Designing a Self-Validating AI Agent

Preamble

AI agents deployed in production must ensure reliability and safety by avoiding hallucinations, errors, and misleading outputs. These issues can undermine trust and the effectiveness of AI solutions. Systems need to be designed to autonomously validate and improve their outputs, ensuring alignment with real-world requirements while reducing operational costs and improving efficiency.

This task challenges you to design such a validation strategy for an AI agent, ensuring it autonomously evaluates its outputs to maintain high accuracy and reliability.

Scenario

A logistics company has deployed an AI agent to generate summaries of delivery performance based on customer feedback. This agent processes customer reviews, identifies themes, and delivers concise summaries to managers. However, the agent currently has the following shortcomings:

- It may hallucinate insights that are not supported by the data.
- It sometimes produces irrelevant or incoherent summaries.
- Logical inconsistencies may occur between the feedback and the generated output.

Your task is to design a strategy to validate and improve the agent's outputs, making it more reliable and capable of functioning autonomously.

Task

Create a system that validates the outputs of the AI agent and ensures they are accurate, relevant, and consistent. Begin by outlining the assumptions, criteria, and analysis required to define the validation structure. Document these points briefly in the `README` file or as comments in the code if time allows. Use these insights to implement a Python-based prototype demonstrating the approach and propose how the system can improve over time.

Deliverables

- A **zip file** containing all necessary files for your project.
  - Include Python script(s) and any supporting data (e.g., mock JSON files) needed to demonstrate the prototype.
  - Ensure the code is well-commented to explain the system's logic and design choices.
  - Optionally, include a *README* file summarizing your assumptions, analysis, and key decisions made during the task.

Time Limit

This task is designed to be completed within 2 hours, and candidates are encouraged to prioritize clarity and showcasing their skills over creating a fully functional system. The

time constraint reflects real-world expectations of efficiency and throughput. Focus on presenting your skills and knowledge effectively within this limit.

Evaluation Criteria

1. **Validation Strategy**:
   - Is the strategy practical and aligned with the task objectives?
   - Are creative and effective solutions proposed?

2. **Technical Proficiency**:
   - Does the Python script demonstrate appropriate use of libraries and methods?
   - Is the code functional and well-structured?

3. **Autonomous Safety Design**:
   - How effectively does the candidate design for autonomous validation and error handling?
   - Is the balance between human input and agent autonomy well-conceived?

4. **Problem Understanding and Preparation**:
   - Does the candidate clearly articulate their assumptions and criteria before crafting the solution?
   - Are the validation strategy and design decisions well-reasoned and justified?

5. **Presentation of Skills**:
   - Ensure candidates demonstrate their thought process and knowledge effectively within the constraints.
   - Is the Python coding style clear and professional?
   - Does the work reflect state-of-the-art (SOTA) knowledge in the domain?
   - Does the submission highlight strong problem-solving skills, even if the code is not executable due to the time limit?

Submission Instructions

- Submit a **zip file** containing all project files.
- Ensure your Python script is well-commented to explain your logic and approach.
- Optionally include a **README file** to outline your analysis, assumptions, and key decisions.