PEMBANGUNAN CHATBOT UNTUK LAYANAN INFORMASI TERKAIT AUTOMATIC FARE COLLECTION (AFC) MENGGUNAKAN FRAMEWORK RASA
(STUDI KASUS PT XYZ)

TUGAS AKHIR

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1) di Program Studi Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sumatera

Oleh:

FUJITA RAHMAH

120140070

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN 2023

DAFTAR ISI

DAFTAR TABEL ..... **iv**

DAFTAR GAMBAR ..... **v**

DAFTAR RUMUS ..... **vi**

DAFTAR LAMPIRAN ..... **vii**

BAB 1 ..... **2**

PENDAHULUAN **2**

1.1. Latar Belakang **2**

1.2 Rumusan Masalah **4**

1.3. Tujuan **4**

1.4 Manfaat **4**

1.5 Batasan Masalah **4**

1.6 Sistematika Penulisan **5**

BAB II **5**

TINJAUAN PUSTAKA **5**

2.1 Tinjauan Pustaka **5**

2.2 Dasar Teori **11**

2.2.1 Chatbot **11**

2.2.2 RASA **14**

2.2.3 Arsitektur RASA **14**

2.2.4 Komponen RASA **15**

2.2.5 DIET Architecture **21**

2.2.6 Transformer **24**

2.2.7 Cara Kerja RASA 28

2.2.8 Kelebihan RASA 28

2.2.9 AFC 29

2.2.10 Telegram 30

2.2.11 Visual Studio Code 30

2.2.12 SQLite3 31

2.2.13 Pengujian Akurasi 33

BAB III 32

METODE PENELITIAN 32

3.1 Alur penelitian 32

3.2 Penjabaran Langkah Penelitian 32

3.2.1 Studi Literatur 32

3.2.2 Pengumpulan Data 33

3.2.3 Persiapan Data 33

3.2.4 Implementasi 36

3.2.5 Evaluasi 41

3.2.6 Analisis dan Pembahasan 41

3.3 Perhitungan Transformer 41

3.4 Alat dan Bahan Tugas Akhir 45

3.4.1 Alat 45

3.4.2 Bahan 46

Daftar Pustaka 47

DAFTAR TABEL

Table 2.1 Rangkuman Penelitian Terdahulu 7
Table 2.2 Tipe Data SQLite3 32
Table 2.3 Pengujian Akurasi 33
Table 3.1 Lexical Featurizer 36
Table 3.2 Count Vectors Featurize.....37

DAFTAR GAMBAR

Gambar 2.1 Cara Kerja Chatbot.....13
Gambar 2.2 Arsitektur RASA.....14
Gambar 2.3 Komponen RASA 15
Gambar 2.4 DIET Arsitektur.....21
Gambar 2.5 Transfomer 24
Gambar 2.6 Attention.....26
Gambar 2.7 Cara Kerja RASA.....28

DAFTAR RUMUS

Rumus 2.1 Attention 25
Rumus 2.2 Positional Encoding 25
Rumus 2.3 Akurasi.....34
Rumus 2.4 Presisi.....34
Rumus 2.5 Recall 34
Rumus 2.6 F1-Score.....34

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Chatbot merupakan program yang dihasilkan dari dampak perkembangan teknologi yang terus maju. Chatbot ialah suatu program computer yang dengannya dapat melakukan percakapan menggunakan metode auditoria atau tekstual [1]. Sistem Chatbot dengan basis AI saat ini banyak digunakan di berbagai sektor industri untuk menunjang kebutuhan pelanggan, bahkan perkembangan dalam dunia pekerjaan [2]. Chatbot dengan basis untuk otomasi proses pelayanan [3]. Chatbot yang dianggap mampu mengurangi biaya customer service dan dapat lebih banyak melayani pelanggan secara realtime [4]. Chatbot berpotensi disebut sebagai bentuk interaksi manusia-mesin yang paling menjanjikan dan canggih [5].

Automatic Fare Collection (AFC) merupakan sistem digitaliasi ticketing, melakukan validasi tanpa kontak dalam hitungan detik. Teknologi ini digunakan oleh PT XYZ sejak awal pengoperasian sebagai fasilitator layanan transportasi perkereta-apian modern. Pemeliharaan fasilitas tiket AFC memerlukan perhatian lebih rinci terutama dari sistem teknologi, sehingga pekerjaan pemeliharaan AFC serta perangkat-perangkat berhubungan dengan sistem tiket mulai dari Passenger Gate, Ticket Vending Machine, Add Value Machine (AVM), Automatic Remaining Value Checking Terminal (ARVCT), dan Ticket Office Machine (TOM) memiliki departemen tersendiri untuk mengurusnya. Walaupun begitu, berbagai permasalahan terkait AFC tidak dapat dihindari [6].

Pembuatan chatbot didasari oleh pemahaman bahwa permasalahan yang dihadapi oleh pihak stasiun dapat berulang atau serupa, sehingga chatbot dapat memberikan solusi yang konsisten dan efisien dalam penanganan permasalahan tersebut. Pembuatan chatbot untuk mengembangkan sebuah layanan yang menjadi gerbang utama dalam menjembatani interaksi dua arah antara pihak stasiun dan pihak maintenance AFC yang dapat meningkatkan efektivitas komunikasi, sehingga jika terdapat permasalahan, pihak stasiun dapat segera mengambil tindakan awal dengan solusi yang diberikan oleh chatbot.

Framework RASA merupakan salah satu *Framework* dalam pembuatan chatbot

yang menerapkan metode deep learning [7]. RASA merupakan solusi yang diadopsi untuk membuat chatbot dengan fokus NLP. RASA dibagi menjadi dua modul, RASA Core dan RASA *Natural Language Understanding* (NLU) [8]. Yang pertama sebagai pengelolaan dialog dan yang kedua bertanggungjawab untuk mengekstraksi entitas dan niat. Chatbot biasa digunakan sebagai sistem layanan informasi.

Alasan kuat menggunakan RASA yakni RASA yang menggunakan DIET architecture dalam pembuatan chatbot, DIET dapat mempelajari hubungan semantik antara kata-kata dan frasa, sehingga dapat mengklasifikasikan input dengan lebih akurat dan lengkap, bahkan untuk input yang mengandung kesalahan dalam pengetikan (disingkat).

Terdapat beberapa penelitian yang telah dilakukan terkait dengan penggunaan RASA *Framework* untuk membangun sebuah chatbot dengan tujuan yang berbeda. Penelitian yang pertama dilakukan oleh Dirko (2021), membahas pembuatan chatbot untuk layanan informasi akademik menggunakan RASA, didapati hasil evaluasi model NLU akurasi sebesar 0.995 dan untuk evaluasi model dialog akurasi 0.7 dan chatbot ini diakses pada web browser sebagai user interface. Kemudian dilakukan penelitian oleh Zein (2022), membahas tentang pembuatan chatbot untuk layanan informasi objek wisata Candi Prambanan, didapati hasil untuk presisi model chatbot sebesar 0.97, recall sebesar 0.94 dan F-1 score sebesar 0.95 dan chatbot ini diakses melalui web browser sebagai user interface. Kemudian juga dilakukan penelitian pembnagunan chatbot menggunakan RASA pada layanan informasi yang dilakukan oleh Ferdian dkk.(2023), dengan membahas pembuatan chatbot memanfaatkan RASA untuk informasi wisata interaktif di Tangerang selatan yang kemudian diintegrasikan dengan telegram [8].

Berdasarkan hasil evaluasi yang telah dilakukan maka penelitian ini akan menggunakan RASA *Framework* untuk membangun chatbot layanan informasi terkait AFC kemudian dihasilkan suatu program nantinya akan diintegrasikan dengan Telegram sebagai user interface. Pemilihan Telegram dikarenakan mengingat pengguna untuk layanan informasi ini hanya digunakan oleh para pegawai di stasiun sehingga lingkup cakupan bukan publik. Suatu penelitian membutuhkan metode uji atau evaluasi produk, yang ditujukan untuk mengetahui

apakah penelitian berhasil dilakukan dengan baik, pada penelitian ini digunakan dua metode evaluasi kinerja yakni *confusion matrix.*

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimana cara membangun layanan informasi berbentuk *Chatbot* menggunakan RASA terkait permasalahan AFC PT XYZ?
2. Bagaimana nilai hasil evaluasi kinerja layanan informasi *chatbot* PT XYZ?

1.3. Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan layanan informasi *chatbot* telegram pada PT XYZ.
2. Mengetahui nilai hasil evaluasi kinerja layanan informasi *chatbot* PT XYZ.

1.4 Manfaat

Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Pihak stasiun dapat dengan mudah dan cepat mendapatkan informasi terkait penanganan awal permasalahan AFC.
2. Penelitian ini diharapkan dapat membuat layanan informasi yang mudah digunakan oleh pihak stasiun dalam mendapatkan informasi terkait permasalahan yang terjadi.
3. Penelitian ini diharapkan dapat menjadi salah satu referensi pengembangan teknologi RASA yang dapat dilakukan oleh pihak yang bekerja di dalam bidang Industri.

1.5 Batasan Masalah

Batasan-batasan permasalahan dalam penelitian ini mencakup hal-hal berikut:

1. Penelitian ini hanya membahas keluhan seputar AFC dengan data yang digunakan berupa keluhan yang sering disampaikan pihak stasiun.

2. Chatbot hanya dapat menjawab keluhan berdasarkan kategori dan kata kunci yang telah disimpan dalam dataset RASA.
3. Pengembangan layanan informasi dibangun menggunakan RASA framework.
4. Chatbot dibuat dalam Visual Studio Code.
5. Layanan informasi ini dapat mengirimkan teks kepada pengguna layanan.
6. Penelitian ini menggunakan Telegram sebagai media dalam melakukan pertukaran pesan.

1.6 Sistematika Penulisan

Adapun sistematika penulisan dalam penelitian adalah sebagai berikut:

1. Bab I Pendahuluan

   Bab ini merangkum beberapa hal yang berkaitan dengan penelitian yakni latar belakang, pengenalan masalah, perumusan pertanyaan penelitian, pendekatan metodologi, sasaran tujuan penelitian, batasan–batasan cakupan penelitian, dampak manfaat hasil penelitian, dan susunan sistematis penulisan.
2. Bab II Landasan Teori

   Bab ini memfokuskan pada pengulasan tinjauan pustaka yang terkait dengan topik yang meliputi chatbot, RASA framework serta metode pengujian.
3. Bab III Metodologi Penelitian

   Bab ini menguraikan tentang metodologi yang akan digunakan dalam penelitian. Ini meliputi pendekatan pembuatan program seperti pembuatan program seperti pembuatan program, pengumpulan dataset, persiapan data, implementasi dan evaluasi terhadap program yang telah dirancang.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pada penelitian ini, peneliti melakukan eksplorasi dan mendapatkan tinjauan pustaka berdasarkan penelitian sebelumnya yang berkaitan dengan pembangunan atau pembuatan chatbot. Peneliti menjadikan penelitian terdahulu sebagai referensi dan landasan perbandingan serta kajian terhadap penelitian yang akan dilakukan. Berikut penelitian-penelitian terdahulu yang berhubungan dengan penelitian ini:

1. Tahun 2022, Laksmi Anindyati, melakukan penelitian terkait dengan Analisis dan Perancangan Aplikasi Chatbot Menggunakan *Framework* RASA dan Sistem Informasi Pemeliharaan Aplikasi (Studi Kasus: Chatbot Penerima Mahasiswa Baru Politeknik Astra). Tujuan penelitian ini yakni sistem informasi yang dibuat dapat membantu meningkatkan efektif dan efisien dalam memberikan informasi terkait administrasi ketika masa penerimaan mahasiswa baru. Metode waterfall. Hasil penelitian tersebut berupa usecase diagram dan juga flowchart. Tahapan Perancangan sistem dilakukan dengan merancang pipeline NLU, arsitektur sistem, perancangan database dalam bentuk physical data model, dan perancangan desain antarmuka (mockup) sistem. Pada penelitian ini hanya mencakup proses analisis dan perancangan [2].
2. Tahun 2023, Aldi Dwi Ferdian, Sariyun Naja Anwar, melakukan penelitian terkait dengan Pengembangan Chatbot untuk Informasi Wisata Interaktif di Tangerang Selatan menggunakan *Framework* Rasa. Tujuan penelitian ini yakni pembuatan chatbot dapat membantu pengunjung mendapatkan informasi aktual dan cepat serta meningkatkan presentase ekonomi di sektor pariwisata dengan masif nya informasi tersebut. Metode waterfall Hasil penelitian Chatbot ini berhasil diimplementasikan sehingga menjadi solusi yang mampu memberikan respon yang relevan dan interaktif terhadap pertanyaan pengguna mengenai informasi wisata di Tangerang Selatan [8].
3. Tahun 2023, Joao Fonseca1 and Fatima Rodrigues, melakukan penelitian terkait dengan ChatBot for student service based on RASA framework.

tujuan penelitian ini yakni membangun chatbot agar memudahkan siswa mendapatkan informasi. Metode prototype Hasil penelitian Chatbot ini menjadi solusi untuk mahasiswa yang ingin mendapatkan informasi seputar administrasi sekolah. Pengujian fungsionalitas dari sistem yang dibangun dinyatakan berhasil dengan angka persenan sebesar 84% [9].

4. Tahun 2022, Zein Hanni Pradana, Hanin Nafi'ah, Raditya Artha Rochmanto, melakukan penelitian terkait dengan Layanan Informasi berbasis Chatbot menggunakan RASA Open-Source Framework di Objek Wisata Candi Prambanan. Tujuan penelitian ini adalah mempercepat pertukaran informasi terkait objek wisata Candi Prambanan oleh pengunjung atau pengguna layanan. Metode CRISP-DM Hasil penelitian Chatbot menjadi solusi untuk sebagai sistem penjawab pertanyaan objek wisata Candi Perambanan. Pengujian fungsionalitas dari sistem yang dibangun dinyatakan berhasil dengan angka persenan akurasi sebesar 91% [10].
5. Tahun 2023, Nicholas Cannavaro, melakukan penelitian terkait dengan Aplikasi Chatbot untuk Layanan Akademik Menggunakan Platform RASA Open Source dengan Fitur Two Stage Fallback. Tujuan penelitian ini adalah aplikasi chatbot dapat meningkatkan efektivitas dan meringankan pekerjaan customer service dalam melayani dan memberikan informasi tanpa batasan kontak fisik. Metode CRISP-DM Hasil penelitian Aplikasi ini dapat membantu untuk meringakan pekerjaan customer service dan memberi kemudahan akses infomasi tanpa batasan kontak fisik. Peneliti melakukan pengujian dan pada iterasi kedua mendapatkan nilai akurasi diatas 90% [11].
6. Tahun 2021, Yurio Windiatmoko, Ridho Rahmadi, Ahmad Fathan Hidayatullah, melakukan penelitian terkait dengan Developing Facebook Chatbot Based on DeepLearning Using RASA *Framework* for University Enquiries. Tujuan penelitian ini adalah memberikan informasi aktual kepada mahasiswa dan calon mahasiswa ketika masa pandemi yang mana sulitnya untuk bertanya secara langsung tatap muka. Metode prototype Hasil penelitian Sistem ini menjadi solusi untuk memberi kemudahan akses

infomasi tanpa batasan kontak fisik oleh mahasiswa. Peneliti melakukan pengujian dan sistem menunjukkan hasil yang cukup baik mendekati skor sempurna pada presisi, recall dan F1 [7].

Table 2.1 Rangkuman Penelitian Terdahulu

No	Nama Penulis, Tahun	Judul	Permasalahan	Metode	Hasil
1	Laksmi Anindyati, 2022	Analisis dan Perancangan Aplikasi Chatbot Menggunakan *Framework* RASA dan Sistem Informasi Pemeliharaan Aplikasi (Studi Kasus: Chatbot Penerima Mahasiswa Baru Politeknik Astra)	Perlunya pengembangan Chatbot Penerimaan Mahasiswa Baru untuk meningkatkan interaksi dan pelayanan kepada calon peserta didik dan mendukung proses penerimaan mahasiswa baru di Politeknik Astra.	Metode Waterfall	Hasil dilakukan proses Analisis kebutuhan sistem RASA dianggap sebagai *Framework* dialog management terbaik kemudian hasil digambarkan menggunakan usecase diagram dan juga flowchart. Tahapan Perancangan sistem dilakukan dengan merancang pipeline NLU, arsitektur sistem, perancangan

No	Nama Penulis, Tahun	Judul	Permasalahan	Metode	Hasil
					database dalam bentuk physical data model, dan perancangan desain antarmuka (mockup) sistem. Pada penelitian ini hanya mencakup proses analisis dan perancangan.
2	Aldi Dwi Ferdian, Sariyun Naja Anwar. 2023	Pengembangan Chatbot untuk Informasi Wisata Interaktif di Tangerang Selatan menggunakan *Framework* Rasa	Para wisatawan menghadapi kendala dalam mendapatkan informasi yang akurat dan lengkap tentang berbagai destinasi wisata di Tangerang Selatan.	Metode Waterfall	Chatbot ini berhasil diimplementasikan sehingga menjadi solusi yang mampu memberikan respon yang relevan dan interaktif terhadap pertanyaan pengguna

No	Nama Penulis, Tahun	Judul	Permasalahan	Metode	Hasil
					mengenai informasi wisata di Tangerang Selatan.
3	Joao Fonseca1 and Fatima Rodrigues. 2023	ChatBot for student service based on RASA framework	Terdapat permasalahan efisiensi waktu klarifikasi pertanyaan siswa dan penurunan tingkay kepuasan siswa terhadap informasi administrasi sekolah.	Metode Prototype	Chatbot menjadi solusi untuk mahasiswa yang ingin mendapatkan informasi seputar administrasi sekolah. Pengujian fungsionalitas dari sistem yang dibangun dinyatakan berhasil dengan angka persenan sebesar 84%
4	Zein Hanni Pradana, Hanin Nafi'ah, Raditya Artha	Layanan Informasi berbasis Chatbot menggunakan RASA	Keterbatasan penyebaran informasi FAQ pariwisata terkhusus candi	Metode CRISP-DM	Chatbot menjadi solusi untuk sebagai sistem penjawab pertanyaan objek wisata Candi Perambanan.

No	Nama Penulis, Tahun	Judul	Permasalahan	Metode	Hasil
	Rochmanto. 2022	Open-Source Framework di Objek Wisata Candi Prambanan	Perambanan karena dampak Pemberlakuan Pembatasan Kegiatan Masyarakat masa Pandemi.		. Pengujian fungsionalitas dari sistem yang dibangun dinyatakan berhasil dengan angka persenan akurasi sebesar 91%
5	Nicholas Cannavaro. 2023	Aplikasi Chatbot untuk Layanan Akademik Menggunakan Platform RASA Open Source dengan Fitur Two Stage Fallback	Pandemi COVID 19 membatasi layanan customer service secara kontak fisik atau tatap muka sehingga mahasiswa sulit mendapatkan informasi.	Metode CRISP-DM yang dipadukan dengan model RAD	Aplikasi ini dapat membantu untuk meringakan pekerjaan customer service dan memberi kemudahan akses infomasi tanpa batasan kontak fisik. Peneliti melakukan pengujian dan pada iterasi kedua mendapatkan

No	Nama Penulis, Tahun	Judul	Permasalahan	Metode	Hasil
					nilai akurasi diatas 90%.
6	Yurio Windiatmoko, Ridho Rahmadi, Ahmad Fathan Hidayatullah. 2021	Developing Facebook Chatbot Based on Deep Learning Using RASA *Framework* for University Enquiries	Pandemi COVID 19 membatasi layanan customer service secara kontak fisik atau tatap muka sehingga mahasiswa sulit mendapatkan informasi aktual kepada customer service .	Metode Prototype	Sistem ini menjadi solusi untuk memberi kemudahan akses infomasi tanpa batasan kontak fisik oleh mahasiswa. Peneliti melakukan pengujian dan sistem menunjukkan hasil yang cukup baik mendekati skor sempurna pada presisi, recall dan F1.

Tabel 2.1 menampilkan ringkasan penelitian sebelumnya yang berkaitan dengan penelitian ini. Ringkasan ini menunjukkan bahwa chatbot di setiap penelitian melakukan tugas yang hampir identik, dengan tugas sebagai pelaku yang secara otomatis bertukar pesan. Sehingga peneliti memilih menggunakan RASA Framework dengan perbedaan dataset yang digunakan.

2.2 Dasar Teori

2.2.1 Chatbot

1. Sejarah Chatbot

Chatbots pertama kali diperkenalkan sebelum komputer pribadi pertama

dikembangkan, khususnya chatbot bernama Eliza yang dikembangkan di MIT Artificial Laboratorium Intelijen Joseph Weizenbaum 1966. Eliza menyamar sebagai psikiater. Eliza memeriksa kata kunci dalam input pengguna dan buat aturan yang mengubah output. Setelah Eliza adalah generasi chatbots Pary Baru Dibuat oleh psikiater Universitas Stanford, Kenneth Colby sebagai upaya untuk mensimulasikan seseorang dengan skizofrenia paranoid. Di atas pada tahun 1995, kemudian mengembangkan A.L.I.C.E (Alicebot) milik Richard Wallace terinspirasi oleh Elisa. Meskipun gagal dalam tes Turing, A.L.I.C.E. selalu salah satu jenis chatbot paling kuat dan dianugerahi Loebner Award, Kompetisi AI Tahunan [12]

Pada dekade pertama abad ke-21, SmarterChild diciptakan oleh ActiveBuddy. Ini adalah upaya pertama untuk membuat chatbot yang tidak hanya bisa menyediakan hiburan tetapi juga untuk memberikan lebih banyak informasi kepada pengguna informasi berguna seperti informasi saham, skor olahraga, kutipan film dan banyak lagi. SmarterChild adalah cikal bakal dari Apple Siri dan Samsung S Voice. Siri adalah asisten pribadi cerdas yang dikembangkan sebagai sebuah proyek berdampingan dengan SRI International dan kemudian diadopsi oleh Apple di iOS 5 untuk iPhone. Itu sudah menjadi bagian dari ekosistem iOS. Siri memungkinkan pengguna terlibat dalam percakapan acak sambil memberikan informasi Informasi berguna mengenai cuaca, saham dan tiket film. Raksasa teknologi seperti Samsung dan Google juga mengikuti jejak Apple dengan mengembangkan asisten AI S Voice dan Google Allo masing-masing. Ada juga asisten rumah yang diaktifkan dengan suara seperti Amazon Alexa dan Google Home, adalah perwakilan lainnya dari chatbot [12]

2. Cara Kerja Chatbot

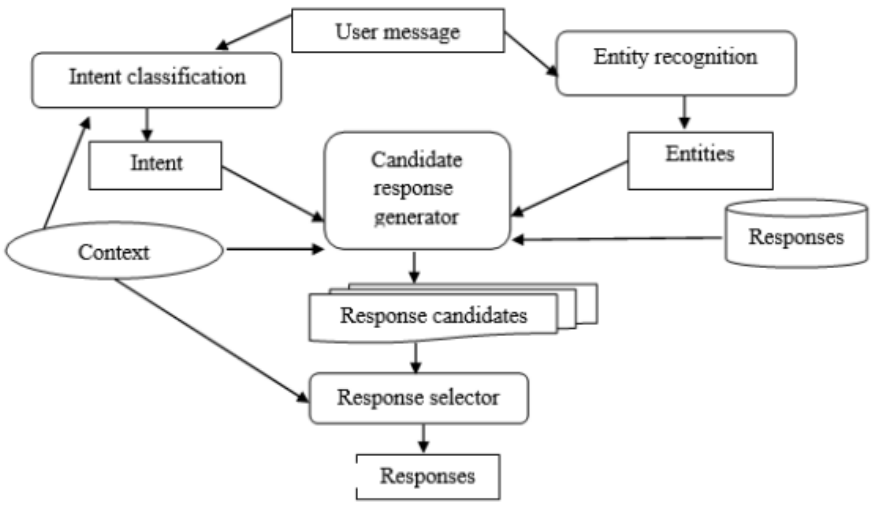
Secara fungsional, chatbot bekerja berdasarkan pengenalan pola dan kumpulan algoritma [13]. Struktur sebuah chatbot, Ada tiga komponen fungsional utama.

1. Intent (Maksud) adalah istilah yang digunakan untuk mendefinisikan memprogram maksud pengguna dari chatbot. Chatbot harus dapat melakukan suatu tindakan berdasarkan “maksud” itu terdeteksi dari pesan pengguna dan dibuat menggunakan teknik yang dikenal sebagai klasifikasi teks dimana tujuan dari program ini adalah mengklasifikasikan

dokumen/kalimat menjadi banyak kelas [12]

2. Entitas (konteks) adalah kata kunci/frasa kunci yang dicari oleh chatbot di postingan pengguna. Entitas ini membantu chatbot untuk mengidentifikasi topik pembicaraan dan memberikan informasi yang ditargetkan untuk pengguna menggunakan teknik yang disebut Entitas Bernama Recognition (NER), adalah metode terkenal untuk mengekstraksi informasi penting dari sebuah teks dan mengklasifikasikannya menjadi ke dalam kategori yang telah ditentukan [12]
3. Responses adalah respon atau jawaban yang diberikan oleh chatbot.

Intents dan Entities (konteks) adalah konsep kunci di balik perilaku chatbot. Intents membuat koneksi antara apa yang dikatakan dan dilakukan pengguna. Apa yang harus dilakukan chatbot? Konteks dinyatakan sebagai nilai string dan digunakan untuk membedakan persyaratan yang mungkin memiliki arti yang berbeda berdasarkan permintaan sebelumnya. Chatbot akan menentukan Respons yang diidentifikasi dengan benar dari repositori kemudian uraikan maksud dan konteks (entitas) yang sesuai. Repositori biasanya terbatas pada blok bangunan linguisstik, meskipun nilai aslinya mungkin berasal dari informasi terstruktur dari mana ia diekstraksi sumber data eksternal (database dan lain-lain). Anda dapat melihat cara kerja chatbot pada Gambar 2.1



Gambar 2.1 Cara Kerja Chatbot

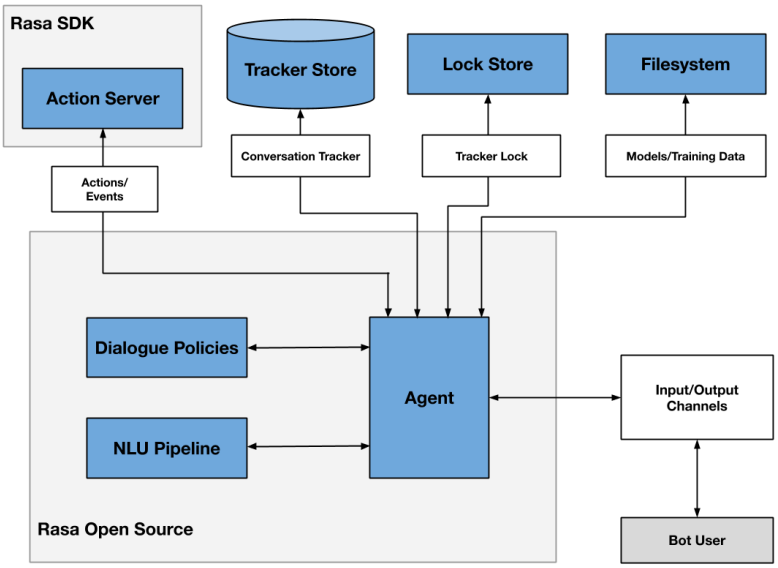
Sumber:

https://journal.ipm2kpe.or.id/index.php/INTECOM/article/download/2704/1599

2.2.2 RASA

Framework RASA adalah sebuah kerangka kerja open source yang digunakan untuk membangun dan mengembangkan chatbot berbasis AI. Kerangka kerja ini menyediakan berbagai fitur untuk mengembangkan chatbot AI yang menggunakan pemahaman bahasa alami (NLU) dan memungkinkan pengguna untuk melatih model dan menambahkan tindakan khusus. Rasa dirancang khusus untuk mempermudah pengembangan, pelatihan, dan implementasi chatbot yang dapat berinteraksi dengan pengguna dalam bahasa alami. Kerangka kerja Rasa dapat digunakan untuk membangun chatbot AI kustom menggunakan Python dan dapat digunakan di berbagai platform.

2.2.3 Arsitektur RASA



Gambar 2.2 Arsitektur RASA

Sumber: https://rasa.com/docs/rasa/arch-overview/

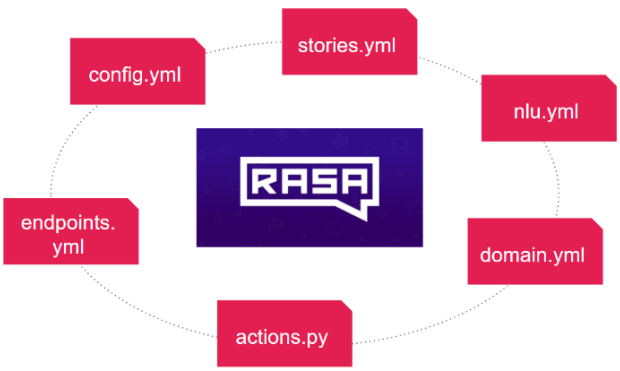
a) NLU adalah bagian yang menangani klasifikasi maksud, ekstraksi entitas, dan pengambilan respons. Ini digambarkan sebagai NLU Pipeline karena memproses ucapan pengguna menggunakan model pipeline yang dilatih NLU.
b) Kebijakan Dialog digambarkan dalam diagram ini. Komponen manajemen dialog memilih apa yang harus dilakukan setelah percakapan berdasarkan konteksnya.
c) Tracker stores, menyimpan data tentang percakapan yang sedang berlangsung antara chatbot dan pengguna.

d) Lock stores, mekanisme ini untuk memastikan bahwa pesan masuk untuk ID chat tertentu diproses dalam urutan yang benar, mengunci percakapan saat pesan diproses secara aktif.

e) Filesystem yang berkaitan dengan model yang dapat disimpan di tempat yang berbeda setelah melakukan training.

f) Server action Rasa adalah komponen yang bertanggung jawab untuk menjalankan tindakan yang dilakukan chatbot sebagai respons terhadap pesan pengguna. Rasa SDK diperlukan untuk mengakses file actions.py pada Rasa karena file ini berisi kode untuk tindakan yang akan dilakukan chatbot. File actions.py adalah file Python yang harus didefinisikan oleh pengembang chatbot. Rasa SDK dokumentasi pindah ke rasa open source untuk menyederhanakan dan mempermudah akses ke dokumentasi dan kode sumber Rasa. Sebelumnya, Rasa SDK tersedia sebagai paket terpisah yang harus diinstal secara terpisah dari Rasa. Hal ini dapat menjadi rumit, terutama bagi pengguna baru.

2.2.4 Komponen RASA



Gambar 2.3 Komponen RASA

Sumber:https://miro.medium.com/v2/resize:fit:1400/1*vodAZhYLQLx_H120xcvw6A.png

Rasa memiliki komponen utama yakni:

1. Natural Language Understanding (NLU)

   Komponen NLU bertanggung jawab untuk memahami maksud pengguna. NLU menggunakan model Natural Language Processing (NLP) untuk menganalisis

pesan pengguna dan menentukan intent pengguna. NLU mencakup config.yml dan nlu.yml

a. Config.yml menentukan komponen dan kebijakan yang akan digunakan model untuk membuat prediksi berdasarkan masukan pengguna. Pada file ini terdapat 3 komponen pembangun utama yakni language, pipeline dan policies.

   Language merupakan kunci untuk menentukan komponen yang digunakan oleh model untuk membuat prediksi NLU. Terkhusus untuk bahasa yang akan digunakan.

   Pipeline juga merupakan kunci untuk menentukan komponen yang digunakan oleh model untuk membuat prediksi NLU. Komponen-komponen membentuk pipeline NLU bekerja secara berurutan untuk memproses input pengguna menjadi output yang terstruktur. Ada komponen untuk preprocessing, feature engineering, intent classification, entity recognition, response selection hingga fallback. Adapun default pipeline yang digunakan RASA:

   1. Preprocessing

   Tahapan awal dalam pengembangan chatbot yang bertujuan untuk mempersiapkan input agar dapat diproses oleh model.

   name: WhitespaceTokenizer

   Memecah kalimat menjadi kata-kata berdasarkan karakter spasi.

   name: RegexFeaturizer

   Untuk membuat representasi vektor pesan pengguna menggunakan ekspresi reguler dan mengekstrak fitur-fitur yang relevan dari kata-kata tersebut.

   2. Feature Engineering

   Tahapan untuk menghasilkan fitur-fitur yang akan digunakan untuk melakukan klasifikasi, pengenalan entitas, atau pembangkitan tanggapan.

   name: LexicalSyntacticFeaturizer

   Membuat fitur leksikal dan sintaksis untuk pesan pengguna guna mendukung ekstraksi entitas

   a) BOS Memeriksa apakah token ada di awal kalimat.
   b) EOS Memeriksa apakah token ada di akhir kalimat.

c) rendah Memeriksa apakah token menggunakan huruf kecil.
d) upper Memeriksa apakah tokennya huruf besar.
e) title Memeriksa apakah token dimulai dengan karakter huruf besar dan semua karakter yang tersisa huruf kecil.
f) digit Memeriksa apakah token hanya berisi angka.
g) prefix5 Ambil lima karakter pertama token.
h) prefix2 Ambil dua karakter pertama token.
i) suffix5 Ambil lima karakter terakhir dari token.
j) suffix3 Ambil tiga karakter terakhir dari token.
k) suffix2 Ambil dua karakter terakhir dari token.
l) suffix1 Ambil karakter terakhir dari token.
m) pos Ambil tag Part-of-Speech dari token (``SpacyTokenizer`` diperlukan).
n) pos2 Ambil dua karakter pertama dari tag Part-of-Speech token (``SpacyTokenizer`` diperlukan).

name: CountVectorsFeaturizer

Membuat representasi bag-of-words dari pesan, intents dan respon.

name: CountVectorsFeaturizer

analyzer: char_wb

Penggunaan karakter n-gram dari teks di dalam baras kata; n-gram di tepi kata diberi spasi.

min_ngram: 1

Pemberian batas bawah untuk n-gram.

max_ngram: 4

Pemberian batas atas untuk n-gram.

3. Intent Classification

Tahapan untuk menentukan maksud dari inputan user.

name: DIETClassifier

Dual Intent Entity Transformer (DIET) digunakan untuk klasifikasi maksud dan ekstraksi entitas.

epochs: 100

Parameter menetapkan berapa kali algoritma akan melihat data pelatihan.

4. Entity Recognition

Tahapan untuk mengidentifikasi entitas-entitas yang relevan dalam input user.

name: EntitySynonymMapper

Memetakan nilai entitas sinonim ke nilai yang sama.

5. Response Selection

Tahapan untuk memilih tanggapan yang paling sesuai untuk diberikan kepada user berdasarkan maksud yang telah diklasifikasikan.

name: ResponseSelector

Penyeleksi mempredikasi respons bot dari serangkaian respons kandidat.

epochs: 100

Parameter menetapkan berapa kali algoritma akan melihat data pelatihan

6. Fallback

Tahapan yang digunakan ketika model tidak dapat menentukan maksud dari input user dengan tingkat kepercayaan yang cukup tinggi.

name: FallbackClassifier

Mengklasifikasikan pesan dengan intent *nlu_fallback* jika skor klasifikasi intent NLU bersifat ambigu. Keyakinannya diatur sama dengan *fallback threshold*

threshold: 0.3

Parameter menetapkan ambang batas untuk memprediksi intent *nlu_fallback*

ambiguity_threshold: 0.1

Memprediksi intent *nlu_fallback* jika perbedaan skor keyakinan untuk dua maksud dengan peringkat tertinggi lebih kecil dari *ambiguity_threshold*

Default tersebut dapat di customize sesuai kebutuhan pengembang.

b. Nlu.yml menyimpan informasi terstruktur tentang pesan pengguna, terdiri dari contoh ucapan pengguna yang dikategorikan berdasarkan intent.

2. Dialogue Management

Dialogue management dalam kerangka kerja Rasa telah berevolusi dari Rasa Core sebelumnya menjadi modul baru dan lebih baik yang memanfaatkan kebijakan untuk memberikan lebih banyak fleksibilitas dan kontrol atas aliran percakapan, sekaligus merangkul kekuatan model bahasa yang besar untuk meningkatkan kemampuan AI percakapan. Komponen Dialogue Management bertanggung jawab untuk mengelola percakapan dengan pengguna. Dialogue Manager menggunakan model NLU untuk memahami maksud pengguna dan model Dialogue Management untuk menentukan respons yang tepat. Mencakup

a. Domain.yml menentukan semesta tempat asisten beroperasi. Ini menentukan maksud, entitas, slot, respons, formulir, dan tindakan yang harus diketahui bot. Ini juga menentukan konfigurasi untuk sesi percakapan.
b. Stories.yml jenis data pelatihan yang digunakan untuk melatih model manajemen dialog asisten. Cerita dapat digunakan untuk melatih model yang mampu menggeneralisasi jalur percakapan yang tidak terlihat.
c. Endpoints.yml digunakan untuk menentukan titik akhir HTTP yang akan digunakan oleh chatbot untuk berkomunikasi dengan server action Rasa. Titik akhir HTTP ini adalah alamat server action Rasa dan port yang digunakan oleh server action Rasa untuk mendengarkan permintaan. Endpoint YML juga dapat digunakan untuk menentukan header HTTP yang akan dikirim oleh chatbot ke server action Rasa. Header HTTP ini dapat digunakan untuk memberikan informasi tambahan tentang pesan yang dikirim oleh chatbot, seperti intent pengguna atau konteks percakapan.
d. Rules.yml pada Rasa adalah file YAML yang berisi aturan yang digunakan oleh Dialogue Manager untuk menentukan respons yang tepat.
e. Config.yml menentukan komponen dan kebijakan yang akan digunakan model untuk membuat prediksi berdasarkan masukan pengguna. Pada file ini terdapat 3 komponen pembangun utama yakni language, pipeline dan policies.

Policies untuk mendefinisikan policies atau kebijakan yang digunakan model untuk prediksi aksi selanjutnya. Adapun default policies yang digunakan RASA:

a. name: MemoizationPolicy

Mengingat *story* dari *train data* lalu mencocokkan percakapan yang berlangsung dengan *story* di *stories.yml file.*

name: TEDPolicy

Kebijakan Transformer Embedding Dialogue (TED) adalah arsitektur *multitask* untuk prediksi tindakan selanjutnya dan pengenalan entitas.

max_history: 5

Parameter ini mengontrol seberapa banyak riwayat dialog yang dilihat model untuk memutuskan tindakan mana yang harus diambil selanjutnya.

epochs: 10

Parameter ini menetapkan berapa kali algoritme akan melihat data pelatihan.

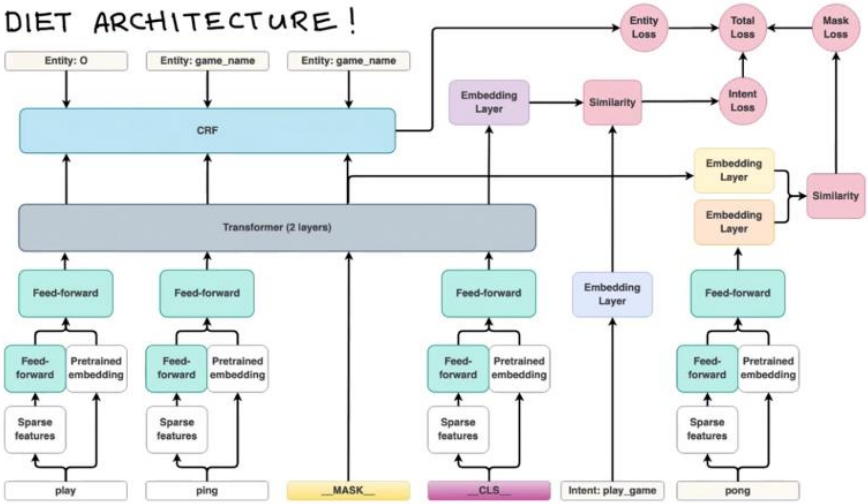
Default tersebut dapat di customize sesuai kebutuhan pengembang.

3. Actions

Komponen Actions bertanggung jawab untuk melakukan tindakan yang diperlukan untuk menanggapi pengguna. Actions dapat berupa tindakan yang sederhana, seperti mengirimkan tanggapan teks, atau tindakan yang lebih kompleks, seperti mengakses data dari sistem back-end. Mencakup

a. Actions.py setiap pesan pengguna, model akan memprediksi tindakan yang harus dilakukan asisten selanjutnya.

2.2.5 DIET Architecture



Gambar 2.4 DIET Arsitektur

Sumber: https://towardsdatascience.com/day-101-of-nlp365-in-depth-study-of-rasas-diet-architecture-3cdc10601599

Berikut adalah penjelasan lengkap mengenai arsitektur DIET (Dual Intent and Entity Transformer) pada RASA NLU dengan menambahkan informasi terkait design decision:

Pada penjelasan ini akan dibagi beberapa bagian penjelasan dengan berlandaskan jenis loss, yakni loss entity, loss intent, loss mask dan diakhiri dengan loss total.

Loss Entity, secara garis besar, data teks input token (kata) akan melewati 2 jalur utama dalam DIET architecture:

1. Pretrained Word Embeddings, token words di-embed menjadi vektor numerik menggunakan pretrained embeddings seperti BERT, GloVe, fastText, dll. Pretrained embeddings ini telah dilatih pada korpus teks besar sebelumnya sehingga dapat menangkap relasi semantik antar kata dengan baik. Output berupa word vector untuk setiap token kata.
2. Sparse Feature Extractor. Menghasilkan sparse vector features dari input kata, seperti encoding kata, n-grams, dll yang dominan nilai 0-nya (sparse). Bersifat lebih spesifik pada domain aplikasi chatbot.

Kedua vektor dari jalur 1 dan 2 kemudian digabungkan (concatenate) dan dimasukkan ke feed-forward neural network layer dengan fungsi aktivasi.

Selanjutnya hasil vektor gabungan ini diumpankan ke dalam beberapa lapisan

Transformer Encoder untuk menangkap konteks kalimat lebih baik. Didalam Transformer terjadi mekanisme attention antar kata yang saling mempengaruhi. Mekanisme attention ini memungkinkan blok transformer untuk mempelajari representasi kontekstual yang lebih kompleks.

Keluaran akhir Transformer kemudian masuk ke CRF (Conditional Random Field) layer untuk melakukan entity extraction. CRF mengklasifikasikan token kata mana yang termasuk dalam entitas tertentu. Blok CRF dapat memberikan akurasi yang lebih tinggi daripada algoritma klasifikasi yang lebih sederhana. Kemudian terjadi loss entity.

Loss Intent, yang dimulai dari singkatan CLS, ide dibaliknya adalah sesuatu yang meringkas keseluruhan ucapan, bukan hanya token tapi keseluruhan kalimat dirangkum, dan ini bergantung dengan apa jenis pre-tained embedding yang diambil. Langkah yang terjadi sama hal nya dengan loss entity hanya saja berbeda pada objek yang diolah yakni yang berupa summarize sentences.

Sparse semua feature dikumpulkan kemudian diinputkan sebagai penjumlahan sehingga semua variabel sparse akan dikodekan satu halt. Lanjut masuk ke lapisan Feed forward. Dan kedua hasil dari proses pre-tained dan sparse di concatenate masuk ke dalam lapisan feed forward. Kemudian diumpankan ke dalam beberapa lapisan Transformer Encoder untuk menangkap konteks kalimat lebih baik. Didalam Transformer terjadi mekanisme attention antar kata yang saling mempengaruhi. Mekanisme attention ini memungkinkan blok transformer untuk mempelajari representasi kontekstual yang lebih kompleks. Selanjutnya hasil dilakukan embedding layer.

Adapun dalam proses loss intent melibatkan pemrosesan intent yang dicek yang masuk kedalam embedding layer dan diakhiri compare dengan hasil CLS untuk di cek similarity antar kedua intent sehingga menghasilkan loss intent jika intent yang dibandingkan sesuai atau sama.

Loss mask, berangkat dari tujuan akhir model yakni untuk memastikan bahwa kita dapat mengklasifikasikan intent dan entity dengan cukup baik namun dalam upaya untuk menjadikan model menjadi model bahasa umum yang merupakan taktik regularisasi, maka ditambahkan gagasan untuk mask kata-kata selama pelatihan. Ide dibalik ini ialah memilih secara acak untuk menghilangkan suatu kata dari

sebuah kalimat kemudian dijadikan mask sehingga menjadi tujuan dari algoritma untuk memprediksi kata apa yang sebenarnya, dan bukan token mask namun berupa objek sehingga tidak ada blok atau proses apapun.

Mask yang melewati tranformer lalu embedding layer, idenya adalah transformer dan embedding layer yang dimiliki dapatkah mereka memprediksi kata apa yang seharusnya di mask, dan jika dapat memprediksi maka model memiliki kualitas yang baik dan akan adanya loss mask.

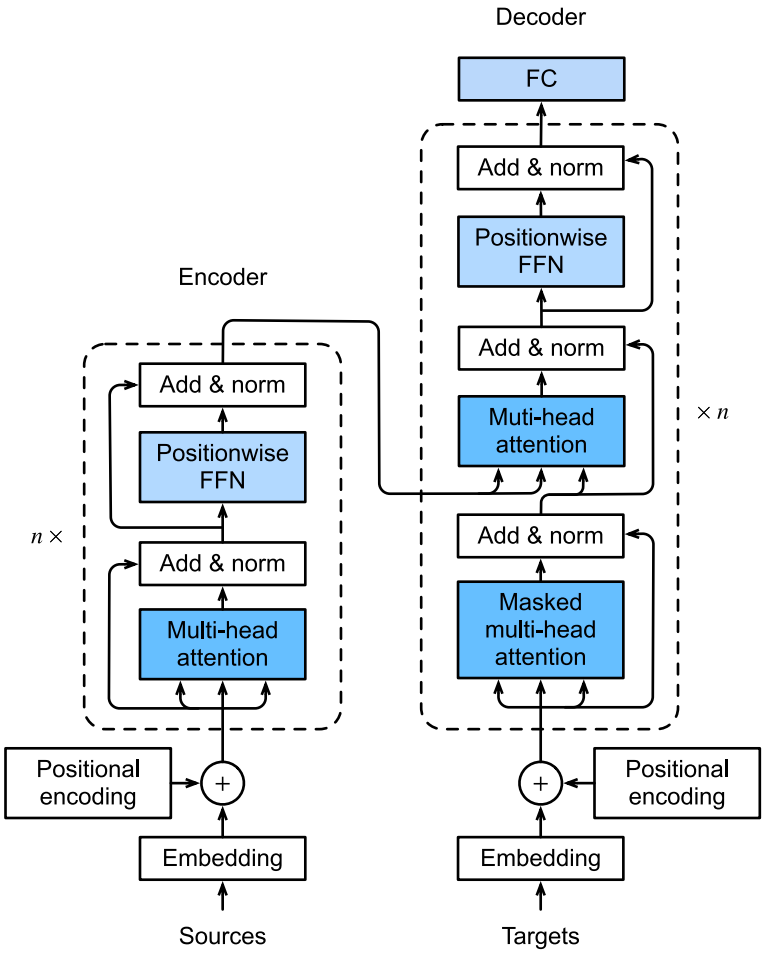
Adapun pemrosesan token yang hilang, pemrosesan sama dengan pemrosesan token sebelumnya hanya saja setelah melewati feed forward lalu akan masuk ke embedding layer dan diakhiri dengan cek similarity antara mask dengan token yang sebenarnya, jika menghasilkan kesamaan maka terjadi loss mask.

Beberapa desain decision:

1. Penggunaan FF layer setelah embeddings dan sparse features.
   Memungkinkan koreksi minor pada embeddings agar lebih sesuai domain.
2. Blok token layer digunakan di banyak tempat: input tokens, masked tokens, intent tokens.
   a. Tujuannya agar dapat belajar representasi token yang umum dan baik.
   b. Karena menerima gradient update dari berbagai loss function.
3. Transformer Encoder
   a. Input vectors saling mempengaruhi lewat attention untuk menangkap konteks.
   b. Output vectors memiliki tambahan konteks.
   c. Memungkinkan alokasi loss secara lebih tepat ke token.
4. Masking
   a. Regularisasi model agar lebih umum dalam domain bahasa.
   b. Dilakukan dengan menghilangkan kata secara acak dan memprediksinya.

Secara keseluruhan DIET bertujuan meminimalkan banyak loss function seperti loss entity, loss intent, loss masked words, dll agar dapat melakukan intent classification dan entity dengan akurasi tinggi. Namun tetap bersifat umum pada domain bahasa.

2.2.6 Transformer



Gambar 2.5 Transfomer

Sumber: https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html

Alur kerja transformer terdiri dari dua tahap utama, yakni tahap encoding dan tahap decoding. Kemudian komponen inti dari transformer ialah penggunaan attention. Fungsi attention dapat digambarkan sebagai pemetaan query dan sekumpulan pasangan key-value ke sebuah output, di mana query, key, value, dan output adalah vektor. Output dihitung sebagai weighted sum dari values, di mana bobot yang ditetapkan untuk setiap value dihitung oleh fungsi kompatibilitas dari query dengan key yang sesuai.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Rumus 2.1 Attention

Sumber:https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Keterangan:

Q = query

K = key

V = value

K^T = key transpose

d_k = dimensi representasi key

Berikut ini penjelasan lebih rinci mengenai alur tahapan dari input hingga output pada arsitektur Transformer:

Tahap encoding

1. Input Text

   Kalimat teks sebagai input user.

2. Tokenization

   Input text dipecah menjadi token (kata).

3. Input Embedding

   Setiap token diubah ke vektor angka dengan embedding layer.

4. Add Positional Encoding

   Menambahkan encoding posisi ke vektor kata dan memberikan informasi urutan kata ke model

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Rumus 2.2 Positional Encoding

Sumber:https://stackoverflow.com/questions/63295569/explanation-about-i-2-in-positional-encoding-in-tensorflow-tutorial-about-trans

Keterangan:

PE = Positional Encoding

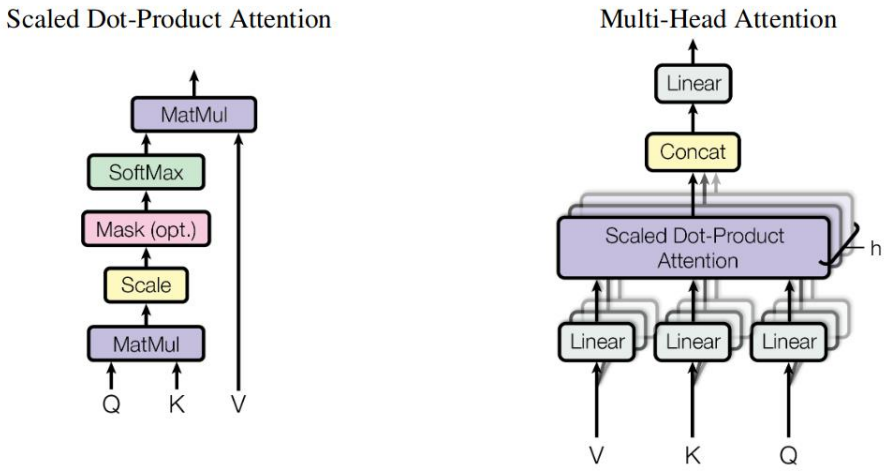
pos = posisi kata dalam kalimat

i = indeks embedding

d_{model} = panjang representasi kata

Hal ini dilakukan dengan menggunakan pengkodean posisi. Menggunakan fungsi sinus dan kosinus. Kosinus untuk setiap urutan ganjil untuk membuat vektor dan sinus untuk urutan yang genap untuk membuat vektor lalu menambahkan vektor-vektor tersebut ke vektor penyematannya yang sesuai.

5. Encoder



Gambar 2.6 Attention

Sumber:https://www.linkedin.com/pulse/unpacking-query-key-value-transformers-analogy-database-mohamed-nabil/

Vektor kata melewati beberapa lapis encoder identik. Terjadi:

1. Multi-headed self-attention layer, menghitung hubungan antar kata. Mekanisme attention bekerja dengan menghitung skor attention untuk setiap token dalam urutan input. Skor ini lalu digunakan untuk menentukan seberapa besar pengaruh token tersebut terhadap token output yang akan dihasilkan.
   a) Input vektor kata melewati 3 linear layer sebagai query, key dan value
   b) Query dan key dilakukan dot product untuk menghitung similarity score
   c) Score diskalakan dan softmax untuk mendapat attention weight

d) Attention weight dikalikan dengan value untuk mendapat output vektor

2. Point wise feed forward layer, mengolah vektor lebih lanjut
   a) Input vektor melewati 2 linear layer dengan activation RELU diantaranya
   b) Output vektor dari linear layer kedua
3. Residual connection dan normalization, stabilisasi jaringan
   a) Output FF layer ditambah dengan input awal encoder (residual)
   b) Kemudian melalui normalization layer

Proses di atas dilakukan beberapa kali pada setiap token kata input. Output akhir encoder berupa representasi vektor untuk setiap token dengan konteks kalimat.

Tahapan Decoding

Pada setiap langkah decoding, model akan menerima representasi dari token-token output yang telah dihasilkan sebelumnya sebagai inputan. Representasi digabungkan dengan representasi kontekstual dari urutan input yang telah dipelajari pada tahapan encoding.

6. Decoder
   a) Bersifat autoregresif, menghasilkan kata per kata
   b) Juga terdiri dari beberapa lapis decoder identik
   c) Output encoder dan decoder saling dihubungkan

1. Masked multi-headed self-attention

   Mirip dengan encoder, beda pada masking untuk mencegah melihat kata masa depan. Metode masking ini bekerja dengan menambahkan nilai negatif tak terhingga ke skor perhatian untuk token output di masa depan.

2. Multi-headed encoder-decoder attention

   Perhatian antara output encoder dan decoder. Prosesnya sama seperti yang dilakukan pada tahap encoding, yaitu dengan menghitung skor perhatian untuk setiap token dalam urutan input. Digunakan oleh decoder untuk memutuskan bagian mana dari encoder yang relevan

3. Point wise feed forward

   Sama seperti di encoder

4. Residual connection dan normalization

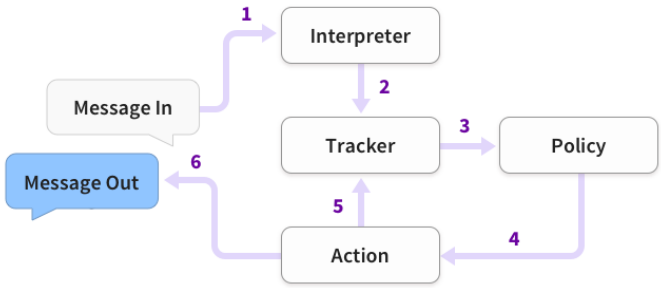
   Sama seperti di encoder

   Proses di ulang hingga decoder menghasilkan token akhir.

Layer akhir -> linear dan softmax -> probabilitas kata

Lapisan linier ini berfungsi untuk menghasilkan prediksi kata untuk token output yang akan dihasilkan.

2.2.7 Cara Kerja RASA



Gambar 2.7 Cara Kerja RASA

Sumber:

https://miro.medium.com/v2/resize:fit:1438/0*rGflH38V9KZPL2J_

Framework RASA bekerja dengan cara menerima pesan dari pengguna, memahami maksud dari pesan tersebut, mengelola percakapan, dan melakukan tindakan yang diminta oleh pengguna.

Proses kerja *Framework* RASA dapat digambarkan sebagai berikut:

1. Pengguna mengirimkan pesan ke chatbot.
2. Komponen NLU memahami maksud dari pesan tersebut.
3. Komponen DM mengelola percakapan antara chatbot dan pengguna.
4. Komponen Action Server melakukan tindakan yang diminta oleh pengguna.

2.2.8 Kelebihan RASA

Framework RASA memiliki beberapa kelebihan, yaitu:

1) Open source: *Framework* ini dapat digunakan secara gratis dan dapat dikembangkan oleh siapa saja.
2) Komunitas yang besar: *Framework* ini memiliki komunitas yang besar yang dapat membantu pengguna dalam pengembangan chatbot.

3) Fitur yang lengkap: *Framework* ini memiliki fitur yang lengkap yang dapat digunakan untuk membangun chatbot yang kompleks. *Framework* RASA dapat digunakan untuk berbagai tujuan, seperti:

1) Customer service: Chatbot dapat digunakan untuk menjawab pertanyaan dan memberikan solusi kepada pelanggan.
2) Penjualan: Chatbot dapat digunakan untuk menawarkan produk atau jasa kepada pelanggan.
3) Pendidikan: Chatbot dapat digunakan untuk memberikan materi pembelajaran kepada siswa.

Framework RASA adalah sebuah kerangka kerja yang *powerfull* dan fleksibel yang dapat digunakan untuk membangun dan mengembangkan chatbot berbasis AI. *Framework* ini memiliki banyak kelebihan, seperti open source, komunitas yang besar, dan fitur yang lengkap.

2.2.9 AFC

Automatic Fare Collection (AFC) merupakan sistem digitaliasi tiketing, melakukan validasi tanpa kontak dalam hitungan detik. Teknologi pengumpulan tarif otomatis menjadikan tingkat transparansi lebih tinggi dan dapat mengurangi tingkat penyalahgunaan, pencurian, dan korupsi karena pembayaran tarif melalui fasilitas ini daripada secara tunai.

Teknologi ini digunakan oleh PT XY sejak awal pengoperasian sebagai fasilitator layanan transportasi perkereta-apian modern. Pemeliharaan fasilitas tiket AFC memerlukan perhatian lebih rinci terutama dari sistem teknologi, sehingga pekerjaan pemeliharaan AFC serta perangkat-perangkat berhubungan dengan sistem tiket mulai dari Passenger Gate, Ticket Vending Machine, Add Value Machine (AVM), Automatic Remaining Value Checking Terminal (ARVCT), dan Ticket Office Machine (TOM) memiliki departemen tersendiri untuk mengurusnya.

Departemen PIHAK 2. Semua infrastruktur dikelola, baik itu berupa pencegahan maupun perbaikan hingga pemecahan masalah ketika terjadi kendala di lapangan terkait penggunaan operasioonal AFC. Dan berbagai permasalahan terkait AFC berbentuk keluhan pelanggan tidak dapat dihindari, seperti TVM yang tidak beroperasi dengan baik, permasalahan terkait pembayaran e-money, dan lain sebagainya.

Keluhan tersebut disampaikan oleh pengguna kepada PIHAK 1 kemudian dilanjutkan ke PIHAK 2 namun bisa juga birokrasi melibatkan pihak stasiun lalu pihak stasiun meneruskan kepada PIHAK 1 dan PIHAK 1 menyampaikan ke PIHAK 2 birokrasi tersebut dilakukan setiap adanya permasalahan terkait AFC.

2.2.10 Telegram

Telegram merupakan salah satu aplikasi perpesanan instan yang sangat populer saat ini. Aplikasi ini memiliki lebih dari 700 juta pengguna aktif di seluruh dunia, termasuk Indonesia. Telegram menawarkan berbagai fitur menarik, seperti pesan terenkripsi end-to-end, grup chat yang besar, dan saluran publik. Telegram didirikan pada tahun 2013 oleh Pavel Durov, pendiri VKontakte, salah satu situs jejaring sosial terbesar di Rusia. Aplikasi ini awalnya dikembangkan sebagai alternatif dari WhatsApp, yang pada saat itu dimiliki oleh Facebook. Telegram tersedia untuk berbagai platform, termasuk Android, iOS, Windows, macOS, dan Linux. Aplikasi ini dapat diunduh secara gratis.

2.2.11 Visual Studio Code

Visual Studio Code telah menjadi editor kode populer untuk pengembang selama beberapa tahun terakhir, tidak hanya karena gratis dan sumber terbuka, tetapi juga karena dukungan komunitas yang baik. Visual Studio Code adalah editor kode lintas platform yang dapat digunakan untuk berbagai bahasa pemrograman, termasuk JavaScript, Python, C++, dan Java. Editor ini juga dilengkapi dengan berbagai fitur dan ekstensi yang dapat disesuaikan untuk memenuhi kebutuhan pengguna.

Kelebihan Visual Studio Code:

1) Gratis dan sumber terbuka: Visual Studio Code tersedia secara gratis untuk digunakan dan dikembangkan oleh siapa saja.
2) Lintas platform: Visual Studio Code dapat digunakan di Windows, macOS, dan Linux.
3) Sederhana dan mudah digunakan: Visual Studio Code memiliki antarmuka pengguna yang sederhana dan mudah dipelajari.
4) Fitur yang lengkap: Visual Studio Code dilengkapi dengan berbagai fitur dan ekstensi yang dapat disesuaikan untuk memenuhi kebutuhan pengguna.
5) Dukungan komunitas yang baik: Visual Studio Code memiliki komunitas

yang besar dan aktif yang dapat memberikan bantuan dan dukungan.

Contoh penggunaan Visual Studio Code:

1) Pengembangan web: Visual Studio Code dapat digunakan untuk mengembangkan aplikasi web menggunakan JavaScript, HTML, dan CSS.
2) Pengembangan perangkat lunak: Visual Studio Code dapat digunakan untuk mengembangkan aplikasi perangkat lunak menggunakan C++, Java, dan Python.
3) Pemrograman data: Visual Studio Code dapat digunakan untuk pemrograman data menggunakan Python, R, dan SQL.

Visual Studio Code adalah editor kode yang kuat dan serbaguna yang dapat digunakan untuk berbagai tujuan. Editor ini gratis dan sumber terbuka, serta memiliki dukungan komunitas yang baik.

2.2.12 SQLite3

SQLite3 adalah sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dan open source di bawah lisensi Public Domain. Karena menjadi sistem open source, orang dapat menggunakan dan mengubahnya secara bebas. Bahkan perangkat lunak yang dibuat dari karya turunan tidak terbatas. SQLite3 memiliki beberapa keunggulan dibandingkan dengan RDBMS lain, termasuk:

a) Ukurannya yang kecil: SQLite3 hanya membutuhkan beberapa megabyte ruang disk untuk diinstal.
b) Performanya yang tinggi: SQLite3 dapat melakukan query dengan cepat, bahkan untuk database yang besar.
c) Kemudahan penggunaan: SQLite3 mudah digunakan dan dipelajari.

SQLite3 mendukung sistem operasi Linux, Windows, dan macOS. Selain itu, SQLite3 memiliki perintah query yang disebut SQL. SQL adalah bahasa standar yang digunakan untuk mengelola database. Ada 6 operasi dasar-dasar yang termasuk dalam SQLite3 termasuk membuat database, membuat tabel, menghapus tabel, menghapus database, sisipkan, perbarui, baca, dan hapus. Selain itu, SQLite3 mendukung banyak tipe data dengan panjang masing-masing. Berikut adalah

format data yang paling umum digunakan:

Table 2.2 Tipe Data SQLite3

Jenis Tipe	**Tipe**	**Keterangan**
Numerik	INT	-2147483648 s/d 2147483647 SIGNED 0 s/d 4294967295 UNSIGNED.
	FLOAT	Bilangan pecahan presisi tunggal.
	DOUBLE	Bilangan pecahan presisi ganda.
	REAL	Bilangan riil.
	DATE	Tanggal dengan format YYYY-MM-DD
	DATETIME	Tanggal dan waktu dengan format : YYYY-MM-DD HH:MM:SS
String	CHAR	String dengan panjang tetap sesuaidengan yang ditentukan. Panjangnya 1-255 karakter

Jenis Tipe	Tipe	Keterangan
	VARCHAR	String dengan panjang yang berubah ubah sesuai dengan yang disimpansaatitu. Panjangnya 1 – 255 karakter
	TEXT	String dengan panjang maksimum65535 karakter
	BLOB	String dengan panjang maksimum65535 karakter

2.2.13 Pengujian Akurasi

Pengujian sistem ini menggunakan confusion matrix untuk menganalisis seberapa baik hasil klasifikasi yang dilakukan untuk mengenali label dari kelas yang berbeda [14].

Table 2.3 Pengujian Akurasi

Klasifikasi	Diklasifikasikan	
	+	-
+	*True Positive*	*False Positive*
-	*False Negatives*	*True Negatives*

Pengukuran akurasi ini terbagi menjadi 4 istilah yaitu :

1. *True Positive* (TP), jumlah data prediksi positif yang terdeteksi benar.
2. *True Negative* (TN), jumlah data prediksi negatif yang terdeteksi benar.
3. *False Positive* (FP), jumlah data prediksi negatif yang terdeteksi data positif.
4. *False Negative* (FN), jumlah data prediksi positif yang terdeteksi data negatif.

Berdasarkan hasil klasifikasi yang dilakukan, maka dihitung:

a. Akurasi, pengukuran kinerja yang menunjukkan tingkat akurasi seluruh prediksi yang dibuat oleh model.

$$\text{Akurasi} = \frac{jumlah\ data\ yang\ uji\ benar}{jumlah\ semua\ data\ uji} \text{ x } 100\%$$

Rumus 2.3 Akurasi

b. Presisi, tingkat akurasi hasil klasifikasi seluruh dokumen dalam sistem.

$$\text{Presisi} = \frac{TP}{TP+FP} \text{ x } 100\%$$

Rumus 2.4 Presisi

c. Recall, tingkat keberhasilan sistem dalam mengenali suatu kategori.

$$\text{Recall} = \frac{TP}{TP+FN} \text{ x } 100\%$$

Rumus 2.5 Recall

d. F1-Score, deskripsi relativitas antara nilai presisi dan recall atau disebut dengan harmonic mean.

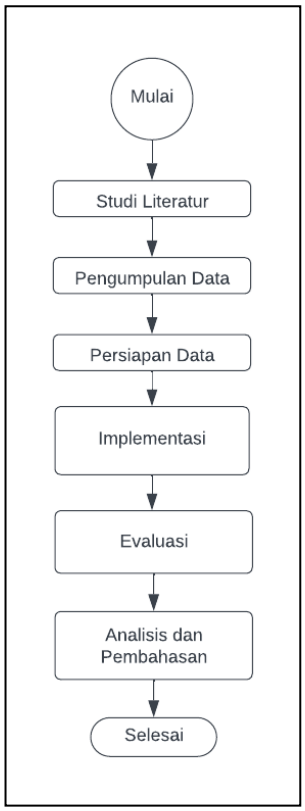
$$\text{F1-Score} = 2 \text{ x } \frac{(presisi\ x\ recall)}{(presisi + recall)} \text{ x } 100\%$$

Rumus 2.6 F1-Score

BAB III

METODE PENELITIAN

3.1 Alur penelitian



Gambar 3.1 Metodologi Penelitian

Pada penelitian akan dilakukan secara berurutan sesuai tahapan yang telah ditentukan.

3.2 Penjabaran Langkah Penelitian

Alur penelitian yang akan di lakukan memiliki beberapa tahapan yang dilakukan secara bertahap

3.2.1 Studi Literatur

Proses studi literatur merupakan tahapan awal yang diperlukan untuk dapat mengetahui dan memahami teori-teori yang dibutuhkan. Tahapan ini penting dilakukan dengan tujuan mengetahui penelitian terdahulu yang berkaitan, memahami teori terkait penelitian, dan beberapa hal lainnya. Rasa framework merupakan framework yang digunakan untuk mengembangkan chatbot. Rasa

framework pertama kali dirilis pada tahun 2017, dan telah digunakan oleh berbagai perusahaan dan organisasi di seluruh dunia. Rasa framework memiliki berbagai fitur yang dapat digunakan untuk mengembangkan chatbot yang kompleks dan efisien. Oleh karena itu, studi literatur merupakan hal yang penting untuk dilakukan dalam penelitian yang menggunakan Rasa framework. Peneliti harus benar-benar mencari teori-teori terkait Rasa framework.

3.2.2 Pengumpulan Data

Pada proses pengumpulan data, penulis melakukan pengumpulan data yang didapatkan dari pihak instansi setelah melakukan penandatanganan Non-Disclosure Agreement (NDA) atau perjanjian rahasia karena data yang didapatkan bersifat rahasia. Data yang diberikan pada penelitian ini berupa list data semua jenis permasalahan yang berjumlah 17 permasalahan terkait AFC yang dituangkan dalam sebuah file format excel.

Dalam file tersebut, data disajikan dalam dua sheet, sheet pertama berisi pengelompokan atau semua jenis permasalahan beserta respon sebanyak 17 jenis masalah dengan detail kolom pertama berisi type permasalahan, kolom kedua problem dan kolom ketiga berisi action atau jawaban dari permasalahan tersebut, berikut salah satu jenis permasalahan dalam sheet 1:

1. Type: TOM
2. Poblem: Input Power Failure
3. Action: Lakukan pengencangan kabel pada UPS TOM, Jika gangguan berulang, hubungi tim AFC Maintenance

Sheet kedua berisi contoh laporan permasalahan yang telah terjadi dan telah dilapor dengan bersumber dari data rekapan report selama ini sebanyak 101 laporan, berikut salah satu contoh laporan gangguan dalam sheet 2:

1. Stasiun : 12
2. Gangguan Fasilitas : TOM
3. Tgl. Pelaporan : 12/09/2023
4. Area Spesifik : Selatan
5. Problem : terdapat tulisan input power failure

5.2.1 Persiapan Data

Chatbot layanan informasi yang dibangun menggunakan RASA Open-Source

Framework memerlukan beberapa file konfigurasi sebagai dataset pelatihan yang meliputi komponen pada RASA. Menggunakan data yang telah ada, data dianalisis dan diseusaikan dengan kebutuhan tiap file komponen RASA. Penjelasan komponen yang dikonfigurasi dituliskan sebagai berikut dengan memberikan contoh satu jenis permasalahan terkait AFC yakni TOM input power:

1. NLU.yml

   intent: TOM_input_power

   examples: |

   a. TOM input power failure
   b. input power gagal

2. Rules.yml

   rule: TOM input power bermasalah

   steps:

   a. intent: TOM_input_power
   b. action: utter_TOM_input_power

3. Stories.yml

   story: TOM input power bermasalah

   steps:

   a. action: action_save_data
   b. intent: TOM_input_power
   c. action: utter_TOM_input_power

4. Test_stories.yml

   story: TOM input power bermasalah

   steps:

   user: |

   a. input power gagal
   b. intent: TOM_input_power
   c. action: utter_TOM_input_power

5. Domain.yml

   intents:

   TOM_input_power

   responses:

utter_TOM_input_power:

text: "Lakukan pengencangan kabel pada UPS TOM, Jika gangguan berulang, hubungi tim AFC Maintenance."

6. Config.yml

Digunakan konfigurasi default RASA framework

a. name: WhitespaceTokenizer
b. name: RegexFeaturizer
c. name: LexicalSyntacticFeaturizer
d. name: CountVectorsFeaturizer
e. name: CountVectorsFeaturizer
   analyzer: char_wb
   min_ngram: 1
   max_ngram: 4
f. name: DIETClassifier
   epochs: 100
   constrain_similarities: true
g. name: EntitySynonymMapper
h. name: ResponseSelector
   epochs: 100
   constrain_similarities: true
i. name: FallbackClassifier
   threshold: 0.3
   ambiguity_threshold: 0.1

7. Credentials.yml

Menggunakan default credentials yang tidak diubah. Adapun code yang dilakukan *uncomment*. Dalam kode, URL API adalah http://localhost:5002/api. URL ini berarti bahwa chatbot dapat diakses di port 5002 pada komputer lokal.

rasa:

url: "http://localhost:5002/api"

8. Endpoints.yml

Menggunakan default endpoints, adapun code yang dilakukan *uncomment.*

Dilakukan pengaktifan domain selektif di konfigurasi endpoints untuk memilih tindakan mana yang harus menerima domain yang telah dituliskan pada actions.py

action_endpoint:

url: "http://localhost:5055/webhook"

5.2.2 Implementasi

Pada tahapan ini, dilakukan konfigurasi framework melibatkan pendefinisian jenis bahasa, spesifikasi pipeline dan policies. Proses pelatihan menggunakan algoritma pembelajaran mesin yang dispesifikasikan pada bagian policies. Bagian ini menentukan proses pembelajaran mesin yang akan digunakan untuk mengolah pesan teks yang diterima dan pesan teks respon ke pengguna.

1) Pipeline Default

a) Preprocessing

a. name: WhitespaceTokenizer

“input”, “power”, “gagal”

b. name: RegexFeaturizer

i. kata "input" akan menjadi "input", "power" akan menjadi "power", dan "gagal" akan menjadi "gagal"

b) Feature Engineering

c. name: LexicalSyntacticFeaturizer

Table 3.4 Lexical Featurizer

Feature Name	**Value**
BOS	TRUE
EOS	TRUE
low	TRUE
upper	FALSE
title	FALSE
digit	FALSE

prefix5	input
prefix2	in
suffix5	gagal
suffix3	gal
suffix2	al
suffix1	l
pos	NOUN
pos2	NN

d. name: CountVectorsFeaturizer
   i. "input": 1
   ii. "power": 1+
   iii. "gagal": 1

e. name: CountVectorsFeaturizer
   analyzer: char_wb
   min_ngram: 1
   max_ngram: 4

Table 3.5 Count Vectors Featurize

Word/Phrase	**Numerical Representation**
input	1
power	1
gagal	1
input_power	1
input_gagal	1
power_gagal	1
input_power_gagal	1

c) Intent Classification

f. name: DIETClassifier
   epochs: 100

constrain_similarities: true

i. Intent: TOM_input_power

ii. Entitas:

1. Problem: input power

d) Entity Recognition

g. name: EntitySynonymMapper

iii. Entitas:

1. Problem: input power

e) Response Selection

h. name: ResponseSelector

epochs: 100

constrain_similarities: true

iv. utter_TOM_input_power

f) Fallback

i. name: FallbackClassifier

threshold: 0.3

ambiguity_threshold: 0.1

(*FallbackClassifier tidak akan digunakan karena ResponseSelector telah menemukan respons yang sesuai.)*

2) Pipeline Replace

a. name: WhitespaceTokenizer

b. name: LexicalSyntacticFeaturizer

c. name: CountVectorsFeaturizer

i. OOV_token: oov

ii. token_pattern: (?u)\b\w+\b

d. name: CountVectorsFeaturizer

i. analyzer: char_wb

ii. min_ngram: 1

iii. max_ngram: 4

e. name: DIETClassifier

i. epochs: 100

ii. ranking_length: 5

f. name: EntitySynonymMapper

3) Pipeline Replace Mask

a. name: WhitespaceTokenizer

b. name: LexicalSyntacticFeaturizer

c. name: CountVectorsFeaturizer

i. OOV_token: oov

ii. token_pattern: (?u)\b\w+\b

d. name: CountVectorsFeaturizer

i. analyzer: char_wb

ii. min_ngram: 1

iii. max_ngram: 4

e. name: DIETClassifier

i. epochs: 100

ii. ranking_length: 5

iii. use_masked_language_model: True

f. name: EntitySynonymMapper

4) Pipeline Light

a. name: WhitespaceTokenizer

b. name: CountVectorsFeaturizer

c. name: CountVectorsFeaturizer

i. analyzer: char_wb

ii. min_ngram: 1

iii. max_ngram: 4

d. name: DIETClassifier

i. epochs: 20

ii. learning_rate: 0.005

iii. num_transformer_layers: 0

iv. embedding_dimension: 10

v. weight_sparcity: 0.90

vi. hidden_layer_sizes:

1. text: [256, 128]

5) Pipeline Heavy
    a. name: SpacyNLP
    b. name: SpacyTokenizer
    c. name: SpacyFeaturizer
    d. name: RegexFeaturizer
    e. name: LexicalSyntacticFeaturizer
    f. name: CountVectorsFeaturizer
        i. analyzer: char_wb
        ii. min_ngram: 1
        iii. max_ngram: 4
    g. name: CountVectorsFeaturizer
    h. name: DIETClassifier

    epochs: 100

    num_transformer_layers: 4

    transformer_size: 256

    use_masked_language_model: True

    drop_rate: 0.25

    weight_sparsity: 0.7

    batch_size: [64, 256]

    embedding_dimension: 30

    hidden_layer_sizes:

        text: [512, 128]

6) Policies
    a. name: MemoizationPolicy
        i. Intent: TOM_input_power
        ii. Respons: Lakukan pengencangan kabel pada UPS TOM, Jika gangguan berulang, hubungi tim AFC Maintenance
    b. name: RulePolicy
        i. (Untuk kalimat "input power gagal", RulePolicy tidak dapat menentukan intent dan respons dengan aturan-aturan tertentu. Oleh karena itu, RulePolicy tidak akan menghasilkan output.)

c. name: UnexpecTEDIntentPolicy
   i. max_history: 5
   ii. epochs: 100
   iii. UnexpecTEDIntentPolicy mungkin akan memprediksi intent *TOM_input_power* dan intent lain yang memiliki kemiripan.

d. name: TEDPolicy
   max_history: 5
   epochs: 100
   constrain_similarities: true
   UnexpecTEDIntentPolicy mungkin akan memprediksi intent *TOM_input_power* dan intent lain yang memiliki kemiripan, serta respons yang paling sesuai dengan intent tersebut

7) Pelatihan
   Latih model menggunakan data dan cerita NLU, simpan model yang dilatih dalam format ./models dengan menggunakan syntax
   rasa train

5.2.3 Evaluasi

Pengujian ini dilakukan untuk mengetahui apakah chatbot berjalan dengan baik atau tidak dan menghasilkan respon yang akurat atau tidak. Adapun metode untuk pengujian adalah *confusion matrix* serta perhitungan nilai precision, recall, dan F1-Score. Evaluasi dilakukan pada model NLU untuk mengukur kemampuan chatbot dalam memahami teks input dan evaluasi model dialog untuk mengukur kemampuan chatbot dalam memberikan respon atau aksi kepada yang sesuai dengan kebutuhan pengguna.

5.2.4 Analisis dan Pembahasan

Pada tahap in, dilakukan analisis lebih lanjut beserta pembahasan secara detail terhadap penelitian yang telah dilakukan. Setelah melakukan analisis menggunakan data yang telah didapatkan maka akan dilakukan proses pembahasan dalam bentuk penulisan laporan

5.3 Perhitungan Transformer

Berikut ini terdapat contoh perhitungan manual dari komponen utama atau

komponen inti dari DIET Arsitektur yakni transformer, dengan menggunakan input permasalahan terkait dengan “TVM input power gagal”

Word Embedding

1. Split token

   TVM, input, power, gagal

2. Word embedding dengan 4 dimensi

   [0.4842, -1.7116, 0.1058, 0.9797], TVM

   [-0.4022, -1.1796, -0.8738, -1.1974], input

   [-0.9172, 0.7308, 1.8257, 1.1559], power

   [-0.3148, 1.5183, -0.1331, -0.6021] gagal

3. Penambahan positional encoding

   Dengan nilai positional encoding

   [0.0000, 1.0000, 0.0000, 1.0000],

   [0.8415, 0.5403, 0.0100, 0.9999],

   [0.9093, -0.4161, 0.0200, 0.9998],

   [0.1411, -0.9900, 0.0300, 0.9996]

   Ditambahkan dengan word embedding

   [0.4842, -0.7116, 0.1058, 1.9797],

   [0.4393, -0.6393, -0.8638, -0.1975],

   [-0.0079, 0.3147, 1.8457, 2.1557],

   [-0.1737, 0.5283, -0.1031, 0.3975]

4. Melakukan perhitungan untuk self attention dimulai dari mendefinisikan query, key dan value. Query, key, dan value yang digunakan bernilai sama dan diambil dari nilai word embedding yang telah ditambahkan positional encoding.

   Query =

   [0.4842, -0.7116, 0.1058, 1.9797],

   [0.4393, -0.6393, -0.8638, -0.1975],

   [-0.0079, 0.3147, 1.8457, 2.1557],

   [-0.1737, 0.5283, -0.1031, 0.3975]

   Key =

   [0.4842, -0.7116, 0.1058, 1.9797],

[0.4393, -0.6393, -0.8638, -0.1975],

[-0.0079, 0.3147, 1.8457, 2.1557],

[-0.1737, 0.5283, -0.1031, 0.3975]

Value =

[0.4842, -0.7116, 0.1058, 1.9797],

[0.4393, -0.6393, -0.8638, -0.1975],

[-0.0079, 0.3147, 1.8457, 2.1557],

[-0.1737, 0.5283, -0.1031, 0.3975]

5. Melakukan dot product antara query dengan key

   Key T=

   [0.4842, 0.4393, -0.0079, -0.1737],

   [-0.7116, -0.6393, 0.3147, 0.5283],

   [0.1058, -0.8638, 1.8457, -0.1031],

   [1.9797, -0.1975, 2.1557, 0.3975]

   (0.4842 x 0.4842) + ((-0.7116) x (-0.7116)) + (0.1058 x 0.1058) +

   (1.9797 x 1.9797) = 0.234 + 0.506 + 0.01119 + 3.91 = 4. 6712

   Hasil

   [4.6712, 0.1853, 4.2351, 0.3160],

   [0.1853, 1.3868, -2.2247, -0.4035],

   [4.2351, -2.2247, 8.1528, 0.8342],

   [0.3160, -0.4035, 0.8342, 0.4779]

6. Melakukan tahapan normalisasi agar stabilitas gradient dalam proses softmax

   Dengan rumus : matriks dot product/ akar(embedding_dimension)

   [2.3356, 0.0926, 2.1176, 0.1580],

   [0.0926, 0.6934, -1.1124, -0.2017],

   [2.1176, -1.1124, 4.0764, 0.4171],

   [0.1580, -0.2017, 0.4171, 0.2390]

   $4.6712 / \sqrt{4} = 2.3356$

7. Melakukan softmax untuk mendapatkan bobot attention

   e^x, di mana e adalah bilangan Euler (sekitar 2.71828)

   1.2354

Nilai Eksponensial dari Matriks Sebelum Softmax:

[10.3357, 1.0970, 8.3112, 1.1712],

[1.0970, 2.0005, 0.3288, 0.8173],

[8.3112, 0.3288, 58.9329, 1.5176],

[1.1712, 0.8173, 1.5176, 1.2700]

Jumlah Nilai Eksponensial: 20.9151

Hasil

[0.4942, 0.0525, 0.3974, 0.0560],

[0.2585, 0.4714, 0.0775, 0.1926],

[0.1203, 0.0048, 0.8530, 0.0220],

[0.2452, 0.1711, 0.3177, 0.2659]

softmax dihitung pada baris pertama dari matriks eksponensial tersebut:

softmax ([10.3357, 1.0970, 8.3112, 1.17120])

Hasilnya adalah: [0.4942, 0.0525, 0.3974, 0.0560]

Setiap baris pada matriks tersebut dihitung menggunakan fungsi softmax secara terpisah dan nilai total dari setiap baris tersebut akan berjumlah 1.

8. Menghitung context

   context = attention_weights * value

   Attention weight

   [0.4942, 0.0525, 0.3974, 0.0560],

   [0.2585, 0.4714, 0.0775, 0.1926],

   [0.1203, 0.0048, 0.8530, 0.0220],

   [0.2452, 0.1711, 0.3177, 0.2659]

   Value

   [0.4842, -0.7116, 0.1058, 1.9797],

   [0.4393, -0.6393, -0.8638, -0.1975],

   [-0.0079, 0.3147, 1.8457, 2.1557],

   [-0.1737, 0.5283, -0.1031, 0.3975]

   Perhitungan

   0. 4942 × 0.4842 + 0. 0525 × 0.4393 + 0. 3974 × (−0.0079) + 0. 0560 × (−0.1737) = 0. 2495 dimensi (1,1)

   Hasil

[0.2495, -0.2306, 0.7346, 1.8468],
[0.2982, -0.3592, -0.2567, 0.6622],
[0.0498, 0.1914, 1.5807, 2.0847],
[0.1452, -0.0434, 0.4372, 1.2423]

Menghasilkan keterkaitan kata yakni “power” dan “TVM” memiliki korelasi konteks paling kuat dalam kalimat ini. Hasil ini selanjutnya akan diimplementasikan ke tahapan Conditional Random Field untuk label entitas yang nantinya akan diolah menjadi fitur-fitur yang nantinya akan digabungkan dengan fitur-fitur lain yang diekstraksi dari kalimat, seperti informasi linguistik, morfologi kata, atau fitur sintaksis lainnya.

5.4 Alat dan Bahan Tugas Akhir

Adapun alat dan bahan yang digunakan dalam penelitian ini adalah sebagai berikut:

5.4.1 Alat

Berikut adalah alat yang digunakan dalam penelitian ini:

1. Spesifikasi Laptop, komputer atau notebook:
    a. Minimum spesifikasi komputer, laptop, atau notebook agar dapat melakukan penelitian ini, sebagai berikut:
        i. Operating System Windows 10 (64-bit), x86-64 based
        ii. Menggunakan Processor Intel Core i5
        iii. Kebutuhan RAM paling minimal 4 GB
        iv. Memiliki ruang memori yang bebas setidaknya 10 GB
    b. Pada penelitian ini penulis menggunakan laptop dengan spesifikasi sebagai berikut:
        i. Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
        ii. RAM 12 GB
        iii. Operating System Windows 11
        iv. SSD 500GB
        v. GPU intel UHD Graphics 620
2. Visual Studio Code
3. SQLite3
4. Telegram

5. RASA Framework

5.4.2 Bahan

Berikut adalah bahan yang digunakan dalam penelitian ini:

1. Dataset bersifat rahasia yang diberikan oleh pihak pertama yakni PT XYZ yang didapatkan melalui hasil laporan yang dikumpulkan.
2. Jurnal penelitian dari penelitian sebelumnya yang dijadikan dasar teori dan rangkaian konsep serta ide untuk mendukung penelitian.

Daftar Pustaka

[1] E. a. P. Pebriantara, "Rancangan Bangun Aplikasi Pembelajaran dengan Memanfaatkan Chatbot API Dialogflow dan Moodle Berbasis Android pada SMA IT ALIA Tangerang"," Vols. 3, hal.328-335, 2018.

[2] L. Anindyati, "ANALISIS DAN PERANCANGAN APLIKASI CHATBOT MENGGUNAKAN FRAMEWORK RASA DAN SISTEM INFORMASI PEMELIHARAAN APLIKASI (STUDI KASUS: CHATBOT PENERIMAAN MAHASISWA BARU POLITEKNIK ASTRA)," *Jurnal Teknologi Informasi dan Ilmu Kopmputer (JTIIK)*, vol. 10, pp. 291-300, 2022.

[3] A. R. D. P. J. &. B. M. JANSSEN, "How to Make chatbots productive–A user-oriented implementation framework," *International Journal of Human - Computer Studies*, 2022.

[4] B. R. N. A. S. RANOLIYA, "Chatbot for university related FAQs," *Internasional Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1525-1530, 2017.

[5] G. C. N. &. A. F. Battineni, "AI Chatbot Design during an Epidemic like the Novel Coronavirus," *Healthcare*, vol. 8, p. 154, 2020.

[6] P. M. Jakarta, Seri Pembelajaran: Menjaga Laju Ratangga, Membangun Budaya, Mewujudkan Layanan Berkelas Dunia, jakarta: Jakartamrt.co.id, 2022.

[7] Y. R. R. H. A. F. Windiatmoko, "Mengembangkan Chatbot Facebook Berbasis Pembelajaran Mendalam Menggunakan Kerangka RASA untuk Pertanyaan Universitas," *ICITDA 2020,* 2020.

[8] A. A. ,. S. Ferdian, "Pengembangan Chatbot untuk Informasi Wisata Interaktif di Tangerang Selatan menggunakan Framework Rasa," *Jurnal Teknologi Dan Sistem Informasi Bisnis,* vol. 5, pp. 476-483, 2023.

[9] J. R. Fonsecal, "ChatBot for student service based on RASA Framework," *Research Square*, 2023.

[10] Z. H. N. H. R. R. A. Pradana, "Chatbot-based Information Service using RASA Open-Source Framework in Prambanan Temple Tourism Object," *Jurnal RESTI*, vol. 6, no. 4, pp. 656-662, 2022.

[11] N. Cannavaro, "Aplikasi Chatbot untuk Layanan Akademik Menggunakan Platform RASA Open Source dengan Fitur Two Stage Fallback," *Jurnal Ilmu Komputer dan Informatika (JIKI)*, vol. 3, no. 1, pp. 53-64, 2023.

[12] R. &. D. A. Khan, "Build Better Chatbots," 2018.

[13] W. Nwankwo, " Interactive Advising with Bots : Improving Academic Excellence in Educational Establishments," *American Journal of Operations Management and Information System*, vol. 3, pp. 6-21, 2018.

[14] J. K. M. &. P. J. Han, "Data Mining: Concepts and Techniques.," 2012.

[15] N. l. W. S. R. A. F. N. W. A. H. S. R. S. D. Ginantra, "Data Mining dan

Penerapan Algoritma," *Yayasan Kita Menulis,* 2021.

[16] F. Rayyan, "Pengembangan Chatbot untuk Aplikasi Online Chat Telegram dengan Pendekatan Klasifikasi Emosi pada Teks Menggunakan Metode Indobert-Lite," *UIN Syarif Hidayatullah,* 2022.

[17] Haviluddin, "Memahami Penggunaan UML (Unified Modelling Language)," *Jurnal Informatika Mulawarman,* vol. 6, no. 1, 2011.

LAMPIRAN