JOURNAL OF
SOFTWARE ENGINEERING AND
COMPUTATIONAL INTELLIGENCE (JSECI)
INFORMATICS ENGINEERING, FACULTY OF COMPUTER SCIENCE
UNIVERSITAS INDO GLOBAL MANDIRI

Implementasi *Chatbot* Telegram Layanan Informasi Akademik Universitas Indo Global Mandiri Menggunakan *Framework* Rasa *Open Source*

Faturrachman[1], Muhammad Haviz Irfani[2*], Zaid Romegar Mair[3], Septa Cahyani[4], Muhammad Ikhwan Jambak[5]
[1,2,3,4,5]Teknik Informatika, Ilmu Komputer dan Sains, Universitas Indo Global Mandiri

, [1]faturrchmn10@gmail.com , [2]m.haviz@uigm.ac.id *, [3]zaidromegar@uigm.ac.id , [4]septacahyani@uigm.ac.id , [5]jambak@uigm.ac.id

ABSTRACT

The utilization of chatbot technology to provide academic information services has become increasingly popular within the higher education environment. This study aims to implement a Telegram-based chatbot for academic information services at Indo Global Mandiri University. The research utilizes the Rasa open source framework, enabling the development of a reliable and customizable chatbot tailored to user needs. The development methodology encompasses stages of design, development, testing, and implementation. The resulting chatbot efficiently delivers academic information instantly to users via the Telegram platform. Test results demonstrate the chatbot's ability to provide responses that align with user needs with a high level of accuracy. The implementation of the Telegram chatbot for academic information services at Indo Global Mandiri University is anticipated to enhance information accessibility and user experience in accessing university academic services.

Keywords: Chatbot, Telegram, Academic Information, Implementation, Rasa Open Source, Framework

ABSTRAK

Penggunaan teknologi chatbot dalam menyediakan layanan informasi akademik telah menjadi semakin populer dalam lingkungan perguruan tinggi. Studi ini bertujuan untuk mengimplementasikan sebuah chatbot berbasis Telegram untuk layanan informasi akademik di Universitas Indo Global Mandiri. Penelitian ini menggunakan framework Rasa open source, yang memungkinkan pengembangan chatbot yang dapat dipenggunalkan dan dapat disesuaikan dengan kebutuhan pengguna. Metode pengembangan yang digunakan meliputi tahap perancangan, pengembangan, pengujian, dan implementasi. Chatbot yang dihasilkan mampu memberikan informasi akademik secara cepat dan instan kepada pengguna melalui platform Telegram. Hasil pengujian menunjukkan bahwa chatbot mampu memberikan respons yang sesuai dengan kebutuhan pengguna dengan tingkat akurasi yang tinggi. Implementasi *chatbot* Telegram untuk layanan informasi akademik di Universitas Indo Global Mandiri diharapkan dapat meningkatkan aksesibilitas informasi dan pengalaman pengguna dalam mengakses layanan akademik universitas.

Kata kunci: *Chatbot*, Telegram, Academic Information, Implementation, *Rasa Open Source*, *Framework*

1. PENDAHULUAN

Perguruan saat ini memberikan layanan informasi secara langsung, menggunakan media sosial instagram, website, *facebook*, email, dan whatsapp. Permasalahan yang timbul pada pemberian layanan informasi khususnya berkaitan dengan akademik adalah respon komunikasi yang belum efisien saat menanggapi pertanyaan mahasiswa, sehingga mengakibatkan lambatnya proses penerimaan informasi oleh mahasiswa yang sangat dibutuhkan. Selain itu, upaya untuk meningkatkan kecepatan respon seringkali memerlukan alokasi biaya tambahan, juga dapat memberikan pengaruh pada anggaran dan ketersediaan sumber daya[1].

Chatbot merupakan program yang dapat digunakan untuk berkomunikasi melalui pesan teks dengan menyampaikan informasi dua arah dalam konsep request-response dari dan kepada pengirim pesan, karena chat berarti komunikasi berbasis teks dan bot berarti program yang diberi pengetahuan untuk memberikan respons kepada pengirim pesan. Pembuatan pola dan respons adalah prosedur umum untuk membuat chatbot. Pola atau pattern dibuat menggunakan konsep pattern matching sederhana yang didaftarkan dalam program Python, dan respons dibuat menggunakan data yang ada di *database* MySql [2].

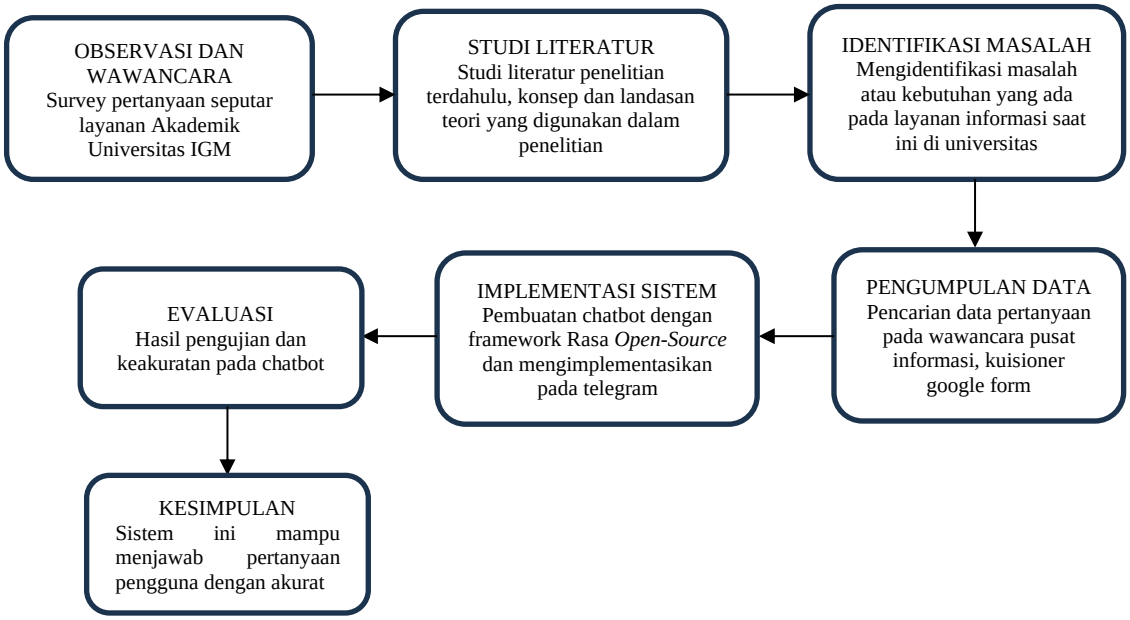
Telegram sebagai aplikasi pesan instan diklaim bisa mengatasi kelemahan yang terdapat pada Whatsapp. Telegram adalah aplikasi berbasis *cloud* dan alat enkripsi. Telegram menyediakan enkripsi *end-to-end*, penghancuran pesan secara otomatis, dan infrastruktur multi-pusatdata. Aplikasi telegram banyak disukai karena *multi platform* dan ringan. Telegram juga menyediakan *Application Programming Interface* (API) dengan dokumentasi yang lengkap untuk pengembangan sehingga pembuatan chatbot di Telegram dapat di lakukan dengan baik [3].

Tujuan dari penelitian ini yaitu mengembangkan implementasi aplikasi chatbot menggunakan platform telegram untuk menyediakan layanan informasi akademik bagi mahasiswa dan calon mahasiswa, selain itu juga, mendapatkan pengetahuan tentang kemampuan chatbot menggunakan *framework* Rasa saat berinteraksi dengan pengguna[4], mengenali bahasa alami, dan memberikan respon yang informatif dan relevan.

Hasil penelitian ini diharapkan dapat bermanfaat bagi calon mahasiswa dan masyarakat agar mendapatkan kemudahan dan cepat dalam mengakses informasi akademik sehingga meningkatkan efisiensi dan produktivitas aktivitas akademik.

2. METODE PENELITIAN

Metode penelitian untuk implementasi chatbot telegram informasi akademik universitas indo global mandiri menggunakan Rasa open-source antara lain sebagai berikut:



Gambar 1. Tahapan Penelitian

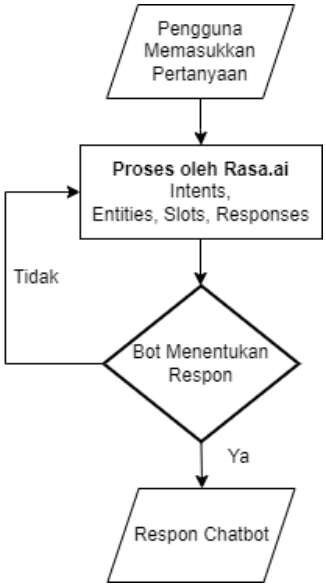
2.1. Studi Literatur

Studi literatur dilakukan dengan mencari dan memahami artikel dan jurnal ilmiah yang berhubungan dengan *chatbot* telegram dengan *framework* Rasa, *Natural Language Undersanding* (NLU).

2.2. Desain Sistem *Chatbot*

Desain sistem *chatbot* yang akan dibahas. *Chatbot* ini dirancang dan dibangun secara bertahap, tahapan pertama yaitu input yang dimasukan oleh *user* atau pertanyaan yang akan ditanyakan oleh *user*. Kemudian setelah *user* memberikan pertanyaan seputar informasi akademik, kalimat pertanyaan tersebut akan diolah oleh Rasa.ai yang telah peneliti pilih dalam membuat sistem *chatbot* ini. Rasa.ai terdiri dari beberapa komponen yaitu:

a. Intents merupakan apa yang diisyaratkan pengguna, jika pengguna mengatakan salah satu hal untuk memulai obrolan seperti hai.
b. Entites adalah potongan data yang dapat di ekstraksi dari pesan pengguna seperti *WhitespaceTokenizer* (tokenizer ini memecah teks masukan menjadi token (kata atau frasa)), dan *RegexFeaturizer* atau *CountVectorsFeaturizer* **(**memproses teks yang sudah dibagi menjadi token). Melanjutkan dari intents, setelah pengguna menyapa bot. Bot meminta informasi kontak mereka sebagai identitas untuk percakapan.
c. Slots adalah memori bot. Informasi apapun yang perlu dipertahankan selama percakpan seperti identitas pengguna akan disimpan sebagai slot.
d. Responses adalah apa yang akan dikatakan bot kepada pengguna sesuai dengan apa yang penguna akan tanyakan.



Sumber : [5]

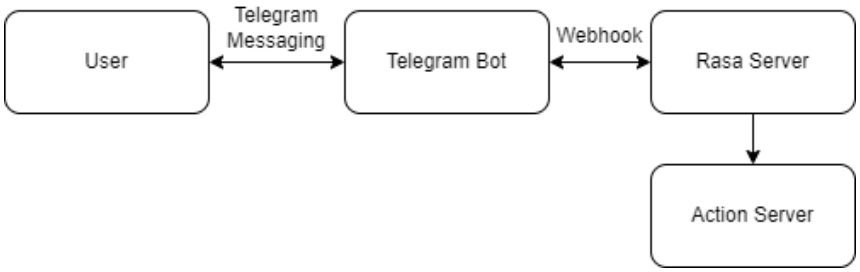
Gambar 2. Flowchart Sistem Rasa Open Source

2.3. *Arsitektur Sistem*

Adapun Komponen Utama dalam arsitektur system antara lain:

1. User (Pengguna): Menggunakan aplikasi Telegram untuk berinteraksi dengan *chatbot*.
2. Telegram Bot: Bot di Telegram yang bertindak sebagai perantara antara pengguna dan server Rasa.

3. Rasa Server: Menjalankan model NLU dan dialog yang dikembangkan menggunakan *framework* Rasa, Rasa secara default mengelola sebagian besar preprocessing ini di tahap Pipeline Rasa NLU[6].
4. Action Server: Server yang menangani tindakan khusus yang mungkin memerlukan logika tambahan atau integrasi dengan layanan eksternal.



Gambar 3. Diagram Arsitektur Sistem

Adapun alur kerja (proses) system yaitu:

a) Pengguna memulai interaksi dengan *chatbot* melalui Telegram.
b) *Frontend* mengirim pesan ke backend untuk diproses.
c) *Backend* memahami permintaan menggunakan model bahasa alami.
d) *Backend* menghasilkan respons yang dipahami oleh *frontend* dan dikirim kembali ke pengguna.

2.4. Pengumpulan Data

Dalam pengumpulan data sebelum dilakukan *preprocessing* yaitu melakukan pengumpulan data untuk mendukung kebutuhan data yang akan digunakan selama proses penelitian. Terlihat Gambar 1 menunjukkan kebutuhan untuk mengembangkan *chatbot* sehingga *friendly* dalam menjawab setiap pertanyaan-pertanyaan yang sering diajukan oleh pengguna. Metode pengumpulan data yang dilakukan pada penelitian ini yaitu kuesioner dan wawancara.

Metode kuesioner, dengan menyebarkan angket/ lembar kuesioner kepada mahasiswa untuk mengumpulkan data jawaban pertanyaan seputar layanan akademik universitas dalam jumlah yang sudah ditetapkan. Kuesioner dapat disebarkan melalui bantuan media *online* (google form).

Metode yang kedua wawancara, dengan melakukan wawancara langsung dengan staf *front office* universitas untuk mendapatkan jawaban yang valid dari pertanyaan kuesioner yang telah dikumpulkan sebelumnya.

2.5. Instalasi Sistem

a. Instalasi *Python*

Langkah pertama sebelum menginstal Rasa *Open Source* yaitu mengecek apakah sudah memiliki *Python* dan Pip, dengan cara menggunakan command berikut di terminal

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
```

dan untuk mengecek version Pip menggunakan command

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$ pip –version
pip 24.0
from    :\Users\fatur\AppData\Local\Programs\Python\Python310\lib\site-
packages\pip (python 3.10)
```

Jika *Python* dan Pip sudah terinstal maka printah ini akan menampilkan nomor versi sesuai dengan yang di instal, dan dapat lanjut ketahap berikutnya. Adapun Saat ini, *framework* Rasa mendukung versi *Python* berikut 3.7, 3.8, 3.9 dan 3.10.

b. *Virtual Environment Setup*

Buat lingkungan virtual baru dengan memilih penerjemah *Python* dan membuat direktori .\\venv untuk menampungnya

Aktifkan lingkungan virtual:

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$.\venv\Scripts\activate
$ python -m venv ./venv
```

Setelah berada di lingkungan virtual maka lanjut ketahap berikutnya. Virtual environment membantu untuk mengelola dependensi proyek *Python* secara terisolasi, sehingga paket yang Pengguna instal tidak berkonflik dengan paket lain yang mungkin sudah terinstal di sistem.

c. Instal Rasa *Open Source*

Gambar 4. Logo Framework Rasa

Pada langkah ini penginstalan dapat dilakukan dengan menggunakan command berikut ini

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$ pip install rasa
```

Perintah ini menggunakan pip, *package installer* untuk *Python*, untuk menginstal paket Rasa dari *Python Package Index* (PyPI). Ketika rasa telah berhasil diinstal maka selanjutnya dapat membuat proyek baru dengan menggunakan perintah command ini

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$ rasa init
```

Perintah **Rasa init** digunakan untuk memulai proyek baru dengan *platform* Rasa. Ini adalah langkah pertama untuk membangun chatbot menggunakan *framework* Rasa. Setelah menjalankan perintah di atas, pengguna akan diminta untuk menjawab beberapa pertanyaan interaktif untuk mengkonfigurasi proyek Rasa yang telah dibuat dan disimpan. Biasanya, pengguna akan ditanya apakah ingin mengunduh *template* proyek dasar. Pilih "y" untuk melanjutkan. Perintah ini akan membuat struktur direktori dan file dasar yang diperlukan untuk memulai proyek chatbot dengan Rasa.

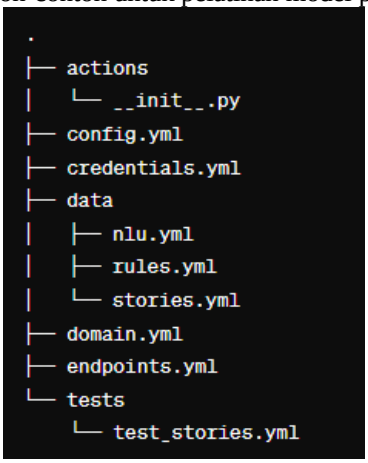
```
To get started quickly, an initial project will be created.
If you need some help, check out the documentation at https://rasa.com/docs/rasa.
Now let's start! 🤓 🤓

? Please enter a path where the project will be created [default: current directory]
2024-05-02 14:50:58 INFO     root   - creating actions
2024-05-02 14:50:58 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
sa\cli\initial_project\actions\actions.py -> .\actions
2024-05-02 14:50:58 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
sa\cli\initial_project\actions\__init__.py -> .\actions
2024-05-02 14:50:58 INFO     root   - creating actions\__pycache__
2024-05-02 14:50:58 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
sa\cli\initial_project\actions\__pycache__\actions.cpython-310.pyc -> .\actions\__pycache__
2024-05-02 14:50:58 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
sa\cli\initial_project\actions\__pycache__\__init__.cpython-310.pyc -> .\actions\__pycache__
2024-05-02 14:50:58 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
sa\cli\initial_project\config.yml -> .
2024-05-02 14:50:59 INFO     root   - copying C:\Users\fatur\AppData\Local\Programs\Python\Python310\
```

Gambar 5. Proses Instalasi Project Rasa

Pada saat proses penginstalan Rasa *Open Source* berjalan maka akan seperti pada Gambar 5 diatas. Setelah proses inisialisasi selesai, pengguna akan mendapatkan struktur direktori seperti berikut:

Penjelasan singkat dari setiap file dan direktori pada Gambar 6 yaitu:

a) actions/: Direktori ini digunakan untuk menyimpan aksi kustom yang ditulis dengan *Python*. File __init__.py adalah placeholder untuk menunjukkan bahwa ini adalah package *Python*.
b) config.yml: File konfigurasi untuk pipeline pemrosesan bahasa alami dan konfigurasi policy untuk pengambilan keputusan.
c) credentials.yml: File ini menyimpan kredensial untuk berbagai saluran input seperti Slack, Facebook Messenger, dll.
d) data/: Direktori ini berisi data pembelajaran untuk model Rasa.
    a. nlu.yml: Berisi contoh-contoh untuk pelatihan model pemahaman bahasa alami (NLU).



Gambar 6. Struktur Direktori Project Rasa

    b. rules.yml: Berisi aturan-aturan sederhana untuk pengambilan keputusan.
    c. stories.yml: Berisi skenario percakapan yang digunakan untuk melatih model dialog.
e) domain.yml: File ini mendefinisikan domain chatbot, termasuk intents, entities, slots, responses, dan actions.
f) endpoints.yml: File konfigurasi untuk menghubungkan server Rasa dengan server lain, seperti server aksi atau *tracker store*.

3. Hasil dan Pembahasan

3.1 Rasa Core

Salah satu komponen utama dari Rasa yang bertanggung jawab untuk mengelola dialog dalam sebuah chatbot atau asisten virtual. Berfungsi menentukan bagaimana chatbot merespons pengguna berdasarkan konteks percakapan dan alur dialog yang telah ditentukan. Beberapa hasil dari fitur *Rasa Core* dengan implementasi yang telah dijalankan.

Gambar 7. Kalimat Pertanyaan Intent

a. **Hasil Fitur Nlu.yml**

Pada Gambar 7 terlihat data **nlu.yml** yang berisi teks yang mewakili berbagai pertanyaan yang relevan dengan chatbot. Agar dapat mendefinisikan data chatbot agar mendukung beberapa jawaban dari pertanyaan user maka perlu menyusun beberapa intent yang sesuai dengan pertanyaan yang akan menjadi fokusan utama chatbot ini,

Tahapan ini dilakukan awal pada saat akan merancang pertanyaan yang akan dimasukan pada chatbot [1]. Ada sekitar 30 intent pertanyaan yang penulis buat pada chatbot ini, dan beberapa intent yang dibuat untuk merespon

pertanyaan diluar konteks. Terlihat beberapa contoh pada Gambar 8, kalimat-kalimat pesan yang digunakan untuk melatih model NLU.

Gambar 8. Kalimat Respon dari Intent

Gambar 9. Alur Percakapan Chatbot

b. **Hasil Domain.yml**

Pada bagian domain.yml ini dibuat ketika intent dan *action* telah tersusun. Beberapa dengan di file nlu.md, file domain.yml digunakan untuk mendefinisikan setiap intent dan disetiap intent hanya bisa memberikan satu respon atau jawaban Gambar 8 yang nantinya akan didistribusikan ke *user* sesuai pertanyaan dari *user* sebelumnya.

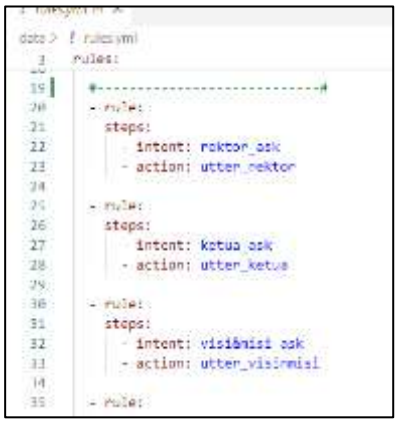
c. **Hasil Stories.yml**

Menentukan percakapan yang akan dilakukan bot dengan menggunakan intent yang telah di definisikan di file nlu.yml. Pada file stories ini percakapan dimulai ketika pertama kali *chatbot* dijalankan maka akan dimulai dengan stories pada baris pertama sebagai *default* percakapan.

Contoh story pada Gambar 9 **(-story: happy path)** diisi sebagai nama story, untuk nama story yang diisi bebas sesuai kebutuhan atau tanda bahwa story tersebut mengarah pada percakapan tertentu. Pada bagian **steps** berisi **intent** dan **action**, intent mengarah pada pertanyaan yang akan diinputkan oleh user dan action adalah respon dari pertanyaan intent user tersebut.

d. **Hasil Rules.yml**

Dalam file rules.yml ini berisi aturan yang menentukan perilaku *Chatbot* berdasarkan dengan kondisi tertentu. Aturan ini berguna untuk menetapkan tindakan tertentu ketika suatu kondisi yang ditentukan terpenuhi (Gambar 10).



Gambar 10. Aturan-aturan respon pada chatbot

3.2. *Training* Rasa NLP

Training framework Rasa NLP pada *chatbot* wajib dilakukan bertujuan untuk memberikan pembelajaran dan pengetahuan sehingga bot bisa memberikan respon sesuai dengan apa yang ditanyakan [7]. *Training* model ini dilakukan setelah semua data yang akan dilakukan traning telah siap, seperti pertanyaan dan jawaban yang berhubungan dengan Layanan Informasi Akademik Universitas Indo Global Mandiri. Untuk melakukan *training* pada Rasa *Open Source* dengan menggunakan command dibawah pada terminal visual code.

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$ rasa train
```

3.3. *Testing* Model

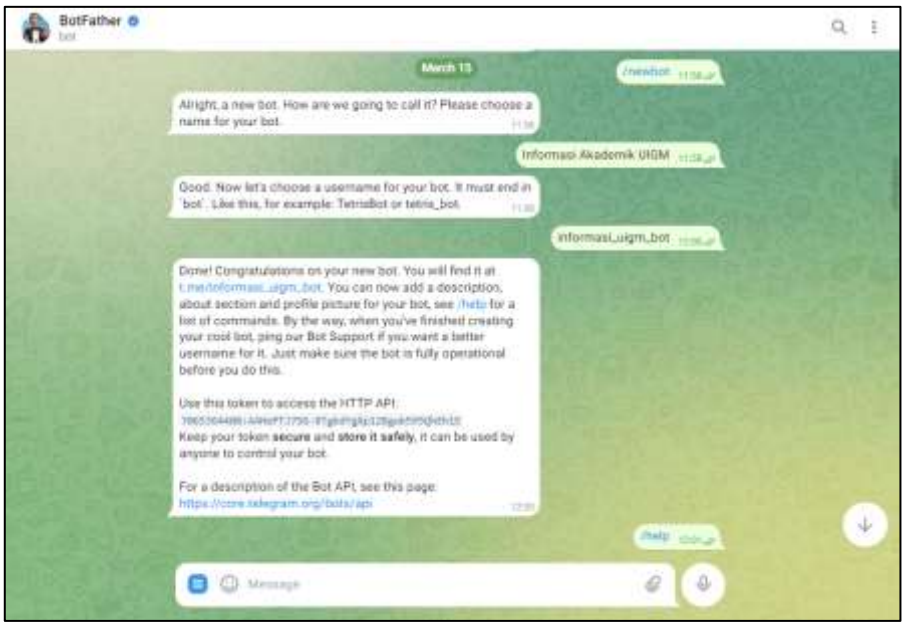
Langkah selanjutnya setelah melakukan *training* data adalah melakukan *testing* model. *Testing* model ini dilakukan pada command prompt dengan perintah (Gambar 11):

```
fatur@DESKTOP-5LIKBTF MINGW64 /c/rasaFolder (master)
$ rasa shell
```

Perintah rasa shell ini digunakan dalam pengembangan aplikasi *chatbot* berbasis Rasa untuk menguji model NLU (*Natural Language Understanding*) dan jalur percakapan secara interaktif pada terminal atau *command line interface*. Berikut ini hasil testing model yang dilakukan pada *command prompt*.

```
2024-06-23 20:09:51 INFO     root  - Connecting to channel 'cmdline' which
s will be ignored. To connect to all given channels, omit the '--connector
2024-06-23 20:09:51 INFO     root  - Starting Rasa server on http://0.0.0.
2024-06-23 20:10:10 INFO     rasa.core.processor  - Loading model models\2
2024-06-23 20:12:10 WARNING  rasa.shared.utils.common  - The UnexpecTED Ir
moved in the future 🔔 Please share your feedback on it in the forum (http
uction.
2024-06-23 20:12:56 INFO     root  - Rasa server is up and running.
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  halo
Hai! Apa kabar kamu?
Your input ->  baik
Senang mendengar mu baik-baik saja :)
Your input ->  siapa rektor uigm?
Dr. H. Marzuki Alie
```

Gambar 11. Testing Model Rasa



Gambar 12. BotFather pada Telegram

3.4. Implementasi Chatbot Telegram

Berikut ini langkah-langkah yang harus dilakukan untuk menghubungkan *chatbot* Rasa *Open Source* ke *platform* telegram

a. Membuat Akun Bot Telegram

Langkah pertama yang harus dilakukan adalah dengan membuat akun bot baru melalui telegram dengan ketik BotFather pada kolom pencarian, kemudian klik start.

Setelah berhasil membuat akun *chatbot*, pengguna akan disajikan dengan tampilan yang menampilkan informasi seperti yang terlihat pada Gambar 12 Tampilan ini berisi detail API yang diperlukan untuk menghubungkan platform Telegram ke platform Rasa *Open Source*.

b. Setup API Credentianls.yml

Project Rasa dan file credentials.yml Rasa akan mengkonfigurasi berbagai saluran pesan. Untuk Telegram, tidak menyediakan slot default apa pun. Oleh karena itu, harus menambahkan slot seperti pada Gambar 13.

credentials.yml

```
# This file contains the credentials for the voice & chat platforms
# which your bot is using.
# https://rasa.com/docs/rasa/messaging-and-voice-channels

rest:
#  # you don't need to provide anything here - this channel doesn't
#  # require any credentials

telegram:
  access_token: "7065364408:AAGhXtfZWAq5lcuvvjwA7PRUtk8XBEGIwwk"
  verify: "informasi_uigm_bot"
  webhook_url: "https://0b32-103-109-200-40.ngrok-free.app/webhooks/telegram/webhook"

#facebook:
#  verify: "<verify>"
#  secret: "<your secret>"
#  page-access-token: "<your page access token>"
```

Gambar 13. Proses Penghubung API Telegram ke Rasa

c. *Start* Ngrok Server

Aplikasi ngrok di dalam direktori proyek Rasa digunakan untuk menjalankan aplikasi lokal (yang berjalan di *localhost*) ke publik melalui URL yang unik. Aplikasi ngrok [8] memproses langsung dari folder proyek tanpa harus mencari atau memindahkan aplikasi ke direktori yang berbeda. Proses konfigurasi dan penggunaan ngrok menjadi lebih mudah dan terintegrasi dengan proyek Rasa yang sedang dijalankan. Koneksi antara *localhost* dan *World Wide Web* dengan cara menjalankan perintah berikut.

```
ngrok http 5005
```

d. *Enable localhost*

Mengaktifkan action_endpoint, memastikan bahwa bot dapat merespons dengan benar terhadap permintaan dan input dari pengguna.

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

e. *Testing Chatbot* Telegram

Buka project Rasa di Visual Studio Code dan buka terminal. Setelah itu, ketikkan perintah berikut.

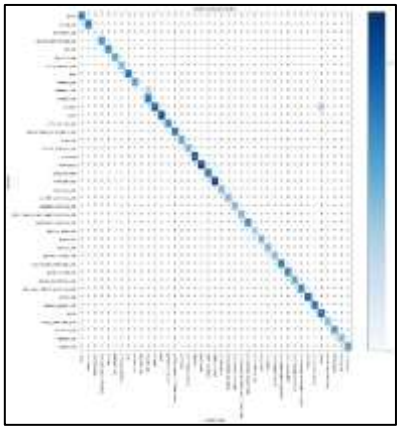
```
fatur@DESKTOP-5LIKBTF MINGW64
C:/Users/fatur/AppData/Local/Programs/Microsoft VS Code
$ rasa run
```

Setelah server Rasa aktif, buka bot telegram dan mengetikkan perintah "/start". Dengan langkah tersebut, pengguna dapat memulai interaksi dengan bot, mengaktifkan fungsi-fungsi pertanyaan yang telah disiapkan dalam bot untuk dijalankan. Proses ini memungkinkan untuk mulai menggunakan bot dengan mengirimkan perintah awal yang telah ditentukan, membuka percakapan untuk berbagai interaksi lebih lanjut dengan bot tersebut.

Gambar 14. Tampilan Hasil Percakapan

3.5. Evaluasi Model NLU

Model NLU yang diuji, dari hasil evaluasi prediksi intent ditunjukkan pada Gambar 14. Hasil ini menunjukkan kemampuan chatbot dalam memahami teks masukkan [9]. Nilai akurasi yang didapatkan adalah 0,977. Nilai rata-rata mikro untuk precision 0,977, recall 0,977 dan F1-Score-nya 0,977. Sementara jika mempertimbangkan beban sampel, kita melihat hasil rata-rata tertimbang yaitu: 0,980 untuk precision, 0,977 untuk recall dan 0,976 untuk F1-Score.



Gambar 15. Intent Confusion Matrix

Perhitungan hasil yang dilakukan menggunakan Rasa *Open Source* dan perhitungan manual menunjukkan hasil yang konsisten dan sama, yang mengindikasikan bahwa kedua metode tersebut dapat diandalkan dalam menilai kinerja model klasifikasi intent.

4. KESIMPULAN

Berdasarkan hasil yang diperoleh bahwa chatbot yang dikembangkan menggunakan Rasa Open Source memiliki performa yang sangat baik dalam mengklasifikasikan *intent* dan memberikan respons yang baik terhadap pertanyaan-pertanyaan dari batasan masalah yang ada. Evaluasi dan pengujian fungsional menunjukkan bahwa chatbot ini dapat digunakan dalam aplikasi nyata. Hasil dari *accuracy, precision, recall*, dan F1-*Score* memiliki nilai rata-rata 0.9 atau sama dengan 90%. Nilai tersebut didapatkan dalam results hasil *confusion matrix* dan perhitungan manual yang memiliki hasil yang sama. Maka dapat disimpulkan bahwa implementasi chatbot pada telegram menggunakan framework Rasa *Open Source* dapat mempermudah pengguna dalam mengembangkan chatbot layanan informasi dan memiliki hasil yang akurat pada chatbot tersebut. Pengembangan selanjutnya dapat dilakukan dengan penggunaan model transformer seperti BERT atau GPT, merancang chatbot berbasis skrip dan templat, dan juga dapat membandingkan hasil *chatbot* pada platform seperti *whatsapp, facebook,* dan *website*.

DAFTAR RUJUKAN

[1] 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 Komputer (JTIIK)*, vol. 10, no. 2, pp. 291–300, 2019.

[2] M. N. Fauzy and K. Kusrini, "Chatbot menggunakan metode fuzzy string matching sebagai virtual assistant pada pusat layanan informasi akademik," *Jurnal Informa: Jurnal Penelitian dan Pengabdian Masyarakat*, vol. 5, no. 1, pp. 61–67, 2019.

[3] M. Furqan, S. Sriani, and M. N. Shidqi, "Chatbot Telegram Menggunakan Natural Language Processing," *Walisongo Journal of Information Technology*, vol. 5, no. 1, pp. 15–26, 2023.

[4] D. Wulandari and J. W. Sasongko, "Implementasi Chatbot Menggunakan Framework Rasa untuk Layanan Informasi Wisata di Kota Pati," *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 6, no. 2, 2023.

[5] D. S. Ramadhan and F. Amin, "Impelementasi Chatbot Menggunakan Framework Rasa Untuk Melayani Informasi Produk Umkm Di Kab. Kendal," *INTECOMS: Journal of Information Technology and Computer Science*, vol. 6, no. 2, pp. 1057–1061, 2023.

[6] Y. Yunefri, Y. F. Ersan, and Sutejo, “Chatbot pada Smart Cooperative Oriented Problem Menggunakan Natural Language Processing dan Naive Bayes Classifier,” *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 4, no. 2, pp. 131–141, 2021.

[7] N. Cholis Anggoro and M. Akbar, “Chatbot Pemilihan Makanan dan Minuman Berdasarkan Kalori menggunakan Natural Language Processing,” *Jurnal ForAI |*, vol. 1, no. 1, 2023.

[8] R. Parlika, D. M. W. Cakra, T. N. Ardhian, and S. Rahmawati, “Sistem Integrasi BOT Register Terhadap Website Pengolah Data Menggunakan Akses NGROK,” *Jurnal Ilmiah Sinus (JIS)*, vol. 19, no. 2, pp. 1–16, Jul. 2021, doi: 10.30646/sinus.v19i2.531.

[9] F. F. Yusron, A. Komarudin, and Melina, “Chatbot Informasi Penerimaan Mahasiswa Baru Menggunakan Metode FastText dan LSTM,” *JOURNAL OF APPLIED COMPUTER SCIENCE AND TECHNOLOGY (JACOST)*, vol. 5, no. 1, pp. 33–39, 2024, doi: 10.52158/jacost.648.