RASA CHATBOT USING NLU

Submitted in partial fulfillment of the requirements for the

award of

Bachelor of Engineering degree in Computer Science and Engineering

By

SURIYA M S (Reg.No - 39110992)
SELVA A (Reg.No - 39110920)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this Project Report is the bonafide work of **SURIYA M S (Reg.No - 39110992) and Selva A (Reg.No - 39110920)** who carried out the Project Phase-2 entitled **"RASA CHATBOT USING NLU"** under my supervision from January 2023 to April 2023.

[illegible]

Internal Guide
Dr. A. MARY POSONIA., Ph.D.

[illegible]

Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 20.04.2023

[illegible]

[illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **Suriya M S (Reg.No- 39110992),** hereby declare that the Project Phase-2 Report entitled **“RASA CHATBOT USING NLU”** done by me under the guidance of **Dr. A. Mary Posonia, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

[illegible]

DATE: 24.04.2023

PLACE: Chennai **SIGNATURE OF THECANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Mary Posonia, M.E., Ph.D,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks toall Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

We introduce a pair of tools, Rasa NLU and Rasa Core, which are open-source python libraries for building conversational software. Their purpose is to make AI based dialogue management and language understanding accessible to non-specialist software developers. In terms of design philosophy, we aim for ease of use, and bootstrapping from minimal (or no) initial training data. Both packages are extensively documented and ship with a comprehensive suite of tests. We introduce Rasa NLU and Core as easy to use tools for building conversational systems, since until now there was no widely-used statistical dialogue system intended for non-specialists. Rasa is already used by thousands of developers worldwide. As with many other conversational systems, our tools are split into natural language understanding (Rasa NLU) and dialogue management (Rasa Core). Section 3 describes the code architecture, in 4 we outline the developer experience, and in 5 demonstrate an example application. The COVID-19 pandemic, a flood of information and news from various sources has raised numerous questions in people's thoughts concerning the disease. COVID-19 is a coronavirus, which encompasses the Severe Acute Respiratory Syndrome (SARS)virus and Middle East Respiratory Symptoms (MERS) virus. Coronavirus is a virus family that comprises virus types that cause the flu and the common cold. Chatbots and voice assistants are becoming increasingly popular ways for people to connect with systems.

TABLE OF CONTENTS

.

LIST OF FIGURES

LIST OF ABBREVIATIONS

RASA	**-**	Response Action
NLU	**-**	Natural Language Understanding
NLP	**-**	Natural Language Processing
AI	-	Artificial Intelligence
NLG	-	Natural Language Generation
VS CODE	-	Visual Studio Code

CHAPTER 1

INTRODUCTION

This describes two open-source Python libraries for conversational software development, RASA Natural Language Understanding (NLU) and RASA Core. You can use these tools to create conversational software like the Messenger platform. Their goal is to make dialogue management and language interpretation based on AI (Artificial Intelligence) available to non-specialist developers. The aim of this project is Artificial Intelligence and Machine Learning developments have contributed to the development of these new types of user interaction. We'll focus on the development of chatbots as a channel for information distribution via an Android application in particular. With over 2.5 billion active users globally and over 3 million devices utilizing this Android application, it is certain to reach a large number of people. The aim of this project is Artificial Intelligence and Machine Learning developments have contributed to the development of these new types of user interaction. We'll focus on the development of chatbots as a channel for information distribution via an Android application in particular. With over 2.5 billion active users globally and over 3 million devices utilizing this Android application, it is certain to reach a large number of people.

A chatbot is proposed in this study to eliminate information pollution, including false information regarding COVID19, and to provide the right responses to people's concerns. In the realm of natural language processing, a chatbot is a popular discussion system (Natural Language Processing). Chatbots are software that allows computers and people to converse. The chatbot is created on the Android platform and uses the RASA open-source framework to answer inquiries regarding covid. Conversational systems are becoming pervasive as a basis for human computer interaction as we seek more natural ways to integrate automation into everyday life. Well-known examples of conversational AI include Apple’s Siri, Amazon’s

A chatbot is a software which is designed to understand what human wants and guides them to their desired output. These chatbots work using Artificial Intelligence (AI) and Natural Language Processing (NLP). The aim of the chatbot is to reduce the work load of humans. Chatbots are used to automate customer service and reduce tedious tasks performed by employees so they can spend their time more productively on higher priority tasks. The 10 chatbots are used in many areas such as health, e-commerce, customer services, education etc. With the increase in the chatbots, many tasks have become easy.

Say previously it was difficult to handle large number of customer calls. Back then people need to wait for more time for their turns when they call for toll- free numbers. Answering the calls would become tedious for service providers also. Thus, if a chatbot is pre-fed with set of questions and answers there won't be any delay in attending customers and the business would acquire a positivegrowth. The evolution of chatbots is based on the complexity of the functionalities, the levels of chatbot can be described. Generally, chatbots can be grouped intofive levels based on the applications and its functionalities. It shows the level of chatbot with its working. The Level 0 chatbots are those which sends some notifications or messages to the customers. They are bots or programs but not persons.

For sending similar data to huge customers these bots are used. The Level 1 chatbots are used for answering FAQ's. Generally in many customer carecenters it will become difficult for persons to answer similar questions again and again. Level 1 bots will be pre-fed with set of questions and answers. These bots will answer on behalf of persons to reduce time and cost. The Level 2 bots are conversational bots which are used in ecommerce and social media platforms. These bots will pop-up when user visits a website. They act as virtual assistants and have information about the context and answer accordingly. The Level 3 bots are advanced and they are highly personalized. They will track even daily activitiesand from user's previous likes and dislikes to give final output. The Level 4chatbots can be thought as future bots where many bots interact with each other and learn from each other to give better results. With the inclusion of chatbot in education, many universities, students and faculties are obtaining advantages. Some of the ML and AI enabled chatbots will also help users interact with studentsto understand the mindset of users and give personalized learning systems.

A number of chatbots are already present in the market. Some 11 examples include Google’s voice assistant, amazon’s Alexa, Siri etc. These assistants orbots will interact with the users and try to understand what user is in need of and gives output accordingly. Much growth of chatbots can be seen in the health, customer services, e-commerce websites. But not much work is done in using chatbot efficiently in the educational field. In this article, we are discussing briefly the chatbot development using the RASA framework. The pandemics have overloaded medical systems around the world, with over 300 million verified Covid-19 cases and over 5 million deaths. A sense of need and urgency is developed for a virtual system that can relieve medical workers and the system of their workload.As a result, a chatbot is created using the Rasa open-source framework.

The aim of this project is Artificial Intelligence and Machine Learning developments have contributed to the development of these new types of user interaction. We'll focus on the development of chatbots as a channel for information distribution via an Android application in particular. With over 2.5 billionactive users globally and over 3 million devices utilizing this Android application, it is certain to reach a large number of people.

A chatbot is proposed in this study to eliminate information pollution, including false information regarding COVID19, and to provide the right responses to people's concerns. In the realm of natural language processing, a chatbot is a popular discussion system (Natural Language Processing).

Chatbots are software that allows computers and people to converse. The chatbot is created on the Android platform and uses the RASA open-source framework to answer inquiries regarding covid.

CHAPTER 2

LITERATURE SURVEY

Battineni, Gopi, Nalini Chintalapudi, and Francesco Amenta [1] Healthcare, "Designing AI Chatbots in Epidemics Like COVID-19." Digital Publishing Interdisciplinary Institute, 2020. The purpose of this document is to offer medical guidance in off-the-grid locations. When using (AIML). This chatbot looks for any and all instances of infection. The foundation of the chatbot engine is idea of backend cognition, which verifies the input through Web API techniques. Your doctor is always accessible via live chat. The drawback is that if the user doesn't confirm a particular answer, the bot won't respond correctly.

Bharti, Urmil, Deepali Bajaj, Hunar Batra, Shreya Lalit, Shweta Lalit, and Aayushi Gangwani. "Medbot: [2] After Covid19, a voice-based AI chatbot will offer telemedicine. 2020 5th International Conference on Electronic & Communication Systems. The authors of this study stressed the value of preserving social distance and noted that it would be extremely risky for medical personnel to physically visit patients. developed telemedicine. NLP, Firebase Cloud Functions, Google Cloud Platform, Dialogflow Conversation API, rule based grammar, and Machine Learning matching are the techniques and technologies employed in this case. The hybrid approach tries matching using two grammars with rules. It switches to an ML match if it doesn't match. With sufficient training terms, it functions accurately.

Bocklisch, T et al [3] introduced the rasa NLU and core for the first time with open-source license. The aim of this study was to provide a dialogue system based on machine learning and understanding the language to the enthusiast who are no such expert in technology. The package they developed was of minimal size and advancement is done in the package. With the efforts of 344 contributers, 244releases of rasa have been released with total of 18023 commits.

Eleni Adamopoulou & Lefteris Moussiades [4] 2020 Designing a chatbot's objectives, procedures, and user requirements is the first step in creating one. The chatbot functionalities are evaluated locally after being built using a programming language or a chatbot application framework. The chatbot is subsequently made accessible to the public 8 on a website or in a data center, and it is linked to one or more channels to send and receive messages.

Goyal et al. [5] This paper examines, the available technologies and implemented the use case prototype to give up insight that chatbot can be used with social networks so that smart chat system can be employed. In the study, a social chatbot for the football has been designed which answer the questions to the Spanish footballleague. Chatbot is deployed with slack client, having text-based interaction. It extracts the information about football players, team and their trainers.

Jiao [6] in this study, Jiao designed a functional framework which implements the principle of RASA NLU and furthermore he integrated the RASA NLU with the neural network methods resulting into an entity extraction system and later on recognizes the intents and related entities. This study showed that Neural network outperforms in with RASA NLU.

Kondapi, R. & Katta, R.K. & Potluri, Sirisha & Bathula, A. & Basha, S.K. [7] (2019). a human-interaction Android chatbot application. The 2019 issue of the International Journal of Recent Technology and Engineering Pacifiurr offers adviceon how to effectively handle and get beyond challenges in order to help in trying circumstances. The user is shown the results of categorising the text in reference to previously accessible data or data that already exists in the database.

Lacerda [8] used the core of rasa and presented a new software stack called as Rasa-ptbr boilerplate for the non-specialist who doesn't much about the internals of the chatbot , considering the chatbot as blackbox. Now, chatbots are intelligent systems which can perform complex task and has many application in robotics and natural language processing.In the recent time, there is development in the field of artificial intelligence, so the chatbots are acting as customer service agents.

Lee, Hyeonhoon, Jaehyun Kang, and Jonghyeon Yeo. [9] " Journal of medical Internet research 23, no. 5: "Expert suggestions using smart phone artificial intelligence chatbot: Development and Deployment" (2021). In their investigation, writers Hyeonhoon et al. In order to establish an artificial intelligence (AI) chatbot, we selected 118,008 sets of symptom claims, created a pipeline of 51,134 words, and used the "alpha" architecture. The server for this is Google Cloud. Computers and computers can both use AI chatbots. mobile phone.

Lei, Hannah, Weiqi Lu, Alan Ji, Emmett Bertram, Paul Gao, Xiaoqian, Jiang, Arko Barman. [10] "Prototype COVID-19 Smart Chatbot for Patient Monitoring (2021) In this article, He suggested a follow-up strategy for sufferer to use to keep track of their health. He used his Med711 and CORD1918 to develop word cloud text preprocessing and extraction for exploratory analysis. The clinical NER used in this case, Med7, does not offer enough information or skills to address fundamental queries. For improved performance, extracted entities and models should be combined.

Martin, Alistair, Jama Nateqi, Stefanie Gruarin, Nicolas Munsch, Isselmou Abdarahmane, Marc Zobel, and Bernhard Knapp.[11] "Digital Risk Screening of Citizens with Chatbots: Artificial Intelligence-Based First Line Defense Against COVID-19," Scientific Reports 10, No (2020). Symtoma was the application employed in this paper's study. created to assist those living in rural regions during COVID-19. This digital health assistant enhances symptom accuracy in selecting COVID19 for research by using a panel of several medical history along with COVID- 19 medical reports. 96.32% were discovered. It accepts 36 distinct languages of free text input. They employed four distinct tactics. Between case presentation and symptom frequency is the SF-DIST (symptom spatial distance), also known as the SFSD (distance normalised by standard deviation). Normal deviation by first principal component, or SF-PCA, and SF-COS (cosine similarity).

Nimavat & Champaneria [12] 2017, The services a chatbot should provide to users and the category it falls under dictate the algorithms or platforms utilized to build them. used the core of rasa and presented a new software stack called as Rasa-ptbr-boilerplate for the non-specialist who doesn't much about the internals of the chatbot, considering the chatbot as blackbox. Now, chatbots are intelligent systems which can perform complex task and has many application in robotics and natural language processing.In the recent time, there is development in the field of artificial intelligence, so the chatbots are acting as customer service agents

Ouerhani, Nourchène, Ahmed Maalel, Henda Ben Ghézala, and Soulaymen Chouri. [13] Deep learning Sentiment Analysis Models for Intelligent Ubiquitous Chatbots for COVID-19 Support During and After Quarantine. (2020). In this study, the Ouerhani et al. approach was used to identify COVID-19. We have created an

AI system that detects, understands, and makes predictions. They recommended combining the use of AI and smartphones to more accurately identify his COVID19 cases in places where inhabitants are isolated. They provided an AI framework for forecasting the severity of the pneumonia and the COVID-19 result. They created an AIbased automated approach for tracking, measuring, and identifying COVID19 positive patients using chest CT images. intended for radiologists and doctors.

2.1 INFERENCES OF LITERATURE SURVEY

In this paper, the battineni in the year of 2020 have predicted the housing price direction using machine laearning techniques. He used multiple techniques of classification and have not used regression and boosting of tress. Eleni Adamopoulou in the year of 2020 have done real estate vakue prediction using linear regression. He had used linear regression and used only naive method. Kondapi in the year of 2019 had done hedonic housing and demand for clean air.He had study of internal and external factor and had only limited method of less complexity.

1. Rasa is an open-source framework for building chatbots that use natural language processing (NLP) and machine learning (ML) techniques.
2. Rasa provides two main components, i.e., Rasa NLU for natural language understanding and Rasa Core for dialogue management.
3. Rasa NLU uses machine learning models to extract the intent and entities from the user's input, while Rasa Core uses a machine learning algorithm to predict the next action of the chatbot.
4. Rasa has been used in various domains, including healthcare, e-commerce, customer service, and education, among others.
5. Rasa has been found to perform better than other chatbot frameworks in terms of accuracy, scalability, and flexibility.
6. Rasa allows developers to create custom actions, integrate external APIs, and deploy chatbots on various platforms, including web, mobile, and messaging apps.
7. Rasa also provides extensive documentation and community support to help developers build and improve their chatbots.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

NLU techniques along with machine learning algorithms creates a conversational AI. Which gives a prediction about the disease the user might have based on their symptoms, creating an interactive medical bot which can hold a conversation with the user and gives appropriate responses and accurate information.

Combining NLU methods with machine learning algorithms produces conversational AI. As a result, an interactive medical bot will be developed that can identify potential ailments from a user's symptoms and engage in conversation with them to provide them correct information and the right responses. The symptoms are transformed into a vector once all of the user's symptoms have been entered. The test set is supplied to the machine learning classification model as this vector.

Contextual understanding is one of the key challenges in building a chatbot is the ability to understand the context of the conversation. Rasa chatbots need tobe able to recognize the intent behind a user's message and respond accordingly. However, this can be difficult when the user's message is ambiguous or the conversation has shifted to a new topic.

User engagement is to be effective, chatbots need to be engaging and able to keep users interested in the conversation. This requires the chatbot to be able to understand the user's needs and preferences, and tailor its responses accordingly.

Natural Language Generation (NLG): NLG is the process of generating human-like responses to user messages. This requires the chatbot to be able to understand the user's intent and generate a response that is both accurate and natural-sounding. However, current NLG models are still far from perfect and often produce responses that are stilted or grammatically incorrect.

1. Handling complex conversations: While Rasa is designed to handle complex conversations, there are still situations where the chatbot may struggle to understand user input or provide appropriate responses. Handling complex conversations, especially in domains such as healthcare,finance, and legal, is still a challenging problem.

2. Integrating with external systems: While Rasa allows developers to integrate external APIs and services, integrating with legacy systems can still be a challenge. Many organizations still use legacy systems that are not compatible with modern APIs, making it difficult to integrate with Rasa.
3. Dealing with multi-lingual input: Rasa is designed to support multiple languages, but handling multi-lingual input is still a challenging problem. Depending on the complexity of the language, it may be difficult for Rasa to accurately understand the user's intent and provide appropriate responses.
4. Ensuring data privacy and security: Rasa, like other chatbot frameworks, requires access to user data to function properly. Ensuring data privacy and security is still an open problem, especially in domains such as healthcare and finance, where data privacy and security are critical.
5. Handling context switching: Context switching is a challenging problem in chatbots. Rasa may struggle to maintain context when users switch between different topics or ask unrelated questions. This can result in apoor user experience and frustration.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

A chatbot is a software application that allows you to have an online chat conversation with an AI-enabled virtual assistant using text or text-to-speech. Chatbot systems are often designed to accurately replicate the way a human would behave as a conversational partner, although many in production are still unable to speak adequately, and none of them can pass the traditional Turing test. The scope of Chatbots is as follows:

Simple user interface: Chatbots often have very simple user interfaces, displaying only what is required to keep a dialogue flowing. With this in mind, we'll employ a straightforward command- line interface that displays queries and responses as adjacency pairs.

Quick response: Instead of taking minutes, the system should respond in a matter of seconds. Recognize queries appropriately: The system must accurately identify questions. This is accomplished through the use of machine learning, which responds to requests based on probabilities and the previous set of questions (i.e., Frequently asked questions). Serverless architectures (Functions-as-a-Service) have gained popularity in recent years as a manner of providing backend services without the need for dedicated infrastructure. Users can deploy stateless functions into platform infrastructures using serverless architecture, and this stateless behavior makes each invocation independent of the prior.

1. Data privacy and security: Rasa chatbots require access to user data to function properly. However, this data can be sensitive, and there is a risk of data breaches or misuse. It is important to ensure that appropriate security measures are in place to protect user data.

2. Bias and discrimination: Chatbots can inadvertently perpetuate biases and discrimination based on the training data they receive. It is essential to ensure that the training data is diverse and unbiased to avoid such issues.

3. Misunderstanding user input: Rasa chatbots use machine learning algorithms to understand user input, and there is a risk of misinterpretation, especially in cases of complex conversations or multi-lingual input. This can result in incorrect responses, leading to a poor user experience.
4. Integration with external systems: Rasa chatbots may require integration with external systems and APIs to provide the desired functionality. However, integrating with legacy systems can be challenging and pose a risk of downtime or malfunction.
5. Technical issues: Rasa chatbots are software applications, and as such, there is a risk of technical issues such as bugs, crashes, and other software-related problems. It is essential to conduct thorough testing and ensure that the chatbot is reliable and performs as expected.
6. Ethical considerations: Chatbots can be used for a variety of purposes, some of which may raise ethical concerns. For example, chatbots used in the healthcare domain must comply with ethical and legal regulations, such as patient confidentiality.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

FLASK - Flask is a micro web framework in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, 13 form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

ANACONDA OPEN-SOURCE SOFTWARE - Anaconda is the open-source Graph Database that is developed using Java technology. It is highly scalable and schema-free (NoSQL). Unlike traditional databases which store data in rows, columns, and tables, it has a completely flexible structure. Therefore, it saves relationships that connect data. The architecture is for traversal of nodes and relationships, optimal management, and storage. Each node contains direct pointers to all the nodes that are connected by relationships. Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

GRAPH DATABASE - A graph database is a database used to model the data in the form of a graph. Here, the nodes of a graph depict the entities while the relationships depict the association of these nodes. Most of the data exists in the form of the relationship between different objects, and more often, the relationship between the data is more valuable than the data 14 they usually do a search through a separate data structure called "index" which is expensive and makes the database slower, whereas graph databases store relationships and connections as first-class entities. Also, can easily retrieve (traverse) connected data faster by indexing starting point and then just chasing the memory pointer.

VS CODE SOFTWARE - Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE. To enable Python in VS code, open the Command Palette (Ctrl+Shift+P) and enter Preferences: Open User Settings. Then set python. defaultInterpreterPath, which is in the Python extension section of User Settings, with the appropriate interpreter.

3.2.1 FUNTIONAL REQUIREMENTS:

1. The system must provide clear information about Admission policy.

2. The system must provide clear and fully detailed information about university colleges.

3. The system must provide clear and fully detailed information about colleges' programs.

4. The system must provide clear and fully detailed information about colleges.

6. The system should clarify information about the permitted secondary school branch for each major.
7. The chatbot should be able to manage the flow of the conversation, maintaining context and providing appropriate responses based on theuser's input. Rasa provides tools to help developers define and train the chatbot's dialogue management.
8. The chatbot should be able to generate natural-sounding responses to user inputs. Rasa provides tools to help developers define and train the chatbot's natural language generation capabilities.
9. The chatbot should be able to integrate with external systems, such as databases, APIs, and other applications, to retrieve and process data as needed. Rasa provides tools to help developers integrate the chatbot with external systems.
10. The chatbot should be able to support multiple languages, allowing users to interact with the bot in their preferred language. Rasa provides tools to help developers create multilingual chatbots.
11. The chatbot should be able to collect and analyze data on user interactions, allowing developers to understand how users are interacting with the bot and identify areas for improvement. Rasa provides tools to help developers collect and analyze chatbot data.

3.2.2 NON-FUNTIONAL REQUIREMENTS:

1. The system shall handle multiple user's inputs, if two or more students are

chatting with the bot, none of the students has to wait too long to be answered by the system.

2. The bot should have a delay in response, to let the student feel like he/she is talking to a human instead of a bot. A little late response from the chatbot makesthe student feel as though he is talking to a human.

3. The system should have the appropriate data set. The correct data set is the basis for the chatbot, when the data set is correct and tuned, the chatbot will be trained on it to give the best possible result.

4. The system should have Data Training. Data training mainly depends on the content targeted at Admission and Registration deanship.

Performance: The chatbot should be able to handle a large number of simultaneous user requests without slowing down or crashing. It should respond quickly to user input, and there should be minimal latency in the conversation.

Usability: The chatbot should be easy to use and understand, with a user-friendly interface and clear instructions. It should be accessible to users with a range of abilities, including those with disabilities.

Security: The chatbot should be secure and protect user data and privacy. It should use encryption and other security measures to prevent unauthorized access or data breaches.

Reliability: The chatbot should be reliable and available to users at all times, with minimal downtime or errors.

Scalability: The chatbot should be able to scale up or down depending on the number of users and their needs. It should be able to handle an increasing number of users without affecting performance.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Rasa is an open-source chatbot development framework that allows developers to create intelligent, conversational chatbots and virtual assistants. The framework provides a range of tools and libraries for building chatbots that can understand and respond to natural language input.

At its core, Rasa uses machine learning algorithms to understand user inputs and generate appropriate responses. The framework includes components for natural language processing (NLP), intent recognition, dialogue management, and response generation. Rasa also provides tools for training and evaluating chatbot models, as well as integrating with various messaging platforms and APIs.

One of the key benefits of using Rasa is its flexibility and customization options. Developers can build chatbots for a wide range of use cases, fromcustomer service and support to marketing and sales. Rasa also allows for the creation of multi-lingual chatbots, making it a great option for businesses with a global customer base.

Overall, Rasa provides a powerful and flexible platform for building intelligent chatbots that can understand and respond to natural language input,and it has gained popularity in the chatbot development community in recent years.

A chatbot is a software which is designed to understand what human wants and guides them to their desired output. These chatbots work using Artificial Intelligence (AI) and Natural Language Processing (NLP). The aim of the chatbot is to reduce the work load of humans. Chatbots are used to automate customer service and reduce tedious tasks performed by employees so they can spend their time more productively on higher priority tasks. The 10 chatbots are used in many areas such as health, e-commerce, customer services, education etc. With the increase in the chatbots, many tasks have become easy. Say previously it was difficult to handle large number of customer calls. Back then people need to wait for more time for their turns when they call for toll-free numbers. Answering the calls would become tedious for service providers also. Thus, if a chatbot is pre-fed with set of questions and answers there won't be any delay in attending customers and the business would acquire a positive growth. The evolution of chatbots is based on the complexity of the functionalities, the levels of chatbot can be described. Generally,

Chatbots can be grouped into five levels based on the applications and its functionalities. It shows the level of chatbot with its working. The Level 0 chatbots are those which sends some notifications or messages to the customers. They are bots or programs but not persons. For sending similar data to huge customers these bots are used. The Level 1 chatbots are used for answering FAQ's. Generally, in many customer care centers it will become difficult for persons to answer similar questions again and again. Level 1 bots will be pre-fed with set of questions and answers. These bots will answer on behalf of persons to reducetime and cost. The Level 2 bots are conversational bots which are used in ecommerce and social media platforms. These bots will pop-up when user visits a website. They act as virtual assistants and have information about the context and answer accordingly. The Level 3 bots are advanced and they are highly personalized. They will track even daily activities and from user's previous likesand dislikes to give final output. The Level 4 chatbots can be thought as futurebots where many bots interact with each other and learn from each other to give better results. With the inclusion of chatbot in education, many universities, students and faculties are obtaining advantages. Some of the ML and AI enabled chatbots will also help users interact with students to understand the mindset of users and give personalized learning systems.

A number of chatbots are already present in the market. Some 11 examples include Google's voice assistant, amazon's Alexa, Siri etc. These assistants or bots will interact with the users and try to understand what user is in need of and gives output accordingly. Much growth of chatbots can be seen in the health, customer services, e-commerce websites. But not much work is done in using chatbot efficiently in the educational field. In this article, we are discussing briefly the chatbot development using the RASA framework. The pandemics have overloaded medical systems around the world, with over 300 million verified Covid-19 cases and over 5 million deaths. A sense of need and urgency is developed for a virtual system that can relieve medical workers and the system of their workload. As a result, a chatbot is created using the Rasa open-source framework. Rasa differs from previous traditional FAQ interactions in that it is based on natural conversations, such as how humans interact with one another, by taking into account what previous contexts were sent and what actions should be taken in relation to the contexts, and gracefully handling the situation, unexpected dialogue, directing the conversation when the user veers off the regular chat path, and improving over time, it goes well beyond FAQ Interactions. Rasa is generally made up of two components: Rasa NLU and Rasa Core.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Rasa Framework serves as the backend infrastructure for our application. It illustrates the architecture of this chatbot. Our chatbot is AI-driven and built on a serverless platform. It uses the rasa framework's Natural Language Processing (NLP) and Natural Language Understanding (NLU) to interpret the user's inquiry anything related to COVID - 19 and return appropriate responses. Reading, decoding, understanding, and making sense of human languages are all aided and facilitated by NLP.

In our architecture, the initial level of processing is concerned with obtaining the input, which might be in the form of text or audio. When a user asks a question or sends a message via audio, the user query or message is converted from audio to text, which is known as Speech-to-Text. In the second level of the process, this extracted text is used as a foundation for performing Natural Language Understanding on the generated text to decode the semantic meaning of the user's input and recognize morphemes at the next level of processing. Because there is no need for audio to text conversion in chat interfaces, this is considered the first level of processing.

Deploying Rasa Open-Source servers as part of the Rasa Enterprise Helm chart will be deprecated in the future. If you are currently using this functionality, note that resource requirements for the rasa-production and rasa-worker deployments is highly dependent on the amount of training data, settings of the training config, and the resulting size of the model being served. The resources below are suggestions to start out with only: Deployment CPU Memory, rasa-production 2 2 GiB, rasa-worker 4 4 GiB. We recommend a size of 10 GiB for the Rasa Enterprise volume claim and 30 GiB for the database volume claim.

Rasa is an open-source framework for building chatbots and virtual assistants. The methodology of building a Rasa chatbot typically involves the following steps:

Define the use case: Identify the specific problem or task that the chatbot is going to solve. This will help determine the conversation flow, user interface, and required integrations.

Define the conversation flow: Map out the conversation flow using a tool like a flowchart or a mind map. This helps to identify the user inputs and the bot responses at different stages of the conversation.

Define the domain: Define the domain of the chatbot, which includes the intents, entities, and actions. Intents represent the user's intention behind the message, entities represent the specific information the user is asking for, and actions represent the response or action that the bot should take.

Train the NLU model: NLU (Natural Language Understanding) is the component that interprets the user's input and extracts the intents and entities. Train the NLU model using the data you have collected for the domain.

Train the Dialogue model: The dialogue model is responsible for predicting the next action based on the current state of the conversation. Train the dialogue model using the data collected for the conversation flow.

Integrate with external APIs: If the chatbot needs to interact with external APIs, such as a weather API or a payment gateway, integrate the APIs with the chatbot.

Test the chatbot: Test the chatbot using a combination of automated and manual testing. Use the test results to refine and improve the chatbot.

Deploy the chatbot: Once the chatbot is ready, deploy it on a server or cloud platform so that users can access it.

Monitor and maintain the chatbot: Monitor the chatbot for performance, errors, and user feedback. Continuously improve the chatbot by adding new features, fixing bugs, and optimizing performance.

Focus on the user experience: The chatbot should be designed with the user in mind, ensuring that the conversation flow is natural and intuitive.

Collect high-quality training data: The accuracy of the chatbot's responses depends heavily on the quality of the training data. Therefore, it is important to collect and label high-quality data for the intents, entities, and actions.

Continuously train and improve the models: The NLU and dialogue models should be regularly trained and updated with new data to ensure that the chatbot stays accurate and up-to-date with the latest information.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

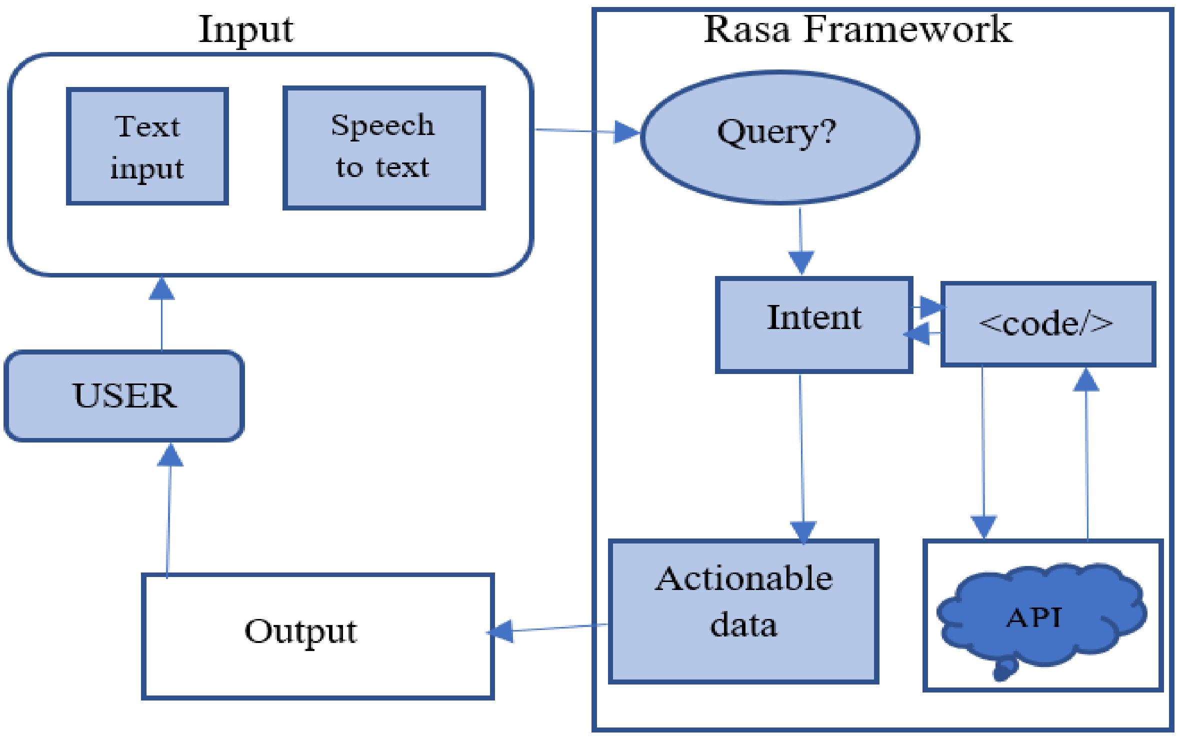


Fig. 4.2 System Architecture

In Figure 4.2 it shows the RASA NLU (an interpreter) receives this message from the end user and produces a structured output that includes plaintext as well as the intent and entities, if any, depicted in Figure. This monitor consistently keeps the discussion in a structured state and takes input from the interpreter. The policy receives the tracker output and acts on the tracker's current state as a result. This policy establishes the proper course of action. The tracker keeps a record of the selected action. The user is given the right response, which employs the same nlu.md-defined intent as the complete response.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

To implement a Rasa chatbot, you will primarily use Rasa Open Source, which is a Python-based framework for building conversational AI assistants. Rasa provides tools for developing both the Natural Language Understanding (NLU) and Dialogue Management components of the chatbot. In addition to Rasa OpenSource, you may also use other software and tools depending on your specific requirements.

4.4 PROJECT MANAGEMENT PLAN

4.4.1 STAGE OF NLU

The user's input is processed in this stage. At this stage, the chatbot will primarily strive to comprehend what the user is attempting to inquire. An intent from a trained set of intents is matched with the input that is provided. To achieve an accurate response from the chatbot, input must be mapped to the appropriate intent. Because different users may have diverse types of queries, a range of examples must be trained for intent categorization. User 18 input must be matched with the intended objective through adequate training. After the intent has been mapped, entities need to be extracted from the input.

4.4.2 STAGE OF ANSWER SELECTION

RASA Core is in charge of providing answers to the users' questions. For each intent, a series of pre-defined action sequences will be used. The Core will choose the action to be taken for each user query. Actions can be static or dynamic. Static actions that start with an absolute keyword cause only certain static messages to be read. Illustration of a possible static action is Find courses in CS, please. Look for CS courses, please. Give me a list of computer courses, please. Please provide me the list of CS courses. In each of these situations, the chatbot must understand that the entity is "CS" and the intent is "search for courses". What users want is expressed in their intentions. A piece of information derived from a query is called an entity. Only a bot that can accurately extract this set of intent entities will be able to produce the desired results. The model's training and subsequent mapping of intent entities will be handled by the RASA NLU.

4.4.3 STAGE OF OUTPUT

The user must receive the response that the model has chosen. The ultimate result is provided by the output stage, also known as the Natural Language Generation step. The output can take many different shapes. It could beexpressed verbally or through written words. Another text-to-voice conversionmechanism must be used if the output is speech. Lastly, a variety of channels can be used to obtain the outputs.

CHAPTER 5
IMPLEMENTATION DETAILS

5.1 DEVELOPMENT SETUP

Determine the goal of your chatbot, and what tasks you want it to perform. Gather training data and Collect examples of user input and expected responses. This data will be used to train the machine learning models that power the chatbot. Define intents and entities as an intent is the goal or purpose of a user's message, while an entity is a piece of information within that message. You'll need to define these for your chatbot to understand user input. Design the conversation flow and determine the different paths the conversation can take and how the chatbot should respond in each scenario. Train the chatbot and use the training data and machine learning algorithms to train the chatbot. Test the chatbot and test the chatbot with sample inputs and evaluate its performance. Deploy the chatbot, oncethe chatbot is trained and tested, deploy it to a platform or integrate it with your website or application.

Chatbots can be as simple as rudimentary programs that answer a simple query with a single-line response, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of personalization as they gather and process information. For example, you probably asked questions to digital customer services or other online chat services. During such a conversation, the chatbot knows how to answer your questions, ask questions, or possibly refer you to a website where your questions can be answered. In the past, chatbots were based entirely on pre-programmed rules, so they had nothing to do with artificial intelligence (AI). Today, however, AI is increasingly used in implementing chatbots, thanks to the enormous developments in natural language processing (NLP). This development is essential because for a chatbot to have some degree of “intelligence” it is necessary it has a certain understanding of language. But howdo we teach a computer to understand language and how does a chatbot applythis language understanding.

In System Engineering and Software Engineering, performance analysis comprises the duties that go into establishing the demands or criteria to satisfy for a new or revised one, while taking into consideration the sometimes-competing requirements of the many stakeholders, such as beneficiaries or users. The success of a development project is dependent on performance analysis. Allrequirements must be documented, actionable, quantifiable, testable, tied to identified business needs or opportunities, and described to a degree of detail adequate for system design. Architectural, structural, behavioural, functional, and non-functional requirements can all be specified. Rasa framework is differentbecause it provides more tools for developers and testers. We can test chatbots in many ways and understand how to improve. We can write test stories, assessing nlu model, checking its performance, assess the intent or entity performance.

Validating Data and Stories

Data validation verifies that no mistakes or major inconsistencies appear in your domain, NLU data, or story data. To validate your data, have your CI run this command: rasa data validate. If you pass a max_history value to one or more policies in your config.yml file, provide the smallest of those values as rasa data validate –max-history . If data validation results in errors, training a model can also fail or yield bad performance, so it's always good to run this check before training a model. By including the - -fail-on-warnings flag, this step will fail on warnings indicating more minor issues.

Writing Test Stories

Testing your trained model on test stories is the best way to have confidence in how your assistant will act in certain situations. Written in a modified story format, test stories allow you to provide entire conversations and test that, given certain user input, your model will behave in the expected manner. This is especially important as you start introducing more complicated stories from user conversations.

Test stories are like the stories in your training data, but include the user message as well. By default, the command will run tests on stories from any files with names starting with test_. You can also provide a specific test stories file or directory with the -- stories argument. You can test your assistant against them by running: rasa test

Evaluating an NLU Model

In addition to testing stories, you can also test the natural language understanding (NLU) model separately. Once your assistant is deployed in the real world, it will be processing messages that it hasn't seen in the training data. To simulate this, you should always set aside some part of your data for testing. You can either: 1) use a held-out test set by shuffling and splitting your NLU data. 2) use cross-validation, which automatically creates multiple train/test splits. It isnecessary to determine the test and production environments, as well as the tools and resources accessible to the specific team. Finally, it is the process of establishing the users' expectations for a new or changed programme. It includes all of the actions that are performed to identify the needs of recorded, actionable, quantifiable, tested, and traceable, which aids in the identification of opportunities and is specified to enable system design. It is critical for project success to examine project requirements when they are obtained as well as during the project's lifespan. It aids in keeping the criteria relevant to the situation. So, this is how we all look at each need at the right level.

Finally, it must verify that each need is atomic, uniquely recognised, and complete, among other things. It might be difficult to determine which component of the system constitutes the critical route. It is usually useful to include a statement about the projected peak number of people who may be expected to utilise the system during peak hours. It allows us to fulfil a variety of functions, including demonstrating that the system meets performance standards. Many of these actions are conducted without a suitably realistic environment. goal-oriented performance objectives This project was created with the general public in mind, who have very little expertise of computer operation but can quickly obtain their connected information. Finally, it is the process of establishing the user's expectations for the necessary Application.

5.2 DEPLOYMENT SETUP

5.2.1 PYTHON

Rasa is built using Python, so you will need to have Python installed on your machine to develop and run a Rasa chatbot. It is built on top of Python and relies on popular Python libraries like TensorFlow, spaCy, and scikit-learn for natural language processing (NLP) and machine learning (ML) tasks.

Rasa provides a set of Python packages that you can use to build different components of a chatbot, such as the NLU (Natural Language Understanding) engine, dialogue management, and integration with external APIs. Additionally, Rasa provides a CLI (command-line interface) tool that makes it easy to train, test, and deploy your chatbot.

Some of the key features of Python include:

1. Easy to learn: Python has a simple syntax and a straightforward learning curve, making it easy for beginners to pick up and start coding.
2. Interpreted: Python is an interpreted language, which means that code can be executed directly without needing to be compiled beforehand.
3. Cross-platform: Python is available for a wide range of platforms, including Windows, macOS, Linux, and even mobile devices like Android and iOS.
4. Large standard library: Python has a large and comprehensive standard library, which includes modules for everything from string manipulation to network programming.
5. Extensible: Python is highly extensible, with a large ecosystem of third-party packages and libraries available for a wide range of applications, including web development, scientific computing, and machine learning.
6. Dynamically typed: Python is dynamically typed, which means that variable types are determined at runtime rather than at compile time.

5.2.2 VS CODE SOFTWARE

Visual Studio Code is a lightweight and versatile code editor developed by Microsoft. It has excellent support for Python, and you can use it to build and debug your Rasa chatbot.
Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

Some of the key features of Visual Studio include:

Code editing: Visual Studio provides a powerful code editor with support for a wide range of programming languages, including C#, C++, Python, and JavaScript. It includes features such as syntax highlighting, code completion, and refactoring tools to help developers write code more efficiently.

Debugging: Visual Studio includes a powerful debugging system that allows developers to find and fix issues in their code. It provides tools for setting breakpoints, stepping through code, and inspecting variables and objects.

Testing: Visual Studio includes tools for testing code, including unit testing and load testing. It allows developers to create and run tests, as well as analyze and report on test results.

Project management: Visual Studio includes tools for managing projects, including version control, project templates, and task management. It allows developers to collaborate with others, track changes to code, and manage project resources.

Extensibility: Visual Studio is highly extensible, with a wide range of extensions and add-ons available from the Visual Studio Marketplace. These extensions can add new features and functionality to the IDE, as well as integrate it with other tools and platforms.

5.2.3 ANACONDA

Anaconda is the open-source Graph Database that is developed using Java technology. It is highly scalable and schema-free (NoSQL). Unlike traditional databases which store data in rows, columns, and tables, it has a completely flexible structure. Therefore, it saves relationships that connect data. The architecture is for traversal of nodes and relationships, optimal management, and storage. Each node contains direct pointers to all the nodes that are connected by relationships.

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

Some of the key features of Anaconda include:

1. Package management: Anaconda includes a powerful package management system that allows users to easily install, update, and remove packages and libraries from their Python or R environment.
2. Environment management: Anaconda provides a way to create and manage multiple isolated environments, each with their own set of packages and dependencies. This makes it easy to work on different projects without worrying about package conflicts.
3. Data science libraries: Anaconda comes with a wide range of data science libraries pre-installed, including NumPy, Pandas, Matplotlib, scikit-learn, and TensorFlow.
4. Integrated development environment (IDE): Anaconda includes an IDE called Anaconda Navigator, which provides a graphical user interface (GUI) for managing environments, packages, and projects.

5. Collaboration and sharing: Anaconda provides a way to easily share and collaborate on projects with others, using tools like Jupyter notebooks and version control systems like Git.

5.2.4 GRAPH DATABASE

A graph database is a database used to model the data in the form of a graph. Here, the nodes of a graph depict the entities while the relationships depict the association of these nodes. Most of the data exists in the form of the relationship between different objects, and more often, the relationship betweenthe data is more valuable than the data 14 they usually do a search through a separate data structure called "index" which is expensive and makes the databaseslower, whereas graph databases store relationships and connections as first- class entities. Also, can easily retrieve (traverse) connected data faster by indexing starting point and then just chasing the memory pointer.

Text editor or IDE: You will need a text editor or Integrated Development Environment (IDE) to write your code. Examples include Visual Studio Code, PyCharm, or Sublime Text.

Git: Git is a version control system that allows you to track changes to your code and collaborate with other developers. It is recommended to use Git to manage your Rasa project.

GitHub: GitHub is a web-based platform that hosts Git repositories and allows for collaboration and sharing of code.

Deployment platform: To deploy your Rasa chatbot, you will need to choose a platform to host your application. Examples include Heroku, AWS, or Google Cloud.

5.3 ALGORITHMS

For NLU, Rasa 2.0 uses a supervised machine learning algorithm called "DIET" (Dual Intent and Entity Transformer). DIET is a deep neural network that combines intent classification and entity recognition into a single model, making it more efficient than previous Rasa NLU algorithms. DIET uses pre-trained word embeddings, character-level embeddings, and intent-specific embeddings to improve its performance on low-shot intent and entity recognition tasks. 19

For Dialogue Management, Rasa 2.0 uses a rule-based approach for handling simple dialogues and a machine learning-based approach for handling more complex conversations.

The machine learning-based approach is based on the "TED" (Transformer Embedding Dialogue) algorithm, which is a deep neural network that uses self-attention mechanisms to learn contextual representations of the dialogue history. TED can handle multi-turn conversations and can learn to predict the next best action to take based on the current state of the conversation.

A diet classifier is a machine learning model that classifies food items into different categories based on their nutritional content. The model is trained on a dataset of food items and their corresponding nutrient values such as protein, fat, carbohydrates, vitamins, minerals, etc. To build a diet classifier, you would first need to gather a dataset of food items with their nutrient values. This dataset can be obtained from various sources such as the USDA Food Composition Databases or other publicly available nutrition databases. Once you have the dataset, you can perform feature engineering to extract relevant features from the nutrient values.

For example, you can calculate the calorie count, macronutrient ratios, or micronutrient levels of each food item. Next, you would split the dataset into training and testing sets and train a machine learning model on the training set. There are several algorithms that can be used for this task, including decision trees, random forests, support vector machines, and neural networks. After training the model, you would evaluate its performance on the testing set and fine-tune its parameters if necessary to improve its accuracy.

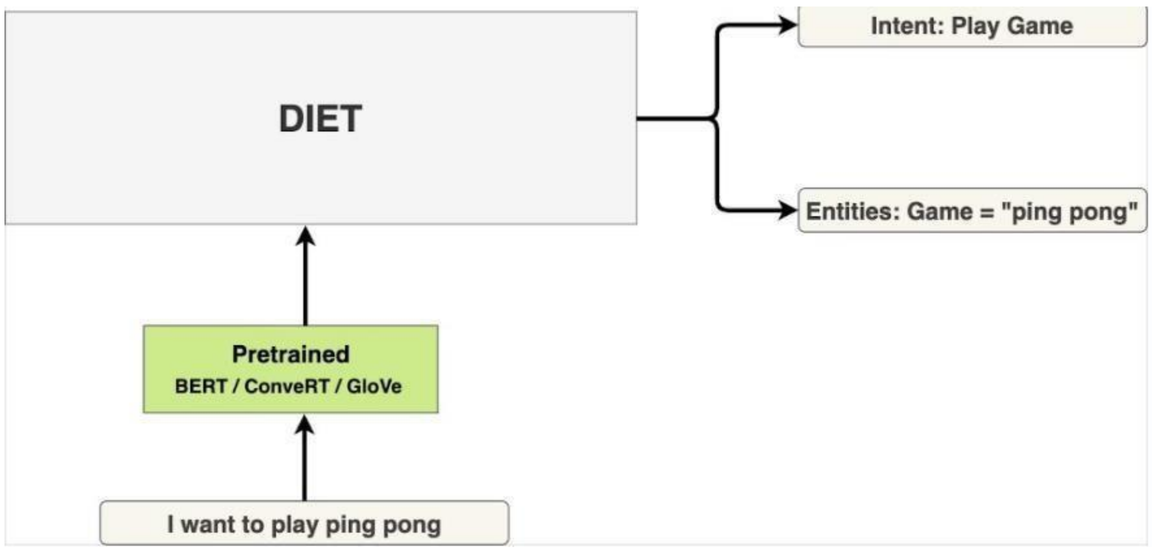


Fig. 5.3 IDEATION FLOW

In Figure 5.3 it shows the DIET uses a sequence model that takes word order into account, thereby offering better performance. It's also a more compact model with a plug-and-play, modular architecture. For instance, you can use DIET to do both intent classification and entity extraction; you can also perform a single task, for example, configure it to turn off intent classification and train it just for entity extraction.

CHAPTER 6
RESULTS AND DISCUSSION

RESULTS

We have successfully proposed a chatbot system to simulate a human conversation. This application presents sophisticated chatbot for users especially during unknown pandemics like covid 19. We can implement face recognition, better voice recognition for different medical cases pandemic. In future we can handle all types of diseases by updating the medical data given. We can expect 24/7 services for easy handling. It summarizes the features of medical related chatbots, how to utilise them, and the drawbacks of applying this cutting-edge technology in the event of a pandemic.

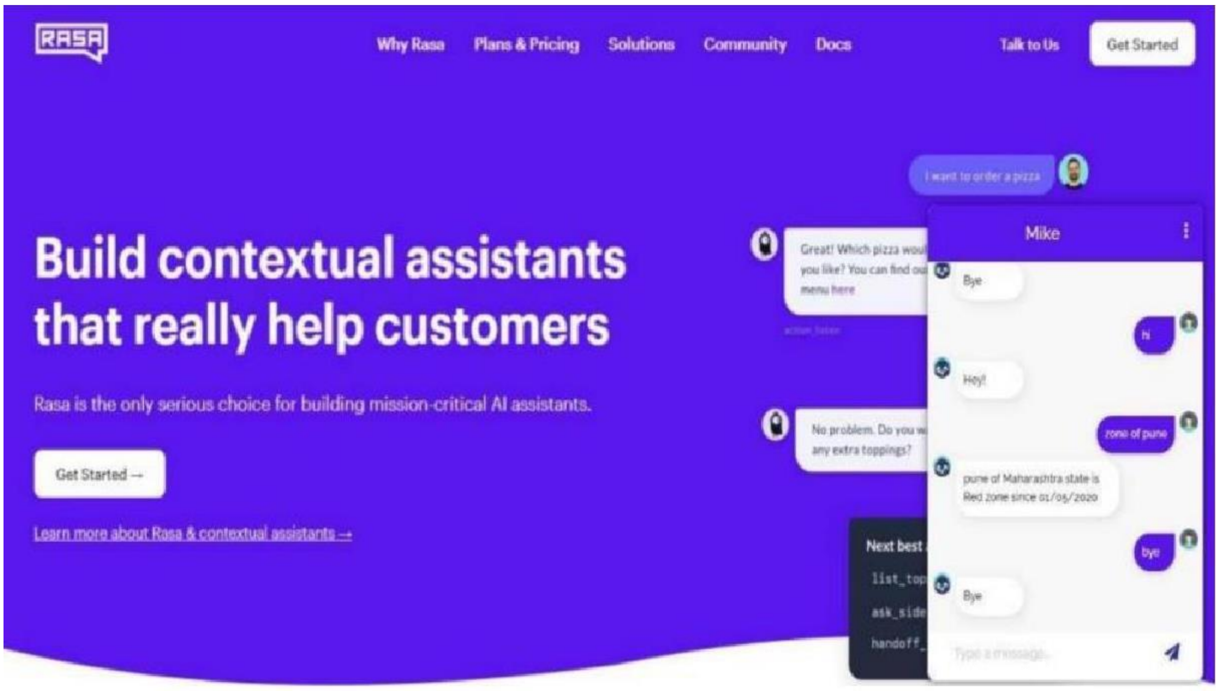
DISCUSSION

It summarizes the features of medical related chatbots, how to utilise them, and the drawbacks of applying this cutting-edge technology in the event of a pandemic. The usage of health chatbots to combat COVID19 is still in its early stages. The researcher will be able to better grasp these innovative technological designs and applications with the aid of our findings as he continues to develop medical related chatbot operable and applications in this battle upon COVID-19.

Chatbots can be as simple as rudimentary programs that answer a simple query with a single-line response, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of personalization as they gather and process information. For example, you probably asked questions to digital customer services or other online chat services. During such a conversation, the chatbot knows how to answer your questions, ask questions, or possibly refer you to a website where your questions can be answered. In the past, chatbots were based entirely on pre-programmed rules, so they had nothing to do with artificial inteligence (AI). Today, however, AI is increasingly used in implementing chatbots,thanks to the enormous developments in natural language processing (NLP).

This development is essential because for a chatbot to have some degree of “intelligence” it is necessary it has a certain understanding of language. But how do we teach a computer to understand language and how does a chatbot apply this language understanding. We will cover the theoretical and technical background of a chatbot based on AI. The coronavirus outbreak has major consequences for society worldwide. People are rightly concerned and have manyurgent questions.

The World Health Organization provides answers to frequently asked questions regarding the coronavirus. However, you may have to search for a while before you have found the right answer to your question. It is vital that people are well informed about current measures. This way we can efficiently limit mass spread. A chatbot could perfectly help with this! In this blog post, we are going to create a chatbot for questions regarding the coronavirus. This will be based on artificial intelligence (AI) and natural language processing (NLP). We would be using AI and NLP used for this chatbot. Hereafter, the implementation of the chatbot will be demonstrated based on the discussed techniques. This project is tounderstand how to help stop the coronavirus by creating the right awareness and learn a bit about AI and NLP in the process.



Build contextual assistants that really help customers

Rasa is the only serious choice for building mission-critical AI assistants.

I want to order a pizza

Great! Which pizza would you like? You can find our menu here

No problem. Do you want any extra toppings?

Bye

hi

Hey!

zone of pune

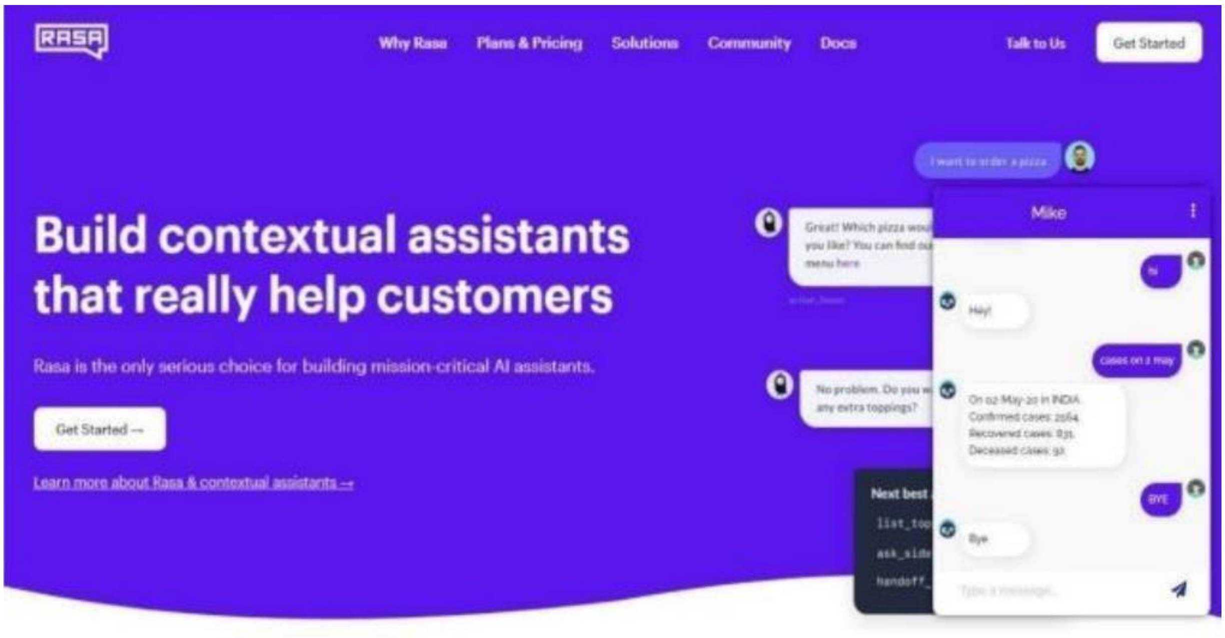
pune of Maharashtra state is Red zone since 01/05/2020

bye

Bye

Fig. 6.1 Zone of a State

In the Figure 6.1 shows the bot where we ask for state and it responses as the shown cases in pune. For example If we give the input as Pune, it shows the cases in pune and active cases.



Build contextual assistants that really help customers

Rasa is the only serious choice for building mission-critical AI assistants.

I want to order a pizza

Great! Which pizza would you like? You can find our menu here

No problem. Do you want any extra toppings?

hi

Hey!

cases on 2 may

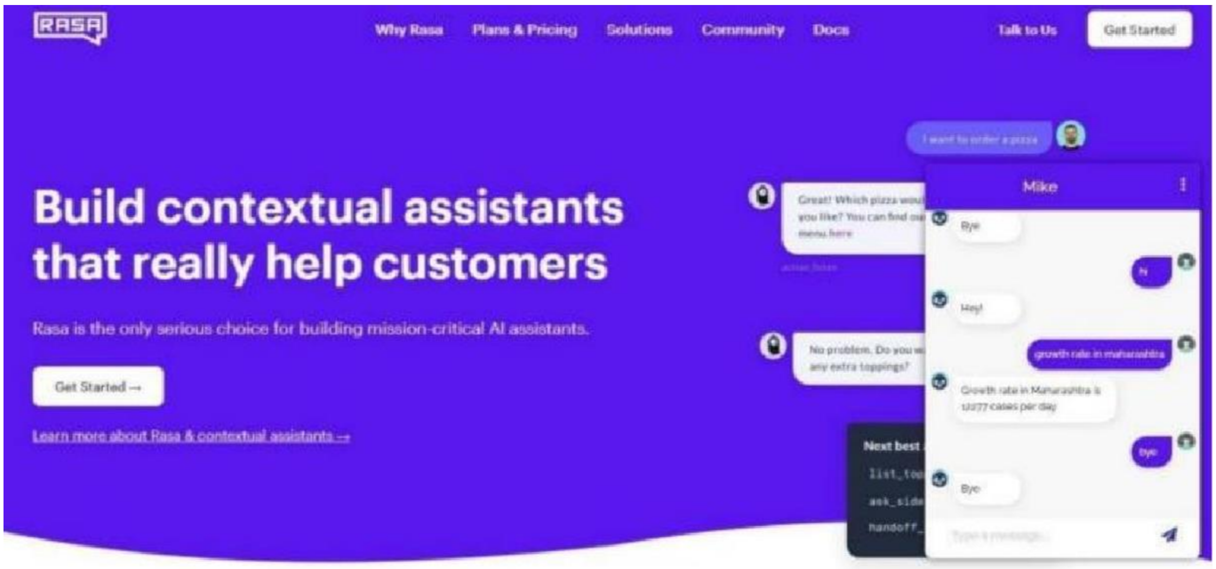
On 02-May-20 in INDIA
Confirmed cases: 2564
Recovered cases: 831
Deceased cases: 92

BYE

Bye

Fig. 6.2 Cases on State

In the Figure 6.2 shows the active cases on the output asked by the user to view the cases on date. For example, if we give the input as some state, it gives us the accurate cases on day and recovered cases too.



Build contextual assistants that really help customers

Rasa is the only serious choice for building mission-critical AI assistants.

I want to order a pizza

Great! Which pizza would you like? You can find our menu here

No problem. Do you want any extra toppings?

Bye

hi

Hey!

growth rate in maharashtra

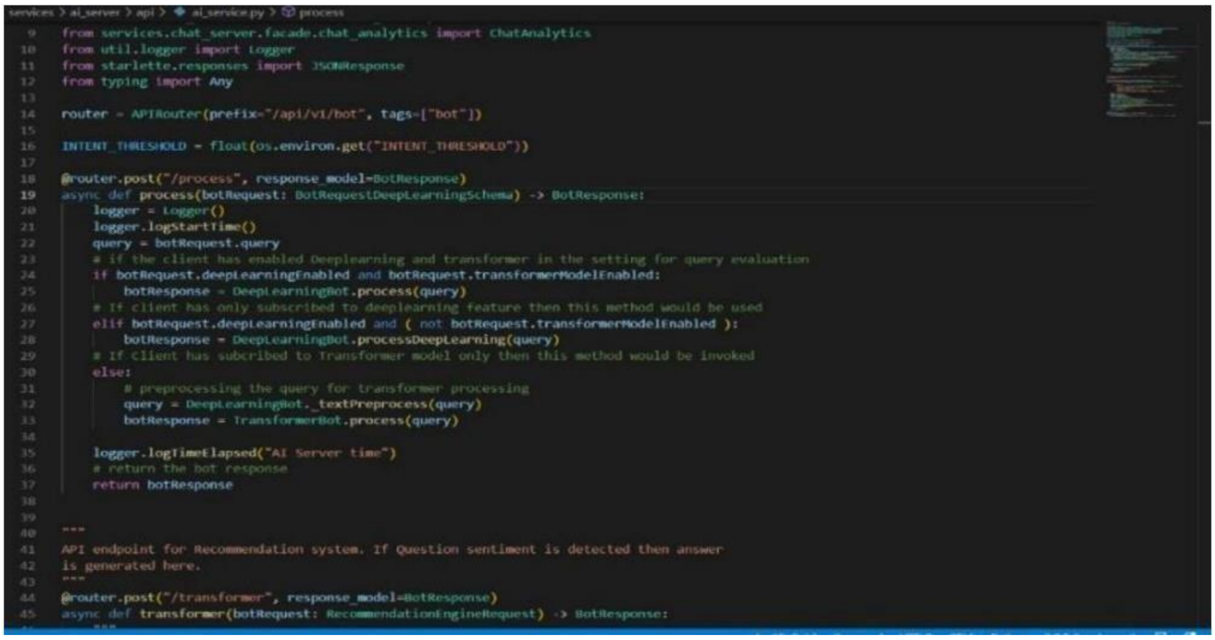
Growth rate in Maharashtra is 12277 cases per day

bye

Bye

Fig. 6.3 Growth Rate

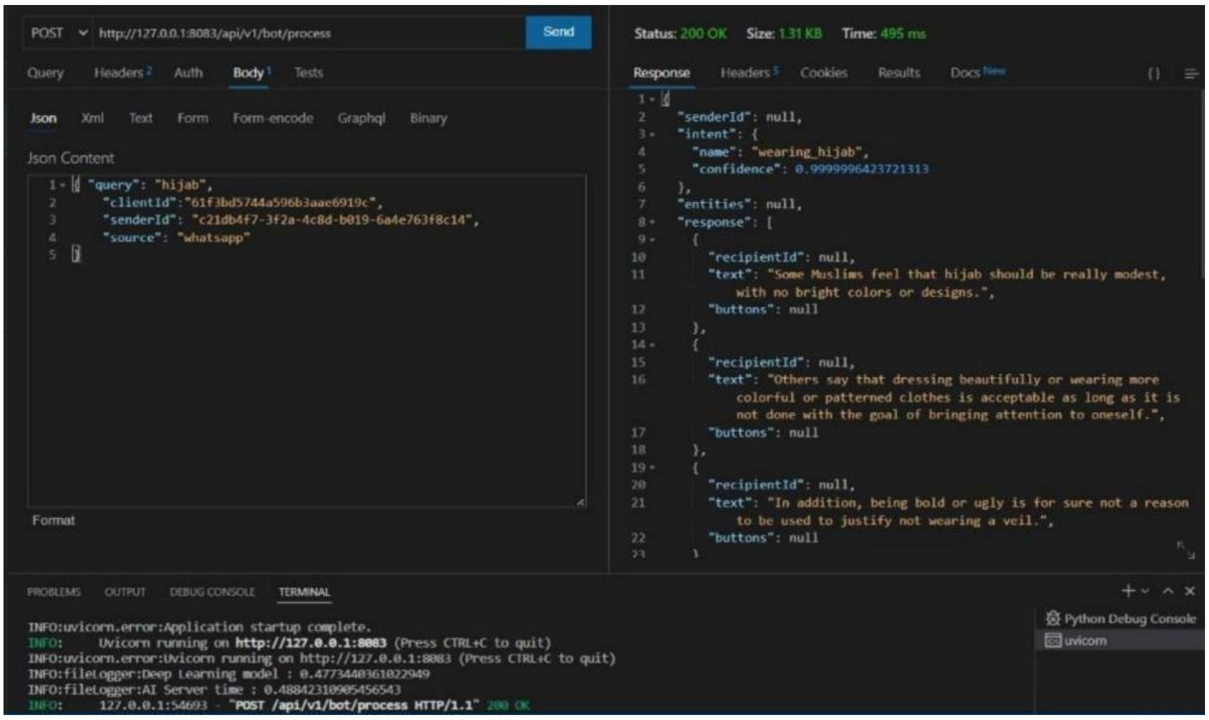
In the Figure 6.3 shows the growth rate of the day, date and time. It also gives the exact growth rate of each and every day, even months and years.



```python
from services.chat_server.facade.chat_analytics import ChatAnalytics
from util.logger import Logger
from starlette.responses import JSONResponse
from typing import Any

router = APIRouter(prefix="/api/v1/bot", tags=["bot"])

INTENT_THRESHOLD = float(os.environ.get("INTENT_THRESHOLD"))

@router.post("/process", response_model=BotResponse)
async def process(botRequest: BotRequestDeepLearningSchema) -> BotResponse:
    logger = Logger()
    logger.logStartTime()
    query = botRequest.query
    # if the client has enabled Deeplearning and transformer in the setting for query evaluation
    if botRequest.deepLearningEnabled and botRequest.transformerModelEnabled:
        botResponse = DeepLearningBot.process(query)
    # If client has only subscribed to deeplearning feature then this method would be used
    elif botRequest.deepLearningEnabled and ( not botRequest.transformerModelEnabled ):
        botResponse = DeepLearningBot.processDeepLearning(query)
    # If Client has subcribed to Transformer model only then this method would be invoked
    else:
        # preprocessing the query for transformer processing
        query = DeepLearningBot._textPreprocess(query)
        botResponse = TransformerBot.process(query)

    logger.logTimeElapsed("AI Server time")
    # return the bot response
    return botResponse


"""
API endpoint for Recommendation system. If Question sentiment is detected then answer
is generated here.
"""
@router.post("/transformer", response_model=BotResponse)
async def transformer(botRequest: RecommendationEngineRequest) -> BotResponse:
```

Fig. 6.4 AI Server

In the Figure 6.4 describes about the AI server used in this project and the AI sever code is been used in VS CODE software.



```
http://127.0.0.1:8083/api/v1/bot/process
```

```
{ "query": "hijab",
    "clientId":"61f3bd5744a596b3aae6919c",
    "senderId": "c21db4f7-3f2a-4c8d-b019-6a4e763f8c14",
    "source": "whatsapp"
}
```

Status: 200 OK   Size: 1.31 KB   Time: 495 ms

```
{
  "senderId": null,
  "intent": {
    "name": "wearing_hijab",
    "confidence": 0.9999996423721313
  },
  "entities": null,
  "response": [
    {
      "recipientId": null,
      "text": "Some Muslims feel that hijab should be really modest,
         with no bright colors or designs.",
      "buttons": null
    },
    {
      "recipientId": null,
      "text": "Others say that dressing beautifully or wearing more
         colorful or patterned clothes is acceptable as long as it is
         not done with the goal of bringing attention to oneself.",
      "buttons": null
    },
    {
      "recipientId": null,
      "text": "In addition, being bold or ugly is for sure not a reason
         to be used to justify not wearing a veil.",
      "buttons": null
    }
```

```
INFO:uvicorn.error:Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8083 (Press CTRL+C to quit)
INFO:uvicorn.error:Uvicorn running on http://127.0.0.1:8083 (Press CTRL+C to quit)
INFO:fileLogger:Deep Learning model : 0.4773440361022949
INFO:fileLogger:AI Server time : 0.48842310905456543
INFO:     127.0.0.1:54693 - "POST /api/v1/bot/process HTTP/1.1" 200 OK
```

Fig. 6.5 Different Queries in AI Server

In the Figure 6.5 shows the different input given as the different queries in AI server, so that it responds many queries. We can keep on add queries and inputs so that it will respond all.

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

In this paper, we have successfully created a chatbot that responds to all covid related queries. One of the main advantages of using Rasa is its flexibility and customizability. Developers have full control over the design and behaviour of their chatbots, and can train them to handle specific use cases and workflows. Rasa's NLU and dialogue management capabilities are also among the best in the industry, allowing chatbots to accurately interpret user inputs and provide relevant responses.

However, building a successful chatbot with Rasa requires significant time and effort. Developers must have a good understanding of machine learning concepts and NLP techniques, and must spend time collecting and annotating training data to train the chatbot effectively. Additionally, Rasa lacks some of the features and integrations available in other chatbot platforms, which may make it less suitable for some use cases. Overall, Rasa is an excellent choice for developers looking for a flexible and customizable platform for building chatbots and voice assistants. With the right expertise and resources, it can be used to create powerful and engaging conversational AI agents that meet a wide range of business needs.

7.2 FUTURE SCOPE

We have successfully proposed a chatbot system to simulate a human conversation. This application presents sophisticated chatbot for users especially during unknown pandemics like covid 19. We can implement face recognition, better voice recognition for different medical cases pandemic. In future we can handle all types of diseases by updating the medical data given. We can expect 24/7 services for easy handling. It summarizes the features of medical related chatbots, how to utilize them, and the drawbacks of applying this cutting-edge technology in the event of a pandemic.

The usage of health chatbots to combat COVID19 is still in its early stages. The researcher will be able to better grasp these innovative technological designs and applications with the aid of our findings as he continues to develop medical related chatbot operable and applications in this battle upon COVID-19. Rasa chatbot can be improved with more advanced NLP capabilities, such as sentiment analysis, intent recognition, and entity extraction. This would make the chatbot more intelligent and accurate in understanding and responding to user queries. Rasa chatbot can be integrated with other technologies such as voice assistants, IoT devices, and smart home systems. This would enhance the chatbot's functionality and make it more user-friend. Rasa chatbot can be extended tosupport multiple languages, making it more accessible to users worldwide. Rasa chatbot can be improved with better analytics and reporting capabilities, allowing businesses to track user interactions and measure the chatbot's effectiveness. Rasa chatbot can be customized and personalized to meet the specific needs of different businesses and industries. This would allow companies to create a chatbot that reflects their brand and provides a unique user experience. Overall, Rasa chatbot has a bright future, with endless possibilities for development and improvement. As AI and machine learning continue to evolve, Rasa will undoubtedly keep up with the latest trends and advancements to deliver a superiorchatbot experience to users.

REFERENCES

[1] Battineni, Gopi, Nalini Chintalapudi, and Francesco Amenta. "AI chatbot design during an epidemic like the novel coronavirus." In Healthcare, vol. 8, no. 2, p. 154. Multidisciplinary Digital Publishing Institute, 2020.

[2] Bharti, Urmil, Deepali Bajaj, Hunar Batra, Shreya Lalit, Shweta Lalit, andAayushi Gangwani. "Medbot: Conversational artificial intelligence powered chatbot for delivering tele- health after covid19." In 2020 5th International Conference on Communication and Electronics Systems (ICCES), pp. 870-875. IEEE, 2020.

[3] Judson, Timothy J., Anobel Y. Odisho, Jerry J. Young, Olivia Bigazzi, David Steuer, Ralph Gonzales, and Aaron B. Neinstein. "Implementation of a digital chatbot to screen health system employees during the COVID-19 pandemic." Journal of the American Medical Informatics Association 27, no. 9 (2020): 1450-1455.

[4] Kondapi, R. & Katta, R.K. & Potluri, Sirisha & Bathula, A. & Basha, S.K. (2019). Pacifiurr: An android chatbot application for human interaction. International Journal of Recent Technology and Engineering.

[5] Lee, Hyeonhoon, Jaehyun Kang, and Jonghyeon Yeo. "Medical Specialty Recommendations by an Artificial Intelligence Chatbot on a Smartphone: Development and Deployment." Journal of medical Internet research 23, no. 5 (2021).

[6] Lei, Hannah, Weiqi Lu, Alan Ji, Emmett Bertram, Paul Gao, Xiaoqian Jiang, and Arko Barman. “COVID-19 Smart chatbot prototype for patient Monitoring(2021).

[7] Martin, Alistair, Jama Nateqi, Stefanie Gruarin, Nicolas Munsch, Isselmou Abdarahmane, Marc Zobel, and Bernhard Knapp. "An artificial intelligence-based

firstline defence against COVID-19: digitally screening citizens for risks via a chatbot." Scientific reports 10, no. 1 (2020)

[8] Matic, Rade, Milos Kabiljo, Miodrag Zivkovic, and Milan Cabarkapa. "Extensible chatbot architecture using metamodels of natural language understanding." Electronics 10, no. 18. 29

[9] Ouerhani, Nourchène, Ahmed Maalel, Henda Ben Ghézala, and Soulaymen Chouri. "Smart Ubiquitous Chatbot for COVID-19 Assistance with Deep learning Sentiment Analysis Model during and after quarantine." (2020).

[10] Windiatmoko, Yurio, Ahmad Fathan Hidayatullah, and Ridho Rahmadi. "Developing FB chatbot based on deep learning using RASA framework for university enquiries." (2020).

APPENDIX

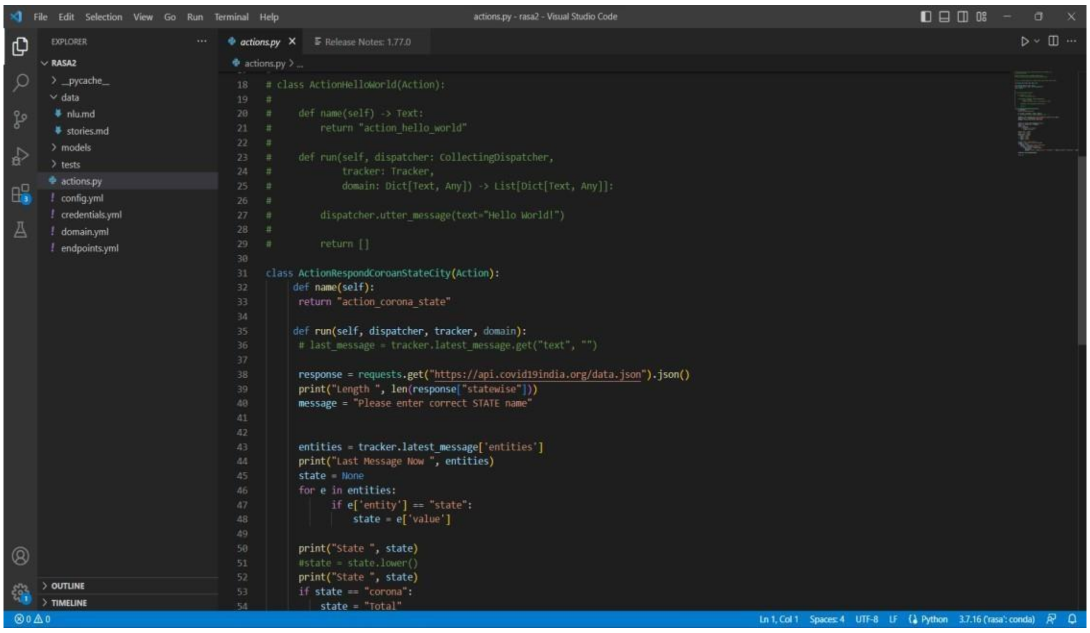
A. SOURCE CODE

```
This files contains your custom actions which can be used to run
 custom Python code.

 See this guide on how to implement these action:
 https://rasa.com/docs/rasa/core/actions/#custom-actions/

This is a simple example for a custom action which utters "Hello World!"
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
import requests


 class ActionHelloWorld(Action):

     def name(self) -> Text:
         return "action_hello_world"

     def run(self, dispatcher: CollectingDispatcher,
             tracker: Tracker,
             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

         dispatcher.utter_message(text="Hello World!")

         return []
class ActionRespondCoroanStateCity(Action):
     def name(self):
      return "action_corona_state"
     def run(self, dispatcher, tracker, domain):
```

```
 # last_message = tracker.latest_message.get("text", "")
      response = requests.get("https://api.covid19india.org/data.json").json()
      print("Length ", len(response["statewise"]))
      message = "Please enter correct STATE name"

      entities = tracker.latest_message['entities']
      print("Last Message Now ", entities)
      state = None
      for e in entities:
            if e['entity'] == "state":
                state = e['value']

      print("State ", state)
      #state = state.lower()
      print("State ",  state)
      if state == "corona":
          state = "Total"
      if state == "india":
          state = "Total"
      # print("State ", state.title())
      message = "Please enter correct STATE name"
      if(state != None):
          message = "Please enter correct STATE name"
          for data in response["statewise"]:
              if data["state"] == state.title():
                  print(data)
                  message = "Active: "+data["active"] +" Confirmed: " + data["confirmed"]
+" Recovered: " + data["recovered"] +" On "+data["lastupdatedtime"]
      dispatcher.utter_message(message)
      return []
```

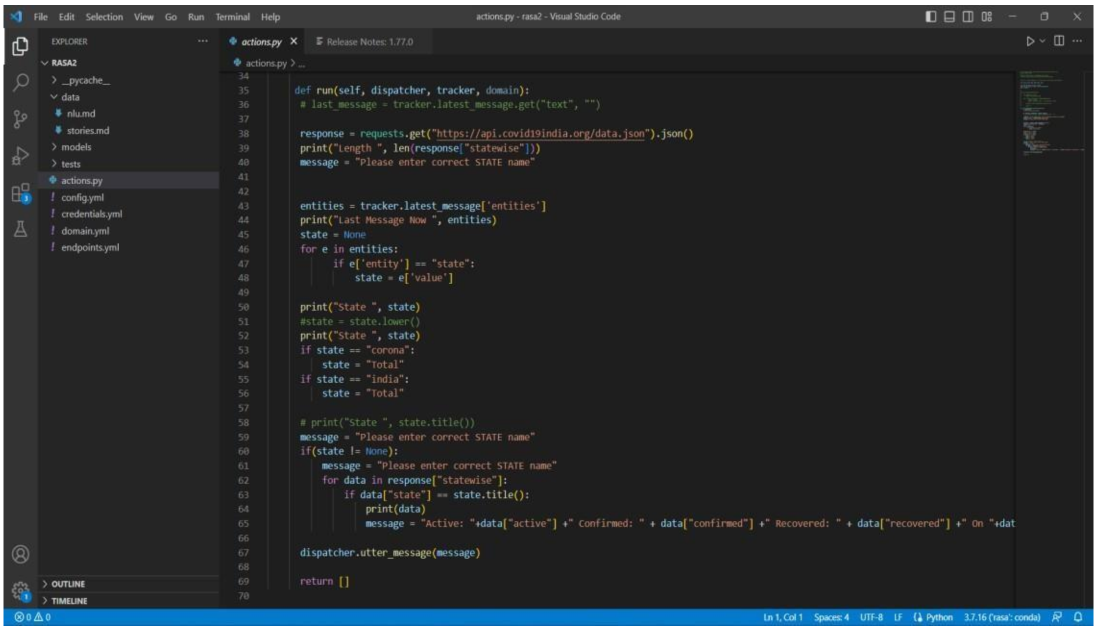
```
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/



language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy

intents:
-      greet
  - goodbye
  - corona_state
actions:
 - action_corona_state
entities:
 - state




responses:
  utter_greet:
  - text: "Hey! How are you?"
  utter_cheer_up:
  - text: "Here is something to cheer you up:"
  image: "https://i.imgur.com/nGF1K8f.jpg"
  utter_did_that_help:
  - text: "Did that help you?"
  utter_happy:
  - text: "Great, carry on!"
  utter_goodbye:
  - text: "Bye"
  utter_iamabot:
  - text: "I am a bot, powered by Rasa."
session_config:
session_expiration_time: 60
carry_over_slots_to_new_session: true
```

B. SCREENSHOTS



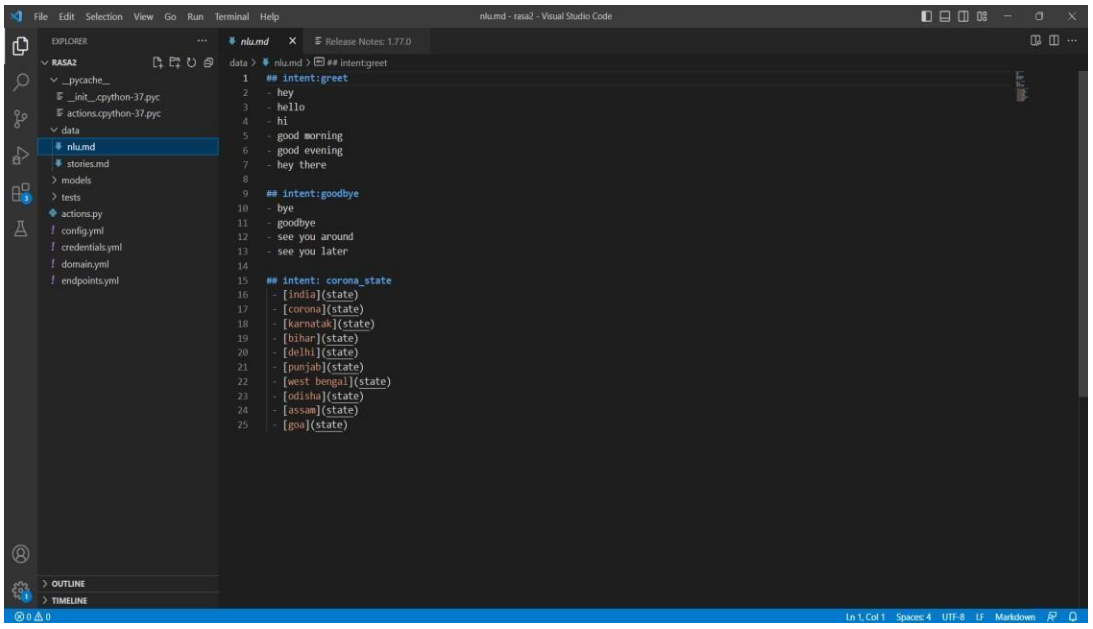
```
# class ActionHelloWorld(Action):
#
#     def name(self) -> Text:
#         return "action_hello_world"
#
#     def run(self, dispatcher: CollectingDispatcher,
#             tracker: Tracker,
#             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
#
#         dispatcher.utter_message(text="Hello World!")
#
#         return []

class ActionRespondCoroanStateCity(Action):
    def name(self):
     return "action_corona_state"

    def run(self, dispatcher, tracker, domain):
     # last_message = tracker.latest_message.get("text", "")

     response = requests.get("https://api.covid19india.org/data.json").json()
     print("Length ", len(response["statewise"]))
     message = "Please enter correct STATE name"


     entities = tracker.latest_message['entities']
     print("Last Message Now ", entities)
     state = None
     for e in entities:
           if e['entity'] == "state":
               state = e['value']

     print("State ", state)
     #state = state.lower()
     print("State ", state)
     if state == "corona":
         state = "Total"
```

Fig 6.6 Backend Source Code Screen



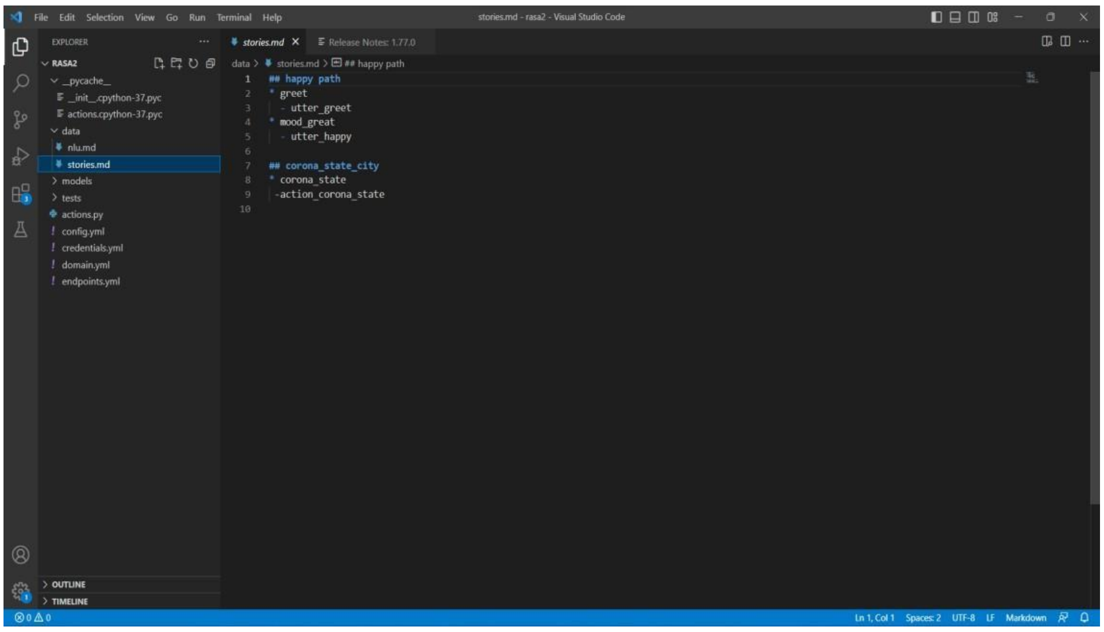
```
    def run(self, dispatcher, tracker, domain):
     # last_message = tracker.latest_message.get("text", "")

     response = requests.get("https://api.covid19india.org/data.json").json()
     print("Length ", len(response["statewise"]))
     message = "Please enter correct STATE name"


     entities = tracker.latest_message['entities']
     print("Last Message Now ", entities)
     state = None
     for e in entities:
           if e['entity'] == "state":
               state = e['value']

     print("State ", state)
     #state = state.lower()
     print("State ", state)
     if state == "corona":
         state = "Total"
     if state == "india":
         state = "Total"

     # print("State ", state.title())
     message = "Please enter correct STATE name"
     if(state != None):
         message = "Please enter correct STATE name"
         for data in response["statewise"]:
             if data["state"] == state.title():
                 print(data)
                 message = "Active: "+data["active"] +" Confirmed: " + data["confirmed"] +" Recovered: " + data["recovered"] +" On "+dat

     dispatcher.utter_message(message)

     return []
```

Fig 6.7 Backend Source Code Screen



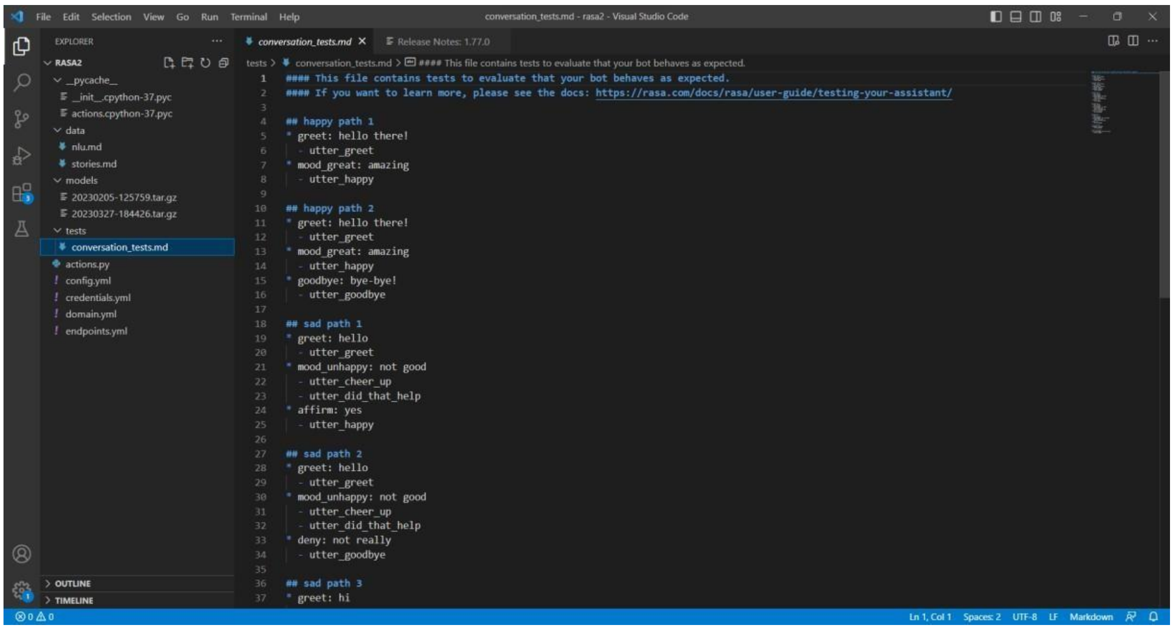
```
## intent:greet
- hey
- hello
- hi
- good morning
- good evening
- hey there

## intent:goodbye
- bye
- goodbye
- see you around
- see you later

## intent: corona_state
 - [india](state)
 - [corona](state)
 - [karnatak](state)
 - [bihar](state)
 - [delhi](state)
 - [punjab](state)
 - [west bengal](state)
 - [odisha](state)
 - [assam](state)
 - [goa](state)
```

Fig 6.8 Backend Source Code Screen



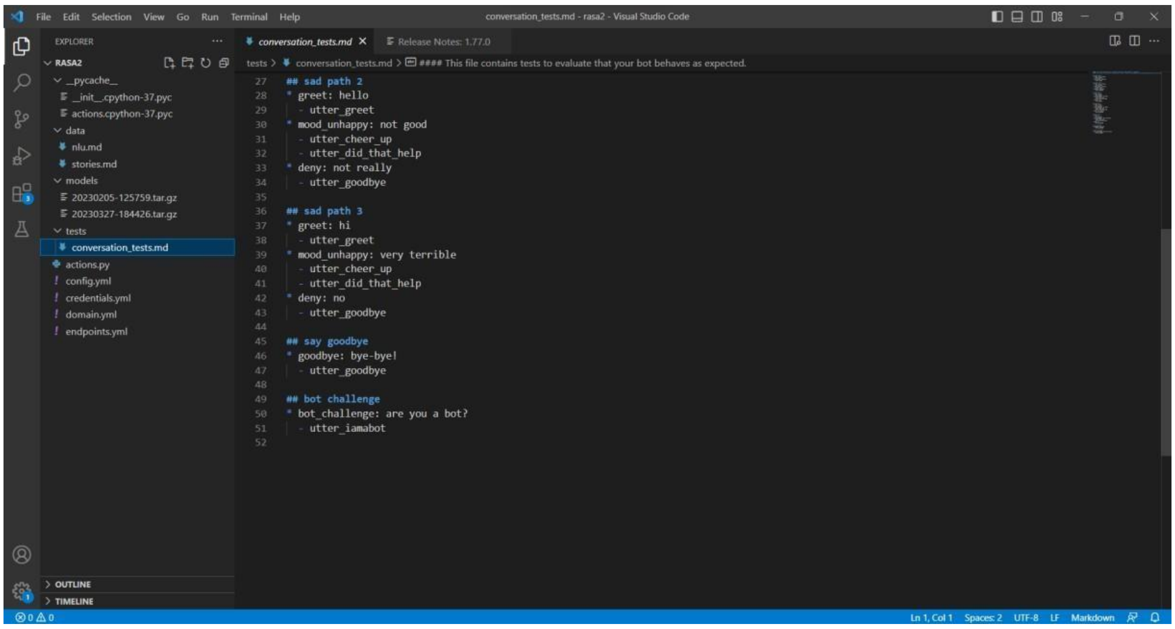
```
## happy path
* greet
  - utter_greet
* mood_great
  - utter_happy

## corona_state_city
* corona_state
 -action_corona_state
```

Fig 6.9 Dashboard Source Code Screen



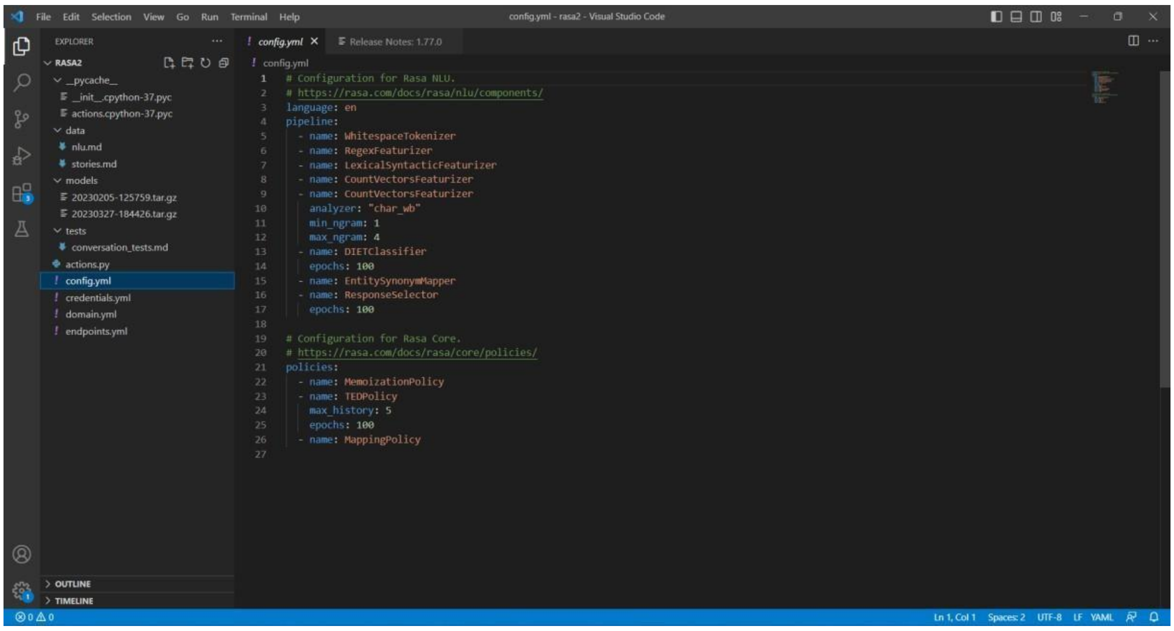
```
#### This file contains tests to evaluate that your bot behaves as expected.
#### If you want to learn more, please see the docs: https://rasa.com/docs/rasa/user-guide/testing-your-assistant/

## happy path 1
* greet: hello there!
  - utter_greet
* mood_great: amazing
  - utter_happy

## happy path 2
* greet: hello there!
  - utter_greet
* mood_great: amazing
  - utter_happy
* goodbye: bye-bye!
  - utter_goodbye

## sad path 1
* greet: hello
  - utter_greet
* mood_unhappy: not good
  - utter_cheer_up
  - utter_did_that_help
* affirm: yes
  - utter_happy

## sad path 2
* greet: hello
  - utter_greet
* mood_unhappy: not good
  - utter_cheer_up
  - utter_did_that_help
* deny: not really
  - utter_goodbye

## sad path 3
* greet: hi
```

Fig 6.10 Cookie Manager Source Code Screen



```
## sad path 2
* greet: hello
  - utter_greet
* mood_unhappy: not good
  - utter_cheer_up
  - utter_did_that_help
* deny: not really
  - utter_goodbye

## sad path 3
* greet: hi
  - utter_greet
* mood_unhappy: very terrible
  - utter_cheer_up
  - utter_did_that_help
* deny: no
  - utter_goodbye

## say goodbye
* goodbye: bye-bye!
  - utter_goodbye

## bot challenge
* bot_challenge: are you a bot?
  - utter_iamabot

```

Fig 6.11 App.Js. Source Code Screen



```yaml
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy

```

Fig. 6.12 Left Pane Source Code Screen

C. REASEARCH PAPER

RASA CHATBOT USING NLU

Suriya M S
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology
Chennai, India
suriyams2001@gmail.com

Selva A
Department of Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India
selvaaks45@gmail.com

A. Mary Posonia
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology Chennai, India
maryposonia.cse@sathyabama.ac.in

D. Usha Nandini
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology Chennai, India
Usha.cse@sathyabama.ac.in

Abstract—**We describe two open-source Python libraries for conversational software development, RASA Natural Language Understanding (NLU) and RASA Core. Youcan use these tools to create conversational software like the Messenger platform. Their goal is to make dialogue management and language interpretation based on AI (Artificial Intelligence) available to non-specialist developers. According to the design philosophies, we strive for usability as well as bootstrapping from little to no initial training data. Both packages come with a broad list of tests and are well-documented. As there are currently no widely used statistical conversation systems designed for non-experts, we present RASA NLU and RASA CORE as easy-to-use tools for building systems. conversation system. Thousands of developers utilise RASA on a daily basis all over the world. We divided our tools into RASA Core dialogue management and RASA NLU for natural language understanding, similar to many other conversational systems. The COVID-19 pandemic and the deluge of news from many sources have caused a lot ofindividuals to doubt their perceptions of the illness. The coronavirus COVID-19, which causes the Middle East Respiratory Symptoms (MERS) virus and the Severe Acute Respiratory Syndrome (SARS) virus, is also known as a coronavirus. A type of viral family called the coronavirus includes the influenzaand common cold virus types. Voice assistants and chatbots are becoming more and more common for connecting users to systems. The days of only using a keyboard to communicate with services are extinct. Developments in artificial intelligence and machine learning had a hand in the creation of these novel forms of user engagement. We will focus on developing chatbots specifically as a means of disseminating information via an Android app. With over 2.5 billion active users globally and more than 3 million devices installed, this Android software is undoubtedly going to be widely used. This study suggests using a chatbot to answer people's worries and get rid of information pollution, including false information on COVID-19. The chatbot is a well-liked communication system that uses a natural language processing aspect. The software known as chatbots enables communication between humans and computers.The chatbot was created for the Android operating system and uses the RASA open-source framework to answer to inquiries about COVID-19.**

Keywords—: *Chatbots, Conda, Natural Language Understanding, RASA, Python.*

I. INTRODUCTION

As we search for more organic ways to incorporate automation into people's daily lives, conversational systems are becoming more and more popular as the foundation for human-computer interaction. Apple's Siri, Amazon's Alexa, and Microsoft's Cortana are well-known instances of conversational AI, but conversational systems are gaining popularity as platforms like Facebook Messenger make chatbot development easier. For conversational systems, typical tasks include 1. Scheduling meetings 2. Booking flights as well as providing clientservice. Current open source libraries are highly regarded and include machine learning algorithm implementations. Maintaining a widely utilised project requires a significant amount of non-research work, and the code produced by research teams frequently falls short of expectations. Rasa Natural Language Understanding and Core seeks to act as a link between machine learning research and practical application by making recent advancements in the field accessible to non-specialists interested in implementing conversational AI systems. Since no statistical dialogue system has been successfully used bynon-specialists, we suggest Rasa NLU and RASA Core as simple tools for creating conversational systems. Thousands of developers use Rasa all over the world. These tools, like many other conversation systems, can be divided into two groups: Conversation Management and NaturalLanguage Understanding (Rasa NLU) (RasaCore). In section 5, we offer a sample application and detail the developer experience as well as the architecture of the code.

Software that can understand what people are doing is known as a chatbot. They are guided to the intended outcome because someone wants them. These chatbots use artificial intelligence, of course (AI). Processing of Language (NLP). The fundamental goal of chatbots is to make life easier for individuals and save them time. Agents may spend more time efficiently on high-priority activities by using chatbots to automate customer support and cut down on boring operations. Health, e-commerce, customer service, education,etc. all use chatbots. The larger one was trickier tohandle, as you mentioned. number of consumer calls. People had to wait longer in line to call toll- free numbers back then. Service providers find it difficult to receive calls again. Ask the chatbot a series of questions and do not delay in serving and responding to customers. The business will have positive growth. chatbot development is that you can describe the function, level of chatbot. Generally speaking, chatbots can be categorised

into five levels based on their use and

functionality. Broadcast chatbots are level 0 chatbots. Notice or message for the client. They are machines.

A program, not a person. To send large amounts of similar data customers use these bots. Level 1

chatbots are used to answer frequently asked questions. It can be challenging for everyone to consistently provide the same answers in many contact centres. Prioritization of Level 1 bots is done by a sequence of questions and responses. In order to save time and money, these bots continue to respond on behalf of a person. Chatbots used on social media and e-commerce sites are level 2 bots. Website users are helped by these robots. They serve as virtual assistants and contextualise the material underneath. Level 3 robot is sophisticated and very customizable. They keep tabs on your everyday activity and add past users' preferences to the finished product. Level 4 chatbots will be considered, the robots of the future, where many robots interact and learn from each other for best results. By integrating an education chatbot, many colleges, students anddepartments benefit. Some ML and AI-enabledchatbots do the same, allowing users to interact with students and understand their thoughts. It provides a personalized learning system with the user. Many chatbots are already available on the market. Examples include Siri, Alexa from Amazon, and Google's Voice Assistant. By communicating with you, these helpers or botsaim to do this. It understands what the user needs and delivers the right results. The strong development of chatbots will be observed in the healthcare sector, the customer area.

Services, e-commerce site. but much has not been done to effectively use chatbots in education.

II. LITERATURE REVIEW

In this paper

[1] Bocklisch, T., et al. originally released theRasa NLU and kernel under an open source licence. The purpose of this research is to developa dialogue system based on machine learning and language comprehension for computer enthusiastswho are not very tech-savvy.

[2] They have created an incredibly small package but have come a long way. 244 rasa releases were made with a total of 18023 commits thanks to the participation of 344 contributors.

[3] For the non-experts who don't know much

about the inner workings of the chatbot and treat

the chatbot as a black box, Lacerda has taken the heart of rasa and created a new software stack called Rasa -ptbr-boilerplate. Today, chatbots are intelligent machines capable of performing difficult tasks. They have a number of applications in robotics and natural language processing. Due to recent advances in artificialintelligence, chatbots now act as customer service representatives.

[4] RASA was found to be more adaptable than other commercial software because to itsscalability and open source licence when compared to Microsoft Bot, Google Dialogflow, and rasa chatbot in the study.

[5] In order to move away from the notion that chatbots may be used with social networks and towards a smart chat system, this article examines the technologies currently in use and builds the use case prototype. A social chatbot for football was created in this project to respond to inquiries from the Spanish football league. With a loose client and text-based communication, chatbots are implemented. Information about football players, their teams, and coaches is extracted.

[6] Eleni Adamopoulou & Lefteris Moussiades., 2020 Designing a chatbot's goals, processes, and user needs is the first step in its creation. Chatbot functionality is evaluated locally after being built using a chatbot programming language or an application framework. The chatbot is then made publicly available on a website or in a data center, and it is linked to one or more channels to send and receive messages.

[7] Nimavat & Champaneria, 2017, The services a chatbot should provide to its users and the category it belongs to will dictate the algorithms or platforms used to build them.

I. PROPOSED SYSTEM

The backend infrastructure for your application is provided by Rasa Framework. The chatbot's architecture is depicted in Figure 3. Our serverless platform-based chatbots are AI-powered. Use the rasa framework's (NLP) and (NLU) to decipher user inquiries pertaining to COVID-19 and provide pertinent answers. NLP supports and facilitates reading, decoding, interpreting, and understanding human language. The initial processing layer in our design is responsible for receiving input in the way of transcript or voice. A user's request is transformed from speech to text when they ask a question or leave a voicemail (speech-to-text). This extracted text is applied to the produced text at the second

level of processing, where it serves as the foundation for conducting natural language comprehension and applying the explication meaning of user input. Find phonemes. Asspeech-to-text conversion is not necessary for the chat interface, is regarded as initial positions of processing.

Implementation -

Install Rasa: First, install Rasa 2.0 by following the instructions provided in the official Rasa documentation.Create a new project: Create a new Rasa project using the command-line

interface (CLI). This command will create a new Rasa project with some default configuration files and directories.

Intents and entities: Intents are the user's intentions, and entities are the parameters that your chatbot needs to understand to fulfill the user's request. In this case, you will need to define an intent to fetch state-wise Covid data and an entity to specify the state. Open the domain.yml file, and add the following intents and entities:

Create stories: Stories are example conversations between the user and the chatbot. Create some stories in the data/stories.yml file that correspond to the intents and entities you defined earlier.

FormAction: A FormAction is used to ask for and validate user input. Define a FormAction called CovidDataForm in the actions.py file, which will prompt the user to enter the state for which they want Covid data. In this action, you can also validate the user input to ensure that the entered state is valid.

Actions: Define an action called ActionFetchCovidData in the actions.py file that fetches and processes the Covid data for the specified state. In this action, you can use an API to fetch the Covid data for the specified state.

Utterances: Define the bot's responses to the user's inputs in the domain.yml file.

File generation logic: Define a function in the actions.py file that generates the requested filetype (PDF or Word) containing the state-wise Covid data. You can use Python libraries like python-docx and reportlab to generate Word and PDF files, respectively.

Utter_download_confirmation response: Definean utterance that confirms to the user that the file generation was successful, and asks the user if they want to download the file.

Action_download_file action: Define an action that sends the generated file to the user. You can use the dispatcher.utter_attachment() method to

send the file to the user. Here's an example implementation.

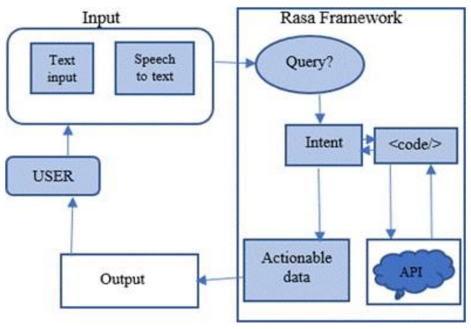


Figure 1. Proposed System Architecture

1. Stage of Natural Language Understanding: The user's input is processed in this stage. At this stage, the chatbot will primarily strive to comprehend what the user is attempting toinquire. An intent from a trained set of intents is matched with the input that is provided. To achieve an accurate response from the chatbot,

input must be mapped to the appropriate intent. Because different users may have diverse types of queries, a range of examples must be trained for intent categorization. User input must be matched with the intended objective through adequate training. After the intent has been mapped,entities need to be extracted from the input.

2. Stage of Answer Selection: RASA Core is in charge of providing answers to the users' questions. For each intent, a series of pre-defined action sequences will be used. The Core will choose the action to be taken for each user query. Actions can be static or dynamic. Static actions that start with an absolute keyword cause only certain static messages to be read. Illustration of a possible static action is Find courses in CS, please. Look for CS courses, please. Give me a list of computer courses, please. Please provide me the list of CS courses. In each of these situations, the chatbot must understand that the entity is "CS" and the intent is "search for courses". What users want is expressed in their intentions. A piece of information derived from a query is called an entity. Only a bot that can accurately extract this set of intent entities will be able to produce the desired results. The model's training and subsequent mapping of intent entities will be handled by the RASA NLU.

3. Stage of output: The user must receive the response that the model has chosen. The ultimate result is provided by the output stage, also known as the Natural Language Generation step. The

output can take many different shapes. It could be expressed verbally or through written words. Another text-to-voice conversion mechanism must be used if the output is speech. Lastly, a variety of channels can be used to obtain the outputs.

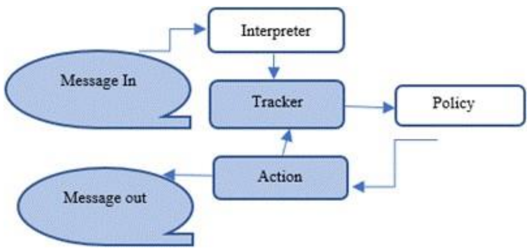


Figure 2. System Architecture of the model

The RASA NLU (an interpreter) receives this message from the end user and produces a structured output that includes plaintext as well as the intent and entities, if any, depicted in Figure. This monitor consistently keeps the discussion in a structured state and takes input from the interpreter. The policy receives the tracker output and acts on the tracker's current state as a result.

This policy establishes the proper course of action. The tracker keeps a record of the selected action. The user is given the right response, whichemploys the same nlu.md-defined intent as the complete response.

ALGORITHM :

For NLU, Rasa 2.0 uses a supervised machine learning algorithm called "DIET" (Dual Intent and

Entity Transformer). DIET is a deep neural network that combines intent classification and entity recognition into a single model, making it more efficient than previous Rasa NLU algorithms. DIET uses pre-trained word embeddings, character-level embeddings, and intent-specific embeddings to improve its performance on low-shot intent and entity recognition tasks.

For Dialogue Management, Rasa 2.0 uses a rule-based approach for handling simple dialogues and a machine learning-based approach for handling more complex conversations. The machine learning-based approach is based on the "TED" (Transformer Embedding Dialogue) algorithm, which is a deep neural network that uses self-attention mechanisms to learn contextual representations of the dialogue history. TED can handle multi-turn conversations and can learn to

predict the next best action to take based on the current state of the conversation.

The following steps are the pseudocode of the implementation.

Step 1: Importing the packages typing,rasa_sdk and requestors.

Step 2: Define the class ActionHelloWorld with Action as the parameter.

Step 3: Define the class ActionRequiredCooanStateCity with action as parameter.

Step 4: Get the data and store the path in the variable response.

Step 5: Printing the number of effected states.storing the entitites using latest_message() function.

Step 6: Run a for loop for every entity and check whether the entity is the effected state.

Step 7: If true printing the state.

Step 8: Checking statewise and if true printing the number of

Step 9: Active and Confirmed cases.

IV. RESULTS & DISCUSSION

It summarizes the features of medical related chatbots, how to utilise them, and the drawbacks of applying these cutting-edge technology in the event of a pandemic. The usage of health chatbots to combat COVID19 is still in its early stages. The researcher will be able to better grasp these innovative technological designs and applications with the aid of our findings as he continues to develop medical related chatbot operable and applications in this battle upon COVID-19.

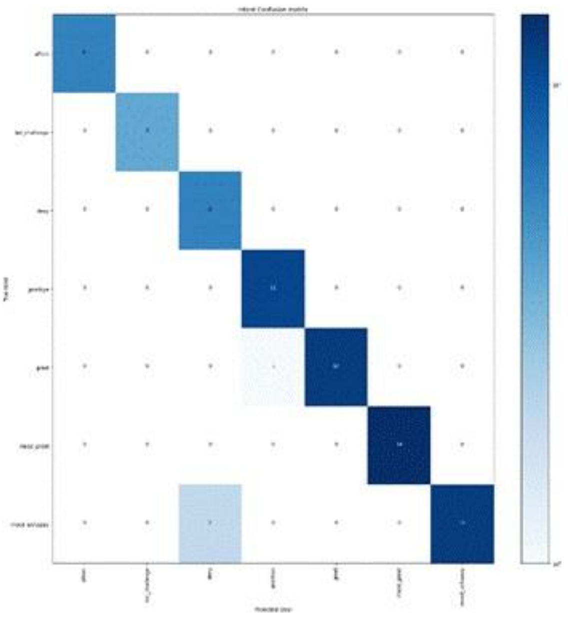


Figure 3.1 shows the interpretations of the output

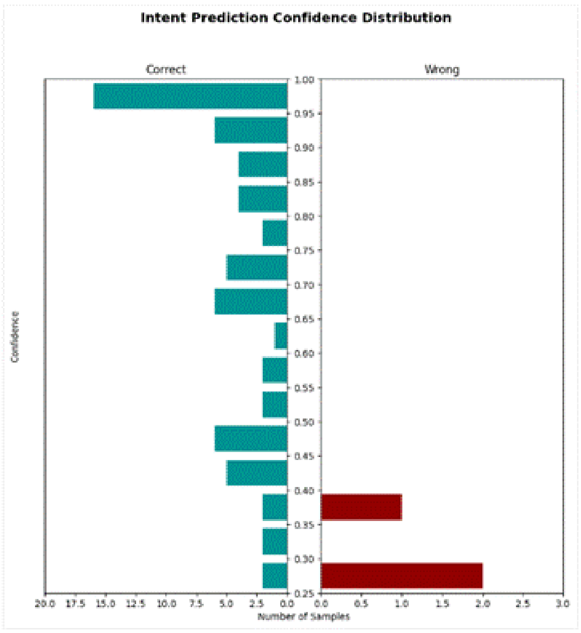
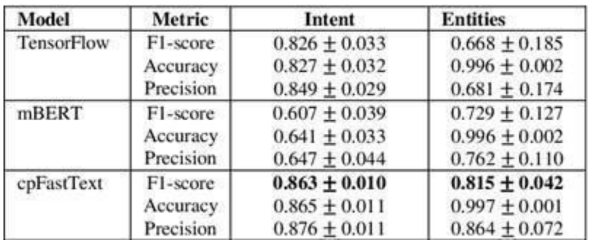


Figure 3.2 shows the number of samples tested and the responded outputs.

Input: "Hi, how are you?"

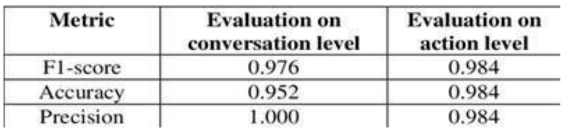
Output : " I am a bot powered by Rasa ".
Input: "Kerala"
Output: "It shows the accurate value of the cases in Kerala"

Figure 3.3 shows the output shows about the working demo model of Rasa incommand prompt. The input is been given to the command prompt and the output is displayed by the Rasa chatbot.



Model	Metric	Intent	Entities
TensorFlow	F1-score	0.826 ± 0.033	0.668 ± 0.185
	Accuracy	0.827 ± 0.032	0.996 ± 0.002
	Precision	0.849 ± 0.029	0.681 ± 0.174
mBERT	F1-score	0.607 ± 0.039	0.729 ± 0.127
	Accuracy	0.641 ± 0.033	0.996 ± 0.002
	Precision	0.647 ± 0.044	0.762 ± 0.110
cpFastText	F1-score	**0.863 ± 0.010**	**0.815 ± 0.042**
	Accuracy	0.865 ± 0.011	0.997 ± 0.001
	Precision	0.876 ± 0.011	0.864 ± 0.072

Figure 3.4 shows the accuracy measurement of intent and entities using TensorFlow, mBERT, cpFastText.



Metric	Evaluation on conversation level	Evaluation on action level
F1-score	0.976	0.984
Accuracy	0.952	0.984
Precision	1.000	0.984

Figure 3.5 describes about the accurate metrics of the implementation.

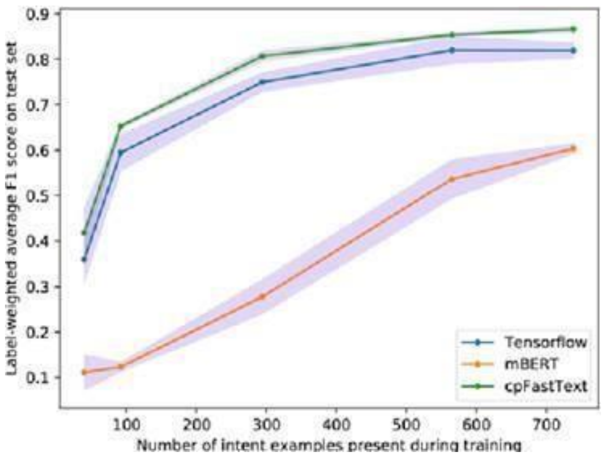


Figure 3.6 shows the graphical comparison of nlu pipelines.
And also the accuracy level of intent and entities of the following evaluation.

V. CONCLUSION & FUTURE WORK

The features of health chatbots, how to utilise them, and the drawbacks of adopting these cutting-edge technology during the epidemic are all covered in this section. In the struggle against COVID19, the usage of health chatbotsis still in its infancy. Our findings, which are crucial for the on-going development of medical chatbot functionality and applications in the medical profession, we hope will aid

researchers in better comprehending the uses and conception of these novel technologies. the conflict with COVID-19.

REFERENCES

1. E. Kasthuri and S. Balaji, "A Chatbot for Changing Lifestyle in Education," Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks
(ICICV), 2021, pp. 1317- 1322. [CrossRef]

2. Smutny, P., & Schreiberova, P. “Chatbots for learning: A review of educational chatbots for the Facebook Messenger,” Computers & Education, 2020. [CrossRef]

3. M. Rakhra et al., "E-Commerce Assistance with a Smart Chatbot using Artificial Intelligence," 2nd International Conference on Intelligent Engineering and Management (ICIEM), 2021, pp. 144-148. [CrossRef]

4. N. Shi, Q. Zeng and R. Lee, "Language Chatbot–The Design and Implementation of English Language Transfer Learning Agent Apps," IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), 2020, pp. 403-407. [CrossRef]

5. N. N. Khin and K. M. Soe, "Question Answering based University Chatbot using Sequence to Sequence Model," 23rd Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA), 2020, pp. 55-59. [CrossRef]

6. S. García-Méndez, F. De Arriba-Pérez, F. J. GonzálezCastaño, J. A. Regueiro-Janeiro and F. Gil-Castiñeira, "Entertainment Chatbot for the Digital Inclusion of Elderly People Without Abstraction Capabilities," in IEEE Access, vol. 9, 2021, pp. 75878-75891. [CrossRef]

7. M. N. Kumar, P. C. L. Chandar, A. V. Prasad and K. Sumangali, "Android based educational Chatbot for visually impaired people," IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2016, pp. 1-4. [CrossRef]

8. G. Daniel and J. Cabot, "The Software Challenges of Building Smart Chatbots," IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSECompanion), 2021, pp. 324-325.

9. E. H. Wu, C. Lin, Y. Ou, C. Liu, W. Wang and C. Chao,
"Advantages and Constraints of a Hybrid Model K-12 Learning Assistant Chatbot," in IEEE Access, vol. 8,2020, pp.77788-77801.

10. L. E. Chen, S. Y. Cheng and J. -S. Heh, "Chatbot: A Question Answering System for Student," International
Conference on Advanced LearningTechnologies (ICALT) 2021, pp. 345-346.