Introduction to Agents, Agentic Supervisors and Agentic RAG

Compiled by D.Gueorguiev, 9/24/2025

# Workflows and Agents

Distinction between workflows and agents

From [1]:

*Workflows are systems where LLMs and tools are orchestrated through predefined code paths.*

*Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.*

When and when not to use agents

From [1]:

*When building applications with LLMs, we recommend finding the simplest solution possible, and only increasing complexity when needed. This might mean not building agentic systems at all. Agentic systems often trade latency and cost for better task performance, and you should consider when this tradeoff makes sense.*

*When more complexity is warranted, workflows offer predictability and consistency for well-defined tasks, whereas agents are the better option when flexibility and model-driven decision-making are needed at scale. For many applications, however, optimizing single LLM calls with retrieval and in-context examples is usually enough.*

A diagram of a workflow

AI-generated content may be incorrect. Figure 1: Visualizing the difference between Workflows and Agents

## Building block: The augmented LLM

The basic building block of agentic systems is an LLM enhanced with augmentations such as retrieval, tools and memory. The current LLMs actively use these capabilities , generating their own search queries, selecting appropriate tools, and determining what information to retain.

A diagram of a process

AI-generated content may be incorrect.

Figure 2: Building an augmented LLM

## Workflow: Prompt chaining

Prompt chaining decomposes a task into a sequence of steps, where each LLM call processes the output of the previous one. One can add programmatic checks (aka gates) on the intermediate steps to ensure the process is progressing as expected.

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 3: The prompt chaining workflow

When to use such workflow:

This workflow is used when the task can be easily decomposed into fixed list of subtasks. The main goal is to trade latency for higher accuracy, by making each LLM call an easier task.

Examples where prompt chaining is useful are:

* Generating marketing copy, then translating it into a different language
* Writing an outline of a document, checking that the outline meets certain criteria, then writing the document based on the outline

## Workflow: Routing

Routing classifies an input and directs it to a specialized follow-up task. This workflow allows for separation of concerns and building more specialized prompts. Without this workflow, optimizing for one kind of input can hurt performance on other inputs.

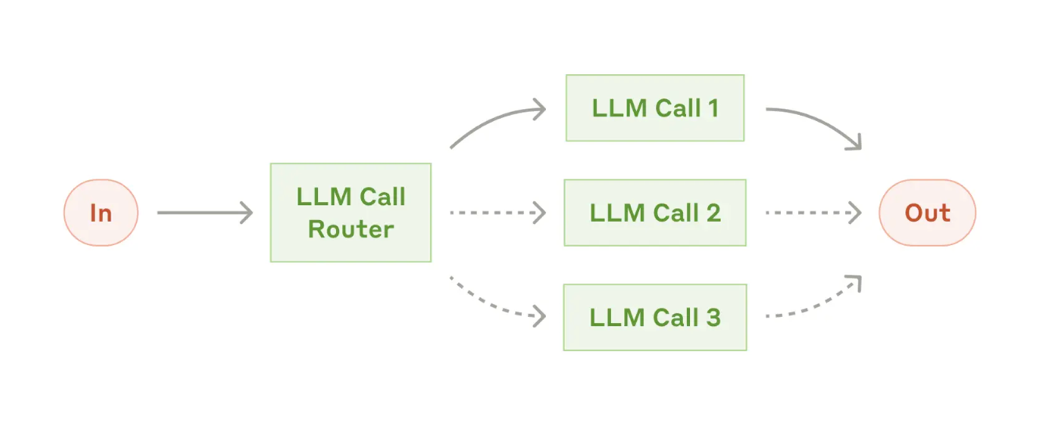


Figure 4: the routing workflow

When to use this workflow:

When there are distinct categories that are better handled separately, and where classification can be handled accurately, either by an LLM or a traditional ML classification model/algorithm.

Examples where routing is useful are:

* Directing different types of customer service queries (general questions, refund requests, technical support) into different downstream processes, prompts and tools.
* Routing easy/common questions to smaller LM and harder/unusual questions to larger/specialized LMs

## Workflow: Parallelization

# Building a Research Agent

# Multi-agent supervisor

# References

[1] <https://www.anthropic.com/engineering/building-effective-agents>

[2] <https://langchain-ai.github.io/langgraph/tutorials/workflows/#agent>

[3] <https://langchain-ai.github.io/langgraph/tutorials/multi_agent/agent_supervisor/>

[4] <https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_agentic_rag/>

[5] <https://academy.langchain.com/courses/take/deep-research-with-langgraph/>

[6] <https://github.com/langchain-ai/deep_research_from_scratch>