Thesis: Building a Chatbot with Rasa using Python

Introduction

Chatbots are revolutionizing human-computer interaction, enhancing customer support, healthcare services, and automation. Rasa, an open-source framework, enables developers to build AI-powered conversational agents with Natural Language Understanding (NLU) and dialogue management. This thesis explores the **development of a chatbot using Rasa and Python**, covering installation, training models, creating APIs, and implementing a custom action using `actions.py`.

1. Installing and Setting Up Rasa

To start with Rasa, install the required packages:

```
pip install rasa
pip install rasa-sdk
```

After installation, initialize a new Rasa project:

```
rasa init --no-prompt
```

This will create a project structure with essential files.

2. Understanding Rasa Project Structure

- **data/nlu.yml** - Contains training examples for user inputs.
- **data/stories.yml** - Defines conversation flows.
- **domain.yml** - Lists intents, responses, actions, and entities.
- **config.yml** - Specifies machine learning models.
- **actions.py** - Implements custom actions using Python code.
- **3. Creating Intents and Training Data**
  - **Modify `data/nlu.yml` to define user intents:**

nlu:

- intent: greet

  examples: |

    - Hello

    - Hi

    - Hey

- intent: goodbye

  examples: |

    - Bye

    - See you later

Define conversation paths in data/stories.yml:

stories:

- story: greeting_story

  steps:

  - intent: greet

  - action: utter_greet

- story: goodbye_story

  steps:

  - intent: goodbye

  - action: utter_goodbye

Define responses in domain.yml:

responses:

  utter_greet:

  - text: "Hello! How can I help you?"

  utter_goodbye:

  - text: "Goodbye! Have a great day!"

4. Implementing a Custom Action in actions.py

For dynamic responses, create a Python action in actions.py:

python

from rasa_sdk import Action

from rasa_sdk.events import SlotSet

```
class ActionProvideInfo(Action):
    def name(self):
        return "action_provide_info"

    def run(self, dispatcher, tracker, domain):
        user_query = tracker.latest_message['text']
        response = f"You asked about: {user_query}. I'll fetch the details."
        dispatcher.utter_message(text=response)
        return []
```

Register this action in domain.yml:

```
actions:
  - action_provide_info
```

Enable custom actions by running:

```
rasa run actions
```

5. Training and Running the Chatbot

Train the model:

bash

CopyEdit

```
rasa train
```

Run the bot:

bash

CopyEdit

```
rasa shell
```

---

6. Exposing the Chatbot as an API

To interact with the chatbot via API, start the Rasa server:

```
rasa run --enable-api --cors "*"
```

Send requests using Python:

python

```
import requests


response = requests.post(
   "http://localhost:5005/webhooks/rest/webhook",
   json={"sender": "user", "message": "Hello"}
)


print(response.json())
```

Conclusion

This thesis provides a structured approach to **building a chatbot with Rasa and Python**, covering **intent classification, dialogue flow, custom actions, and API integration**. Rasa's flexible framework makes it suitable for **AI-driven automation and enterprise solutions**.