Swiss Data Science Center

Introduction to Agentic AI

Context

Swiss Data Science Center

- LLMs can not perform tasks except answering questions based on its internal knowledge
- Updated and additional knowledge can be added using **Retrieval Augmented Generation** (RAG)
- However, the classical RAG approach use mostly static information embedded in a vector database
- What if we could use additional source of information like a human would do to answer questions? (What is the weather in Lausanne tomorrow?; Can you program an appointment for me at the doctor? …)

What is Agentic AI?

Swiss Data Science Center

- Agentic AI uses the concept of Agents
- An LLM Agent is a LLM that is able to use **“Tools”**
- These Tools can be anything that can be programmatically called, for example, any API (e.g. *get_weather(lat,lon)* ; *set_appointment(name, date),* …)

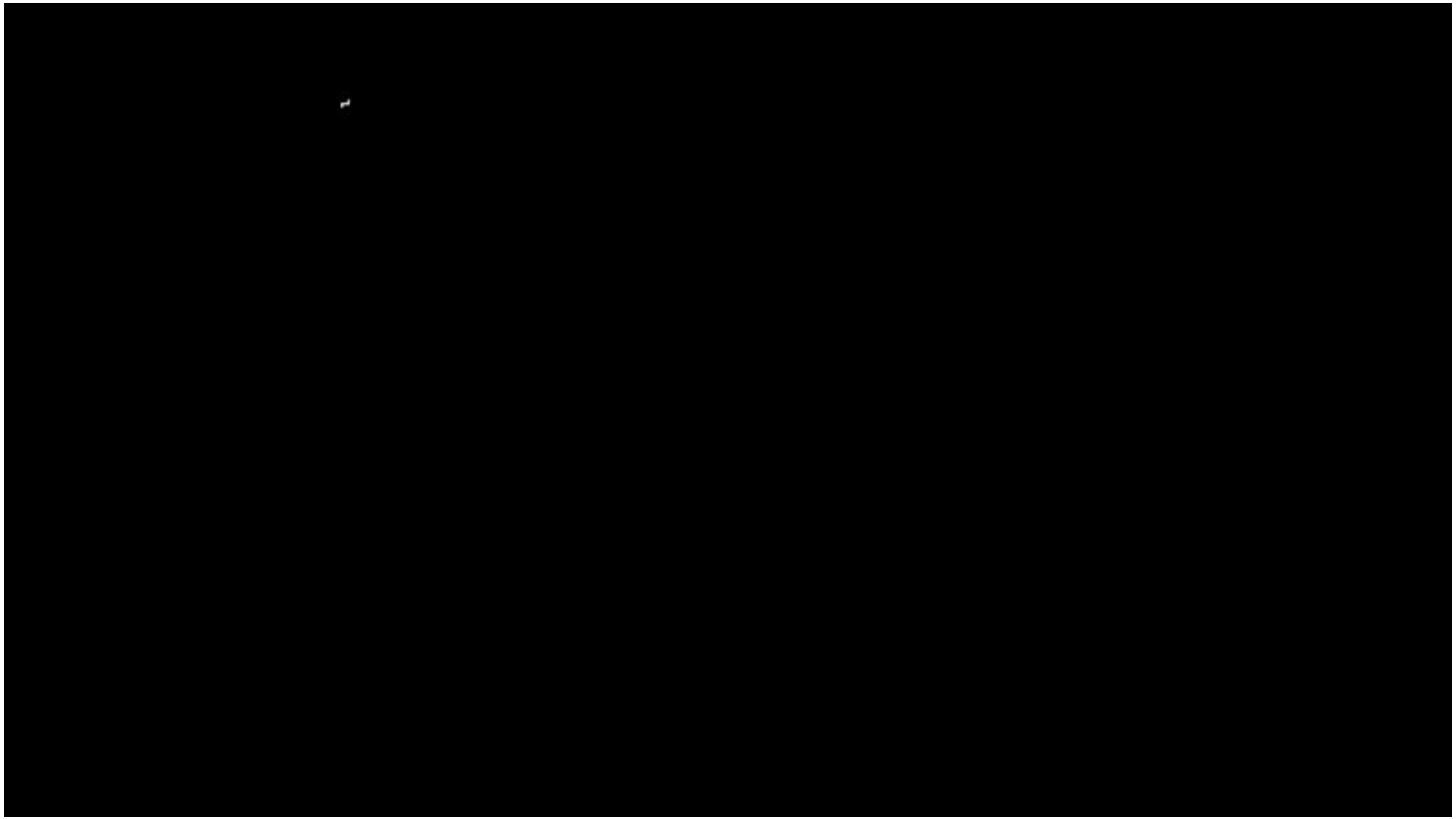
Different Levels of Agentic AI

Swiss Data Science Center

Agency Level	Description	How that's called	Example Pattern
☆☆☆	LLM output has no impact on program flow	Simple Processor	`process_llm_output(llm_response)`
★☆☆	LLM output determines an if/else switch	Router	`if llm_decision(): path_a() else: path_b()`
★★☆	LLM output determines function execution	Tool Caller	`run_function(llm_chosen_tool, llm_chosen_args)`
★★★	LLM output controls iteration and program continuation	Multi-step Agent	`while llm_should_continue(): execute_next_step()`
★★★	One agentic workflow can start another agentic workflow	Multi-Agent	`if llm_trigger(): execute_agent()`

An Agent Thought Process

Swiss Data Science Center



When to use Agents?

Swiss Data Science Center

Don’t use Agents if:

- The workflow is entirely deterministic and can be handled by traditional rule-based systems
- When perfect reliability and consistency are required (LLMs introduce errors)

Use Agents if:

- The task involves complex reasoning across multiple steps
- For problems requiring dynamic adaptation to new information

Agentic AI: Making a Plan

Swiss Data Science Center

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Act 4: Finish[yes] ✗

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓

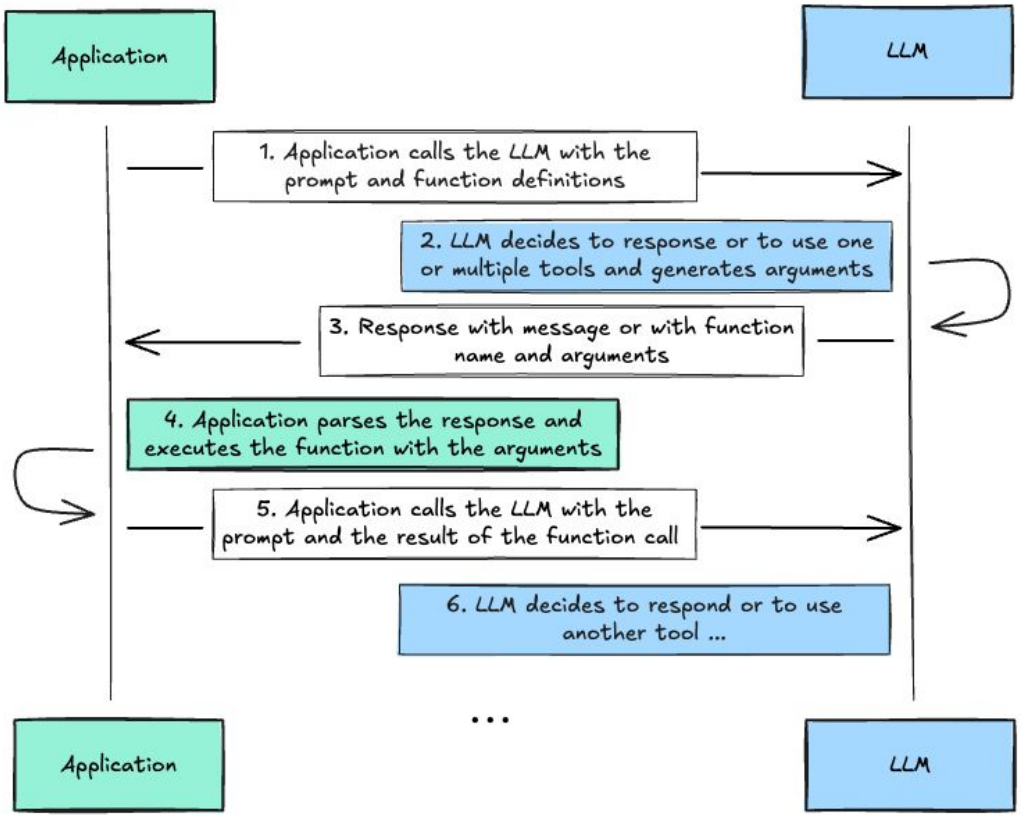
Yao, Shunyu et al. “ReAct: Synergizing Reasoning and Acting in Language Models.” *ArXiv* abs/2210.03629 (2022): n. pag.

Agentic AI: How does an Agent take action?

Swiss Data Science Center

LLM can only generate text, they can not “call” a tool.

This is the responsibility of the environment (e.g. Python) around the LLM.

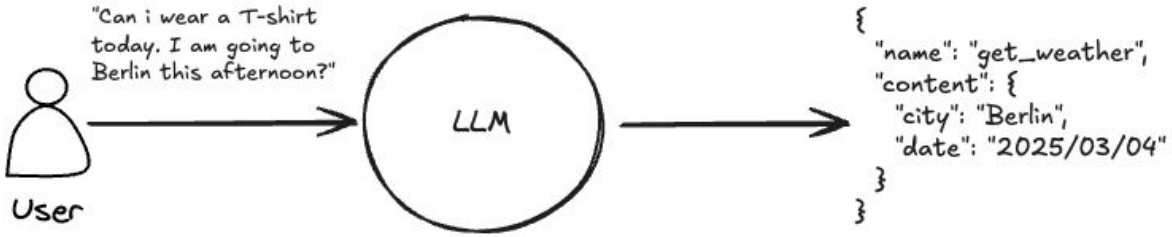


Agentic AI: How does an Agent take action?

Swiss Data Science Center

Example:

1. **User Query: “What is the weather in Lausanne?”**
2. **Agent LLM:**
   a. Input: What is the weather in Lausanne?
   b. Thought: I need to call the weather API
   c. Action: Use the function **get_weather(location)** with **location=Lausanne**
3. **The environment (e.g. python):**
   a. Call the function **get_weather(Lausanne)**
   b. Output: “8°C Cloudy”
4. **Agent LLM:**
   a. Input: Conversation History + “8°C Cloudy”
   b. Output: It is currently cloudy in Lausanne with a temperature of 8°C



Code Agent vs JSON Agent

Swiss Data Science Center

Instruction: Determine the most cost-effective country to purchase the smartphone model "CodeAct 1". The countries to consider are the USA, Japan, Germany, and India.

Available APIs
[1] lookup_rates(country: str) -> (float, float)
[2] convert_and_tax(price: float, exchange_rate: float, tax_rate: float) -> float
[3] estimate_final_price(converted_price: float, shipping_cost: float) -> float
[4] lookup_phone_price(model: str, country: str) -> float
[5] estimate_shipping_cost(destination_country: str) -> float

LLM Agent using [Text/JSON] as Action

Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

Action
Text: lookup_rates, Germany
JSON: {"tool": "lookup_rates", "country": "Germany"}

Environment 1.1, 0.19

Action
Text: lookup_phone_price, CodeAct 1, Germany
JSON: {"tool": "lookup_phone_price", "model": "CodeAct 1", "country": "Germany"}

Environment 700

Action
Text: convert_and_tax, 700, 1.1, 0.19
JSON: {"tool": "convert_and_tax", "price": 700, "exchange_rate": 1.1, "tax_rate": 0.19}

Environment 916.3

[... interactions omitted (look up shipping cost and calculate final price) ...]

Action
Text: lookup_rates, Japan
JSON: {"tool": "lookup_rates", "country": "Japan"}

[... interactions omitted (calculate final price for all other countries)...]

Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

> Fewer Actions Required!

CodeAct: LLM Agent using [Code] as Action

Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

Action

```
countries = ['USA', 'Japan', 'Germany', 'India']
final_prices = {}

for country in countries:
    exchange_rate, tax_rate = lookup_rates(country)
    local_price = lookup_phone_price("xAct 1", country)
    converted_price = convert_and_tax(
      local_price, exchange_rate, tax_rate
    )
    shipping_cost = estimate_shipping_cost(country)
    final_price = estimate_final_price(converted_price, shipping_cost)
    final_prices[country] = final_price

most_cost_effective_country = min(final_prices, key=final_prices.get)
most_cost_effective_price = final_prices[most_cost_effective_country]
print(most_cost_effective_country, most_cost_effective_price)
```

Control & Data Flow of Code Simplifies Complex Operations

Re-use `min` Function from Existing Software Infrastructures (Python library)

Environment 1.1, 0.19

Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

Wang, Xingyao et al. “Executable Code Actions Elicit Better LLM Agents.” *ArXiv* abs/2402.01030 (2024): n. pag.

Code Agent Advantages

Swiss Data Science Center

Code Agents have several advantages compared to standard JSON Agents:

- Code actions are much more concise than JSON
- On average, Code actions require 30% fewer steps than JSON -> 30% less tokens -> 30% cheaper
- Code enables to re-use tools from common libraries
- Code gets better performance in benchmarks

Model	Average ↓	GAIA subset	MATH subset	SimpleQA subset
gpt-4.5-preview Vanilla score below	78.8%	56.3%	92.0%	88.0%
anthropic/claude-3-7-sonnet-20250219	77.3%	50.0%	96.0%	86.0%
o1 Vanilla score below	76.7%	50.0%	96.0%	84.0%
o3-mini	75.0%	46.9%	98.0%	80.0%
anthropic/claude-3-5-sonnet-latest Vanilla score below	74.8%	56.3%	86.0%	82.0%
deepseek-ai/DeepSeek-R1	72.7%	50.0%	94.0%	74.0%
Qwen/QwQ-32B	68.5%	37.5%	94.0%	74.0%
gpt-4o Vanilla score below	64.1%	34.4%	78.0%	80.0%
Qwen/Qwen2.5-72B-Instruct Vanilla score below	64.0%	28.1%	76.0%	88.0%

https://huggingface.co/spaces/smolagents/smolagents-leaderboard

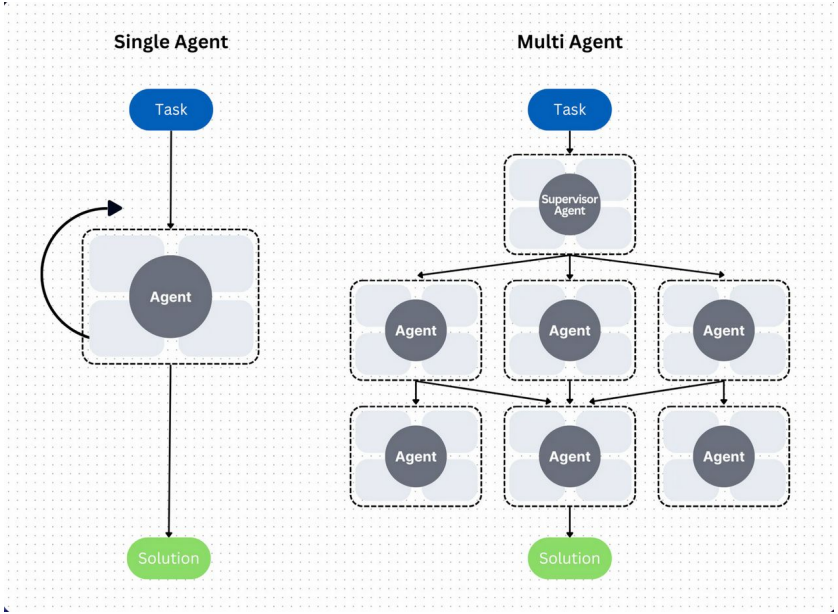
Multi-Agent System

Swiss Data Science Center

Multi-agent structures allow to separate memories between different sub-tasks, with two great benefits:

- Each agent is more focused on its core task, thus more performant

- Separating memories reduces the count of input tokens at each step, thus reducing latency and cost.



Hands-On

Swiss Data Science Center

Objectives:

- Understand why it's helpful to have agentic capabilities
- Understand how to use the *smolagents* library
- Understand the difference between a Tool Calling Agent and a Code Agent
- Implement a custom Agent leveraging the RAG pipeline that we implemented before