Introduction to Agents

Yogesh Haribhau Kulkarni

YHK

Outline

Introduction

YHK

Future AI?

- What are future AI applications like?
  - Generative: Generate content like text & image
  - Agentic: Execute complex tasks on behalf of human
- How do we empower every developer to build them?:
  - Co-Pilots
  - Autonomous

YHK

Introduction to AI Agents

- 2024 is expected to be the year of AI agents.
- AI agents combine multiple components to solve complex problems.
- Shifting from monolithic models to compound AI systems.
- Compound AI systems use system design for better problem solving.
- AI agents improve with reasoning, acting, and memory components. (ReAct = Reasoning + Acting)

YHK

What is an Agent?

- Agent acts, take you from one state to the other state, provides value by workflow automation. (ReAct paper: Reasoning and Action), it can plan and make decisions.
- Agents have access to tools (ToolFormer paper) e.g. Search APIs, booking, send email etc.
- Interacting of external environment and other Agents, etc.
- Memory to keep the history of conversations/actions done so far.
- May have human-in-loop to keep it sane in the wild-world.
- Agents were there from 1950's but they are effective because of LLMs.

YHK

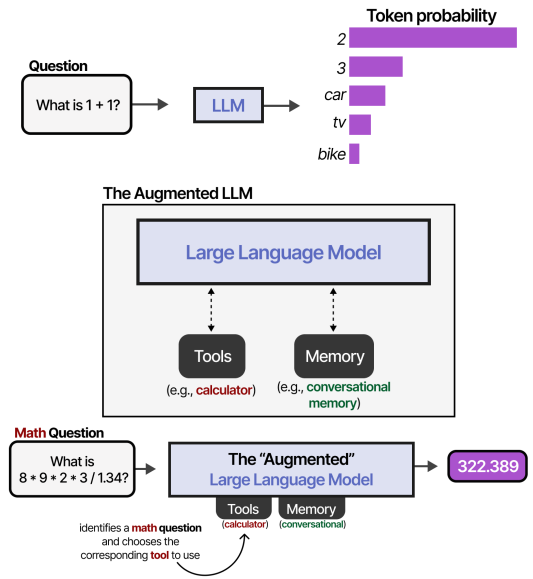
What are Agents?

- Agents are systems where LLMs dynamically direct their own processes and tool usage
- Can operate autonomously over extended periods using various tools
- Distinct from workflows: agents have dynamic control vs. predefined code paths
- Essential component in modern AI systems with varying degrees of autonomy

YHK

What Are LLM Agents?

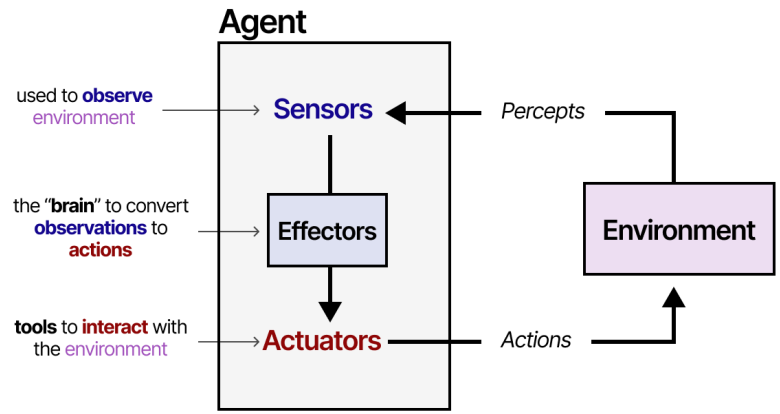
- LLMs predict the next token in a sequence—no memory.
- Conversations are simulated by sampling many tokens.
- LLMs struggle with tasks like basic arithmetic.
- Augmented LLMs use tools and memory to enhance capabilities.
- An Agent perceives and acts upon its environment.
- Agentic LLMs use text as input (sensor) and tools as actuators.
- Planning and reasoning enable agent-like behavior.
- LLM Agents vary in autonomy based on system design.



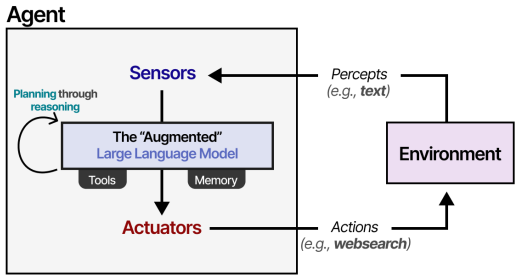
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?

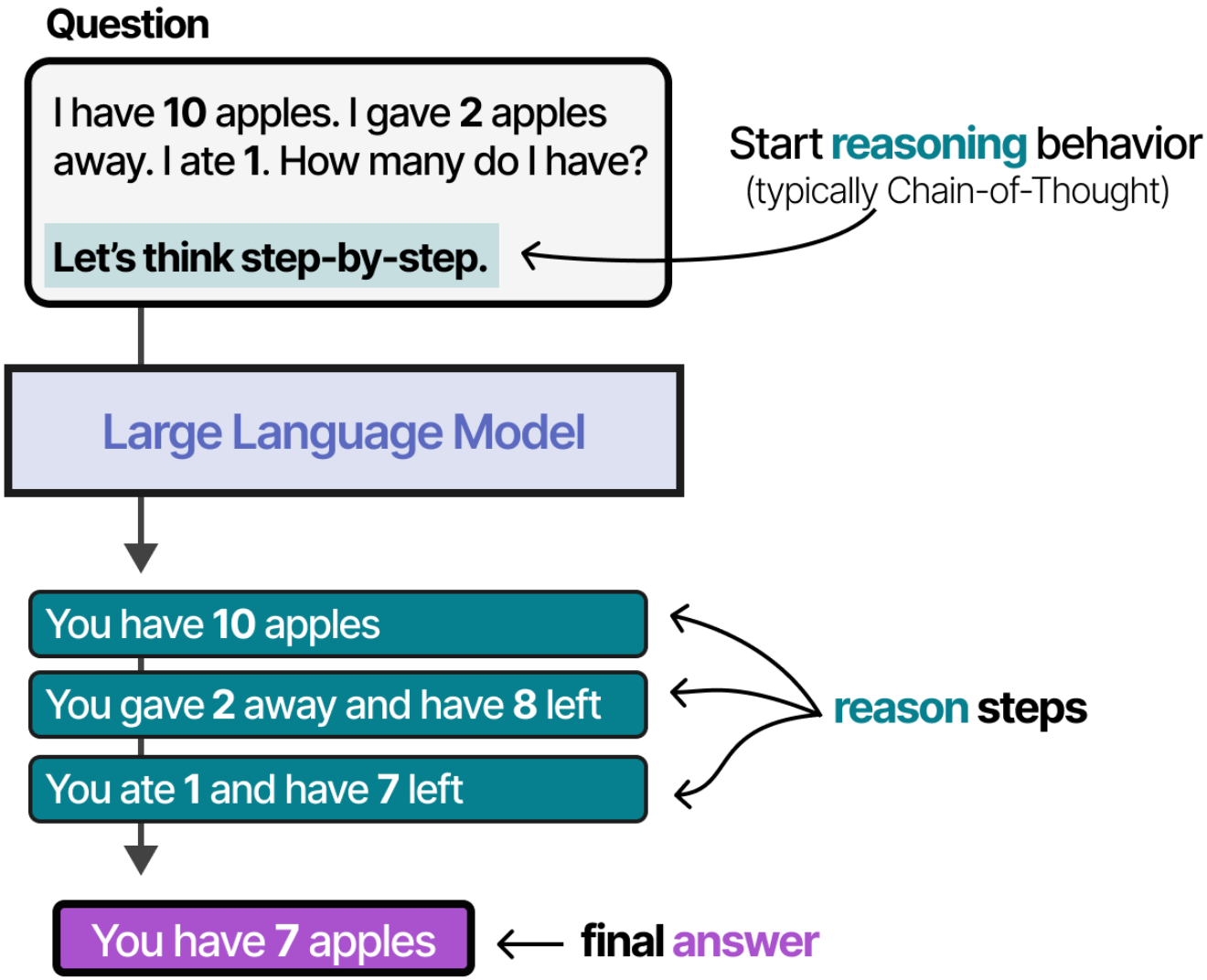


(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)



(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

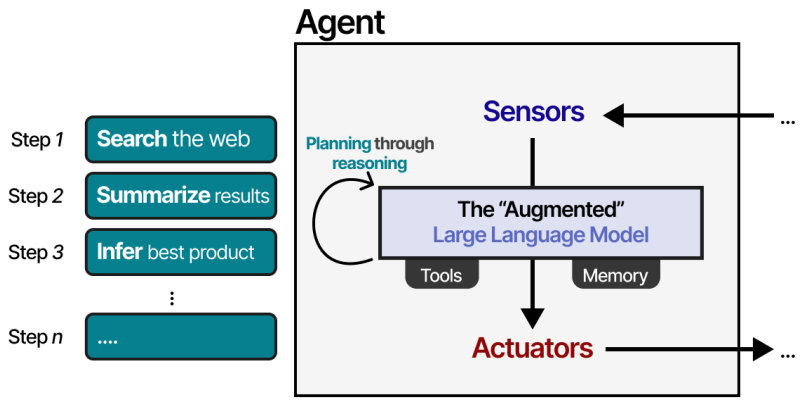
What Even Is an AI Agent?



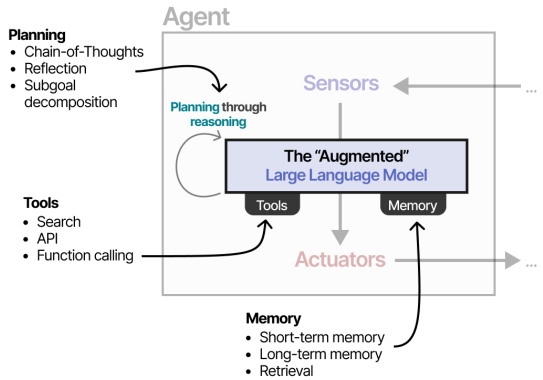
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?



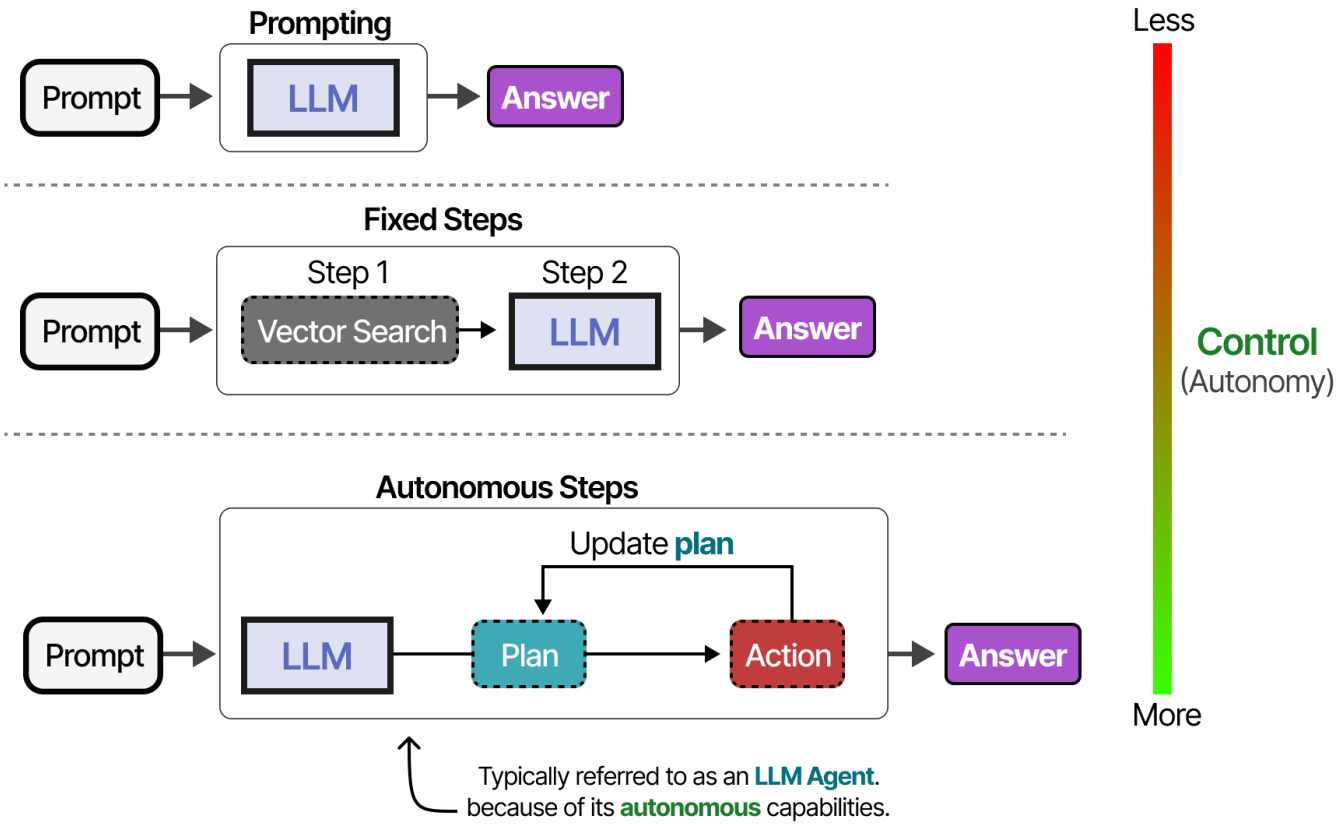
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)



(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?



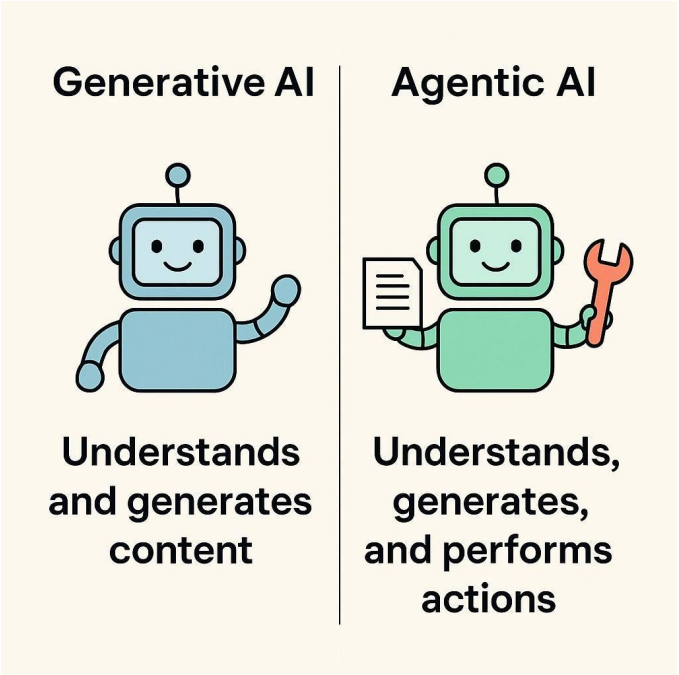
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?

No widely accepted definition exists, but here's a practical one:

- **Generative AI:** Great at understanding and generating content
- **Agentic AI:** Goes further, understands, generates content, **and performs actions**



(Ref: Agentic AI For Everyone - Aish & Kiriti)

The key differentiator is the ability to **take action**, not just respond

YHK

The Evolution of AI Capabilities

- **Traditional Programming:** Needed code to operate
- **Traditional ML:** Needed feature engineering
- **Deep Learning:** Needed task-specific training
- **ChatGPT (2022):** Could reason across tasks without training
  - Zero-shot learning (no examples needed)
  - In-context learning (understands from instructions)
- **Agents (2024):** Can actually **do things**, not just talk

YHK

Why Does "Taking Action" Matter?

- In 2022, ChatGPT was revolutionary because AI felt conversational
- By 2024, people wanted more than conversation, they wanted **execution**
- Examples of what users now expect:
  - Instead of listing leads ? **email them directly**
  - Instead of summarizing docs ? **file and create workflow tasks**
  - Instead of suggesting products ? **customize landing pages**
- This shift from **information** to **action** defines the agent era

YHK

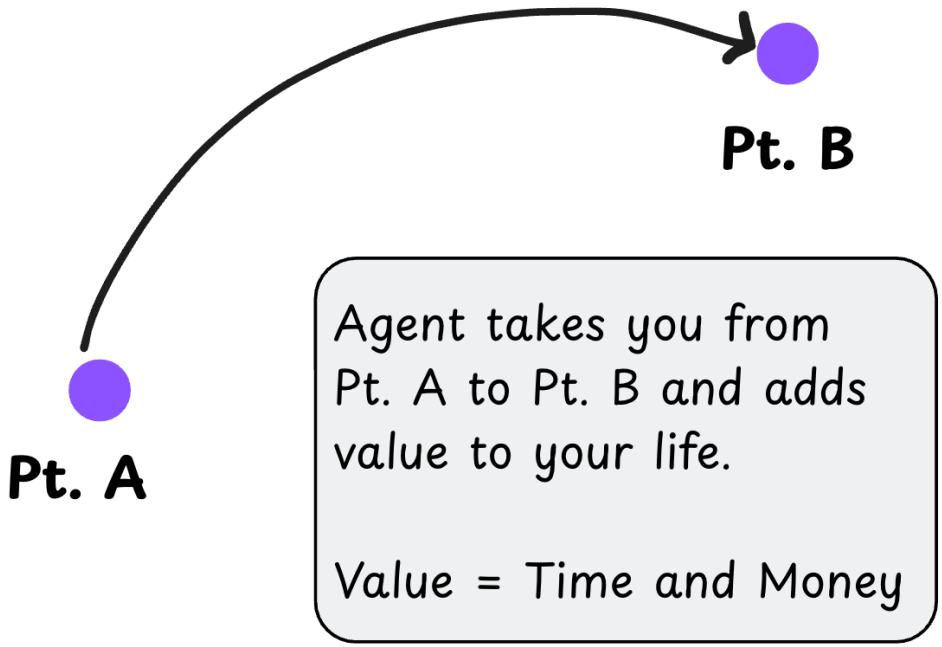
How Do Agents Take Action?

- The magic lies in **tools** and **function calling**
- Agents are paired with APIs, plugins, or external systems
- Instead of just text responses, LLMs output structured commands:
  - "Call the send_email() function with these inputs..."
  - "Fetch records from CRM using this query..."
  - "Schedule a meeting for Tuesday at 2PM..."
- **Mental model:** LLM = brain, Tools = hands
- Without tools, agents just talk. With tools, they act.

YHK

Defining AI Agents with an example

- Planning a trip involves many complex tasks
- Point A: Just discussing the trip
- Point B: All bookings and itinerary ready
- AI Agents aim to take you from A to B
- First idea: Agent adds value by saving time/money

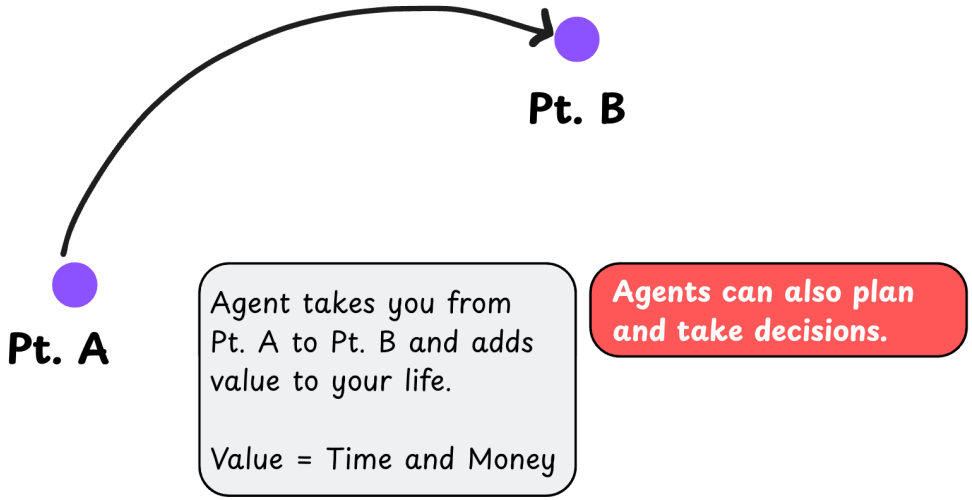


(Ref: Vizuara AI Agents Bootcamp)

YHK

Evolving Definition of Agents

- Not all tools from A to B are agents (e.g., cars)
- Agents must plan and make decisions
- Second definition includes decision-making ability
- Example: Choosing flights based on budget
- Planning daily itinerary needs contextual judgment

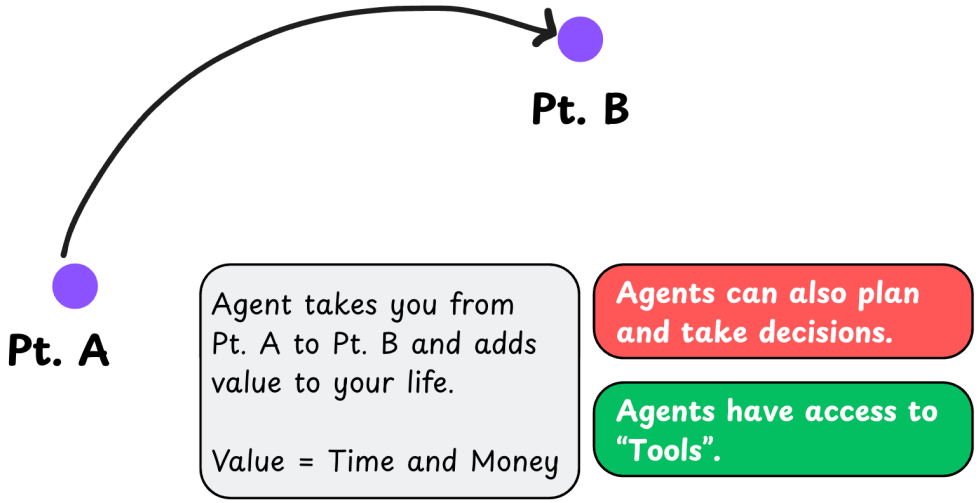


(Ref: Vizuara AI Agents Bootcamp)

YHK

Agents Need Tools

- Even self-driving cars plan but are not agents
- Agents need access to external tools
- Tools = Access to services (e.g., Gmail, Booking)
- Agents perform tasks using these tools
- Third definition adds tool access to capabilities

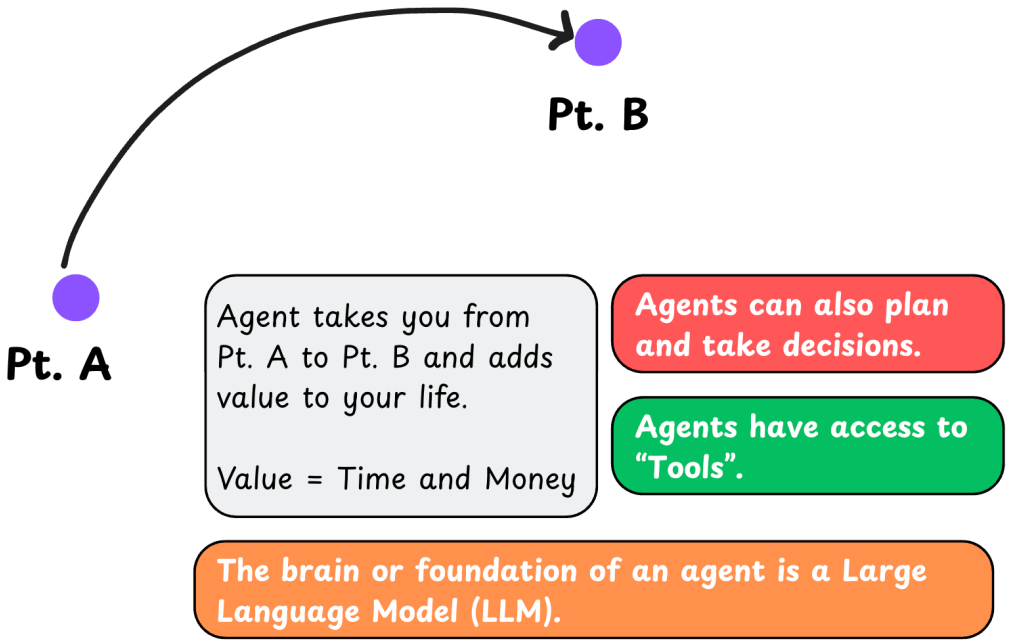


(Ref: Vizuara AI Agents Bootcamp)

YHK

Rise of LLMs in Agents

- Transformers (2017) enabled powerful LLMs
- LLMs understand and generate human language
- Agents use LLMs for reasoning and planning
- LLMs enable understanding of webpages and writing emails
- Fourth definition: Agents are LLMs with tools and planning ability



(Ref: Vizuara AI Agents Bootcamp)

YHK

What Is an Agent? (Technical Definition)

- Agent acts and takes you from one state to another, providing value through workflow automation
- Based on ReAct paradigm: **Reasoning + Acting**
- Key capabilities:
  - Can plan and make decisions
  - Has access to tools (search APIs, booking, email, etc.)
  - Interacts with external environments and other agents
  - Maintains memory of conversations and actions
  - May include human-in-the-loop for safety
- Agents existed since the 1950s but are now effective because of LLMs

YHK

Two Ways to Define Agents

Technical View:

- LLM (brain)
- + Tools (hands)
- + Planning (strategy)
- + Memory (context)
- + State management

Business View:

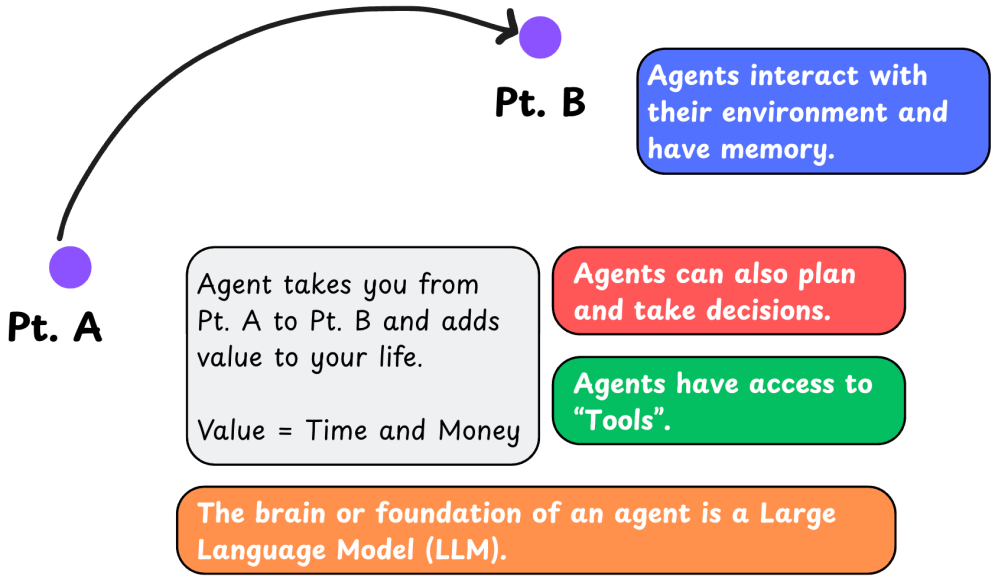
- Systems that complete tasks end-to-end
- Focus on outcomes, not components
- Solve real-world problems
- Provide measurable value

Important: Today's agents are **engineering wrappers** around AI models, the intelligence comes from the LLMs, agents help act on that intelligence.

YHK

Final Definition of Agents

- Agents can learn from feedback and environment
- Agents interact with tools, humans, and websites
- They improve with experience (memory)
- Fifth definition: LLMs + Tools + Planning + Learning
- Agents evolve over time via memory and feedback



(Ref: Vizuara AI Agents Bootcamp)

YHK

Understanding Agency

- Agency = Level of autonomy an agent has
- Low agency → less value
- High agency → high value
- More autonomous agents can handle complex tasks
- Agency is key to measuring agent usefulness

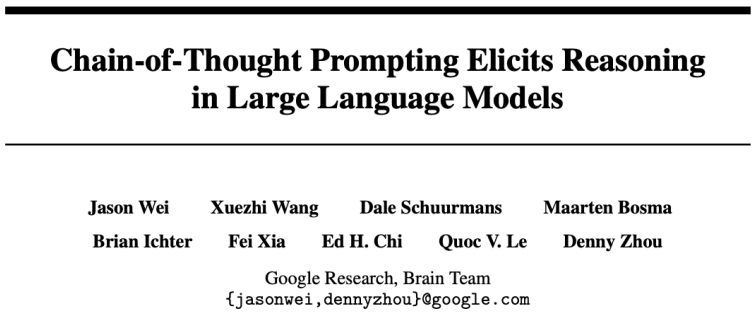
Agency Level	Description	Name	Example
●○○○○	Agent does not influence what happens next	Simple Processor	Grammar checker that rewrites sentences
●●○○○	Agent determines basic control flow	Router	Customer Query → Tech Support or Sales
●●●○○	Agent determines function execution	Tool Caller	A smart calendar assistant that spots "let's meet on Tuesday" and books the meeting
●●●●○	Lays out a short plan and carries it step by step	Multi-step Agent	Personal travel planner that gathers flight options, hotels, local activities
●●●●●	One agentic workflow starts another agentic workflow	Multi-agent	Travel planner agent → Booking agent ← Email agent

(Ref: Vizuara AI Agents Bootcamp)

YHK

Papers that Shaped AI Agents

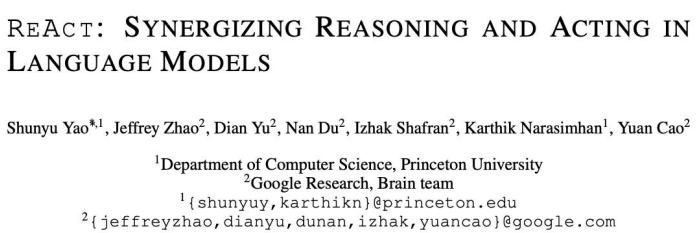
- Core research papers laid the foundation
- Introduced key frameworks and architectures
- Sparked recent boom in agent development
- Include Transformer and Agentic frameworks
- Major driving force in LLM-based agent systems



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

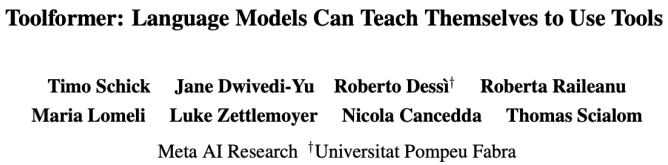
Google Research, Brain Team
{jasonwei,dennyzhou}@google.com



REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao*,1, Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com



Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì† Roberta Raileanu
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research †Universitat Pompeu Fabra

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park
Stanford University
Stanford, USA
joonspk@stanford.edu

Joseph C. O'Brien
Stanford University
Stanford, USA
jobrien3@stanford.edu

Carrie J. Cai
Google Research
Mountain View, CA, USA
cjcai@google.com

Meredith Ringel Morris
Google DeepMind
Seattle, WA, USA
merrie@google.com

Percy Liang
Stanford University
Stanford, USA
pliang@cs.stanford.edu

Michael S. Bernstein
Stanford University
Stanford, USA
msb@cs.stanford.edu

(Ref: Vizuara AI Agents Bootcamp)

YHK

When to Use Agents?

- Best suited for tasks requiring flexibility and model-driven decision-making
- Consider tradeoffs: agents increase latency and cost for better task performance
- Recommended for open-ended problems with unpredictable steps
- Simple solutions preferred - single LLM calls with retrieval often sufficient

YHK

Future AI Applications

- What are future AI applications like?
  - **Generative:** Generate content like text and images
  - **Agentic:** Execute complex tasks on behalf of humans
- How do we empower every developer to build them?
  - **Co-Pilots:** Human-AI collaboration
  - **Autonomous:** Independent task execution
- 2024 is expected to be the year of AI agents

YHK

Agents using SmolAgents

YHK

SmolAgents at a Glance: Many Agents, One Framework

- Unified framework with multiple agent types and capabilities
- Vision + Browser Agents interpret visual data and navigate web
- Code Agents write Python code as actions for complex tasks
- Tool-Calling Agents use structured JSON calls or text instructions
- Multi-Agents orchestrate multiple agents in hierarchical workflows
- Retrieval Agents use RAG to pull information from knowledge bases
- Built-in tools: web search, Python interpreter, speech-to-text
- Customizable tool functions enable real-world agent actions

YHK

Code vs JSON: SmolAgent's Code-First Philosophy

- Agents call tools by writing code instead of filling JSON schemas
- Traditional frameworks use structured JSON: {"action": "Search", "input": "tutorials"}
- Code-first offers full Python logic: loops, branches, complex operations
- Multiple tool calls in one go reduces iteration cycles by 30%
- Fewer reasoning steps while achieving better results than JSON approaches
- Code provides visible thought process, simplifying debugging
- JSON option available via ToolCallingAgent for simpler tasks
- Safety handled through secure interpreters and sandboxes

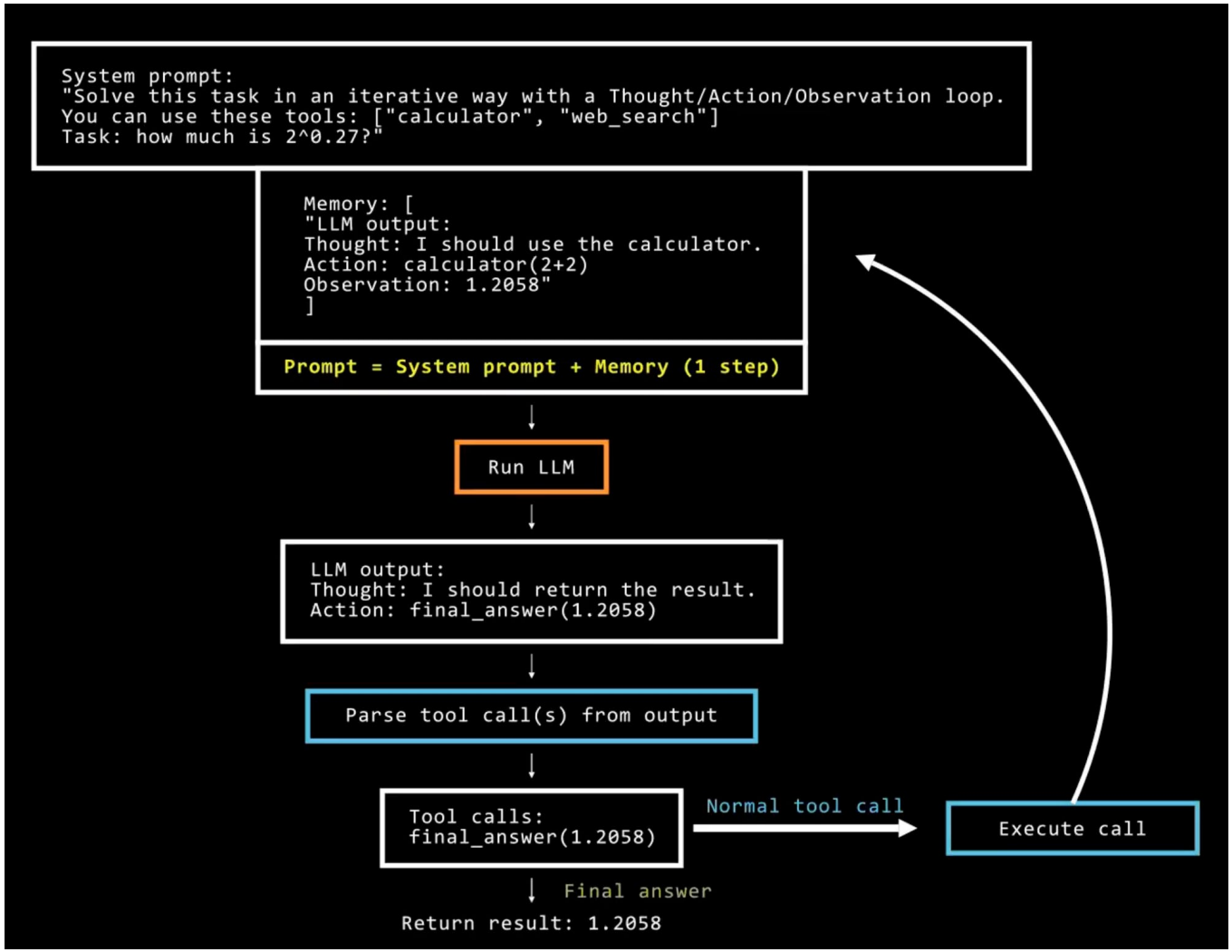
YHK

Inside the CodeAgent Loop: How Does It Work?

- Agent receives task and initializes system prompt with available tools
- LLM generates next action as executable Python code snippet
- Generated code runs and executes tool calls, returning results
- Results captured and appended to agent's conversation history
- Agent uses updated context to decide next action iteratively
- Agent calls final_answer(answer_text) when task complete
- Memory ensures continuity, prevents repetition and contradictions
- Error handling allows reconsideration when code bugs or failures occur
- Follows ReAct paradigm (Reason + Act) with code-based actions

YHK

Inside the CodeAgent Loop: How Does It Work?



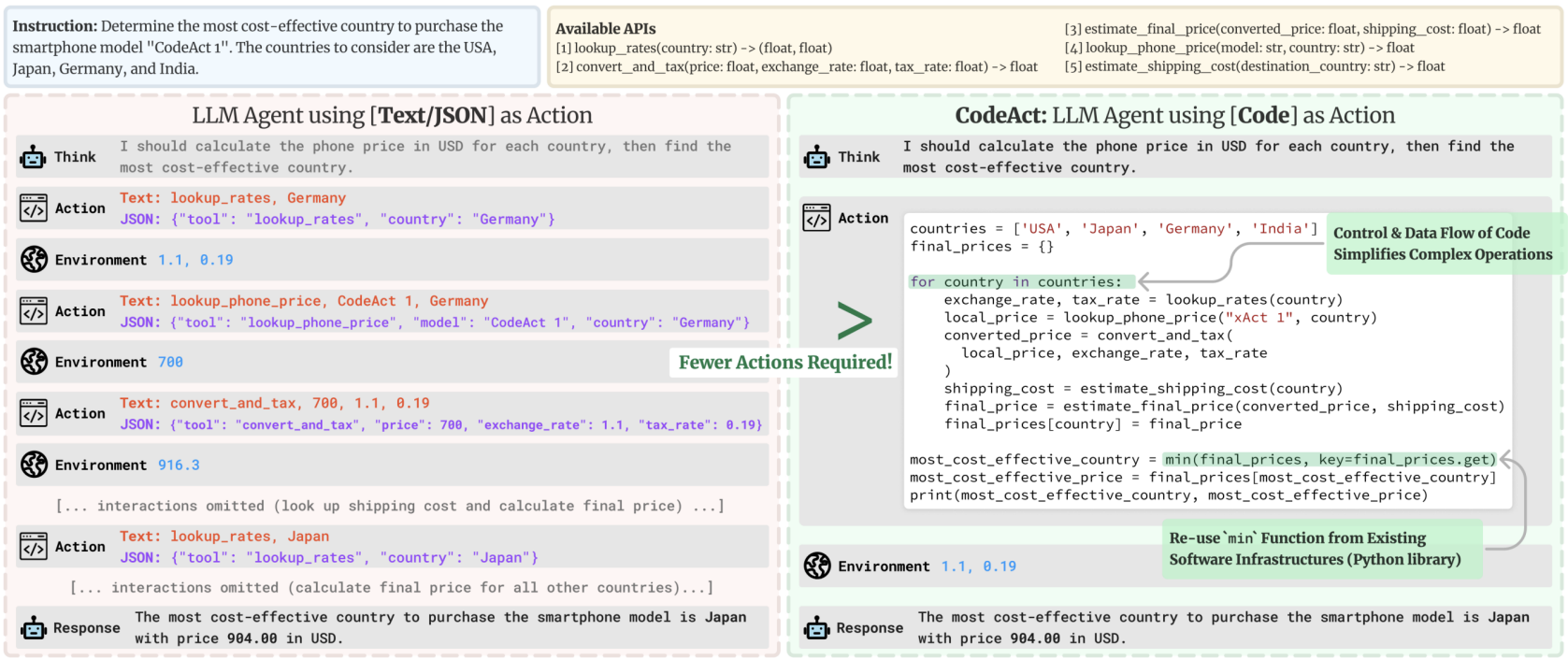
(Ref: Vizuara AI Agents Bootcamp Day 6)

ToolCallingAgent vs CodeAgent: A Quick Comparison

- Both use same underlying multi-step ReAct logic
- ToolCallingAgent uses structured JSON calls for predictable interactions
- CodeAgent enables complex logic and multi-step operations in single thought
- ToolCallingAgent better for quick prototyping and safe environments
- CodeAgent superior for complex sequences requiring full programming
- Both accomplish complex tasks but serve different use cases
- Framework allows choosing between styles based on specific needs
- Code approach requires safety considerations but offers greater capability

YHK

ToolCallingAgent vs CodeAgent



(Ref: Writing actions as code snippets or JSON blobs - Hugging Face)

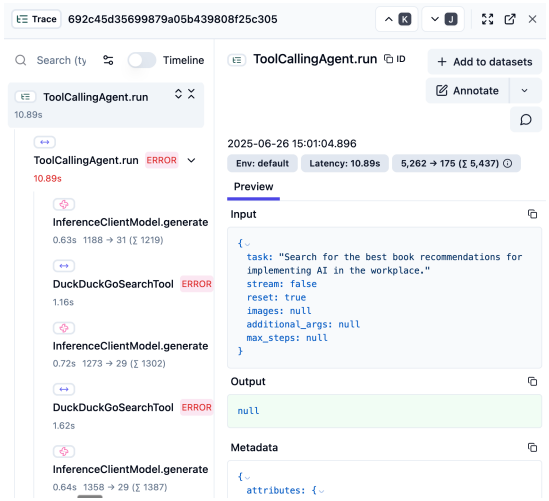
Vision Agents

- Vision Agents accept and process image inputs
- Analyze visual data: charts, screenshots, webpage images
- Enable UI automation by interpreting visual elements
- Support vision models for image-based tasks
- Control web browsers by interpreting screenshots and page elements
- Combine visual perception with other tools in agent toolbox
- Enable multimodal AI interactions beyond text-only processing

YHK

LangFuse: Observability for LLM Agents

- Observability platform tailored for LLM agents
- Enables tracking, visualizing, and debugging agent outputs
- Logs agent actions, tool calls, and reasoning steps
- Provides insights into agent behavior and performance bottlenecks
- Helps understand logic flow and identify improvement areas
- Essential companion for building robust, transparent AI agents
- Facilitates effortless monitoring and debugging of agent workflows



(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

Basic Agent Setup: CodeAgent

- Minimal agent requires model and tools arguments
- HfApiModel leverages Hugging Face Inference API
- CodeAgent executes Python code snippets at each step
- add_base_tools=True includes default toolbox
- Built-in safety with predefined safe functions only
- Execution stops on illegal operations or Python errors

```python
from smolagents import CodeAgent, HfApiModel

# Initialize model using HF API
model = HfApiModel(
    model_id="meta-llama/Llama-3.3-70B-Instruct"
)

# Create agent with default tools
agent = CodeAgent(
    tools=[],
    model=model,
    add_base_tools=True
)

# Run agent with task
result = agent.run(
    "Calculate the 118th Fibonacci number"
)
```

YHK

Custom Tool Creation: @tool Decorator

- @tool decorator converts functions into agent tools
- Requires clear name, type hints, and description
- Args section describes each parameter for LLM
- Tool description baked into agent's system prompt
- Same format as apply_chat_template tool schemas

```python
from smolagents import tool, CodeAgent, HfApiModel
from huggingface_hub import list_models

@tool
def model_download_tool(task: str) -> str:
    """
    Returns most downloaded model for given task.

    Args:
        task: Task for download count.
    """
    most_downloaded_model = next(iter(
        list_models(filter=task, sort="downloads",
                    direction=-1)
    ))
    return most_downloaded_model.id

# Use custom tool
agent = CodeAgent(
    tools=[model_download_tool],
    model=HfApiModel()
)
result = agent.run(
    "Most downloaded text-to-video model?"
)
```

YHK

Multi-Agent Systems: ManagedAgent

- Hierarchical multi-agent systems for better specialization
- ManagedAgent encapsulates agents as tools for manager
- Requires agent, name, and description arguments
- Separate tool sets and memories for efficient specialization
- Manager agent coordinates specialized sub-agents
- Better performance on complex benchmarks

```python
from smolagents import (CodeAgent, HfApiModel,
                        DuckDuckGoSearchTool, ManagedAgent)

model = HfApiModel()

# Create specialized web search agent
web_agent = CodeAgent(
    tools=[DuckDuckGoSearchTool()],
    model=model
)

# Wrap as managed agent
managed_web_agent = ManagedAgent(
    agent=web_agent,
    name="web_search",
    description="Runs web searches. "
                "Give query as argument."
)

# Create manager agent
manager_agent = CodeAgent(
    tools=[],
    model=model,
    managed_agents=[managed_web_agent]
)

result = manager_agent.run(
    "Who is the CEO of Hugging Face?"
)
```

YHK

ToolCallingAgent vs CodeAgent Examples

- ToolCallingAgent uses JSON-like action blobs
- CodeAgent executes Python code for actions
- Same initialization pattern, different execution styles
- ToolCallingAgent safer, CodeAgent more flexible
- additional_authorized_imports for CodeAgent only

```python
from smolagents import (CodeAgent, ToolCallingAgent,
                        HfApiModel)

model = HfApiModel()

# ToolCallingAgent - JSON-style actions
tool_agent = ToolCallingAgent(
    tools=[],
    model=model
)
result1 = tool_agent.run(
    "Get title of https://huggingface.co/blog"
)

# CodeAgent - Code execution with imports
code_agent = CodeAgent(
    tools=[],
    model=model,
    additional_authorized_imports=[
        'requests', 'bs4'
    ]
)
result2 = code_agent.run(
    "Get title of https://huggingface.co/blog"
)
```

YHK

Agent Inspection and Gradio UI

- agent.logs stores fine-grained execution logs
- write_inner_memory_from_logs() creates LLM-viewable memory
- GradioUI provides interactive chat interface
- reset=False maintains conversation memory
- Visualize agent thoughts and execution process

```python
from smolagents import (CodeAgent, HfApiModel,
                        GradioUI, load_tool)

# Load tool from Hub
image_tool = load_tool("m-ric/text-to-image")

# Initialize agent
model = HfApiModel()
agent = CodeAgent(
    tools=[image_tool],
    model=model
)

# Launch interactive interface
GradioUI(agent).launch()

# Inspect agent after run
print(agent.logs)  # Fine-grained logs
agent.write_inner_memory_from_logs()
# LLM-viewable memory

# Continue conversation
agent.run("Follow up question", reset=False)
```

YHK

Multi-Agents using SmolAgents

YHK

Why Multi-Agent Systems? From Solo to Teamwork

- Single agents handling everything aren't always ideal for complex tasks
- Multi-agent systems enable specialized agents working together toward common goals
- Each agent focuses on specific sub-tasks or roles with clear responsibilities
- Agents communicate and coordinate like an AI team
- Brings teamwork concepts to AI agents for better performance
- More robust and performant than monolithic single agents

YHK

Key Advantages of Multi-Agent Systems

- **Divide and Conquer:** Each agent has single responsibility, simplifying complex problems
- **Parallelism & Speed:** Agents work in parallel or sequentially with less overhead
- **Improved Focus & Memory:** Each agent maintains its own context, reducing prompt size
- **Fault Tolerance:** If one agent fails, others can catch issues or request retries
- **Specialization:** Different skill sets assigned to different agents (researcher, calculator, planner)
- **Cost Efficiency:** Smaller prompts per agent reduce latency and token costs

YHK

Case Study: Harry Potter Trip Planner

- Complex task: Plan worldwide tour of Harry Potter filming locations
- Added twist: Include nearby cricket stadiums for sports fans
- Calculate travel times from Pune, India to each location
- Multiple sub-tasks requiring different specialized skills
- Perfect example of multi-step planning problem
- Demonstrates why single vs multi-agent approaches matter

(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

Single Agent Approach: Tools and Challenges

- **Tools Used:** Web Search, Webpage Reader, Travel Time Calculator, Mapping Library
- Agent equipped with all necessary capabilities for end-to-end solution
- Successfully gathered data but hit significant limitations
- **Long Context Issue:** Prompt grew very large with all information
- **High Token Usage:** Expensive and slow due to context bloat
- **Complex Reasoning:** Juggling multiple tasks in one sequence increased errors
- Agent struggled to complete task within reasonable steps and token limits

YHK

Single Agent Approach: Tools and Challenges

```
agent = CodeAgent(
    model=model,
    tools=[GoogleSearchTool(), VisitWebpageTool(), calculate_cargo_travel_time],
    additional_authorized_imports=["pandas"],
    max_steps=20,
)
```

```
[ ] result = agent.run(task)
```

```
New run

Find all Harry Potter filming locations in the world, calculate the time to transfer via passenger plane to
here (we're in Pune, 18.5204° N, 73.8567° E), and return them to me as a pandas dataframe.
Also give me some cricket stadiums if there are any nearby to the filming locations

InferenceClientModel - Qwen/Qwen2.5-Coder-32B-Instruct
Step 1
```

```
result
```

	Location	Latitude	Longitude	Travel Time to Pune	Nearby Cricket Stadiums
0	Leavesden Studios	51.690144	-0.418115	9:03:36	Abbots Langley Cricket Club, Everett Rovers, W...
1	Hogwarts Express station	51.536600	-0.141140	9:02:24	Lord's Cricket Ground
2	Edinburgh Castle	55.948610	-3.200830	9:15:36	Grange Cricket Club
3	Gloucester Cathedral	51.867590	-2.246770	9:12:00	Seat Unique Stadium
4	Lacock Abbey	51.414750	-2.117180	9:11:24	Lacock Rec
5	Alnwick Castle	55.415583	-1.705920	9:09:36	Alnwick Cricket Club
6	Durham Cathedral	54.773500	-1.575770	9:09:00	Emirates Riverside, Durham University Ground
7	Hampton Court Palace	51.403330	-0.337500	9:03:00	Hampton Court Green, East Molesey Cricket Club
8	Elstree Borehamwood Studios	51.658190	-0.269030	9:03:00	Lord's Cricket Ground
9	London Film Studios	51.552000	-0.097000	9:01:48	Lord's Cricket Ground, Queen's Cricket Club

(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

Multi-Agent Solution: Division of Labor

- **Browser Agent:** Specialist for web searching and reading tasks
- Responsible for finding filming locations and nearby stadiums
- Has access to internet tools and travel time calculator
- Returns structured data without worrying about final report
- **Manager Agent:** High-level project manager and finalizer
- Delegates work to browser agent and processes results
- Handles map generation and final output compilation
- Uses GPT-4 level model for analysis and output generation

YHK

Multi-Agent Solution: Division of Labor

CodeAgent | gpt-4o
├── ✅ *Authorized imports:* ['geopandas', 'plotly', 'plotly.io', 'plotly.express', 'shapely', 'json', 'pandas', 'numpy', 'kaleido']
├── 🛠️ *Tools:*

Name	Description	Arguments
calculate_cargo_travel_time	Calculate the travel time for a cargo plane between two points on Earth using great-circle distance.	origin_coords (`array`): Tuple of (latitude, longitude) for the starting point destination_coords (`array`): Tuple of (latitude, longitude) for the destination cruising_speed_kmh (`number`): Optional cruising speed in km/h (defaults to 750 km/h for typical cargo planes)
final_answer	Provides a final answer to the given problem.	answer (`any`): The final answer to the problem

└── 🤖 *Managed agents:*
    └── web_agent | CodeAgent | Qwen/Qwen2.5-Coder-32B-Instruct
        ├── ✅ *Authorized imports:* []
        ├── 📝 *Description:* Browses the web to find information
        └── 🛠️ *Tools:*

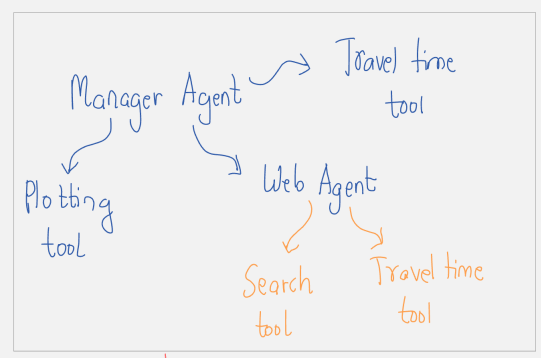
Name	Description	Arguments
web_search	Performs a google web search for your query then returns a string of the top search results.	query (`string`): The search query to perform. filter_year (`integer`): Optionally restrict results to a certain year
visit_webpage	Visits a webpage at the given url and reads its content as a markdown string. Use this to browse webpages.	url (`string`): The url of the webpage to visit.
calculate_cargo_travel_time	Calculate the travel time for a cargo plane between two points on Earth using great-circle distance.	origin_coords (`array`): Tuple of (latitude, longitude) for the starting point destination_coords (`array`): Tuple of (latitude, longitude) for the destination cruising_speed_kmh (`number`): Optional cruising speed in km/h (defaults to 750 km/h for typical cargo planes)
final_answer	Provides a final answer to the given problem.	answer (`any`): The final answer to the problem

(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

Multi-Agent Benefits: Performance and Efficiency

- **Focused Context:** Each agent maintains smaller, specialized context
- Browser agent only keeps web search results and related info
- Manager agent starts with high-level view, doesn't need search details
- **Better Performance:** Each agent specialized and efficient at its task
- **Lower Token Usage:** Smaller prompts per agent reduce cost and latency
- Successful implementation of complex trip planner
- Generated comprehensive DataFrame and interactive map visualization



(Ref: Vizuara AI Agents Bootcamp Day 6)

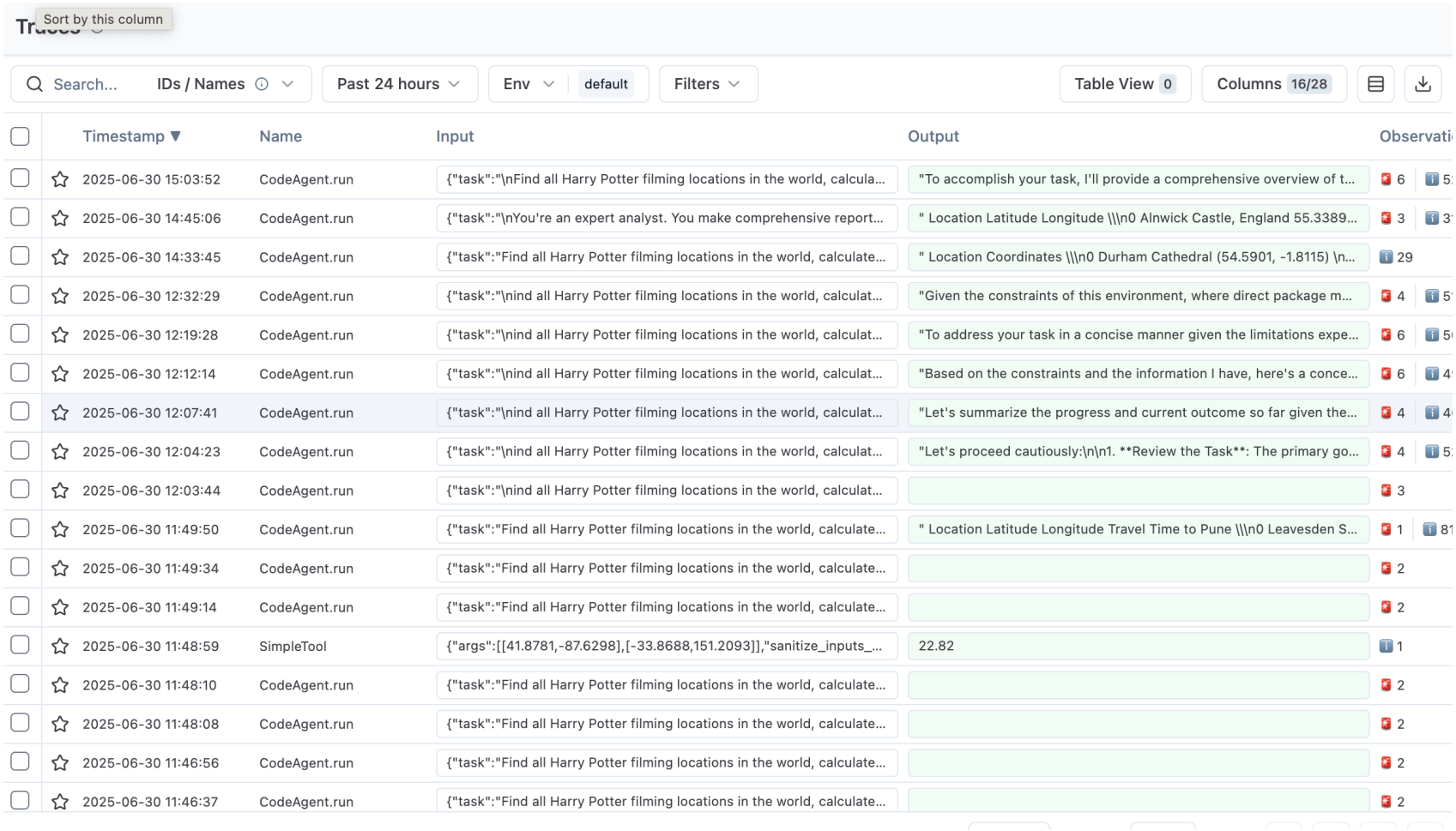
YHK

Langfuse: Tracing and Observability

- Langfuse serves as control center and black box recorder for AI agents
- **Tracing:** Records sequence of operations and model calls in each task
- Logs agent messages, tool usage, intermediate results, and final outputs
- Provides timeline view showing steps in chronological order
- Essential for debugging when results are wrong or suboptimal
- Can pinpoint which step led agent astray in complex workflows
- Integration via environment variables and SDK with OpenTelemetry

YHK

Langfuse: Tracing and Observability

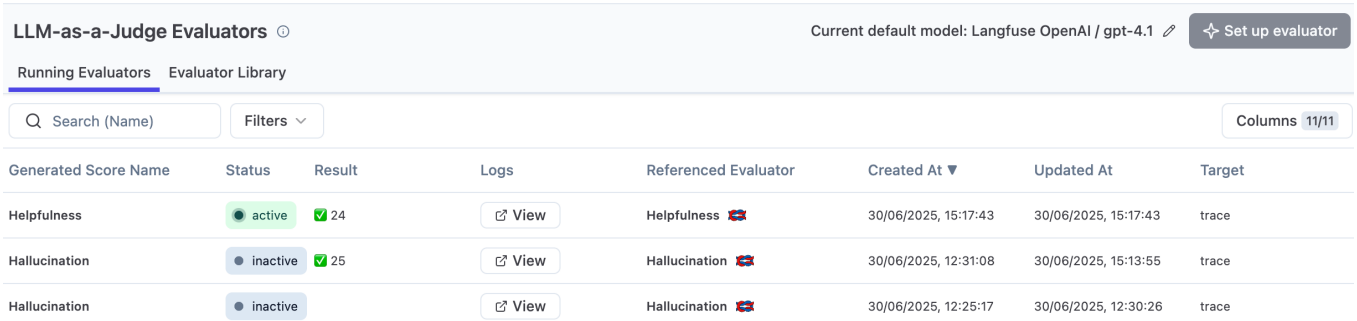


(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

Building Evaluation Dashboard

- Langfuse enables building evaluation dashboard on top of traced data
- Define metrics to score each agent run and view aggregate analytics
- **Key Metrics:** Hallucinations, Trustworthiness, Relevance, Correctness
- **Efficiency Tracking:** Tokens used and execution time per step
- Built-in support for model-based evaluation using LLM as judge
- Visual summary shows performance across many runs
- Filtering capabilities to investigate specific failures or improvements



(Ref: Vizuara AI Agents Bootcamp Day 6)

YHK

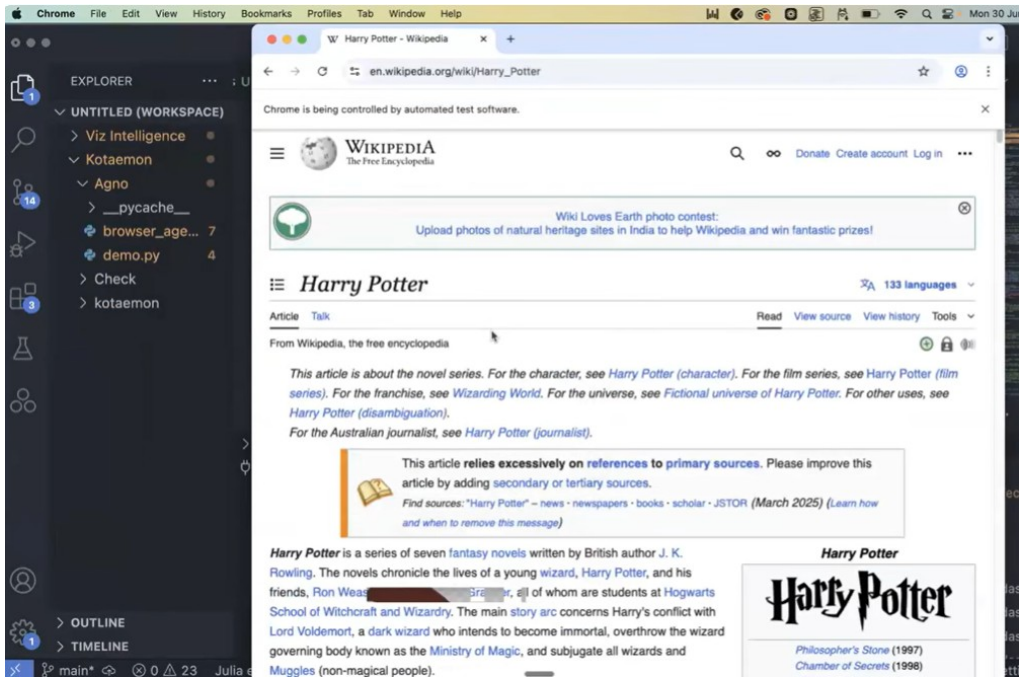
Quality Metrics for Agent Evaluation

- **Hallucinations (Faithfulness):** Check if agent outputs information not grounded in sources
- **Trustworthiness:** Measure reliability and correctness of answers
- Integration with external evaluators like Cleanlab's Trustful LLM
- **Relevance:** Ensure outputs actually relevant to user's query
- **Correctness & Completeness:** Verify all locations found and correctly matched
- Custom metrics for domain-specific checks supported
- Automated evaluation prompts check answers against retrieved context

YHK

Browser Agents: AI with Internet Superpowers

- Specialized agents designed to interact with the web
- Can perform searches, click links, scrape and read webpages
- Extract information from online sources in real-time
- Serve as eyes and ears of AI in vast world of online data
- Bridge gap when LLM lacks factual data in internal knowledge
- Use tools like Google Search API and web page readers
- Essential for up-to-date information retrieval tasks



(Ref: Vizuara AI Agents Bootcamp Day 6)

Key Takeaways: Multi-Agent Architecture Benefits

- Multi-agent architectures open new possibilities and efficiencies
- Agent collaboration solves complex tasks more cleanly than solo agents
- Right tools distributed among expert agents yield better results
- Tracing and evaluation become safety net for autonomous agents
- Langfuse provides detailed execution tracing and evaluation dashboards
- Metrics like hallucination, trustworthiness, and relevance ensure quality
- Essential for building reliable, factual, and trustworthy AI systems

YHK

References

- CS 194/294-196 (LLM Agents) - Lecture 3, Chi Wang and Jerry Liu
- LLM Powered Autonomous Agents Lil'Log
- Power of Autonomous AI Agents - Yogesh Kulkarni
- Microsoft AutoGen- Yogesh Kulkarni
- Microsoft AutoGen using Open Source Models- Yogesh Kulkarni
- A CAMEL ride - Yogesh Kulkarni
- Autonomous AI Agents (LLM, VLM, VLA) - Code Your Own AI
- Awesome LLM-Powered Agent https://github.com/hyp1231/awesome-llm-powered-agent
- Autonomous Agents (LLMs). Updated daily https://github.com/tmgthb/Autonomous-Agents

YHK

Thanks . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 3 pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

(https://medium.com/@yogeshharibhaukulkarni)

(https://www.linkedin.com/in/yogeshkulkarni/)

(https://www.github.com/yogeshhk/)

YHK

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page will go away because LaTeX now knows how many pages to expect for this document.