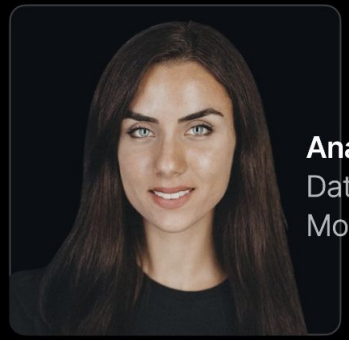
pyladies amsterdam x mollie

Introduction to LLM Agents with LangChain

Ana Chaloska
Maria Bader, PhD

25 June 2024

Who are we?



Ana Chaloska
Data Scientist
Mollie

Maria Bader
Data Scientist
Mollie

A big shoutout to ...

... PyLadies Amsterdam, for building this community

... #Mollie and Ksenia Zvereva (Mollie Developer Community) for hosting us

... #women-at-mollie and #data for support

.... The awesome volunteers that helped us set up and prepare the workshop

.... YOU - for joining and making this a success!

Workshop goals

coding / theory

- Understand how LLMs can perform tasks beyond text generation
- Grasp the fundamental components of an LLM Agent
- Build an LLM Agent with LangChain, starting from the individual components
- Look beyond packages at the core structure of an agent

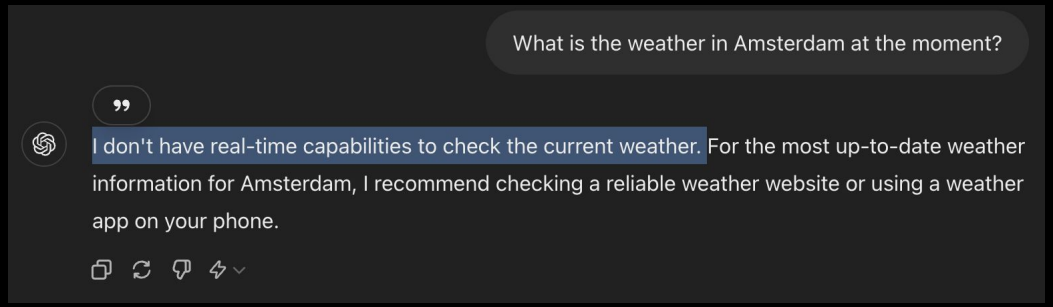
What LLMs can do

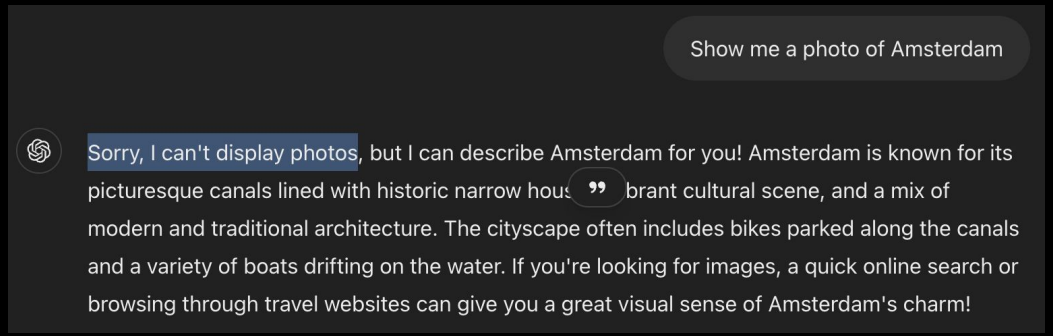
- **Question-answering** (e.g. MollieGPT)
- Text **summarization**
- **Sentiment** analysis
- Generate **code**
- **Translate** text
- Generate **embeddings** (BCC Classifier)
- etc

What LLMs can't do

- Usually **computational/mathematical** tasks
- Awareness about **real-time** information
- Provide **non-text output**

What LLMs can't do



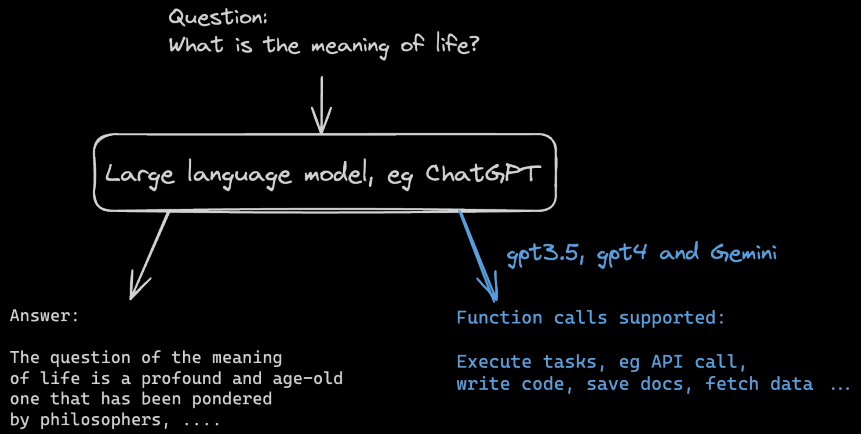


Can we help LLMs?

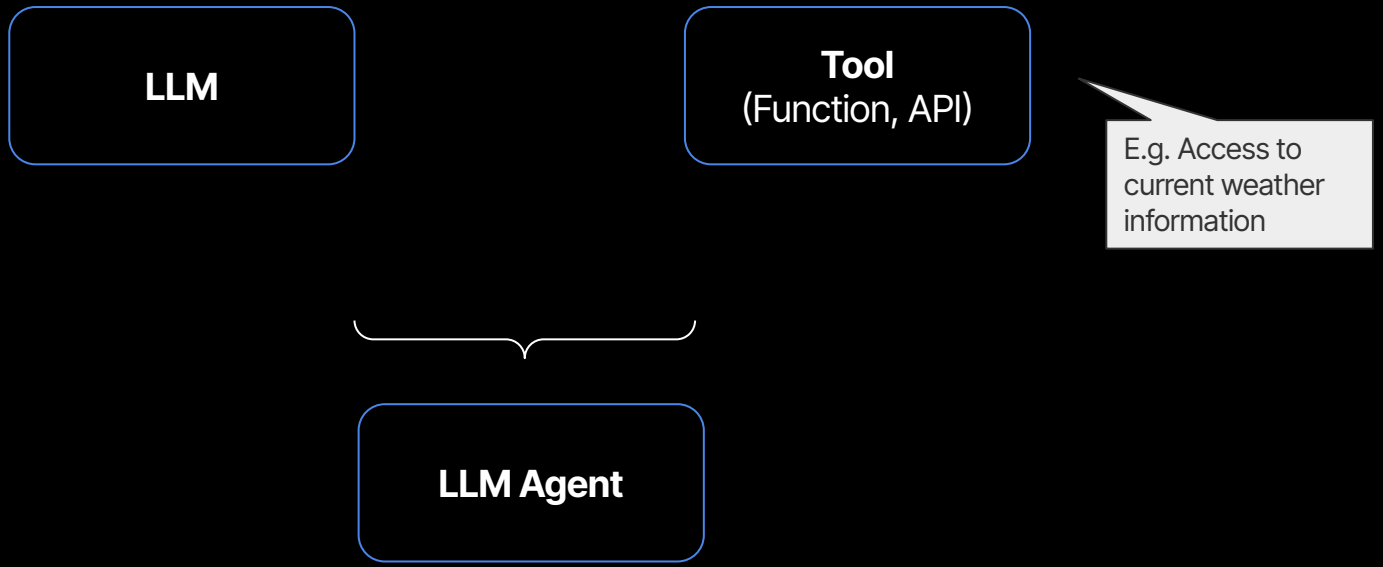
Yes.

How?

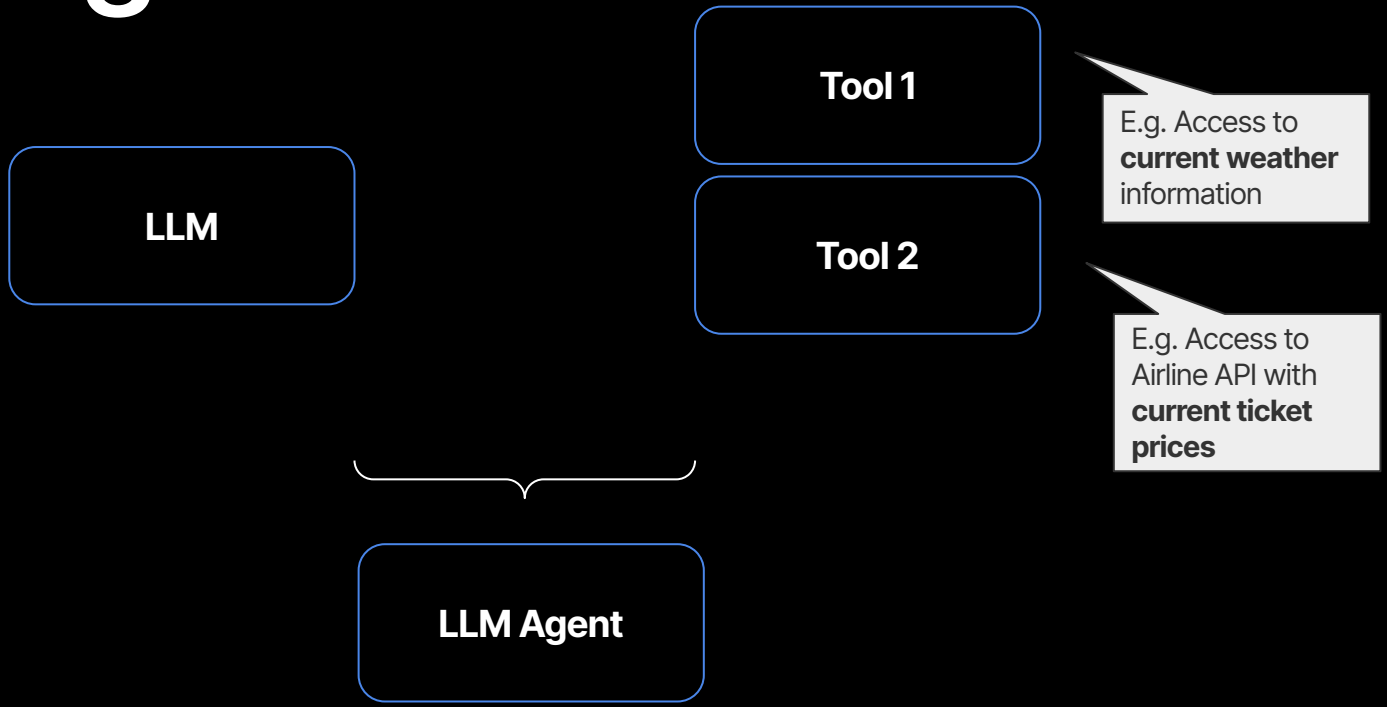
Nowadays there are LLMs that handle not only text but also **function calls** (logic in a function, API etc).



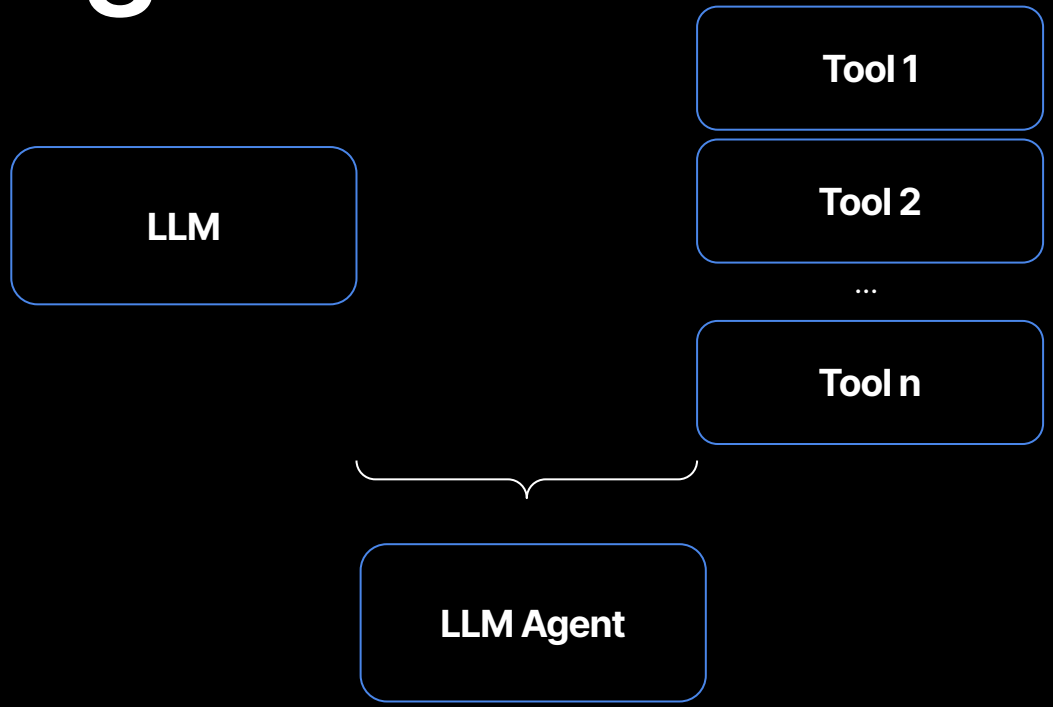
LLM Agents


LLM
Tool
(Function, API)
E.g. Access to current weather information
LLM Agent

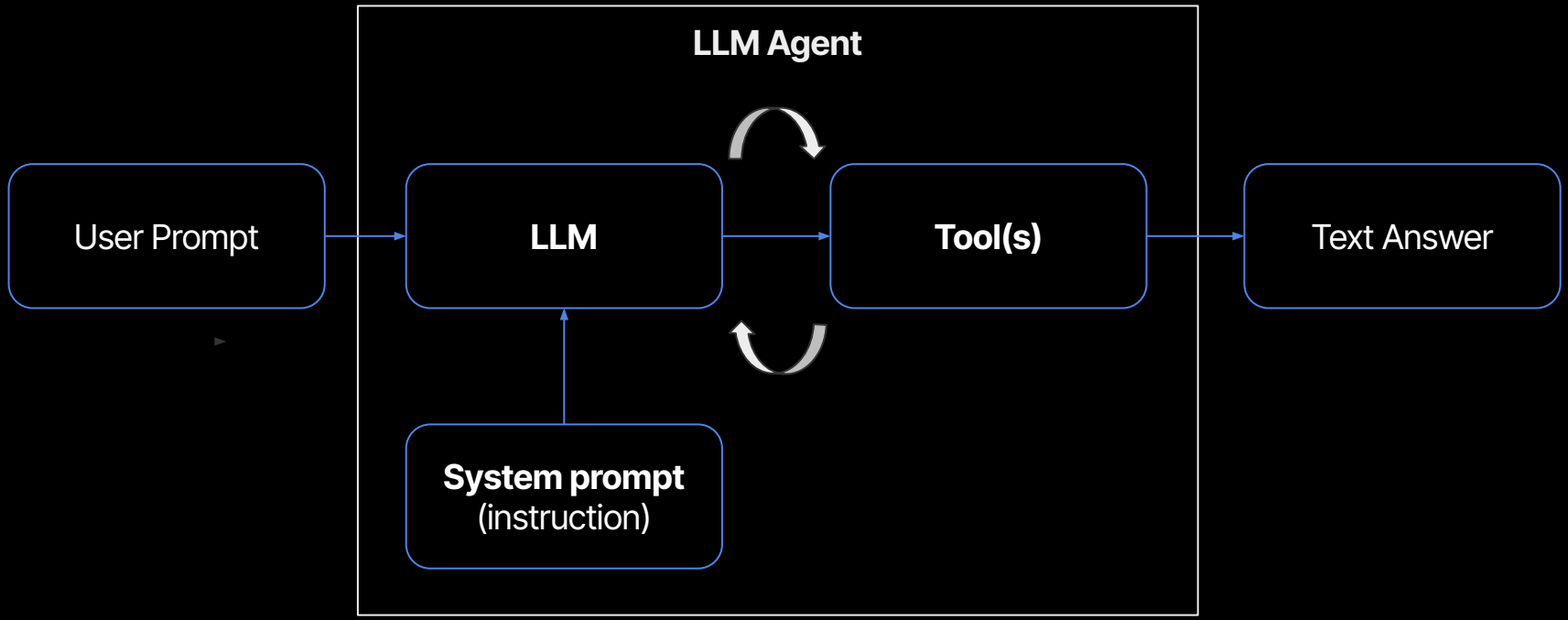
LLM Agents


Tool 1
Tool 2
E.g. Access to **current weather** information
E.g. Access to Airline API with **current ticket prices**
LLM
LLM Agent

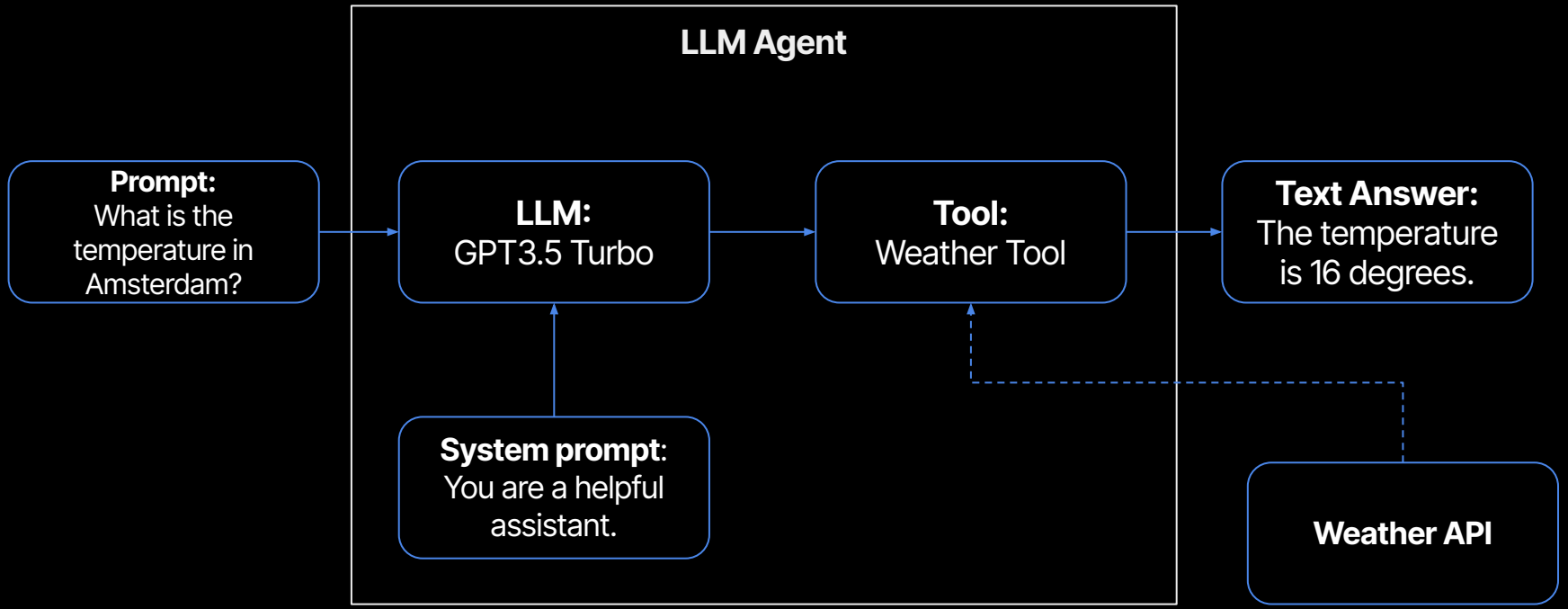
LLM Agents



LLM Agents

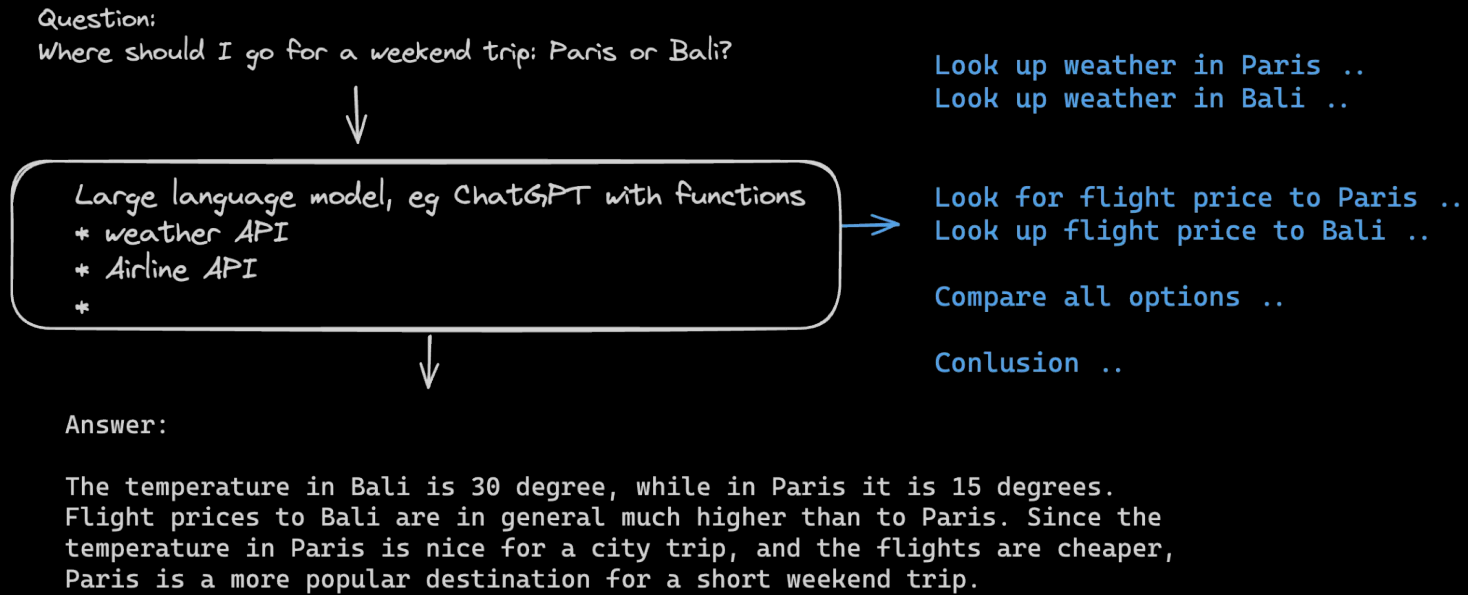

LLM Agent
User Prompt
LLM
Tool(s)
Text Answer
System prompt
(instruction)

LLM Agents


LLM Agent
Prompt:
What is the temperature in Amsterdam?
LLM:
GPT3.5 Turbo
Tool:
Weather Tool
Text Answer:
The temperature is 16 degrees.
System prompt:
You are a helpful assistant.
Weather API

LLM agents

LLM choses from a list of provided functions which actions to take to complete a task.



Getting started with the workshop

The use case

Summer holidays are coming up and you still don't know where to go. Oh no!

You decide to build a tool that helps you get information on holiday locations. For example, you would like to to find out how big a specific city is, what sights are there to see, how the weather there is, and you would like to get a drawing of that place, to get a first impression. Because who does not like art?

You will implement this through an LLM agent, who has access to

- the wikipedia API,
- the virtual crossing weather API,
- the HuggingFace API to generate images.

Our toolbox

ChatGPT

ChatGPT 3.5 turbo
[API reference]

LangChain

Framework for developing LLM applications

- Integrations for all common chat models,
- Retrievers and vector stores,
- Toolkits, RAG and Agent wrappers,
- Tutorials

What are tools?

Tools are interfaces that an LLM can use to interact with the world.

Function

Input schema

Description

Create tool with `StructuredTool` class

Name

Return result directly to user?

```python
# define the function
def wikipedia_caller(query:str) ->str:
    """This function queries wikipedia through a search query."""
    return api_wrapper.run(query)

# Input parameter definition
class QueryInput(BaseModel):
    query: str = Field(description="Input search query")

# the tool description
description: str = (
        "A wrapper around Wikipedia. "
        "Useful for when you need to answer general questions about "
        "people, places, companies, facts, historical events, or other subjects. "
        "Input should be a search query."
    )


# fuse the function, input parameters and description into a tool.
my_own_wiki_tool = StructuredTool.from_function(
    func=wikipedia_caller,
    name="wikipedia",
    description=description,
    args_schema=QueryInput,
    return_direct=False,
)
```

Let’s get coding

Option 1: Local Jupyter Lab

- Clone this git repo
- Follow these steps to set up the environment
- Copy the API keys here to helper_functions/keys.py
- Open 1_workshop_tools.ipynb and start coding!

Option 2: Google Colab

- In google colab > open notebook > github > paste the repository link > click on “1_workshop_tools.ipynb”
- Copy the code here into the first cell of the notebook and compile it.
- Copy the API keys here to helper_functions/keys.py
- You are ready to start

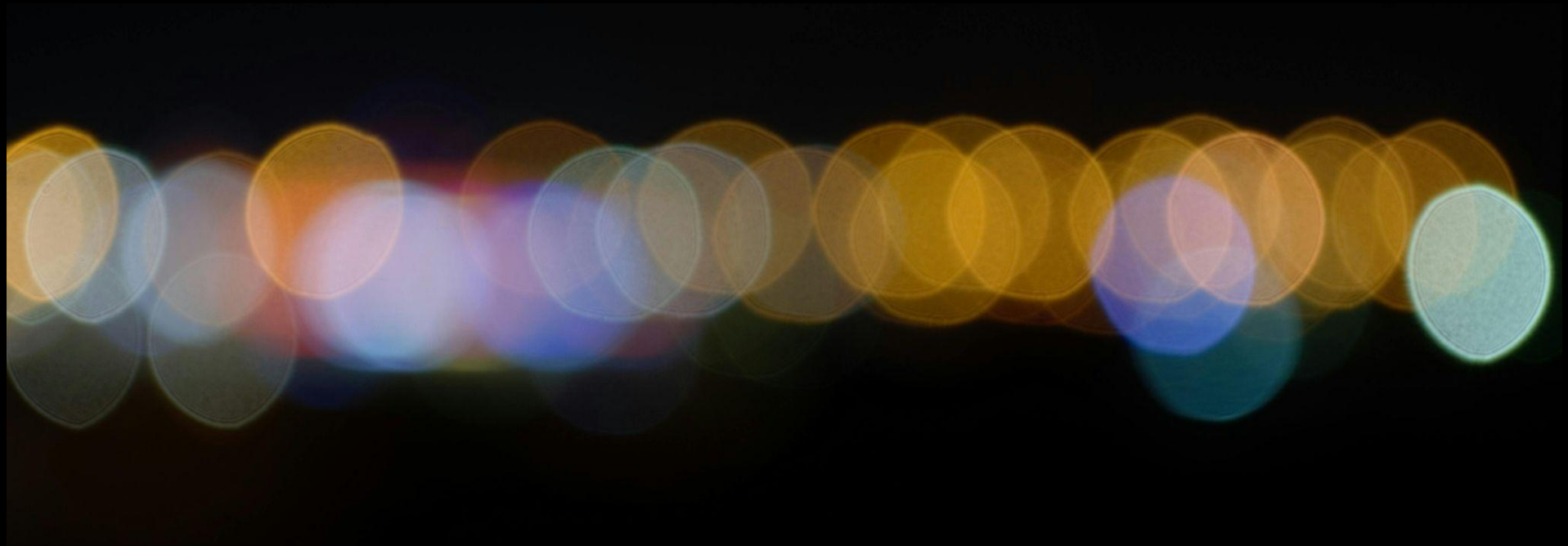
Link to the repo

Link to the slides

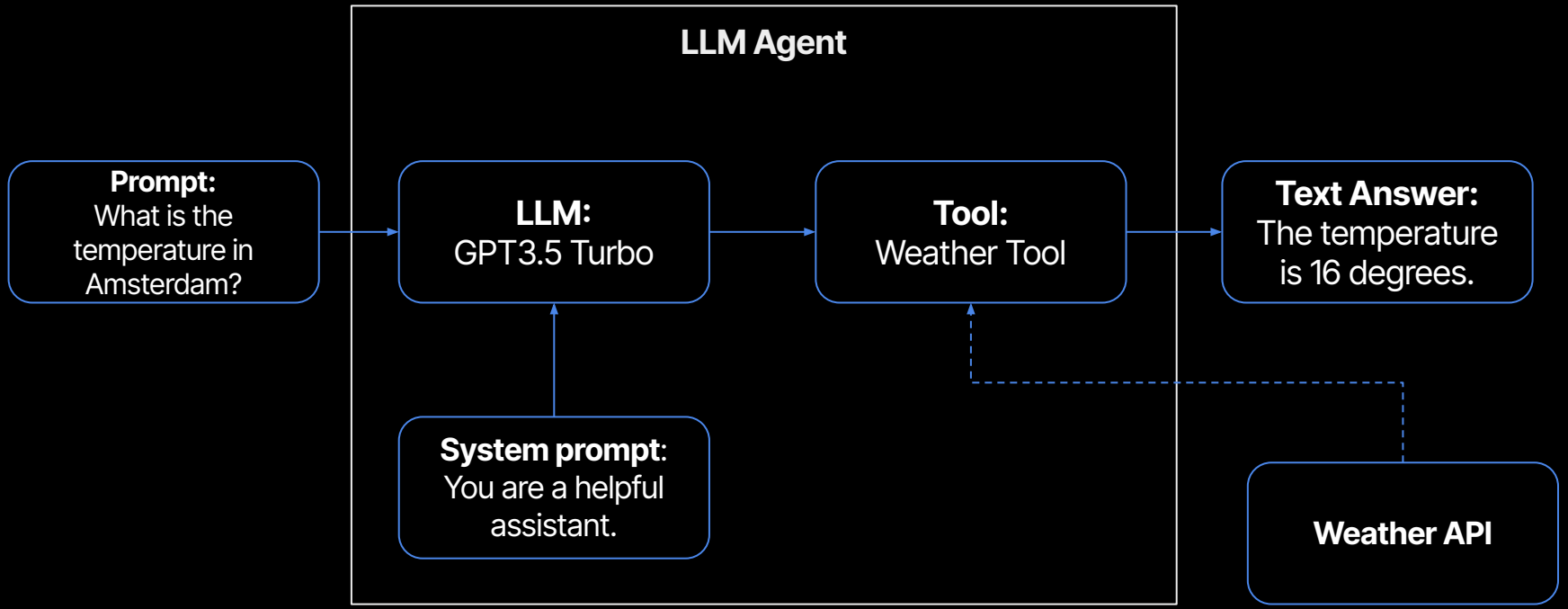
Setting up your environment

- Clone the repository with `git clone https://github.com/pyladiesams/introduction-to-llm-agents-with-langchain-jun2024`
- Set up a virtual environment using `virtualenv`:
  - `pip install virtualenv`
  - Install environment: `python3 -m venv venv`
  - Activate enviroment: `source venv/bin/activate`
  - Install dependencies in environment: `pip install -r requirements.txt`
  - Select `venv` kernel in your notebook

Time for a short break



LLM Agents


LLM Agent
Prompt:
What is the temperature in Amsterdam?
LLM:
GPT3.5 Turbo
Tool:
Weather Tool
Text Answer:
The temperature is 16 degrees.
System prompt:
You are a helpful assistant.
Weather API

Agents with LangChain

Allow and LLM to choose a sequence of actions from a list of tools.


```python
from langchain.agents import create_tool_calling_agent # set up the agent
from langchain.agents import AgentExecutor # execute agent
```

```python
# Define  the agent (load the LLM and the list of tools)
agent = create_tool_calling_agent(llm = llm, tools = tools, prompt = prompt)
agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)
```

```python
agent_executor.invoke({"input": question})
```

```python
[docs]def create_tool_calling_agent(
    llm: BaseLanguageModel, tools: Sequence[BaseTool], prompt: ChatPromptTemplate
) -> Runnable:
```

```python
    llm_with_tools = llm.bind_tools(tools)

    agent = (
        RunnablePassthrough.assign(
            agent_scratchpad=lambda x: format_to_tool_messages(x["intermediate_steps"])
        )
        | prompt
        | llm_with_tools
        | ToolsAgentOutputParser()
    )
    return agent
```

[source]

Tools, prompt, LLM and scratchpad are combined into a runnable chain.

👉 2_workshop_agents.ipynb

Why does this all work?

An LLM can either output a string response or a function call.

```
response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  tools = function_description,
  messages=messages,
  )
```

```
Question: what is the meaning of life?
Answer: Ah, the age-old question! The meaning of life is a deep philosophical ···
Function call: None
```

```
Question: How many people live in Paris?
Answer: None
Function call: [ChatCompletionMessageToolCall(id='call_kgdkOUCz5gaGbXFwhNgxvcl0',
function=Function(arguments='{"query":"Population of Paris"}', name='wikipedia'),
type='function')]
```

```
while [LLM output is function call]:
    Execute the functions
    Try to answer the questions with the function output
```

👉 3_workshop_advanced.ipynb (optional)

Thank you

Scan to connect with
Ana Chaloska

Scan to connect with
Maria Bader

pyladies amsterdam x mollie