សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Machine Learning Integration in Intelligence Systems

By: SEK SOCHEAT

Advisor to DCS and Lecturer

Mobile: 017 879 967

Email: socheatsek@norton-u.com

socheat.sek@gmail.com

សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Intelligence System Development

2024 – 2025

Y4E1 – DCS – NU

Table of Contents:

Machine Learning Integration in Intelligence Systems

1. Integrating ML with IS for Knowledge Enhancement

Definition

- An Intelligence system is a computer program that replicates (clone) the decision-making ability of a human Intelligence within a specific domain, providing reliable solutions based on domain-specific knowledge.

Components

- Key components include a **knowledge base** for storing facts and rules, an **inference engine** to deduce conclusions, and a **user interface** for user interaction.

Applications

- Intelligence systems are used in fields like medicine (e.g., MYCIN for diagnosis), engineering (design optimization), and business (decision support).

Advantages

- They provide consistent, unbiased performance, make Intelligence knowledge accessible, and are cost-effective for repetitive tasks.

Challenges

- Challenges include complex knowledge acquisition, limited applicability to specific domains, and the need for regular updates.

1.2. Introduction to Machine Learning

Definition

- Machine Learning (ML) is a branch of artificial intelligence that enables systems to learn and improve from data without explicit programming, allowing them to adapt and make data-driven predictions or decisions.

Components

- Key components of ML include **datasets** for training, **algorithms** for learning patterns (e.g., decision trees, neural networks), and **models** that generalize learned patterns for prediction.

Applications

- ML is used in various fields such as healthcare (predicting diseases), finance (fraud detection), and technology (speech recognition and recommendation systems).

Advantages

- ML enables automation, handles complex and large datasets efficiently, and improves accuracy by continuously learning from new data.

Challenges

- Challenges include data dependency, high computational requirements, potential bias in training data, and difficulties in interpreting complex models.

1.3. Integration of Machine Learning with IS

Definition:

- Combining ML with IS to enhance decision-making by enabling adaptive and data-driven improvements.

Purpose:

- Overcomes static rule limitations in ES by incorporating ML's ability to learn from data and update knowledge dynamically.

Approaches:

- ***Knowledge Enhancement:*** ML identifies patterns to expand the Intelligence system's rule base.
- ***Decision Support:*** ML refines ES recommendations using predictive models.

Applications:

- Adaptive medical diagnosis (learning new disease patterns).
- Automated customer support with dynamic query handling.
- Real-time fault detection in engineering systems.

Challenges:

- Complexity in integrating dynamic learning into static rule-based structures.
- Ensuring explain-ability and transparency of ML-enhanced decisions.

Scenario: IT Helpdesk Chatbot

- **Intelligence System:** Handles routine queries like password resets.
- **Machine Learning:** Predicts customer sentiment and recommends personalized responses.

Code Implementation:

1. Import Libraries
2. Intelligence System for Fixed Queries
3. Machine Learning for Sentiment Analysis
4. Integration of Intelligence System and ML
5. Example Interaction

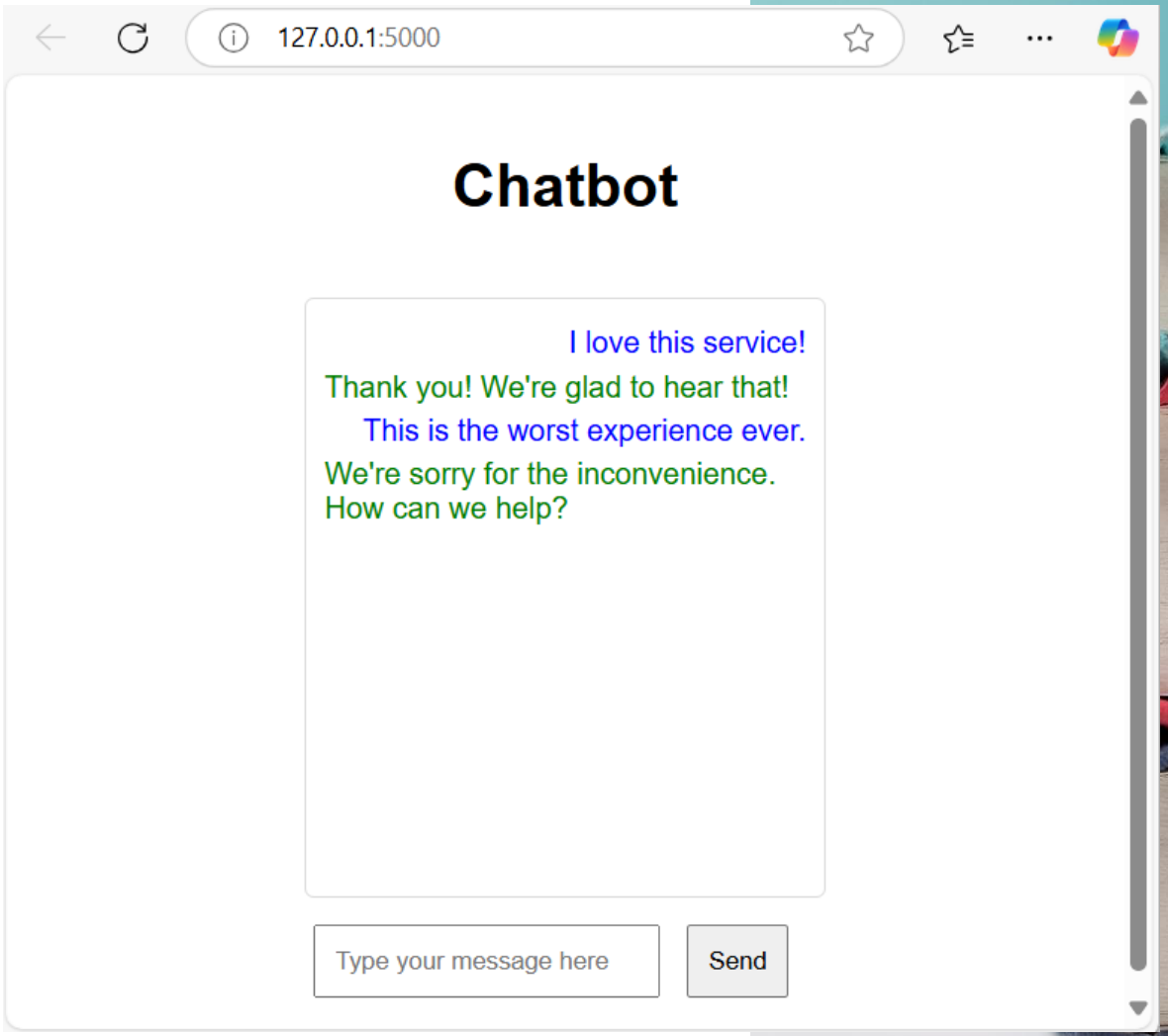
Output

Scenario: IT Helpdesk Chatbot

Required Libraries and Installation Commands

You can install all dependencies in one command:

pip install flask flask-cors scikit-learn



"Sentiment-Aware IT Helpdesk Chatbot for Personalized Customer Support"

This chatbot specializes in providing intelligent, sentiment-driven assistance for IT helpdesk scenarios. It combines predefined Intelligence rules with machine learning to resolve common queries, analyze user sentiment, and deliver adaptive, empathetic responses tailored to individual user needs.

Scenario: IT Helpdesk Chatbot


```python
from flask import Flask, request, jsonify, render_template
from flask_cors import CORS
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

app = Flask(__name__)
CORS(app)  # Enable CORS for cross-origin requests

# Training data
data = [
    ("I am very happy with your service!", "positive"),
    ("This is the worst experience ever.", "negative"),
    ("I love how easy it is to use this product.", "positive"),
    ("I am disappointed with the results.", "negative"),
]
texts, labels = zip(*data)
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)
model = MultinomialNB()
model.fit(X, labels)

@app.route("/", methods=["GET"])
def index():
    return render_template("chat.html")

@app.route("/chat", methods=["POST"])
def chat():
    user_input = request.json.get("message")
    if not user_input:
        return jsonify({"error": "No message provided!"}), 400

    user_vector = vectorizer.transform([user_input])
    sentiment = model.predict(user_vector)[0]
    response = "Thank you! We're glad to hear that!" if sentiment == "positive" else "We're sorry for the inconvenience. How can we help?"
    return jsonify({"response": response})

if __name__ == "__main__":
    app.run(debug=True)
```

Python: app.py

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chatbot</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        .chat-container {
            width: 50%;
            max-width: 600px;
            margin-top: 20px;
        }
        .messages {
            border: 1px solid #ddd;
            border-radius: 5px;
            height: 300px;
            overflow-y: auto;
            padding: 10px;
            margin-bottom: 10px;
        }
        .message {
            margin: 5px 0;
        }
        .message.user {
            text-align: right;
            color: blue;
        }
        .message.bot {
            text-align: left;
            color: green;
        }
        input, button {
            padding: 10px;
            margin: 5px;
        }
    </style>
</head>
```

templates/chat.html

```
46
47 <body>
48     <h1>Chatbot</h1>
49     <div class="chat-container">
50         <div id="messages" class="messages"></div>
51         <input type="text" id="userInput" placeholder="Type your message here">
52         <button onclick="sendMessage()">Send</button>
53     </div>
54     <script>
55         function sendMessage() {
56             const userInput = document.getElementById("userInput");
57             const messagesDiv = document.getElementById("messages");
58             const message = userInput.value;
59
60             if (!message.trim()) return;
61
62             // Display user message
63             const userMessage = document.createElement("div");
64             userMessage.className = "message user";
65             userMessage.textContent = message;
66             messagesDiv.appendChild(userMessage);
67
68             // Send message to server
69             fetch("/chat", {
70                 method: "POST",
71                 headers: {
72                     "Content-Type": "application/json"
73                 },
74                 body: JSON.stringify({ message: message })
75             })
76             .then(response => response.json())
77             .then(data => {
78                 const botMessage = document.createElement("div");
79                 botMessage.className = "message bot";
80                 botMessage.textContent = data.response;
81                 messagesDiv.appendChild(botMessage);
82                 messagesDiv.scrollTop = messagesDiv.scrollHeight;
83                 userInput.value = "";
84             })
85             .catch(error => console.error("Error:", error));
86         }
87     </script>
88 </body>
89 </html>
```

templates/chat.html

2. Reinforcement Learning and Adaptive IS

- **Reinforcement Learning (RL)**

  Reinforcement Learning is a machine learning approach where an agent learns by interacting with its environment, receiving rewards for beneficial actions, and penalties for unfavorable ones, aiming to maximize long-term rewards.

- **Adaptive Intelligence Systems**

  Adaptive Intelligence Systems extend traditional rule-based systems by incorporating learning capabilities, allowing them to dynamically update rules and improve decision-making based on new data or feedback.

2.1. Understanding Reinforcement Learning (RL)

Definition

- Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties, guiding it to achieve a long-term objective.

Key Components:

- ***Agent:*** The decision-maker (e.g., a chatbot, robot, or software system).
- ***Environment:*** The system or space in which the agent operates.
- ***State:*** The current situation or context perceived by the agent.
- ***Action:*** A decision or move made by the agent to change the state.
- ***Reward:*** Feedback indicating the value of an action in achieving the goal.

2. Reinforcement Learning and Adaptive IS

How It Works:

- The agent observes the environment and selects an action.
- After performing the action, the environment provides feedback (reward or penalty).
- The agent updates its strategy (policy) to maximize cumulative rewards over time.

Example Applications:

- ***Game Playing:*** Training AI to play chess or Go (e.g., AlphaGo).
- ***Robotics:*** Teaching robots to walk or perform tasks like assembly.
- ***Recommendation Systems:*** Improving suggestions by learning user preferences dynamically.

2. Reinforcement Learning and Adaptive IS

Benefits:

- Learns from trial and error without requiring labeled data.
- Adapts dynamically to changes in the environment.
- Optimizes long-term rewards rather than immediate gains.

Challenges:

- Requires significant computational resources for complex tasks.
- Balancing exploration (trying new actions) and exploitation (using known strategies).
- May struggle in environments with delayed rewards.

2.2. Adaptive Intelligence Systems

Definition:

- Adaptive Intelligence Systems are intelligent systems that modify their knowledge, rules, or behavior based on changes in the environment, feedback, or user interactions. Unlike traditional Intelligence systems, they evolve dynamically to improve performance and accuracy over time.

Key Features:

- ***Dynamic Rule Adjustment:*** Updates or refines rules based on new data or user feedback.
- ***Learning Capability:*** Incorporates machine learning techniques like reinforcement learning to adapt decision-making.
- ***Self-Optimization:*** Adjusts system parameters to enhance performance for specific tasks.

2.2. Adaptive Intelligence Systems

How It Works:

- The system observes the environment or user interactions.
- It learns patterns or updates rules based on this data.
- Applies new knowledge to improve decision-making or recommendations dynamically.

Example Applications:

- ***Healthcare:*** Adapts treatment recommendations based on patient feedback and new medical research.
- ***Customer Support:*** Learns from unresolved queries to improve future responses.
- ***Autonomous Vehicles:*** Adjusts driving strategies based on road conditions and traffic patterns.

2.2. Adaptive Intelligence Systems

Benefits:

- Handles evolving problems and environments effectively.
- Reduces reliance on manual updates to knowledge bases.
- Increases user satisfaction by improving responses over time.

Challenges:

- ***Complexity:*** Requires advanced algorithms and sufficient computational resources.
- ***Transparency:*** Adaptations can make decisions harder to interpret.
- ***Data Dependency:*** Needs continuous access to quality data for effective learning.

2.3. Integration of RL in Intelligence Systems

Definition:

Integrating Reinforcement Learning (RL) into Intelligence Systems combines rule-based reasoning with the adaptability of RL, enabling the system to learn from interactions and improve decision-making dynamically. This integration enhances the traditional static nature of Intelligence systems by introducing learning and adaptation capabilities.

How It Works:

- ***State Representation:*** The Intelligence system represents the current problem or scenario as a state.
- ***Action Selection:*** RL determines the best action to take, either by exploring new strategies or exploiting learned ones.
- ***Reward Feedback:*** After performing an action, the system evaluates its effectiveness and assigns a reward or penalty.
- ***Policy Update:*** RL updates its policy (decision-making model) based on the reward to improve future actions.

2.3. Integration of RL in Intelligence Systems

Example Workflow:

1. Healthcare System:

- ***Scenario:*** A patient management Intelligence system uses RL to optimize treatment plans.
- ***State:*** Patient symptoms and history.
- ***Action:*** Recommending a treatment or diagnostic test.
- ***Reward:*** Positive if the treatment is effective, negative if it fails.

2. Customer Support:

- ***Scenario:*** An Intelligence system for customer service learns to resolve complex queries.
- ***State:*** User's issue and sentiment.
- ***Action:*** Provide a resolution or escalate the issue.
- ***Reward:*** Positive if the user is satisfied, negative if unresolved.

2.3. Integration of RL in Intelligence Systems

Benefits of Integration:

- ***Dynamic Learning:*** Adapts to new scenarios without manual updates to rules.
- ***Optimized Decision-Making:*** Learns the best actions over time to maximize long-term success.
- ***Resilience in Dynamic Environments:*** Thrives in uncertain or evolving domains where predefined rules alone may fall short.

Challenges:

- ***Exploration vs. Exploitation:*** Balancing the need to explore new strategies with leveraging proven ones.
- ***Complexity:*** Requires computational resources and effective state-action-reward modeling.
- ***Reward Design:*** Defining meaningful rewards to guide learning effectively.

2.4. Practical Activity

An example of an Adaptive Intelligence System using Flask with a simple Reinforcement Learning (RL) simulation. The system acts as a customer support chatbot that uses RL to optimize responses based on user feedback.

How It Works:

1. **States**: Represent user issues (e.g., "late_delivery", "wrong_item").
2. **Actions**: Possible responses (e.g., "apologize", "offer_discount", "escalate").
3. **Reinforcement Learning**: Updates the Q-table based on user feedback:
    - Positive feedback (+1) increases the Q-value for the selected action.
    - Negative feedback (-1) decreases the Q-value, encouraging exploration of other actions.

2.4. Practical Activity

1. Defining States and Actions

```
states = ["late_delivery", "wrong_item", "technical_issue"]

actions = ["apologize", "offer_discount", "escalate"]
```

- **States:** Represent different situations or problems the agent (e.g., chatbot) might encounter.
  - **"late_delivery":** A customer reports that their delivery is delayed.
  - **"wrong_item":** A customer receives an incorrect item.
  - **"technical_issue":** A customer experiences a technical problem.

- **Actions:** Represent possible responses or decisions the agent can make to address a problem.
  - **"apologize":** The bot apologizes for the issue.
  - **"offer_discount":** The bot offers a discount as compensation.
  - **"escalate":** The bot escalates (increment) the issue to a human agent.

2.4. Practical Activity

2. Initializing the Q-Table

q_table = {state: {action: 0.0 for action in actions} for state in states}

- The Q-table stores Q-values, which estimate the "quality" of each action for a given state.
- **Structure:** The Q-table is a nested dictionary:
  - ***Outer dictionary:*** Keys are states ("late_delivery", "wrong_item", etc.).
  - ***Inner dictionary:*** Keys are actions ("apologize", "offer_discount", etc.), and values are initialized to 0.0.

2.4. Practical Activity

2. Initializing the Q-Table


```
{
    "late_delivery": {
        "apologize": 0.0,
        "offer_discount": 0.0,
        "escalate": 0.0
    },
    "wrong_item": {
        "apologize": 0.0,
        "offer_discount": 0.0,
        "escalate": 0.0
    },
    "technical_issue": {
        "apologize": 0.0,
        "offer_discount": 0.0,
        "escalate": 0.0
    }
}
```


Purpose:

- The Q-values represent the expected reward for taking a particular action in a specific state.
- Initially, all Q-values are 0.0 because the agent has no prior knowledge of the environment.

2.4. Practical Activity

3. Learning Parameters

learning_rate = 0.1

discount_factor = 0.9

exploration_rate = 0.1

• **Learning Rate (learning_rate)**: Determines how much recent feedback influences Q-value updates.

  • ***Low (0.1):*** Relies on past knowledge.

  • ***High (1.0):*** Prioritizes recent feedback.

• **Discount Factor (discount_factor)**: Balances immediate vs. future rewards.

  • ***Low (0.1):*** Focuses on immediate rewards.

  • ***High (0.9):*** Considers long-term rewards.

• **Exploration Rate (exploration_rate)**: Controls how often the agent tries random actions.

  • ***Example:*** With 0.1, the agent explores 10% of the time and exploits 90%.

2.4. Practical Activity

How It All Works Together:

- **Q-Table Initialization:** Provides a starting point where all actions have equal value (0.0).
- **States and Actions:** Defines the problem space the agent interacts with.
- **Learning Parameters:**

- ‘learning_rate’ adjusts how quickly the agent updates its Q-values based on feedback.

- ‘discount_factor’ ensures the agent considers long-term outcomes.

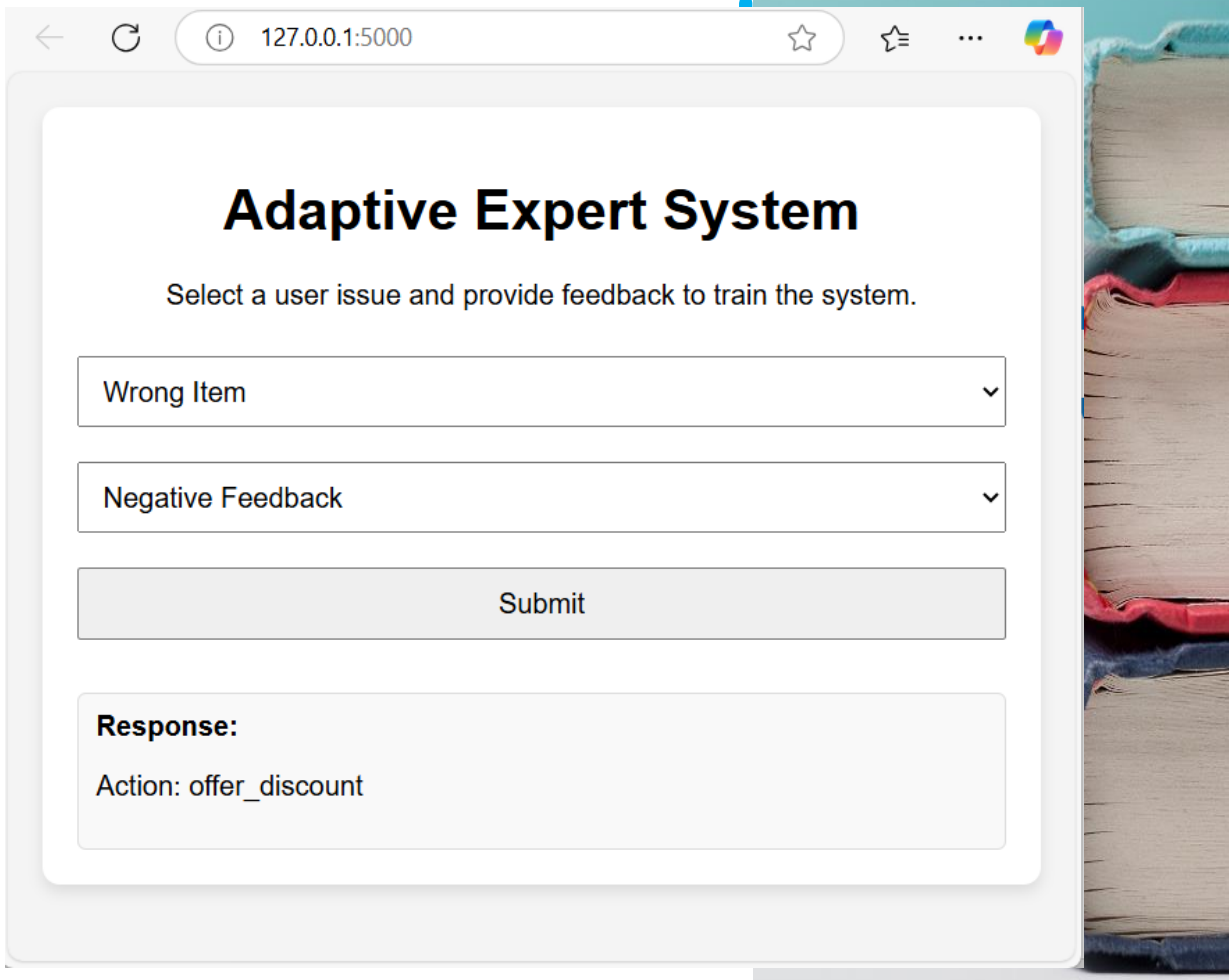
- ‘exploration_rate’ balances trying new actions (exploration) with leveraging learned strategies (exploitation).

```python
from flask import Flask, request, jsonify, render_template
import random

app = Flask(__name__)

# Q-learning setup (same as before)
states = ["late_delivery", "wrong_item", "technical_issue"]
actions = ["apologize", "offer_discount", "escalate"]
q_table = {state: {action: 0 for action in actions} for state in states}
learning_rate = 0.1
discount_factor = 0.9

@app.route("/", methods=["GET"])
def index():
    return render_template("index.html")

@app.route("/interact", methods=["POST"])
def interact():
    data = request.get_json()
    user_state = data.get("state")
    user_feedback = data.get("feedback")

    if user_state not in states or user_feedback not in [-1, 1]:
        return jsonify({"error": "Invalid state or feedback"}), 400

    # RL logic
    if random.uniform(0, 1) < 0.1:  # Exploration
        chosen_action = random.choice(actions)
    else:  # Exploitation
        chosen_action = max(q_table[user_state], key=q_table[user_state].get)

    reward = user_feedback
    current_q = q_table[user_state][chosen_action]
    max_future_q = max(q_table[user_state].values())
    q_table[user_state][chosen_action] = current_q + learning_rate *
    (reward + discount_factor * max_future_q - current_q)

    return jsonify({"response": f"Action: {chosen_action}"})

if __name__ == "__main__":
    app.run(debug=True)
```

Python:

adaptive_chatbot.py

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Adaptive Expert System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 0;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 600px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }
        input, select, button {
            padding: 10px;
            margin: 10px 0;
            width: 100%;
            box-sizing: border-box;
            font-size: 16px;
        }
        .output {
            margin-top: 20px;
            background: #f9f9f9;
            padding: 10px;
            border-radius: 5px;
            border: 1px solid #ddd;
            text-align: left;
        }
    </style>
</head>
```



templates/index.html

```
66
67      <script>
68          // Function to send feedback to the Flask server
69          function sendFeedback() {
70              const state = document.getElementById("state").value;
71              const feedback = parseInt(document.getElementById("feedback").value);
72              const responseBox = document.getElementById("response");
73
74              fetch("/interact", {
75                  method: "POST",
76                  headers: {
77                      "Content-Type": "application/json"
78                  },
79                  body: JSON.stringify({ state: state, feedback: feedback })
80              })
81              .then(response => response.json())
82              .then(data => {
83                  if (data.response) {
84                      responseBox.textContent = data.response;
85                  } else if (data.error) {
86                      responseBox.textContent = "Error: " + data.error;
87                  }
88              })
89              .catch(error => {
90                  responseBox.textContent = "Error connecting to server.";
91                  console.error(error);
92              });
93          }
94      </script>
95  </body>
96  </html>
97
```

templates/index.html

3. Homework

1. What are the core components of an Intelligence System, and how do they function together?

2. Describe the main types of Machine Learning and provide examples of their applications.

3. How can Machine Learning enhance an Intelligence System's decision-making capabilities? Provide an example of integration.

4. Explain the key components of Reinforcement Learning and their roles in the learning process.

5. What are Adaptive Intelligence Systems, and how do they differ from traditional Intelligence Systems?

Thank You

SEK SocheaT

socheatsek@norton-u.com