RASA is an open source Machine Learning framework for automated text and voice messaging.

Component of RASA

* RASA NLU
* RASA CORE

**RASA Installation for Chatbot**

* Install the anaconda3.
* Install the visual studio 2019 C++ Build Tools
* pip install rasa-x --extra-index-url <https://pypi.rasa.com/simple>
* rasa init

**RASA Configuration**

Let me explain about files, which are created as Initial project structure of Rasa.

**\_iniy\_.py** - an empty file that helps python find your actions.

**actions.py** - code for your custom actions. In-case you want Rasa to call external server via REST API or API call, you can define your Custom Actions here. Remember you can create multiple Python Script for Rasa Custom Action.

**Config.yml** - configuration of your NLU and Core models. In-case you are dealing with Tensor flow or Spacy, you need to define such pipeline here. To handle this file, you show know about Machine Learning and Deep Learning.

**Credentials.yml** - details for connecting to other services. In case you want to build Bot on Facebook Messenger, Microsoft Bot Framework, you can maintain such credential and token here. So basically you just need to add Facebook, slack and Bot framework related configuration, rasa will automatically do rest for you. Remember that you need to host Rasa over https domain. During development, you can use ngrok as a testing tool.

**Data/nlu.md** - your NLU training data. Here you can define Intent. Like Order Pizza or Book Uber. You need to add related Sentences for that Intent. Remember if you are using Rasa-X, your training Intent and Data will be added automatically.

**Data/stories.md** - your stories. This is required for Rasa Core. There is something called “Dialog Flow in Rasa” where Rasa Core controls the flow of the conversation between you and chatbot, so for that flow, you need to train chatbot using these stories. So in case you want your chatbot to be very perfect on different context (stories) you can add those stories here.

**Domain.yml** - your assistant’s domain. This file combines Different Intent which chatbot can detect and list of Bot replies. Remember you can define your Custom Action Server Python method name here (in underscore format), so that Rasa will call that python method for you.

**Endpoints.yml** - details for connecting to channels like FB messenger. This is mainly used for production setup. You can configure your Database like redis so that Rasa can store tracking information.

**models/<timestamp>.tar.gz** your initial model.

**Use of RASA-X**

This is the newest addition to Rasa Stack. You can user Rasa-X to try your chatbot on Browser. You can download training Data.

* rasa train

NLU:

Rasa NLU is an open-source natural language processing tool for intent classification, response retrieval and entity extraction in chatbots. For example, taking a sentence like

"I am looking for a Mexican restaurant in the center of town"

{

**"intent"**: "search\_restaurant",

**"entities"**: {

**"cuisine"** : "Mexican",

**"location"** : "center"

}

}

Language Support

You can use Rasa to build assistants in any language you want! Rasa’s supervised\_embeddings pipeline can be used on training data in any language. This pipeline creates word embedding from scratch with the data you provide.

Pre-trained word vectors

* Spacy
* MITIE

**Messaging and Voice Channels**

* To make your assistant available on a messaging platform you need to provide credentials in a Credentials.yml file.

List of assistant available on

* Your own website
* FB messenger
* Slack
* Telegram
* Twilio
* Microsoft Bot Framework
* Cisco WebEx Teams
* Rocket chat
* Matter Most
* Google hangouts chat
* Custom connectors.

**SLOTS**

Slots are your bot’s memory. Key value pair store that provide by user.

Types:

* Text Slots
* Boolean
* Categorical
* Float
* List
* Unfeaturized

Example:

Slots:

EntityName:

-type: text

Share the Chat bot global Link

* Install the ngrok to create the global link

**RASA Web Chat Integration**

* Configure the Credentials.yml

socketio:  
 user\_message\_evt: user\_uttered  
 bot\_message\_evt: bot\_uttered  
 session\_persistence: true

* Create the index.html file in chat bot project and added it.

<div id="webchat"/>  
<script src="https://storage.googleapis.com/mrbot-cdn/webchat-latest.js"></script>  
// Or you can replace latest with a specific version  
<script>  
 WebChat.default.init({  
 selector: "#webchat",  
 initPayload: "/get\_started",  
 customData: {"language": "en"}, // arbitrary custom data. Stay minimal as this will be added to the socket  
 socketUrl: "http://localhost:5005",  
 socketPath: "/socket.io/",  
 title: "ChatBot",  
 subtitle: "Demo Chat",  
 })  
</script>

* Open the terminal run the command: rasa run -m models --enable-api --cors "\*" –debug
* Go to the project folder click the index.html page.

**Voice Bot using Chat Bot:**

* Install the speech recognition developed by Google

Pip install SpeechRecognition PyAudio