Introduction to LangChain & LangGraph

Maryam Berijanian, Dr. Dirk Colbry

10/8/2025

References Used Throughout the Slides

1. A Hands-On Guide to Building Intelligent Systems, Antonio Gulli

2. LangChain Academy – curated courses & tutorials.
GitHub: https://github.com/langchain-ai/langchain-academy.git

3. LangGraph Cookbook

4. LangChain Tutorials

5. Interrupt – the AI Agent Conference by LangChain

6. Prof. Ghassemi Lectures and Tutorials, AI Agents lectures

Agenda

Agenda

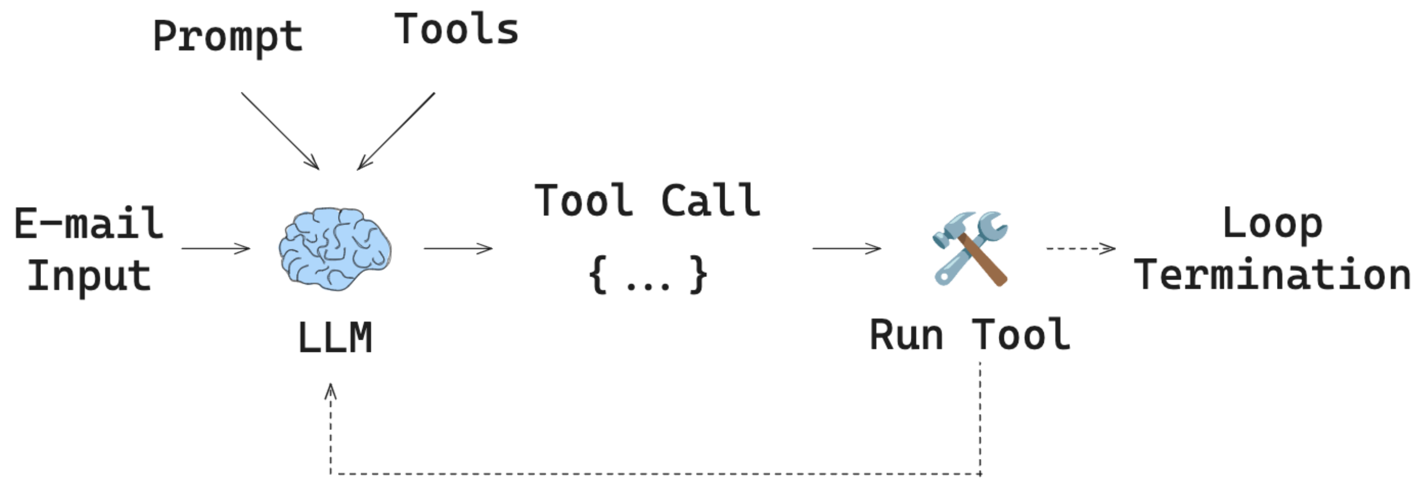
0) Introduction

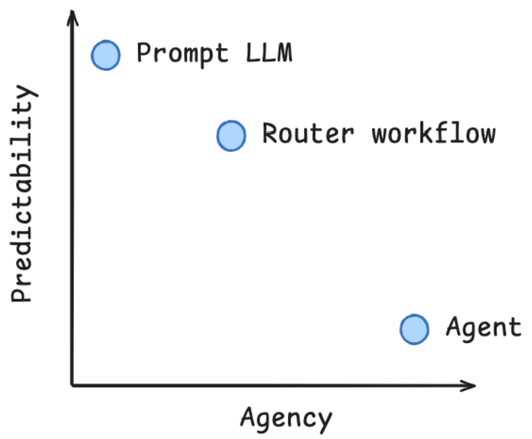
- LLMs make it possible to embed intelligence into a new class of applications.
- Chat models are the foundation of LLM applications. They are typically accessed through a chat interface that takes a list of messages as input and returns a message as output.

LLM vs Agent

- LLM: a text generator (no goals, tools, memory by itself).
- Consider loop: *Observe* → *Decide* → *Act* → *Reflect*.
- It is incredibly difficult to build systems that reliably execute on these tasks.
- Agents can automate a wide range of tasks that were previously impossible.
- Agent: LLM + tools + memory + policy that decides what to do next (answer, search, call a function, delegate, clarify).

Agent Example





LangChain vs LangGraph

LangChain

- LangChain is a framework to help developers build applications using power of LLMs.
- Provides abstractions and components to easily create complex workflows, or "chains," by combining different parts.
- Provides a standardized interface for chat models, making it easy to access many different providers.

LangChain vs LangGraph

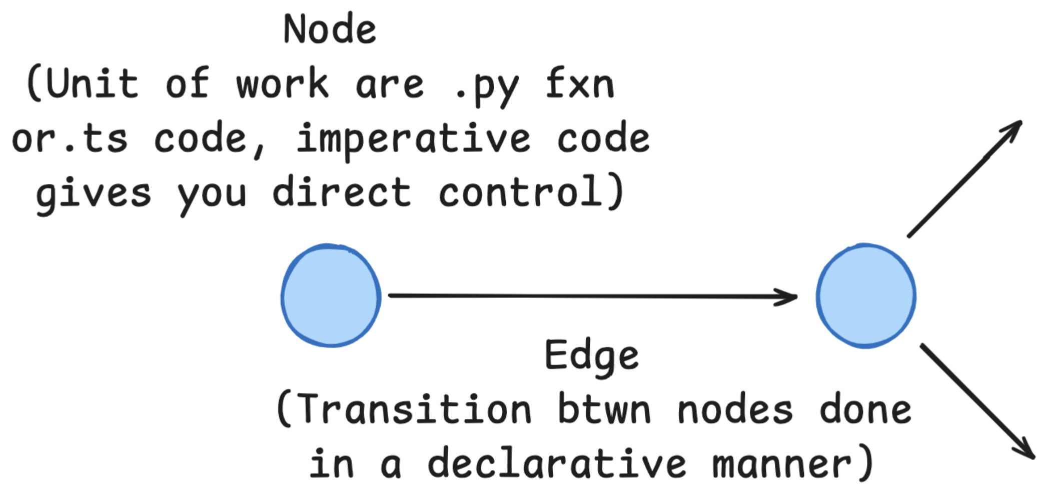
LangGraph

- LangGraph: an extension of LangChain
- Enables creation of multi-agent applications by representing the workflow as a graph.
- Helps developers add better precision and control into agent workflows
- Suitable for complexity of real-world systems.

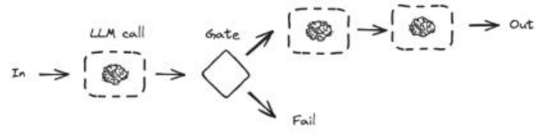
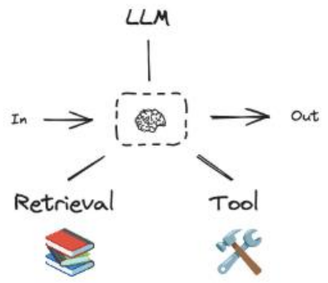
Tip: Start with LangChain for simple flows; move to LangGraph for collaboration/routing/retries/long-lived state.

Node and Edge

LangGraph allows you to define nodes (which can be LangChain runnables or other Python functions) and edges (which define the transitions between nodes).


Node
(Unit of work are .py fxn or.ts code, imperative code gives you direct control)
Edge
(Transition btwn nodes done in a declarative manner)

Ecosystem

LangSmith	Observability (Tracing and evaluation)	🔍
LangGraph	Orchestration (Application control flow)	Workflow / Agent 
LangChain	Integrations (Standard interfaces)	Augmented LLM 

Agenda

1) Setup for Google Colab

- Requirements for today:
  - **HuggingFace account (for LLMs)**
  - **Personal Google account (for Google Colab)**
- Google Colab:
  - Free tier: usually NVIDIA T4 GPU. It's not the fastest GPU, but it's available for free.
  - Paid subscription (Colab Pro): faster GPUs, more memory, etc.
  - Google Colab Pro is free for students and teachers! Link: https://colab.research.google.com/signup

Link to GitHub
Please Leave a ⭐ !

https://github.com/maryambrj/langchain-langgraph-tutorials.git

1) Setup for Google Colab

🛑⚠️ **Please save a copy before starting to edit the current notebook!** `File -> Save a copy in …`

- Running this entire workshop **without paid API keys**.
  - Using models from **HuggingFace**.
- Steps for authenticating with HuggingFace Hub :
  - Create a token in settings: https://huggingface.co/settings/tokens
  - Choose “write” option
  - Set it as a secret in Google Colab ("🔑" icon, left panel)
  - Name the secret key `HF_TOKEN`
  - Restart the session + select a GPU

1) Setup for Google Colab

- Install:
  - `` `transformers` ``
  - `` `accelerate` ``
  - `` `langchain` ``
  - `` `langgraph` ``
  - `` `langchain-huggingface` ``
  - `` `langchain-community` ``
  - `` `sentence-transformers` ``
  - `` `faiss-cpu` ``
- Load a small chat model **TinyLlama 1.1B Chat**

Agenda

2) Basic Usage

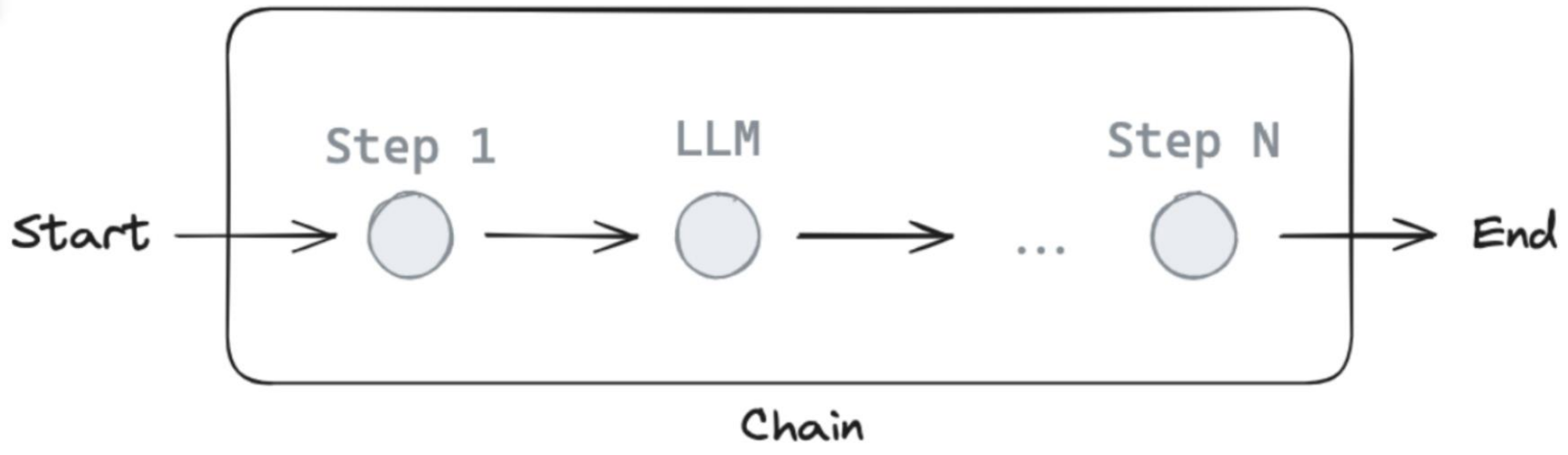
LangChain Expression Language (LCEL)

- A declarative way to compose chains.
- Combine various LangChain components, like prompts, models, and output parsers, into complex workflows.
- Build pipelines with features like streaming, async support, and parallel execution.
- Creation of chains by **piping components together using the `|` operator.**

Agenda

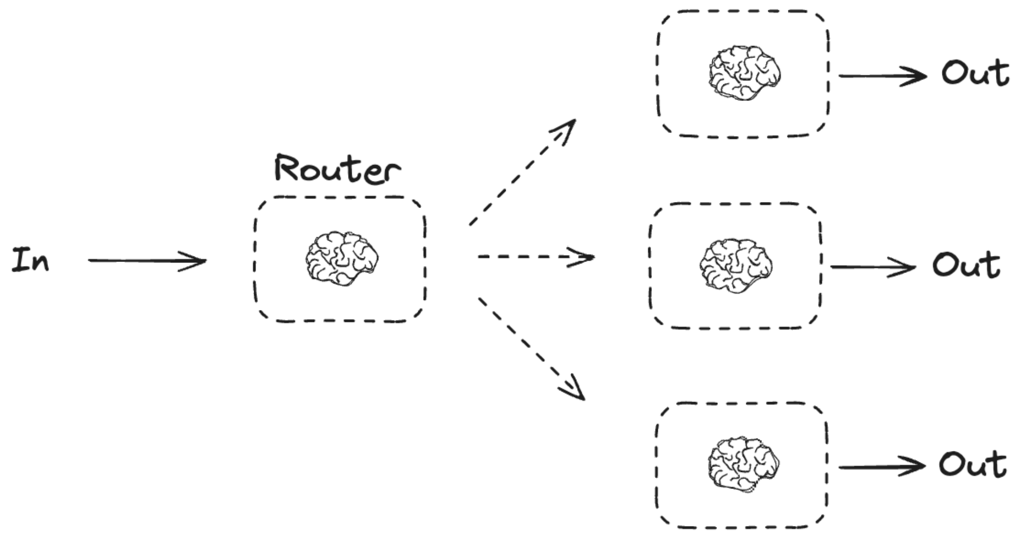
3) Routing vs Chaining

- **Chaining** can be thought of as a linear path:
  - `Step A` (Planner) -> `Step B` (Researcher) -> `Step C` (Writer).
- A fixed sequence of steps

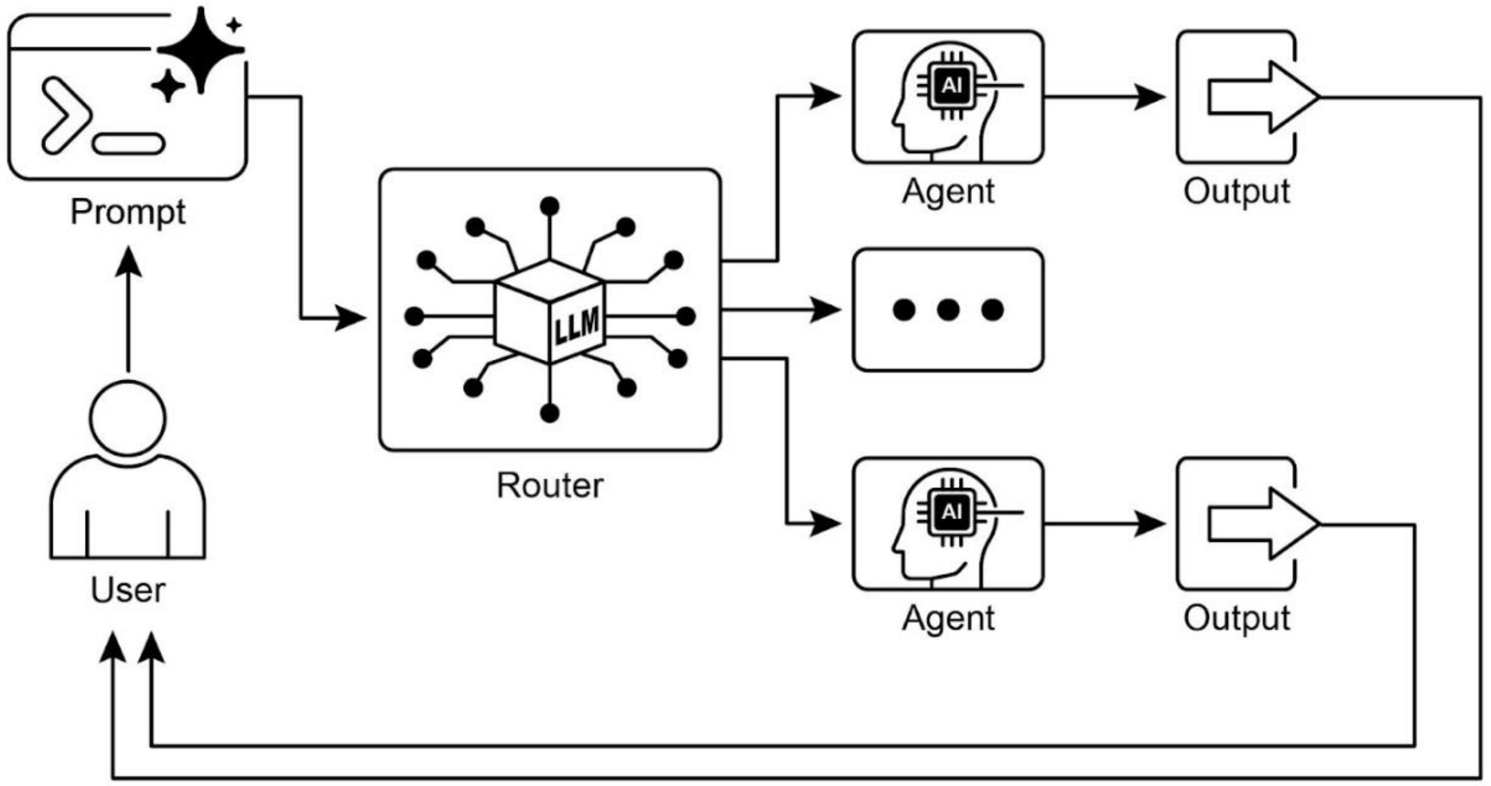


3) Routing vs Chaining

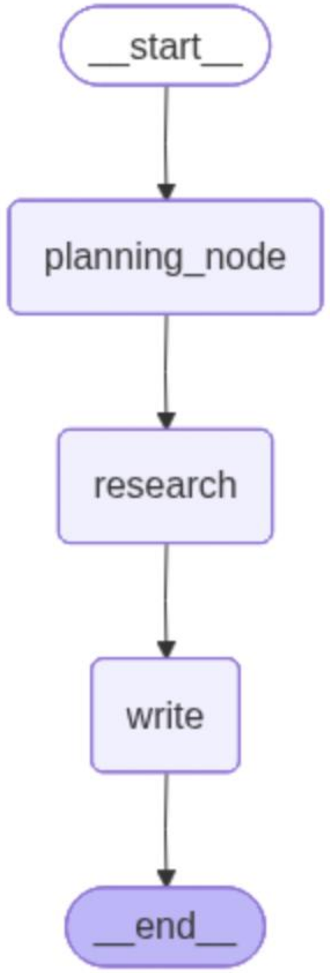
- **Routing**: More complex applications require dynamic workflows where the next step depends on the output of the previous one or on some condition.
  - Workflows with **loops** and **dynamic** paths
- Allows for **conditional** paths:
  - After `Step A` (Planner), based on the result, the workflow might go to `Step B` or `Step C`
- **LangGraph** supports conditional edges and routing logic

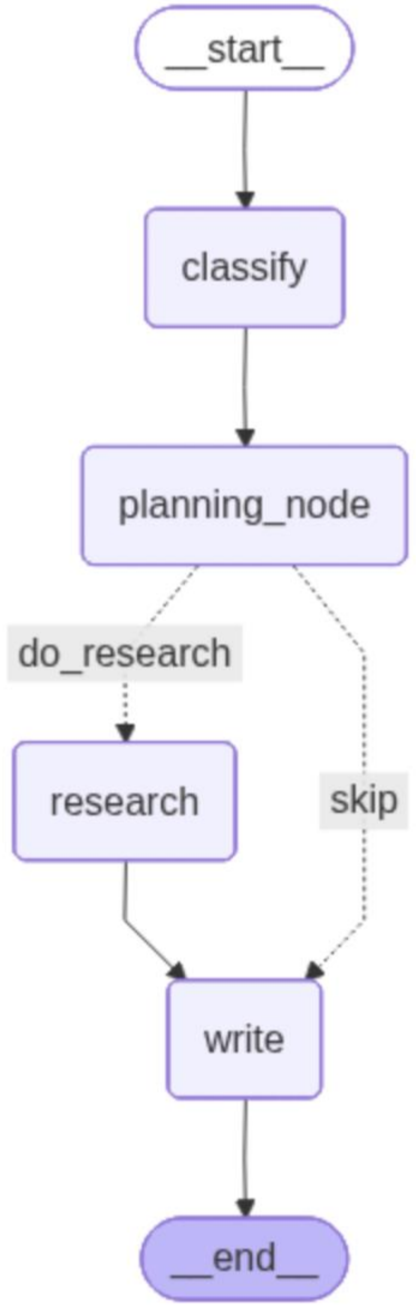


LLM as a Router



Example: Research Agent



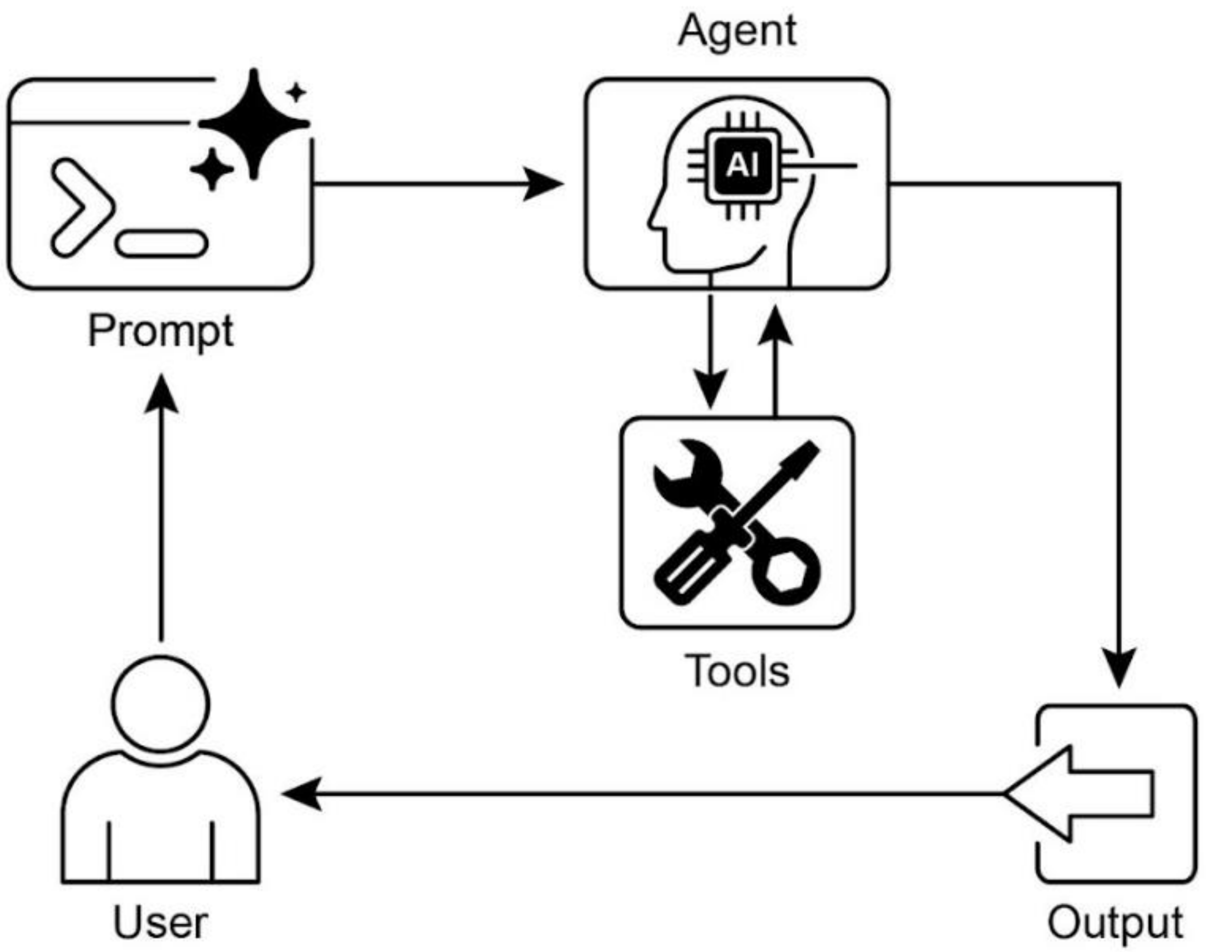


Agenda

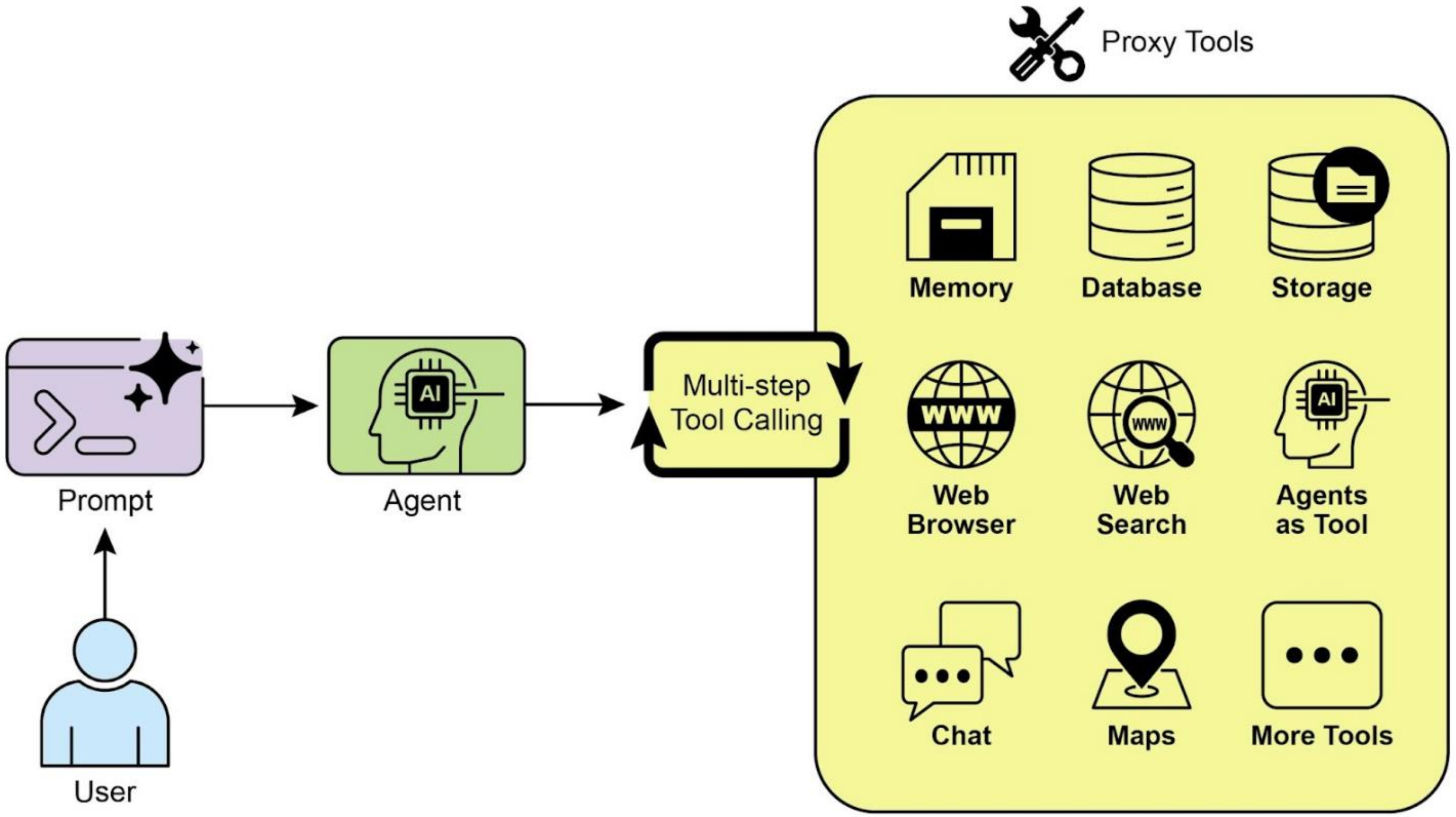
4) Tool Use

- Tools: external functionalities agents can use to extend their capabilities
- Can be anything: from a calculator or a search engine stub to more complex integrations with databases, APIs, or other services.
- Creating tools: can be done using the `` `@tool` `` decorator, which transforms Python functions into callable tools.

4) Tool Use



4) Tool Use

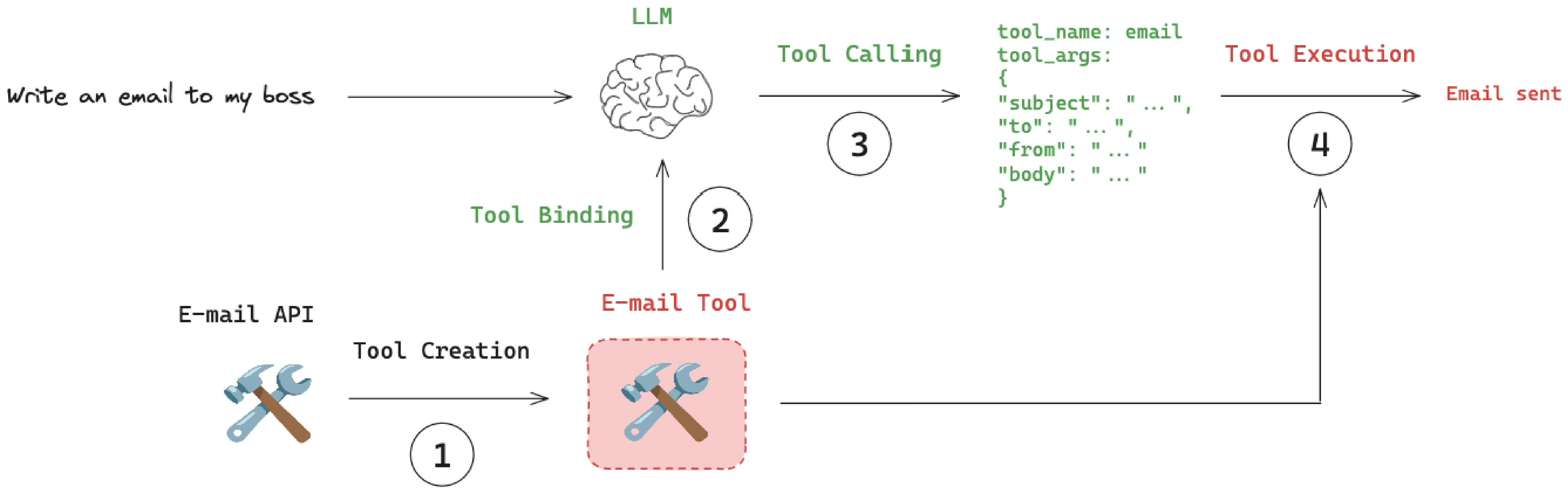


Proxy Tools
Prompt
Agent
User
Multi-step Tool Calling
Memory
Database
Storage
Web Browser
Web Search
Agents as Tool
Chat
Maps
More Tools

4) Tool Use

- LangChain provides a framework for defining and integrating various types of tools
- Agents can decide when and how to use these tools to achieve their goals.
- They automatically infer the tool's name, description, and expected arguments from the function definition.

4) Tool Use


LLM
Write an email to my boss
Tool Calling
3
tool_name: email
tool_args:
{
"subject": " ... ",
"to": " ... ",
"from": " ... "
"body": " ... "
}
Tool Execution
4
Email sent
Tool Binding
2
E-mail API
Tool Creation
1
E-mail Tool

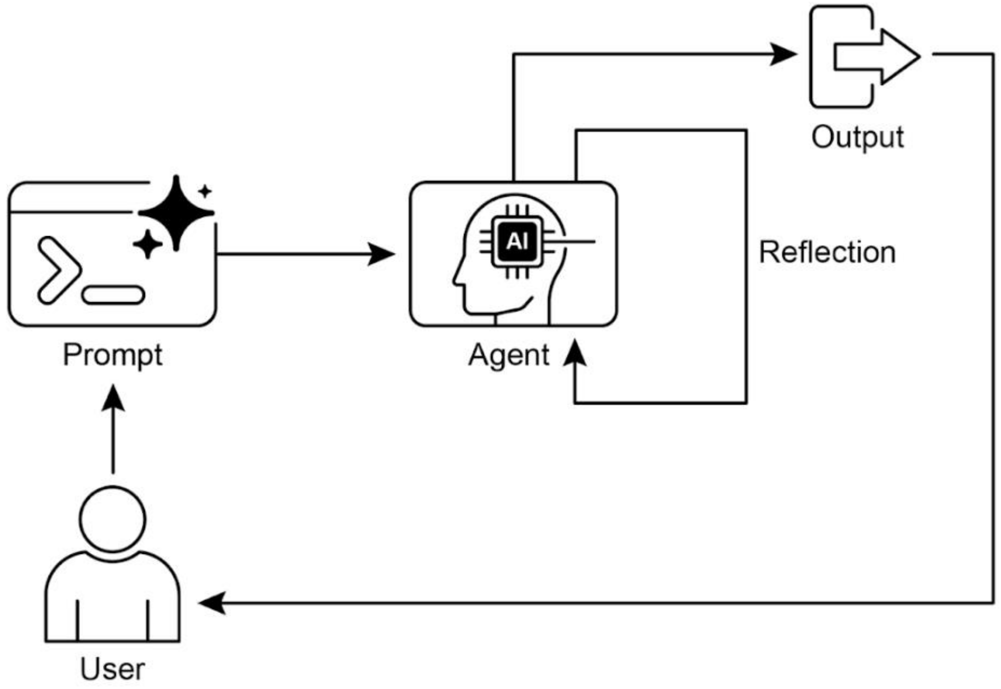
Agenda

5) Reflection

- Agents can improve their outputs through reflection, a process where they evaluate and refine their work.
- This allows the agent to iteratively improve its response.

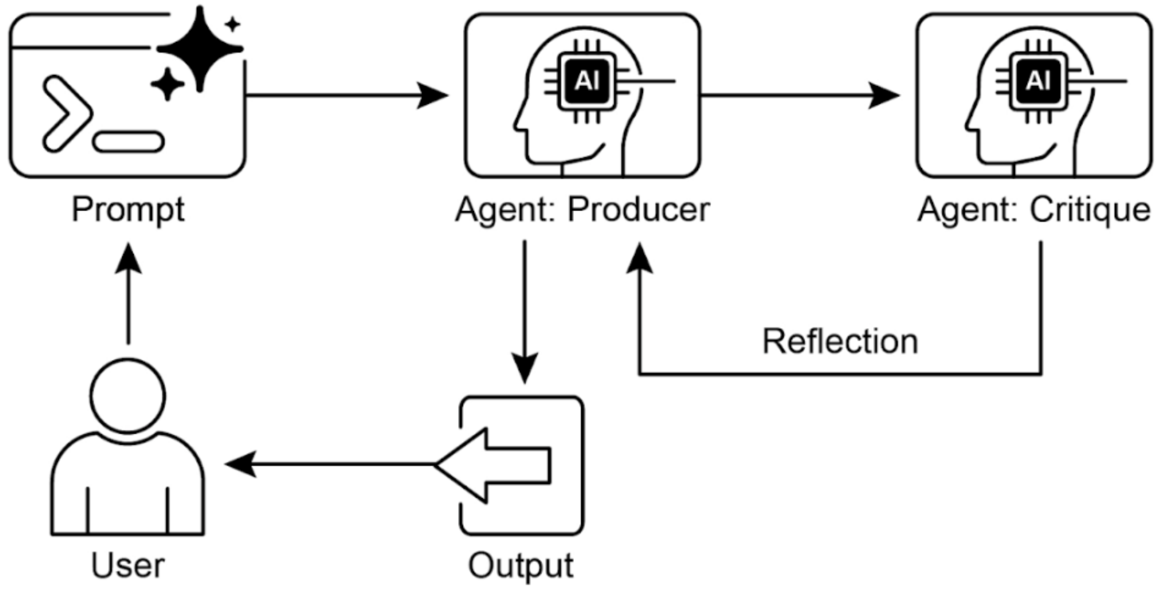
Two main types of reflection:

- **Self-reflection**: agent critiques **its *own*** generated output.



5) Reflection

- **External reflection**: A ***separate* agent** or component acts as a critic, providing feedback on the initial draft generated by the primary agent.
- External feedback can offer a different perspective.

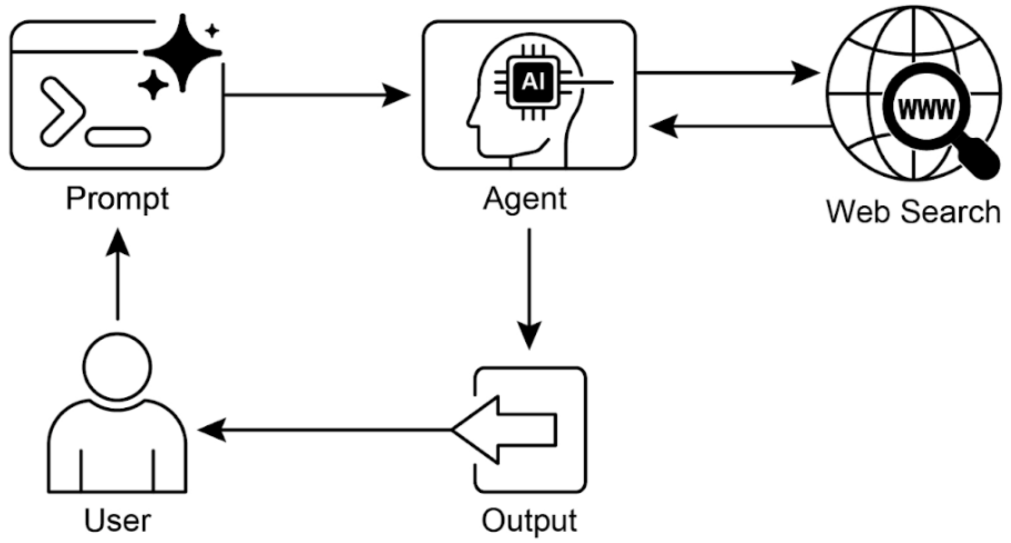


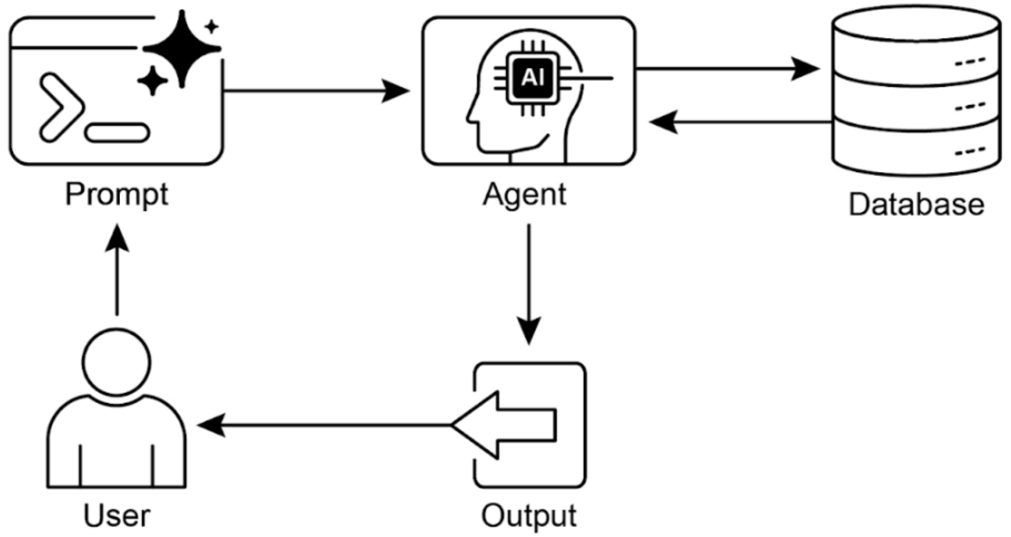
Agenda

6) Retrieval-Augmented Genertion (RAG)

- A technique that combines the power of LLMs with external knowledge sources to generate more accurate and informed responses.
- Instead of relying solely on the knowledge encoded in the model's parameters during training, RAG systems retrieve relevant information from a separate knowledge base
    - Database
    - Collection of documents
    - Internet
- Use this retrieved information as context when generating an answer.
- Reduce the likelihood of:
    - Hallucination
    - Providing outdated information

6) Retrieval-Augmented Genertion (RAG)

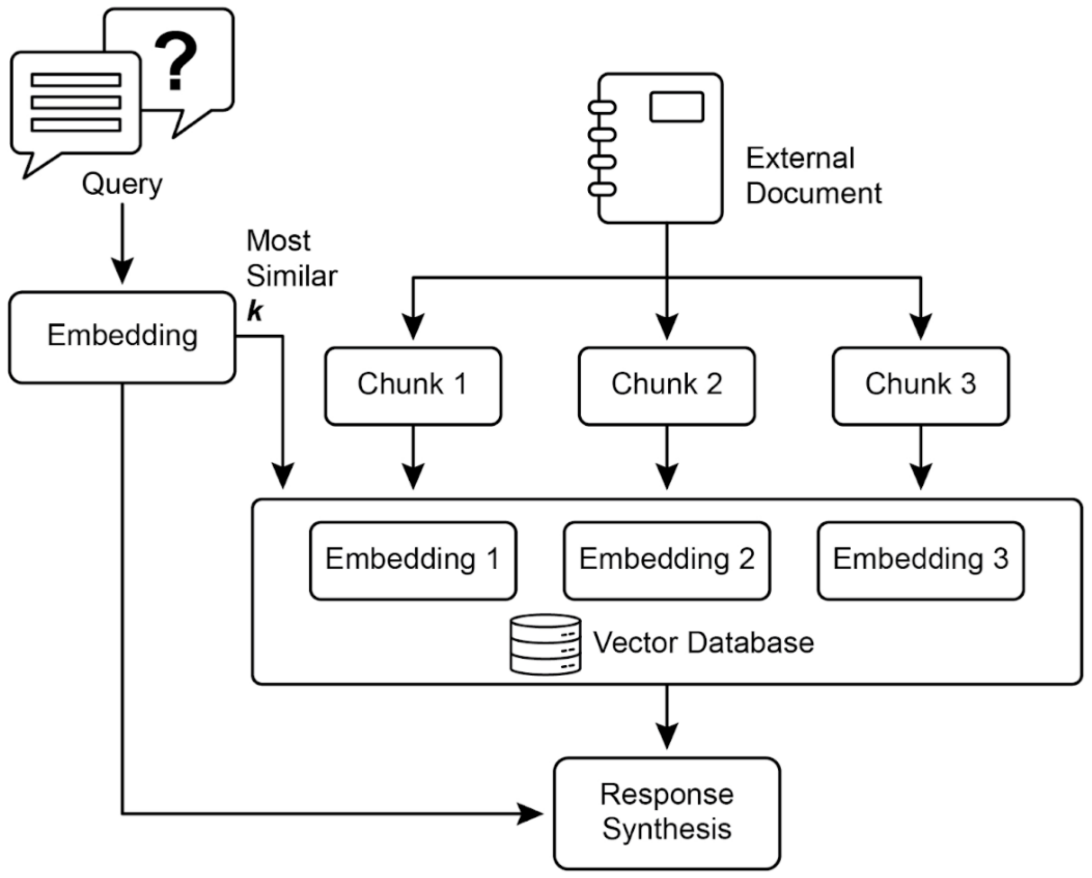




6) Retrieval-Augmented Genertion (RAG)

RAG Core Concepts:

- Chunking
- Embeddings
- Vector database

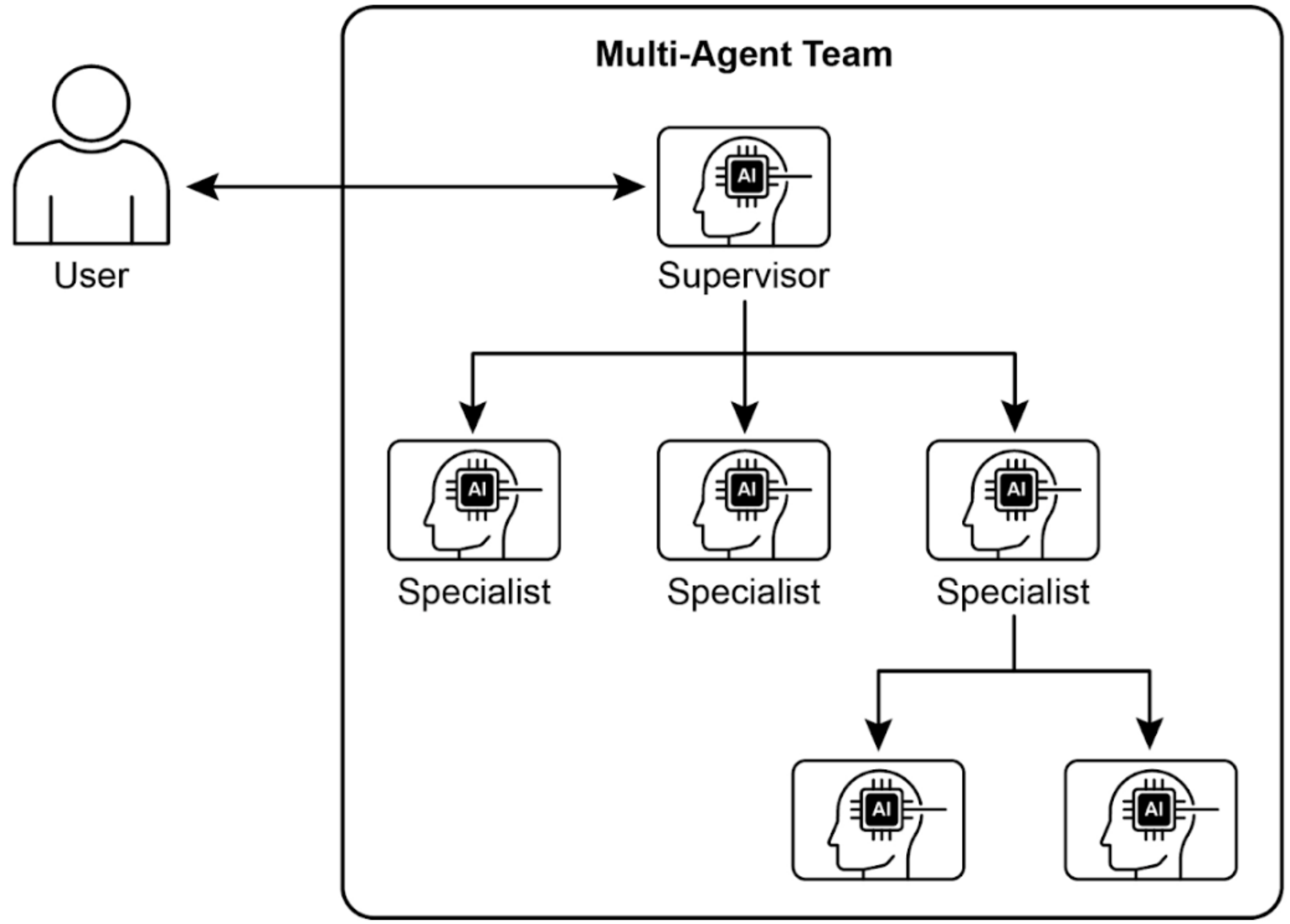


Agenda

7) Multi-Agent Collaboration

- Multiple independent or semi-independent agents work together to achieve a common goal.
- Interaction and synergy between agents.
- Each agent has a defined role, specific goals, and access to different tools or knowledge bases.

7) Multi-Agent Collaboration



7) Multi-Agent Collaboration

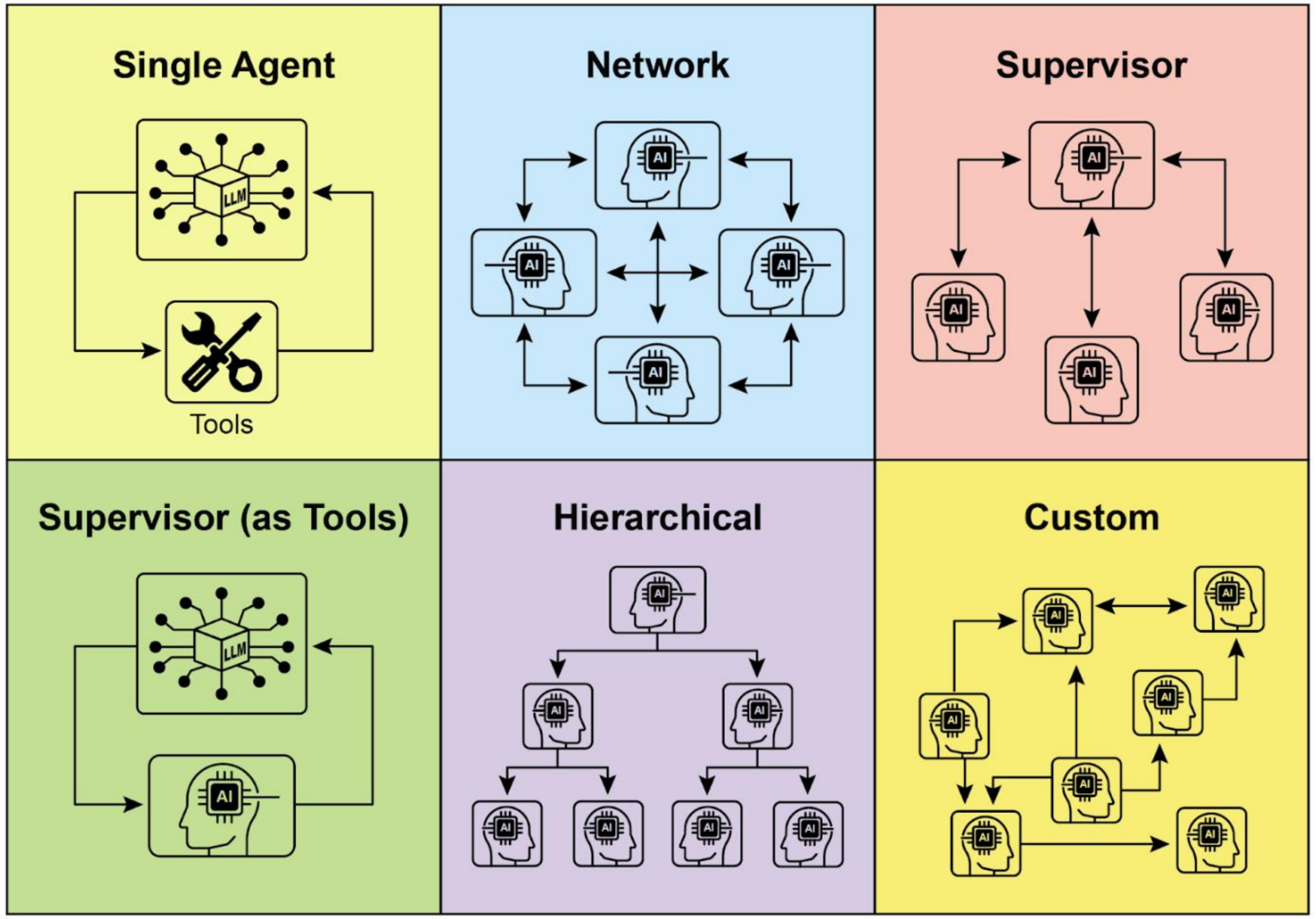


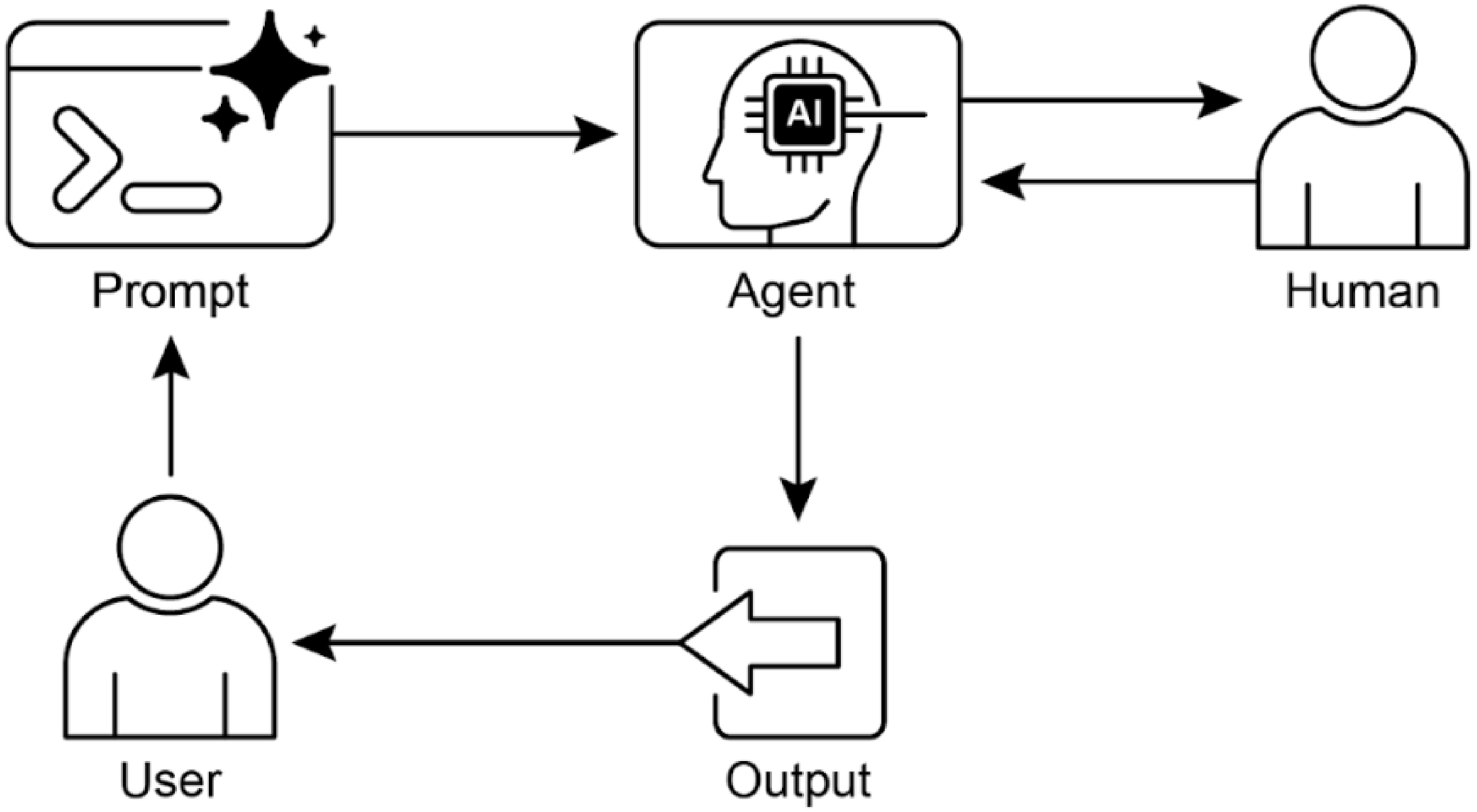
Fig. 2: Agents communicate and interact in various ways.

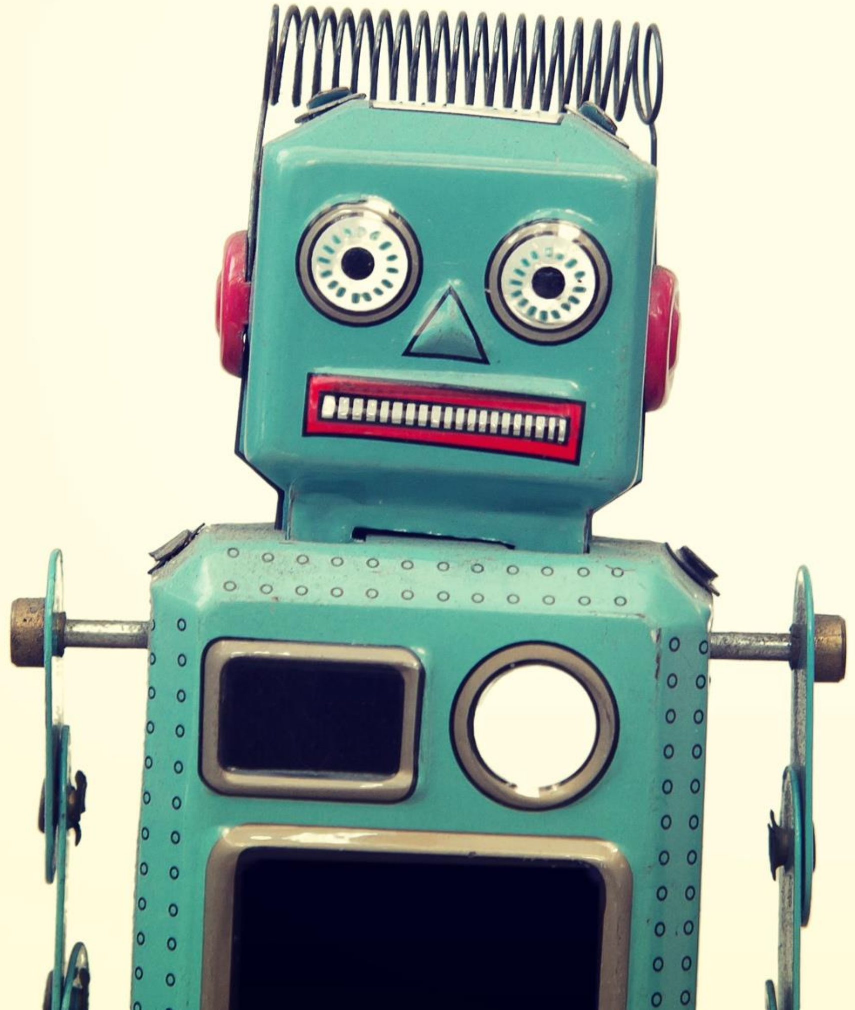
Agenda

8) Human in the Loop

- Integrates AI with human input to enhance agent capabilities.
- Optimal AI performance frequently requires human insight
  - In scenarios with **high complexity** or **ethical** considerations.
- Lack of scalability, operators cannot manage millions of tasks
- Requires a hybrid approach combining automation for scale and HITL for accuracy.
- The effectiveness is heavily dependent on the **expertise** of the human operators.

8) Human in the Loop





Demo:
Resume Chatbot

References

1. A Hands-On Guide to Building Intelligent Systems, Antonio Gulli

2. LangChain Academy – curated courses & tutorials.
GitHub: https://github.com/langchain-ai/langchain-academy.git

3. LangGraph Cookbook

4. LangChain Tutorials

5. Interrupt – the AI Agent Conference by LangChain

6. Prof. Ghassemi Lectures and Tutorials, AI Agents lectures

Thank you for your attention!

Maryam Berijanian berijani@msu.edu

Dirk Colbry colbrydi@msu.edu

Julian Venegas venegas5@msu.edu

https://github.com/maryambrj/langchain-langgraph-tutorials.git

Please Leave a ★ !