**Module 20**

Let's say you are a dentist, and you would like to build a chatbot using RASA so that the customers can ask simple queries like (address, timings, book appointments)

­­ **Virtual Environment and Rasa Shell**

(python\_10) C:\Users\Lenovo>python -m venv rasa\_env

(python\_10) C:\Users\Lenovo>rasa\_env\Scripts\activate

(rasa\_env) (python\_10) C:\Users\Lenovo>pip install rasa

(rasa\_env) (python\_10) C:\Users\Lenovo>cd C:\Users\Lenovo\Downloads\Study material\AI\ChatBots Using RASA and BERT\Assignment\dentist\_bot

(rasa\_env) (python\_10) C:\Users\Lenovo\Downloads\Study material\AI\ChatBots Using RASA and BERT\Assignment\dentist\_bot>rasa train

2024-05-21 21:13:45 INFO rasa.engine.training.hooks - Finished training component 'RulePolicy'.

2024-05-21 21:13:45 INFO rasa.engine.training.hooks - Starting to train component 'TEDPolicy'.

Processed trackers: 100%|██████████████████████████████████████████████████| 4/4 [00:00<00:00, 255.35it/s, # action=16]

Epochs: 100%|█████████████████████████████████████| 100/100 [00:22<00:00, 4.48it/s, t\_loss=28.4, loss=28.3, acc=0.875]

2024-05-21 21:14:08 INFO rasa.engine.training.hooks - Finished training component 'TEDPolicy'.

Your Rasa model is trained and saved at 'models\20240521-211319-vicious-owl.tar.gz'.

(rasa\_env) (python\_10) C:\Users\Lenovo\Downloads\Study material\AI\ChatBots Using RASA and BERT\Assignment\dentist\_bot>rasa run --enable-api

(rasa\_env) (python\_10) C:\Users\Lenovo\Downloads\Study material\AI\ChatBots Using RASA and BERT\Assignment\dentist\_bot>rasa shell

**Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Tue May 21 21:10:10 2024

@author: Lenovo

"""

import os

# Directory and file structure for RASA chatbot project

# os.makedirs('dentist\_bot', exist\_ok=True)

# os.chdir('dentist\_bot')

# Writing configuration files

config\_yml = """

language: en

pipeline:

- name: WhitespaceTokenizer

- name: RegexFeaturizer

- name: LexicalSyntacticFeaturizer

- name: CountVectorsFeaturizer

- name: DIETClassifier

epochs: 100

- name: EntitySynonymMapper

- name: ResponseSelector

epochs: 100

- name: FallbackClassifier

threshold: 0.3

ambiguity\_threshold: 0.1

policies:

- name: MemoizationPolicy

- name: TEDPolicy

max\_history: 5

epochs: 100

- name: RulePolicy

"""

domain\_yml = """

intents:

- greet

- ask\_address

- ask\_timings

- book\_appointment

- affirm

- deny

entities:

- time

- date

slots:

time:

type: text

mappings:

- type: from\_entity

entity: time

date:

type: text

mappings:

- type: from\_entity

entity: date

responses:

utter\_greet:

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

utter\_address:

- text: "Our clinic is located at 123 Dental Street."

utter\_timings:

- text: "We are open from 9 AM to 5 PM, Monday to Saturday."

utter\_ask\_date:

- text: "When would you like to book the appointment?"

utter\_ask\_time:

- text: "At what time would you prefer?"

utter\_confirm\_appointment:

- text: "Your appointment is booked for {date} at {time}."

utter\_fallback:

- text: "I'm sorry, I didn't understand that. Can you please rephrase?"

actions:

- action\_book\_appointment

session\_config:

session\_expiration\_time: 60

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

"""

data\_yml = """

version: "2.0"

nlu:

- intent: greet

examples: |

- Hi

- Hello

- Hey there

- intent: ask\_address

examples: |

- Where is the clinic located?

- What is the address?

- intent: ask\_timings

examples: |

- What are your working hours?

- When are you open?

- intent: book\_appointment

examples: |

- I want to book an appointment

- Can I schedule a visit?

- intent: affirm

examples: |

- Yes

- Sure

- intent: deny

examples: |

- No

- Not now

stories:

- story: book an appointment

steps:

- intent: book\_appointment

- action: utter\_ask\_date

- intent: inform

entities:

- date: "2023-12-01"

- action: utter\_ask\_time

- intent: inform

entities:

- time: "10:00 AM"

- action: action\_book\_appointment

- action: utter\_confirm\_appointment

rules:

- rule: greet the user

steps:

- intent: greet

- action: utter\_greet

- rule: provide address

steps:

- intent: ask\_address

- action: utter\_address

- rule: provide timings

steps:

- intent: ask\_timings

- action: utter\_timings

- rule: fallback

steps:

- intent: nlu\_fallback

- action: utter\_fallback

"""

action\_py = """

from typing import Any, Text, Dict, List

from rasa\_sdk import Action, Tracker

from rasa\_sdk.executor import CollectingDispatcher

class ActionBookAppointment(Action):

def name(self) -> Text:

return "action\_book\_appointment"

def run(self, dispatcher: CollectingDispatcher,

tracker: Tracker,

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

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

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

dispatcher.utter\_message(text=f"Your appointment is booked for {date} at {time}.")

return []

"""

# Writing files to the project directory

with open('config.yml', 'w') as file:

file.write(config\_yml)

with open('domain.yml', 'w') as file:

file.write(domain\_yml)

with open('data/nlu.yml', 'w') as file:

file.write(data\_yml)

os.makedirs('actions', exist\_ok=True)

with open('actions/actions.py', 'w') as file:

file.write(action\_py)

print("RASA chatbot project structure created successfully.")

**Output:**

