Intro to AI Agentic Workflows

Emerging Trends in Agentic AI

Learning Objectives

By the end of this lesson, you will be able to:

- **Describe** recent advancements in AI agent design and workflows.
- **Implement** and test emerging agentic AI techniques, including ReAct, reflection agents, and multi-agent collaboration.
- **Evaluate** the impact of feedback loops and self-improving mechanisms in AI agent workflows.

Introduction

Agentic AI systems are evolving rapidly, integrating more advanced reasoning capabilities, **multi-agent collaboration**, and **self-improving mechanisms**. Unlike traditional AI workflows, these newer approaches allow agents to learn dynamically, collaborate in solving complex problems, and refine their responses over time. This lesson explores three key trends shaping the future of agentic AI:

1. **ReAct (Reason + Act) Agents** - Agents that integrate reasoning and action for more sophisticated decision-making.
2. **Reflection Agents** - Agents that evaluate past outputs and refine their responses.
3. **Multi-Agent Collaboration** - Agents working together to perform complex tasks through specialized roles.

Trend 1: ReAct (Reason + Act) Agents

Traditional AI agents either **predict** answers or **execute** tasks, but ReAct agents do both. By reasoning through steps and acting accordingly, these agents can **self-correct**, solve problems iteratively, and refine their processes.

Implementing a ReAct Agent

The following example demonstrates how to implement a basic ReAct agent in LangChain:

Copy

```
from langchain.chat_models import ChatOpenAI
from langchain.agents import initialize_agent, AgentType
from langchain.tools import Tool

# Define a basic reasoning function
def reasoning_function(query):
    return f"Thinking step-by-step about: {query}"

# Define a reasoning tool
reasoning_tool = Tool(
    name="ReActReasoning",
    func=reasoning_function,
    description="Provides structured reasoning before acting."
)

# Set up the agent
llm = ChatOpenAI(model="gpt-4", temperature=0.5)
react_agent = initialize_agent(
    tools=[reasoning_tool],
    llm=llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

```
# Test the ReAct agent
response = react_agent.run("How do airplanes stay in the air?")
print(response)
```

Trend 2: Reflection Agents

Reflection agents improve by **analyzing their past outputs** and adjusting their approach over time. This technique is inspired by human learning, where iterative feedback improves future decision-making.

Implementing a Reflection Agent

Here's how to create an agent that **reflects on its responses** and refines them:

Copy

```
reflection_memory = {}

def reflection_tool(query: str, response: str):
    """Stores and refines agent responses based on feedback."""
    if query in reflection_memory:
        reflection_memory[query].append(response)
    else:
        reflection_memory[query] = [response]
    return f"Updated knowledge for: {query}"

reflection_agent = initialize_agent(
    tools=[Tool(name="ReflectionMemory", func=reflection_tool, description="Store
    llm=llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)

# Test reflection mechanism
response = reflection_agent.run("What is machine learning?")
print(response)
```

Trend 3: Multi-Agent Collaboration

Instead of using **one generalist AI agent**, multi-agent systems distribute tasks among specialized agents, improving efficiency and scalability. This technique is useful for AI **research assistants, task automation, and complex workflows.**

Implementing Multi-Agent Collaboration

Below is an example where one agent retrieves information while another processes it:

Copy

```
def data_retriever(query: str):
    return f"Retrieved information for: {query}"

def data_analyzer(query: str):
    return f"Analyzing data for: {query}"

retriever = Tool(name="Retriever", func=data_retriever, description="Fetches rele
analyzer = Tool(name="Analyzer", func=data_analyzer, description="Analyzes fetche

multi_agent = initialize_agent(
    tools=[retriever, analyzer],
    llm=llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)

# Test multi-agent workflow
response = multi_agent.run("Summarize recent advancements in AI ethics.")
print(response)
```

Coding Challenge: Experimenting with Emerging Trends

Goal: Modify one of the above implementations to explore an emerging trend further.

Tasks:

1. Modify the ReAct agent to **combine reasoning and multi-step action execution**.

2. Expand the reflection agent by **integrating stored responses into new outputs**.
3. Create a multi-agent system where one agent **fact-checks another agent's response** before delivering a final output.

< Previous