# Rasa Tips

1. nlu.md , domain.yml is connected == user input
2. domain.yml – actions, response connected == robot response

# Rasa Chatbot –NOT WORKING

<https://www.youtube.com/watch?v=GwaSJUlB8oA&list=PLtFHvora00y8NBwCMoNnPqii-y2-gyl5p&index=2>

1. Create new env – python3.7
2. > pip install rasa-x --extra-index-url <https://pypi.rasa.com/simple>
3. > pip install rasa[spacy]
4. > python -m spacy download en\_core\_web\_md
5. > python -m spacy link en\_core\_web\_md en
6. > pip install sanic==19.12.2
7. > cd project - dir
8. > rasa init --no-prompt

## Run chatbot

1. rasa x

Error

ERROR: Could not install packages due to an EnvironmentError: [Errno 2] No such file or directory: 'c:\\users\\my\\appdata\\roaming\\python\\python36\\site-packages\\pycparser-2.20.dist-info\\METADATA'

Solution:- Use python3.7 or create new environment

## Ubuntu

1. > sudo apt update

# Sumi raj book

## environment

Chat bot rasa

1. > conda create -n sumit\_raj\_chatbot python=3.5

spacy==2.0.11

python -m spacy download en

rasa\_core==0.11.1

pip install rasa-nlu==0.13.2

pip install rasa-core-sdk==0.11.0

pip install sklearn\_crfsuite

pip install setuptools==39.1.0

## Create data

1. Install node.js
2. > npm install I –g rasa-nlu-trainer
3. Create new folder in project directory name = data
4. Move to data folder
5. Create data.json file

{

  "rasa\_nlu\_data": {

    "common\_examples": [

      {

        "text": "Hello",

        "intent": "greeting",

        "entities": []

      },

      {

        "text": "I want to know my horoscope",

        "intent": "get\_horoscope",

        "entities": []

      },

      {

        "text": "Can you please tell me horoscope?",

        "intent": "get\_horoscope",

        "entities": []

      },

      {

        "text": "Please subscribe me",

        "intent": "subscription",

        "entities": []

      },

      {

        "text": "19-01",

        "intent": "dob\_intent",

        "entities": [

          {

            "start": 0,

            "end": 2,

            "value": "19",

            "entity": "DD"

          },

          {

            "start": 3,

            "end": 5,

            "value": "01",

            "entity": "MM"

          }

        ]

      }

    ],

    "regx\_features": [],

    "entity\_synonyms": []

  }

}

1. > rasa-nlu-trainer
2. Create intent and data

## Train data

1. Create **config.json** project directory – outside data folder

{

    "pipeline": "tensorflow\_embedding",

    "path" : "./models/nlu",

    "data": "./data/data.json"

}

1. Create mTrain.py

from rasa\_nlu.training\_data import load\_data

from rasa\_nlu.model import Trainer

from rasa\_nlu import config

from rasa\_nlu.model import Interpreter

def train\_horoscopebot(data\_json, config\_file, model\_dir):

    training\_data = load\_data(data\_json)

    trainer = Trainer(config.load(config\_file))

    trainer.train(training\_data)

    model\_directory = trainer.persist(model\_dir, fixed\_model\_name = 'schoolbot')

def predict\_intent(text):

    interpreter = Interpreter.load("./models/nlu/default/schoolbot")

    print(interpreter.parse(text))

train\_horoscopebot('./data/data.json', 'config.json', './models/nlu')

1. Python mTrain.py
2. Create mPredict.py

from rasa\_nlu.training\_data import load\_data

from rasa\_nlu.model import Trainer

from rasa\_nlu import config

from rasa\_nlu.model import Interpreter

def train\_horoscopebot(data\_json, config\_file, model\_dir):

    training\_data = load\_data(data\_json)

    trainer = Trainer(config.load(config\_file))

    trainer.train(training\_data)

    model\_directory = trainer.persist(model\_dir, fixed\_model\_name = 'schoolbot')

def predict\_intent(text):

    interpreter = Interpreter.load("./models/nlu/default/schoolbot")

    print(interpreter.parse(text))

while True:

    text = input("User input (q - quit): ")

    if text == "q":

        break

    predict\_intent(text)

1. Python mPredict.py
2. Test sentence

## Actions

Next action to be taken in response to a dialog state

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import unicode\_literals

import requests

from rasa\_core\_sdk import Action

from rasa\_core\_sdk.events import SlotSet

class GetTodaysHoroscope(Action):

    def name(self):

        return "get\_todays\_horoscope"

    def run(self, dispatcher, tracker, domain):

        # type: (Dispatcher, DialogueStateTracker, Domain) -> List[Event]

        user\_horoscope\_sign = tracker.get\_slot('horoscope\_sign')

        base\_url = "http://horoscope-api.herokuapp.com/horoscope/{day}/{sign}"

        url = base\_url.format(\*\*{'day': "today", 'sign': user\_horoscope\_sign})

        #http://horoscope-api.herokuapp.com/horoscope/today/capricorn

        res = requests.get(url)

        todays\_horoscope = res.json()['horoscope']

        response = "Your today's horoscope:\n{}".format(todays\_horoscope)

        dispatcher.utter\_message(response)

        return [SlotSet("horoscope\_sign", user\_horoscope\_sign)]

class SubscribeUser(Action):

    def name(self):

        return "subscribe\_user"

    def run(self, dispatcher, tracker, domain):

        # type: (Dispatcher, DialogueStateTracker, Domain) -> List[Event]

        subscribe = tracker.get\_slot('subscribe')

        if subscribe == "True":

            response = "You're successfully subscribed"

        if subscribe == "False":

            response = "You're successfully unsubscribed"

        dispatcher.utter\_message(response)

        return [SlotSet("subscribe", subscribe)]

## Template Creation

1. Horoscope\_domain.yml

slots:

  horoscope\_sign:

    type: text

  DD:

    type: text

  MM:

    type: text

  subscribe:

    type: bool

intents:

 - greeting

 - get\_horoscope

 - subscription

 - dob\_intent

entities:

 - horoscope\_sign

 - DD

 - MM

 - subscribe

 - dob\_intent

templates:

  utter\_greet:

    - "Hello! What can I do for you?"

  utter\_ask\_horoscope\_sign:

    - "What is your horoscope sign?"

  utter\_ask\_dob:

    - "What is your DOB in DD-MM format?"

  utter\_subscribe:

    - "Do you want to subscribe for daily updates?"

actions:

 - utter\_greet

 - utter\_ask\_horoscope\_sign

 - utter\_ask\_dob

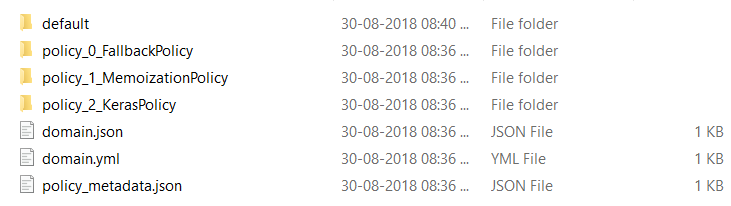
 - utter\_subscribe

 - get\_todays\_horoscope

 - subscribe\_user

## RealTime Training

1. Paste this code in model directory



1. Create **endpoints.yml**

action\_endpoint:

  url: http://localhost:5055/webhook

# nlg:

#   url: http://localhost:5056/nlg

core\_endpoint:

  url: http://localhost:5005

1. Create **train\_online.py** for intractive learning.

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import unicode\_literals

import logging

from rasa\_core import utils, train

from rasa\_core.training import online

from rasa\_core.interpreter import NaturalLanguageInterpreter

logger = logging.getLogger(\_\_name\_\_)

def train\_agent(interpreter):

    return train.train\_dialogue\_model(domain\_file="horoscope\_domain.yml",

                                      stories\_file="./data/stories.md",

                                      output\_path="./models/dialogue",

                                      nlu\_model\_path=interpreter,

                                      endpoints="endpoints.yml",

                                      max\_history=2,

                                      kwargs={"batch\_size": 50,

                                              "epochs": 200,

                                              "max\_training\_samples": 300

                                              })

if \_\_name\_\_ == '\_\_main\_\_':

    utils.configure\_colored\_logging(loglevel="DEBUG")

    nlu\_model\_path = "./models/nlu/default/horoscopebot"

    interpreter = NaturalLanguageInterpreter.create(nlu\_model\_path)

    agent = train\_agent(interpreter)

    online.serve\_agent(agent)

1. Open new cmd
2. > python -m rasa\_core\_sdk.endpoint --actions actions
3. Run this in fresh cmd
4. Create new file **stories.md** in data folder

**## Generated Story 3797421409943253925**

\* greeting

    - utter\_greet

1. > python train\_online.py

# Rasa tutorial

## installation

1. Create new env
2. > conda create --name installingrasa python==3.7.6
3. > conda install ujson
4. > conda install tensorflow
5. > pip install rasa
6. install rasa x 0.23.3
7. > curl -sSL -o install.sh <https://storage.googleapis.com/rasa-x-releases/0.23.3/install.sh>
8. > pip install rasa-x==0.23.3 --extra-index-url https://pypi.rasa.com/simple

## init

<https://www.youtube.com/watch?v=rlAQWbhwqLA&list=PL75e0qA87dlHQny7z43NduZHPo6qd-cRc>

1. > rasa init
2. > project folder name
3. > y
4. > y

## Create nlu training data

1. Move to ./data/nlu.md
2. Add the following intents

**## intent:inform**

- [Sitka](location)

- [Virjinia](location)

- [San Francisco](location)

**## intent:search\_provider**

- I need a [hospital](facility)

- find me a nearby [hospital](facility)

- show me [home helth agencies](facility)

- [hospital](facility)

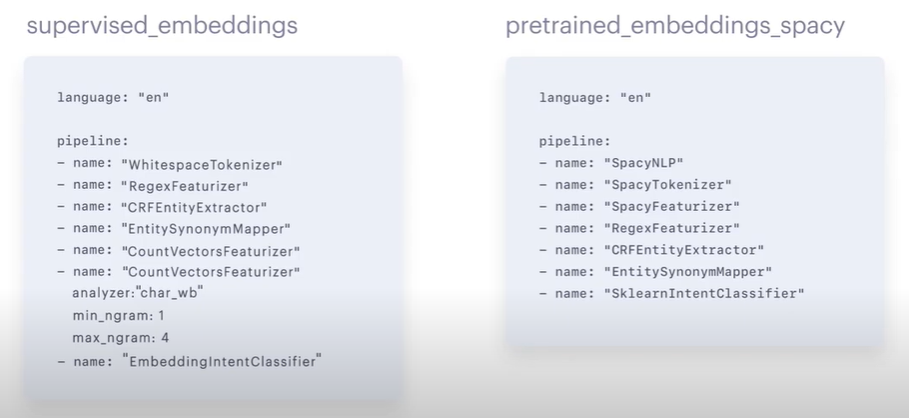
- find me a nearby [hospital](facility) in [san francisco](location)

- i need a [home health agency](facility)

## Pre configured pipeline

1. > rasa train nlu
2. Testing
3. > rasa shell nlu
4. > Hello there
5. > I am looking for hospital
6. Available pipelines





Named entity recognition : -DucklingHttpExtractor

## Dialogue management

1. Add this story to ./data/stories.md

**## search hospital happy path**

\* greet

  - utter\_greet

\* search\_provider{"facility": "hospital", "location": "san francisco"}

  - action\_facility\_search

\* thanks

  - utter\_goodbye

### domain

1. Open domain.yml file
2. Add all entities in entity tag

  - inform

  - search\_provider

1. Add entity and actions

entities:

  - location

  - facility

actions:

  - utter\_how\_can\_i\_help

  - action\_facility\_search

1. Add slot

slots:

  location:

    type: text

    auto\_fill: False

1. Complete code

intents:

  - greet

  - goodbye

  - affirm

  - deny

  - mood\_great

  - mood\_unhappy

  - bot\_challenge

  - inform

  - search\_provider

entities:

  - location

  - facility

actions:

  - utter\_how\_can\_i\_help

  - action\_facility\_search

slots:

  location:

    type: text

  facility:

    type: text

  address:

    type: unfeaturized

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."

  utter\_how\_can\_i\_help:

  - text: "Hello i am a medical locator. I can help you find nearest hospital. How can i help?"

  - text: "Hi my name is medical locator. How can i help today?"

session\_config:

  session\_expiration\_time: 60

  carry\_over\_slots\_to\_new\_session: true

### action

1. Open action.py
2. Add this code

from typing import Any, Text, Dict, List

from rasa\_sdk import Action, Tracker

from rasa\_sdk.executor import CollectingDispatcher

class ActionFacilitySearch(Action):

    def name(self) -> Text:

        return "action\_facility\_search"

    def run(self, dispatcher: CollectingDispatcher,

            tracker: Tracker,

            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        dispatcher.utter\_message(text="Sure i am on it!")

        return []

1. Name of class should be included in domain.yml file
2. Complete code stories.md

**## search hospital happy path**

\* greet

  - utter\_how\_can\_i\_help

\* search\_provider{"facility": "hospital", "location": "san francisco"}

  - action\_facility\_search

  - slot{"address" : "300 Hyde St, San Francisco"}

\* thanks

  - utter\_goodbye

**## happy path**

\* greet

  - utter\_greet

\* mood\_great

  - utter\_happy

**## sad path 1**

\* greet

  - utter\_greet

\* mood\_unhappy

  - utter\_cheer\_up

  - utter\_did\_that\_help

\* affirm

  - utter\_happy

**## sad path 2**

\* greet

  - utter\_greet

\* mood\_unhappy

  - utter\_cheer\_up

  - utter\_did\_that\_help

\* deny

  - utter\_goodbye

**## say goodbye**

\* goodbye

  - utter\_goodbye

**## bot challenge**

\* bot\_challenge

  - utter\_iamabot

1. Actions.py

# 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

from rasa\_sdk.events import SlotSet

class ActionFacilitySearch(Action):

    def name(self) -> Text:

        return "action\_facility\_search"

    def run(self, dispatcher: CollectingDispatcher,

            tracker: Tracker,

            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        facility = tracker.get\_slot("facility")

        address = "300 Hyde St, San Francisco"

        dispatcher.utter\_message(text="Here is the address of the {}:{}".format(facility, address))

        return [SlotSet("address", address)]

## train

1. Open cmd
2. > rasa train

## endpoints

1. Open endpoints.yml
2. Uncomment this line

action\_endpoint:

 url: "http://localhost:5055/webhook"

## Testing

1. Open cmd
2. > rasa run actions
3. Open a fresh cmd
4. > rasa shell

## policies

1. Actions.yml



1. Fallback policy



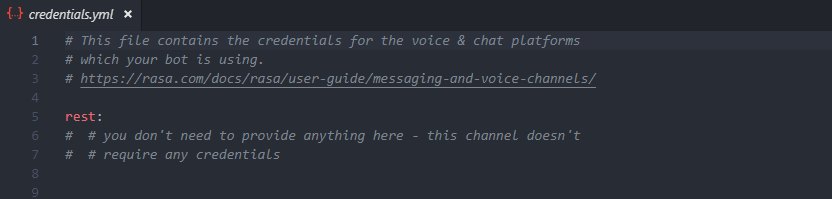
## Implementing custom actions with backend integrations, forms and fallback

1. Go to this site
2. > <https://data.medicare.gov/Hospital-Compare/Hospital-General-Information/xubh-q36u>
3. Form Action
4. Open action.py

# Rest api connection

<https://forum.rasa.com/t/rest-api-implementation/12624/3>

Step1: you need to ensure your credentials.yml has the following content:

[[](https://forum.rasa.com/uploads/default/original/2X/7/76dc4a48e6edc9edc9e3ae1351bbfd59b755465b.png)](https://forum.rasa.com/uploads/default/original/2X/7/76dc4a48e6edc9edc9e3ae1351bbfd59b755465b.png" \o "image.png)

Step2: Once you have trained your bot, you can start your bot server by running the below command

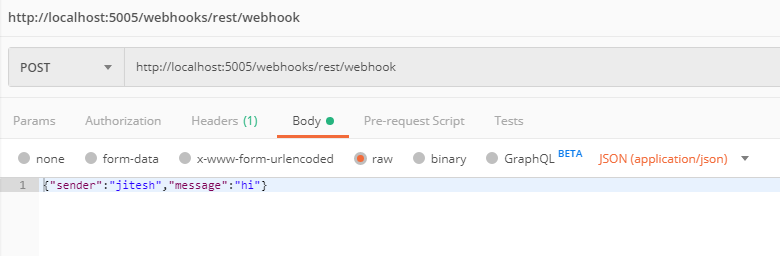
rasa run -m models --enable-api --cors “\*” --debug

Once you run this command, you can see the below output in the terminal image

Step3: Once your server is up and running you can access your bot throught REST api, given below:

[http://localhost:5005/webhooks/rest/webhook 195](http://localhost:5005/webhooks/rest/webhook)

Step4: Now you can test the api:

[[](https://forum.rasa.com/uploads/default/original/2X/f/f73e7831628e284858b341f9e496bf8d4edc3761.png)](https://forum.rasa.com/uploads/default/original/2X/f/f73e7831628e284858b341f9e496bf8d4edc3761.png" \o "image.png)

**[image.png](https://forum.rasa.com/uploads/default/original/2X/f/f73e7831628e284858b341f9e496bf8d4edc3761.png" \o "image.png)**