Clase 154 — Agentes: tool use, ReAct, multi-agent

Parte: 2 — Deep Learning · Fuente: Yao et al. (2023) ReAct + Schick et al. (2023) Toolformer + Anthropic agent patterns (2024). Duración estimada: 95 min.

Clase 154 — Agentes: tool use, ReAct, multi-agent

> *Parte: 2 — Deep Learning · Fuente: Yao et al. (2023) ReAct + Schick et al. (2023) Toolformer + Anthropic agent patterns (2024). Duración estimada: 95 min.*

Objetivo

Construir agentes con LLMs: el LLM planifica, invoca tools (funciones, MCP servers, APIs), observa los resultados y decide el siguiente paso. Cubrir el paradigma ReAct (Reasoning + Acting), tool use moderno (function calling structured), y patrones multi-agent (workflows, swarms, supervisores).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Definir tools con JSON schema; pasar a la API del LLM.
- Implementar loop ReAct manual: prompt → LLM → tool call → execute → observation → prompt.
- Usar LangGraph o AutoGen o CrewAI para orquestar multi-agent.
- Diseñar patrones: workflow (lineal), router (decide ruta), evaluator-optimizer (loop con auto-crítica).
- Reconocer cuándo NO usar agentes (tareas determinísticas → workflow simple; agentes solo si hace falta razonamiento dinámico).

Temas

- Tool use con OpenAI/Anthropic function calling.
- ReAct prompt template.
- Loops: while LLM produces tool_call, execute and feed observation.
- LangGraph: state machines para agentes.
- AutoGen / CrewAI: multi-agent frameworks.
- Patrones (Anthropic 2024): prompt chaining, routing, parallelization, orchestrator-workers, evaluator-optimizer.
- Token cost: agentes con loops pueden gastar mucho.

Definiciones y características

- Tool: función con schema + descripción que el LLM puede invocar.
- Function calling: el LLM devuelve structured {tool_name, arguments} en lugar de texto.
- ReAct: prompt template Thought → Action → Observation → Thought → ....
- Agent: LLM en loop con tools y memoria de pasos previos.
- Supervisor pattern: 1 LLM coordina varios sub-agents.
- Token bound: agentes pueden hacer N iteraciones; siempre poner max_iterations.

Dataset / recursos

- Anthropic Claude API o OpenAI API (function calling).
- Librerías: anthropic, openai, langgraph, crewai, autogen.

Ejercicios

1. Tool básico: definir weather(city) y search(query) tools en Claude API. Pedirle que use ambos.
2. ReAct loop manual: implementar el loop sin librería, en Python puro.
3. LangGraph: definir un grafo: router → tool → evaluator → end. Compilar y ejecutar.
4. Multi-agent (CrewAI): 3 agents (researcher, writer, editor) para producir un blog post.
5. Evaluator-optimizer: agent que escribe + agent que critica + loop hasta approved=True.

Homework verificable

Agente de investigación con tools:

1. Tools: web_search, fetch_url, extract_text, save_note.
2. Prompt: "investigá X y produce un resumen de 200 palabras citando fuentes".
3. Loop ReAct hasta produce el resumen.
4. Reportar # de iteraciones, # de tool calls, costo en tokens.

Criterio de aceptación: resumen producido con ≥ 3 fuentes citadas; max_iterations no excedido.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Loop infinito	LLM repite mismas tool calls. Fix: max_ite
Tool falla, LLM no maneja	Fix: pasar el error como observation.
Costo explota	Agentes son caros. Fix: usar Haiku/GPT-min
Tool schema mal	LLM no llama. Fix: descriptions claras, sc
Output no estructurado	Fix: forzar JSON schema en respuesta final

Preguntas frecuentes

Agentes vs workflows simples?

Workflow simple si la secuencia es fija. Agente cuando el path depende del input. Anthropic recomienda: empieza simple, agregá agentic complexity solo cuando justifica.

LangGraph vs CrewAI vs AutoGen?

- LangGraph: state machines explícitos, control fino.
- CrewAI: roles + delegation, fácil para multi-agent estilo "team".
- AutoGen: conversaciones entre agents, research-oriented.

MCP en agentes?

Sí — los MCP servers son una fuente de tools estándar. Combinar MCP + agente.

Eval de agentes?

Difícil. AgentBench, τ-Bench, SWE-Bench (coding). Para custom: LLM-as-judge + human spot check.

Memoria persistent?

Vector DB + tool recall(query). O frameworks como mem0.

Referencias

- Yao et al. (2023), ReAct: Synergizing Reasoning and Acting, ICLR.
- Schick et al. (2023), Toolformer, NeurIPS.
- Anthropic (2024), Building Effective Agents — patterns guide.
- LangGraph docs, CrewAI, AutoGen.

Siguiente clase

Clase 155 — LLM Evaluation: MMLU, MT-Bench, LLM-as-judge, evals propios

Apéndice: notebook (primer bloque)

Implementamos un loop ReAct desde scratch (sin LLM real, usando un mock determinístico) y luego 2 agentes coordinados por un manager.

```
import re, random
random.seed(42)
```

Archivos complementarios

- notebook.ipynb