Lecture 16: Agent Evaluation Techniques

🎯 Learning Objectives

By the end of this lecture, you should be able to:

- Understand why evaluating agentic systems is different from evaluating static models.
- Learn key metrics for measuring agent performance.
- Use logging and evaluation frameworks to analyze behavior.
- Design experiments to benchmark your agent's reasoning, reliability, and effectiveness.

🧩 Key Concepts

Why Agent Evaluation Is Challenging

- Agents are **interactive**, **stateful**, and **stochastic**.
- Performance varies by:
  - Prompt quality
  - Tool behavior
  - External inputs and memory state
- Requires measuring behavior **over time**, not just static accuracy.

Key Evaluation Metrics

- **Task Success Rate**: Did the agent accomplish its goal?
- **Accuracy**: Was the final answer or result correct?
- **Coherence**: Were the reasoning steps logical?
- **Efficiency**: Number of steps, time, or tool calls used.
- **Robustness**: Ability to recover from hallucinations or errors.
- **User Satisfaction**: Subjective evaluation in real use.

Types of Evaluation

- **Automated Evaluation**:
  - Use LLMs or rules to score outputs.
  - Helpful for scale but may lack nuance.
- **Human Evaluation**:
  - Provides richer insight but less scalable.
- **Hybrid Approaches**:
  - Combine automated scoring + human spot checks.

🛠️ Required Tools/Libraries

- Python

- LangChain (optional eval utilities)
- TruLens (for LLM trace analysis)
- OpenAI API

  pip install trulens-eval langchain openai

🔬 Hands-on Exercise: Evaluate Two Agent Versions

Goal: Compare two agent configurations on the same task.

Step 1: Define evaluation task

```
"Research and summarize the main benefits of LangChain."
```

Step 2: Run two versions of the agent

```
- Version A: Basic agent with no memory.
- Version B: Agent with memory + planning.

Collect outputs, intermediate thoughts, and tool calls.
```

Step 3: Score them manually

```
Criteria:
  - Correctness (1–5)
  - Coherence (1–5)
  - Tool usage (1–5)
  - Overall effectiveness (1–5)

Record in a table or spreadsheet.
```

Step 4: Use LLM to auto-evaluate

```
prompt = f"""
Agent Output: {output}
Score from 1–5: How accurate, clear, and useful is this response?
"""

response = openai.ChatCompletion.create(...)
```

Bonus:

- Add logging middleware to capture reasoning chains automatically.
- Create a leaderboard for agent configurations.
- Use feedback to fine-tune memory settings, tool usage, and planning logic.
