuk mendapatkan daftar nilai siswa yang bersangkutan.

Selain itu, ada juga percakapan untuk menanyakan waktu sholat. Dimulai dengan bertanya langsung dengan *intent menanyakan jadwal sholat muslim*. Kemudian chatbot menanyakan lokasi tertentu sebagai *entity*, setelah itu melanjutkan *query* penyelesaian langsung ke *API* https://api.banghasan.com/sholat/format/json/jadwal/kota/[*kota*]/tanggal/[*tanggal.sekarang*].

Kemudian alur percakapan menanyakan ramalan cuaca. Dimulai dengan bertanya langsung dengan *intent ramalan cuaca*. Kemudian chatbot menanyakan tentang lokasi sebagai *entity*, setelah itu melanjutkan untuk mengarahkan kueri penyelesaian ke API http://api.openweathermap.org/data/2.5/weather?q=[*lokasi,id*]&APPID=[*your_app_id*]&units=metric, kemudian hasil return api diterjemahkan ke dalam bahasa Indonesia.

Pada table 4.4 menampilkan hasil evaluasi berbagai jenis arsitektur *RNN* sesuai skenario pada subbab 3.3.1 sebelumnya. Perbandingan dilakukan terhadap arsitektur Standar RNN, LSTM, dan GRU dengan mempertimbangkan nilai dimensi filter *RNN* yaitu 32, 64 dan 128 pada masing-masing arsitektur. Di sini terlihat bahwa evaluasi keseluruhan arsitektur tidak terlalu jauh, tetapi nilai kinerja terbaik dicapai oleh arsitektur *RNN GRU* dengan dimensi filter *RNN* 128 dengan skor *Average Precision* (ap) 0.9925, *Average Recall* (ar) 0.9793, *Average F1* (af) 0.9831.

Tabel 4.4 Perbandingan RNN Kinerja Arsitektur

Model	filter dimensi	ap	ar	af	avg_all
GRU	**128**	**0,9793**	**0,9831**	**0,9888**	**0,9925**
SimpleRNN	32	0,9912	0,9843	0,9862	0,9875
SimpleRNN	128	0,9912	0,9843	0,9862	0,9874
LSTM	32	0,9912	0,9843	0,9862	0,9873
GRU	32	0,9912	0,9843	0,9862	0,9872
LSTM	64	0,9843	0,9912	0,9862	0.9872
LSTM	128	0.9843	0.9912	0.9862	0.9862
GRU	64	0.9925	0.9793	0.9831	0.9855
SimpleRNN	64	0.9925	0.9793	0.9831	0.9844

Pada Table 4.4 untuk mempermudah pencarian nilai evaluasi terbaik, dibuat nilai rata-rata secara keseluruhan (*avg_all*) dan didapat urutan berdasarkan nilai tersebut. Didapati nilai terbaik pada arsitektur GRU, dimana arsitektur terbaik ini selanjutnya akan diuji secara ketepatan disaat memberikan respons chat.

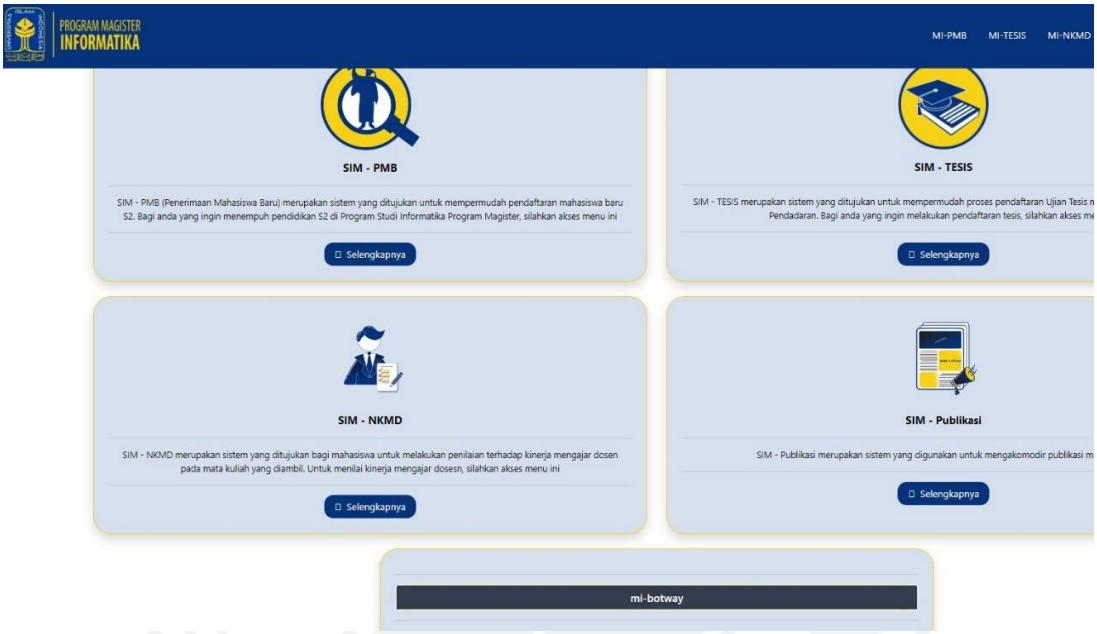
Tabel 4.5 menunjukkan perbandingan *respon chat* antara model *GRU-128* dan *LSTM-32* sebagai model *default model framework RASA*. Model *GRU-128* memberikan respon yang lebih tepat dibandingkan dengan *default model framework RASA LSTM-32* mengacu pada penelitian sebelumnya (Windiatmoko dkk., 2021).

Tabel 4.5 Perbandingan framework default GRU-128 dan LSTM-32

Chat Pengguna	Respons GRU-128	Respons LSTM-32
cek nilai	minta nama lengkap atau NIM nya dong :)	tuliskan nama lengkap atau NIM nya ?
yurio windiatmoko	daftar nilai mata kuliah Analitik Big Data 3 A mata kuliah Analitik Teks 3 A mata kuliah Data Insight 3 A- mata kuliah Jauh Belajar 3 A mata kuliah Metodologi Penelitian 2 B + mata kuliah Pembelajaran Mesin 3 A mata kuliah Pemodelan Kausal 3 A	terima kasih :)

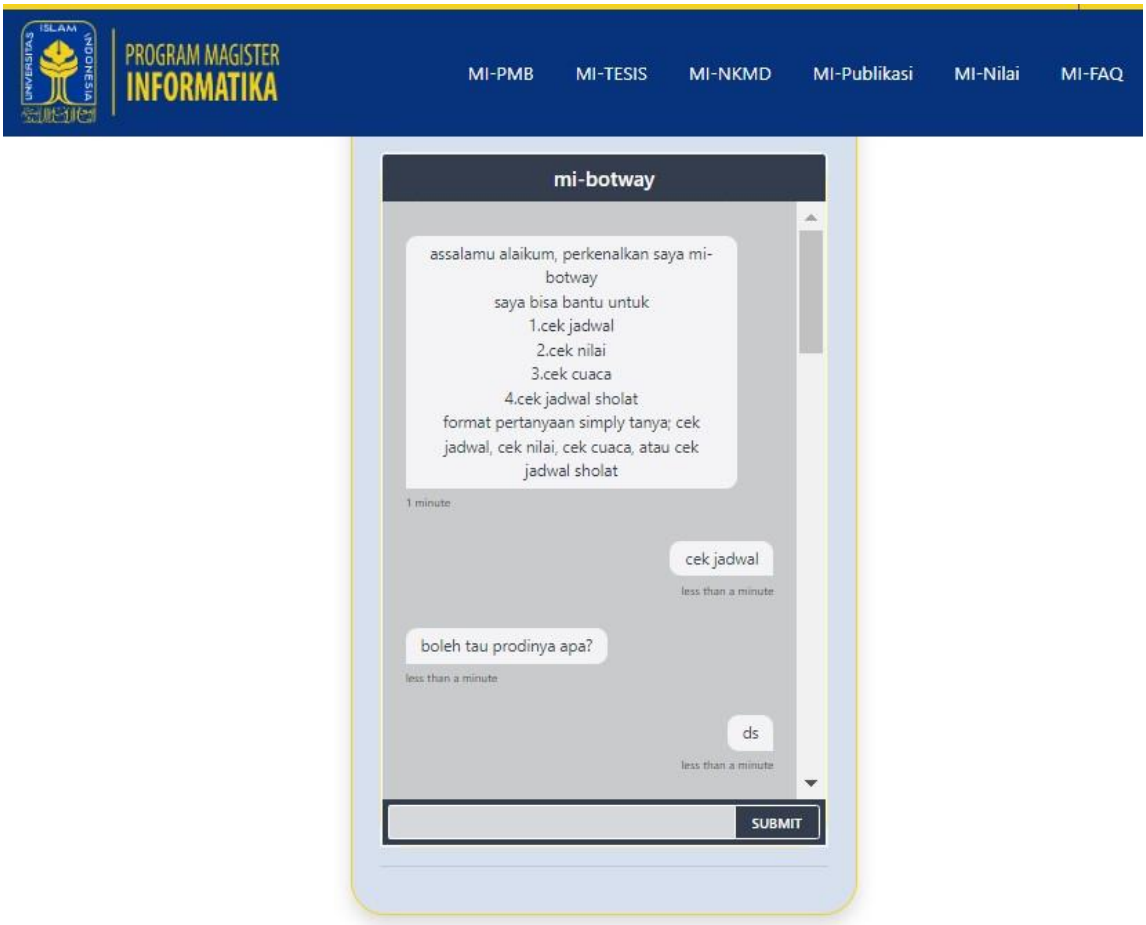
4.2 Implementasi Interface

Implementasi chatbot mi-botway berada pada *web chat*, di beranda *mi-gateway UII* yang telah disematkan, sehingga chatbot ini disebut mi-botway. Posisi modul chatbot ada di badan html beranda mi-gateway bagian bawah sesuai dengan gambar 4.1.



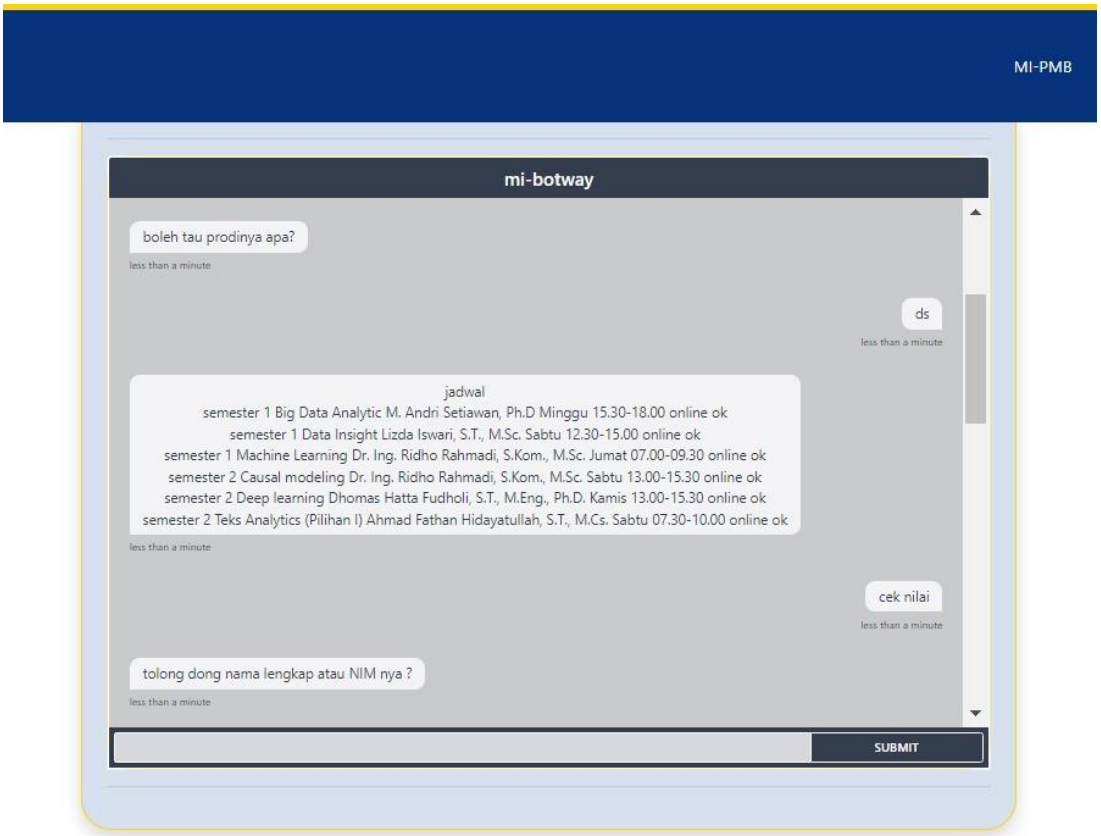
Gambar 4.1 Mi-botway pada beranda *Mi-gateway* bagian bawah

User selanjutnya melakukan *klik* pada *button* mi-botway maka akan muncul welcome message *"assalamu alaikum, perkenalkan saya mi-botway, saya bisa bantu untuk cek jadwal, cek nilai, cek cuaca, cek jadwal sholat, format pertanyaan simply tanya; cek jadwal, cek nilai, cek cuaca, atau cek jadwal sholat"* lalu sesuai dengan tata cara pada *welcome message* tersebut user bertanya dengan format *cek jadwal*, dan mi-botway pun menanyakan balik perihal *prodi*, dan user menjawab *ds* sesuai gambar 4.2.



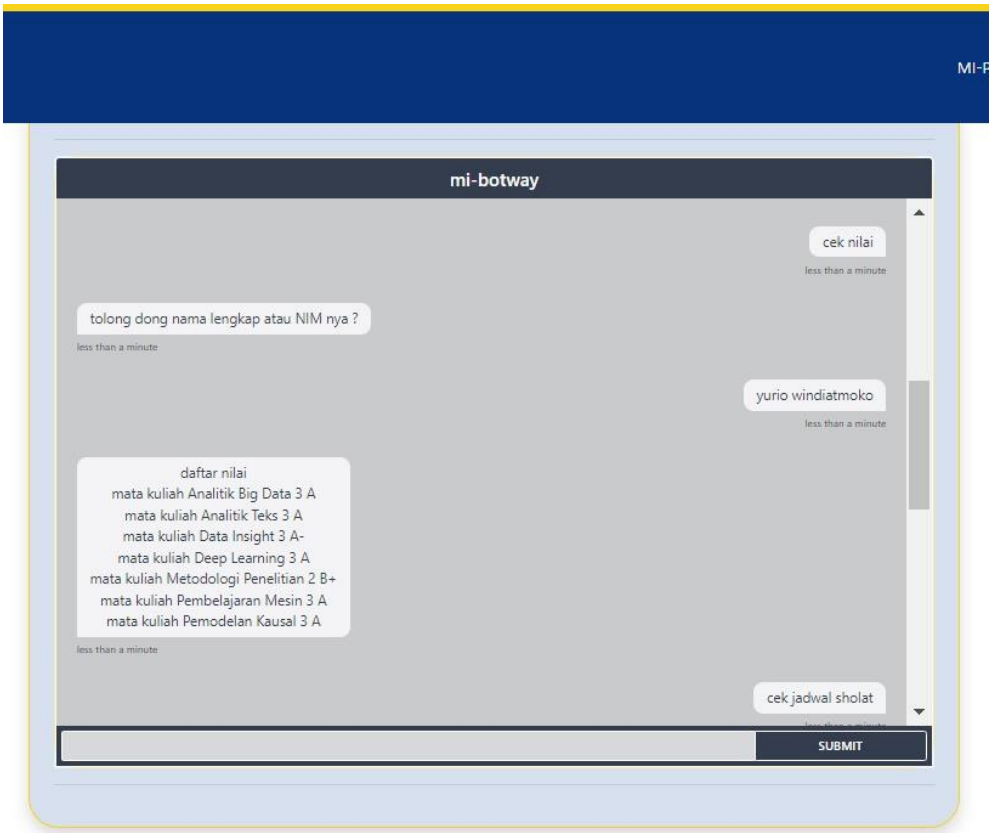
Gambar 4.2 *Welcome message* mi-botway

Maka *key entity ds* pun didapat sebagai *filter query* pada *table jadwal database*, dan dijawab oleh chatbot berupa *return daftar jadwal* kuliah *ds* sesuai gambar 4.3.



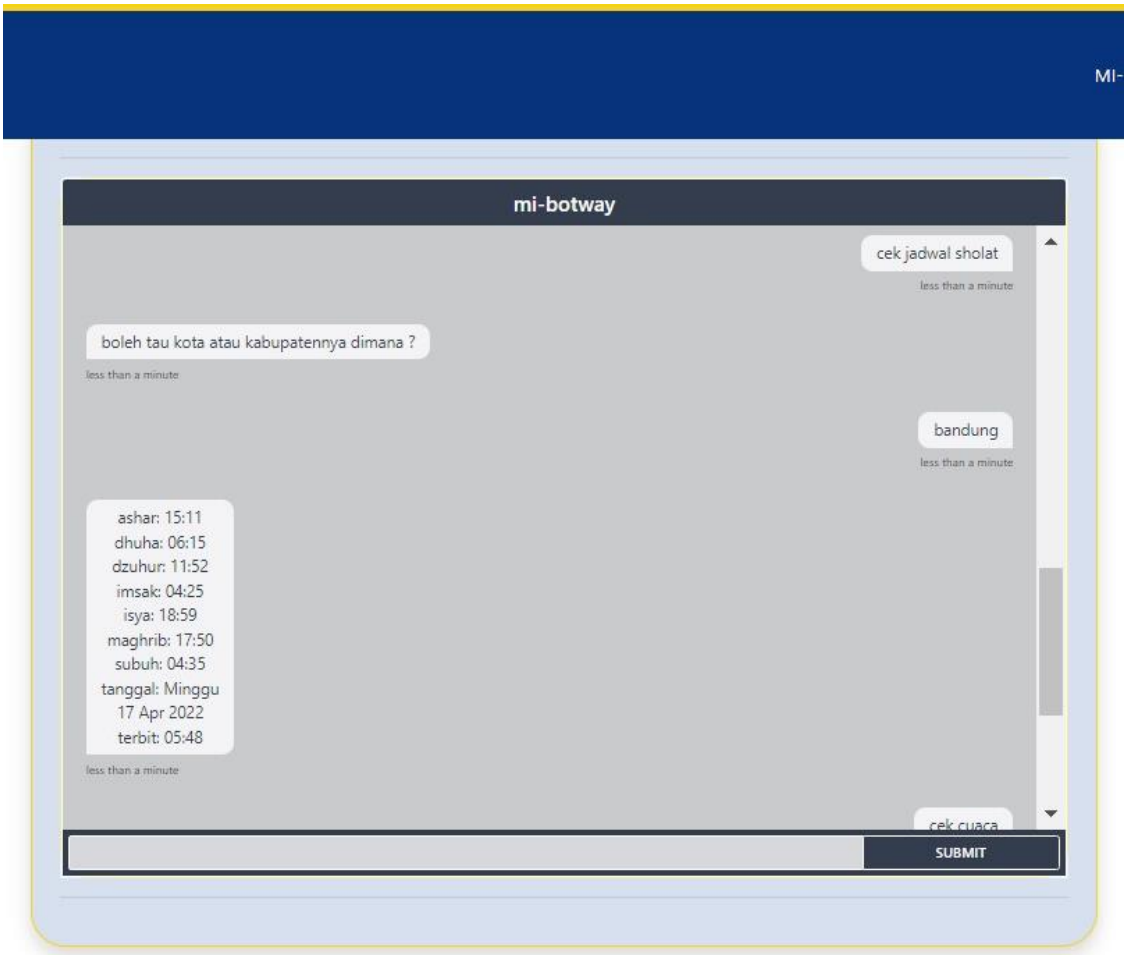
Gambar 4.3 *Return* daftar jadwal kuliah *DS*

Selanjutnya *user* mencoba untuk menyakan perihal nilai, maka format bertanya "*cek nilai* ", dan chatbot pun bertanya balik perihal nama lengkap atau NIM mahasiswa. *User* menjawab "*Yurio windiatmoko* "sebagai nama mahasiswa terkait. Maka key entity *nama mahasiswa* pun didapat sebagai filter query pada *table nilai database*, maka *return* daftar nilai mahasiswa terkait sesuai gambar 4.4.



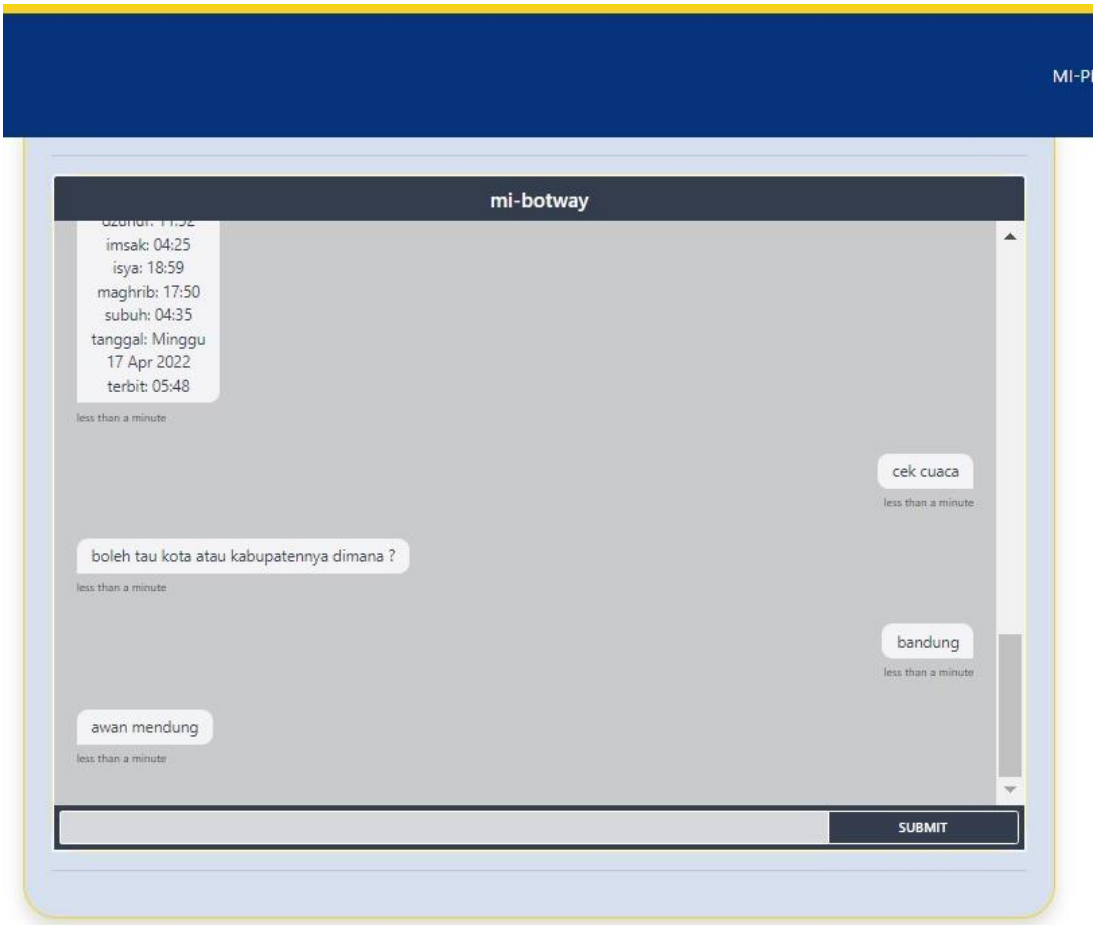
Gambar 4.4 *User* bertanya pada Mi-botway *cek nilai*

Selanjutnya *user* mencoba untuk menyakan perihal jadwal sholat, maka *user* bertanya dengan format "*cek jadwal sholat* ". Maka chatbot pun bertanya balik tentang lokasi atau area *user* terkait jadwal sholat tersebut. *User* menjawab "*bandung* ", maka key *entity bandung* pun didapat sebagai filter query pada *table jadwal_ds database* (daftar sholat). Maka *return* daftar jadwal sholat hari ini di kota bandung tersebut sesuai gambar 4.5.



Gambar 4.5 *User* bertanya tentang jadwal sholat

Kemudian *user* mencoba untuk menyakan perihal prediksi cuaca suatu are tertentu. *User* bertanya dengan format "*cek cuaca* ", maka chatbot bertanya balik perihal area terkait cuaca tersebut. *User* menjawab "*bandung*", maka *key entity bandung* pun didapat sebagai filter query pada *table kota database*, dan dijawab oleh chatbot *return* prediksi cuaca di kota bandung untuk hari ini yaitu *awan pecah* yang menyimpulkan akan adanya hujan sesuai gambar 4.6.



Gambar 4.6 *User* bertanya perihal prediksi cuaca

4.3 Uji Validasi Kegunaan Pada User

Hasil model model chatbot dan platform mi-botway kemudian dipresentasikan kepada mahasiswa *MI UII* yang relevan secara konteks dengan bidang ini, melalui sebuah kuesioner. Melalui kuesioner dimaksudkan untuk mendengar pendapat mereka terhadap platform mi-botway yang telah dibuat. Dari 10 responden dengan rentang usia 25-35 tahun, 100% responden merupakan mahasiswa Magister Informatika UII.

Kuesioner evaluasi model terdiri dari 6 pertanyaan yang dilambangkan dengan K1-K6, dengan pilihan jawaban berkisar antara skor 1-4. Skor 1 mengartikan tidak setuju terhadap pernyataan. Skor 2 mengartikan kurang setuju pernyataan. Skor 3 mengartikan setuju pernyataan, serta skor 4 mengartikan sangat setuju terhadap pernyataan. Serta 1 pertanyaan K7 yang merupakan pendapat saran isian bebas mengenasi fitur *platform* mi-botway.

Selanjutnya 7 pernyataan (K1-K7) pada kuesioner evaluasi model adalah sebagai berikut:

K1: Mi-botway mudah dioperasikan.

K2: Mi-botway menyediakan informasi yang informatif dan mudah dipahami.

K3: Tampilan mi-botway menarik.

K4: Adanya mi-botway membantu saya untuk mendapatkan info jadwal kuliah lebih mudah

K5: Adanya mi-botway membantu saya untuk mendapatkan info nilai akademik lebih mudah

K6: Semoga mi-botway segera diimplementasikan di kampus magister UII

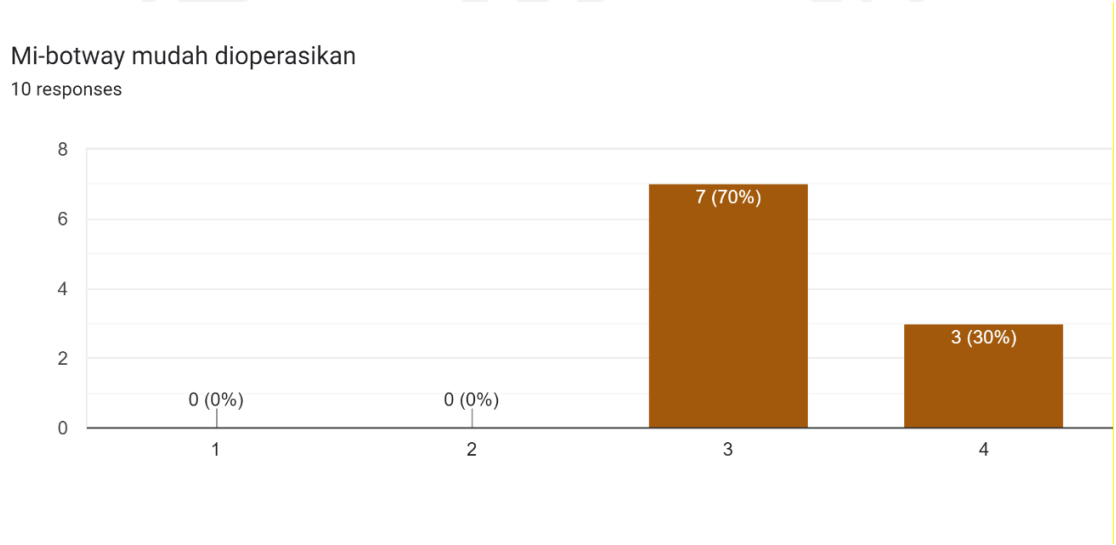
K7: Kira-kira fitur apalagi yang bisa berguna untuk diaplikasikan pada mi-botway

Adapun hasil dari evaluasi platform mi-botway adalah seperti yang terlihat pada Tabel 4.7.

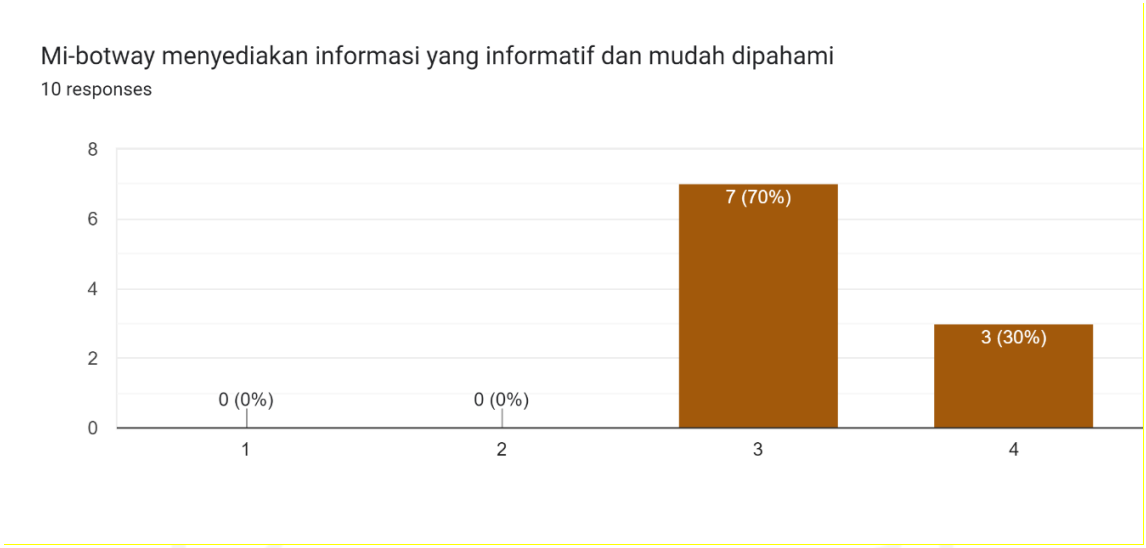
Tabel 4.6 Hasil Evaluasi Kuesioner Terhadap Mi-botway

	Tanggapan Terhadap Mi-botway			
	1	**2**	**3**	**4**
K1	0 (0%)	0 (0%)	7 (70%)	3 (30%)
K2	0 (0%)	0 (0%)	7 (70%)	3 (30%)
K3	0 (0%)	4 (40%)	4 (40%)	2 (20%)
K4	0 (0%)	0 (0%)	7 (70%)	3 (30%)
K5	0 (0%)	0 (0%)	6 (60%)	4 (40%)
K6	0 (0%)	0 (0%)	3 (30%)	7 (70%)

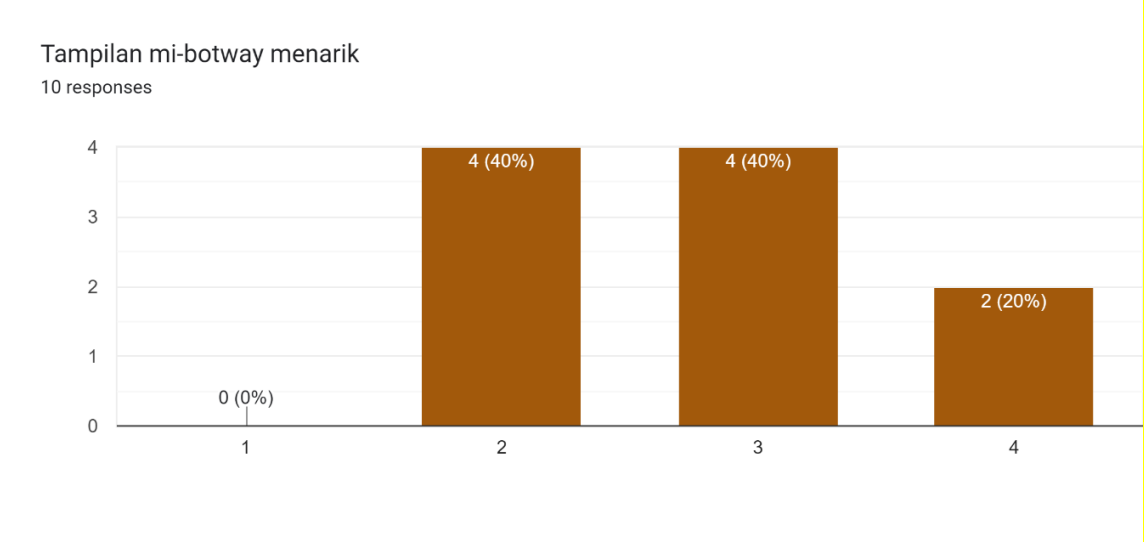
Sedangkan diagram dari masing-masing pernyataan adalah sebagai berikut:



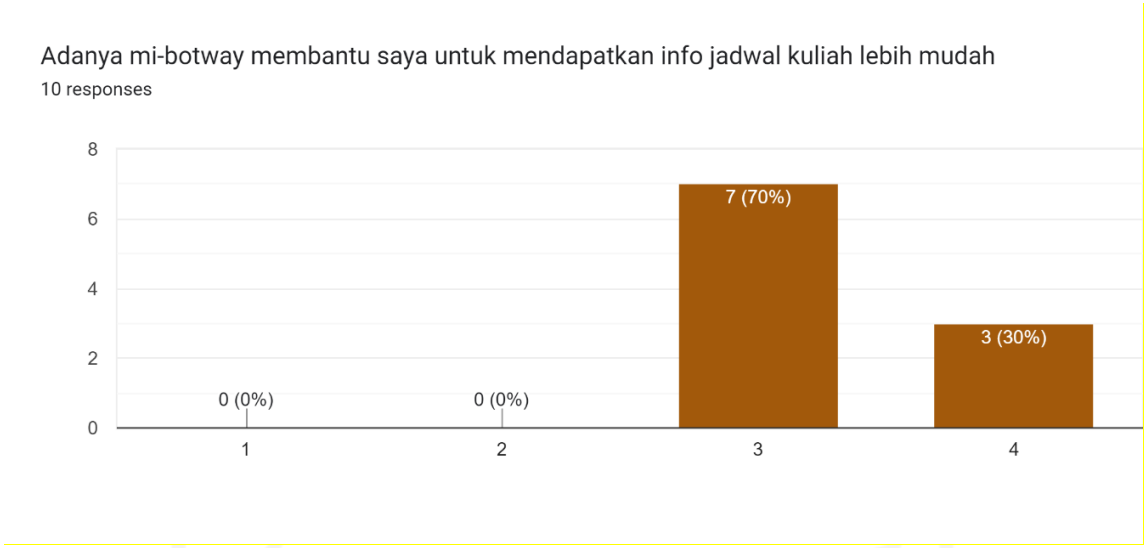
Gambar 4.7 Tanggapan terhadap mi-botway (K1)



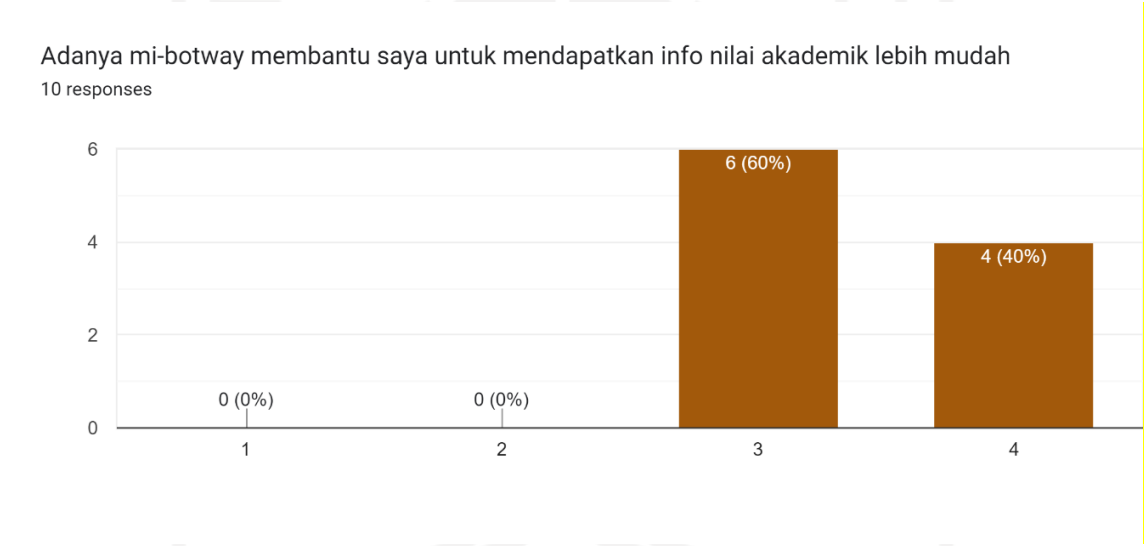
Gambar 4.8 Tanggapan terhadap mi-botway (K2)



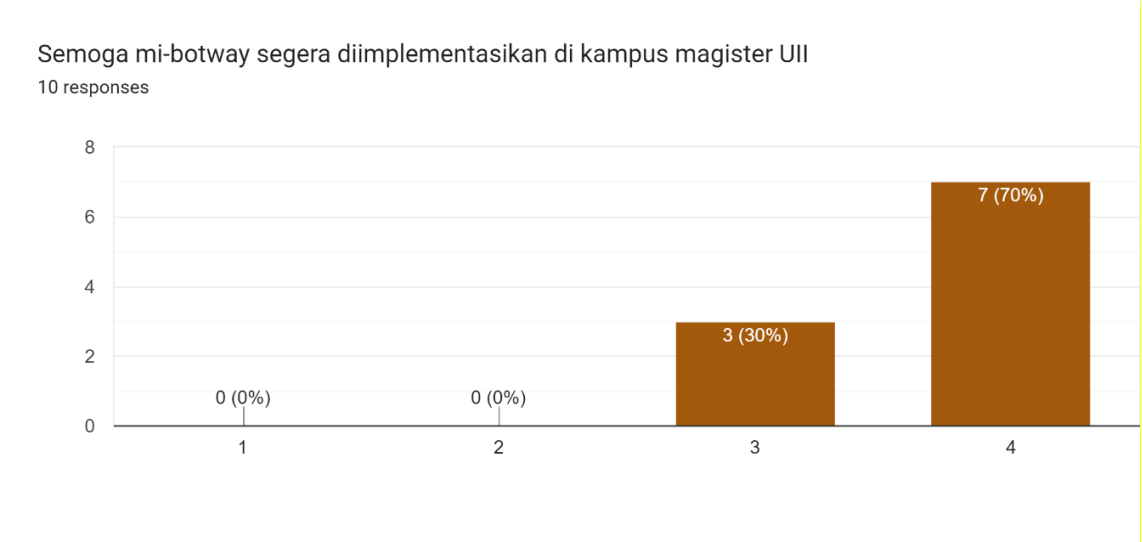
Gambar 4.9 Tanggapan terhadap mi-botway (K3)



Gambar 4.10 Tanggapan terhadap mi-botway (K4)



Gambar 4.11 Tanggapan terhadap mi-botway (K5)



Gambar 4.12 Tanggapan terhadap mi-botway (K6)

Berdasarkan Tabel 4.6 dan histogram pada Gambar 4.7 hingga Gambar 4.12 dapat dilihat bahwa tanggapan responden terhadap mi-botway didominasi oleh responden yang setuju terhadap pernyataan. K1 dengan jumlah responden sebanyak 7 (70%) memberikan tanggapan setuju bahwasanya mi-botway mudah dioperasikan. Pernyataan K2 didominasi oleh responden yang setuju dengan jumlah responden sebanyak 7 (70%), Pernyataan K3 didominasi oleh responden yang setuju dan kurang setuju dengan jumlah responden sebanyak 4 (40%) setuju dan 4(40%) kurang setuju, ini dikarenakan memang tidak banyak dilakukan modifikasi terhadap tampilan *frontend* mi-botway. Pernyataan K4 didominasi oleh responden yang setuju dengan jumlah responden sebanyak 7 (70%). Pernyataan K5 didominasi oleh responden yang setuju dengan jumlah responden sebanyak 6 (60%), dan didominasi oleh responden yang sangat setuju terhadap pernyataan K6 dengan jumlah responden sebanyak 7 (70%).

Secara keseluruhan jumlah responden yang memilih sangat setuju terhadap berbagai tanggapan pernyataan mi-botway ini sebanyak 22 (36.6%), responden yang memilih setuju sebanyak 34 (56.6%), responden yang memilih kurang setuju sebanyak 4 (6,6%), dan responden yang memilih tidak setuju sebanyak 0 (0%).

Adapun kuesioner pertanyaan K7 yang merupakan isian, disini terdapat beberapa saran yang menarik dari rekan-rekan mahasiswa sesuai yang tertera pada table 4.8. Dari sini akan sangat bermanfaat untuk pengembangan chatbot mi-botway selanjutnya. Ataupun menjadikannya saran riset lanjutan bagi peneliti atau mahasiswa yang berminat untuk melanjutkan.

Tabel 4.7 Tanggapan terhadap mi-botway (K7)

Fitur apalagi yang bisa berguna untuk diaplikasikan pada mi-botway
Memahami typo nama, kota, nim, dll
Masih ada daerah yg blm ada untuk prediksi cuaca, mungkin bisa ditambahkan fitur jadwal kuliah kalau dosen kosong
Cek jadwal bimbingan dosen cek fasilitas kampus dan prodi
-
Hilangkan fitur yg tidak perlu. Seperti cek cuaca dan cek jadwal sholat. Saya rasa itu kurang berguna jika diletakkan di chatbot kampus. Saya bisa akses dari hp.
Tambahkan fitur detail tentang mata kuliah. Misal siapa dosennya, seputar mata kuliah, dan requirement apa saja yg diperlukan. Bisa juga tambahkan fitur detail tentang kampus atau dosen, seperti history pendidikan, publikasi, dan riset saat ini (karena ini yg kemungkinan bisa diakses secara publik)
Mungkin ada icon nya, jadi lebih menarik.
Database bisa diperbanyak biar ga itu saja pertanyaannya
Akan lebih bijak jika utk fitur nilai ada authentikasi perihal mahasiswa terkait atau ortu mahasiswa
Cek dosen pembimbing available slot

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dalam penelitian ini, sistem chatbot telah dikembangkan untuk membantu menjawab beberapa pertanyaan yang diajukan oleh berbagai pihak kepada pihak universitas. Karena chatbot belum digunakan secara operasional di kampus, efektivitas dan kemudahan penggunaan bagi pengguna tidak dapat diukur, melainkan diukur melalui kuesioner yang ditujukan kepada mahasiswa magister informatika UII.

Kinerja chatbot yang utama diukur dari ketepatannya dalam menghasilkan kalimat yang dapat dibaca pengguna dengan benar dan tepat. Selain itu, chatbot memiliki waktu respon yang cukup cepat dari model inferensi dalam waktu kurang dari satu detik. Selain itu, hasil evaluasi model chatbot menunjukkan hasil yang cukup baik mendekati nilai sempurna untuk presisi, recall dan f1 yang telah dijelaskan pada bagian sebelumnya. Skor dari angka-angka tersebut semuanya mencapai nilai 1, hanya saja niat salam muslim adalah 0,75 pada skor presisi dan 0,85 pada skor f1, juga pada niat menyapa 0,85 pada skor recall dan 0,92 pada skor f1.

Sedangkan kinerja chatbot berdasarkan hasil evaluasi terhadap para mahasiswa telah memberikan respons positif terhadap pengembangan chatbot mi-botway ini. Ditujukan dari dominasi tanggapan setuju terhadap total semua pernyataan. Adapun tanggapan yang sangat setuju diperlihatkan dari tanggapan terhadap pernyataan mi-botway yang ingin segera diimplementasikan di ranah kampus Magister Informatika UII.

Namun penelitian ini masih dalam tahap pengembangan pertama dengan data simulasi. Penelitian ini juga telah mengimplementasikan model deep learning untuk chatbot, terutama dengan mengimplementasikan intent classifier dan entity recognition pada RASA NLU dan kebijakan dialog pada RASA core, kemudian mengintegrasikannya dengan platform web apps. Sehingga dengan adanya penelitian ini diharapkan dapat menjadi pemantik awal untuk membangun sistim *smart campus* di universitas.

5.2 Saran

Untuk pekerjaan di masa depan, kami sarankan untuk mengembangkan dengan memperluas Data dialog dan lebih banyak fitur untuk membantu otomatisasi tugas di perguruan tinggi, maka dengan evolving dan mengarahkan chatbot sebagai domain pertanyaan terbuka, untuk aplikasi yang lebih luas untuk menjadi 'chatbot saya tahu segalanya'. Ataupun bisa menambahkan fitur-fitur baru sesuai Tabel 4.7 pada bab sebelumnya. Seperti menambahkan fitur untuk mengecek jadwal ketersediaan bimbingan dosen, atau mengecek jadwal availabilitas fasilitas kampus dan prodi.

Daftar Pustaka

Al-fakhri, S., Lutfi, H. U., Wardana, W. K., Munawar, G., Kom, S., Wisnuadhi, B., Si, S., & Kunci, K. (2019). *Aplikasi Chatbot Informasi Kampus Polban Menggunakan Aplikasi LINE Messenger*. *November*, 302–313.

API banghasan, D. (2022). *Api Banghasan*. Api.Banghasan.Com/. https://api.banghasan.com/

Arora, A. S., Bacouel-Jentjens, S., Sepehri, M., & Arora, A. (2020). *Sustainable Innovation: Trends in Marketing and Management* (A. S. Arora, S. Bacouel-Jentjens, M. Sepehri, & A. Arora, Ed.; 1 ed.). Palgrave Pivot Cham. https://doi.org/https://doi.org/10.1007/978-3-030-30421-8

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). *Rasa: Open Source Language Understanding and Dialogue Management*. 1–9.

Csaky, R. K. (2019). *Deep Learning Based Chatbot Models*.

Dadang, W. (2018). *Memahami Kecerdasan Buatan berupa Deep Learning dan Machine Learning*. 06 Februari 2018. https://warstek.com/2018/02/06/deepmachinelearning/

Dictionary, the O. A. L. (2022). *Definition of intent noun*. The Oxford Advanced Learner's Dictionary. https://www.oxfordlearnersdictionaries.com/definition/english/intent_1?q=intent

Docker, O. D. (2022). *Use volumes*. Docs.Docker.Com. https://docs.docker.com/storage/volumes/

Docs of RASA, O. (2022). *Command Line Interface*. Rasa.Com. https://rasa.com/docs/rasa/2.x/command-line-interface/#rasa-init

Docs Stable, P. (2022). *pandas documentation*. Pandas.Pydata.Org/Pandas-Docs/Stable. https://pandas.pydata.org/pandas-docs/stable/

Glass, E. (2022). *How To Install Nginx on Ubuntu 20.04*. Www.Digitalocean.Com. https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04

Goutte, C., & Gaussier, E. (2005). *Ch10_Witnesses[8463].Pdf*. *April*. https://doi.org/10.1007/978-3-540-31865-1

Heidi, E. (2022). *How To Secure Apache with Let's Encrypt on Ubuntu 20.04*. Www.Digitalocean.Com. https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-20-04

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jiao, A. (2020). An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network. *Journal of Physics: Conference Series*, *1487*(1). https://doi.org/10.1088/1742-6596/1487/1/012014

Lafferty, J., & Mccallum, A. (2001). *Conditional Random Fields Probabilistic Models*. *2001*(June), 282–289.

Lutkevich, B. (2022). *DEFINITION framework*. Www.Techtarget.Com. https://www.techtarget.com/whatis/definition/framework

Muhamad Alfarisi, H. (2020). *Mengenal Perbedaan Artificial Intelligence, Machine Learning, Neural Network & Deep Learning (Part III)*. Medium.Com. https://haiqalmuhamadalfarisi.medium.com/mengenal-perbedaan-artificial-intelligence-machine-learning-neural-network-deep-learning-part-bef034a145bd

Official Docs, N. (2022). *nginx documentation*. Nginx.Org. https://nginx.org/en/docs/

Phi, M. (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Towardsdatascience.Com. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Prashant, B. P., Anil, M. S., & Dilip, K. M. (2017). *Online Chatting System for College Enquiry using Knowledgeable Database PARTIAL FULFILLMENT OF THE REQUIREMENTS OF BACHELOR OF ENGINEERING (Computer Shri Chhatrapati Shivajiraje College of*.

STUDI INFORMATIKA PROGRAM MAGISTER, P. (2022). *Mi - Gateway UII*. Migateway-Uii.Id. https://migateway-uii.id/

project package index, T. (2022a). *petl 1.7.11*. Www.Python.Org/Psf-Landing. https://pypi.org/project/petl/

project package index, T. (2022b). *tabula-py 2.5.1*. Www.Python.Org/Psf-Landing. https://pypi.org/project/tabula-py/

Rana, M. (2019). *Digital Commons @ Georgia Southern EagleBot : A Chatbot Based Multi-Tier Question Answering System for Retrieving Answers From Heterogeneous Sources Using BERT*. 1–54.

RASA, O. D. of. (2022a). *Actions*. Rasa.Com. https://rasa.com/docs/rasa/actions/

RASA, O. D. of. (2022b). *Domain*. Rasa.Com. https://rasa.com/docs/rasa/domain/

RASA, O. D. of. (2022c). *Introduction to Rasa Open Source & Rasa Pro*. Rasa.Com. https://rasa.com/docs/rasa/

RASA, O. D. of. (2022d). *Policies*. Rasa.Com. https://rasa.com/docs/rasa/policies/

RASA, O. D. of. (2022e). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#intent

RASA, O. D. of. (2022f). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#entity

RASA, O. D. of. (2022g). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#tracker

RASA, O. D. of. (2022h). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#slot

RASA, O. D. of. (2022i). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#policy

RASA, O. D. of. (2022j). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#action

RASA, O. D. of. (2022k). *Rasa Glossary*. Rasa.Com. https://rasa.com/docs/rasa/glossary/#template--response--utterance

RASA, O. D. of. (2022l). *Training Data Format*. Rasa.Com. https://rasa.com/docs/rasa/training-data-format/

RASA, O. D. of. (2022m). *Your Own Website*. Rasa.Com. https://rasa.com/docs/rasa/connectors/your-own-website/

Rasa Repository Chatbot Widget, O. (2022). *RasaHQ/chatroom*. Github.Com. https://github.com/RasaHQ/chatroom

Santoso, A., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah. *Emitor: Jurnal Teknik Elektro*, *18*(01), 15–21. https://doi.org/10.23917/emitor.v18i01.6235

Setiawan, E. (2022). *Kamus Besar Bahasa Indonesia (KBBI) Kamus versi online/daring (dalam jaringan)*. Kbbi.Web.Id. https://kbbi.web.id/entitas

sheets, P. (2022). *daftar-kabupaten-kota-di-indonesia*. Google Spreadsheets. https://docs.google.com/spreadsheets/d/1QGcHFHSyCkrNhp6FfQQ3LmU6v7F8dSem37pMq5vtFnw/edit#gid=985278376

Sheets, P. (2022). *nilai_kuliah*. Google Spreadsheets. https://docs.google.com/spreadsheets/d/1nFMAQcrPCdMA-fW-CIf3vmL8qM0nlivAJexP1kUOfcE/edit#gid=0

Thakkar, J., Raut, P., Doshi, Y., & Parekh, K. (2018). Erasmus AI Chatbot. *International Journal of Computer Sciences and Engineering*, *6*(10), 498–502. https://doi.org/10.26438/ijcse/v6i10.498502

wikipedia. (2022). *Web platform*. En.Wikipedia.Org. https://en.wikipedia.org/wiki/Web_platform

Wikipedia, the free encyclopedia. (2022a). *Docker (software)*. En.Wikipedia.Org. https://en.wikipedia.org/wiki/Docker_(software)

Wikipedia, the free encyclopedia. (2022b). *Web server*. En.Wikipedia.Org. https://en.wikipedia.org/wiki/Web_server

Wikipedia bahasa Indonesia, ensiklopedia bebas. (2022). *Daftar kota di Indonesia menurut provinsi*. Id.Wikipedia.Org. https://id.wikipedia.org/wiki/Daftar_kota_di_Indonesia_menurut_provinsi

Windiatmoko, Y. (2022a). *Yuriowindiatmoko2401/chatbot-uii-2*. Github.Com. https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/blob/master/pre_utils/pdf_df_load.py

Windiatmoko, Y. (2022b). *Yuriowindiatmoko2401/chatbot-uii-2*. Github.Com. https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/blob/master/pre_utils/csv_df_load

Windiatmoko, Y. (2022c). *Yuriowindiatmoko2401/chatbot-uii-2*. Github.Com. https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/blob/master/pre_utils/get_js_kota_load.py

Windiatmoko, Y. (2022d). *Yuriowindiatmoko2401/chatbot-uii-2*. Github.Com. https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/blob/master/pre_utils/get_js_kota_load.py

Windiatmoko, Y. (2022e). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/data/nlu.md

Windiatmoko, Y. (2022f). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/domain.yml

Windiatmoko, Y. (2022g). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/data/stories.md

Windiatmoko, Y. (2022h). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/credentials.yml

Windiatmoko, Y. (2022i). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/endpoints.yml

Windiatmoko, Y. (2022j). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/frontend/index.html

Windiatmoko, Y. (2022k). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/docker-compose.yml

Windiatmoko, Y. (2022l). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo

Windiatmoko, Y. (2022m). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/actions.py#L36

Windiatmoko, Y. (2022n). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/actions.py#L102

Windiatmoko, Y. (2022o). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/actions.py#L171

Windiatmoko, Y. (2022p). *Yuriowindiatmoko2401/mi-botway-monorepo*. Github.Com. https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo/blob/master/backend/actions.py#L230

Windiatmoko, Y., Rahmadi, R., & Hidayatullah, A. F. (2021). Developing Facebook Chatbot Based on Deep Learning Using RASA Framework for University Enquiries. *IOP Conference Series: Materials Science and Engineering*, *1077*(1), 012060. https://doi.org/10.1088/1757-899x/1077/1/012060

Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., & Weston, J. (2018). StarSpace: Embed all the things! *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 5569–5577.

Yoo, S. Y., & Jeong, O. R. (2020). Auto-growing knowledge graph-based intelligent chatbot using BERT. *ICIC Express Letters*, *14*(1), 67–73. https://doi.org/10.24507/icicel.14.01.67

LAMPIRAN

Lampiran *logs* saat training model GRU 128

https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/blob/master/model_eval_results/GRU_128_rnn_size/model_train_logs.md

```
```bash
(rasa_16) ➜  chatbot-uii-2 git:(master)
rasa train -vv --dump-stories --force --debug

2021-05-16 03:03:41 DEBUG    rasa.skill  - Selected skills:
2021-05-16 03:03:41 DEBUG    pykwalify.compat    - Using yaml library: /home/yurio/anaconda3/envs/rasa_16/lib/python3.7/site-packages/ruamel/yaml/__init__.py

Training Core model...
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Generated trackers will be deduplicated based on their unique last 5 states.
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Number of augmentation rounds is 3
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Starting data generation round 0 ... (with 1 trackers)
Processed Story Blocks: 100%|████████████████████████████████████| 16/16 [00:00<00:00, 5334.57it/s, # trackers=1]
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Finished phase (15 training samples found).
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Data generation rounds finished.
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Found 0 unused checkpoints
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Starting augmentation round 0 ... (with 15 trackers)
Processed Story Blocks: 100%|████████████████████████████████████| 16/16 [00:00<00:00, 538.12it/s, # trackers=14]
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Finished phase (226 training samples found).
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Starting augmentation round 1 ... (with 50 trackers)
Processed Story Blocks: 100%|████████████████████████████████████| 16/16 [00:00<00:00, 210.84it/s, # trackers=34]
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Finished phase (692 training samples found).
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Starting augmentation round 2 ... (with 50 trackers)
Processed Story Blocks: 100%|████████████████████████████████████| 16/16 [00:00<00:00, 204.13it/s, # trackers=31]
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Finished phase (1130 training samples found).
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Found 1130 training trackers.
```
```

```
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - Subsampled to 500 augmented training trackers.
2021-05-16 03:03:41 DEBUG    rasa.core.training.generator  - There are 15 original trackers.
2021-05-16 03:03:41 DEBUG    rasa.core.agent  - Agent trainer got kwargs: {'dump_stories': True}
2021-05-16 03:03:41 DEBUG    rasa.core.featurizers  - Creating states and action examples from collected trackers (by MaxHistoryTrackerFeaturizer(SingleStateFeaturizer))...
Processed trackers: 100%|██████████| 15/15 [00:00<00:00, 4298.91it/s, # actions=52]
2021-05-16 03:03:41 DEBUG    rasa.core.featurizers  - Created 52 action examples.
Processed actions: 52it [00:00, 15127.19it/s, # examples=52]
2021-05-16 03:03:41 DEBUG    rasa.core.policies.memoization  - Memorized 52 unique examples.
2021-05-16 03:03:41 DEBUG    rasa.core.featurizers  - Creating states and action examples from collected trackers (by MaxHistoryTrackerFeaturizer(BinarySingleStateFeaturizer))...
Processed trackers: 100%|██████████| 515/515 [00:00<00:00, 2027.97it/s, # actions=596]
2021-05-16 03:03:41 DEBUG    rasa.core.featurizers  - Created 596 action examples.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking (Masking)            (None, 5, 47)             0
_________________________________________________________________
gru (GRU)                    (None, 128)               67584
_________________________________________________________________
dense (Dense)                (None, 23)                2967
_________________________________________________________________
activation (Activation)      (None, 23)                0
=================================================================
Total params: 70,551
Trainable params: 70,551
Non-trainable params: 0
_________________________________________________________________

2021-05-16 03:03:41 INFO     rasa.core.policies.keras_policy  - Fitting model with 596 total samples and a validation split of 0.1
2021-05-16 03:03:41 DEBUG    rasa.core.policies.policy  - Parameters ignored by `model.fit(...)`: {}
Epoch 1/100
 - 0s - loss: 2.5289 - acc: 0.4446
Epoch 2/100
 - 0s - loss: 1.9795 - acc: 0.4832
Epoch 3/100
 - 0s - loss: 1.6548 - acc: 0.4933
Epoch 4/100
 - 0s - loss: 1.2960 - acc: 0.6309
Epoch 5/100
 - 0s - loss: 1.1541 - acc: 0.7148
Epoch 6/100
 - 0s - loss: 0.9928 - acc: 0.7970
Epoch 7/100
 - 0s - loss: 0.8658 - acc: 0.8507
```

```
Epoch 8/100
 - 0s - loss: 0.7437 - acc: 0.8826
Epoch 9/100
 - 0s - loss: 0.6206 - acc: 0.9077
Epoch 10/100
 - 0s - loss: 0.5605 - acc: 0.8926
Epoch 11/100
 - 0s - loss: 0.4722 - acc: 0.9060
Epoch 12/100
 - 0s - loss: 0.4316 - acc: 0.9077
Epoch 13/100
 - 0s - loss: 0.3678 - acc: 0.9195
Epoch 14/100
 - 0s - loss: 0.3357 - acc: 0.9195
Epoch 15/100
 - 0s - loss: 0.3248 - acc: 0.9128
Epoch 16/100
 - 0s - loss: 0.2801 - acc: 0.9245
Epoch 17/100
 - 0s - loss: 0.2537 - acc: 0.9329
Epoch 18/100
 - 0s - loss: 0.2640 - acc: 0.9211
Epoch 19/100
 - 0s - loss: 0.2380 - acc: 0.9279
Epoch 20/100
 - 0s - loss: 0.2174 - acc: 0.9379
Epoch 21/100
 - 0s - loss: 0.1927 - acc: 0.9430
Epoch 22/100
 - 0s - loss: 0.2118 - acc: 0.9329
Epoch 23/100
 - 0s - loss: 0.1963 - acc: 0.9379
Epoch 24/100
 - 0s - loss: 0.1772 - acc: 0.9430
Epoch 25/100
 - 0s - loss: 0.1877 - acc: 0.9430
Epoch 26/100
 - 0s - loss: 0.1778 - acc: 0.9446
Epoch 27/100
 - 0s - loss: 0.1921 - acc: 0.9480
Epoch 28/100
 - 0s - loss: 0.1689 - acc: 0.9581
Epoch 29/100
 - 0s - loss: 0.1618 - acc: 0.9530
Epoch 30/100
 - 0s - loss: 0.1680 - acc: 0.9480
Epoch 31/100
 - 0s - loss: 0.1175 - acc: 0.9698
Epoch 32/100
 - 0s - loss: 0.1293 - acc: 0.9664
Epoch 33/100
 - 0s - loss: 0.1363 - acc: 0.9530
Epoch 34/100
 - 0s - loss: 0.1322 - acc: 0.9648
Epoch 35/100
```

```
125  - 0s - loss: 0.1375 - acc: 0.9631
126  Epoch 36/100
127  - 0s - loss: 0.1111 - acc: 0.9732
128  Epoch 37/100
129  - 0s - loss: 0.1270 - acc: 0.9648
130  Epoch 38/100
131  - 0s - loss: 0.1286 - acc: 0.9681
132  Epoch 39/100
133  - 0s - loss: 0.1030 - acc: 0.9748
134  Epoch 40/100
135  - 0s - loss: 0.1111 - acc: 0.9715
136  Epoch 41/100
137  - 0s - loss: 0.0914 - acc: 0.9799
138  Epoch 42/100
139  - 0s - loss: 0.0960 - acc: 0.9765
140  Epoch 43/100
141  - 0s - loss: 0.1139 - acc: 0.9732
142  Epoch 44/100
143  - 0s - loss: 0.1020 - acc: 0.9715
144  Epoch 45/100
145  - 0s - loss: 0.0735 - acc: 0.9849
146  Epoch 46/100
147  - 0s - loss: 0.1018 - acc: 0.9732
148  Epoch 47/100
149  - 0s - loss: 0.0880 - acc: 0.9732
150  Epoch 48/100
151  - 0s - loss: 0.1061 - acc: 0.9681
152  Epoch 49/100
153  - 0s - loss: 0.0674 - acc: 0.9849
154  Epoch 50/100
155  - 0s - loss: 0.0882 - acc: 0.9732
156  Epoch 51/100
157  - 0s - loss: 0.0822 - acc: 0.9732
158  Epoch 52/100
159  - 0s - loss: 0.0739 - acc: 0.9883
160  Epoch 53/100
161  - 0s - loss: 0.0869 - acc: 0.9698
162  Epoch 54/100
163  - 0s - loss: 0.0607 - acc: 0.9815
164  Epoch 55/100
165  - 0s - loss: 0.0879 - acc: 0.9765
166  Epoch 56/100
167  - 0s - loss: 0.0741 - acc: 0.9732
168  Epoch 57/100
169  - 0s - loss: 0.0786 - acc: 0.9832
170  Epoch 58/100
171  - 0s - loss: 0.0516 - acc: 0.9832
172  Epoch 59/100
173  - 0s - loss: 0.0571 - acc: 0.9849
174  Epoch 60/100
175  - 0s - loss: 0.0825 - acc: 0.9698
176  Epoch 61/100
177  - 0s - loss: 0.0397 - acc: 0.9899
178  Epoch 62/100
179  - 0s - loss: 0.0560 - acc: 0.9849
```

```
Epoch 63/100
 - 0s - loss: 0.0725 - acc: 0.9732
Epoch 64/100
 - 0s - loss: 0.0647 - acc: 0.9815
Epoch 65/100
 - 0s - loss: 0.0713 - acc: 0.9765
Epoch 66/100
 - 0s - loss: 0.0438 - acc: 0.9899
Epoch 67/100
 - 0s - loss: 0.0551 - acc: 0.9866
Epoch 68/100
 - 0s - loss: 0.0773 - acc: 0.9765
Epoch 69/100
 - 0s - loss: 0.0395 - acc: 0.9866
Epoch 70/100
 - 0s - loss: 0.0663 - acc: 0.9765
Epoch 71/100
 - 0s - loss: 0.0518 - acc: 0.9799
Epoch 72/100
 - 0s - loss: 0.0414 - acc: 0.9916
Epoch 73/100
 - 0s - loss: 0.0773 - acc: 0.9748
Epoch 74/100
 - 0s - loss: 0.0564 - acc: 0.9832
Epoch 75/100
 - 0s - loss: 0.0646 - acc: 0.9782
Epoch 76/100
 - 0s - loss: 0.0683 - acc: 0.9748
Epoch 77/100
 - 0s - loss: 0.0480 - acc: 0.9866
Epoch 78/100
 - 0s - loss: 0.0783 - acc: 0.9732
Epoch 79/100
 - 0s - loss: 0.0514 - acc: 0.9815
Epoch 80/100
 - 0s - loss: 0.0455 - acc: 0.9849
Epoch 81/100
 - 0s - loss: 0.0584 - acc: 0.9799
Epoch 82/100
 - 0s - loss: 0.0559 - acc: 0.9815
Epoch 83/100
 - 0s - loss: 0.0351 - acc: 0.9866
Epoch 84/100
 - 0s - loss: 0.0522 - acc: 0.9849
Epoch 85/100
 - 0s - loss: 0.0500 - acc: 0.9849
Epoch 86/100
 - 0s - loss: 0.0656 - acc: 0.9732
Epoch 87/100
 - 0s - loss: 0.0615 - acc: 0.9815
Epoch 88/100
 - 0s - loss: 0.0415 - acc: 0.9849
Epoch 89/100
 - 0s - loss: 0.0659 - acc: 0.9782
Epoch 90/100
```

```
235    - 0s - loss: 0.0583 - acc: 0.9832
236   Epoch 91/100
237    - 0s - loss: 0.0657 - acc: 0.9799
238   Epoch 92/100
239    - 0s - loss: 0.0339 - acc: 0.9916
240   Epoch 93/100
241    - 0s - loss: 0.0726 - acc: 0.9732
242   Epoch 94/100
243    - 0s - loss: 0.0455 - acc: 0.9849
244   Epoch 95/100
245    - 0s - loss: 0.0459 - acc: 0.9832
246   Epoch 96/100
247    - 0s - loss: 0.0586 - acc: 0.9815
248   Epoch 97/100
249    - 0s - loss: 0.0248 - acc: 0.9883
250   Epoch 98/100
251    - 0s - loss: 0.0681 - acc: 0.9799
252   Epoch 99/100
253    - 0s - loss: 0.0526 - acc: 0.9849
254   Epoch 100/100
255    - 0s - loss: 0.0449 - acc: 0.9832
256
257   2021-05-16 03:03:52 INFO      rasa.core.policies.keras_policy  - Done fitting keras policy
model
258   2021-05-16 03:03:52 INFO     rasa.core.agent  - Persisted model to '/tmp/tmptyczmb5c/core'
259   Core model training completed.
260
261   Training NLU model...
262
263    2021-05-16 03:03:52 INFO      rasa.nlu.training_data.loading  - Training data format of
/tmp/tmpx1uborj_/c4ad5b981a144b59b43b300db332b53a_nlu.md is md
264   2021-05-16 03:03:52 INFO     rasa.nlu.training_data.training_data  - Training data stats:
265
266      - intent examples: 693 (16 distinct intents)
267        - Found intents: 'mood_tidak_senang', 'intent_minta_daftar_nilai', 'menantang_bot',
'intent_prediksi_cuaca',      'intent_minta_jadwal',      'intent_salam',      'intent_confirm_kota',
'intent_berpisah',      'intent_minta_jadwal_saja',      'intent_menyapa',      'terima_kasih',
'intent_confirm_nama_atau_nim',     'mood_baik',     'intent_menolak',     'intent_jadwal_sholat',
'intent_setuju'
268
269      - entity examples: 645 (4 distinct entities)
270      - found entities: 'NIM', 'kota', 'nama', 'konsentrasi'
271
272   2021-05-16 03:03:52 DEBUG     rasa.nlu.training_data.training_data  - Validating training
data...
273      2021-05-16 03:03:52 INFO          rasa.nlu.model    - Starting to train component
WhitespaceTokenizer
274   2021-05-16 03:03:52 INFO     rasa.nlu.model  - Finished training component.
275   2021-05-16 03:03:52 INFO     rasa.nlu.model  - Starting to train component RegexFeaturizer
276   2021-05-16 03:03:52 INFO     rasa.nlu.model  - Finished training component.
277      2021-05-16 03:03:52 INFO          rasa.nlu.model    - Starting to train component
CRFEntityExtractor
278   2021-05-16 03:03:52 INFO     rasa.nlu.model  - Finished training component.
279      2021-05-16 03:03:52 INFO          rasa.nlu.model    - Starting to train component
EntitySynonymMapper
```

```
2021-05-16 03:03:52 INFO rasa.nlu.model - Finished training component.
2021-05-16 03:03:52 INFO rasa.nlu.model - Starting to train component CountVectorsFeaturizer
2021-05-16 03:03:52 INFO rasa.nlu.model - Finished training component.
2021-05-16 03:03:52 INFO rasa.nlu.model - Starting to train component EmbeddingIntentClassifier
2021-05-16 03:03:52 DEBUG rasa.nlu.classifiers.embedding_intent_classifier - Check if num_neg 20 is smaller than number of intents 16, else set num_neg to the number of intents - 1
2021-05-16 03:03:52 INFO rasa.nlu.classifiers.embedding_intent_classifier - Accuracy is updated every 10 epochs
Epochs:
100%|
300/300 [00:10<00:00, 28.38it/s, loss=0.015, acc=0.999]
2021-05-16 03:04:03 INFO rasa.nlu.classifiers.embedding_intent_classifier - Finished training embedding classifier, loss=0.015, train accuracy=0.999
2021-05-16 03:04:03 INFO rasa.nlu.model - Finished training component.
2021-05-16 03:04:03 INFO rasa.nlu.model - Successfully saved model into '/tmp/tmptyczmb5c/nlu'
NLU model training completed.
Your Rasa model is trained and saved at '/home/yurio/kuliah/thesis_mania/mi-botway/chatbot-uii-2/models/20210516-030403.tar.gz'.
```

Lampiran *logging* model lainnya terdapat di link https://github.com/Yuriowindiatmoko2401/chatbot-uii-2/tree/master/model_eval_results di tiap folder model tersebut .

Lampiran *form* kuesioner terdapat pada link https://docs.google.com/forms/d/e/1FAIpQLSea9hy8g_ulzk23tC_vBQIPz-sIoM58QSPG3IwOM28nj9iYkA/viewform

Lampiran hasil kuesioner terdapat pada link https://docs.google.com/spreadsheets/d/1sF00LqFUfC713nZNdNYoGzu0j3ObmbrOtfeTI5eSy8k/edit?usp=sharing

Github utama mi-botway terdapat pada link https://github.com/Yuriowindiatmoko2401/mi-botway-monorepo

Github pengembangan mi-botway awalan terdapat pada link https://github.com/Yuriowindiatmoko2401/chatbot-uii-2