Name : Aryan Deepak Saraf
Div : D20B
Roll No. : 49

AI&DS2 Experiment 04

Aim : To build an adaptive and contextual Cognitive based Customer service application / Insurance / Healthcare Application / Smarter Cities / Government etc.

Theory :

Introduction

Cognitive-based applications are intelligent systems that simulate human-like understanding and decision-making. They use Artificial Intelligence (AI), Machine Learning (ML), and contextual reasoning to process data and provide meaningful insights. Unlike traditional applications, which work only on predefined rules, cognitive applications can adapt to user needs and respond based on context.

In real life, customer service, insurance, healthcare, smarter cities, and government services all require systems that are not only intelligent but also personalized and situation-aware. Adaptive and contextual applications make interactions more natural and effective.

Key Concepts

1. Cognitive Computing
    - Combines AI, ML, Natural Language Processing (NLP), and reasoning.
    - Learns from past data and improves over time.
    - Simulates human-like problem solving.

2. Adaptivity
    - The ability of the system to learn and remember user preferences, actions, or history.
    - Example: remembering which plant the user is growing and when it was last watered.

3. Contextual Intelligence
    - The ability of the system to understand the situation and provide different responses depending on the context.
    - Example: suggesting watering only if it has been several days since the last watering.

4. Knowledge Representation and Reasoning
    - Stores user actions and plant care history.
    - Applies logical reasoning to decide what advice to give in the current context.

Implementation in Smart Gardening Assistant

In this experiment, we extended our Smart Gardening Assistant to make it both adaptive and contextual.

- Multiple Plant Support:
  The system can now manage more than one plant at the same time. Each plant has its own record for watering and fertilizing history.

- Adaptive Learning:
  The assistant remembers details such as:
  - Plant names added by the user.
  - Last watering date for each plant.
  - Last fertilizing date for each plant.

- Contextual Advice:
  Based on the number of days passed since the last watering or fertilizing, the assistant provides different advice.
  - If watering was done recently, it says *"No need today."*
  - If watering is overdue, it suggests *"Time to water your plant."*
  - If fertilizer has not been given for more than 15 days, it recommends fertilization.

- Personalized Status Check:
  The user can check the status of any plant (e.g., Rose, Tulip, etc.) and get advice that is specific to that plant's care history.

Applications

1. Customer Service
   - Personalized responses based on user history and context.
   - Example: remembering past complaints or product details.

2. Insurance
   - Adaptive claim processing by analyzing user's case history.
   - Contextual decision-making in fraud detection or claim approval.

3. Healthcare
   - Patient-specific reminders for medicine, diet, or appointments.
   - Context-aware recommendations based on health condition.

4. Smarter Cities
   - Adaptive traffic systems that respond to real-time conditions.
   - Contextual waste management based on locality data.

5. Government Services
   - Personalized citizen services based on profile and past interactions.
   - Contextual verification in identity and security systems.

6. Smart Gardening (Our Implementation)
   - Tracks multiple plants and adapts to each plant's history.

- Provides contextual gardening tips based on time and previous care.

Advantages

- Personalization: Every user or plant gets unique recommendations.
- Automation: Reduces manual effort in remembering plant schedules.
- Efficiency: Context-aware responses save time and improve accuracy.
- Scalability: Can easily be extended for more plants, more users, or even connected with IoT sensors.

Code :

```
import datetime
```

Step 1: Initialize Plant Profiles (Dictionary for multiple plants)

```
plant_profiles = {}
```

Step 2: Function to update plant details (Adaptive learning)

```
def update_plant_profile(action, plant_name):
    today = datetime.date.today()
    if plant_name not in plant_profiles:
        plant_profiles[plant_name] = {"last_watered": None, "last_fertilized":
None}

    if action == "water":
        plant_profiles[plant_name]["last_watered"] = today
    elif action == "fertilize":
        plant_profiles[plant_name]["last_fertilized"] = today
```

Step 3: Contextual Advice based on history

```
def gardening_advice(plant_name):
    today = datetime.date.today()
    response = [f"Status for {plant_name}:"]

    if plant_name not in plant_profiles:
        return f"No data available for {plant_name}. Please record some actions
first."

    profile = plant_profiles[plant_name]

    # Context: check watering
    if profile["last_watered"]:
        days_since_water = (today - profile["last_watered"]).days
        if days_since_water >= 3:
```

```
            response.append(f"It's been {days_since_water} days since watering.
Time to water {plant_name}.")
        else:
            response.append(f"{plant_name} was watered {days_since_water} days
ago. No need today.")
    else:
        response.append(f"I don't know when {plant_name} was last watered.
Please update me.")

    # Context: check fertilizer
    if profile["last_fertilized"]:
        days_since_fert = (today - profile["last_fertilized"]).days
        if days_since_fert >= 15:
            response.append(f"It's been {days_since_fert} days since
fertilizing. Add organic fertilizer today.")
        else:
            response.append(f"Fertilizer was added {days_since_fert} days ago.
Wait before the next dose.")
    else:
        response.append(f"I don't know when {plant_name} was last fertilized.
Please update me.")

    return "\n".join(response)
```

Step 4: Chat simulation

```
print("Adaptive & Contextual Smart Gardening Assistant (Multiple Plant
Support)")
print("Commands: 'add plant <name>', '<plant> watered', '<plant> fertilized',
'status <plant>', 'exit'\n")

while True:
    user_input = input("You: ").lower()

    if user_input == "exit":
        print("Assistant: Goodbye! Happy Gardening")
        break

    elif user_input.startswith("add plant"):
        plant = user_input.split("add plant")[-1].strip()
        plant_profiles[plant] = {"last_watered": None, "last_fertilized": None}
        print(f"Assistant: Got it! I will remember your plant {plant}.")

    elif "watered" in user_input:
        plant = user_input.replace("watered", "").strip()
        update_plant_profile("water", plant)
        print(f"Assistant: Recorded watering for {plant} today.")
```

```
    elif "fertilized" in user_input:
        plant = user_input.replace("fertilized", "").strip()
        update_plant_profile("fertilize", plant)
        print(f"Assistant: Recorded fertilizing for {plant} today.")

    elif user_input.startswith("status"):
        plant = user_input.replace("status", "").strip()
        print("Assistant:\n", gardening_advice(plant))

    else:
        print("Assistant: Sorry, I didn’t understand. Try 'status Rose', 'Rose watered', 'Rose fertilized'.")
```

```
Adaptive & Contextual Smart Gardening Assistant (Multiple Plant Support)
Commands: 'add plant <name>', '<plant> watered', '<plant> fertilized', 'status <plant>', 'exit'

You: status
Assistant:
 No data available for . Please record some actions first.
You: add plant Rose
Assistant: Got it! I will remember your plant rose.
You: add plant Tulip
Assistant: Got it! I will remember your plant tulip.
You: Rose watered
Assistant: Recorded watering for rose today.
You: status Rose
Assistant:
 Status for rose:
rose was watered 0 days ago. No need today.
I don't know when rose was last fertilized. Please update me.
You: status Tulip
Assistant:
 Status for tulip:
I don't know when tulip was last watered. Please update me.
I don't know when tulip was last fertilized. Please update me.
You: Tulip fertilized
Assistant: Recorded fertilizing for tulip today.
You: status Tulip
Assistant:
 Status for tulip:
I don't know when tulip was last watered. Please update me.
Fertilizer was added 0 days ago. Wait before the next dose.
You: Tulip watered
Assistant: Recorded watering for tulip today.
You: status
Assistant:
 No data available for . Please record some actions first.
You: status Tulip
Assistant:
 Status for tulip:
tulip was watered 0 days ago. No need today.
Fertilizer was added 0 days ago. Wait before the next dose.
You: exit
Assistant: Goodbye! Happy Gardening
```

```
plant_profiles
```

```
{'rose': {'last_watered': datetime.date(2025, 8, 31), 'last_fertilized': None},
 'tulip': {'last_watered': datetime.date(2025, 8, 31),
  'last_fertilized': datetime.date(2025, 8, 31)}}
```

Conclusion : In this experiment, we extended our Smart Gardening Assistant into an adaptive and contextual system that can manage multiple plants, remember watering and fertilizing schedules, and provide personalized advice. This approach demonstrates how cognitive systems can be applied not only in gardening but also across domains like customer service, healthcare, insurance, smarter cities, and government.

49_Prac_04