# Innomatics Research Labs

## Hyderabad

# Internship Project Report

## Oct 2021- Jan 2022

Project Title:

## COVID-19 ChatBot using RASA


**Submitted By:**

Niji Narayanankutty

K. Shravya

Shubham Luharuka

Rishabh Kumar

Yug Rawal

**Under the Guidance of:**

Kanav Bansal

(Chief Data Scientist)

Innomatics Research Labs

# Abstract

Chatbots are software programs that interact with people over voice or text. Users seek quick, accurate responses when searching for information or assistance, and this leads to the development of chatbots. These Chabot have played almost important roles in health care field before covid 19. So in this pandemic situation to provide the people with the necessary information we have developed a covid 19 Chabot using the Rasa framework. Rasa is a python framework that helps us to build any kind of Chatbot easily. It based on NLU (Natural Language Processing) which offer the possibility to understand what the user wants. In this report we are explaining our approach and algorithm of our Covid-19 Chatbot.

**Table of Content**

**Topic**


1. **Introduction**
    a. **Purpose**
    b. **Goal**
    c. **Technology Stack**
    d. **Working of Chatbot**
2. **Introduction to RASA**
    a. **RASA Framework**
    b. **RASA Architecture**
    c. **Initialization of RASA**
    d. **Working Phases**
3. **Developing Chatbot without RASA**
4. **Conclusion and Future Scope**

# INTRODUCTION

## [1. a] PURPOSE

Coronavirus disease 2019 (COVID-19) is a contagious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The first known case was identified in Wuhan, China, in December 2019. The disease has since spread worldwide, leading to an ongoing pandemic. The only way to control the pandemic situation is to provide necessary information and awareness about the current situation and about the disease. Artificially intelligent (AI) based conversational agents, otherwise known as Chabot, are the latest inventions utilized to combat the novel SARS-CoV-2 coronavirus (COVID-19). Chabot enable users to communicate and interact with software applications that use AI-based tools. People can utilize the Chabot to find answers to their questions on protecting themselves from the Coronavirus, to understand the facts and news related to the disease, and to contribute to preventing its spread.

In this project we have developed a Chabot using Rasa framework. The Chatbot is a conversational agent which talk with a client or user and provide answers to the users' requests independently of the human intervention. Rasa is a python framework that helps us to build any kind of Chatbot easily. It based on NLU (Natural Language Processing) which offer the possibility to understand what the user wants.

## [1. b] Goal

Goals of the the projects that we made are :-

- To provide correct information related to Covid 19, by reducing the myths and rumors about it.
- To provide the best facilities and needs to all the people.
- To avoid and panic situation and myths regarding the covid 19
- To reduce the spread of this virus by accepting and following Social distancing norms.

## [1.c] Technology Stack

- Model Developement (Rasa Framework, Python)
- Backend (Flask, )
- Frontend (HTML, CSS)
- Dataset: IBM Developer Covid-19 Questions
- Other Framework (Tensorflow)

## [1. d] Working of Chatbot

The Chatbot is a conversational agent which talk with a client or user and provide answers to the users' requests independently of the human intervention. In this project we have created Chabot using rasa framework.

The Chatbot is composed by some specific terms: Intents: It represent intentions that the user want to express when he submit his message to the Chatbot. Inside the intents, we provide some phrases that the user may ask, and we also provide some responses that the Chatbot must use to answer to the user. Entities: Entities are keywords that represent some specifics data that the Chatbot may use to perform the discussion with the user. Entities are used to extract some values inside the user input (message).

# INTRODUCTION TO RASA

## [2. a] Rasa Framework

Rasa is a tool to build custom AI chatbots using Python and natural language understanding (NLU). Rasa provides a framework for developing AI chatbots that uses natural language understanding (NLU). It also allows the user to train the model and add custom actions.


## Training of Chabot

Training a Chabot occurs at a considerably faster and larger scale than human education. Chabot is nourished with a large number of conversation logs, and from those logs, the Chabot can understand what type of question needs, what kind of answers.

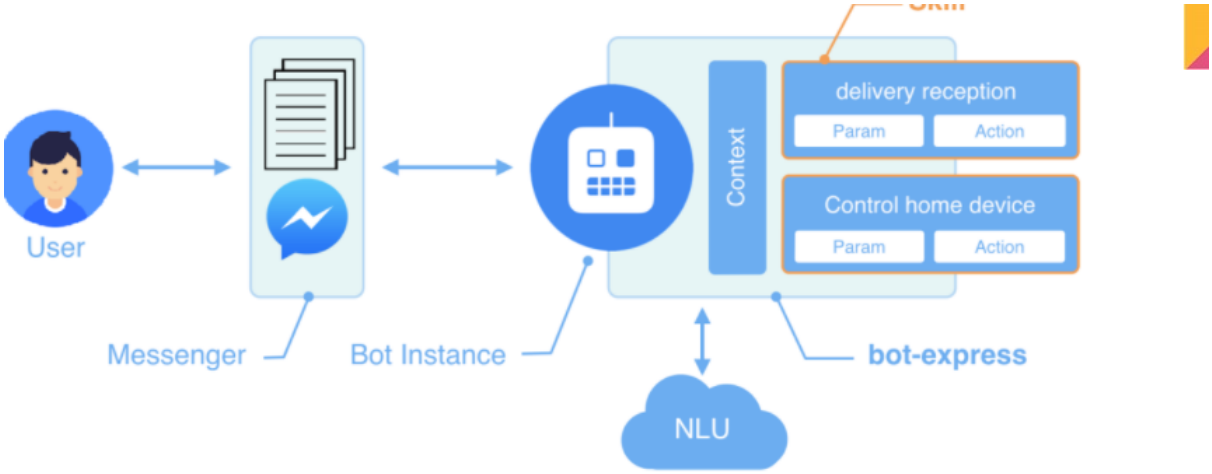
### Architecture & Work Methods of Chabot

The Chabot work based on three classification methods:

**Pattern Matches**:

Bots utilize pattern matches to group the text and it produces an appropriate response from the clients. “Artificial Intelligence Markup Language (AIML), is a standard structured model of these Patterns. For every sort of question, a remarkable pattern must be accessible in the database to give a reasonable response. With a number of pattern combinations, it makes a hierarchical structure. We utilize algorithms to lessen the classifiers and produce the more reasonable structure.

**2. Natural Language Understanding (NLU)**



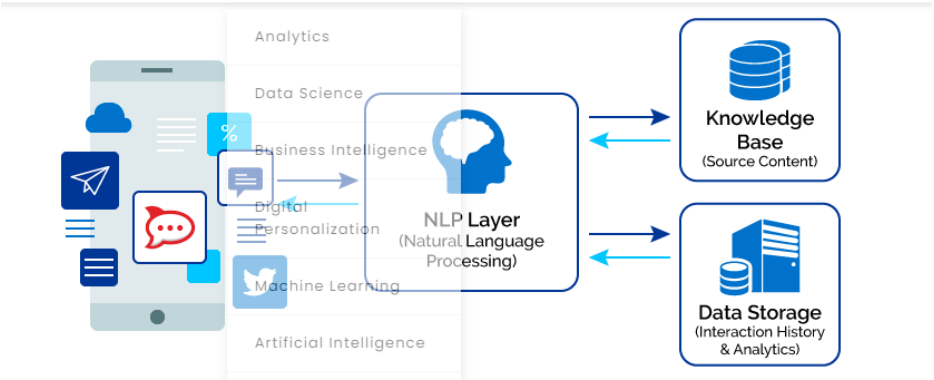

This NLU has 3 specific concepts as follows:

**Entities:** This essentially represents an idea to your chatbot. For example, it may be a payment system in your E-commerce chatbot.

**Context:** When a natural language understanding algorithm examines a sentence, it doesn’t have the historical backdrop of the user’s text conversation. This implies that, if it gets a response to a question it has been recently asked, it won’t recall the inquiry. So, the phases during the conversation of chat are separately stored.

**Expectations**: This is what a chatbot must fulfill when the customer says sends an inquiry. Which can be the same for different inquiries. For example, the goal triggered for, “I want to purchase a white pair of shoes”, and “Do you have white shoes? I want to purchase them” or “show me a white pair of shoes”, is the same: a list of shops selling white shoes. Hence, all user typing text show a single command which is the identifying tag; white shoes.

**3. Natural Language Processing (NLP)**




(NLP) Natural Language Processing Chatbots finds a way to convert the user's speech or text into structured data. Which is then utilized to choose a relevant answer. Natural Language Processing includes the following steps;

**Tokenization:** The NLP separates a series of words into tokens or pieces that are linguistically representative, with a different value in the application.

**Sentiment Analysis:** It will study and learn the user's experience, and transfer the inquiry to a human when necessary

**Normalization:** This program model processes the text to find out the typographical errors and common spelling mistakes that might alter the intended meaning of the user's request.

**Named Entity Recognition:** The program model of chatbot looks for different categories of words, similar to the name of the particular product, the user's address or name, whichever information is required.

**Dependency Parsing:** The Chatbot searches for the subjects, verbs, objects, common phrases and nouns in the user's text to discover related phrases that what users want to convey.

## [2. b] Architecture

The diagram below shows Rasa has two main components:

- **Rasa NLU (Natural Language Understanding):** Rasa NLU is an open-source natural language processing tool for intent classification (decides what the user is asking), extraction of the entity from the bot in the form of structured data and helps the Chabot understand what user is saying.
- **Rasa Core** (Dialogue management): a Chabot framework with machine learning-based dialogue management which takes the structured input from the NLU and predicts the next best action using a probabilistic model like LSTM neural network rather than if/else statement. Underneath the hood, it also uses reinforcement learning to improve the prediction of the next best action.

In other words, the Rasa NLU's job is to interpret the input provided by the user in the form of structured data and Rasa Core's job is to decide the next set of actions performed by the Chabot. Rasa Core and Rasa NLU are independent of each other and can be used separately.

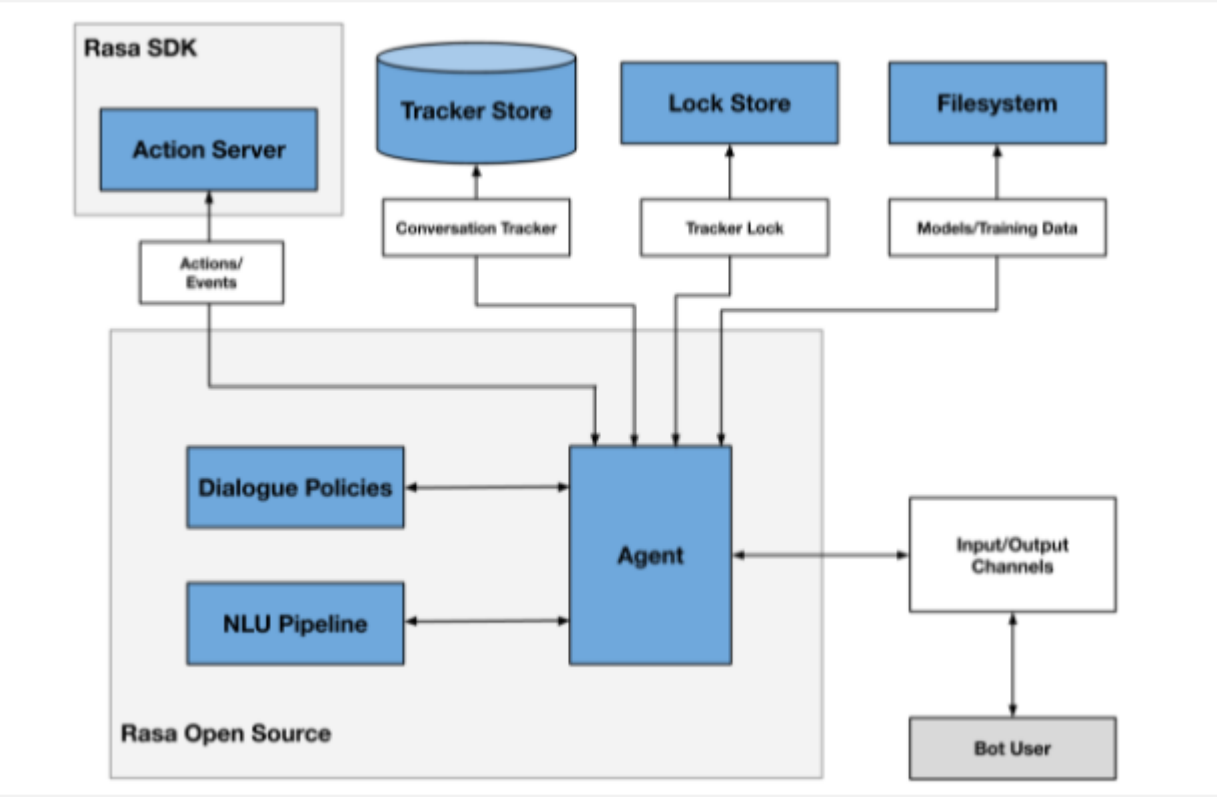
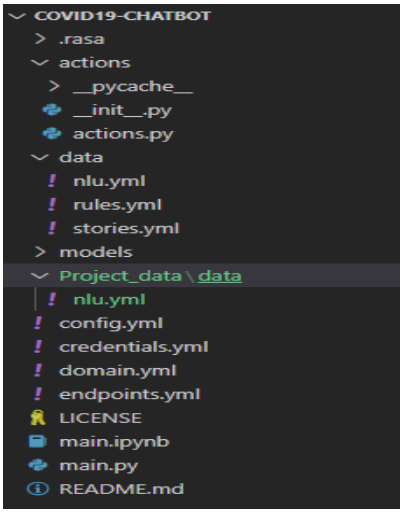



Fig: Image is showing the working flow of RASA Framework.

## [2. c] Initialization of RASA

After installing RASA using "pip install rasa" in cmd. You have to make run the "rasa init" in the cmd. It will download the file structure for storing our data in the the and code for model development and configuring our files.

Below fig is displaying the file structure in current working directory.



Description of files that rasa contained after initialization.

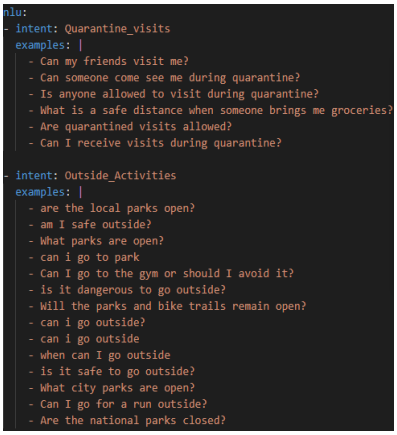
**Action folder:**

**__init__.py**: An empty file that helps python to locate your actions.

**actions.py**: This file is used for creating custom actions. In case you want to call an external server or fetch external API data, you can define your actions here.

**Data Folder:** This describes the different types of training data that go into a Rasa assistant and how this training data is structured. Rasa Open Source uses YAML as a unified and extendable way to manage all training data, including NLU data, stories, and rules.

**nlu.md:** In this file, we define our intents. These intents are then used in training the NLU model.

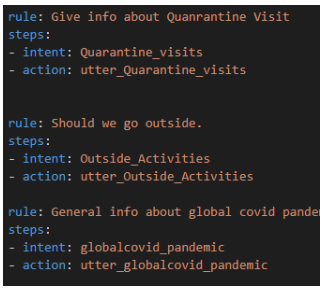


```
nlu:
- intent: Quarantine_visits
  examples: |
    - Can my friends visit me?
    - Can someone come see me during quarantine?
    - Is anyone allowed to visit during quarantine?
    - What is a safe distance when someone brings me groceries?
    - Are quarantined visits allowed?
    - Can I receive visits during quarantine?

- intent: Outside_Activities
  examples: |
    - are the local parks open?
    - am I safe outside?
    - What parks are open?
    - can i go to park
    - Can I go to the gym or should I avoid it?
    - is it dangerous to go outside?
    - Will the parks and bike trails remain open?
    - can i go outside?
    - can i go outside
    - when can I go outside
    - is it safe to go outside?
    - What city parks are open?
    - Can I go for a run outside?
    - Are the national parks closed?
```

**stories.md:** Stories are the sample conversation between a user and bot in the form of intents, responses, and actions. Rasa stories are a form of training data used to train the Rasa's dialogue management models.

**rules.md:** Rules are a type of training data used to train your assistant's dialogue management model. Rules describe short pieces of conversations that should always follow the same path. rules don't have the power to generalize to unseen conversation paths. Combine rules and stories to make your assistant robust and able to handle real user behavior.



```
rule: Give info about Quanrantine Visit
steps:
- intent: Quarantine_visits
- action: utter_Quarantine_visits


rule: Should we go outside.
steps:
- intent: Outside_Activities
- action: utter_Outside_Activities

rule: General info about global covid pande
steps:
- intent: globalcovid_pandemic
- action: utter_globalcovid_pandemic
```

**Models:**

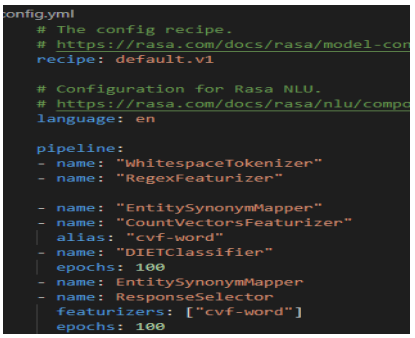
**models/<timestamps>.tar.gz:** the initial model, all the trained models stored in the models folder. For retraining the model, we use rasa train command.

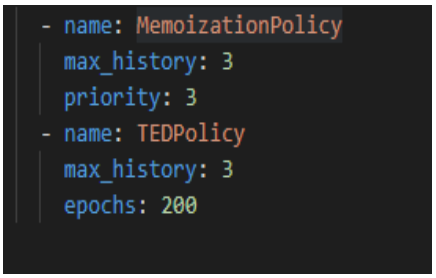
**Project/data file:**

**config.yml**: This file defines the configuration of the NLU and core model. If you are using any model outside the NLU model, you must define the pipeline here. It defines the components and policies that your model will use to make predictions based on user input.

The recipe key allows for different types of config and model architecture. Currently, only "default.v1" is supported.

The language and pipeline keys specify the components used by the model to make NLU predictions. The policies key defines the policies used by the model to predict the next action.




**credentials.yml:** This file is used to store credentials for connecting to external services such as Facebook Messenger, Slack, etc.
**domain.yml:** It specifies the intents, entities, slots, responses, forms, and actions your bot should know about. It also defines a configuration for conversation sessions.

**Intent:** The intents key in your domain file lists all intents used in your NLU data and conversation training data. **Entities:** The entities section lists all entities that can be extracted by any entity extractor in your NLU pipeline.

**endpoints.yml:** This defines the details for connecting channels like Slack, FB messenger, etc. for storing chats data in the online databases like Redis, etc.

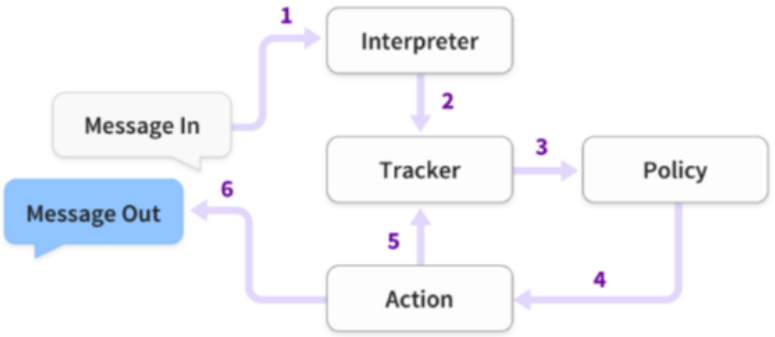
## [2.d] Working Phases

- The User submit the **request** (a message is to the **Interpreter**)
- In the **Interpreter** section, we have **Rasa NLU** which capture the request, and extract **intents** to understand what the user want, and at the same time it extracts some

specifics keywords (**entities**) which represent the important data. Then it sends the result to the **Tracker**. The rest of the elements below is contained inside **Rasa Core**.

- The **Tracker** here, is used to store the conversation history in memory. By this fact, the Chatbot should know in where level the conversation is, it maintains the conversation state.
- After that, the **Policy** comes; it chooses the action which will be execute at every step in the dialogue, regarding the history of the conversation.
- The **Action** here, look inside the history for consulting the state of the conversation. This step can trigger another action.
- The action is executed, and an **output** is generated (message from the Chatbot) to the user.

Working overflow of the Rasa Framework.



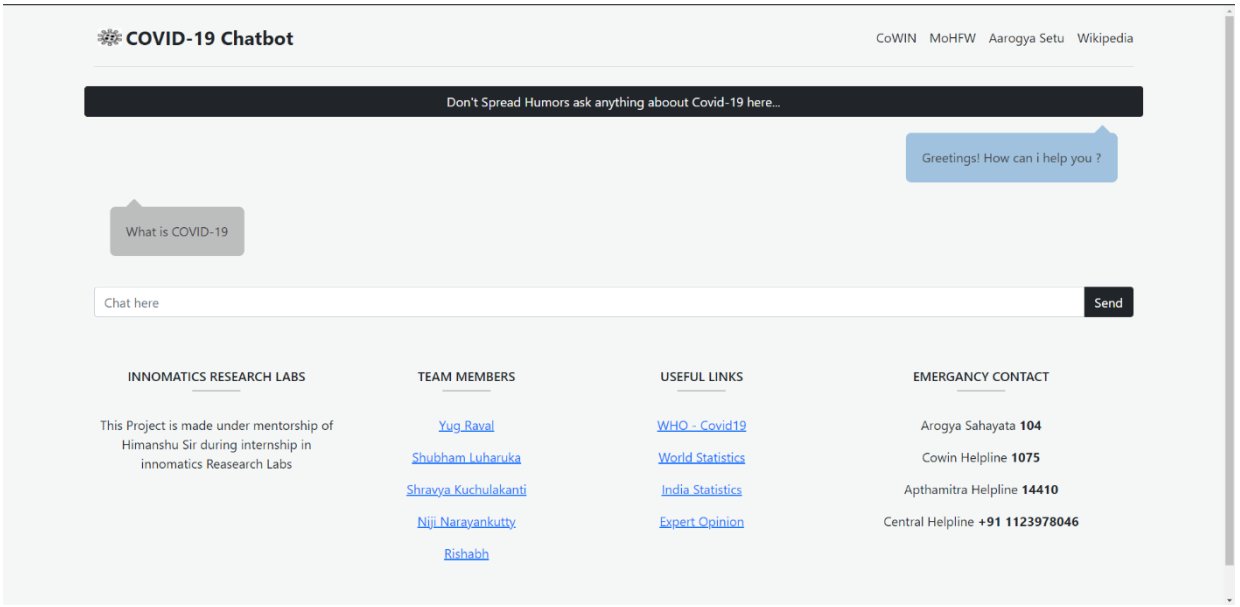

Functioning of Rasa

# Frontend

We Have made our frontend using Bootstrap version v5.1 I which we have used inbuilt CSS and JS for our project we have also taken care that our application should be lightweight and easy to load on any devices whether device is small screen like mobile phones, or it is large screen devices like Television or Projector. We have made sure that our application is responsive on any screen size and on any bandwidth of internet.

Here are some snapshots on different screen size,

1. Laptop Screen ( ROG StrixG731 – 1920 X 937 )




# Working of Chabot without Rasa

To build a Chabot in Python, you have to import all the necessary packages and initialize the variables you want to use in your Chabot project**.**

1. **Prepare the Dependencies**

The first step in creating a Chabot in Python with the Chabot library is to install the library in your system. It is best if you create and use a new Python virtual environment for the installation. To do so, you have to write and execute this command in your Python terminal:

2. **Import Classes**

Importing classes is the second step in the Python chatbot creation process. All you need to do is import two classes – ChatBot from chatterbot and ListTrainer from chatterbot.trainers. To do this, you can execute the following command:

3. **Create and Train the Chatbot**

This is the third step on creating chatbot in python. The chatbot you are creating will be an instance of the class “ChatBot.” After creating a new ChatterBot instance, you can train the bot to improve its performance. Training ensures that the bot has enough knowledge to get started with specific responses to specific inputs. You have to execute the following command now:

Here, the argument (that corresponds to the parameter name) represents the name of your Python Chabot. If you wish to disable the bot's ability to learn after the training, you can include the "read_only=True" command. The command "logic_adapters" denotes the list of adapters used to train the Chabot.
While the "chatterbot.logic.MathematicalEvaluation" helps the bot to solve math problems, the "chatterbot.logic.BestMatch" helps it to choose the best match from the list of responses already provided.

**4. Communicate with the Python Chatbot**
To interact with your Python chatbot, you can use the .get_response() function.

**5. Train your Python Chatbot with a Corpus of Data**
For training python chatbot even further, you can use an existing corpus of data. Here's an example of how to train your Python chatbot with a corpus of data provided by the bot itself:

# CONCLUSION AND FUTURE SCOPE:

We can add two more features to make this application more interactive:

1. **Doctor and vaccine booking**: The user can book a doctor appointment and also they register for the vaccine.
2. **Voice over text**: In current version user can interact with bot only through text we can add voice messages in newer version.