

Penerapan Framework Rasa untuk Membangun Sistem FAQ Bot Sebagai Layanan Informasi BIRO 3

Rivai H. H. Talenggoran¹, Lucia D. Krisnawati², Gloria Virginia³
Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana
Jl. Dr. Wahidin Sudirohusodo 5-25, Yogyakarta

¹rivai.hayashi@ti.ukdw.ac.id

²krisna@staff.ukdw.ac.id

³virginia@staff.ukdw.ac.id

Abstract—Duta Wacana Christian University (UKDW), specifically Bureau 3 (BIRO 3), provides a variety of information services accessible through both online channels, such as Instagram, WhatsApp, and email, and offline methods. However, BIRO 3 has not yet implemented an FAQ Bot system, which means that frequently asked questions are still answered manually. This practice renders the question-and-answer process regarding campus information repetitive and time-consuming. This research aims to implement the RASA Open Source framework to develop an FAQ Bot system to automate the retrieval of information for frequently asked questions at BIRO 3, thereby enhancing the efficiency of its information services.

The system was developed using RASA Open Source and implemented on the Telegram messaging platform. The evaluation was conducted through a two-fold approach. First, internal testing of the RASA model on the validation dataset yielded optimal results, achieving accuracy, precision, and F1-scores of 1.000. On the test data, the model demonstrated strong performance with an accuracy of 0.915, a precision of 0.928, and an F1-score of 0.912. Second, functional testing was performed by engaging users in predefined scenarios. This second phase of testing resulted in a functional accuracy of 95% based on 200 collected data points.

The user testing results indicate that the FAQ Bot system was successfully developed and capable of achieving a functional accuracy rate of 95%. Despite its high performance, limitations were identified in the form of eleven false positive and false negative cases out of the 200 data points. This suggests that the model has not yet perfectly learned to comprehend all variations of user input. Therefore, recommendations such as expanding the training dataset and exploring modifications to the RASA Open Source framework are proposed to refine the system's capabilities, enabling it to handle all types of inquiries accurately.

Keywords : FAQ Bot, RASA Open Source, Telegram

Intisari—Universitas Kristen Duta Wacana(UKDW), khususnya BIRO 3 memiliki berbagai layanan informasi yang dapat diakses baik secara *online* seperti *Instagram*, *whatsapp*, *Email* maupun *offline*. Namun saat ini BIRO 3 belum mengimplementasikan sistem *FAQ Bot*, sehingga pertanyaan yang sering diajukan masih dijawab secara manual. Hal ini membuat aktivitas tanya jawab mengenai informasi kampus menjadi repetitif dan memakan waktu. Penelitian ini bertujuan untuk menerapkan *RASA Open Source* untuk mengembangkan sistem *FAQ Bot* guna mempermudah pencarian informasi terkait pertanyaan yang sering ditanyakan di BIRO 3 secara otomatis serta meningkatkan efisiensi layanan informasi.

Sistem dikembangkan dengan *RASA Open Source* dan diimplementasikan pada antarmuka *Telegram*. Evaluasi dilakukan melalui dua pendekatan. Pertama, pengujian internal model RASA menunjukkan hasil optimal pada data validasi dengan nilai akurasi, presisi, dan skor-F1 mencapai 1.000. Pada data uji (*test data*), model menunjukkan performa yang kuat dengan akurasi 0.915, presisi 0.928, dan skor-F1 0.912. Kedua, pengujian fungsional dilakukan dengan melibatkan pengguna melalui skenario. Hasil pengujian kedua mendapatkan akurasi sebesar 95% dari 200 data yang dikumpulkan

Hasil pengujian pengguna menunjukkan bahwa sistem FAQ Bot berhasil dikembangkan dan mampu mencapai tingkat akurasi fungsional sebesar 95%. Meskipun performanya tinggi, masih ditemukan kelemahan berupa sebelas kasus *false positive* dan *negative* dari 200 data, yang mengindikasikan bahwa model belum sempurna dalam memahami semua variasi input. Oleh karena itu, saran seperti penambahan data latih dan eksplorasi modifikasi framework RASA Open Source direkomendasikan untuk menyempurnakan kemampuan sistem agar dapat menangani seluruh jenis pertanyaan secara akurat.

Kata Kunci—FAQ Bot, RASA Open Source, Telegram

I. PENDAHULUAN

A. Latar Belakang

Universitas Kristen Duta Wacana(UKDW), khususnya BIRO 3 memiliki berbagai layanan informasi yang dapat diakses baik secara *online* seperti *Instagram*, *whatsapp*, *Email* maupun *offline*. Namun saat ini BIRO 3 belum mengimplementasikan sistem *FAQ Bot*, sehingga pertanyaan yang sering diajukan masih dijawab secara manual. Hal ini membuat aktivitas tanya jawab mengenai informasi kampus menjadi repetitif dan memakan waktu. Berdasarkan survei yang dilakukan peneliti di BIRO 3, masalah yang sering muncul berkaitan dengan layanan kerja. Mahasiswa sering mengajukan pertanyaan di luar jam kerja, sehingga jawaban baru bisa diberikan setelah jam kerja, yang berdampak pada rendahnya efisiensi layanan informasi. Selain itu, minimnya fitur layanan informasi membuat mahasiswa mengulang pertanyaan dasar yang sebenarnya sudah terdokumentasi di layanan informasi yang ada seperti *Instagram*.

FAQ Bot merupakan salah satu jenis *chatbot* yang mampu untuk menjawab pertanyaan yang sering ditanyakan. Cara kerja *FAQ Bot* adalah dengan mengenali input dari pengguna dengan mencocokkan pola teks dalam basis

pengetahuan. Setelah ditemukan kesesuaian pola, *FAQ Bot* akan mengeluarkan jawaban yang sudah disiapkan sebelumnya[16]. *Chatbot* saat ini sudah mulai banyak digunakan di berbagai sektor seperti pendidikan. Teknologi ini bermanfaat untuk diimplementasikan di dunia pendidikan khususnya universitas dengan melayani atau menjawab pertanyaan dari pengguna, baik mahasiswa maupun bukan mahasiswa. Dengan teknologi ini, kualitas akan pelayanan informasi universitas dapat meningkat sehingga menambah daya tarik universitas[6].

Penelitian ini bertujuan untuk menerapkan *RASA Open Source* untuk mengembangkan sistem *FAQ Bot* guna mempermudah pencarian informasi terkait pertanyaan yang sering ditanyakan di BIRO 3 secara otomatis serta meningkatkan efisiensi layanan informasi.

B. Tinjauan Pustaka

Peneliti [14] mengembangkan *chatbot* yang memberikan jawaban untuk setiap pertanyaan berdasarkan *dataset* pertanyaan yang sering ditanyakan menggunakan *Artificial Intelligence Markup Language(AIML)* dan *Laten Semantic Analysis(LSA)* pada Universitas Manipal. Pertanyaan umum, berbasis template seperti sambutan dan lainnya akan dijawab menggunakan *AIML*, sementara pertanyaan berbasis layanan akan menggunakan *LSA* untuk memberikan repons kapan saja. *Chatbot* ini membantu mahasiswa mendapatkan informasi seperti peringkat universitas, ketersediaan layanan, lingkungan universitas, pembaruan mengenai aktifitas yang berlangsung di kampus, dan banyak informasi akademik lainnya.

Peneliti [11] Mangotra et al., (2024) melakukan penelitian dan pengembangan *FAQ Chatbot* dengan menggunakan arsitektur *Deep Learning*. Penelitian ini mengembangkan *FAQ Chatbot* dengan *dataset* pertanyaan yang sering ditanyakan di kampus Indira Gandhi Delhi Technical University for Women sebanyak 75 data dan membandingkan 2 arsitektur *Deep Learning(Bidirectional LSTM dan FFNN)* untuk melihat arsitektur mana yang lebih baik. Hasilnya adalah dengan 550 *epoch*, *Feed-Forward Neural Network* menghasilkan akurasi pada data latihan sebanyak 80% dan data validasi sebanyak 48.27%. Dengan jumlah *epoch* yang sama, *Bidirectional LSTM* menghasilkan akurasi pada data latihan sebanyak 88.6% dan data validasi sebanyak 27.56%. Ini menyimpulkan bahwa *LSTM* lebih baik daripada *FFNN*.

Peneliti [17] melakukan penelitian dan pengembangan *Chatbot* untuk menjawab pertanyaan seputar perguruan tinggi Kathmandu Engineering College menggunakan *RASA Open Source*. *RASA* terdiri dari dua komponen utama: *RASA Core* dan *RASA NLU*. *RASA NLU* digunakan untuk menentukan niat pengguna dan mengekstrak entitas yang diperlukan dari input pengguna. *RASA Core* bertugas menghasilkan output dengan membangun model probabilistik. *Chatbot* ini dirancang khusus untuk Kathmandu Engineering College dan menunjukkan kinerja yang kuat selama pengujian yang dilakukan dalam kerangka *RASA*. Untuk mengevaluasi kinerjanya, digunakan metrik evaluasi seperti presisi, recall, dan skor F1. Hasil evaluasi menunjukkan nilai rata-rata 1 untuk setiap metrik, yang menandakan performa yang sangat baik dalam menjawab pertanyaan pengguna.

Hasil tinjauan dari penelitian-penelitian sebelumnya memperlihatkan bahwa *chatbot* dan *FAQ Bot* dapat diimplementasikan di berbagai institusi, khususnya universitas. Berbagai metode telah dipilih untuk membangun sistem *chatbot* menggunakan *Deep Learning*, *AIML* dan framework seperti *RASA Open Source*. Peneliti mengimplementasikan gabungan dari beberapa metode yang sudah ditinjau, yaitu penggunaan *Rasa Open Source* untuk membangun *FAQ Bot* dengan algoritma *Machine Learning* yang mampu menjawab pertanyaan yang sering ditanyakan, dengan subjek penelitian adalah BIRO 3. Yang menjadi perbedaan dengan hasil tinjauan adalah perbedaan metode dan topik, dimana hasil tinjauan menggunakan *Deep Learning*, *AIML*, dan *RASA* untuk membangun *chatbot* untuk kebutuhan skala universitas, sedangkan penelitian ini akan melakukan penelitian untuk membangun sistem *FAQ Bot* menggunakan framework *Rasa Open Source*, khususnya di BIRO 3.

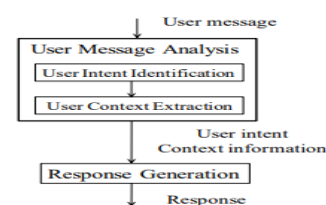
II. LANDASAN TEORI

A. NLP

Natural Language Processing (NLP) merupakan bagian dari kecerdasan buatan untuk mengembangkan sistem otomatis yang dapat memahami atau menghasilkan teks dalam bahasa alami. Bidang *NLP* terdiri dari dua sub-bidang: *Natural Language Understanding(NLU)* dan *Natural Language Generation*. *Natural Language Understanding(NLU)* mengambil teks sebagai input. *NLU* berusaha mengembangkan teknik untuk memahami dan menginterpretasikan teks dalam bahasa alami guna melakukan berbagai keputusan: bertindak sesuai instruksi tertentu, mengekstrak informasi relevan dari teks, memberikan jawaban atas pertanyaan, mengklasifikasikan atau merangkum dokumen, atau membuat representasi perantara dari teks yang dapat digunakan untuk pemrosesan lebih lanjut dalam aplikasi lain. Tujuan utama dari *Natural Language Understanding (NLU)* adalah untuk memahami teks sebagaimana manusia [1].

B. Chatbot

Chatbot adalah program komputer yang dapat mensimulasikan percakapan manusia. *Chatbot* dapat memahami dan memproses *request* pengguna secara otomatis dengan cara mengidentifikasi pola atau kata kunci yang relevan dengan *request* pengguna mengirim respons secara otomatis berdasarkan dengan hasil identifikasi tersebut [8].



Gambar 1 Struktur umum chatbot

Gambar 1 menunjukkan struktur umum dari sebuah *Chatbot*. *Chatbot* terdiri dari komponen-komponen seperti

User message analysis dan Response generation. User message analysis menggunakan teknik Natural Language Processing untuk memahami apa yang dibicarakan pengguna. Komponen ini terdiri dari dua tugas utama : identifikasi *intent* dan ekstraksi *entities*. Response Generation Component dalam chatbot bertugas untuk menghasilkan respon yang akan disampaikan kepada pengguna. Salah satu model yang dipakai untuk menghasilkan respon adalah *Pattern-Based Model*. Untuk model berbasis pola, chatbot mencocokkan pesan pengguna dengan setiap pola pertanyaan-jawaban yang mendasarinya untuk menghasilkan respons [7].

C. FAQ Bot

FAQ Bot merupakan salah satu jenis *chatbot* yang dipakai untuk menjawab beberapa pertanyaan yang sering ditanyakan. Cara kerja FAQ Bot adalah dengan mengenali input dari pengguna dengan mencocokkan pola teks dalam basis pengetahuan. Setelah ditemukan kesesuaian pola, FAQ Bot akan mengeluarkan jawaban yang sudah disiapkan sebelumnya[16]. FAQ Bot dapat membantu pengguna mendapatkan informasi tanpa intervensi dari manusia. Hal ini dapat meningkatkan pengalaman pengguna dengan membuat layanan informasi yang mudah diakses [2].

D. RASA Open Source

RASA Open Source adalah kerangka kerja Open Source untuk membangun chatbot AI yang terdiri dari dua komponen: RASA NLU dan RASA Core. Rasa Core adalah komponen yang mengelola mesin dialog dalam kerangka kerja ini dan menciptakan chatbot yang lebih kompleks dengan kustomisasi. RASA NLU membantu para pengembang dengan teknologi dan alat yang diperlukan untuk menangkap dan memahami input pengguna, serta menentukan niat dan entitas (Ingale et al.,2020). Di dalam RASA NLU terdapat 1 file komponen yang digunakan untuk dapat memahami input pengguna yaitu *nlu.yml* dan *config.yml*. Sedangkan RASA Core terdapat 3 file yang digunakan untuk mengelola dialog, mengatur percakapan dan melatih model. File tersebut adalah *rules.yml* dan *domain.yml*. selain RASA NLU dan RASA Core, terdapat file yang digunakan untuk konfigurasi konektor dengan Telegram, yaitu file *credentials.yml*. Berikut merupakan penjelasan masing-masing file:

1. *nlu.yml* : berisi contoh-contoh kalimat dari pengguna yang dikelompokkan berdasarkan *inten*. Di dalam file ini terbagi atas dua bagian, yaitu *intent* dan *examples*. *Intent* Merupakan Tujuan atau maksud yang ingin disampaikan pengguna melalui input yang diberikan. Umumnya dalam konteks *machine learning* disebut sebagai kelas. *Examples* Merupakan contoh kalimat dari pengguna yang dikelompokkan berdasarkan *intent*. Dalam konteks *Machine Learning* disebut sebagai fitur [23]. Contoh isi dari file *nlu.yml* dapat dilihat di Gambar 2.

```
nlu:
- intent: start_session
  examples: |
    - /start

- intent: greet
  examples: |
    - Halo
    - Hai
    - Selamat pagi
    - Selamat siang
    - Selamat sore
    - Selamat malam
    - Apa kabar?
    - Hi

- intent: ask_basiswa_jenis
  examples: |
    - Apa saja jenis beasiswa yang tersedia di UKDM?
    - Apa saja macam beasiswa yang ada di UKDM?
    - Apa saja tipe beasiswa yang tersedia di UKDM?
    - Apa saja jenis program beasiswa yang tersedia di UKDM?
    - Apa saja beasiswa jenis yang tersedia di UKDM?
    - Apa jenis beasiswa yang tersedia di UKDM?
    - Sebutkan macam-macam beasiswa yang disediakan di UKDM?
    - Di UKDM, beasiswa pilihannya apa aja?
    - Apa jenis beasiswa di UKDM?
    - Info beasiswa ukdw
    - Apa saja jenis beasiswa yang ditawarkan di UKDM?
```

Gambar 2. *nlu.yml*

2. *domain.yml* : merupakan tempat untuk mendefinisikan input dan output dari RASA. Domain mendefinisikan semua hal yang diketahui dan dapat dilakukan oleh RASA. Ini merupakan file konfigurasi pusat yang menjadi jembatan antara RASA NLU dan RASA Core[21]. Terdapat bagian *intent* yang berfungsi untuk mendefinisikan *intent* yang ada di file *nlu.yml*, dan bagian *response* yang berfungsi untuk mendefinisikan jawaban-jawaban dari *intent* yang ada. Jawaban dari *intent* biasanya didefinisikan dengan penulisan “utter_[intent]” Contoh isi dari file *domain.yml* dapat dilihat di Gambar 3.

```
intents:
- ask_basiswa_jenis
- ask_basiswa_mahasiswa_aktif
- ask_basiswa_mahasiswa
- ask_basiswa_kriteria_pendaftaran
- ask_basiswa_periode_pendaftaran
- ask_basiswa_syarat_umum
- ask_basiswa_cara_pendaftaran
- ask_basiswa_mahasiswa_baru
- ask_basiswa_definisi_pembiaran_parsial
- ask_basiswa_proses_selksi
- ask_basiswa_syarat_ipk_minimal
- ask_basiswa_dukungan_penerima
- ask_basiswa_kontak_informasi
- ask_dana_kegiatan_prosedur_pengajuan
- ask_dana_kegiatan_dukungan_lomba_eksternal
- ask_dana_kegiatan_waktu_pencairan
- ask_dana_kegiatan_kewajiban_ipj
- greet
- nlu_fallback
- start_session

responses:
utter_ask_basiswa_jenis:
- text: |
  Di UKDM tersedia beberapa jenis beasiswa, antara lain:
  - UKDM Scholarship: Beasiswa penuh selama 8 semester untuk maha
  - Beasiswa Talenta Duta Bangsa: Beasiswa penuh untuk mahasiswa
  - Beasiswa KIP Kuliah: Beasiswa penuh untuk mahasiswa baru dari
  - Beasiswa Afirmasi Pendidikan (ADIK): Untuk mahasiswa baru as
  - Beasiswa Scranton: Untuk mahasiswa berprestasi (Sarjana & Pas
  - Lainnya untuk mahasiswa aktif: Beasiswa Bank BPD DIY, Beasis
```

Gambar 3 domain.yml

3. *rules.yml* : merupakan tempat yang digunakan untuk mendefinisikan aturan yang dipakai untuk manajemen dialog percakapan sistem. Rules mendeskripsikan potongan-potongan aturan yang harus selalu mengikuti alur yang sama [21]. Contoh isi dari file *rules.yml* dapat dilihat di Gambar 4

```
rules:
- rule: Menjawab jenis-jenis beasiswa
  steps:
  - intent: ask_basiswa_jenis
  - action: utter_ask_basiswa_jenis

- rule: Menjawab beasiswa untuk mahasiswa aktif
  steps:
  - intent: ask_basiswa_mahasiswa_aktif
  - action: utter_ask_basiswa_mahasiswa_aktif
```

Gambar 4 rules.yml

4. *stories.yml* : merupakan tempat untuk mendefinisikan dialog manajemen yang lebih dalam. Bagian ini merupakan representasi percakapan antara pengguna dan *chatbot*.

```
stories:
# Multi-turn stories (>= 3 langkah) yang berawal dari pertanyaan
- story: Alur jenis beasiswa → minta detail → minta syarat → minta
  steps:
  - intent: ask_beasiswa_jenis
  - action: utter_ask_beasiswa_jenis
  - intent: ask_beasiswa_detail_jenis
  - action: utter_ask_beasiswa_detail_jenis
  - intent: ask_beasiswa_syarat_umum
  - action: utter_ask_beasiswa_syarat_umum
  - intent: ask_beasiswa_periode_pendaftaran
  - action: utter_ask_beasiswa_periode_pendaftaran
  - intent: goodbye
  - action: utter_goodbye
```

Gambar 5 *stories.yml*

5. *config.yml* : merupakan tempat untuk mendefinisikan komponen-komponen yang akan digunakan untuk melatih model dan memproses *input* pengguna dengan cara menjalankan tiap komponen satu per satu dan mengembalikan *output* yang terstruktur [19]. Contoh isi dari file *config.yml* dapat dilihat di Gambar 5.

6.

```
pipeline:
- name: WhitespaceTokenizer
- name: CountVectorsFeaturizer
  analyzer: word
  min_ngram: 1
  max_ngram: 5
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 5
- name: LogisticRegressionClassifier
  max_iter: 500
- name: FallbackClassifier
  threshold: 0.7

policies:
- name: RulePolicy
```

Gambar 6 *config.yml*

Ccnfig.yml akan mendefinisikan komponen-komponen yang di pakai di penelitian ini. Berikut komponen-komponen yang dipakai[18] :

- *Whitespace Tokenizer* : komponen yang digunakan untuk membagi teks menjadi token individu berdasarkan spasi. Outputnya adalah *token*. Contoh jika input pengguna adalah “apa jenis beasiswa yang ada”, outputnya adalah [“apa”, “jenis”, “beasiswa”, “yang”, “ada”]
- *Count Vector Featurizer* : komponen yang digunakan untuk menghasilkan fitur untuk klasifikasi *intent*. Komponen ini mengubah hasil tokenisasi menjadi representasi *Bag-of-Words (BoW)* dengan cara menghitung frekuensi kemunculan setiap kata dan kalimat. Hasil perhitungan tersebut dikonversi menjadi vektor numerik yang merepresentasikan setiap contoh teks berdasarkan jumlah komunculan tokennya. Output dari fitur ini adalah *sparse features* . Contoh jika hasil tokenisasi adalah [“apa”, “jenis”, “beasiswa”, “yang”, “ada”], outputnya adalah [1, 1, 1, 1, 1]. Panjang vektor dan nilai dari tiap indeks di vektor tergantung dari banyaknya data yang ditokenisasi dan bervariasi.
- *Logistic Reggresion Classifier* : komponen yang digunakan untuk klasifikasi *intent* dalam sistem

RASA. Model ini dirancang untuk bekerja menggunakan *sparse_feature*, seperti representasi *bag-of-words* hasil dari komponen sebelumnya(*Count Vector Featurizer*).

- *Fallback Classifier* : komponen yang digunakan untuk memproses output jika probabilitas(dalam konteks RASA sering disebut *confidence*) yang dihasilkan oleh algoritma klasifikasi rendah. Saat probabilitas prediksi *intent* berada di bawah ambang batas(*threshold*) yang ditentukan atau terjadi ambiguitas antara dua *intent* teratas, komponen ini akan memetakan pesan pengguna menjadi *intent* khusus *nlu_fallback*. Contoh nilai *threshold* di set ke 0.7. jika hasil klasifikasi untuk pesan “apa jenis beasiswa yang ada” menghasilkan probabilitas di kelas jenis_beasiswa=0.69, kelas greet=0.50, dan kelas goodbye=0.60, maka karena nilai probabilitas tertinggi (0.69) < 0.7, pesan tersebut akan dianggap *fallback* dan dipetakan ke *intent nlu_fallback*.
 - *Rule Policy* : komponen yang digunakan untuk menangani bagian percakapan yang mengikuti aturan tetap. Komponen ini membuat prediksi berdasarkan aturan-aturan yang telah ditetapkan dalam file *rules.yml*.
7. *credentials.yml* : merupakan bagian dari *RASA Open Source* yang berfungsi sebagai jembatan komunikasi antara *RASA Open Source* dengan aplikasi pesan instan *Telegram*. File ini berisi konfigurasi-konfigurasi yang digunakan untuk komunikasi dengan Telegram seperti access token yang merupakan token unik dari telegram, verify yang merupakan nama pengguna, dan url webhook yang digunakan untuk untuk menerima update pesan dari Telegram dan meneruskannya ke *RASA Open Source* agar dapat memproses serta menanggapi pesan pengguna secara *real-time*[22]. Contoh isi dari file *credentials.yml* dapat dilihat di Gambar 6.

```
telegram:
  access_token: "490161424:AAGlRxinBRtKgb21_rLOEmtDFZMXB168
  verify: "your_bot"
  webhook_url: "https://your_url.com/webhooks/telegram/webhook"
```

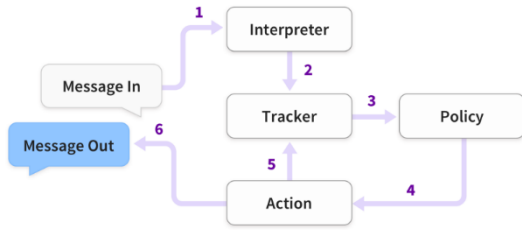
Gambar 2.7 *credentials.yml*

E. Alur Kerja Chatbot menggunakan RASA

Berikut merupakan alur kerja RASA seperti pada Gambar 2.7 [4]:

1. Pesan diterima dan diteruskan ke *Interpreter* untuk mengekstrak dan memprediksi *intent*(*RASA NLU*).
2. *Tracker* mempertahankan status percakapan. Ia menerima pemberitahuan bahwa pesan baru telah diterima(*RASA Core*).
3. *Policy* menerima status terkini dari *Tracker*(*RASA Core*).
4. *Policy* menentukan tindakan apa yang akan diambil selanjutnya berdasarkan *intent* yang diprediksi(*RASA Core*).
5. *Action* yang dipilih dicatat oleh *Tracker*(*RASA Core*).

6. *Action* tersebut dieksekusi (mengirim pesan kepada pengguna) (*RASA Core*).



Gambar 8 Alur kerja RASA Open Source

F. Webhook

Webhook merupakan salah satu metode yang dipakai untuk membangun komunikasi di *Bot Engine Telegram*. Metode ini mengirimkan data langsung ke pihak ketiga segera setelah *Bot Engine Telegram* menerima informasi tersebut [24]. *RASA Open Source* menggunakan *Webhook* untuk berkomunikasi dengan *Telegram*.

G. Docker

Docker memungkinkan aplikasi dikemas dan dijalankan dalam lingkungan terisolasi yang disebut *container*. *Container* bersifat ringan karena tidak memerlukan beban tambahan dari *hypervisor*, melainkan berbagi kernel mesin *host* dengan batasan ketat terhadap sumber daya mesin yang bisa mereka lihat atau gunakan [15].

H. Confusion Matrix

Confusion matrix merupakan alat evaluasi dalam bentuk tabel matriks yang dipakai untuk menggambarkan hasil kinerja model dengan membandingkan hasil prediksi dengan label asli dari data. Tabel confusion matrix dapat dilihat di Gambar 8[12].

		Actual Values	
		1 (Positive)	0 (Negative)
Predicted Values	1 (Positive)	TP (True Positive)	FP (False Positive) Type I Error
	0 (Negative)	FN (False Negative) Type II Error	TN (True Negative)

Gambar 9 Tabel Confusion Matrix

Ada 4 bagian yang dipakai sebagai representasi hasil di tabel tersebut. Berikut merupakan penjelasan 4 representasi hasil tersebut [12]:

- **True Positive (TP)** : merupakan data positif yang diprediksi benar sebagai positif. Contohnya *input* pengguna “apa saja jenis beasiswa” dengan kelas *jenis_beasiswa*, dan model yang dibuat memprediksi bahwa input tersebut merupakan kelas *jenis_beasiswa*.
- **True Negative (TN)** : merupakan data negatif yang diprediksi benar sebagai negatif . Contohnya *input* pengguna “apakah dana kegiatan wajib lpij” dengan kelas *dana_wajib_lpij*, dan model yang dibuat memprediksi bahwa input tersebut merupakan kelas *dana_wajib_lpij*, yang bukan kelas *jenis_beasiswa*.
- **False Positive (FP)** : merupakan data negatif yang salah diprediksi sebagai data positif. Contohnya *input*

pengguna “apa saja jenis beasiswa” dengan kelas *jenis_beasiswa*, dan model yang dibuat memprediksi bahwa input tersebut merupakan kelas *syarat_beasiswa*.

- **False Negative (FN)** : merupakan data positif yang salah diprediksi sebagai data negatif. Contohnya *input* pengguna “apa saja jenis beasiswa” dengan kelas *jenis_beasiswa*, dan model yang dibuat memprediksi bahwa input tersebut merupakan kelas *dana_kegiatan*.

Tabel *confusion matrix* digunakan untuk menghitung *performance metrics* untuk pengukuran dan evaluasi model.. Berikut merupakan penjelasan beberapa *performance metrics* yang dipakai [12]:

- **Akurasi** : merupakan gambaran seberapa akurat model untuk klasifikasi. Akurasi adalah rasio prediksi benar di keseluruhan data. Akurasi didapatkan dengan Rumus 2.1.

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

- **Recall** : merupakan gambaran keberhasilan model dalam menemukan informasi. Recall adalah prediksi benar positif yang dibandingkan dengan data keseluruhan yang benar positif. Recall didapatkan dengan Rumus 2.2.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.2)$$

- **Presisi** : merupakan gambaran tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Presisi adalah rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Presisi didapatkan dengan Rumus 2.3.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.3)$$

Selain yang disebutkan sebelumnya, terdapat model lain yang digunakan untuk mengukur dan mengevaluasi matriks [10] :

- **Skor-F1** : merupakan rata-rata nilai harmonik dari presisi dan recall untuk klasifikasi. Skor-F1 didapatkan dengan Rumus 2.4.

$$\text{Skor-F1} = \left(\frac{\text{Recall}^{-1} + \text{Presisi}^{-1}}{2} \right) = 2 \frac{\text{Presisi} \cdot \text{Recall}}{\text{Presisi} + \text{Recall}} \quad (2.4)$$

- **Cross-Validation** : Teknik untuk menguji generalisasi model dengan membagi data menjadi beberapa bagian (*fold*) untuk menguji dan melatih model secara bergantian. Tujuannya memastikan model tidak hanya menghafal data latih, tapi juga bisa bekerja baik pada data baru. Contoh jika nilai *fold* = 3, dan data latih = 600 maka tiap *fold* berukuran 200 data latih ; *Fold 1* : 1-200, *Fold 2* : 201-400, *Fold 3* : 401-600. Iterasi pertama, *Fold 1* menjadi data uji ; *Fold 2* dan *Fold 3* menjadi data latih. Iterasi kedua *Fold 2* menjadi data uji ; *Fold 1* dan *Fold 3* menjadi data latih. Iterasi ketiga *Fold 3* menjadi data uji ; *Fold 1* dan *Fold 2* menjadi data latih. Hasil perhitungannya dengan melihat rata-rata perhitungan matriks masing masing *fold*. Contoh perhitungan matriks *Accuracy* dengan *fold* = 3 dapat dilihat di Rumus 2.5.

$$\text{Avr} = \frac{\text{Acc Fold 1} + \text{Acc Fold 2} + \text{Acc Fold 3}}{3} \quad (2.5)$$

I. Black Box Testing

Black box testing adalah salah satu metode untuk menguji sistem yang berfokus pada input-output tanpa melihat bagian dalam sebuah sistem. Ada beberapa jenis teknik yang dipakai metode ini, salah satunya yaitu *use-case*. Teknik ini digunakan untuk merancang kasus uji berdasarkan skenario interaksi pengguna dengan sistem untuk mencapai tujuan tertentu [13].

J. Easy Data Augmentation

Easy Data Augmentation (EDA) merupakan metode augmentasi teks yang bertujuan untuk menambah data dengan teknik-teknik yang ada di dalamnya. Ada 4 teknik yang bisa dipakai di metode ini seperti *Synonym Replacement (SR)* untuk mengganti kata dengan sinonimnya, *Random Insertion (RI)* untuk menyisipkan kata di dalam data secara acak, *Random Swap (RS)* untuk menukar urutan kata didalam data secara acak, dan *Random Deletion (RD)* untuk menghapus kata didalam data secara acak. Rumus yang dipakai untuk menentukan jumlah kata yang diubah berdasarkan teknik yang telah disebutkan sebelumnya dapat dilihat di Rumus 2.6. Parameter α yang akan menentukan jumlah perubahan kata sebanyak n didalam data [3].

$$n = \alpha \times l \quad (2.6)$$

Contoh di dalam data latih terdapat kalimat “apa jenis beasiswa yang ada”. Misal nilai $\alpha = 0.2$ maka:

$$l = 5$$

$$n = \alpha \times l = 0.2 \times 5 = 1 \text{ perubahan kata}$$

Synonym Replacement (SR) : ganti 1 kata dengan sinonimnya

= “apa macam beasiswa yang ada”

Random Insertion (RI) : sisipkan 1 kata acak dari kosakata
= “apa jenis beasiswa prestasi yang ada”

Random Swap (RS) : tukar posisi 1 pasang kata acak
= “jenis apa beasiswa yang ada”

Random Deletion (RD) : hapus 1 kata secara acak
= “apa jenis beasiswa ada”

III. METODOLOGI PENELITIAN

A. Pengumpulan Data

Tahap awal melibatkan pengumpulan pertanyaan yang sering diajukan (*FAQ*) di BIRO 3. Ini dilakukan melalui wawancara dengan kepala dan wakil kepala BIRO 3 untuk mengidentifikasi kegori pertanyaan umum dan identifikasi masalah yang relevan. Dilanjutkan dengan pengisian spreadsheet oleh 3 divisi layanan informasi BIRO 3 untuk mencatat detail informasi yang relevan dengan penelitian.

B. Pengolaan Data

Tahap selanjutnya yaitu pengolaan data yang dimulai dengan melakukan seleksi dan anotasi data. Data mentah diseleksi berdasarkan relevansi dengan kategori *FAQ* yang telah diidentifikasi. Setiap pertanyaan diberi label secara manual dengan intent spesifik menggunakan format ask_[kategori_awal]_[kategori_lanjutan] untuk dataset RASA. Outputnya adalah kolom kategori pertanyaan menjadi kelas dan kolom pertanyaan yang relevan menjadi

fitur di masing masing kelas. Berikut contoh data yang telah dianotasi. Contoh hasil anotasi dan seleksi data dapat dilihat di Tabel 1.

TABEL I
CONTOH HASIL ANOTASI DAN SELEKSI DATA

Fitur	Kelas
Apa saja jenis beasiswa yang tersedia di UKDW?	ask_beasiswa_jenis
Apakah ada beasiswa untuk mahasiswa aktif?	ask_beasiswa_mahasiswa_aktif
Apakah ada beasiswa khusus untuk mahasiswi?	ask_beasiswa_mahasiswi
Siapa saja yang bisa mendaftar beasiswa?	ask_beasiswa_kriteria_pen_daftar
Kapan periode pendaftaran beasiswa dibuka?	ask_beasiswa_periode_pen_daftaran

Setelah dianotasi dan diseleksi, data kemudian diaugmentasi menggunakan metode Easy Data Augmentation dengan teknik *Synonym Replacement (SR)*, *Random Insertion (RI)*, *Random Swap (RS)*, dan *Random Deletion (RD)*. Empat teknik tersebut dipakai dengan menggunakan kombinasi nilai α 0.25, 0.50, 0.75 untuk mendapatkan hasil augmentasi yang beragam. Teknik augmentasi data ini dilakukan secara manual, menghasilkan 20 ragam fitur untuk dijadikan dataset. Tabel 2 adalah contoh data hasil augmentasi pada kelas ask_beasiswa_jenis:

TABEL II
CONTOH AUGMENTASI DATA

Kelas	Fitur Awal	Tipe augmentasi dan nilai α	Fitur yang diaugmentasi	Keterangan
ask_beasiswa_jenis	Apa saja jenis beasiswa yang tersedia di UKDW?	$\alpha = 0.25$ – SR	Apa saja macam beasiswa yang ada di UKDW?	Perubahan 2 dari 8 kata.
		$\alpha = 0.25$ – SR	Apa saja tipe beasiswa yang tersedia di UKDW?	Perubahan 2 dari 8 kata.
		$\alpha = 0.25$ – RI	Apa saja jenis program beasiswa yang tersedia di UKDW?	Perubahan 1 kata.

C. Pengembangan Sistem

Tahap ini dilakukan untuk merancang sistem menggunakan *RASA Open Source*, yang meliputi penyusunan data latih, pemilihan *pipeline*, pelatihan, pengujian dan validasi performa model oleh sistem, penyusunan konfigurasi kredensial *Telegram*, *deployment* ke

kanal komunikasi *Telegram* menggunakan *Docker*, dan pengujian oleh pengguna. Berikut detailnya:

1. *Penyusunan Data Latih* : Bagian ini akan menjelaskan tentang penyusunan data latih di *RASA Open Source* yang meliputi beberapa file seperti *nlul.yml*, *rules.yml*, dan *domain.yml*.

- File *nlul.yml* digunakan untuk mendefinisikan *dataset*. Kelas dan fitur disusun dalam format *YAML* di file *nlul.yml* untuk menjadi *dataset*. Kelas ditulis sebagai *intent*, sedangkan fitur akan ditulis sebagai *examples*.
- File *rules.yml* digunakan untuk mendefinisikan aturan-aturan yang akan didefinisikan oleh *RASA Open Source*. jika suatu *intent* terdeteksi, maka asisten akan langsung menjalankan *action* yang sesuai dengan *intent*.
- File *domain.yml* digunakan sebagai pusat dari manajemen dialog *RASA Open Source*. kelas maupun respons yang akan sistem berikan kepada user semua didefinisikan disini.

2. *Pemilihan Pipeline* : Bagian ini akan menjelaskan tentang pemilihan *pipeline* di *RASA Open Source* yang meliputi file *config.yml*. Dalam file *config.yml* terdapat *pipeline* mendefinisikan serangkaian komponen yang akan memproses *input* teks mentah dari pengguna secara berurutan untuk memahaminya. Setiap komponen dalam *pipeline* mengambil *output* dari komponen sebelumnya sebagai *inputnya*. Komponen yang dipakai adalah sebagai berikut :

- *WhitespaceTokenizer* dengan fungsi untuk Menghapus tanda baca dan simbol, Memecah teks menjadi *token* berdasarkan spasi, Menghapus token yang merupakan emoji, Menghapus *token* kosong yang mungkin dihasilkan.
- *CountVectorFeaturizer* dengan fungsi membuat gabungan *bag-of-word* dari semua fitur kata dan karakter unik dari setiap kata unik dari hasil tokenisasi sebelumnya. Menghasilkan *vocabulary* gabungan fitur tersebut. Semua diatur dalam parameter *word* untuk kata, *char_wb* untuk karakter, dan *n-gram* bawaan komponen tersebut.
- *LogisticRegressionClassifier* dengan fungsi mengklasifikasi setiap *intent* berdasarkan *vocabulary* yang dibuat oleh komponen *CountVectorFeaturizer*.
- *FallbackClassifier* dengan fungsi untuk menangani pesan yang tidak bisa dipahami oleh sistem dengan cara memeriksa skor *confidence/probabilitas* yang dihasilkan oleh model.
- *RulePolicy* dengan fungsi mengatur alur percakapan berdasarkan aturan-aturan yang sudah didefinisikan di file *rules.yml*.

Pada file ini dilakukan pemilihan *pipeline* untuk melatih model. Pemilihan *pipeline* dilakukan dengan membuat 4 kombinasi *pipeline* berdasarkan parameter dan komponen yang tersedia. 4 konfigurasi ini meliputi kombinasi dari parameter *n-gram* di komponen *CountVectorFeaturizer* dan juga komponen *WhitespaceTokenizer default* yang

dibandingkan dengan komponen *custom WhitespaceTokenizer* yang dibuat peneliti dengan menambahkan fungsi *stemmer*. Komponen lain selain *CountVectorFeaturizer* dan *WhitespaceTokenizer* dibiarkan default. *Pipeline* dipilih dari hasil perbandingan hasil evaluasi dan test akurasi, presisi, skor-f1 pada data validasi dan data uji masing-masing hasil pelatihan model dengan *pipeline* menggunakan metode *cross-validation*. Untuk lebih detail, berikut tabel-tabel perbandingan 4 kombinasi *pipeline* tersebut.

TABEL III
PIPELINE 1

Komponen	Fungsi	Parameter
WhitespaceTokenizer	Menghapus tanda baca dan simbol	-
	Memecah teks menjadi token berdasarkan spasi	-
	Menghapus token yang merupakan emoji	-
	Menghapus token kosong yang mungkin dihasilkan	-
CountVectorFeaturizer	bag-of-word dari semua fitur kata	Word, Min-gram = 1, Max-gram = 3
CountVectorFeaturizer	bag-of-word dari semua fitur karakter	Char_wb, Min-gram = 1, Max-gram = 3

TABEL IV
PIPELINE 2

Komponen	Fungsi	Parameter
WhitespaceTokenizer	Menghapus tanda baca dan simbol	-
	Memecah teks menjadi token berdasarkan spasi	-
	Menghapus token yang merupakan emoji	-
	Menghapus token kosong yang mungkin dihasilkan	-
	Mengubah kata ke bentuk dasarnya	
CountVectorFeaturizer	bag-of-word dari semua fitur kata	Word, Min-gram = 1, Max-gram = 3
CountVectorFeaturizer	bag-of-word dari semua fitur karakter	Char_wb, Min-gram = 1, Max-gram = 3

TABEL V
PIPELINE 3

Pipeline	Komponen	Fungsi	Parameter
Pipeline 3	Whitespac eTokenize r	Menghapus tanda baca dan simbol	-
		Memecah teks menjadi token berdasarkan spasi	-
		Menghapus token yang merupakan emoji	-
		Menghapus token kosong yang mungkin dihasilkan	-
	CountVec torFeatur er	bag-of-word dari semua fitur kata	Word, Min- gram = 1, Max-gram = 5
	CountVec torFeatur er	bag-of-word dari semua fitur karakter	Char_wb , Min-gram = 1, Max-gram = 5

TABEL VI
PIPELINE 4

Pipeline	Komponen	Fungsi	Parameter
Pipeline 4	Whitespac eTokenize r	Menghapus tanda baca dan simbol	-
		Memecah teks menjadi token berdasarkan spasi	-
		Menghapus token yang merupakan emoji	-
		Menghapus token kosong yang mungkin dihasilkan	-
		Mengubah kata ke bentuk dasarnya	
	CountVec torFeatur er	bag-of-word dari semua fitur kata	Word, Min- gram = 1, Max-gram = 5
	CountVec torFeatur er	bag-of-word dari semua fitur karakter	Char_wb , Min-gram = 1, Max-gram = 5

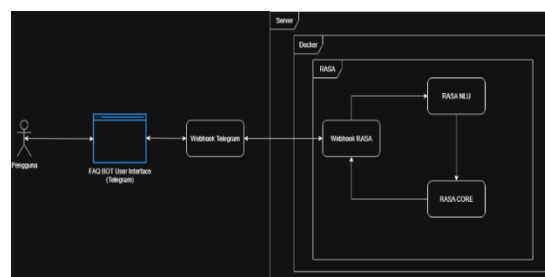
3. *Pelatihan, Pengujian dan Validasi Performa Model oleh RASA* : Bagian ini akan menjelaskan pelatihan, pengujian, dan validasi performa model oleh *RASA Open Source*. Model dilatih menggunakan perintah “*rasa train*” dengan data dan *pipeline* yang telah dikonfigurasi. Setelah pelatihan selesai, langkah selanjutnya yaitu menguji dan memvalidasi model menggunakan metode *cross-validation* dengan perintah

di “*rasa test nlu –cross-validation*” dengan nilai *fold default*.

4. *Penyusunan konfigurasi kredensial Telegram* : Bagian ini akan menjelaskan tentang konfigurasi kredensial *Telegram* di *RASA Open Source* yang terdapat di *file credentials.yml*. Fungsinya adalah untuk menyimpan semua konfigurasi yang dibutuhkan oleh *RASA Open Source* untuk terhubung dengan *Telegram*. Terdapat parameter seperti *token*, *verify*, dan *webhooks*.
5. *Deployment ke Telegram menggunakan Docker* : Setelah dikembangkan, sistem *FAQ Bot* diintegrasikan ke dalam *container Docker*. *Container Docker* kemudian ini dijalankan di *server* sehingga *FAQ Bot* dapat beroperasi secara stabil dan dapat diakses oleh pengguna. Dengan integrasi ini, sistem dapat menerima pesan dari pengguna *Telegram* dan mengirimkan *respons* kembali melalui antarmuka *Telegram*.

D. Diagram Alir Arsitektur Sistem

Gambar 10 menunjukkan diagram alir arsitektur sistem yang dibangun, di mana pengguna berinteraksi dengan *FAQ Bot* melalui antarmuka *Telegram*, kemudian semua pesan dikirim ke *webhook Telegram* yang meneruskannya ke *webhook RASA* pada *container Docker*. Di sana, *RASA NLU* melakukan klasifikasi *intent*, setelah itu *RASA Core* menentukan alur dialog dan respons yang sesuai sebelum mengirimkannya kembali melalui *webhook Telegram* ke pengguna.



Gambar 10 Diagram alir arsitektur sistem

E. Pengujian Sistem

Pengujian dilakukan dengan memberikan skenario kepada responden untuk menguji apakah sistem dapat memberikan output yang diharapkan berdasarkan input responden. Hasil pengujian kemudian di evaluasi secara manual untuk melakukan perbandingan antara output yang dihasilkan sistem dan output yang diharapkan. Data yang dievaluasi adalah data pengujian yang telah dilakukan sebelumnya. Data ini memuat informasi yang terdiri dari inputan pengguna dan output yang diberikan sistem berdasarkan prediksi input pengguna. Proses evaluasi performa menggunakan *performance metrics*, dimana pada bagian ini dilakukan perhitungan metrik akurasi. Berikut merupakan tabel skenario pengujian yang terdapat di Tabel 7.

TABEL VII

SKENARIO PENGUJIAN

Skenario pengujian	Input Pengguna	Output yang Diharapkan	Output
Membuka sesi awal dengan chatbot		start_session	
Membuka/ Memulai Percakapan		greetings	
Bertanya Jenis Beasiswa		beasiswa_jenis	
Bertanya Beasiswa untuk Mahasiswa Aktif		beasiswa_mahasiswa_aktif	
Bertanya Beasiswa untuk Mahasiswi		beasiswa_mahasiswi	
Bertanya Kriteria Pendaftaran Beasiswa		beasiswa_kriteria_pendaftaran	
Bertanya Periode Pendaftaran Beasiswa		beasiswa_periode_pendaftaran	
Bertanya Syarat Umum Beasiswa		beasiswa_syarat_umum	
Bertanya Cara Mendaftar Beasiswa		beasiswa_cara_pendaftaran	
Bertanya Perbedaan Beasiswa Penuh dan Parsial		beasiswa_definisi_penuh_parsial	
Bertanya Proses Seleksi Beasiswa		beasiswa_proses_seleksi	
Bertanya syarat IPK minimal Beasiswa		beasiswa_syarat_ipk_minimal	
Bertanya Pendampingan Penerima Beasiswa		beasiswa_dukungan_penerima	
Bertanya Kontak Informasi Beasiswa		beasiswa_kontak_informasi	
Bertanya Prosedur Pengajuan Dana Kegiatan		dana_kegiatan_prosedur_pengajuan	
Bertanya Dukungan Dana untuk lomba eksternal		dana_kegiatan_dukungan_lomba_eksternal	
Bertanya Waktu Pencairan Dana Kegiatan		dana_kegiatan_waktu_pencairan	
Bertanya Kewajiban LPJ Dana Kegiatan		dana_kegiatan_kewajiban_lpj	
Berantanya Diluar Konteks		nlu_fallback	
Berantanya Beasiswa untuk Mahasiswa Baru		beasiswa_mahasiswa_baru	

telah diketahui, maka ada aksi yang berupa respons yang sesuai dengan *intent* tersebut. Respons tersebut disimpan di dalam *domain.yml* yang berisi 17 data jawaban dari masing-masing *intent*. Berikut contoh penyusunan data latih untuk kelas *ask_jenis_beasiswa* dalam di file *nlu.yml* :

- intent: ask_beasiswa_jenis
examples: |
- Apa saja jenis beasiswa yang tersedia di UKDW?
- Apa saja tipe beasiswa yang tersedia di UKDW?
- Sebutkan macam-macam beasiswa yang disediakan di UKDW?
- Di UKDW, beasiswa pilihannya apa aja?
- Apa jenis beasiswa di UKDW?
- info beasiswa ukdw
- UKDW menawarkan beasiswa apa saja?
- Tolong jelaskan pilihan beasiswa di UKDW.
- Kalau mau cari beasiswa di UKDW, pilihannya apa aja ya?
- opsi beasiswa ukdw
- Info dong, beasiswa UKDW itu ada apa aja sih?
- list beasiswa ukdw
- jenis beasiswa

Berikut contoh penyusunan *rules* untuk intent *ask_beasiswa_jenis* :

- rule: Menjawab jenis-jenis beasiswa
steps:
- intent: ask_beasiswa_jenis
- action: utter_ask_beasiswa_jenis

Berikut contoh penyusunan *domain* untuk intent *ask_beasiswa_jenis* :

intents:
- ask_beasiswa_jenis
responses:
utter_ask_beasiswa_jenis:
- text: |
Di UKDW tersedia beberapa jenis beasiswa, antara lain:
- UKDW Scholarship: Beasiswa penuh selama 8 semester untuk mahasiswa baru dengan prestasi luar biasa.
- Beasiswa Talenta Duta Wacana: Beasiswa penuh untuk mahasiswa baru dari keluarga tidak mampu.
- Beasiswa KIP Kuliah: Beasiswa penuh untuk mahasiswa baru dari keluarga kurang mampu.
- Beasiswa Afirmasi Pendidikan (ADiK): Untuk mahasiswa baru asal Papua, Papua Barat, anak TKI, dan daerah 3T.
- Beasiswa Scranton: Untuk mahasiswi berprestasi (Sarjana & Pascasarjana).
- Lainnya untuk mahasiswa aktif: Beasiswa Bank BPD DIY, Beasiswa ADARO, Beasiswa Prestasi Akademik, dll.

Berikut contoh penyusunan *stories* untuk intent *ask_beasiswa_jenis* :

stories:
- story: Alur jenis beasiswa → minta detail → minta syarat → minta periode pendaftaran → closing
steps:
- intent: ask_beasiswa_jenis

IV. HASIL DAN ANALISIS

A. Implementasi Sistem

Bagian ini akan menjelaskan implementasi dan konfigurasi dari setiap komponen yang dibutuhkan di *RASA Open Source* :

1. *Penyusunan Data Latih* : Pada penelitian ini penyusunan data latih terdiri dari tiga *files* yaitu *nlu.yml*, *rules.yml*, dan *domain.yml*. *nlu.yml* mendefinisikan 17 *intent* yang dipakai untuk *dataset*. *rules.yml* mendefinisikan 17 *rules* untuk manajemen dialog berbasis aturan di sistem. Jika suatu intent

- action: utter_ask_basiswa_jenis
- intent: ask_basiswa_detail_jenis
- action: utter_ask_basiswa_detail_jenis
- intent: ask_basiswa_syarat_umum
- action: utter_ask_basiswa_syarat_umum
- intent: ask_basiswa_periode_pendaftaran
- action: utter_ask_basiswa_periode_pendaftaran
- intent: goodbye
- action: utter_goodbye

2. *Pemilihan Pipeline* : Bagian ini akan menjelaskan implementasi dari pemilihan pipeline yang dikonfigurasi untuk membangun model di file *config.yml*. Berikut merupakan hasil evaluasi akurasi, presisi, skor-f1 menggunakan *cross-validation* dengan parameter default (*folds* = 5) dari 4 kombinasi pipeline yang dibuat di Tabel 3, Tabel 4, Tabel 5, dan Tabel 6. Hasilnya dapat dilihat di Tabel 4.1.

TABEL VIII
PERBANDINGAN PIPELINE

Pipeline	Akurasi		F1-skor		Presisi	
	Validasi	Test	Validasi	Test	Validasi	Test
1	1.000	0.898	1.000	0.896	1.000	0.912
2	1.000	0.901	1.000	0.899	1.000	0.920
3	1.000	0.879	1.000	0.875	1.000	0.897
4	1.000	0.915	1.000	0.912	1.000	0.928

Hasil perbandingan di Tabel 8 menunjukkan bahwa *pipeline 4* mempunyai nilai train dan test tertinggi, maka dari itu model dibangun menggunakan *pipeline 4* di Tabel 9.

TABEL IX
PIPELINE 4

Komponen	Fungsi	Parameter
WhitespaceTokenizer	Menghapus tanda baca dan simbol	-
	Memecah teks menjadi token berdasarkan spasi	-
	Menghapus token yang merupakan emoji	-
	Menghapus token kosong yang mungkin dihasilkan	-
	Mengubah kata ke bentuk dasarnya	-
CountVectorFeaturizer	bag-of-words dari semua fitur kata	Word, Min-gram = 1, Max-gram = 5
CountVectorFeaturizer	bag-of-words dari semua fitur karakter	Char_wb, Min-gram = 1, Max-gram = 5

Berikut merupakan penulisan untuk konfigurasi *file config.yml* menggunakan *pipeline 4* dan komponen *default* lainnya :

```
pipeline:
- name: ModifiedWhitespaceTokenizer
- name: CountVectorsFeaturizer
  analyzer: word
  min_ngram: 1
  max_ngram: 5
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 5
- name: LogisticRegressionClassifier
  max_iter: 500
- name: FallbackClassifier
  threshold: 0.7
```

```
policies:
- name: RulePolicy
```

3. *Pelatihan Model oleh RASA* : Bagian ini akan menjelaskan pelatihan, pengujian, dan validasi performa model oleh *RASA Open Source*. Setelah pipeline dipilih dan data pelatihan siap, langkah selanjutnya adalah melatih model *RASA Open Source*. Model dilatih menggunakan perintah "*rasa train*". Perintah ini menginstruksikan *RASA Open Source* untuk memvalidasi semua file data (*nlu.yml*, *rules.yml*) dan konfigurasi (*domain.yml*, *config.yml*), melatih model berdasarkan *nlu.yml* dan pipeline di *config.yml*, melatih model dialogue management (*RASA Core*) berdasarkan *rules.yml*, *domain.yml*, dan *policies* di *config.yml*. Selama proses pelatihan, *RASA Open Source* menampilkan *log* yang berisi informasi tentang kemajuan pelatihan *NLU* dan *Core*. Hasil dari tahap ini adalah sebuah file model *RASA* yang terkompresi (*.tar.gz*). Model ini berisi semua informasi yang diperlukan untuk memahami input pengguna (*NLU*) dan memutuskan bagaimana merespons (*Core*). Proses pelatihan dapat dilihat di Gambar 11

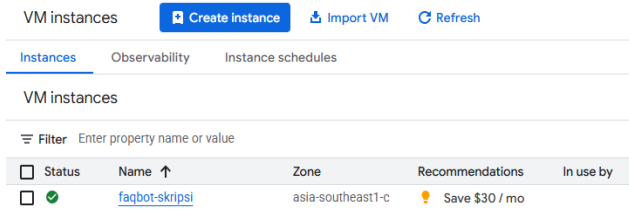
Gambar 11 proses pelatihan model

Penyusunan konfigurasi kredensial Telegram : Bagian ini merupakan implementasi konfigurasi kredensial *Telegram* di *RASA Open Source* yang terdapat di *file credentials.yml*.

- *credentials.yml* : Berikut penyusunan *credentials* untuk sistem *RASA Open Source* yang dibangun :

```
telegram:
access_token:"7522447101:AAHvFPPRH5HeDeH7tR8edZrEB-4-a1Q6spg"
verify:"BIRO3_FAQ_bot"
webhook_url:"https://rivaitalenggoran.site/webhooks/telegram/webhook"
```

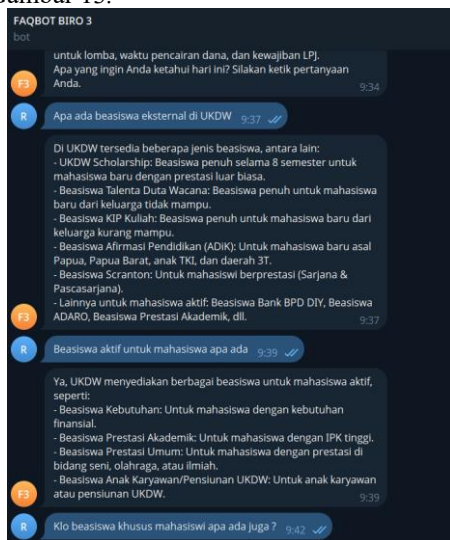
Deployment ke Telegram menggunakan Docker : Bagian ini merupakan penjelasan integrasi sistem. Sistem *FAQ Bot* yang telah dibangun menggunakan RASA kemudian diintegrasikan ke dalam *Container Docker*. *Container* yang telah dibangun kemudian disimpan di dalam *repository* di *dockerhub* untuk di *pull* ke dalam *virtual machine* di *Google Cloud*. Sistem dijalankan di *virtual machine* yang sudah memuat *container* sistem RASA di dalamnya. *Virtual machine* dapat dilihat di Gambar 12.



Gambar 12 Virtual Machine di Google Cloud

B. Implementasi Antarmuka

Antarmuka menggunakan *telegram*. Membuat *bot* Sistem ini menggunakan antarmuka berbasis *bot telegram*. *Bot* diinisiasi dengan mendaftarkannya menggunakan *botfather*. *Channel* dinamakan FAQBOT BIRO 3. *Channel bot* dapat dilihat di Gambar 13.



Gambar 13 Channel Bot Telegram

C. Pengujian dan Pembahasan

1. *Pengujian* : Pengujian sistem dilaksanakan dengan pembuatan 20 skenario untuk menguji *intent* dan output apa yang diharapkan dari skenario yang telah dibuat. Skenario dapat dilihat di Tabel 7. Pengujian dilaksanakan dengan 10 responden yang terdiri dari 6 Mahasiswa FTI UKDW dan 4 Staff BIRO 3. Pada pengujian ini, peneliti meminta setiap responden untuk memberikan bentuk pertanyaan yang variatif, seperti bentuk pertanyaan formal, informal, tipo atau tidak tipo, tetapi masih dalam lingkup *intent* yang diberikan. Ini dilakukan untuk menguji kemampuan sistem dalam menangani dan memahami variasi bentuk pertanyaan pengguna. Berikut ini merupakan hasil dari pengujian 10

responden dengan bentuk *confusion matrix* yang bisa dilihat pada Tabel 10.

TABEL X
CONFUSION MATRIX

Intent	TP	T N	FP	FN
start_session	10	0	0	0
greetings	10	0	0	0
beasiswa_jenis	9	0	0	1
beasiswa_mahasiswa_aktif	10	0	0	0
beasiswa_mahasiswi	9	0	1	0
beasiswa_kriteria_pendaftar	10	0	0	0
beasiswa_periode_pendaftaran	10	0	0	0
beasiswa_syarat_umum	10	0	0	0
beasiswa_cara_pendaftaran	10	0	0	0
beasiswa_mahasiswa_baru	10	0	0	0
beasiswa_definisi_penuh_parsial	10	0	0	0
beasiswa_proses_seleksi	9	0	1	0
beasiswa_syarat_ipk_minimal	9	0	0	1
beasiswa_dukungan_penerima	8	0	0	2
beasiswa_kontak_informasi	9	0	0	1
dana_kegiatan_prosedur_pengajuan	10	0	0	0
dana_kegiatan_dukungan_lomba_eksternal	10	0	0	0
dana_kegiatan_waktu_pencairan	9	0	0	1
dana_kegiatan_kewajiban_lpj	9	0	0	1
nlu_fallback	8	0	2	0
TOTAL	189	0	4	7

Hasil pengujian diatas dihitung menggunakan rumus akurasi pada Rumus 2.1 dan mendapatkan nilai sebagai berikut :

$$TP = 189, TN = 0, FP = 4, FN = 7$$

$$= TP + TN / TP + TN + FP + FN$$

$$= 189 + 0 / 189 + 0 + 4 + 7$$

$$= \mathbf{0.945 \text{ atau } 95 \%}$$

2. *Pembahasan* : Pengujian model NLU dilakukan terhadap 200 sampel data menghasilkan akurasi sebesar 95%. Hasil ini menunjukkan bahwa model yang dikembangkan memiliki kinerja yang cukup baik dan efektif dalam memahami pertanyaan pengguna dengan domain yang telah didefinisikan sebelumnya. Ini menandakan bahwa baik proses augmentasi, maupun proses pelatihan model dengan konfigurasi yang telah

ditetapkan cukup mampu menggeneralisasi input pengguna dengan cukup baik. Meskipun begitu, analisis *intent* diperlukan untuk mengidentifikasi area yang memerlukan perbaikan. Banyak *intent* mencapai hasil sempurna contohnya beasiswa_mahasiswa_aktif, tetapi juga ada beberapa kasus dimana model salah mengklasifikasikan *intent*, yang memberikan wawasan penting untuk perbaikan.

V. KESIMPULAN

Sistem layanan informasi *FAQ Bot* dengan tujuan untuk menjawab pertanyaan yang sering ditanyakan BIRO 3 telah berhasil dikembangkan. Dimulai dari pengembangan metodologi penelitian yang mendukung seperti pengumpulan dan pengolahan data, pengembangan sistem sampai ke pengujian sistem yang dilaksanakan dengan cukup baik. Sistem ini dapat dikatakan cukup mampu untuk memenuhi kebutuhan fungsional dari layanan informasi BIRO 3 berdasarkan hasil pengujian pengguna dengan tingkat akurasi sebesar 95%. Ada beberapa hal yang perlu menjadi pertimbangan kedepannya untuk pengembangan sistem. Seperti masih adanya beberapa pertanyaan yang tidak bisa dijawab oleh sistem, baik bentuknya berupa ketidakpahaman sistem, atau kesalahan sistem dalam memahami input pengguna. Ini mengindikasikan bahwa model masih dikatakan belum sempurna untuk menangani semua inputan.

UCAPAN TERIMA KASIH

Saya mengucapkan terima kasih yang sebesar-besarnya kepada Biro 3 atas dukungan administratif dan teknis yang sangat membantu selama proses penelitian ini. Terima kasih juga saya sampaikan kepada dosen pembimbing yang telah membimbing dengan penuh kesabaran dan memberikan arahan yang sangat berarti dalam setiap tahap penelitian. Kepada keluarga tercinta, saya sangat bersyukur atas doa, dukungan moral, dan semangat yang tak pernah putus. Tak lupa, kepada sahabat-sahabat yang selalu hadir memberi motivasi dan bantuan, baik dalam bentuk diskusi maupun dukungan emosional

DAFTAR PUSTAKA

- [1] AMINI, H., FARAHNAK, F., & KOSSEIM, L. (2019). FRONTIERS IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE. IN H. AMINI, F. FARAHNAK, & L. KOSSEIM, *FRONTIERS IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE* (pp. 35-55). MONTREAL: WORLD SCIENTIFIC.
- [2] Aziz, S. N., Nathasia, N. D., & Hassolthine, C. R. (2024). Pembuatan aplikasi chatbot faq untuk motionbank. *Journal of Information System, Applied, Management, Accounting and Research*, 118-127.
- [3] Azizah, A. N., Asy'ari, M. F., Prasatya, I. W., & Purwitasari, D. (2023). EASY DATA AUGMENTATION UNTUK DATA YANG IMBALANCE PADA KONSULTASI KESEHATAN DARING. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 1095-1104.
- [4] Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 1-9.
- [5] Caballero, M. (2021). A BRIEF SURVEY OF QUESTION. *International Journal of Artificial Intelligence & Applications*, 1-7.
- [6] Guntoro, Costaner, L., & Lisnawita. (2020). Aplikasi Chatbot untuk Layanan Informasi dan Akademik Kampus Berbasis Artificial Intelligence Markup Language (AIML). *Digital Zone: Jurnal Teknologi Informasi Dan Komunikasi*, 291-300.
- [7] Hien, H. T., Nguyen, C. P., Hoai, N. N., & Nhung, H. T. (2018). Intelligent Assistants in Higher-Education Environments: The FITEBot, a Chatbot for Administrative and Learning Support. *SoICT' 18: Ninth International Symposium on Information and Communication Technology* (pp. 1-8). Viet Nam: Association for Computing Machinery.
- [8] Hikmah, N., Ariyanti, D., & Pratama, F. A. (2022). Implementasi Chatbot Sebagai Virtual Assistant di Universitas Panca Marga Probolinggo menggunakan Metode TF-IDF. *Jurnal Teknologi Informasi dan Multimedia*, 133-148.
- [9] Ingale, N., Jha, T. A., Dixit, R., & Borate, V. K. (2020). College Enquiry Chatbot Using Rasa. *International Journal of Scientific Research in Science and Technology*, 210-215.
- [10] Kurniawati, Y. E. (2024, October 31). *Model Evaluation Metrics untuk Machine Learning*. Retrieved from Sis Binus: <https://sis.binus.ac.id/2024/10/31/model-evaluation-metrics-untuk-machine-learning/>
- [11] Mangotra, H., Dabas, V., Khetharpal, B., Verma, A., Shinghal, S., & Mohapatra, A. K. (2024). University Auto Reply FAQ Chatbot Using NLP and Neural Networks. *Artificial Intelligence and Applications*, 126-134.
- [12] Nugroho, K. S. (2019, November 13). *Confusion Matrix untuk Evaluasi Model pada Supervised Learning*. Retrieved from Medium: <https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f>
- [13] Pratama, S. D., Lasimin, & Dadaprawira, M. N. (2023). Pengujian Black Box Testing Pada Aplikasi Edu Digital Berbasis Website Menggunakan Metode Equivalence Dan Boundary Value. *Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD*, 560-569.
- [14] Raniloya, B. R., Raghuwanshi, N., & Singh, S. (2017). Chatbot for University Related FAQs. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1525-1530). Udupi: IEEE.
- [15] Senington, R., Pataki, B., & Wang, X. V. (2018). Using docker for factory system software management: Experience report. *procedia CIRP*, 659-664.
- [16] Sethi, F. (2020). FAQ (Frequently Asked Questions) ChatBot for Conversation. *International Journal of Computer Sciences and Engineering*, 1.
- [17] Sigdel, Y., Shrestha, S., & Neupane, N. (2024). College Chatbot Using RASA. *KEC Journal of Science and Engineering*, 39-43.
- [18] Vdb, M. (2025, 4 25). *Components*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/components>
- [19] Vdb, M. (2025, 4 25). *Model Configuration*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/model-configuration>
- [20] Vdb, M. (2025, 4 25). *Policies*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/policies>
- [21] Vdb, M. (2025, 4 25). *Rasa Glossary*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/glossary>
- [22] Vdb, M. (2025, 4 25). *Telegram*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/connectors/telegram>
- [23] Vdb, M. (2025, 4 25). *Training Data Format*. Retrieved from Rasa: <https://legacy-docs-oss.rasa.com/docs/rasa/training-data-format>
- [24] Widya, M. A., & Airlangga, P. (2020). PENGEMBANGAN TELEGRAM BOT ENGINE MENGGUNAKAN METODE WEBHOOK DALAM PENINGKATAN WAKTU LAYANAN E-GOVERNMENT. *SAINTEKBU*, 13-22.