

**Project CovoBot**

A real-life NLP chatbot developed to answer queries regarding

The COVID-19 pandemic.

**DOCUMENTATION**



**TABLE OF CONTENTS:**

Background and Motivation……………………………………………………………….....3

Problem Statement…………………………………………………………………………...4

CovoBot Features…………………………………………………………………………….4

Technology Stack…………………………………………………………………………….4

Process Flow…………………………………………………………………………………5

[NLU………………………………………………………………………………………….](#_Toc76306103)5

NLU working: ………………………………………….…………………….…………...5

Tokenizers: ………………………………………………………………………………..7

Featurizers and its Types…………………………………………………………………..8

Training Data and its Purpose……………………………………………………………..8

Intent Classification and Entity Extraction……………………………………………….10

Dialog Management System…………………………………………………………….…...11

Stories, Rules, Domains…………………………………………………………….……12

Slots, Actions……………………………………………………………………….……14

Responses, Forms………………………………………………………………….…….15

APIs…………………………………………………………………………………….……18

[Public COVID API………………………………………………………………….…….](#_Toc76306106)19

[Unstructured Data API…………………………………………………………………....](#_Toc76306107)19

Detailed Architecture Design………………………………………………………………. 20

Deployment………………………………………………………………………………….21

System Validation…………………………………………………………………………...22

Conclusion…………………………………………………………………………………..23

Future Scope………………………………………………………………………………...23

References…………………………………………………………………………………..23

**BACKGROUND AND MOTIVATION:**

Artificial intelligence (AI) has already begun to permeate our daily lives, altering the way we think and work. Artificial intelligence (AI) chatbots like Alexa, Siri, and others are already commonplace in our daily lives for tasks like dialling a number, playing music, or getting directions. Conversational AI, also referred to as Chatbots, is a new technology that allows information to be accessed via text or voice-based interaction and is proving its worth during the coronavirus crisis and showing how this new communication channel can be leveraged in the years to come by a variety of groups and institutions.

Chatbots are only going to get bigger, because we can now get whatever bit of information, we need from them. Receiving information on a drug or sickness is a common request in the Healthcare sector, as is getting flight information for the Aviation industry, or the latest stock prices for the Finance industry.

Many corporations and organisations are leading the way in implementing chatbots to deliver COVID-19 information. Worldwide health organisations like WHO and Centre for Disease Control and Prevention, CDC have integrated chatbots into their websites to keep millions of people informed on the disease's progress and symptoms as they emerge. Anyone may use the chatbots from a wide range of devices if they have access to them (online computer, smart phone, or analogue phone in some cases). The use of chatbots for COVID-19 information can minimise the amount of time that hospitals spend on the phone. Chatbots' interactive symptom-checking function might help cut down on emergency room visits and the number of patients who need to be seen there.

CovoBot, is a real life chatbot which is developed to help consumers to be aware of every single information about this pandemic. A user should be able to communicate with the bot anytime and should be able to get the latest status on COVID-19. It is a Basic Conversational integrated with Covid-19 Trackers API’S to get real time details of cases across India, and eventually a simple communication channel that a user can use for conversations.

**PROBLEM STATEMENT:**

CovoBot, is a real life chatbot which is developed to help consumers to be aware of every single information about this pandemic. A user should be able to communicate with the bot anytime and should be able to get the latest status on COVID-19. It is a Basic Conversational integrated with Covid-19 Trackers API’S to get real time details of cases across India, and eventually a simple communication channel that a user can use for conversations.

**FEATURES:**

A chatbot built using RASA, to answer queries related to COVID-19, its symptoms, precautions, and queries related to the vaccines, its complications, side effects, doses, dates, etc. This product will act as a bot to communicate with users using a chat interface such as Slack or Telegram and will be able to understand user intention, manage conversation flow, and be able to retrieve answers using some external knowledge base of API. Concepts of Natural Language Processing and Python to develop the chatbot, where we develop intents, entities, stories, actions (these are the requirements for the working of RASA).

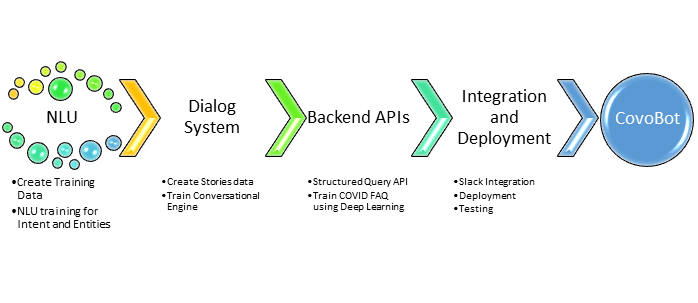
Various natural language processing algorithms were used to process datasets. The chatbot answers questions, with the RASA framework and uses DIET Classifier for better intent classification.

**TECHNOLOGY STACK:**

|  |  |
| --- | --- |
| Python 3.7 | An interpreted high-level general-purpose programming language |
| RASA | It is a tool to build custom AI chatbots using Python and natural language understanding (NLU). |
| SpaCy | It is used for pre-processing of the utterances, tokenization, and featurization. |
| DIET classifiers | Dual Intent Entity Transformer (DIET) used for intent classification and entity extraction |
| TensorFlow | open-source software library for machine learning and artificial intelligence |
| NLTK | The Natural Language Toolkit (NLTK) a standard python library with prebuilt functions and utilities for the ease of use and implementation |
| Telegram | It is a free and open source, cross-platform, cloud-based instant messaging software. |
| Requests/urllib | The urllib. request module defines functions and classes which help in opening URLs (mostly HTTP). |

**PROCESS FLOW:**

Prerequisites - Visual Studio, Python and its libraries, RASA, SpaCy, NLP libraries.



**NLU:**

NLU stands for Natural Language Understanding. As the name hints, it is responsible for understanding the input messages of the user. It is used to recognize the user’s wants and also extract information such as names, locations, etc. from the messages (user input).

Natural language understanding is an artificial intelligence technology, its main job is understanding spoken or written words and phrases.

It turns language, known technically as unstructured data, into a machine-readable format, known as structured data. This enables other computer systems to process the data to fulfil user requests. Most of the time, NLU is found in chatbots, voice bots, and voice assistants, but it can theoretically be used in any application that aims to understand the meaning of the typed text.

**Working of NLU:** NLU systems work by analysing input text, and using that to determine the meaning behind the user’s request. It does that by matching what’s said to training data that corresponds to an ‘intent’. Identifying that intent is the first job of an NLU.

The NLU pipeline is defined in the `config.yml` file in Rasa. This file describes all the steps in the pipeline that will be used by Rasa to detect intents and entities. It starts with text as input and it keeps parsing until it has entities and intents as output.

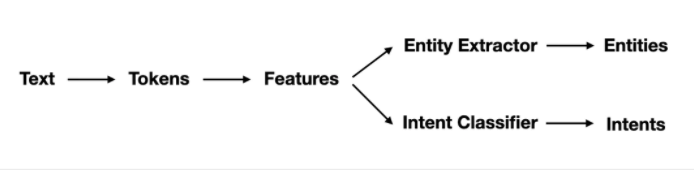
There are different types of components that you can expect to find in a pipeline. The main ones are:

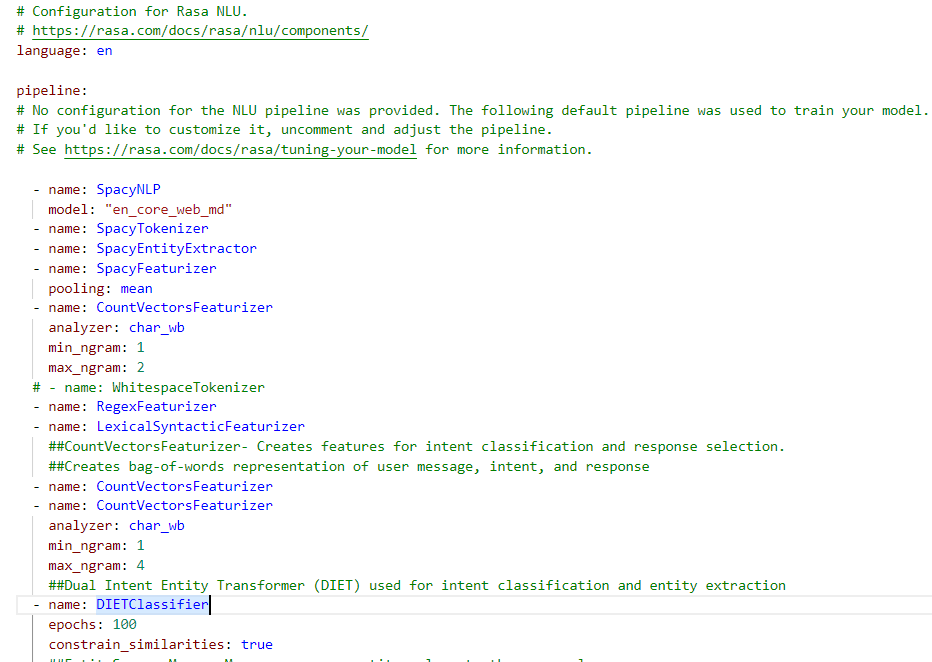
1. Tokenizers

2. Featurizers

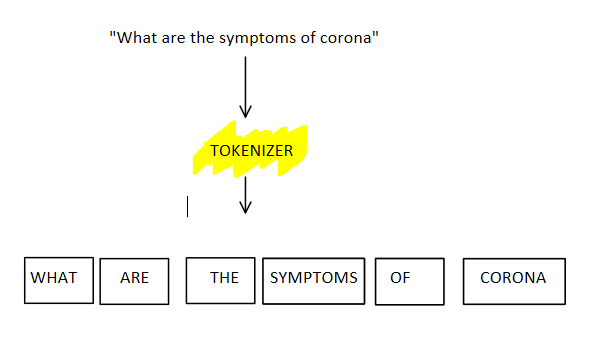
3. Intent Classifiers

4. Entity Extractors



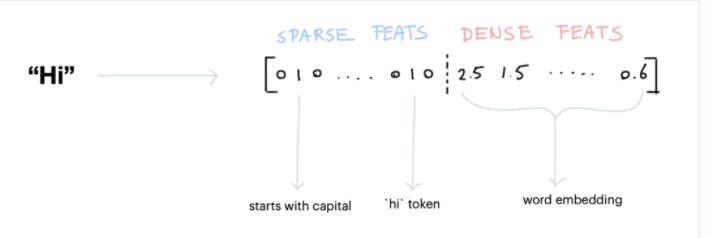


**Tokenizer:** Tokenization consists in getting the list of words (tokens) from an utterance. For example, the sentence “What are the symptoms of corona” would be tokenized as:



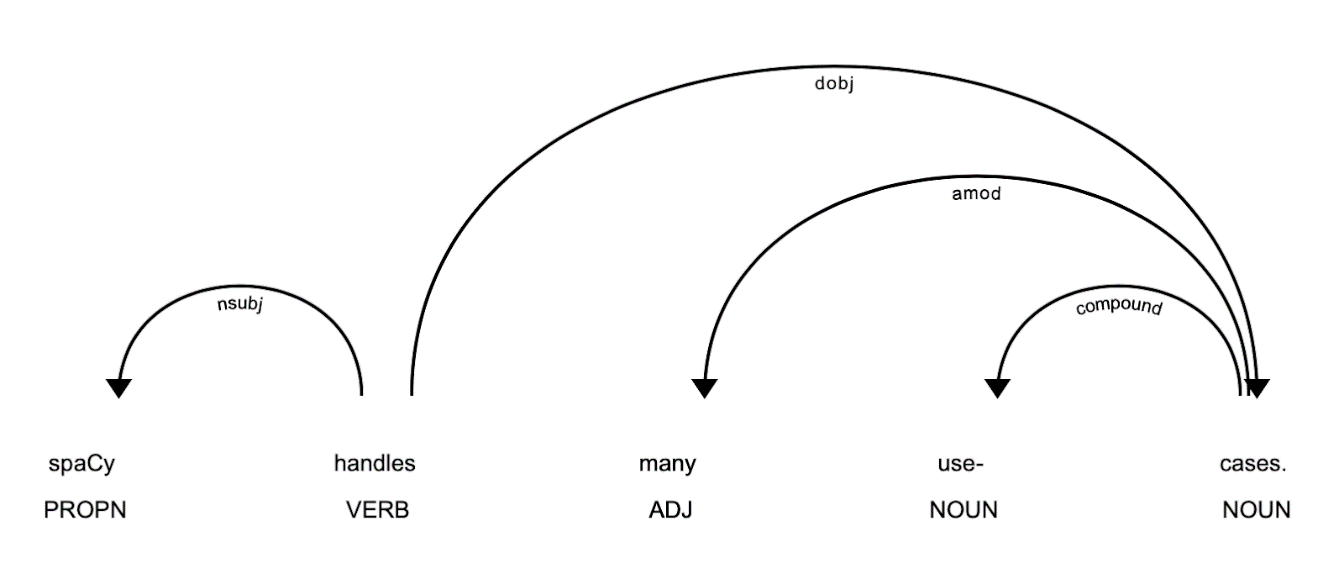
Many features are derived from words rather than sentences. Tokens resulting from this step will be used downstream the pipeline for features extraction.

**Featurizer:** Featurizers generate numeric features for machine learning models. The diagram below shows how the word "Hi" might be encoded.



There are two types of features:

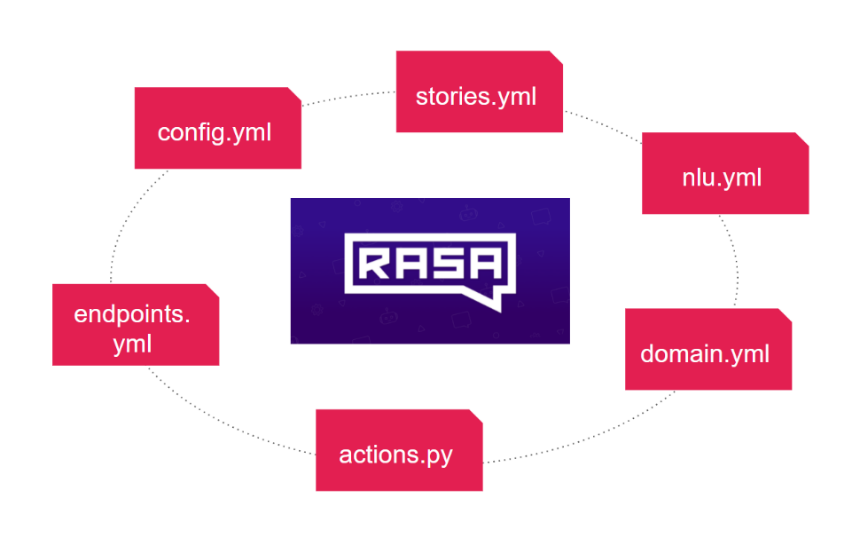
·       **Sparse Features:** usually generated by a CountVectorizer. Note that these counts may represent subwords as well. We also have a LexicalSyntacticFeaturizer that generates window-based features useful for entity recognition. When combined with SpaCy, the LecticalSyntacticFeaturizer can be configured to also include part of speech features.



*Lexical featurizer*

·     **Dense Features:** these consist of many pre-trained embeddings. Commonly from SpaCyFeaturizers or from huggingface via LanguageModelFeaturizers. If you want these to work, you should also include an appropriate tokenizer in your pipeline. More details are in the documentation.

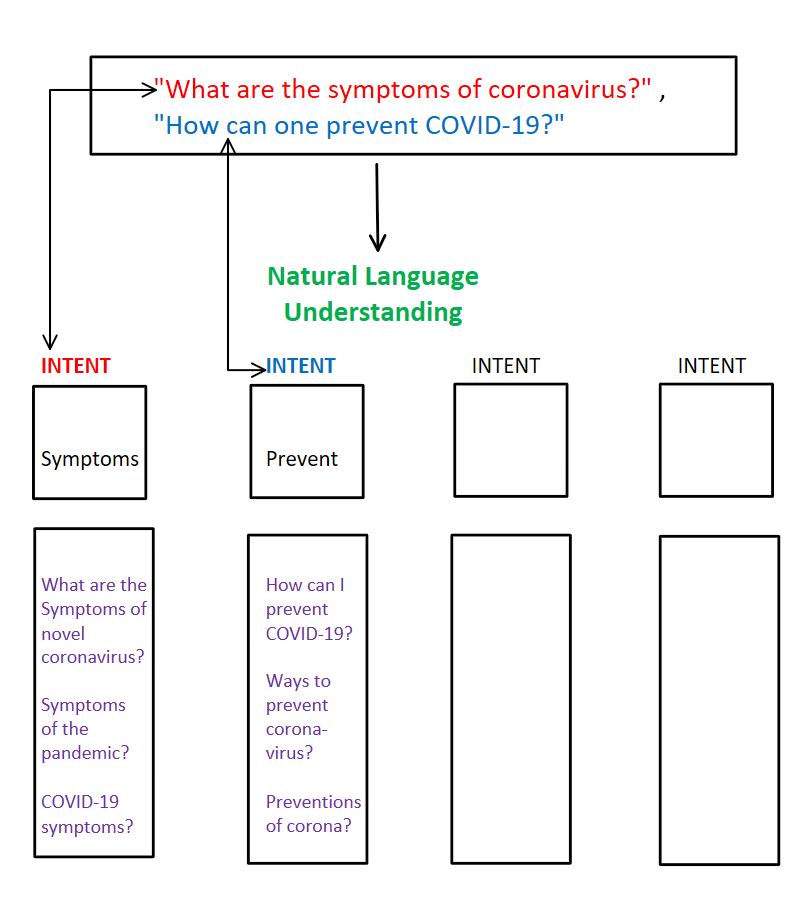
**Create Training Data:**

****

Training data, also called ‘sample utterances’ are simply written examples of the kind of things people are likely to say to a chatbot or voicebot.

The training data is organized into ‘buckets’, with each bucket containing different examples of phrases that have the same or similar meaning. For example: “What are the symptoms of novel coronavirus?” and “symptoms of coronavirus?” could both be examples of training data that you’d put into one ‘bucket’. That’s because both of those phrases mean the same thing: the user wants to know the symptoms of coronavirus.

Whereas; “how can I prevent covid19?” and “how can one prevent oneself from getting infected from covid19?” would both be examples of training data that you’d put into a different ‘bucket’. That’s because both of those phrases mean the user is wanting to know the prevention of coronavirus.



Each ‘bucket’ is given a specific label and that label is, you guessed it, the ‘intent’.

NLU training data stores structured information about user messages (input)**.** NLU training data consists of example user utterances categorized by intent. Training examples can also include entities.

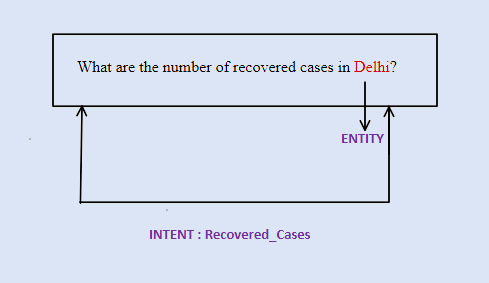
**data/nlu.md**: In this file, we define our intentions. These intents are then used in training the NLU model.

**Purpose of Training Data**: The purpose of providing training data to NLU systems isn’t to give it explicit instructions about the exact phrases you want it to listen out for. It’s to give it samples of the kind of things you want it to listen out for.

Once you’ve given the NLU training data, whenever it processes a string of text from a user, it uses its knowledge, built over millions of prior interactions with all other customers and training data gather over its entire lifetime, to determine the likelihood of a given utterance corresponding to one of your intents.

That means that a user utterance doesn’t have to match a specific phrase in your training data. Similar enough phrases could be matched to a relevant intent, providing the ‘confidence score’ is high enough.

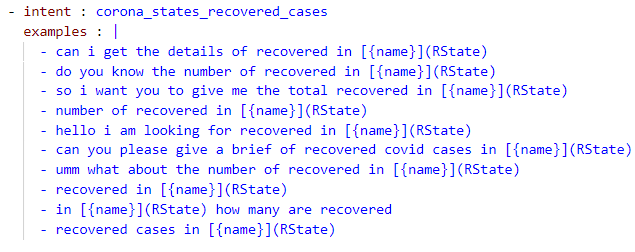
**NLU training for Intent and Entities:**

****

**Intent**: An intent is an action that a user intends to perform. You can think of it as the meaning behind what the user said. Identifying the intent/purpose of the user’s message.

NLU systems determine the intent by taking the input text written by the user (or transcribed by a speech to text system), and matching that to similar text in its ‘training data’ (more on training data in a moment).

For example: consider “What is the number of recovered cases?” is the input message. The intent of this message is to get details of recovered cases. NLU will have to rightly identify this. This is termed as Intent classification.

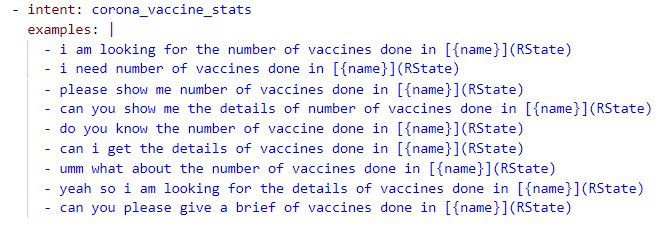


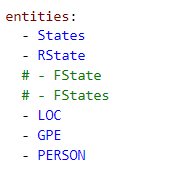
**Entity**: An entity is a specific piece of data or information that’s particularly important, sometimes crucial, for a given intent. Input messages may contain information like name, place etc. These are details which need to be extracted.

Entities can have a variety of values, such as dates, times, locations, towns, cities, numbers or any words, phrases or values that you specify.

Once you have your intents, entities and sample utterances, you have what’s known as a language mode

For example: consider “What is the number of vaccines done in Delhi?”. Here the bot needs to extract the name “Delhi” which is an entity. This is termed as Entity Extraction.

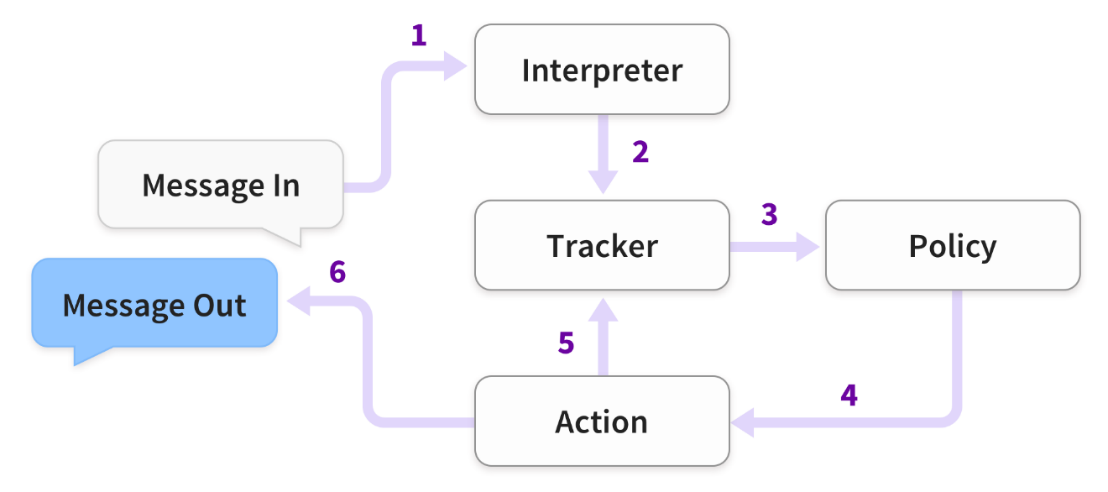




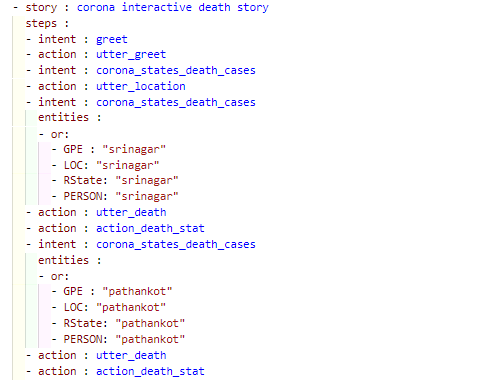
**Dialog Management System:**

a.  A dialogue system is a computer system that is able to engage in an interactive dialogue with a human user about a particular topic.

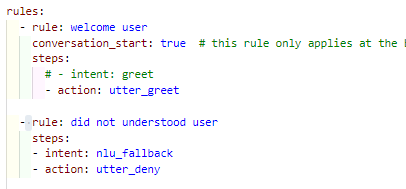
b. Dialog Management determines the actual context of the dialogue. For example, the user might say “I want to know the current stat” and the bot applies NLU on the user input. Then bot will find out it needs more information in order to find locations such as Delhi, Srinagar etc. so it will ask back “Please mention the location”.



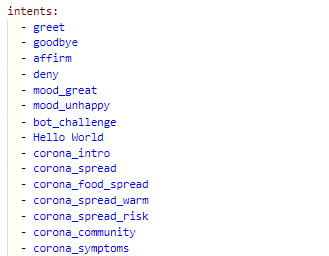
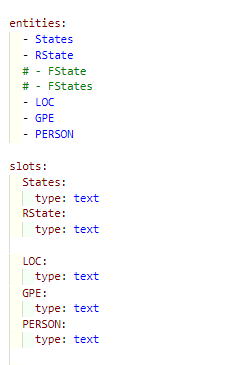
**Create Stories data**: A story is a representation of a conversation between a user and an AI assistant, converted into a specific format where user inputs are expressed as intents (and entities when necessary), while the assistant's responses and actions are expressed as action names.



**Create Rules**: Rules are listed under the rules key and look similar to stories. A rule also has a steps key, which contains a list of the same steps as stories do. Rules can additionally contain the *conversation\_started* and conditions keys.

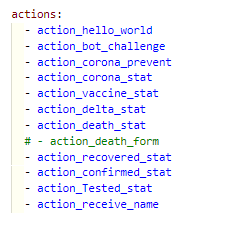


**Create Domains**: The domain defines the universe in which your assistant operates. It specifies the intents, entities, slots, responses, forms, and actions your bot should know about. It also defines a configuration for conversation sessions.

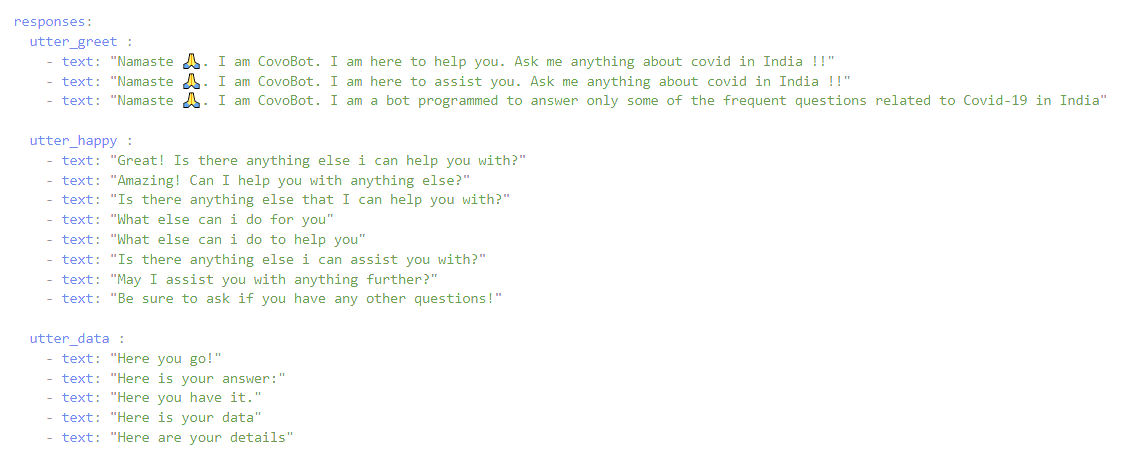
*entities*

*intents*

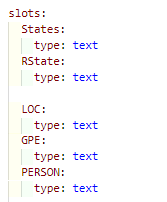


*actions*

*Responses*



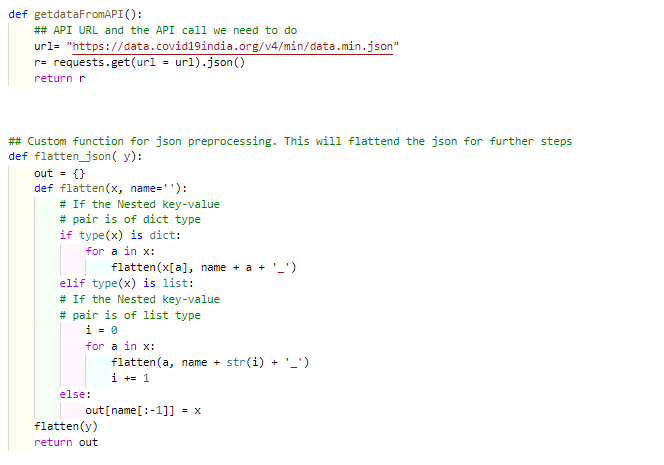
**Create Slots**: Slots are your bot’s memory. They act as key-value stores which can be used to store information the user provided as well as information gathered about the outside world.



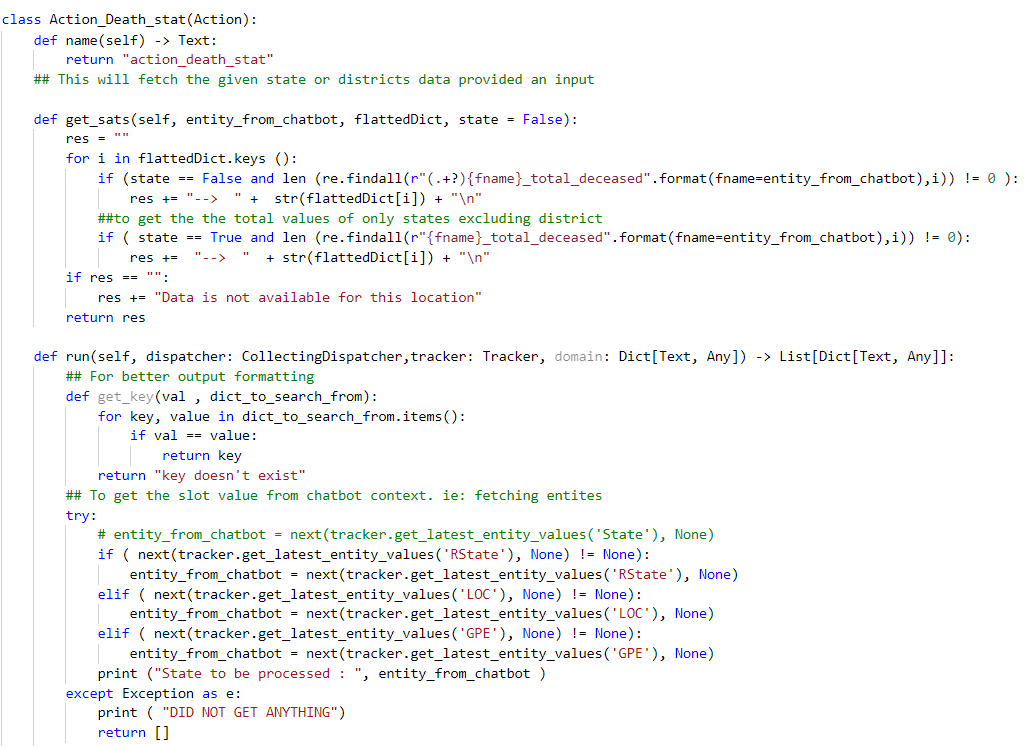
**Create Actions:**

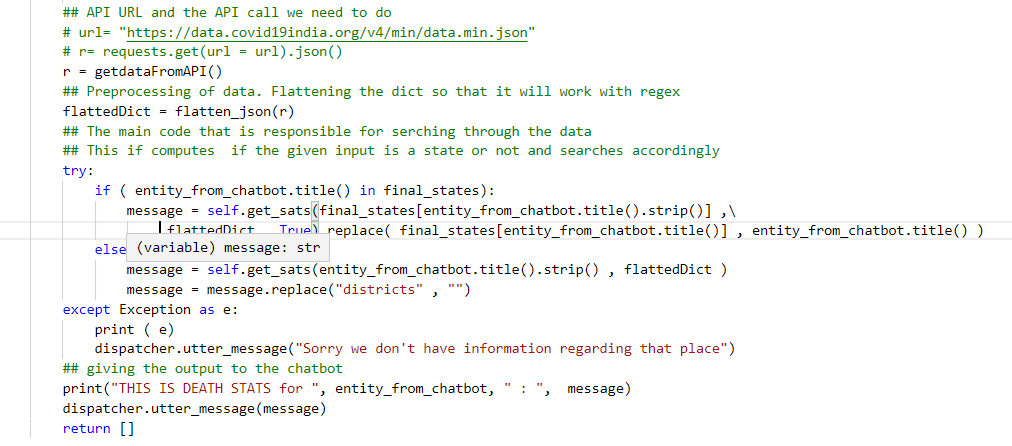
**Actions**: After each user message, the model will predict an action that the assistant should perform next.

Because the data was in JSON, a nested dictionary, we flattened it with a function.

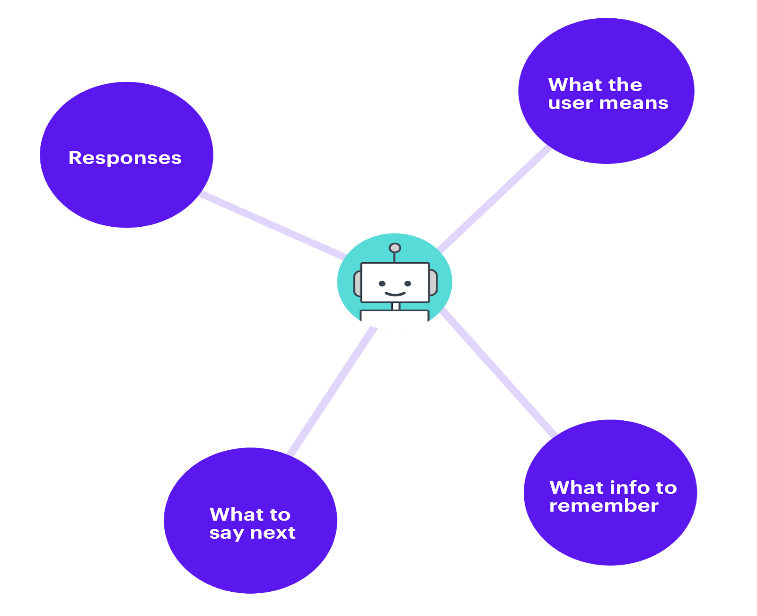


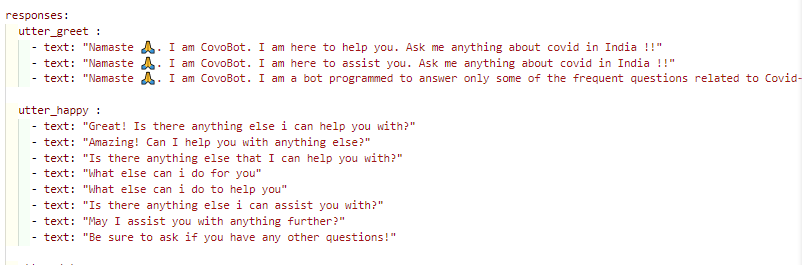
Then we used a function to extract the death stats from the API:





**Responses:** A response is a message the assistant will send back to the user.





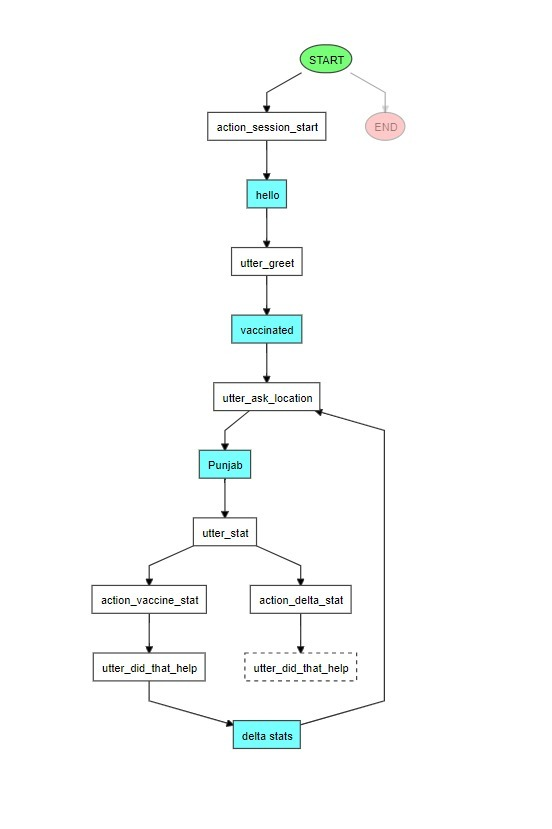
**Custom Actions:**

A custom action is an action that can run any code you want. This can be used to make an API call, or to query a database for example.

**Forms:**

Forms are a special type of custom action, designed to handle business logic.

**The overall visualisation:**



**Backend API’s:**

We used an API from the net, which, for now, is only aimed/trained at solving queries, such as giving the total confirmed cases in a state, or delta cases in a district, or the number of vaccinated people, and many more such queries. APIs managed by government agencies to get all structured data related answers. Example API:

[**https://data.covid19india.org/v4/min/data.min.json**](https://data.covid19india.org/v4/min/data.min.json)

****

JSON-formatted data makes up this API's content. A state name is followed by the total number of stats for the state, as well as a district name and the total stats for the district.

API to get unstructured text as an answer. Since there can be scenarios where information we are looking for present in documents such as side effects of vaccines are part of a document or there can be a set of FAQs like symptoms of Covid etc. In order to get such answers, we need a set of text documents having all the details of the subject or a set of common FAQs to get answers from.

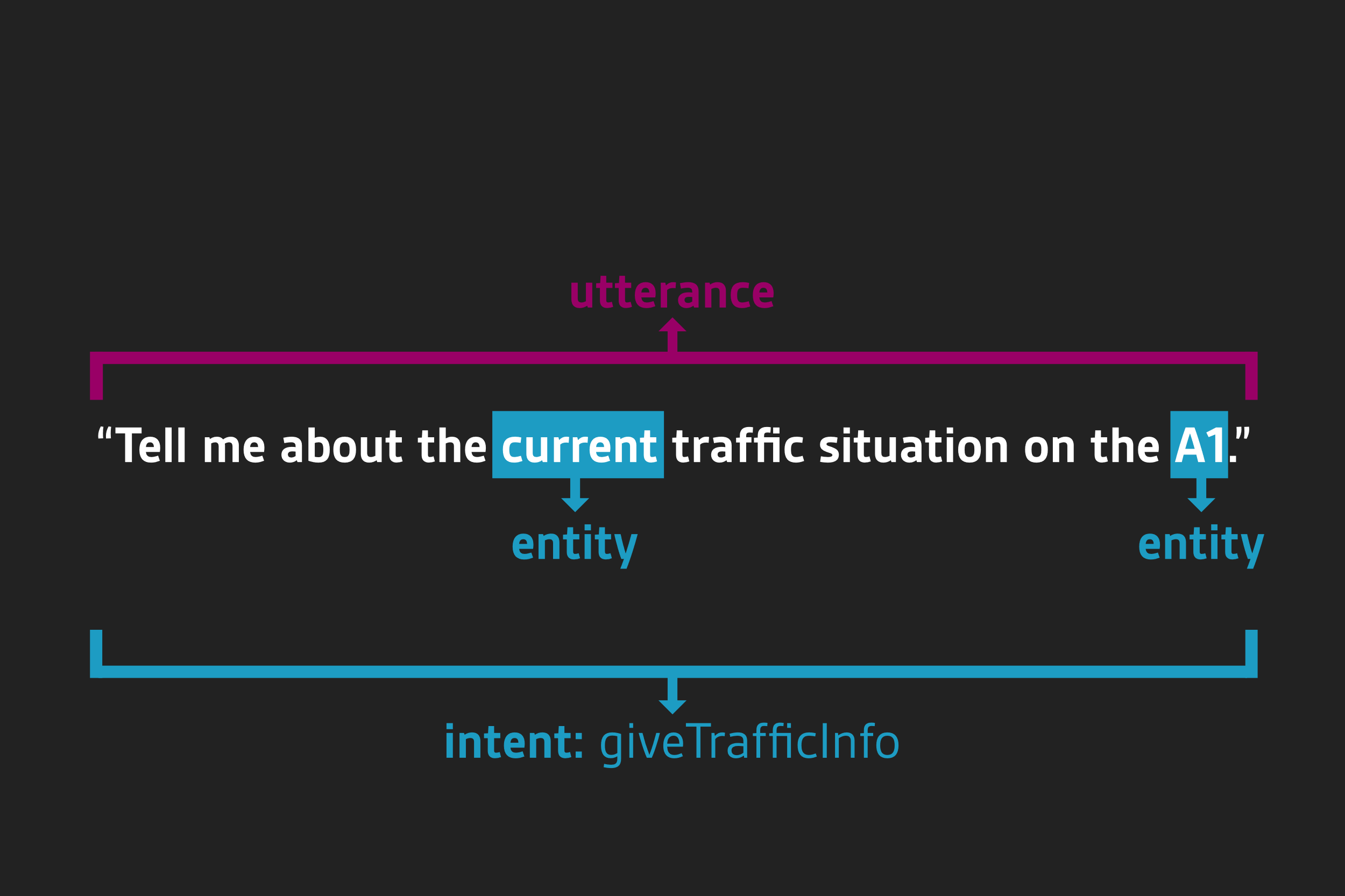
**endpoints.yml**: this file contains different endpoints the bot can use; the model runs on the localhost. Interactions with the bot can happen over the exposed webhooks/<channel>/webhook endpoints.



**credentials.yml**: this file contains the credentials for the chat platform and the details for connecting to other services, you just need to add Telegram, slack, and Bot framework related configuration, rasa will automatically do the rest for you.

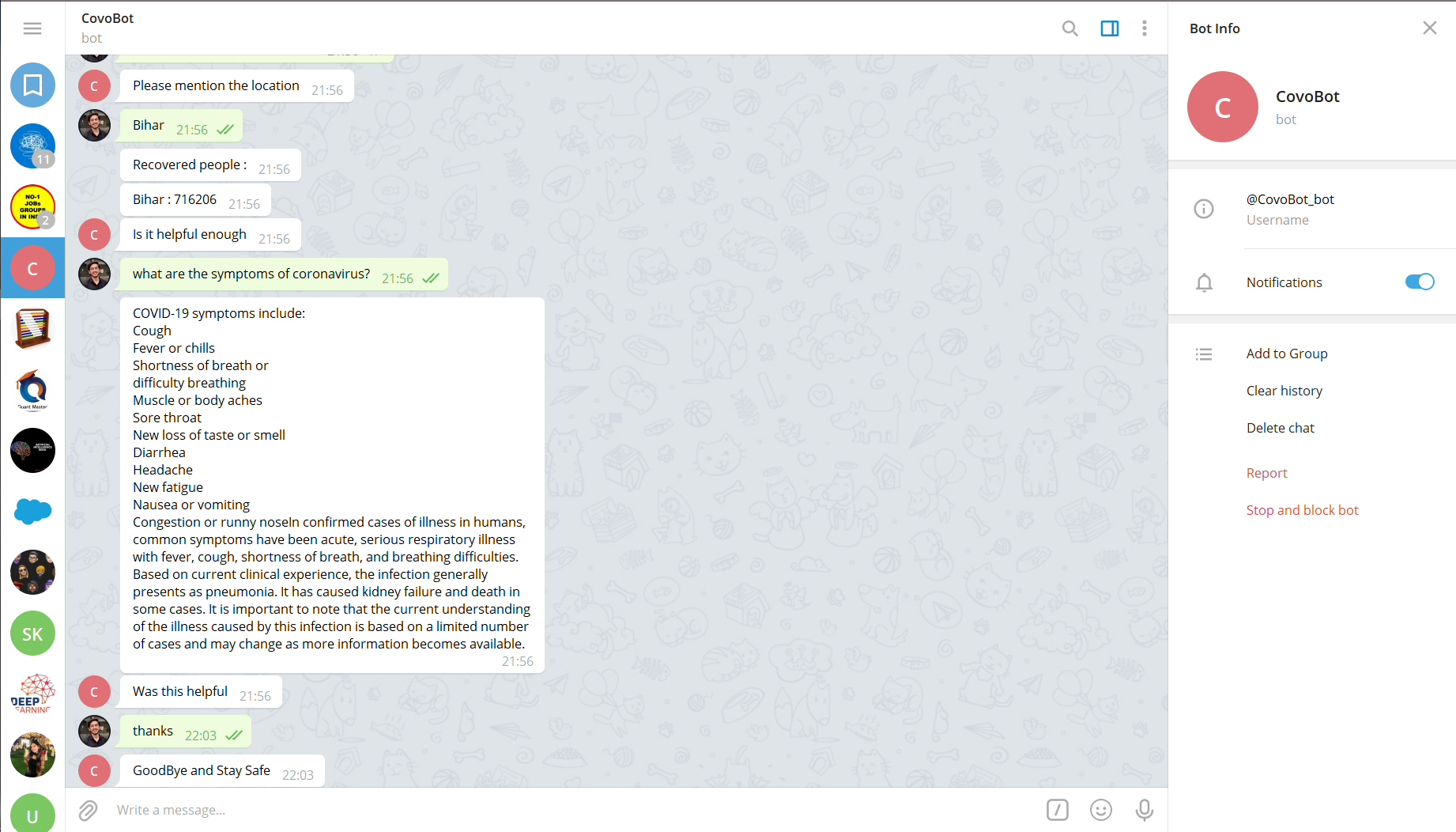
**Detailed Architecture Design:**

**NLU**



Intent: Classification

Entities: Information Extraction

****

*Telegram Integration*

**APIs**



****



**Slot Checker**

**Check if all required information is provided**

*ML model to classify intents and extract entities*

**NLU Trained Model**

**Dialog Trained Model**

**Dialog Management**

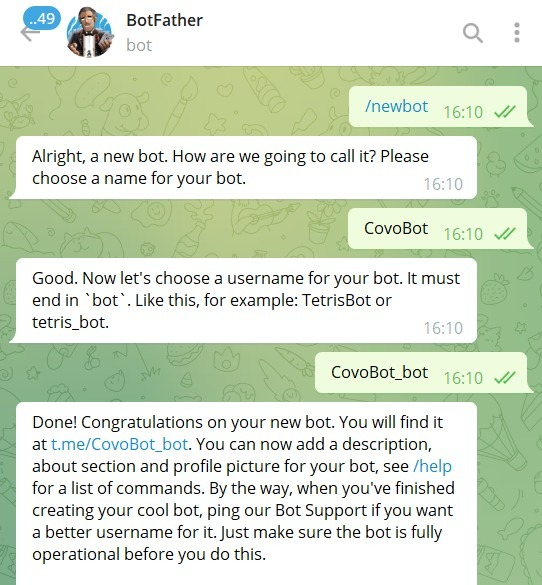
**Integration:**

Telegram Integration: Getting Credentials#

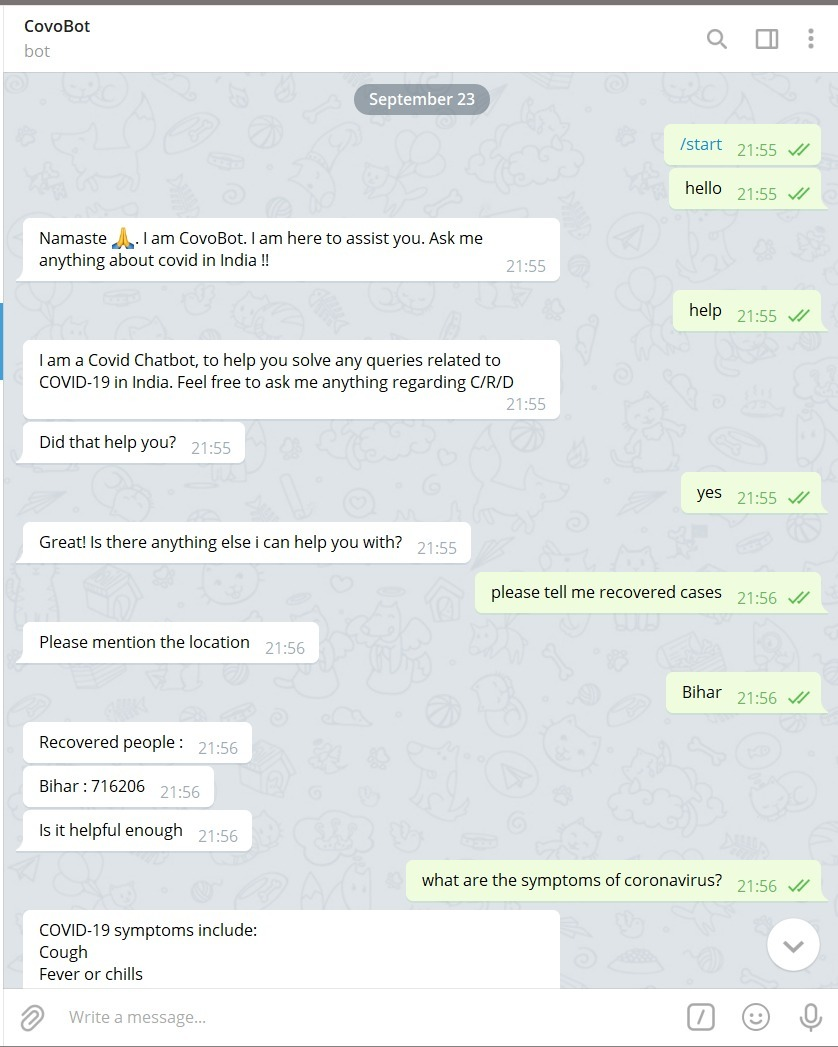
How to get the Telegram credentials: You need to set up a Telegram bot.

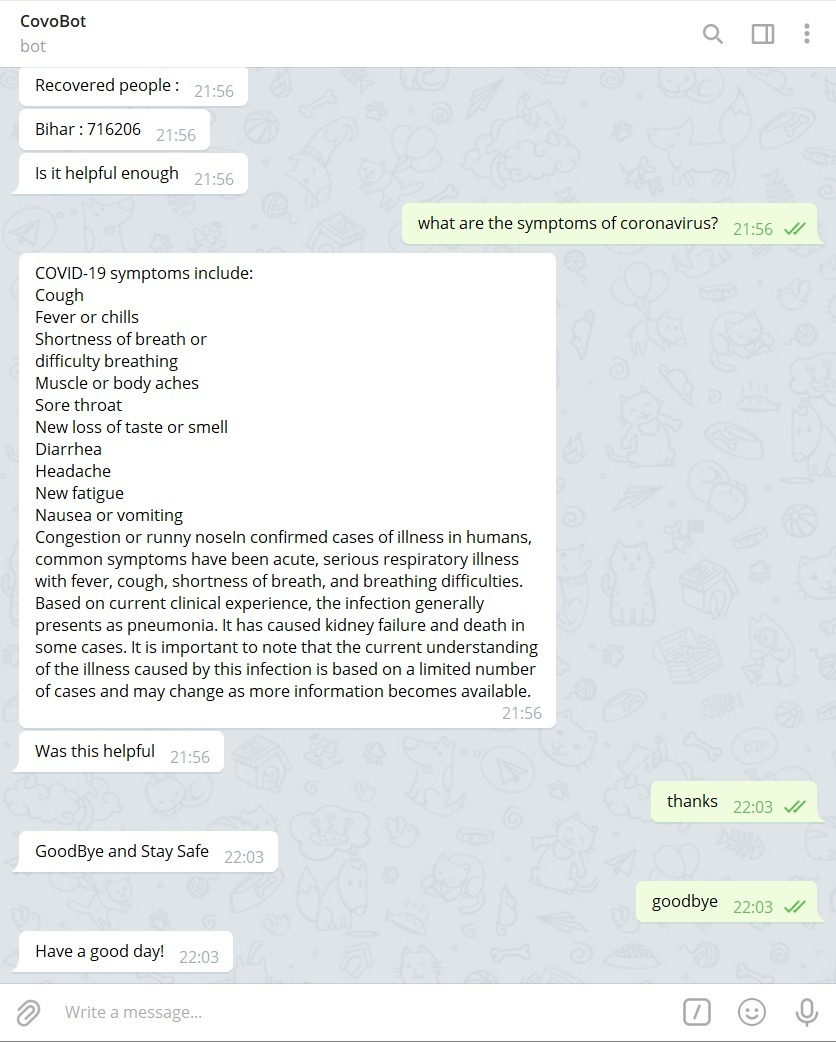
To create the bot, go to *Bot Father*, enter */newbot* and follow the instructions. The URL that Telegram should send messages to will look like *http://<host>:<port>/webhooks/telegram/webhook*, replacing the host and port with the appropriate values from your running Rasa X or Rasa Open Source server.

At the end you should get your *access\_token* and the username you set will be your *verify*. If you want to use your bot in a group setting, it's advisable to turn on group privacy mode by entering */setprivacy*. Then the bot will only listen when a user's message starts with /bot.



**Testing:**





**CONCLUSION:** In this project, an artificially intelligent chatbot is developed to answer questions on COVID-19 utilising natural language processing, rasa, and machine learning**.** Datasets were processed using a variety of natural language processing methods. An artificial neural network is used to build the model, and it is trained using processed data so that our chatbot can provide the right response. The chatbot is evaluated by putting it to the test with a diverse collection of queries. Also, as the dataset grows, the chatbot's accuracy is expected to improve.

**FUTURE SCOPE:** When it comes to curated content, chatbots have a lot of potential. The data may be tailored to fit the specific requirements and symptoms of the user. Answers to particular inquiries can be supplied in a more engaging manner and more quickly than standard techniques of internet search. Additionally, the data may be customised based on the user's location and local laws.

Many corporations and organisations are leading the way in implementing chatbots to deliver COVID-19 information. Worldwide health organisations like WHO and CDC have integrated chatbots into their websites to keep millions of people informed on the disease's progress and symptoms as they emerge. Anyone may use the chatbots from a wide range of devices if they have access to them (online computer, smart phone, or analogue phone in some cases). The use of chatbots for COVID-19 information can minimise the amount of time that hospitals spend on the phone. Chatbots' interactive symptom-checking function might help cut down on emergency room visits and the number of patients who need to be seen there.

The COVID-19 pandemic is a boost for chatbot technology, making it easier for individuals all across the world to use it for healthcare. After the epidemic is over, the use of chatbots in a larger range of healthcare applications will only increase. As a result, public and private parties must work together to develop governance structures that maximise the advantages while reducing the dangers.

**SOURCES:**

* <https://rasa.com/>
* <https://www.analyticsvidhya.com/blog/2019/04/learn-build-chatbot-rasa-nlp-ipl/>
* <https://www.machinelearningplus.com/nlp/chatbot-with-rasa-and-spacy/>
* <https://medium.com/data-science-community-srm/developing-chatbots-with-rasa-intuition-to-implementation-39c1dd34274c>
* <https://vux.world/what-is-natural-language-understanding-nlu-a-beginners-guide/>