Rasa Introduction

Author: Deebul Nair, Prof. Dr. Teena Hassan

Institute: Hochschule Bonn-Rhein-Sieg

Today's Lab

- Installing Rasa
- Steps involved in creating an assistant
- Example Create a weather assistant

  (Github Link)[https://github.com/deebuls/rasa-weather-assistant-v3]

Rasa: Introduction

- Rasa is an open-source machine learning framework that enables developers to build and deploy conversational AI assistants.
- Rasa allows developers to build chatbots and voice assistants that can handle complex natural language processing (NLP) tasks and multi-turn conversations.

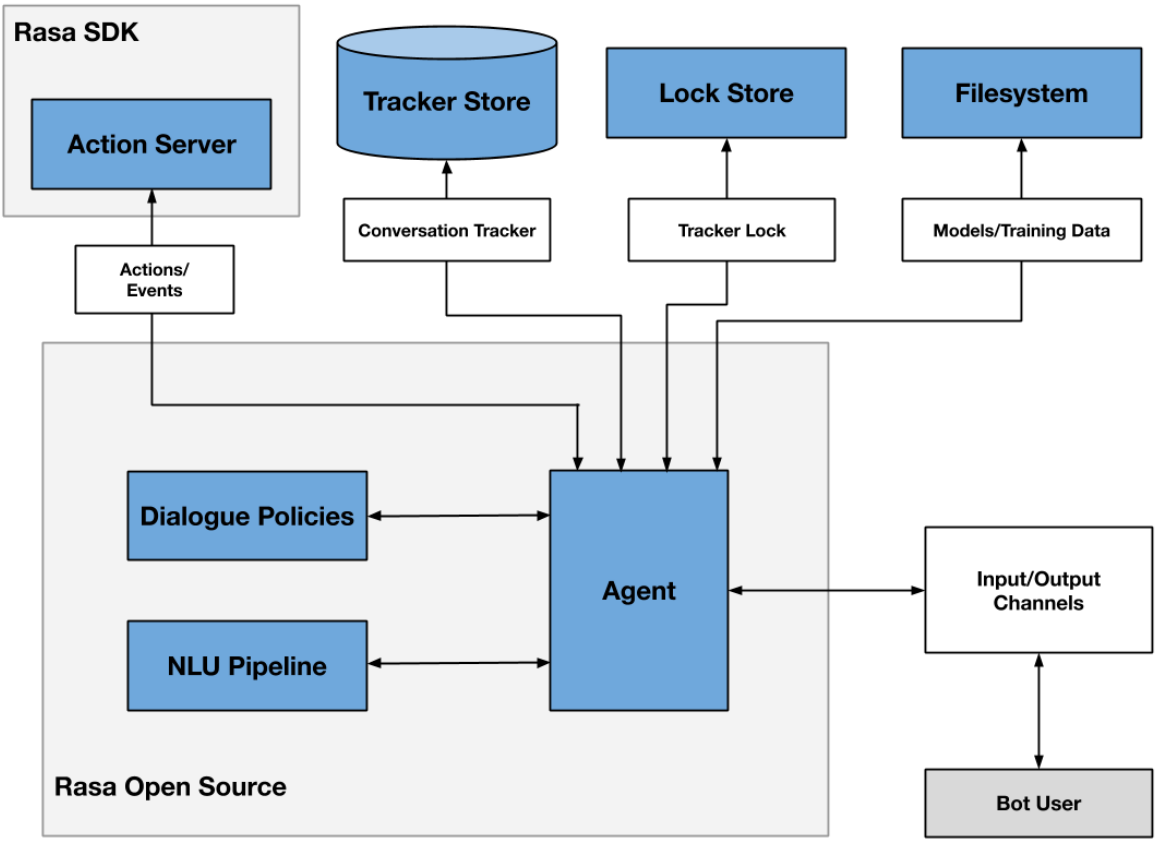
Rasa: Introduction

- Rasa uses the concept of 5 levels of assistant
  - Level 1: Notification Assistants
    - Capable of sending simple notifications, like a text message, push notification, or WhatsApp message.
  - Level 2: FAQ Assistants
    - Can answer simple questions, like FAQs.
    - Often constructed around a set of rules or a state machine.
  - Level 3: Contextual Assistants
    - Able to understand the context of the conversation.
    - Capable of understanding and responding to different and unexpected inputs
    - Can learn from previous conversations and improve in accuracy over time
  - Level 4: Personalised Assistants
    - The next generation of AI assistants, that will get to know you better over time
  - Level 5: Autonomous Organization of Assistants
    - AI assistants that know every customer personally

Source : Rasa Handbook

Rasa: Internals

- Rasa provides two main components:
  - Rasa NLU (Natural Language Understanding): A tool for understanding user input and extracting important information, such as user intents and entities.
  - Rasa Dialogue Policies: A tool for building conversational AI assistants that can handle multi-turn conversations.



Rasa Playground

https://rasa.com/docs/rasa/next/playground/

- Understand NLU
- Understand responses
- Understand stories
- Train
- Download

Building a Weather Assistant with Rasa

In this section, we'll take a look at how to build a weather assistant using Rasa. We'll be following the tutorial provided in the blog: https://pub.towardsai.net/rasa-101-building-a-weather-assistant-244489316d11

- Steps in creating assistant are :
- Defining the NLU -> yaml
- Defining the stories -> yaml
- Defining the actions -> Python
- Defining the domain -> yaml
- Training
- Testing

Rasa Installaiton

1. Install Rasa The first step is to install Rasa. You can do this by running the following command in your terminal:

```
$> sudo apt update
$> sudo apt install python3-dev python3-pip
```

- Setting Virtual env

```
$> python3 -m venv ./venv
$> source ./venv/bin/activate
```

- Install rasa in venve

```
$> pip3 install rasa
```

2. Create a new Rasa project Next, we'll create a new Rasa project. You can do this by running the following command:

```
$> rasa init --no-prompt
```

This will create a new Rasa project with some default files and folders.

Rasa Weather Assistant

1. Should begin a conversation
2. Introduce itself
3. Get information form user about the location and time for which they want weather information
4. Provide weather information
5. Close conversation

Define the NLU

- NLU example

```
nlu:


- intent: weather
  examples: |
    - tell me the weather
    - I want to know the weather
    - weather please
    - what's the weather today
    - what's the temperature right now
    - hows the weather going to be
    - i want to know the temperature

- intent: weather_city
  examples: |
    - tell me the weather in [Paris](location)
    - I want to know the weather in [London](location)
    - weather for [Athens](location) please
    - what's the weather today in [Berlin](location)
    - what's the temperature in [Washington](location) right now
    - hows the weather in [Brussels](location)
    - i want the temperature at [Rome](location)

- intent: city
  examples: |
    - [Liverpool](location)
    - [Milan](location)
    - [Frankfurt](location)
    - [Porto](location)
    - [Madrid](location)
    - [Barcelona](location)
    - [Oslo](location)
```

Define the weather assistant stories

In Rasa, stories define the possible paths that a conversation can take. To define the stories for our weather assistant, we'll create a new file called stories.yml in the ./data folder.

The stories.yml file should contain the following:

```
stories:

- story: ask weather happy path
  steps:
  - intent: weather
  - action: utter_ask_location
  - intent: city
    entities:
    - location: Madrid
  - slot_was_set:
    - location: Madrid
  - action: action_get_weather
  - slot_was_set:
    - location: null

- story: ask weather with city happy path
  steps:
  - intent: weather_city
    entities:
    - location: Rome
  - slot_was_set:
    - location: Rome
  - action: action_get_weather
  - slot_was_set:
    - location: null
```

Define the weather assistant actions

In Rasa, actions define the behavior of our assistant. To define the actions for our weather assistant, we'll create a new file called actions.py in the ./actions folder.

The actions.py file should contain the following:

```
class ActionWeather(Action):
    def name(self) -> Text:
        return "action_weather"

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

        city = tracker.get_slot("city")
        weather = get_weather(city)

        if weather is None:
            dispatcher.utter_message("Sorry, I could not get the weather for that city. Please try again.")
        else:
            dispatcher.utter_message("The weather in {} is {}.".format(city, weather))

        return []

def get_weather(city: Text) -> Optional[Text]:
    # code to get the weather for the given city
    return "sunny"
```

Define the weather assistant domain

In Rasa, the domain defines the actions, intents, entities, and templates that our assistant can use. To define the domain for our weather assistant, we'll create a new file called domain.yml in the ./data folder.

The domain.yml file should contain the following:

```
 version: "3.1"
intents:
- greet
- goodbye
- affirm
- deny
- mood_great
- mood_unhappy
- bot_challenge
- weather
- weather_city
- city
entities:
- location
slots:
  location:
    type: text
    influence_conversation: true
    mappings:
    - type: from_entity
      entity: location
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_ask_location:
  - text: For which city?
actions:
- action_get_weather
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
```

This defines three intents: greet, goodbye, and inform. It also defines an entity called city, which represents the city the user wants the weather for.

Training and Deploy

```
$> rasa train
```

```
$> rasa run actions
```

Another command line shell

```
$> rasa shell
```

Any problems

1. Say hi and then ask weather

```
$> rasa visualize
```

How to debug

1. Visualize
2. Learn

References

1. https://rasa.com/docs/rasa/arch-overview
2. https://cdn2.hubspot.net/hubfs/6711345/ebook-v3.pdf?hsCtaTracking=2cf912f3-4137-4338-829e-08bb4713f0f6%7Cda22eae5-512d-48fe-b46a-c74517f3d870
3. https://rasa.com/blog/