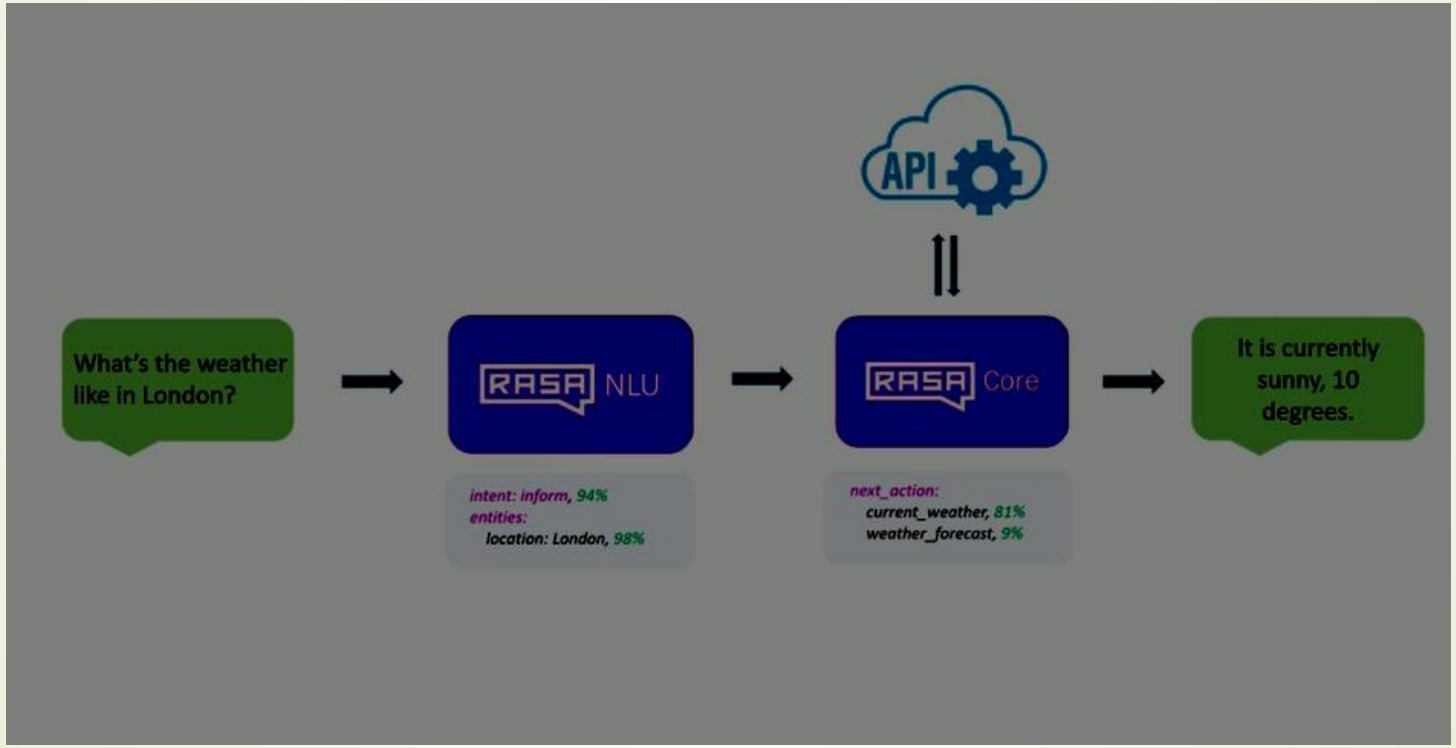
RASA: Conversational AI Framework

RASA Fwk


API
What's the weather like in London?
RASA NLU
RASA Core
It is currently sunny, 10 degrees.
intent: inform, 94%
entities:
location: London, 98%
next_action:
current_weather, 81%
weather_forecast, 9%

Rasa : introduction

- Rasa is an open source Conversational AI framework
- You don't need to worry about putting your data in someone else's cloud as in Dialogflow, Microsoft LUIS
- Rasa has two main components—Rasa NLU and Rasa Core.
- NLU stands *Natural Language Understanding*.

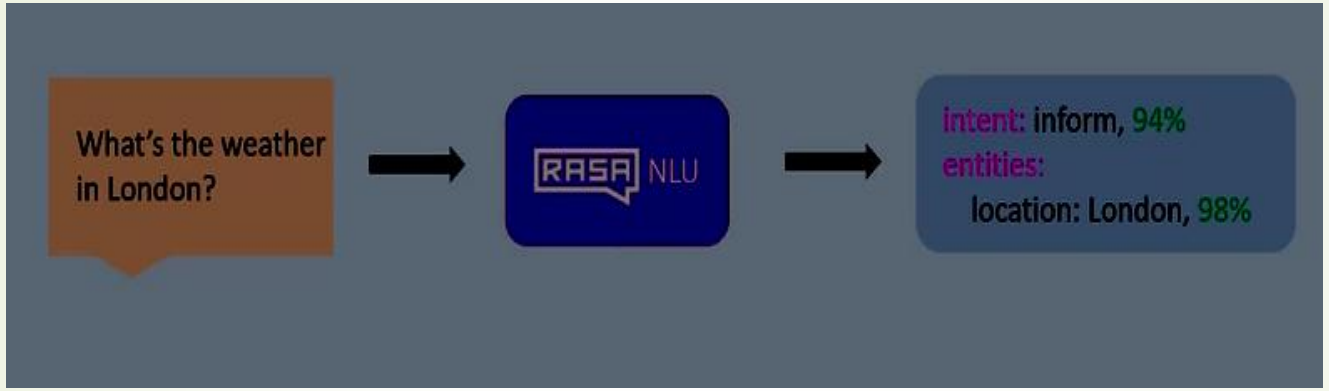
NLU

- Suppose the user says "I want to order a book"
- NLU's job is to take this input, understand the *intent* of the user and find the *entities* in the input.
- For example, in the above sentence, the intent is *ordering* and the entity is *book*.
- Rasa NLU internally uses Bag-of-Word (BoW) algorithm to find intent and Conditional Random Field (CRF) to find entities
- Although you can use other algorithms for finding intent and entities using Rasa. You have to create a custom pipeline to do that.

config_spacy.json

- {
- "pipeline":"spacy_sklearn",
- "path":"./models/nlu",
- "data":"./data/data.json"
- }
- Pipeline- we use sk learn
-

Trained NLU
converts utterance to intents



Data.json file

```
"rasa_nlu_data":{
    "common examples":[
    {
      "text":"Hello",
      "intent":"greet"
      "entities":[]
    },
    {
       "text":"goodbye",
       "intent":"goodbye",
       "entities":[]
    }
    ]
  }
}
```

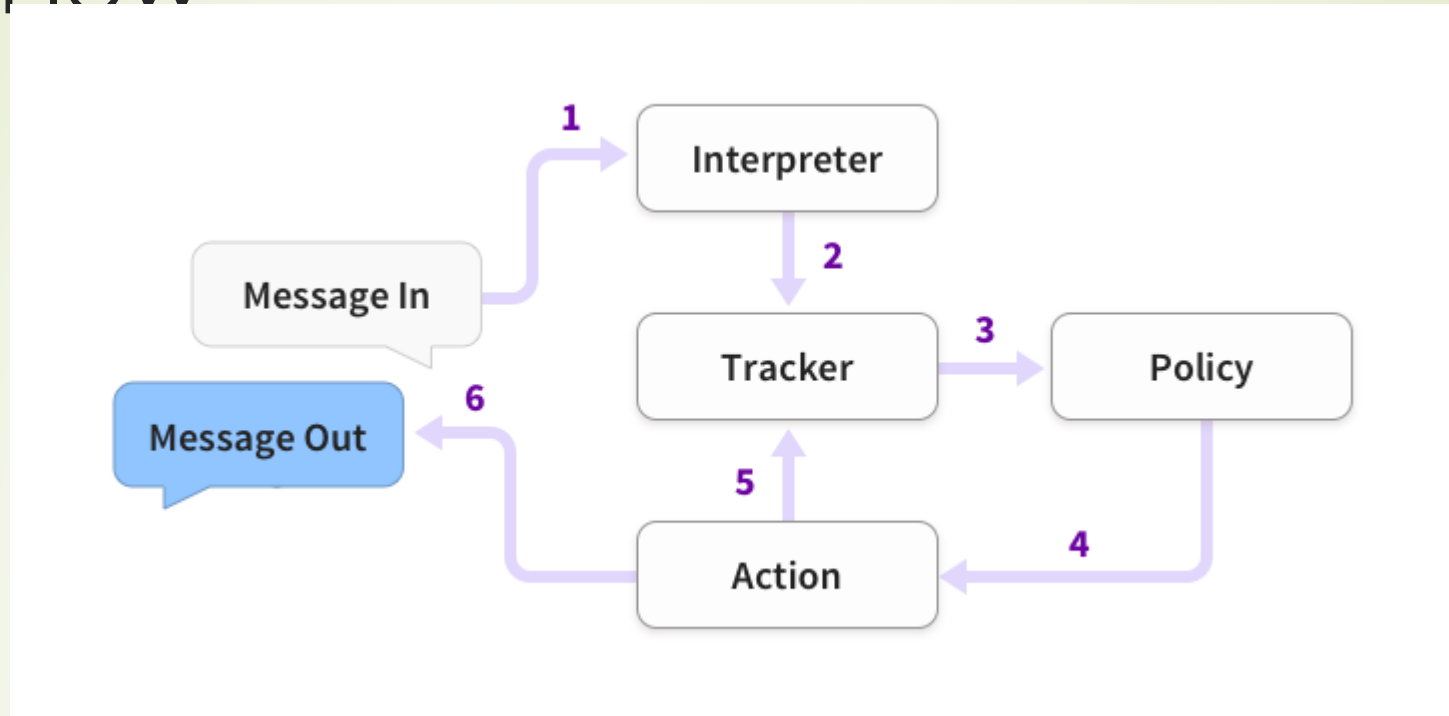
Training NLU

- def train_nlu(data, configs, model_dir):
- training_data = load_data(data)
- trainer = Trainer(config.load(configs))
- trainer.train(training_data)
- model_directory = trainer.persist(model_dir, fixed_model_name = 'weathernlu')
- print(interpreter.parse(u"Me flying to banaglore today. is the weather too hot there."))

RASA core

- The job of Rasa Core is to essentially generate the reply message for the chatbot.
- It takes the output of Rasa NLU (*intent* and *entities*) and applies Machine Learning models to generate a reply.
- Lets look at the next slide in more details .

Message Flow



As seen in the above diagram, the input message is interpreted by an Interpreter to extract intent and entity.

It is then passed to the Trackerthat keeps track of the current state of the conversation.

The Policy applies Machine Learning algorithm to determine what should be the reply and choses Action accordingly.

Action updates the Tracker to reflect the current state.

Domain

- slots: Like placeholders to hold context of conversation
- intents: Same intents as defined in nlu model
- entities: Same as nlu. Pairs with slots and provides values to it.
- Templates: Diff response scenarios are defined here.
- actions:various actions that bot can perform

actions.py

- Import rasa.core.actions import Action
- This is for all your custom action definitions
- Give a name to your action
- Define run: actual implementaion of the actions
  - Enterprise APIs – Authenticate first
  - Enterprise APIs – Execute the API then

Dialogue management: Rasa Core

- Dialogue management is the job of Rasa Core. Before we build the dialogue model, we need to define how we want the conversation to flow. Essentially we are creating a set of training examples for the dialogue model. We will create a file stories.md and put it in the data folder
- Example stories md
- ## story_001
  * greet
  - utter_greet
  * order_product
  - utter_ask_product_name
  * order_product[router=829]
  - slot{"router": "829"}
  - action_order_product
  * goodbye
  - utter_goodbye

Training the dialogue model

To train the dialogue model, we will write a function train_dialogue. The function needs 3 parameters—domain file, stories file and a path where you want to save your dialogue model after training

Training dialogue model

```
def train_dialogue(domain_file = 'weather_domain.yml',
                                    model_path = './models/dialogue',
                                    training_data_file = './data/stories.md'):
agent = Agent(domain_file, policies = [MemoizationPolicy(),
KerasPolicy(max_history=3, epochs=200, batch_size=50)])
        data = agent.load_data(training_data_file)


        agent.train(data)

        agent.persist(model_path)
        return agent
```

Running the bot

```
def run_weather_bot(serve_forever=True):
        interpreter = RasaNLUInterpreter('./models/nlu/default/weathernlu')
        action_endpoint = EndpointConfig(url="http://localhost:5055/webhook")
        agent = Agent.load('./models/dialogue', interpreter=interpreter,
    action_endpoint=action_endpoint)
        rasa_core.run.serve_application(agent ,channel='cmdline')


        return agent
```

Running the bot

- Create the nlu model
- Python nlu_model.py
- Create the dialogue model
- Python dialogue_management_model.py
- Enable running the bot.
- Now you can test the bot