Agentics Tutorial

Lecture 2: Conversational Agents

Guest lectures at Columbia University Class on Agentic AI, Fintech, and the Data Economy. - Prof. Agostino Capponi

Alfio Massimiliano Gliozzo

IBM research

gliozzo@us.ibm.com

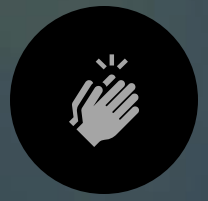
Recap from Lecture 1 - LLMs

Word Embeddings

Large Language Models

Enhancing LLMs with human supervision

Structured Decoding



Hands on: your first llm call

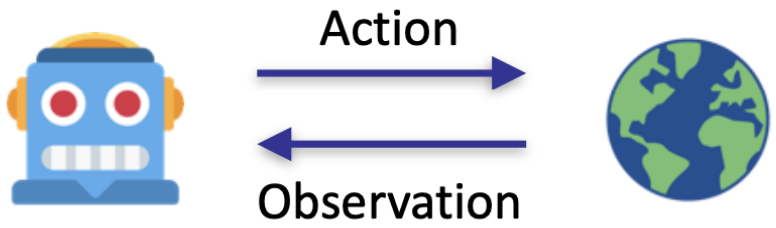
Outline

- Classical vs. GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

Outline

- Classical vs. GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

What is “agent”?



- An “intelligent” system that interacts with some “environment”
  - Physical environments: robot, autonomous car, …
  - Digital environments: DQN for Atari, Siri, AlphaGo, …
  - Humans as environments: chatbot
- Define “agent” by defining “intelligent” and “environment”
  - It changes over time!
  - Exercise question: how would you define “intelligent”?

From https://llmagents-learning.org/f24

Classical Agents vs. GenAI Agents

Artificial Intelligence – A Modern Approach 1st Ed. 1995

Intelligent Agent in Classical AI

Rational behavior maximizes expected utility

Symbolic Models

- Environment state description model
- Agent action description model
- Logic/Probability-based language

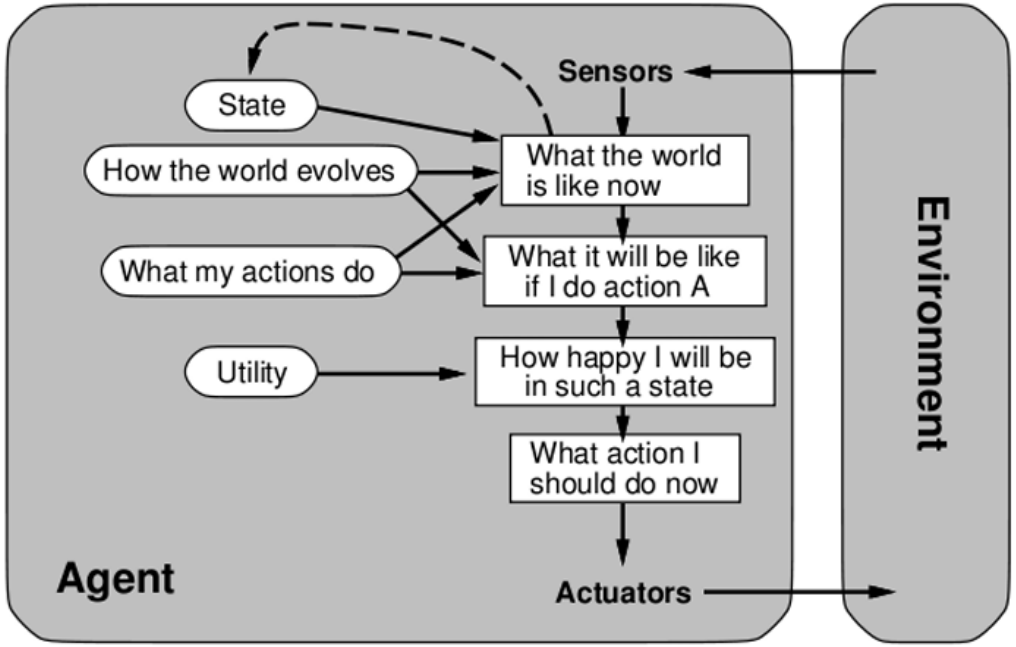
Automated Reasoning

- Symbolic/Probabilistic Reasoning
- Answers questions such as
  - "what is the current world state?"
  - "what is the effect of actions?"

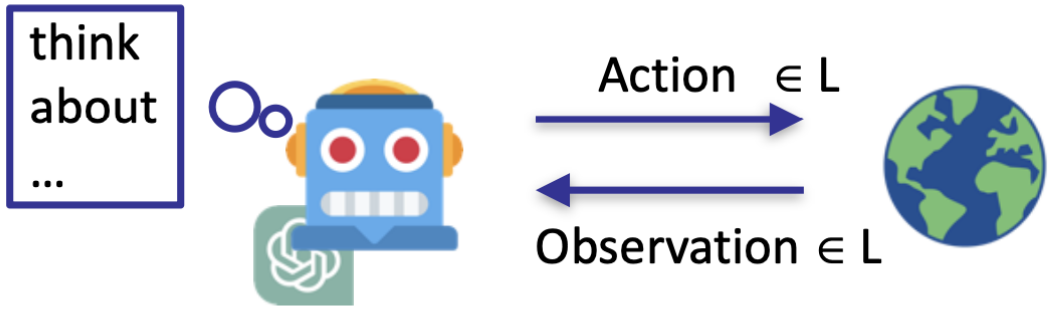
Optimal Policy Computation

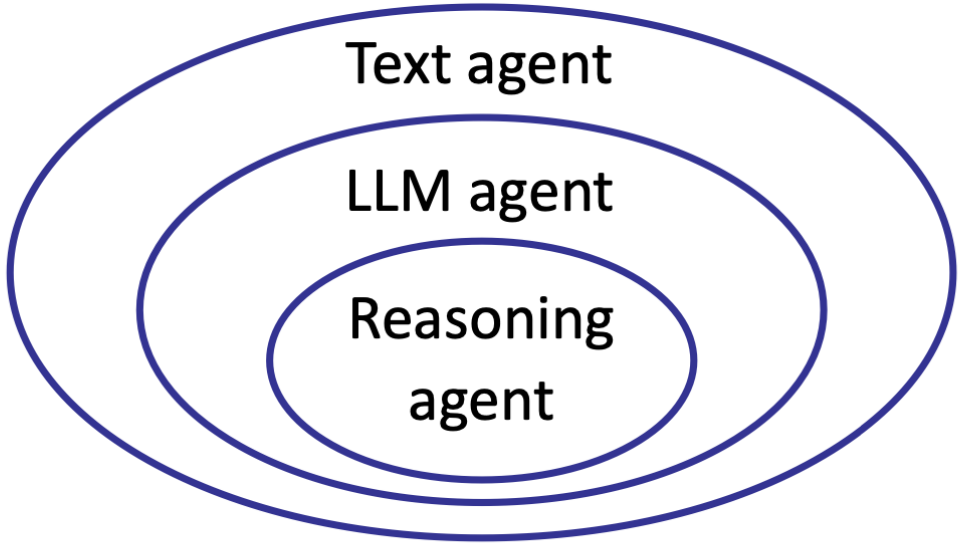
- Automated planning/Scheduling
- Probabilistic Inference
- Stochastic Programming

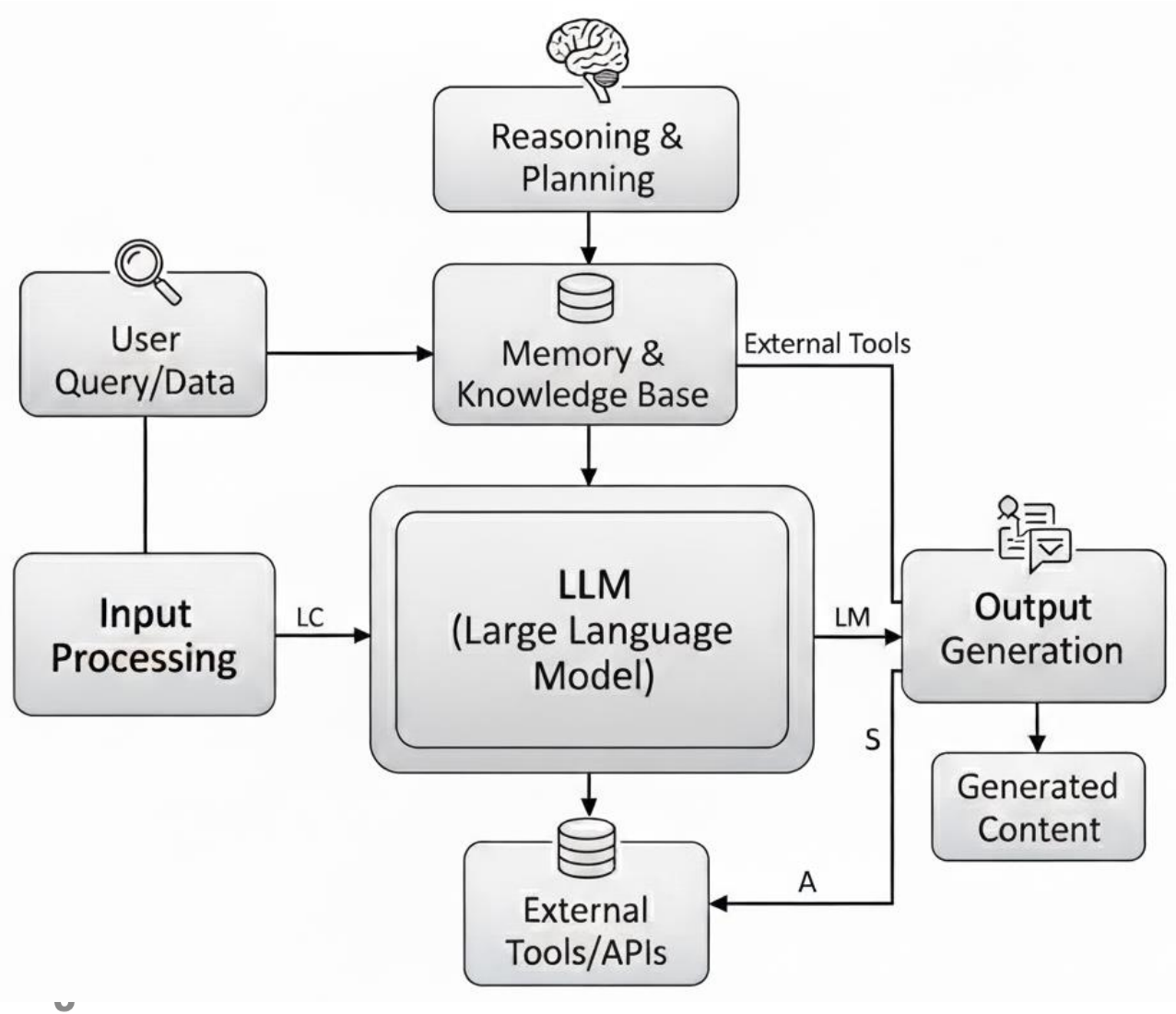
AIMA 1st ed. Chapter 2



What is “LLM agent”?







From https://llmagents-learning.org/f24

Classical Agents vs. GenAI Agents

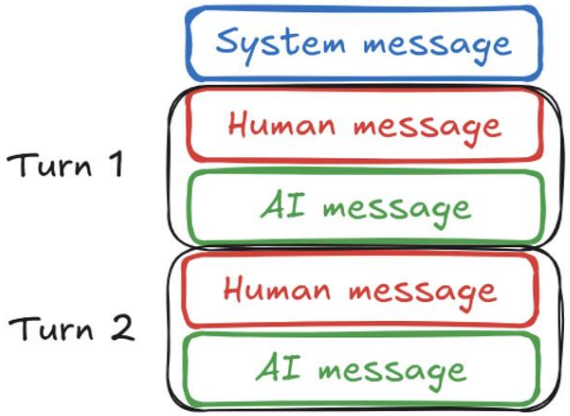
	Classical	GenAI
Approaches	Symbolic reasoning Expert systems	Text/pattern generation Pre-train and retrieve with generalization
Knowledge	Intensive knowledge engineering Narrow domains	Human annotations Trained on all data across all domains
Computation	CPU	GPUs
Strength	1. Predictable 2. Reliable 3. Transparent 4. Sound	1. Creative 2. General 3. Robust 4. Adaptable
Weakness	1. Fails out of pre-defined scopes/rules 2. Difficult to scale up and generalize 3. Difficult to adapt to new domains / task	1. Hallucinations 2. Imprecise 3. Difficult to control

The strength of one approach are the weakness of the other

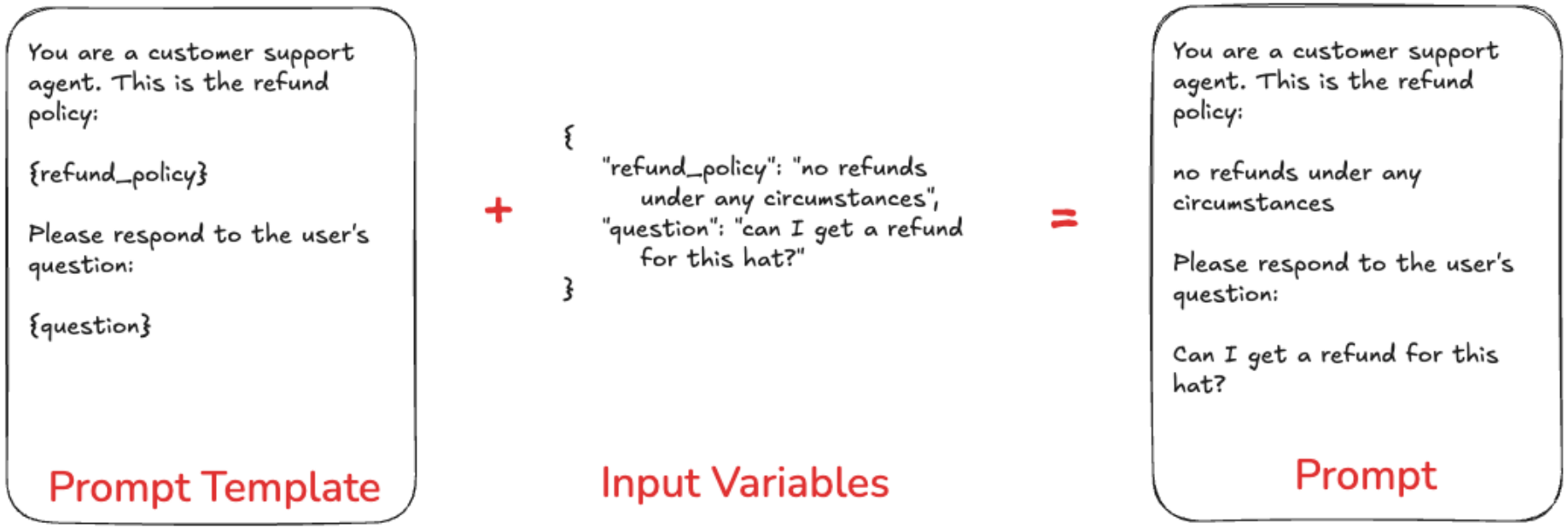
Outline

- Classical Agents vs GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

Chat Templates



- Predefined **input/output formatting schemes** for LLMs.
- Most Modern LLM have been finetuned to understand chat templates
- When conversations get long they might consume most of the prompt token budget.

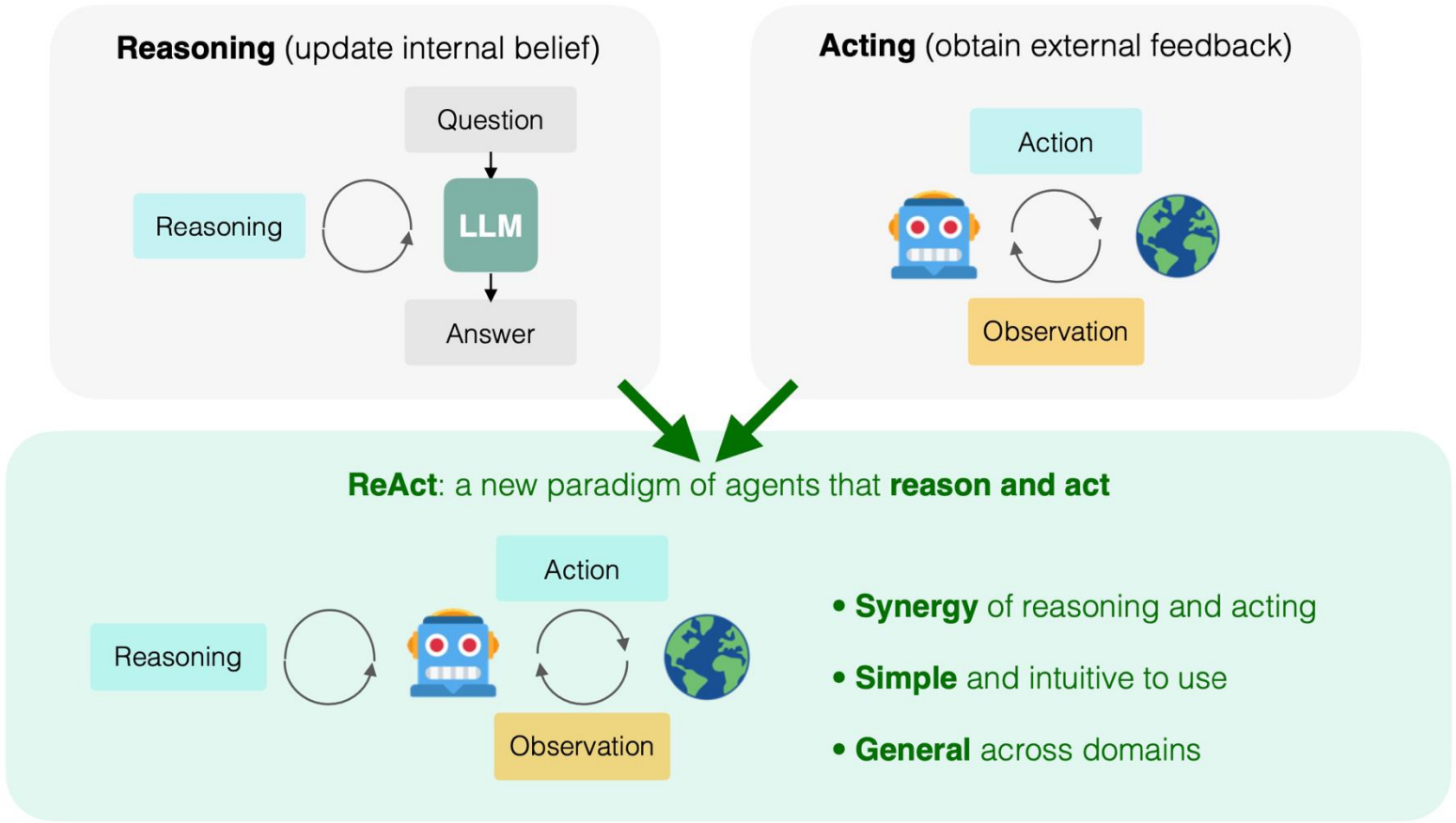


The REACT Framework

From https://llmagents-learning.org/f24

ReAct (Reasoning + Acting) framework

- LLM generates a "**Thought**", an "Action" such as web-search, and an "**Observation**" (the result of the action).
- Repeat the loop until task is complete



"ReAct: Synergizing Reasoning and Acting in Language Models", Yao el al 2022

ReAct Prompts

Source https://language-agent-tutorial.github.io/slides/II-Foundations-Reasoning-Memory.pdf

ReAct prompt

Question Which magazine was started first Arthur's Magazine or First for Women?
Thought 1 I need to search Arthur's Magazine and First for Women, and find which was started first.
Action 1 Search[Arthur's Magazine]
Observation 1 Arthur's Magazine (1844-1846) was an American literary periodical published in Philadelphia …
Thought 2 Arthur's Magazine was started in 1844. I need to search First for Women next.
Action 2 Search[First for Women]
Observation 2 First for Women is a woman's magazine published by Bauer Media Group … was started in 1989…
Thought 3 First for Women was started in 1989. 1844 (Arthur's Magazine) < 1989 (First for Women), so Arthur's Magazine was started first.
Action 3 Finish[Arthur's Magazine]

Reason-only prompt
(Chain-of-thought, CoT)

Question Which magazine was started first Arthur's Magazine or First for Women?
Thought Arthur's Magazine was started in 1844. First for Women was started in 1989. 1844 (Arthur's Magazine) < 1989 (First for Women), so Arthur's Magazine was started first.
Answer Arthur's Magazine

Act-only prompt
(similar to WebGPT)

Question Which magazine was started first Arthur's Magazine or First for Women?
Action 1 Search[Arthur's Magazine]
Observation 1 Arthur's Magazine (1844-1846) was an American literary periodical published in Philadelphia …
Action 2 Search[First for Women]
Observation 2 First for Women is a woman's magazine published by Bauer Media Group … was started in 1989…
Action 3 Finish[Arthur's Magazine]

Standard prompt

Question Which magazine was started first Arthur's Magazine or First for Women?
Answer Arthur's Magazine

Outline

- Classical Agents vs GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

Tools

Tools are **external functions, APIs, or services** that an LLM can call to perform tasks it cannot do on its own.

Why Tools?

- Extend model capabilities
- Enable **real-time data access**
- Ensure more **accurate and grounded answers**
- Allow integration with external systems

Examples

- 🔍 search(query) → fetch web results
- 🌦️ get_weather(city, unit) → retrieve live weather
- 🗄️ sql_query(statement) → query a database
- ➗ math_solver(expression) → perform precise calculations

Tool Calling

LLMs use Structured Decoding to generate function calls

INPUT

Question: What's the weather like today in New York in Celsius?

Pydantic Definition

```
class GetWeather(BaseModel):
    city: str = Field(..., description="Name of the city to fetch weather for")
    unit: Literal["celsius", "fahrenheit"] = Field( "celsius",
        description="Temperature unit"
    )
```

JSON SCHEMA

Equivalent to GetWeather.schema_json()

```
{"title": "GetWeather",
 "type": "object",
 "properties": {"city": {
    "type": "string",
    "description": "Name of the city to fetch weather for" },
  "unit": {
    "type": "string",
    "enum": ["celsius", "fahrenheit"],
    "description": "Temperature unit",
    "default": "celsius"} },
 "required": ["city"]}
```

OUTPUT

```
{"tool_call": {
   "name": "get_weather",
   "arguments": {
            "city": "New York",
            "unit": ”celsius”}}}
```

Tool Execution

the generated arguments are passed as arguments to invoke functions, which are excecuted and returned to the system as a tool message

INPUT Question: What's the weather like today in New York in Celsius?

LLM

Tool Call

```
{"tool_call": {
   "name": "get_weather",
   "arguments":
            {"city": "New York",
            "unit": ”celsius”}}}
```

CPU API

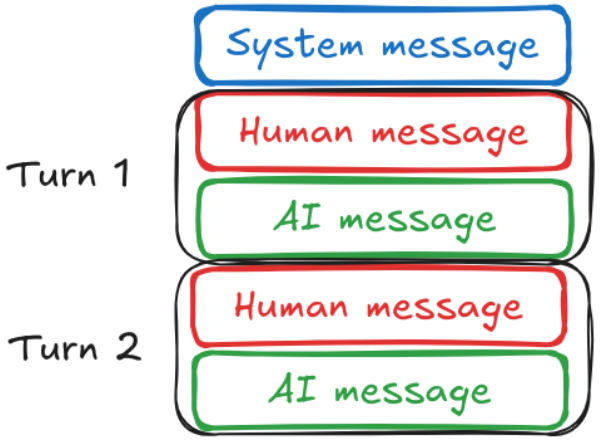
Tool Execution

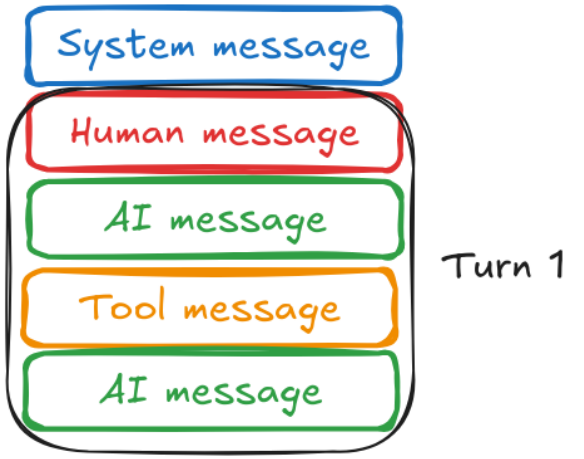
```
{"city": "New York",
 "temperature": 18,
 "unit": "celsius",
 "condition": "Partly cloudy"}
```

LLM

System Answer

It’s currently **18**°C and partly cloudy in New York.





Make your own tool in CrewAI

Duck Duck Go Search Tool

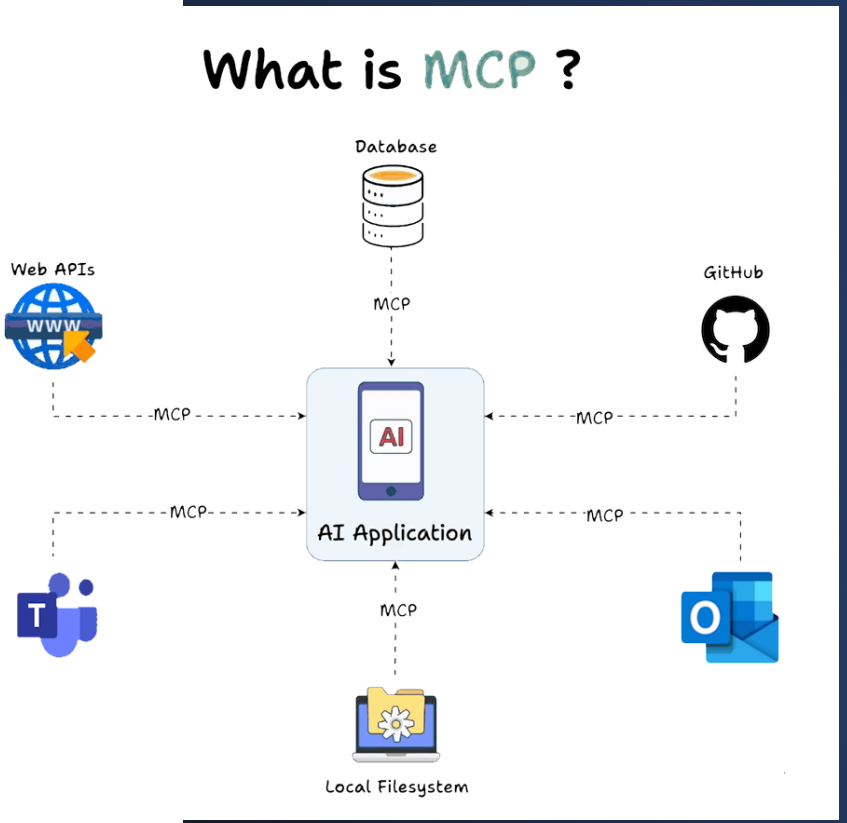
```python
from crewai.tools import tool
from ddgs import DDGS

## Define a Crew AI tool to get news for a given date using the DDGS search engine
@tool("fetch_data_async")
def web_search(query: str) -> str:
    """ Fetch web search results for the given query using DDGS."""
    return str(DDGS().text(query, max_results=10))
```

Pre-defined tools:

- Langchain tools https://python.langchain.com/docs/integrations/tools/
- CrewAI Tools https://docs.crewai.com/en/tools/overview

Model Context Protocol (MCP)

- **MCP servers** provide **structured APIs** for LLM to call during a conversation.
  - **Tools**
  - Data
  - Actions
- **Standardize communication** between LLMs and external resources
- Enable **secure, permissioned access** to systems



Use Existing MCP server in CrewAI

```
fetch_params=StdioServerParameters(
    command="uvx",
    args=["mcp-server-fetch"],
    env={"UV_PYTHON": "3.12", **os.environ},
)
search_params=StdioServerParameters(
    command="python3",
    args=[os.getenv("MCP_SERVER_PATH")],
    env={"UV_PYTHON": "3.12", **os.environ},
)
with MCPServerAdapter(fetch_params) as fetch_tools, \
    MCPServerAdapter(search_params) as search_tools:
    print(f"Available tools from Stdio MCP server: {[tool.name for tool in fetch_tools]}")
    print(f"Available tools from Stdio MCP server: {[tool.name for tool in search_tools]}")
    tools=fetch_tools + search_tools
```

List of available MCPs
https://github.com/modelcontextprotocol/servers?tab=readme-ov-file#-reference-servers

Make your own MCP server

You can define your own MCP server by simply wrapping any function with the @mcp.tool() decorator
You'll need to pass the pytho

```python
from mcp.server.fastmcp import FastMCP
from ddgs import DDGS
mcp = FastMCP("Search")

@mcp.tool()
def web_search(query:str, max_results:int) -> list[str]:
    """return spippets of text extracted from duck duck go search for the given
    query :  using DDGS search operators
    max_results: number of snippets to be returned, usually 5 - 20"""

    search_results = DDGS().text(query, max_results=max_results)
    return [f'{x["title"]}\n{x["body"]}\n{x["href"]}' for x in search_results ]

if __name__ == "__main__":
    mcp.run(transport="stdio")
```

```python
search_params=StdioServerParameters(
    command="python3",
    args=[os.getenv("MCP_SERVER_PATH")],
    env={"UV_PYTHON": "3.12", **os.environ},
)
```

Outline

- Classical Agents vs GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Multi Agent Systems: CrewAI
- Hands on: build simple agent with tools in CrewAI

Reasoning and Planning with LLMs

Classical AI

Reasoning: derive a new statement from existing statements

- Prove a new theorem given axioms and proven theorems
- Compute posterior probability of some event given prior probability

Planning: find a sequence of statements that achieves the objective

- search a sequence of actions that reaches the goal state given action model
- optimize policy that maximizes expected utility given trajectories

GenAI

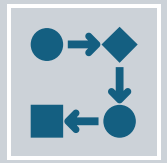
Reasoning and Planning are specific form of text that can be generated by LLM based when prompted appropriately

- If the LLM generated text is a set of actions needed to achieve a goal, it is called **planning**
- If the generated text provides an explanation for a statement given a set of premises, then it is called **reasoning**

Approaches:

- Chain Of Thought
- Plan-and-Execute

Chain of Thought



guides the LLM to articulate its reasoning process step-by-step before providing the final answer

ask the model to "**Think step-by-step**," or provide an example of a chain of thought

Example Question

User: *If Alice has 3 apples and buys 4 more, then gives 2 to Bob, how many apples does she have left?*

CoT Prompt: *Think step by step – Hidden from user*

Chain of Thought (reasoning steps) – *Hidden from user*
1.Alice starts with 3 apples.
2.She buys 4 more → total now = 3 + 4 = 7.
3.She gives 2 apples to Bob → total now = 7 − 2 = 5.

Final Answer - *what the user sees*

Alice has **5 apples** left.

"Chain of Thought Prompting Elicits Reasoning in Large Language Models" Wei et al 2022

Plan-and-Execute

- **Planning** → The LLM (or a dedicated "planner agent") produces a structured *plan* of actions to solve the user's task.
- **Execution** → The agent (or another LLM instance) follows the plan step by step, calling tools, retrieving knowledge, or producing outputs.
- **Key Characteristics**
  - **Decomposition**: Breaks complex tasks into manageable subtasks.
  - **Separation of Concerns**: One component focuses on reasoning (the *plan*), another on action (the *executor*).
  - **Flexibility**: Plans can be revised dynamically if execution reveals errors.
  - **Transparency**: The plan is interpretable (auditable by humans).

Wang, X., Wei, J., Schuurmans, D., Le, Q., & Zhou, D. (2023). *Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. ACL 2023*
https://arxiv.org/pdf/2305.04091

Example Plan: literature review

```
Thought: Okay, I understand the task. I need to create a literature report on Large
Language Models (LLMs) based on academic sources from arXiv. The report should cover
several interesting aspects of LLMs, providing detailed and accurate answers supported by
evidence from the search results. The final answer must be a JSON object containing the
title, abstract, full report in markdown format, and citations with URLs, authors, titles,
and relevant text.
Here's my plan:
1.  **Define the "Interesting Aspects":** I will focus on the following aspects:
    *   LLM Architectures and Training Techniques
    *   LLM Limitations and Challenges
    *   Ethical Considerations of LLMs
    *   Applications of LLMs
    *   Emerging Trends in LLM Research
2.  **Search arXiv:** I will use the `web_search` tool to search arXiv for relevant papers
on each aspect. I will start with broad queries and refine them based on the initial
results. I will aim for 3-5 papers per aspect initially.
3.  **Extract and Summarize Information:** For each relevant paper found, I will extract
key information, including the main findings, methodologies, and conclusions. I will
summarize this information concisely.
4.  **Synthesize and Structure the Report:** I will organize the summarized information
into a structured document. Each section will address one of the "interesting aspects" of
LLMs. Within each section, I'll present the synthesized information, citing the source
papers from arXiv.
5.  **Provide Supporting Passages:** I will include direct quotes or paraphrased passages
from the source papers to support my statements and conclusions.
6.  **Review and Refine:** I will review the report for accuracy, clarity, and
completeness. I will ensure that all sources are properly cited and that the report
addresses the task requirements.
```

Tool Executions

🔧 Agent Tool Execution

Agent: **Doc Searcher**

Thought: The first step to complete the task is to identify key papers related to Large Language Models on arXiv. I'll use the 'web_search' tool to find the most relevant academic articles on the topic. After identifying the papers, I'll use the 'fetch' tool to download and analyze them. The result of this analysis will be used to compose the literature report.

Using Tool: **web_search**

Tool Input

```
"{\"query\": \"Large Language Models site:arxiv.org\", \"max_results\": 5}"
```

Tool Output

```
[2307.05782] Large Language Models
by MR Douglas · 2023 · Cited by 44 — Abstract:Artificial intelligence is making spectacular progress, and one of the best examples is the development of large language models ...
https://arxiv.org/abs/2307.05782
```

🔧 Agent Tool Execution

Agent: **Doc Searcher**

Thought: Thought: The web search returned a promising paper titled "Large Language Models" by MR Douglas. Now, I'll fetch the content of this paper from arXiv for further analysis.

Using Tool: **fetch**

Tool Input

```
"{\"url\": \"https://arxiv.org/abs/2307.05782\"}"
```

Tool Output

```
Contents of https://arxiv.org/abs/2307.05782:
[View PDF](/pdf/2307.05782)

> Abstract:Artificial intelligence is making spectacular progress, and one of the best examples is the development of large language models (LLMs) such as OpenAI's GPT series. In these lectures, written for readers with a background in mathematics or physics, we give a brief history and survey of the state of the art, and describe the underlying transformer architecture in detail. We then explore some current ideas on how LLMs work and how models trained to predict the next word in a text are able to perform other tasks displaying intelligence.

|  |  |
| --- | --- |
| Comments: | 47 pages (v2: added references, corrected typos) |
| Subjects: | Computation and Language (cs.CL); High Energy Physics - Theory (hep-th); History and Overview (math.HO); Computational Physics (physics.comp-ph) |
| MSC classes: | 68T01 |
| ACM classes: | I.2.7 |
| Cite as: | [arXiv:2307.05782](https://arxiv.org/abs/2307.05782) [cs.CL] |
|  | (or [arXiv:2307.05782v2](https://arxiv.org/abs/2307.05782v2) [cs.CL] for this version) |
|  | <https://doi.org/10.48550/arXiv.2307.05782> arXiv-issued DOI via DataCite |

## Submission history

From: Michael R. Douglas [[view email](/show-email/4d495717/2307.05782)]
 **[[v1]](/abs/2307.05782v1)**
Tue, 11 Jul 2023 20:21:02 UTC (330 KB)
**[v2]**
Fri, 6 Oct 2023 12:13:46 UTC (331 KB)
```

Outline

- Classical Agents vs GenAI Agents
- Conversational Agents
- Tools
- Reasoning
- Agentic Platforms: CrewAI
- Hands on: build simple agent with tools in CrewAI

Mainstream Agentic Platforms

LangChain

 Large ecosystem, many integrations

 Chain/graph abstractions for tool use

 Flexible but can feel heavyweight

link

LangGraph

 Graph-based state machines for LLMs

 Clear async flow control

 Strong for experimental architectures

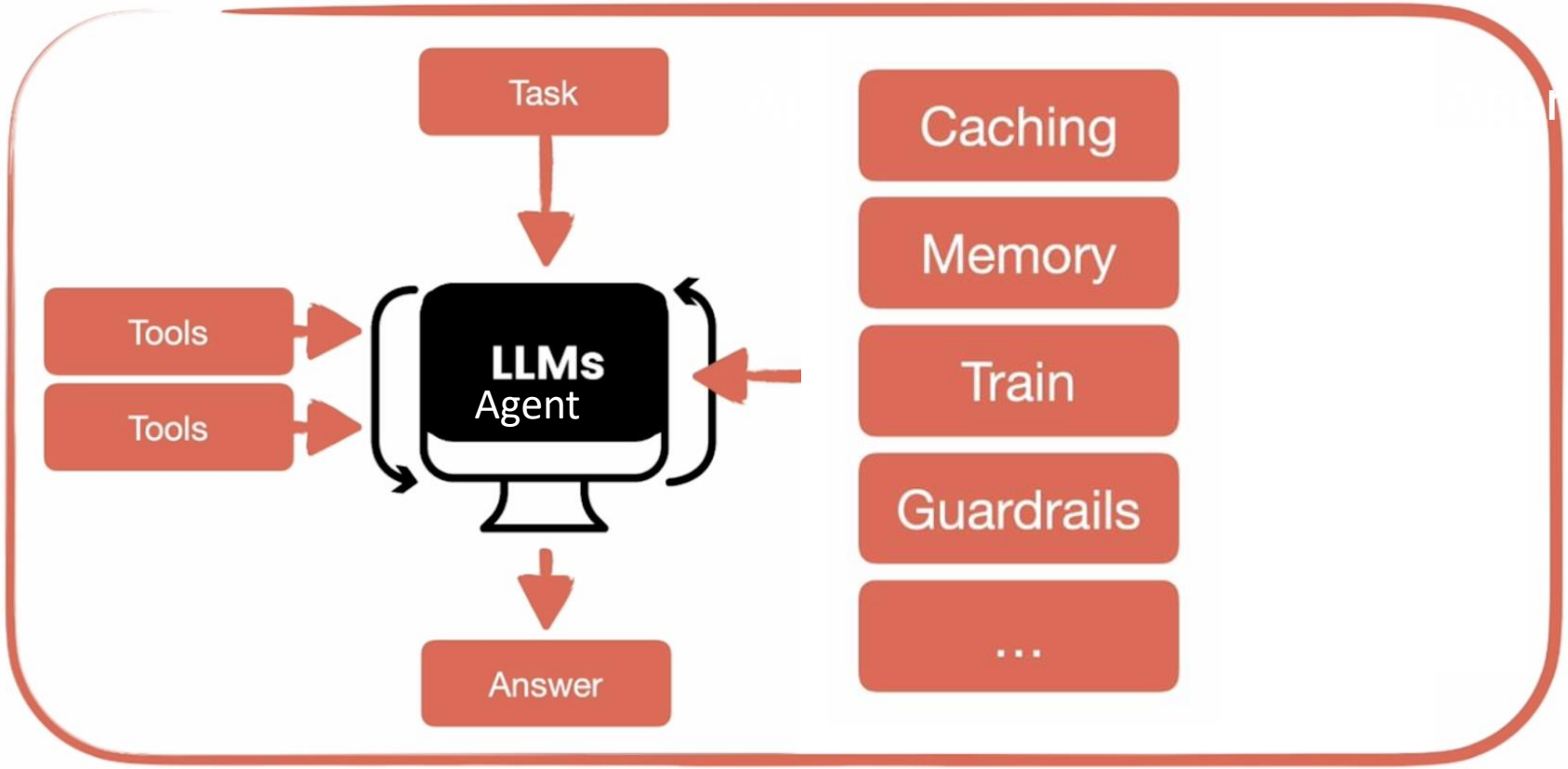
CrewAI

 Multi-agent collaboration framework

 Task delegation & role specialization

 Good for simulations and workflows

CrewAI Agents



Crew AI is a platform to build LLM based multi agent systems

From: https://learn.deeplearning.ai/courses/practical-multi-ai-agents-and-advanced-use-cases-with-crewai/

CrewAI: Tasks

Agents

- Role
- Goal
- Backstory

```
researcher:
 role: >
   {topic} Senior Data Researcher
 goal: >
   Uncover cutting-edge developments in {topic}
 backstory: >
   You're a seasoned researcher with a knack for uncovering...
```

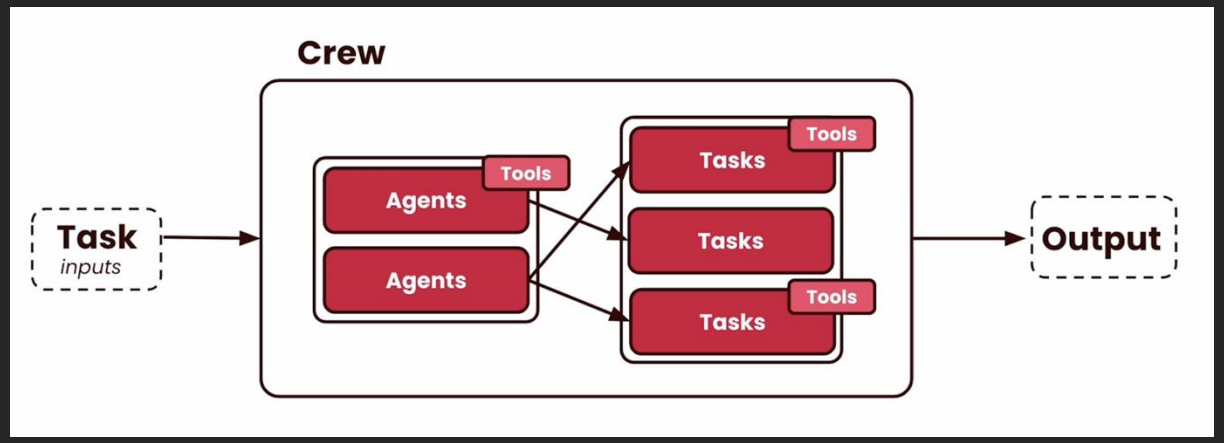
Tasks

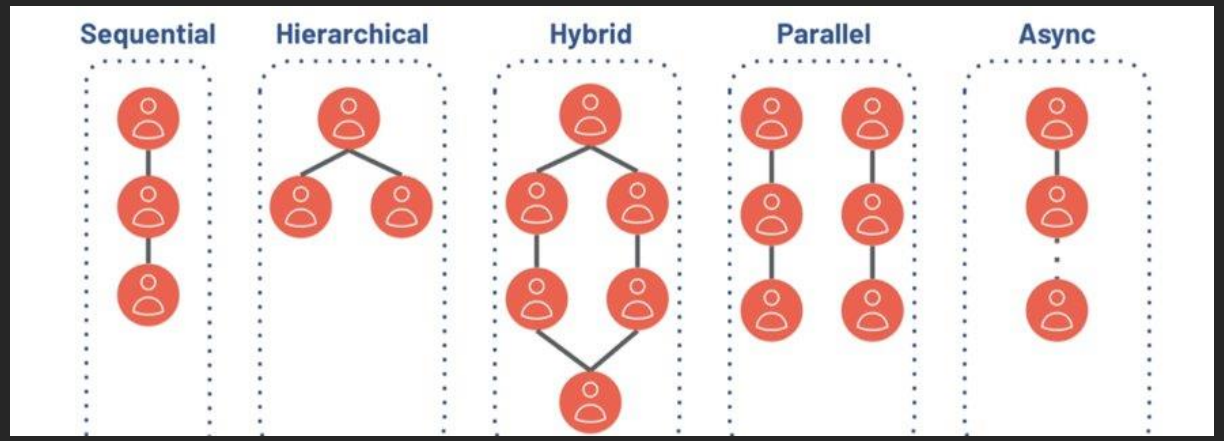
- Description
- Expected Output
- Agent

```
research_task:
 description: >
   Conduct a thorough research about {topic}...
 expected_output: >
   10 points of the most relevant information about {topic}...
 agent: researcher
```

Agents have identities and can be instructed to execute tasks

From: https://learn.deeplearning.ai/courses/practical-multi-ai-agents-and-advanced-use-cases-with-crewai/





Crews

(Multi Agent Systems)

Crews combine and execute multiple agents to achieve an higher level goal.

Task are executed asynchronously following different workflow

Hands on: Build a simple agent with tool

Hands on Examples

- Tutorials:
  - https://colab.research.google.com/github/IBM/Agentics/blob/conversational_agent/tutorials/conversational_agent.ipynb
- Examples:
  - examples/conversational_agent.py
  - examples/crew_ai_web_search_report.py
- Homework:
  - Fork Agentics
  - Set up agentics locally and try out examples
  - Build your own tools
    - E.g. Yahoo! Finance APIs
    - …
  - Be creative and try out interesting use cases for those
  - Push your examples in your fork and share with other students to get feedback (optional)

References

OnLine Courses:

- Berkley MOOC on LLMs https://rdi.berkeley.edu/llm-agents/f24 (free online)
- CrewAI Tutorial @ DeepLearning.ai
  - https://learn.deeplearning.ai/courses/practical-multi-ai-agents-and-advanced-use-cases-with-crewai/
- Langchain tutorial: https://learn.deeplearning.ai/courses/langchain/

Agentic Platforms:

- Langchain: https://python.langchain.com/
- Langgraph: https://langchain-ai.github.io/langgraph/
- CreawAI: https://www.crewai.com/

Relevant Publications

- Yao, Shunyu; Zhao, Jeffrey; Yu, Dian; Du, Nan; Shafran, Izhak; Narasimhan, Karthik; Cao, Yuan ReAct: Synergizing Reasoning and Acting in Language Models *(ICLR), 2023.*
- Wang, X., Wei, J., Schuurmans, D., Le, Q., & Zhou, D. (2023). *Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. ACL 2023*