IERG5050 AI Foundation Models, Systems & Applications
Spring 2025

Agentic AI Applications

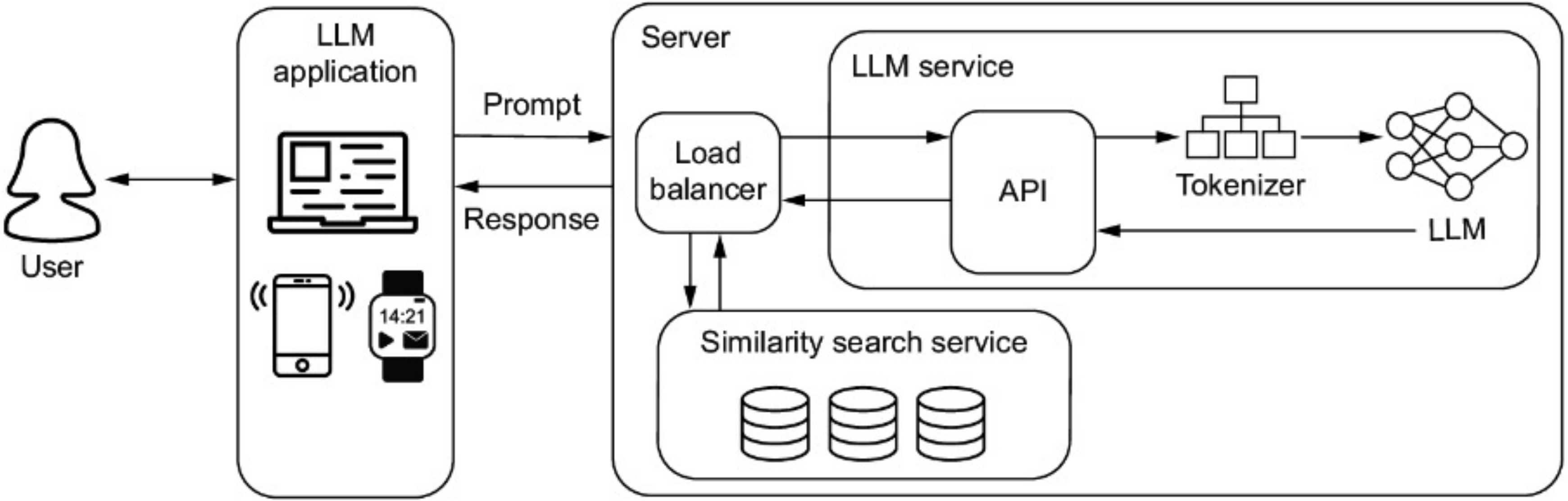
Prof. Wing C. Lau
wclau@ie.cuhk.edu.hk
http://www.ie.cuhk.edu.hk/wclau

Acknowledgements

Many of the slides in this lecture are adapted from the sources below. Copyrights belong to the original authors.

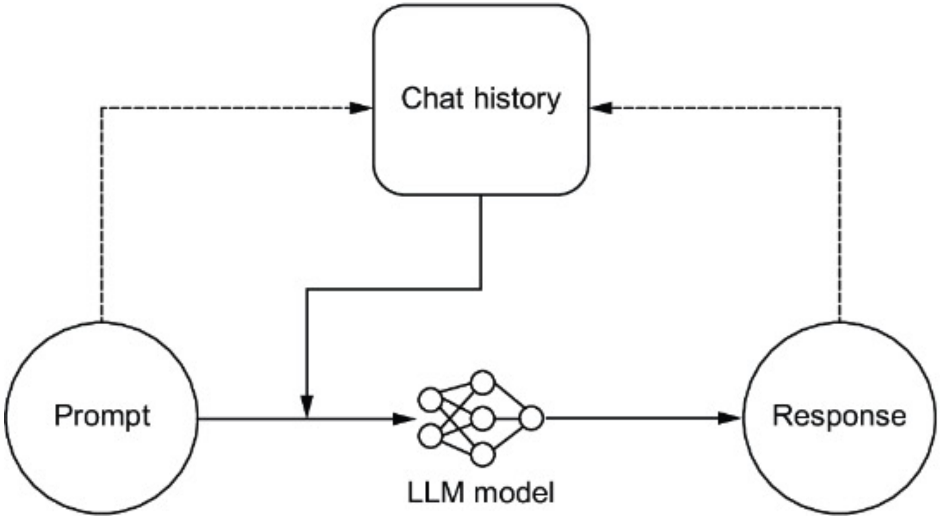
- LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/
- LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/
- AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/
- Building LLM Powered Applications By Valentina Alto, Packt Publishing, May 2024
- AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications
- LLMs in Production by by Christopher Brousseau, Matthew Sharp, Jan 2025, Manning Publications
- Shunyu Yao, "LLM agents: brief history and overview," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents/assets/llm_agent_history.pdf, https://www.youtube.com/watch?v=RM6ZArd2nVc
- Chi Wang, "Agentic AI Frameworks & AutoGen," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents-mooc/slides/autogen.pdf, https://www.youtube.com/watch?v=OOdtmCMSOo4
- Jerry Liu, "Building a Multimodal Knowledge Assistant (LlamaIndex)," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents-mooc/slides/MKA.pdf, https://www.youtube.com/watch?v=OOdtmCMSOo4
- Ruslan Salakhutdinov, "Multimodal Autonomous AI Agents," Guest Lecture for UC Berkeley MOOC on Advanced Large Language Model Agents, Spring 2025, https://rdi.berkeley.edu/adv-llm-agents/slides/ruslan-multimodal.pdf, https://www.youtube.com/live/RPINOYM12RU
- Nicolas Chapados, "AI Agents for Enterprise Workflows," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents/assets/agentworkflows.pdf, https://www.youtube.com/live/-yf-e-9FvOc
- Ben Mann, "Measuring Agent capabilities and Anthropic's RSP," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents/assets/antrsp.pdf, https://www.youtube.com/live/6y2AnWol7oo
- Caiming Xiong, "Multimodal Agents – from Perception to Action," Guest Lecture for UC Berkeley MOOC on Advanced Large Language Model Agents, Spring 2025, https://rdi.berkeley.edu/llm-agents-mooc/slides/Multimodal_Agent_caiming.pdf, https://www.youtube.com/live/n__Tim8K2IY
- Omar Khattab, "Compound AI Systems & the DSPy Framework," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents-mooc/slides/dspy_lec.pdf, https://www.youtube.com/live/JEMYuzrKLUw
- Graham Neubig, "Agents for Software Development," Guest Lecture for UC Berkeley MOOC on Large Language Model Agents, Fall 2024, https://rdi.berkeley.edu/llm-agents-mooc/slides/neubig24softwareagents.pdf, https://www.youtube.com/live/f9L9Fkg-8K4

An LLM-powered Application

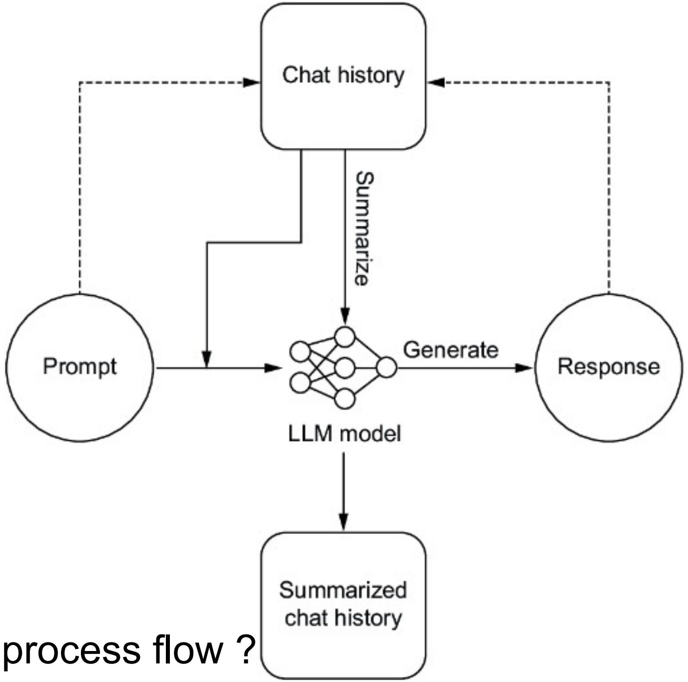


Source: LLMs in Production by By Christopher Brousseau, Matthew Sharp, Jan 2025, Manning Publications

Process-Flow of a Sample LLM-powered Application: a Minimalist ChatBot



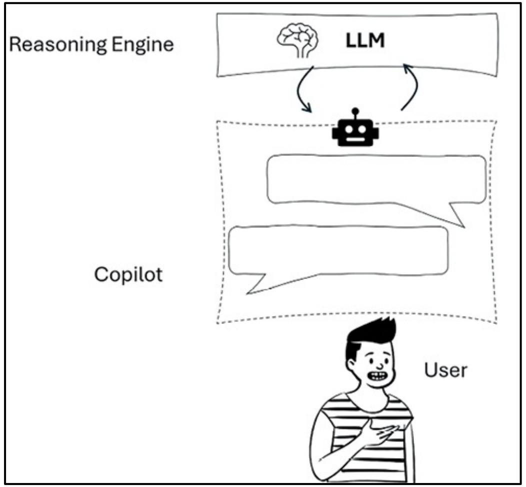
OR



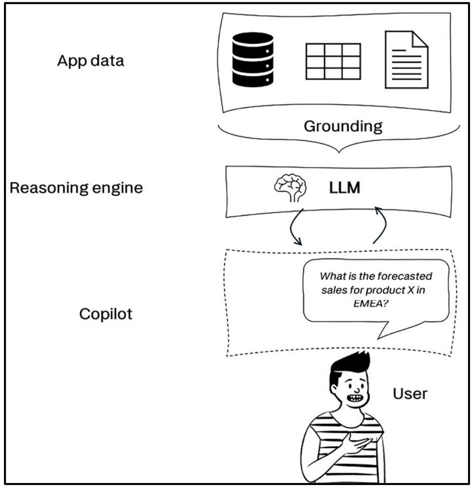
Any other common component needed in the process flow ?

Source: LLMs in Production by By Christopher Brousseau, Matthew Sharp, Jan 2025, Manning Publications

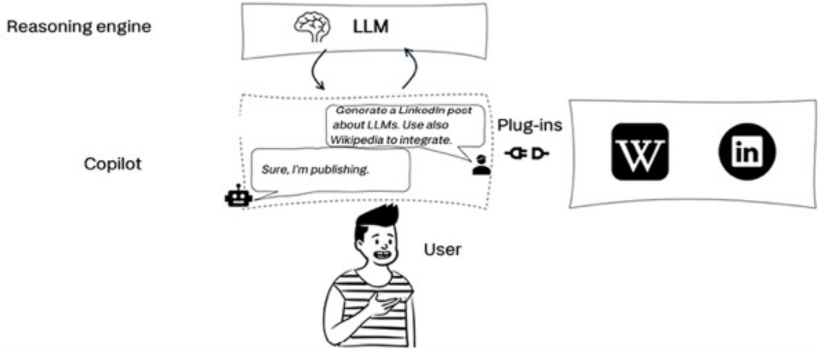
Copilot (aka AI Assistant): An LLM-powered Application



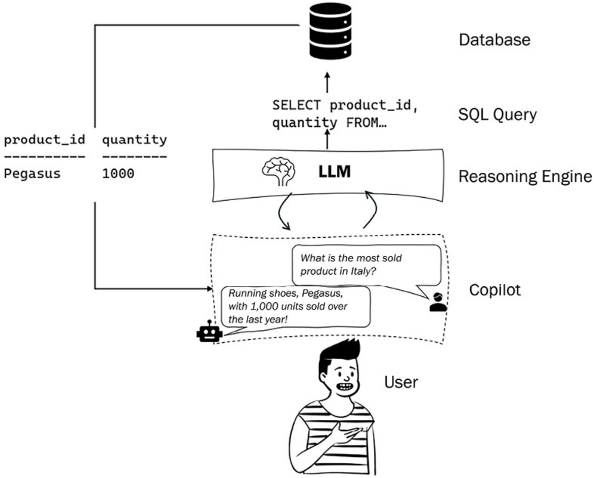
An LLM-powered Copilot



A Copilot w/ grounding

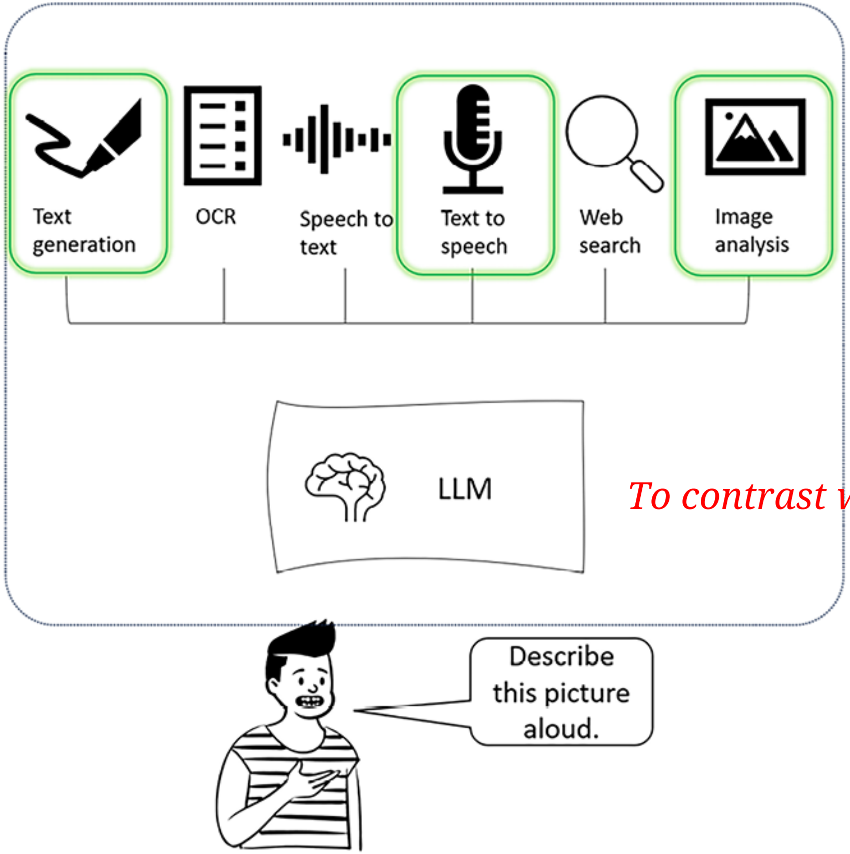


A conversational UI to reduce the gap between the user and the database or ext. tools



Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

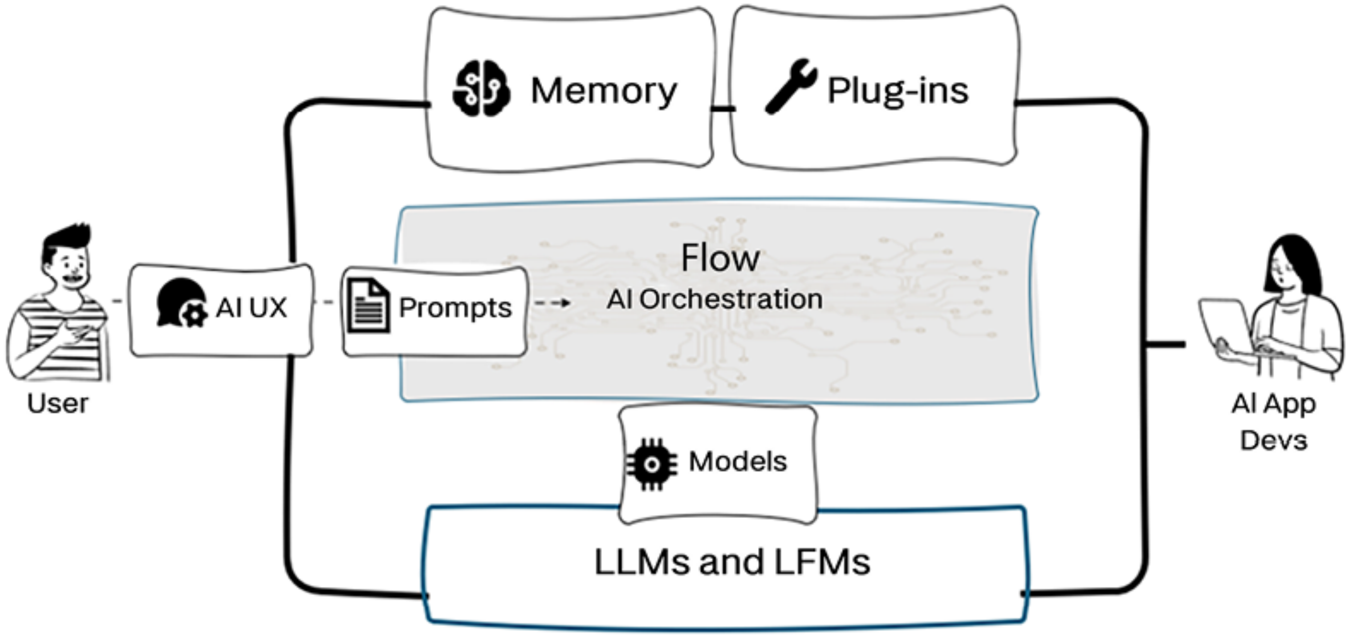
A Multimodal Application via an LLM w/ Single-modal Tools



To contrast with a Multimodal-NATIVE LLM (later)

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

High-level Architecture of an LLM-powered Application



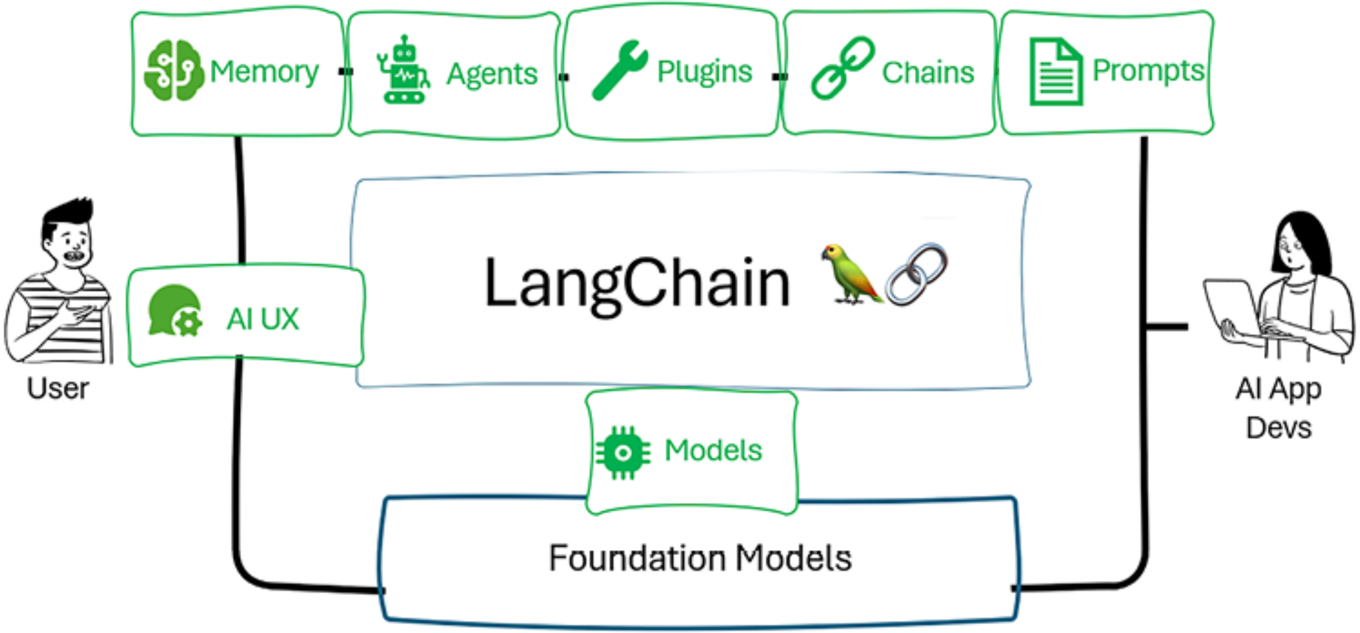
Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

LLM-application/ Agent Development Frameworks

Framework	Best For	Deployment	Scalability	Current Adoption
LangChain & LangGraph	Enterprise AI apps, multi-agent workflows	Self-hosted, Cloud	High – widely used in production	Most adopted
Autogen & Semantic Kernel	Microsoft ecosystem, scalable AI apps	Azure, Self-hosted	High – enterprise-scale	Growing in enterprises
LlamaIndex	AI-powered search, RAG	Cloud, Self-hosted	High – efficient data processing	Strong in AI search
AutoGPT	No-code AI automation, continuous agents	Cloud-first, some self-hosted options	Medium – automation focused	Popular for prototyping
CrewAI	Workflow-based multi-agent AI apps	Cloud, Self-hosted	Medium – still early-stage	Fast-growing
PydanticAI	FastAPI & Pydantic-based AI apps	Self-hosted	Low – lightweight framework	Niche adoption
Spring AI	Java-based AI applications	Self-hosted, Cloud	Medium – depends on Java ecosystem	Popular in Java world
Haystack	LLM-powered search & RAG	Self-hosted, Cloud	High – built for production	Strong in search applications

Source: https://medium.com/@varmalearn/a-casual-guide-to-the-top-agentic-ai-frameworks-in-2025-9299dd94e034

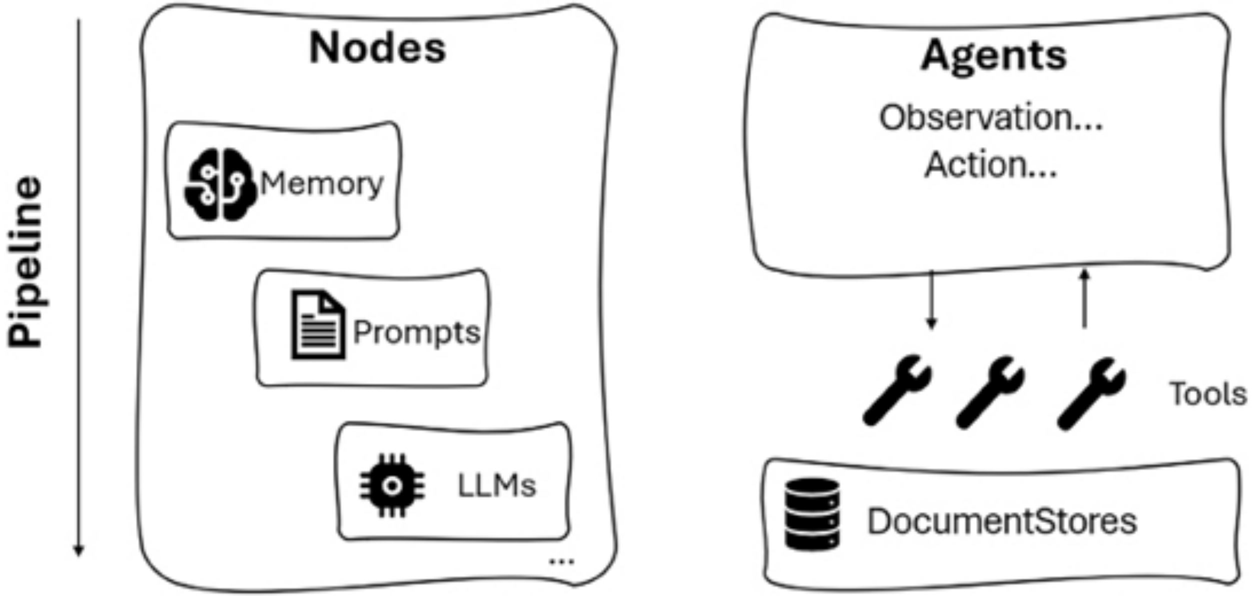
LangChain: a framework for building LLM-based Applications



Components of LangChain

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

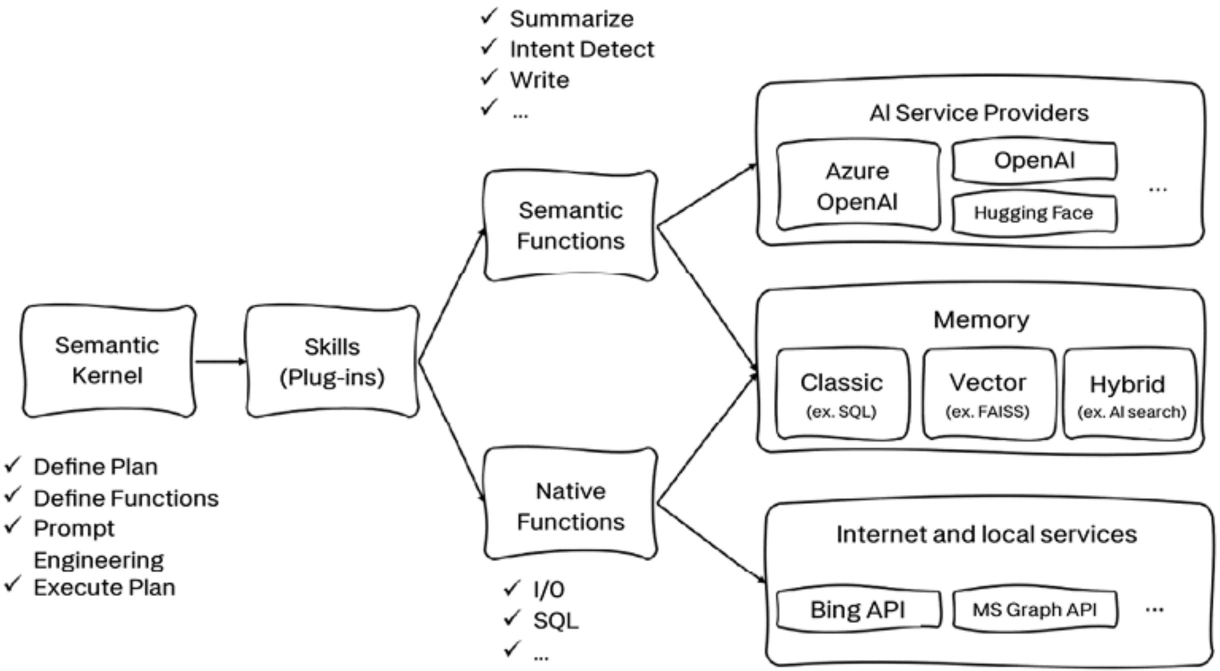
Haystack: a framework for building LLM-based Applications



Haystack's components

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

Semantic Kernel: a framework for building LLM-based Applications



Anatomy of Semantic Kernel

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

LangChain vs. Haystack vs. Semantic Kernel

Feature	LangChain	Haystack	Semantic Kernel
LLM support	Proprietary and open-source	Proprietary and open source	Proprietary and open source
Supported languages	Python and JS/TS	Python	C#, Java, and Python
Process or-chestration	Chains	Pipelines of nodes	Pipelines of func-tions
Deployment	No REST API	REST API	No REST API
Feature	LangChain	Haystack	Semantic Kernel

Table 2.1: Comparisons among the three AI orchestrators

More on Semantic Kernel as well as other Agentic AI frameworks later…

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

Developing LLM-based Applications with LangChain

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

LangChain Overview

Open-source development framework for LLM applications

Python and JavaScript (TypeScript) packages

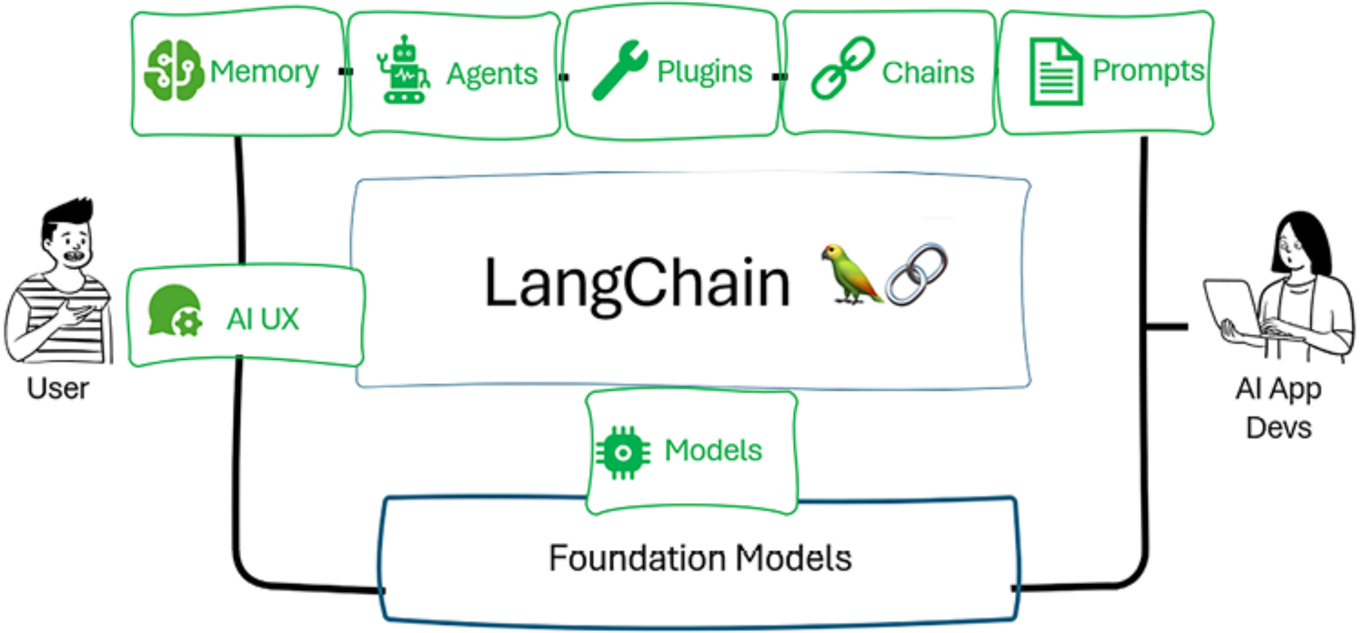
Focused on composition and modularity

Key value adds:

1. Modular components
2. Use cases: Common ways to combine components

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain: a framework for building LLM-based Applications



Components of LangChain

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

LangChain DeepLearning.AI

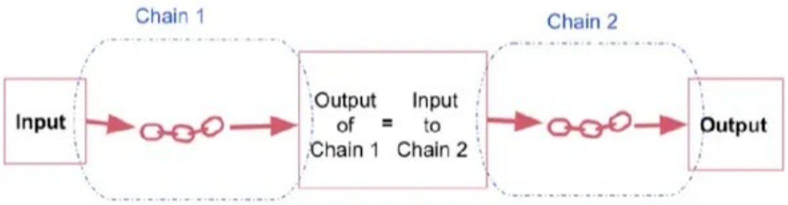
Components of LangChain

- Models
  - LLMs: 20+ integrations
  - Chat Models
  - Text Embedding Models: 10+ integrations
- Prompts
  - Prompt Templates
  - Output Parsers: 5+ implementations
    - Retry/fixing logic
  - Example Selectors: 5+ implementations
- Indexes
  - Document Loaders: 50+ implementations
  - Text Splitters: 10+ implementations
  - Vector stores: 10+ integrations
  - Retrievers: 5+ integrations/implementations
- Chains
  - Prompt + LLM + Output parsing
  - Can be used as building blocks for longer chains
  - More application specific chains: 20+ types
- Agents
  - Agent Types: 5+ types
    - Algorithms for getting LLMs to use tools
  - Agent Toolkits: 10+ implementations
    - Agents armed with specific tools for a specific application

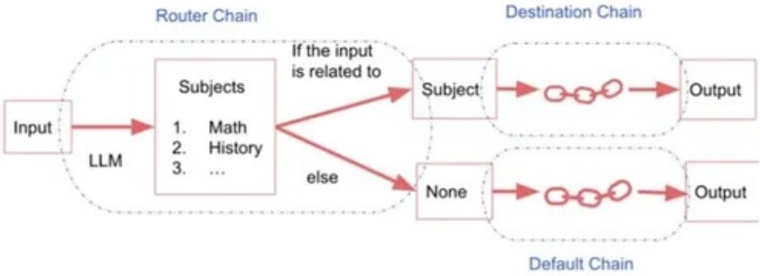
Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Process Flow Orchestration with “Chains”

Simple Sequential Chain: One i/p, One o/p

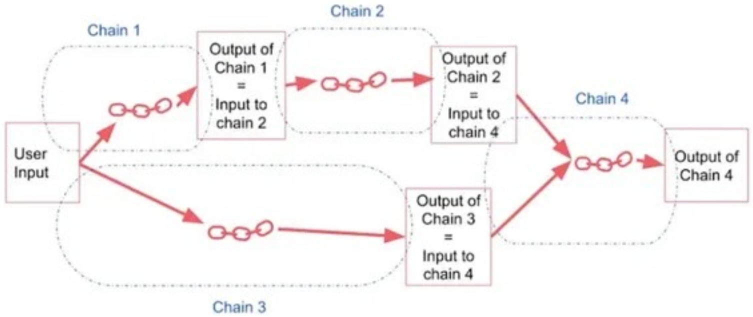


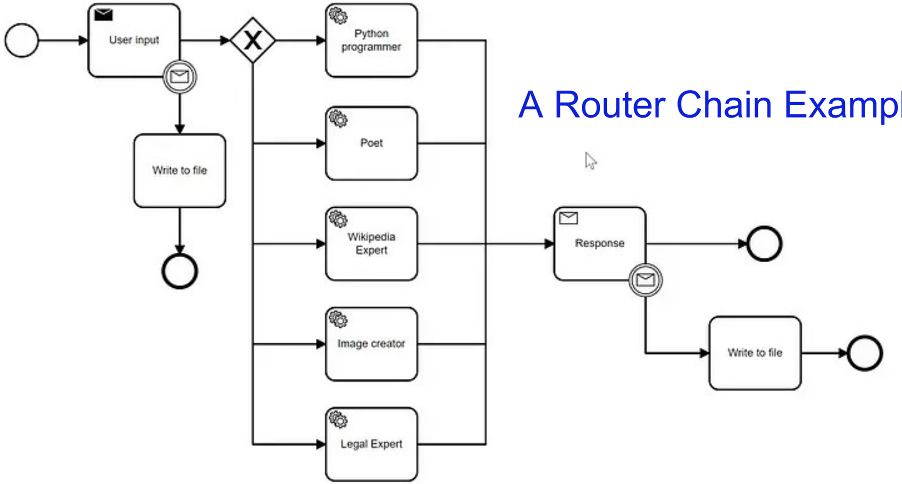
Router Chain:



A <u>Chain</u> is sequence of calls. Those calls can be to LLMs, External Tools, or Data Processing steps

Sequential Chain: Multiple i/p, Multiple o/p

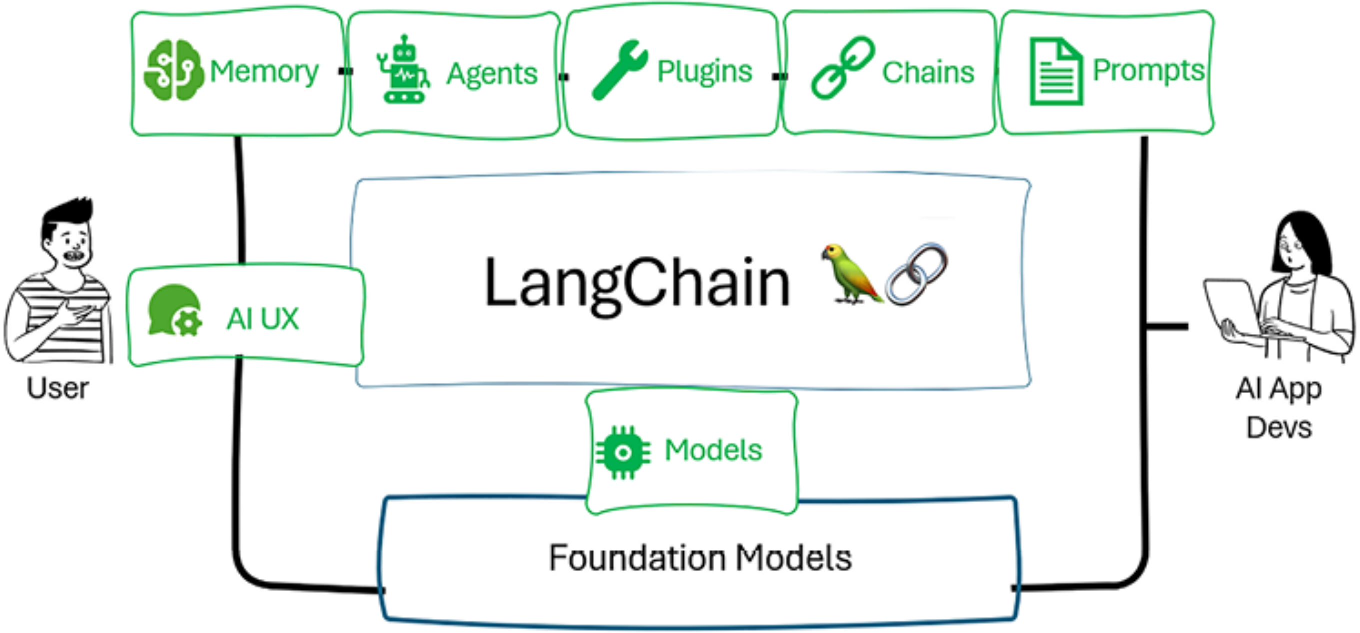




A Router Chain Example

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng.

LangChain: a framework for building LLM-based Applications



Components of LangChain

Source: Building LLM Powered Applications by Valentina Alto, Packt Publishing, May 2024

LangChain Memory Types

ConversationBufferMemory

- This memory allows for storing of messages and then extracts the messages in a variable.

ConversationBufferWindowMemory

- This memory keeps a list of the interactions of the conversation over time. It only uses the last K interactions.

ConversationTokenBufferMemory

- This memory keeps a buffer of recent interactions in memory, and uses token length rather than number of interactions to determine when to flush interactions.

ConversationSummaryMemory

- This memory creates a summary of the conversation over time.

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain Memory Types

Vector data memory

- Stores text (from conversation or elsewhere) in a vector database and retrieves the most relevant blocks of text.

Entity memories

- Using an LLM, it remembers details about specific entities.

You can also use multiple memories at one time. E.g., Conversation memory + Entity memory to recall individuals.

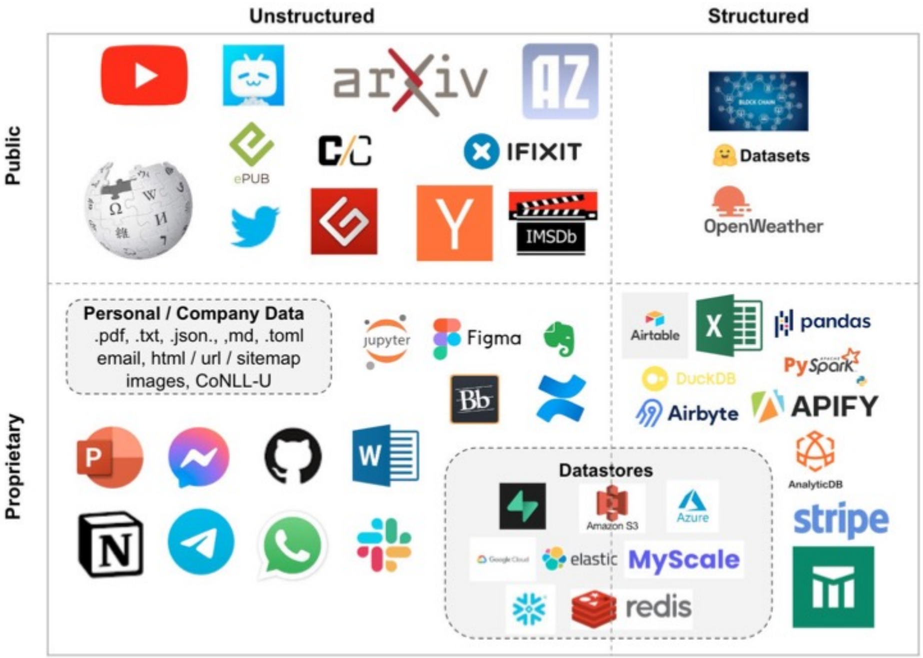
You can also store the conversation in a conventional database (such as key-value store or SQL)

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Document Loading with LangChain

- Loaders deal with the specifics of accessing and converting data
  - Accessing
    - Web Sites
    - Data Bases
    - YouTube
    - arXiv
    - ...
  - Data Types
    - PDF
    - HTML
    - JSON
    - Word, PowerPoint...
- Returns a list of `Document` objects:

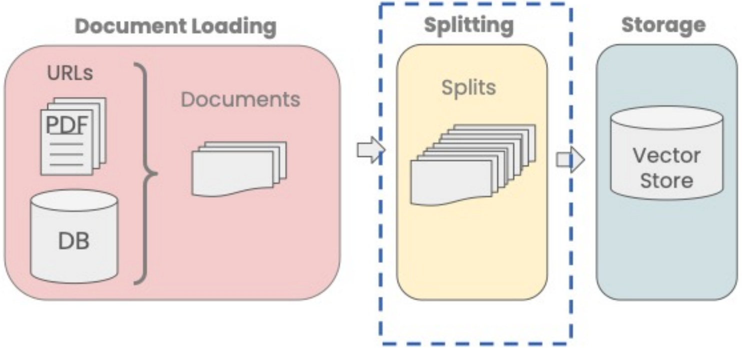
```
[
Document(page_content='MachineLearning-Lecture01 \nInstructor (Andrew Ng): Okay.
Good morning. Welcome to CS229....',
metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 0})
...
 Document(page_content='[End of Audio]  \nDuration: 69 minutes  ',
 metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 21})
]
```



Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Document Splitting with LangChain

- Splitting Documents into smaller chunks
  - Retaining meaningful relationships!



...
on this model. The Toyota Camry has a head-snapping
80 HP and an eight-speed automatic transmission that will
...

Chunk 1: on this model. The Toyota Camry has a head-snapping

Chunk 2: 80 HP and an eight-speed automatic transmission that will

Question: What are the specifications on the Camry?

Example Splitter

```
langchain.text_splitter.CharacterTextSplitter(
    separator: str = "\n\n"
    chunk_size=4000,
    chunk_overlap=200,
    length_function=<builtin function len>,
)
```

Methods:

create_documents() - Create documents from a list of texts.

split_documents() - Split documents.

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

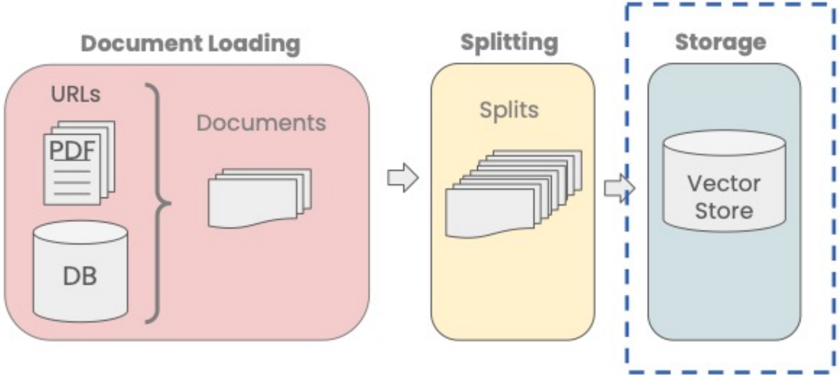
Different Types of Splitters in LangChain

langchain.text_splitter.

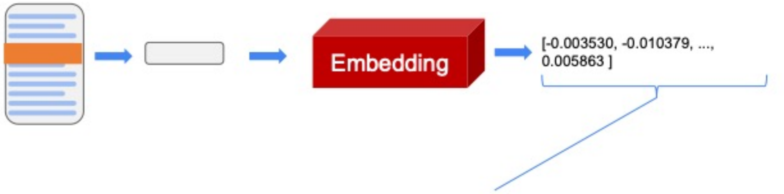
- **CharacterTextSplitter()**- Implementation of splitting text that looks at characters.
- **MarkdownHeaderTextSplitter**() - Implementation of splitting markdown files based on specified headers.
- **TokenTextSplitter**() - Implementation of splitting text that looks at tokens.
- **SentenceTransformersTokenTextSplitter**() - Implementation of splitting text that looks at tokens.
- ***RecursiveCharacterTextSplitter()*** - Implementation of splitting text that looks at characters. Recursively tries to split by different characters to find one that works.
- **Language()** – for CPP, Python, Ruby, Markdown etc
- **NLTKTextSplitter**() - Implementation of splitting text that looks at sentences using NLTK (Natural Language Tool Kit)
- **SpacyTextSplitter() -** Implementation of splitting text that looks at sentences using Spacy

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Support of Vector Store in LangChain

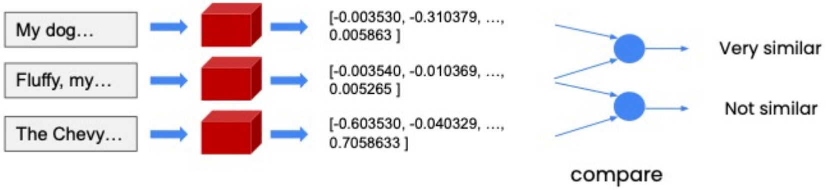


Embeddings



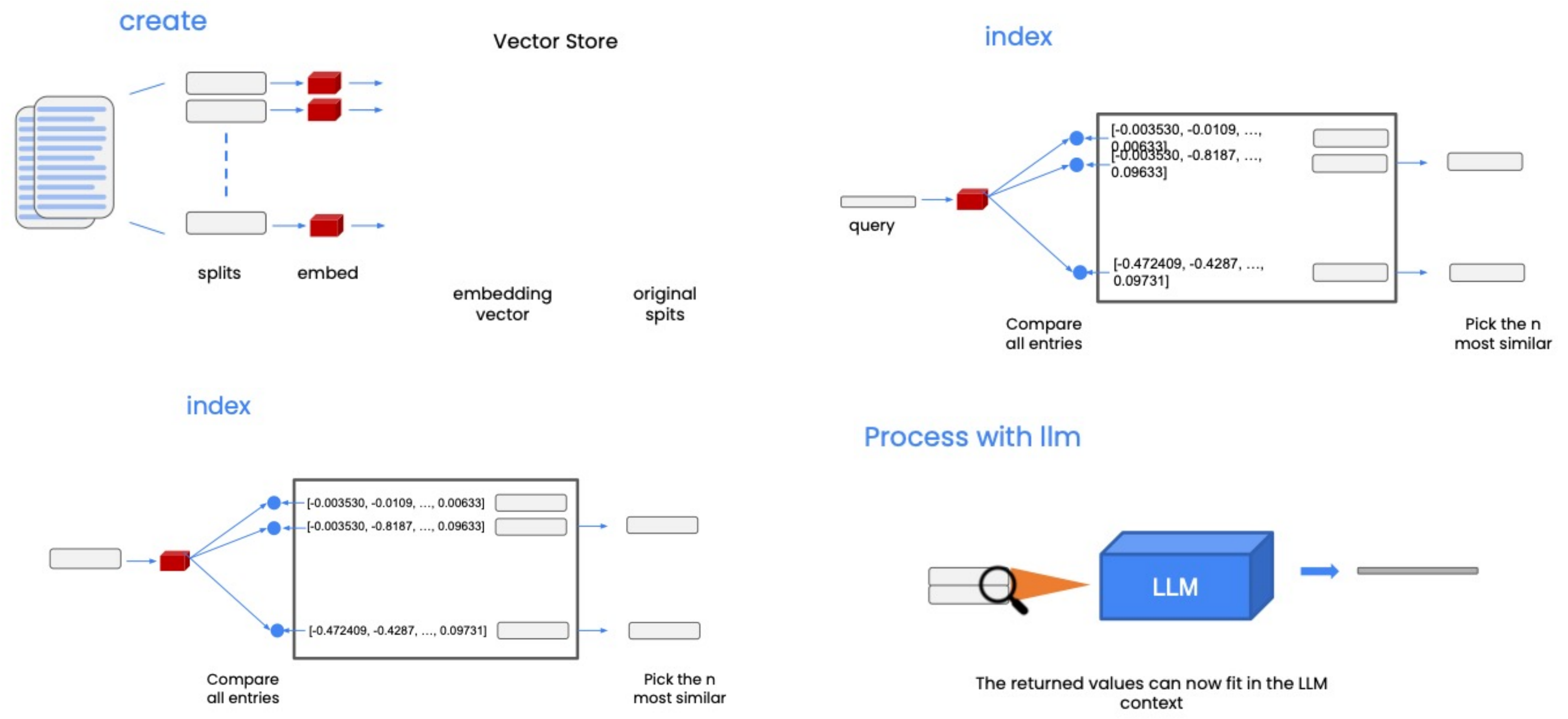
- Embedding vector captures content/meaning
- Text with similar content will have similar vectors

1) My dog Rover likes to chase squirrels.
2) Fluffy, my cat, refuses to eat from a can.
3) The Chevy Bolt accelerates to 60 mph in 6.7 seconds.



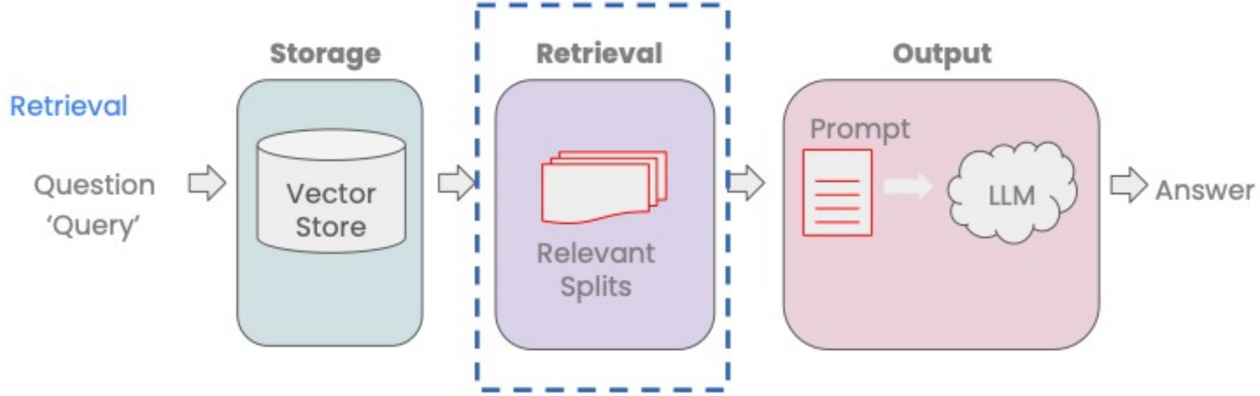
Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Basic RAG support w/ Vector Store in LangChain



Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Retrieval Support in LangChain

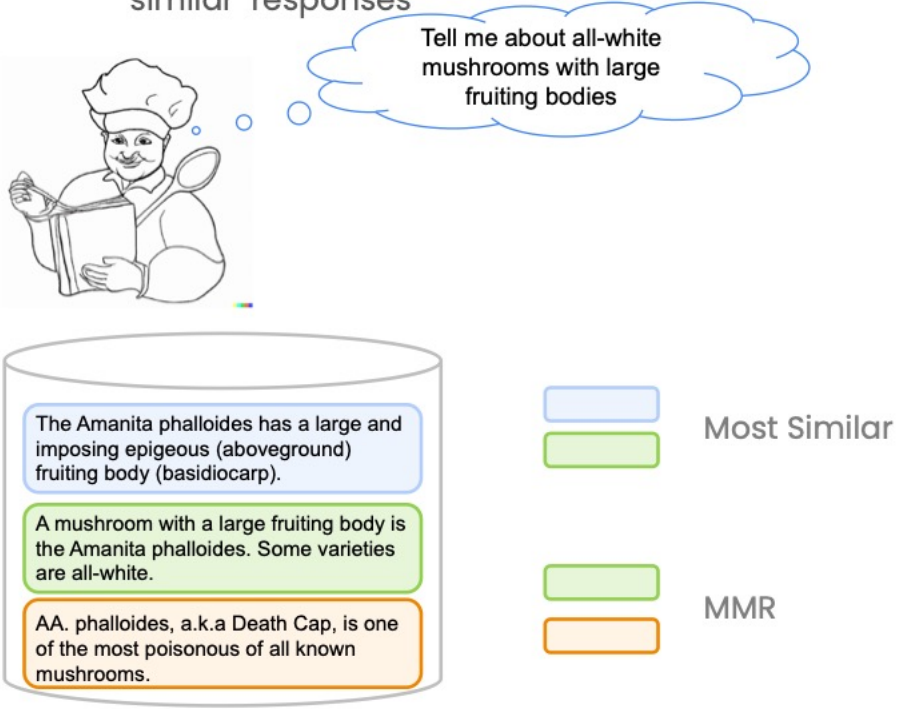


- Accessing/indexing the data in the vector store
  - Basic semantic similarity
  - Maximum marginal relevance
  - Including Metadata
- LLM Aided Retrieval

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

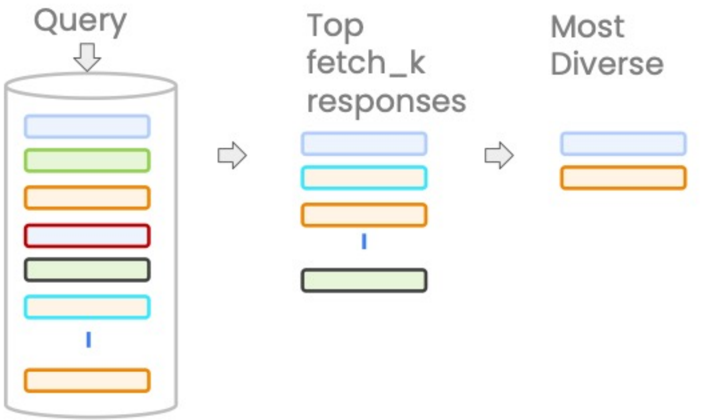
Retrieval based on Maximum Marginal Relevance (MMR)

- You may not always want to choose the most similar responses



MMR algorithm

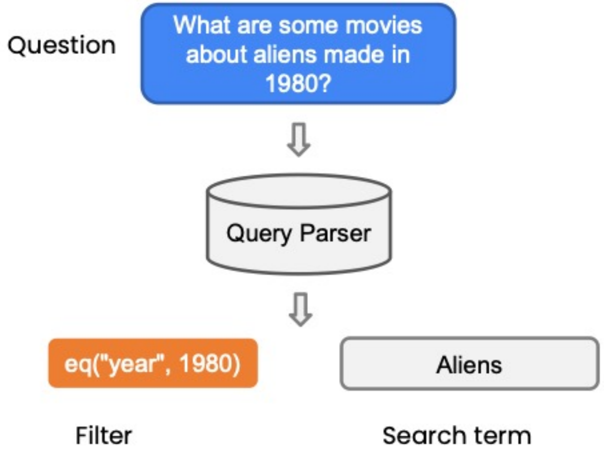
- Query the Vector Store
- Choose the `fetch_k` most similar responses
- Within those responses choose the `k` most diverse

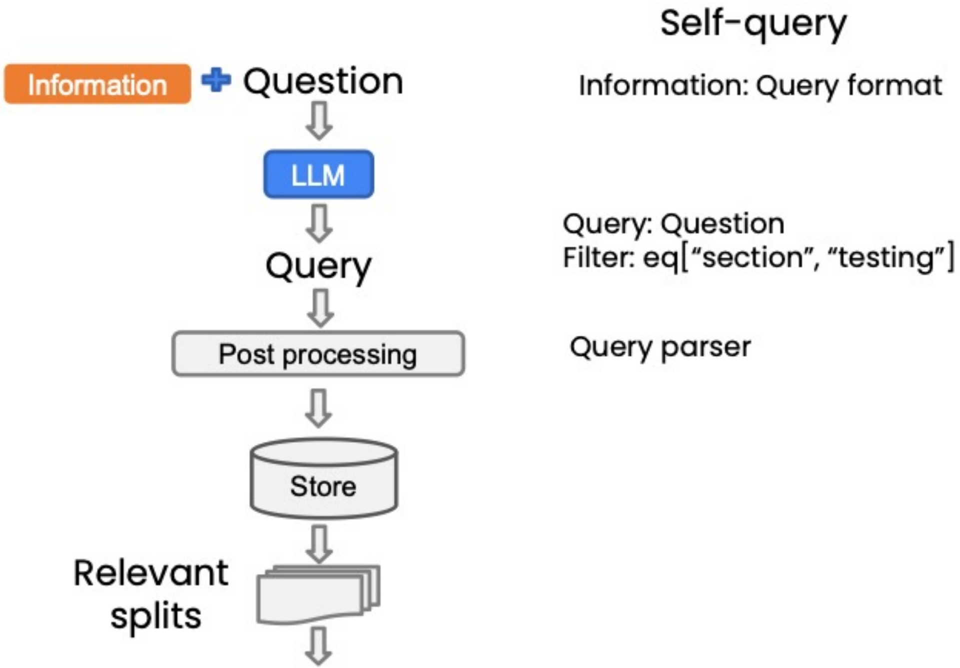


Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

LLM-aided Retrieval

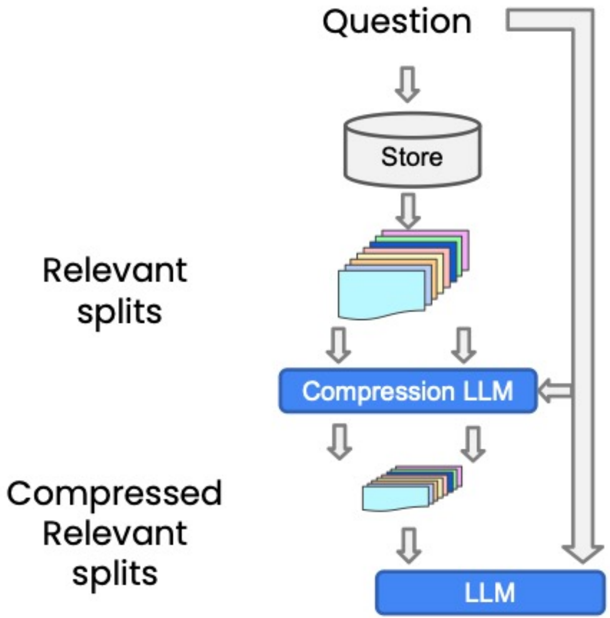
- There are several situations where the Query applied to the DB is more than just the Question asked.
- One is SelfQuery, where we use an LLM to convert the user question into a query





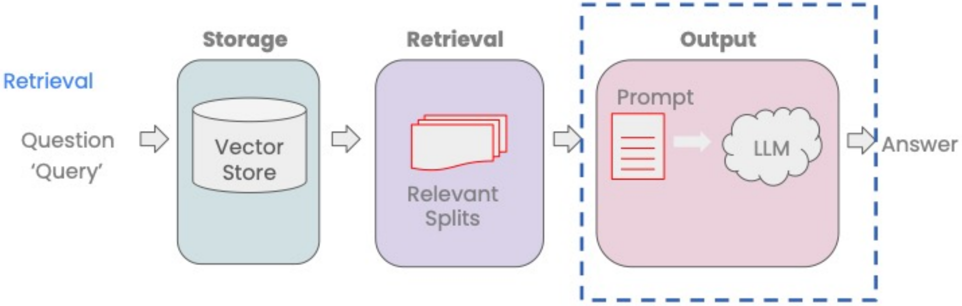
Compression Support

- Increase the number of results you can put in the context by shrinking the responses to only the relevant information.

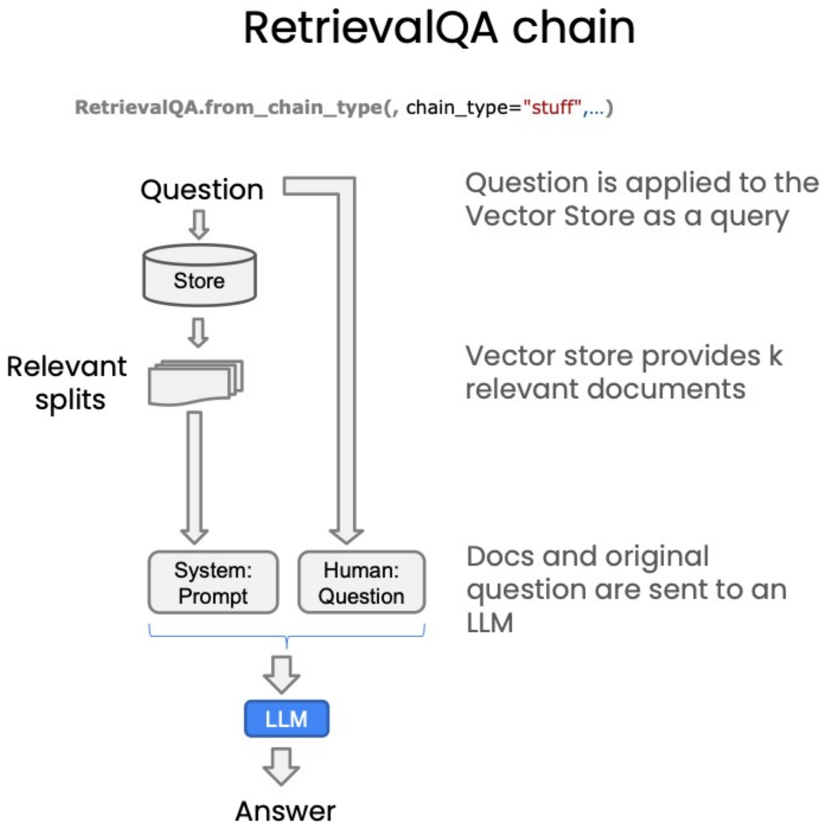


Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Q&A Support via Retrieval



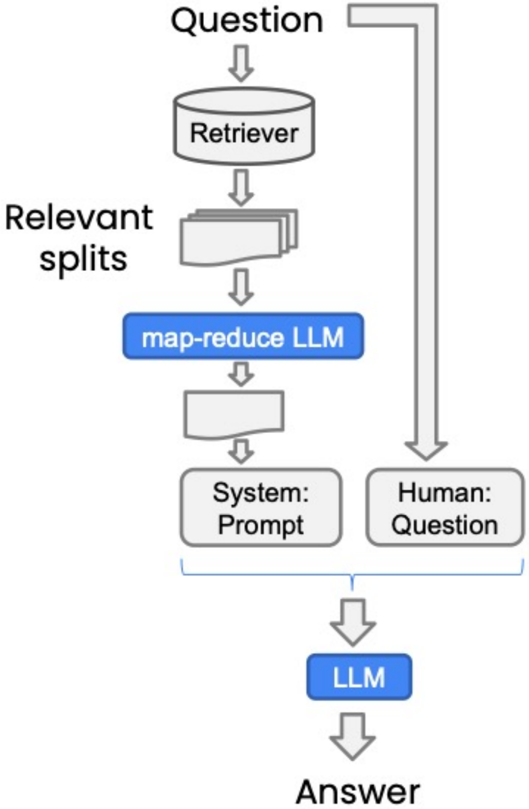
- Multiple relevant documents have been retrieved from the vector store
- Potentially compress the relevant splits to fit into the LLM context
- Send the information along with our question to an LLM to select and format an answer



Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

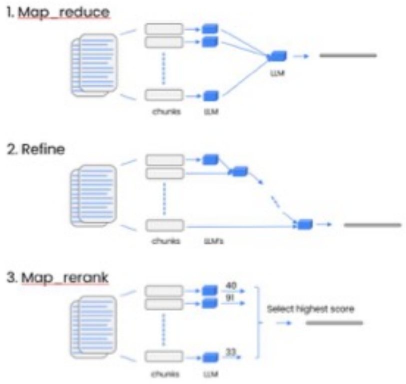
Retrieval Chain with LLM Selection

```
RetrievalQA.from_chain_type(, chain_type="map_reduce",...)
```



You many have too many docs to fit into an LLM context. The solution is to use an LLM to select the 'most relevant' information

3 additional methods



Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Facilitating Q&A over Documents with LangChain

Stuff method

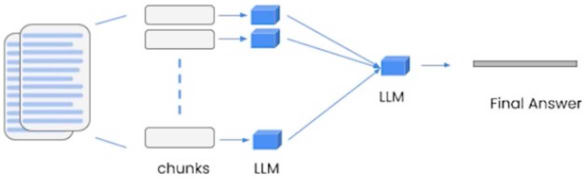
Stuffing is the simplest method. You simply stuff all data into the prompt as context to pass to the language model.

Pros: It makes a single call to the LLM. The LLM has access to all the data at once.

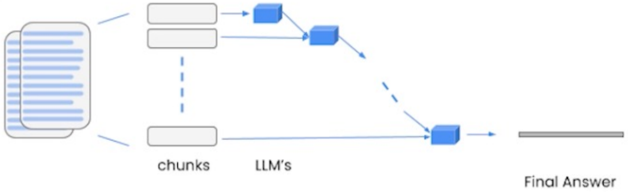
Cons: LLMs have a context length, and for large documents or many documents this will not work as it will result in a prompt larger than the context length.

3 additional methods

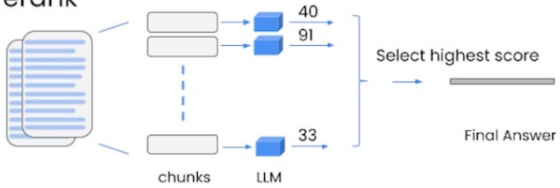
1. Map_reduce



2. Refine



3. Map_rerank



Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Agent Support in LangChain

Agent refers to the idea of using large language models as reasoning engine to determine which actions to take and in what order.

An action can be using a tool and observing its output and deciding what to return to the user.

```
from langchain.agents import
initialize_agent, AgentType

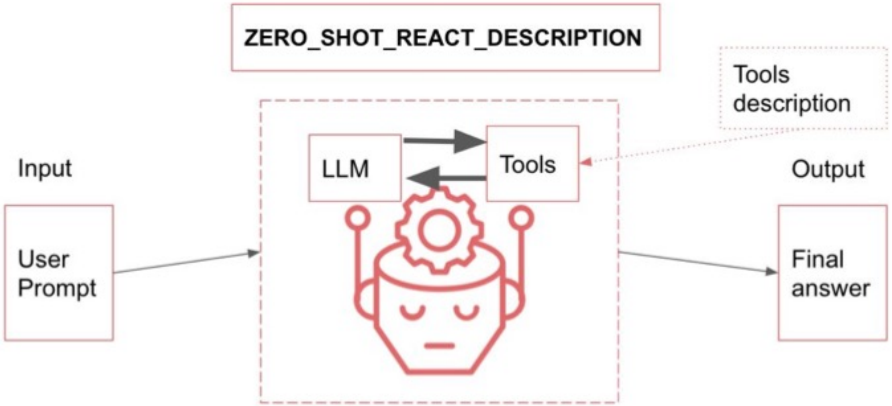
agent = initialize_agent(tools, llm,
       agent=AgentType.ZERO_SHOT_REACT_DESCRI
PTION,       verbose=True)
```

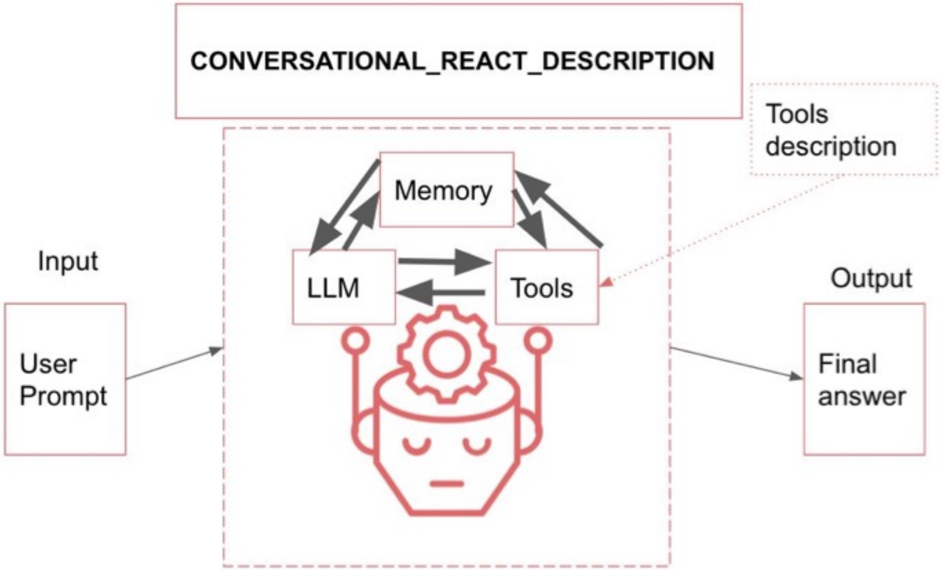
To construct a minimalist Agent, we need:

- PromptTemplate: this is responsible for taking the user input and previous steps and constructing a prompt to send to the language model
- Language Model: this takes the prompt constructed by the PromptTemplate and returns some output
- Output Parser: this takes the output of the Language Model and parses it into an AgentAction or AgentFinish object.

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

More Agent Support in LangChain





Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Implementation Examples with LangChain

Source: LangChain Chat with your Data – a short course by Harrison Chase, https://www.deeplearning.ai/short-courses/langchain-chat-with-your-data/

Interfacing with existing LLMs via "Models”

```
In [ ]: # !pip install openai
```

```
In [1]: import openai
        import os

        from dotenv import load_dotenv, find_dotenv
        _ = load_dotenv(find_dotenv()) # read local .env file
        openai.api_key = os.environ['OPENAI_API_KEY']
```

```
In [2]: def get_completion(prompt, model="gpt-3.5-turbo"):
            messages = [{"role": "user", "content": prompt}]
            response = openai.ChatCompletion.create(
                model=model,
                messages=messages,
                temperature=0,
            )
            return response.choices[0].message["content"]
```

```
In [3]: get_completion("What is 1+1?")

Out[3]: 'As an AI language model, I can tell you that the answer
        to 1+1 is 2.'
```


Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Models, Prompts and Parsers

```
In [4]: customer_email = """
        Arrr, I be fuming that me blender lid \
        flew off and splattered me kitchen walls \
        with smoothie! And to make matters worse, \
        the warranty don't cover the cost of \
        cleaning up me kitchen. I need yer help \
        right now, matey!
        """
```

```
In [5]: style = """American English \
        in a calm and respectful tone
        """
```

```
In [ ]: prompt = f"""Translate the text \
        that is delimited by triple backticks
        into a style that is {style}.
        text: ```{customer_email}```
        """
        print(prompt)
```


Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Setting up/ Using Prompt Templates in LangChain

```
In [6]: prompt = f"""Translate the text \
that is delimited by triple backticks
into a style that is {style}.
text: ```{customer_email}```
"""

print(prompt)
```

```
Translate the text that is delimited by triple backticks
into a style that is American English in a calm and respec
tful tone
.
text: ```
Arrr, I be fuming that me blender lid flew off and splatte
red me kitchen walls with smoothie! And to make matters wo
rse, \
the warranty don't cover th
n. I need yer help right no
```
```

```
In [7]: response = get_completion(prompt)
```

```
In [8]: response
```

```
Out[8]: "I'm really frustrated that my blender lid flew off and ma
de a mess of my kitchen walls with smoothie! To add insult
to injury, the warranty doesn't cover the cost of cleaning
up my kitchen. I could really use your help right now, my
friend."
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/
```

Setting up/ Using Prompt Templates in LangChain

```
In [9]: from langchain.chat_models import ChatOpenAI
```

```
In [11]: chat = ChatOpenAI(temperature=0.0)
         chat

Out[11]: ChatOpenAI(verbose=False, callbacks=None, callback_manager
         =None, client=<class 'openai.api_resources.chat_completio
         n.ChatCompletion'>, model_name='gpt-3.5-turbo', temperatur
         e=0.0, model_kwargs={}, openai_api_
         se=None, openai_organization=None,
         ax_retries=6, streaming=False, n=1,
```

```
In [12]: template_string = """Translate the text \
         that is delimited by triple backticks \
         into a style that is {style}. \
         text: ```{text}```
         """
```

```
In [13]: from langchain.prompts import ChatPromptTemplate

         prompt_template = ChatPromptTemplate.from_template(template_
```

```
In [14]: prompt_template.messages[0].prompt

Out[14]: PromptTemplate(input_variables=['style', 'text'], output_p
         arser=None, partial_variables={}, template='Translate the
         text that is delimited by triple backticks into a style th
         at is {style}. text: ```{text}```\n', template_format='f-s
         tring', validate_template=True)
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Setting up/ Using Prompt Templates in LangChain

```
In [16]: prompt_template.messages[0].prompt.input_variables

Out[16]: ['style', 'text']
```


```
In [17]: customer_style = """American Eng
in a calm and respectful tone
"""

In [ ]: customer_email = """
Arrr, I be fuming that me blende
flew off and splattered me kitch
with smoothie! And to make matte
the warranty don't cover the cos
cleaning up me kitchen. I need y
right now, matey!
"""
```


```
In [17]: customer_style = """American English \
in a calm and respectful tone
"""

In [18]: customer_email = """
Arrr, I be fuming that me blender lid \
flew off and splattered me kitchen walls \
with smoothie! And to make matters worse, \
the warranty don't cover the cost of \
cleaning up me kitchen. I need yer help \
right now, matey!
"""

In [19]: customer_messages = prompt_template.format_messages(
                             style=customer_style,
                             text=customer_email)

In [20]: print(type(customer_messages))
         print(type(customer_messages[0]))

<class 'list'>
<class 'langchain.schema.HumanMessage'>
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Setting up/ Using Prompt Templates in LangChain

```
In [21]: print(customer_messages[0])

content="Translate the text that is delimited by triple ba
ckticks into a style that is American English in a calm an
d respectful tone\n. text: `
blender lid flew off and spl
smoothie! And to make matter
ver the cost of cleaning up
ght now, matey!\n```\n" addi

In [22]: customer_response = chat(cus

In [23]: print(customer_response.cont

I'm really frustrated that m
e a mess of my kitchen walls
rustration, the warranty doe
g up my kitchen. Can you ple
```

```
In [23]: print(customer_response.content)

I'm really frustrated that my blender lid flew off and mad
e a mess of my kitchen walls with smoothie. To add to my f
rustration, the warranty doesn't cover the cost of cleanin
g up my kitchen. Can you please help me out, friend?

In [24]: service_reply = """Hey there customer, \
the warranty does not cover \
cleaning expenses for your kitchen \
because it's your fault that \
you misused your blender \
by forgetting to put the lid on before \
starting the blender. \
Tough luck! See ya!
"""

In [25]: service_style_pirate = """\
a polite tone \
that speaks in English Pirate\
"""
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

Prompting and Parsing with LangChain

In [24]:
```
service_reply = """Hey there customer, \
the warranty does not cover \
cleaning expenses for your kitchen \
because it's your fault that \
you misused your blender \
by forgetting to put the lid on before \
starting the blender. \
Tough luck! See ya!
"""
```

In [25]:
```
service_style_pirate = """\
a polite tone \
that speaks in English Pirate\
"""
```

In [26]:
```
service_messages = prompt_template.format_messages(
    style=service_style_pirate,
    text=service_reply)

print(service_messages[0].content)
```

```
Translate the text that is delimited by
nto a style that is a polite tone that
irate. text: ```Hey there customer, the
cover cleaning expenses for your kitche
fault that you misused your blender by
he lid on before starting the blender.
```
```

In [27]:
```
service_response = chat(service_messages)
print(service_response.content)
```

```
Ahoy there, matey! I must kindly inform ye that the warran
ty be not coverin' the expenses o' cleaning yer galley, as
'tis yer own fault fer misusin' yer blender by forgettin'
to put the lid on afore startin' it. Aye, tough luck! Fare
well, me hearty!
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/
```

Why LangChain introduces Prompt Templates ?

~~~~
prompt = """
Your task is to determine if
the student's solution is
correct or not.
~~~~

~~~~
To solve the problem do the following:
- First, work out your own solution to the problem.
- Then compare your solution to the student's solution
and evaluate if the student's solution is correct or not.
...
Use the following format:
Question:
```
question here
```
Student's solution:
```
student's solution here
```
Actual solution:
```
...
steps to work out the solution and your solution here
```
Is the student's solution the same as actual solution \
just calculated:
```
yes or no
```
Student grade:
```
correct or incorrect
```

Question:
```
{question}
```
Student's solution:
```
{student_solution}
```
Actual solution:
"""
~~~~

Prompts can be long and detailed.

Reuse good prompts when you can!

LangChain also provides prompts for common operations.

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain o/p Parsing works w/ Prompt Templates

```
EXAMPLES = ["""
Question: What is the elevation range
for the area that the eastern sector
of the Colorado orogeny extends into?

Thought: I need to search Colorado orogeny, find
the area that the eastern sector of the Colorado
orogeny extends into, then find the elevation range
of the area.

Action: Search[Colorado orogeny]

Observation: The Colorado orogeny was an
episode of mountain building (an orogeny) in
Colorado and surrounding areas.

Thought: It does not mention the eastern sector.
So I need to look up eastern sector.
Action: Lookup[eastern sector]

...

Thought: High Plains rise in elevation from
around 1,800 to 7,000 ft, so the answer is 1,800 to
7,000 ft.

Action: Finish[1,800 to 7,000 ft]""",
]
```

LangChain library functions parse the LLM's output assuming that it will use certain keywords.

Example here uses **Thought**, **Action**, **Observation** as keywords for Chain-of-Thought Reasoning. (ReAct)

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain Output Parsing: Extraction and Formatting

```
In [28]: {
           "gift": False,
           "delivery_days": 5,
           "price_value": "pretty affordable!"
         }

Out[28]: {'gift': False, 'delivery_days': 5, 'price_value': 'pretty
         affordable!'}
```

```
In [ ]: customer_review = """\
This leaf blower is pretty amazing.  It has four settings:\
candle blower, gentle breeze, windy city, and tornado. \
It arrived in two days, just in time for my wife's \
anniversary present. \
I think my wife liked it so much she was speechless. \
So far I've been the only one using it, and I've been \
using it every other morning to clear the leaves on our lawr
It's slightly more expensive than the other leaf blowers \
out there, but I think it's worth it for the extra features.
"""

review_template = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? Ans
delivery_days: How many days did it take for the product to
price_value: Extract any sentences about the value or price,

Format the output as JSON with the following keys:
gift
delivery_days
price_value

text: {text}
"""
```


Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain Output Parsing: Extraction and Formatting

In [30]:
```
from langchain.prompts import ChatPromptTemplate

prompt_template = ChatPromptTemplate.from_template(review_te
print(prompt_template)
```

```
input_variables=['text'] output_parser=None partial_variab
les={} messages=[HumanMessagePromptTemplate(prompt=PromptT
emplate(input_variables=['text'], output_parser=None, part
ial_variables={}, template='For the following text, extrac
t the following information:\n\ngift: Was the item purchas
ed as a gift for someone else? Answer True if yes, False i
f not or unknown.\ndelivery_days: How many days did it tak
e for the product to arrive? If this information is not fo
und, output -1.\nprice_value: Extract any sentences about
the value or price, and output them
ython list.\n\nFormat the output as
g keys:\ngift\ndelivery_days\nprice_
\n', template_format='f-string', val
additional_kwargs={})]
```

In [31]:
```
messages = prompt_template.format_messages(text=customer_rev
chat = ChatOpenAI(temperature=0.0)
response = chat(messages)
print(response.content)
```

```
{
    "gift": true,
    "delivery_days": 2,
    "price_value": ["It's slightly more expensive than the
other leaf blowers out there, but I think it's worth it fo
r the extra features."]
}
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain Output Parsing: Extraction and Formatting

```
In [31]: messages = prompt_template.format_messages(text=customer_rev
         chat = ChatOpenAI(temperature=0.0)
         response = chat(messages)
         print(response.content)
```

```
{
    "gift": true,
    "delivery_days": 2,
    "price_value": ["It's slight
other leaf blowers out there, but
r the extra features."]
}
```

```
In [33]: response.content.get('gift')
```

```
---------------------------------------------------------------
------------------
AttributeError                            Traceback (most
recent call last)
Cell In[33], line 1
----> 1 response.content.get('gift')

AttributeError: 'str' object has no attribute 'get'
```

```
In [34]: from langchain.output_parsers import ResponseSchema
         from langchain.output_parsers import StructuredOutputParser
```

```
In [ ]: gift_schema = ResponseSchema(name="gift", description="Was t
        delivery_days_schema = ResponseSchema(name="delivery_days",
        price_value_schema = ResponseSchema(name="price_value", desc

        response_schemas = [gift_schema, delivery_days_schema,price_
```

Source: LangChain for LLM Application Development – a short course by Harrison Chase, Andrew Ng, https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/

LangChain Output Parsing: Extraction and Formatting

```
In [34]: from langchain.output_parsers import ResponseSchema
         from langchain.output_parsers import StructuredOutputParser
```

```
In [ ]: gift_schema = ResponseSchema(name="gift", description="Was t
        delivery_days_schema = ResponseSchema(name="delivery_days",
        price_value_schema = ResponseSchema(name="price_value", desc

        response_schemas = [gift_schema, delivery_days_schema,price_
```

```
In [38]: format_instructions = output_parser.get_format_instructions(
```

```
In [39]: print(format_instructions)
```

```
The output should be a markdown code sn
the following schema, including the lea
"\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item
t for someone else? Answer True if yes,
known.
        "delivery_days": string  // How
ake for the product to arrive? If this
found, output -1.
        "price_value": string  // Extract any sentences ab
```

```
In [ ]: review_template_2 = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? Ans
delivery_days: How many days did it take for the product to
price_value: Extract any sentences about the value or price,

text: {text}

{format_instructions}
"""

prompt = ChatPromptTemplate.from_template(template=review_te

messages = prompt.format_messages(text=customer_review,
                                  format_instructions=format_i
```
```

LangChain Output Parsing: Extraction and Formatting

```
In [41]: print(messages[0].content)

For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? A
nswer True if yes, False if not or unknown.
delivery_days: How many days did it take for the product t
o arrive? If this information is not found, output -1.
price_value: Extract any sentences about the value or pric
e, and output them as a comma separated Python list.

text: This leaf blower is pretty amazing.  It has four set
tings:candle blower, gentle breeze, windy city, and tornad
o. It arrived in two days, just in time for my wife's anni
versary present. I think my wife liked it so much she was
speechless. So far I've been the only one using it, and
I've been using it every other morning to clear the leaves
on our lawn. It's slightly more expensive than the other l
eaf blowers out there, but I think it's worth it for the e
xtra features.


The output should be a markdown code snippet formatted in
the following schema, including the leading and trailing
"\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item purchased as a gif
t for someone else? Answer True if yes, False if not or un
known.
        "delivery_days": string  // How many days did it t
ake for the product to arrive? If this information is not
found, output -1.
        "price_value": string  // Extract any sentences ab
out the value or price, and output them as a comma separat
ed Python list.
```

```
In [42]: response = chat(messages)

In [43]: print(response.content)

```json
{
        "gift": true,
        "delivery_days": "2",
        "price_value": ["It's slightly more expensive than
the other leaf blowers out there, but I think it's worth i
t for the extra features."]
}
```

In [44]: output_dict = output_parser.parse(response.content)

In [45]: output_dict

Out[45]: {'gift': True,
 'delivery_days': '2',
 'price_value': ["It's slightly more expensive than the ot
her leaf blowers out there, but I think it's worth it for
the extra features."]}

In [46]: type(output_dict)

Out[46]: dict

In [49]: output_dict.get('delivery_days')

Out[49]: '2'
```
```

Enpowering LLM-based Applications with External Actions

How ChatGPT interacts w/ the Outside World via Plugins

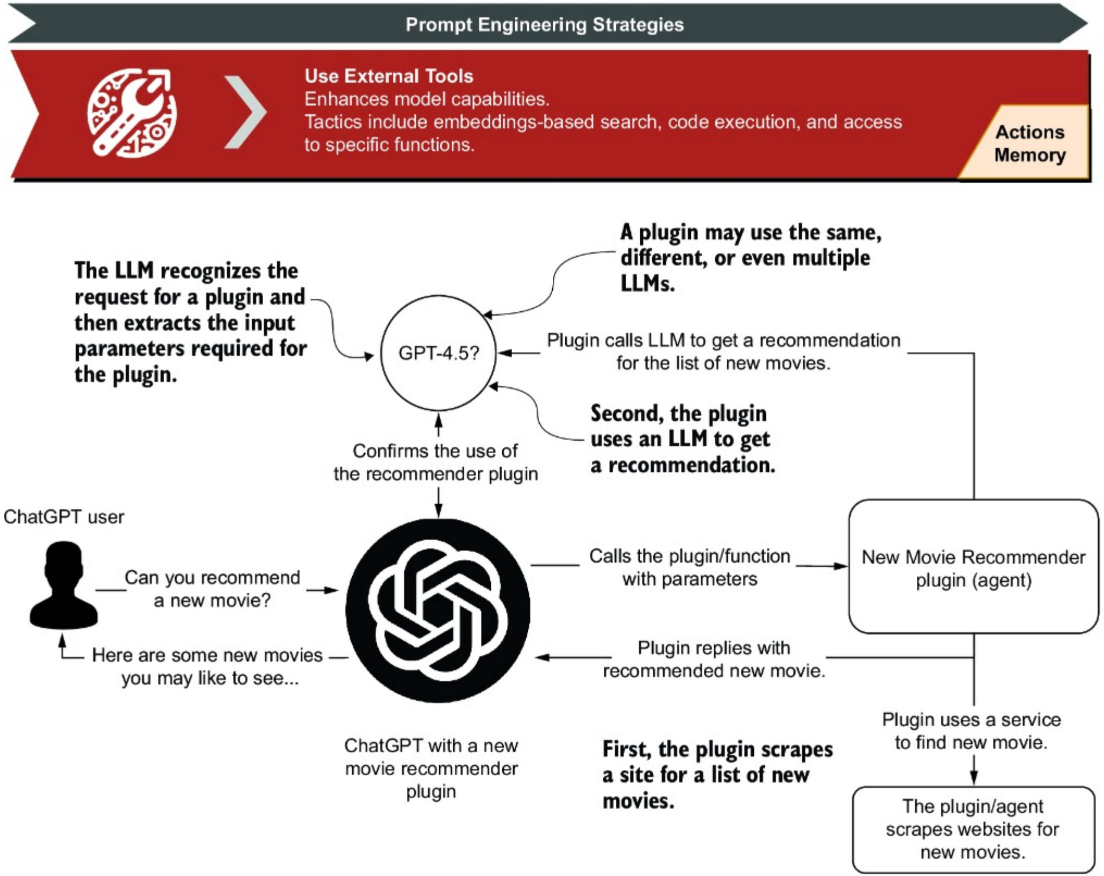
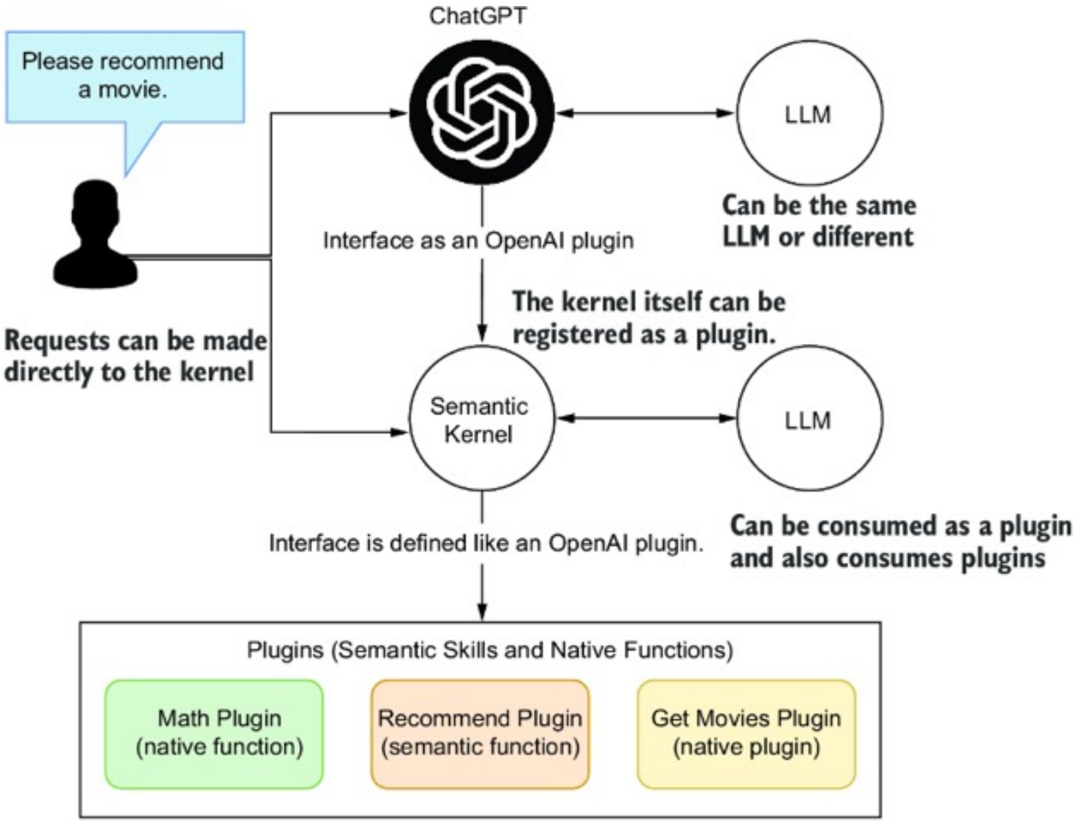


Figure 5.1 How a ChatGPT plugin operates and how plugins and other external tools (e.g., APIs) align with the Use External Tools prompt engineering strategy

Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Microsoft's Semantic Kernel as a Plugin for ChatGPT



Semantic Kernel (SK) is another open source project from Microsoft intended to help build AI applications. At its core, the project is best used to define actions, or what the platform calls semantic plugins, which are wrappers for skills and functions.

This figure shows how the SK can be used as a plugin and a consumer of OpenAI plugins.

The SK relies on the OpenAI plugin definition to define a plugin. That way, it can consume and publish itself or other plugins to other systems.

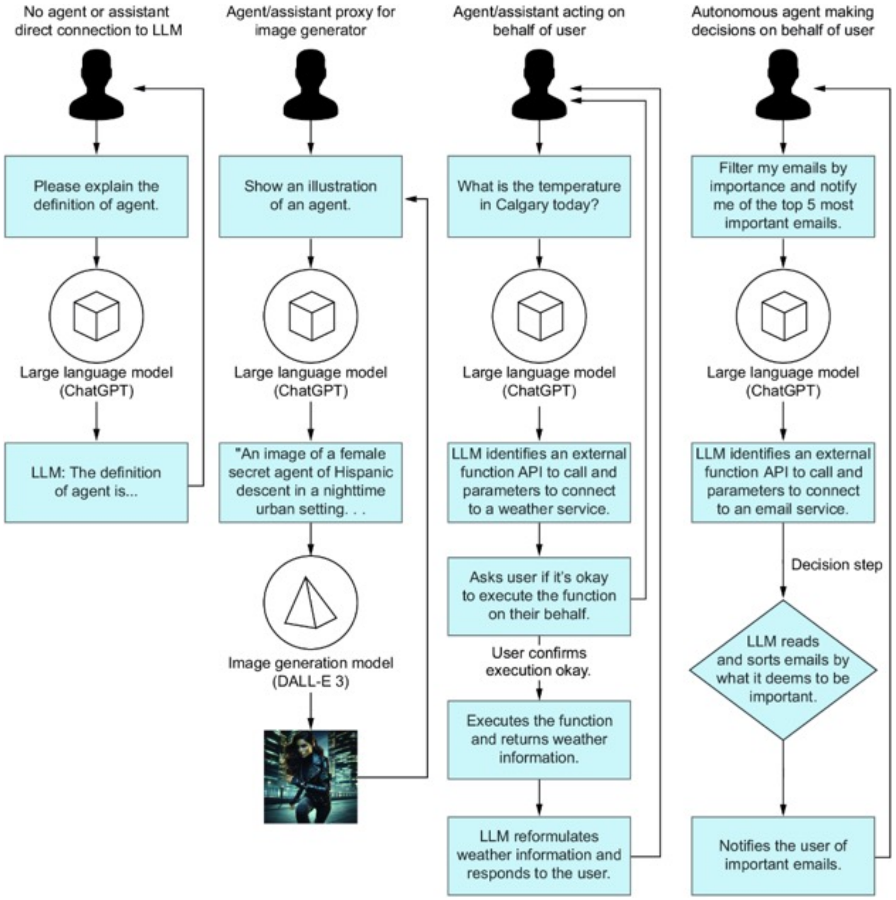
An OpenAI plugin definition maps precisely to the function definitions. This means that SK is the orchestrator of API tool calls, aka plugins.

That also means that SK can help organize multiple plugins with a chat interface or an agent.

Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

From LLM-powered Application to AI Agent

From “LLM-powered Application” to “AI Agent”



Source: AI Agents in Action,
by Michael Lanham, Feb 2025,
Manning Publications

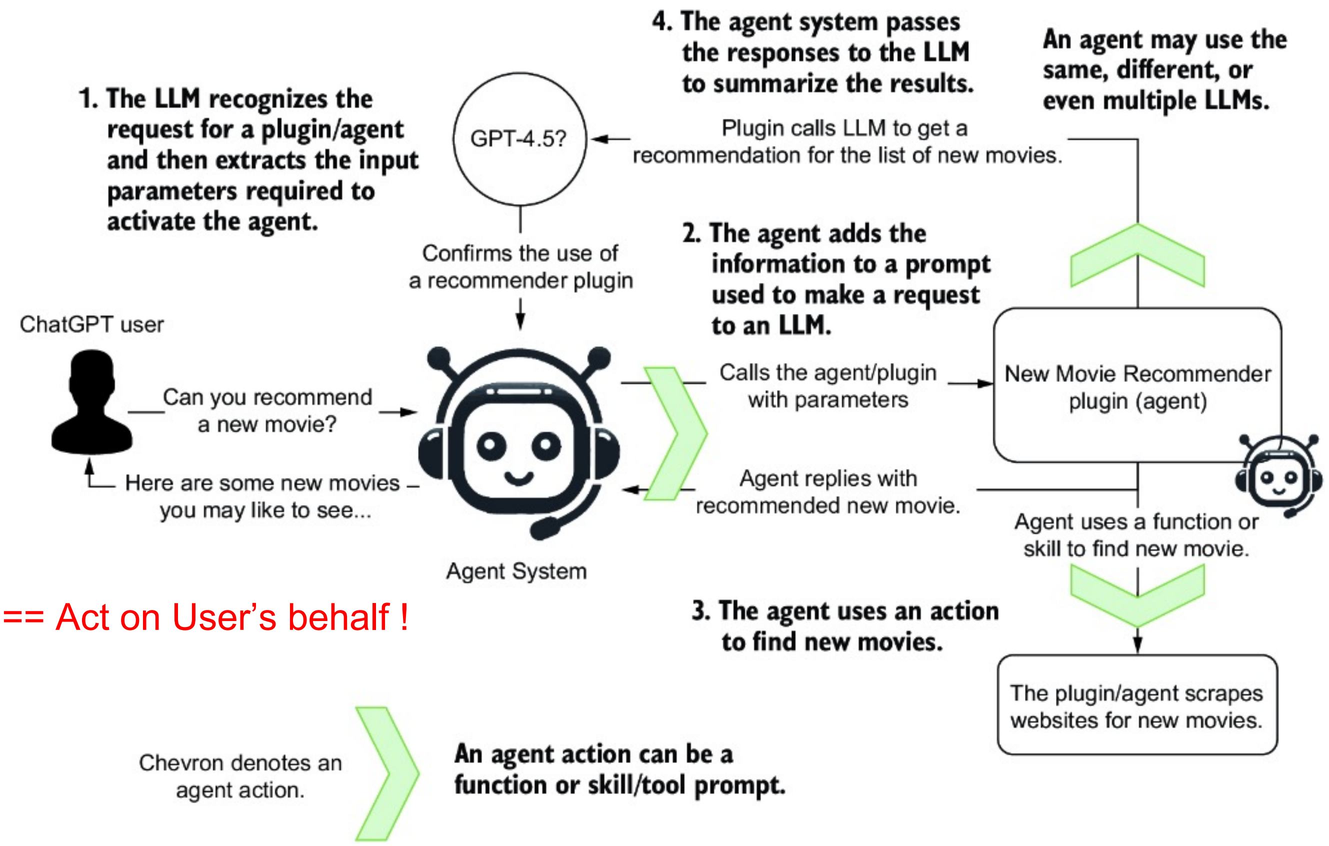
Agent == Act on “YOUR” behalf
at various level of autonomy !

“YOUR” can be == Users

“YOUR” can also be an Ext. App

The differences between the LLM interactions from direct action compared to using proxy agents, agents, and autonomous agents

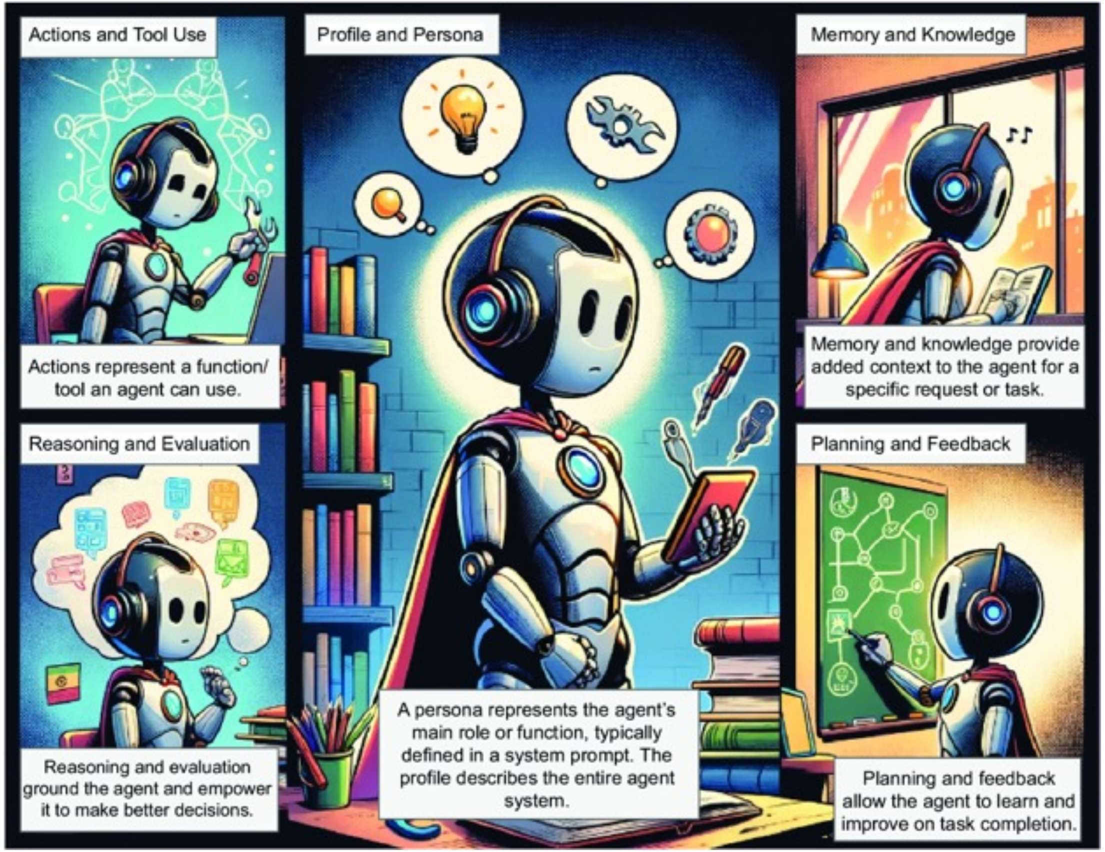
How an Agent uses Actions to perform External Tasks



Agent == Act on User's behalf !

Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

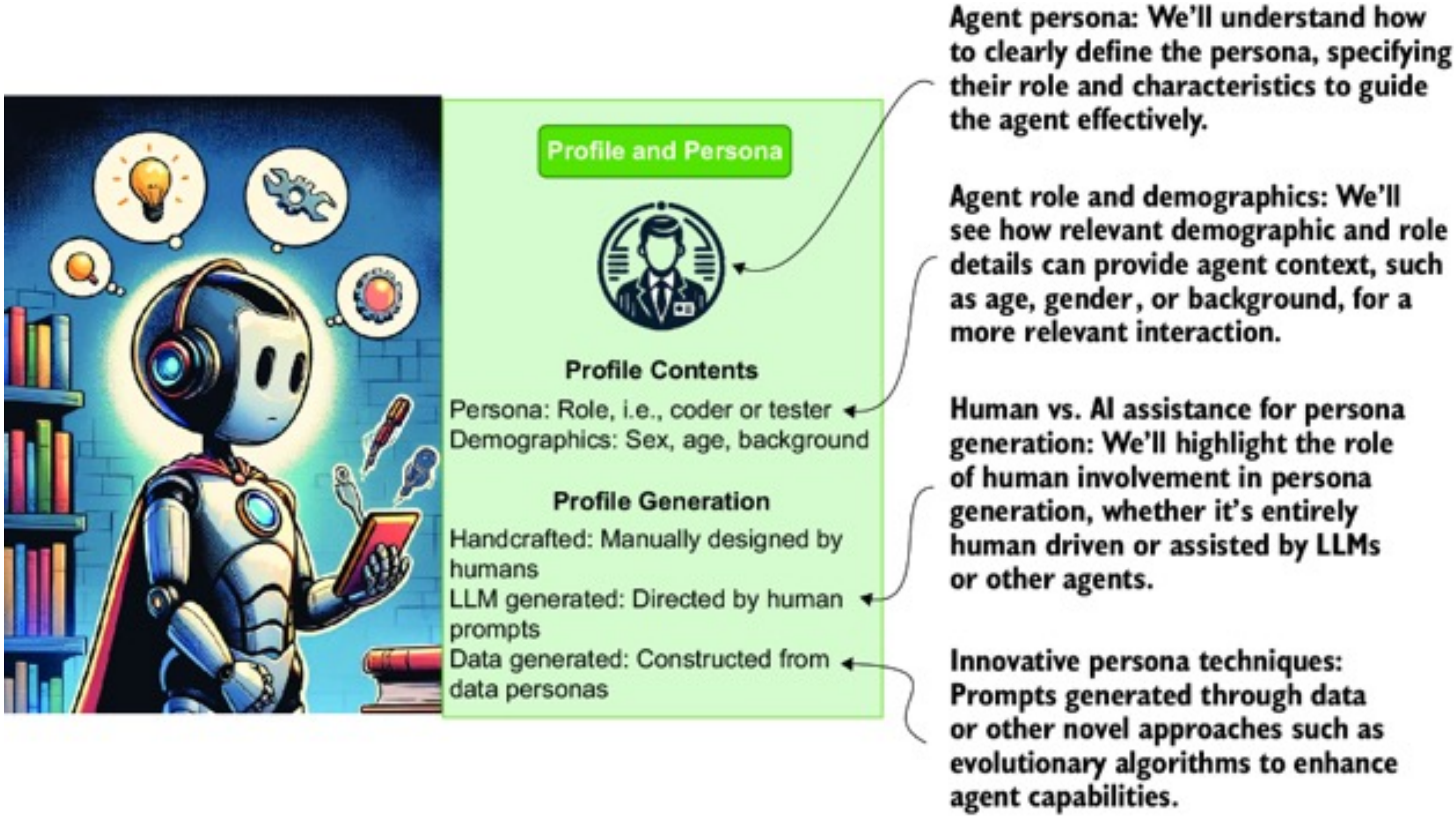
Key Functional Components of a Single-Agent System



Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

In-Depth Look at the
Key Functional Components of an Agent

The Profile and Persona Component of an Agent



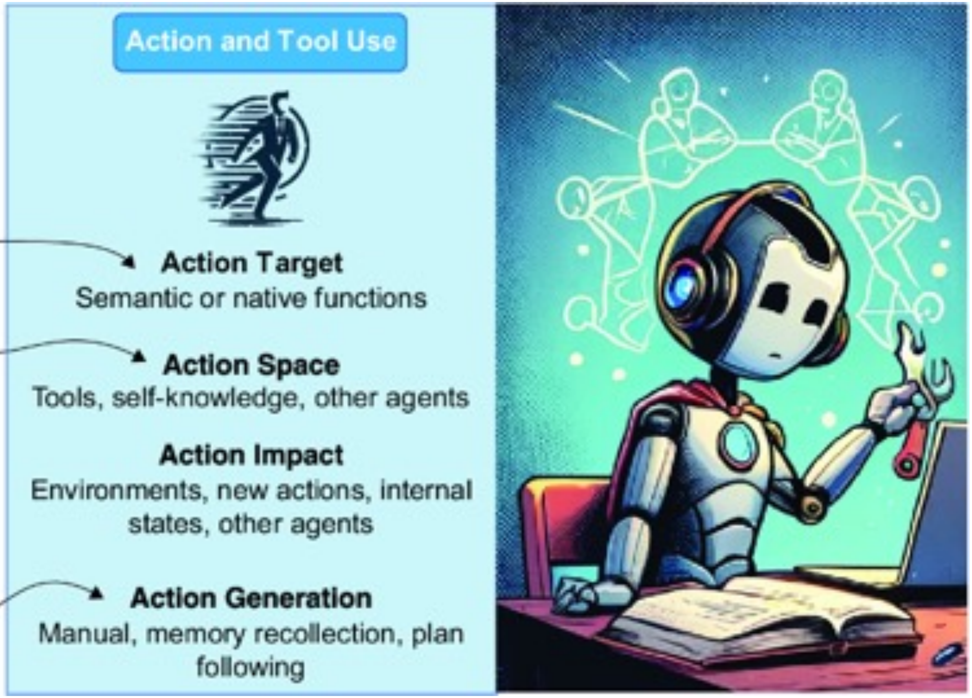
Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Agent Action and Tool Use

Action targets: We'll learn the importance of defining action targets, whether for task completion, exploration, or communication, to clarify the agent's objectives.

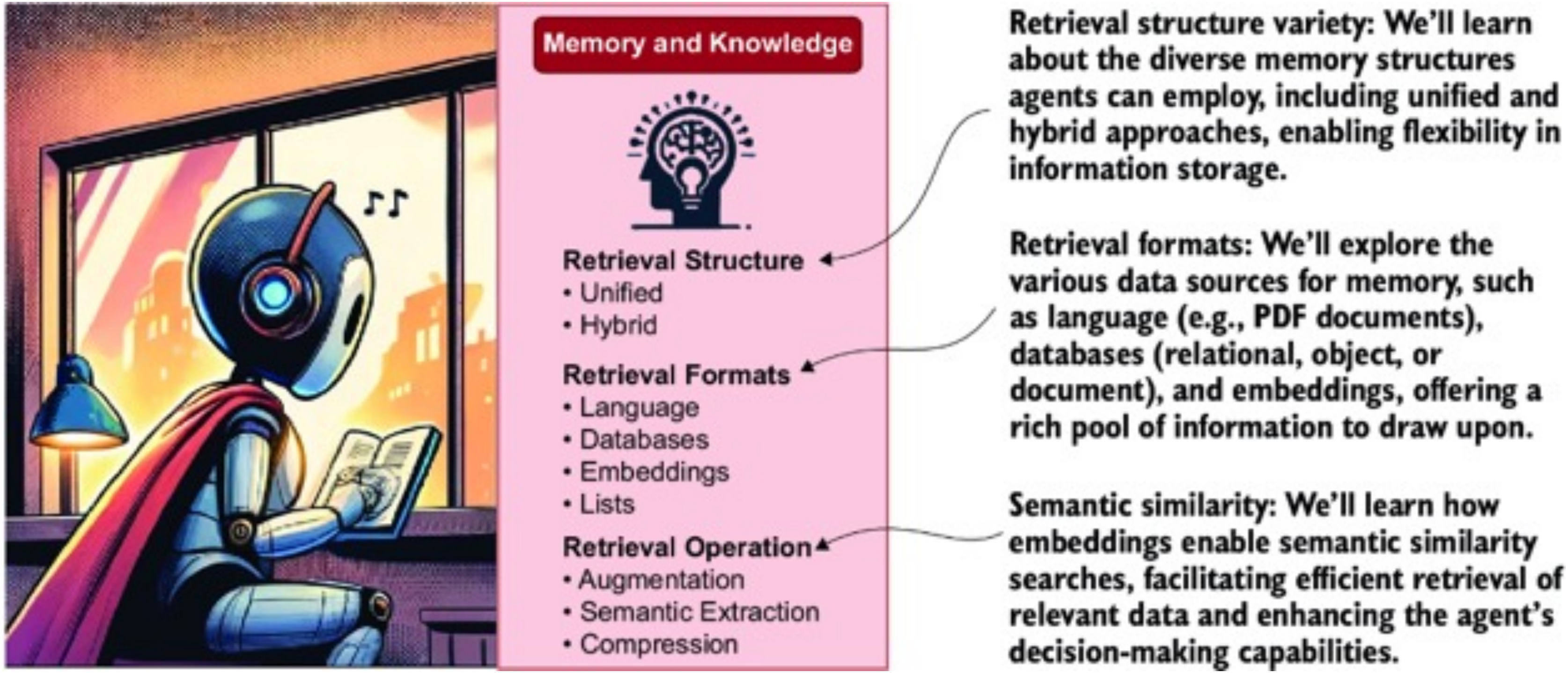
Action space and impact: We'll learn the significance of understanding how actions affect task completion and their effect on the agent's environment, internal states, and self-knowledge.

Action generation methods: We'll see the various ways actions can be generated, such as manual creation, memory recollection, or plan following, to illustrate the diversity of agent behaviors.



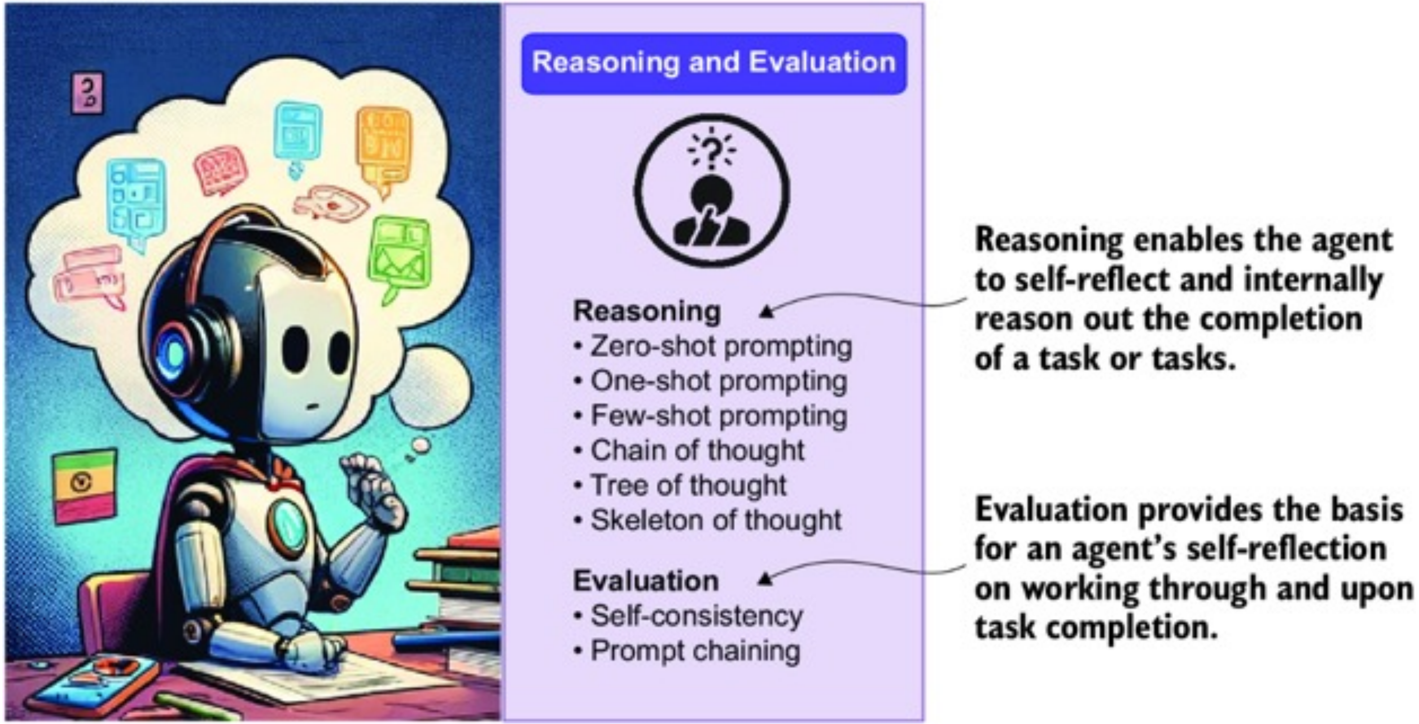
Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Agent Memory and Knowledge



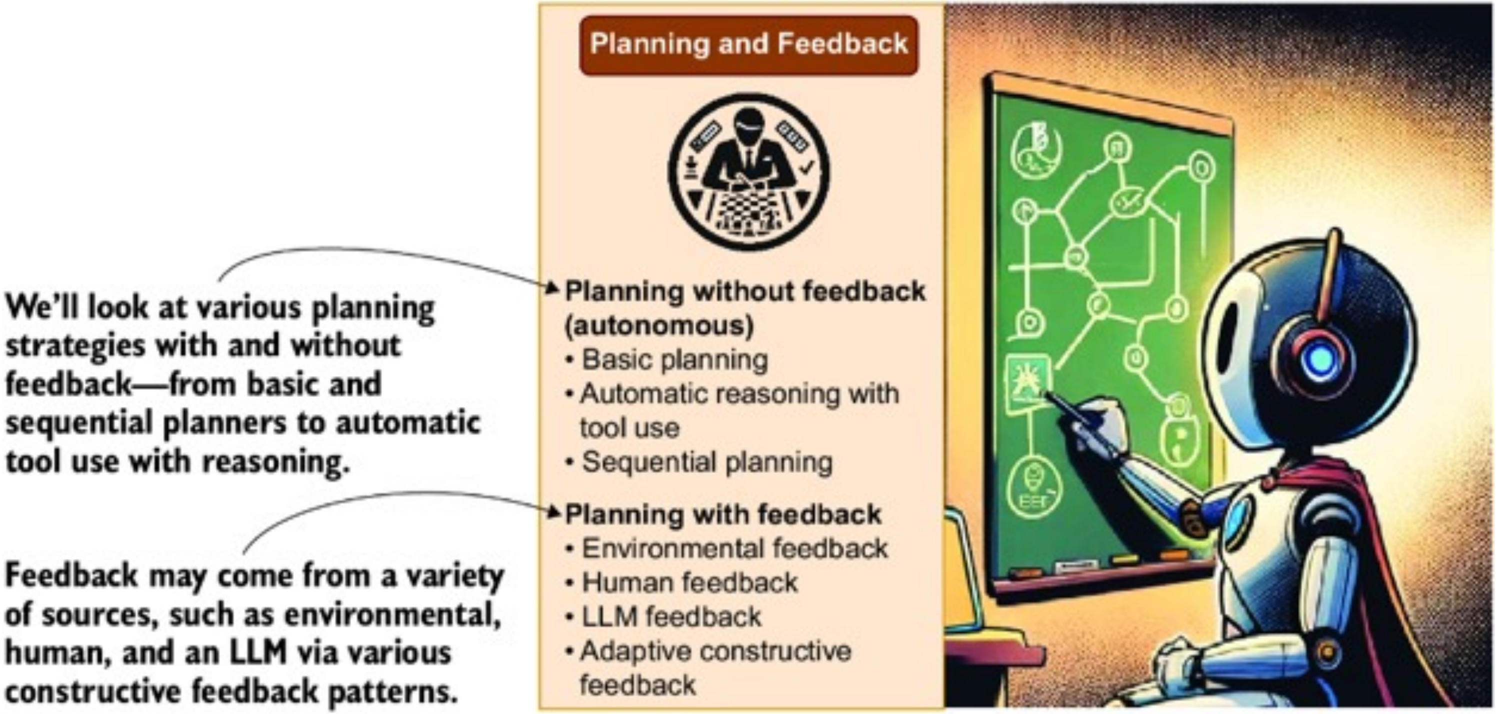
Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Reasoning and Evaluation Functions of an Agent



Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Planning and Feedback Functions of an Agent



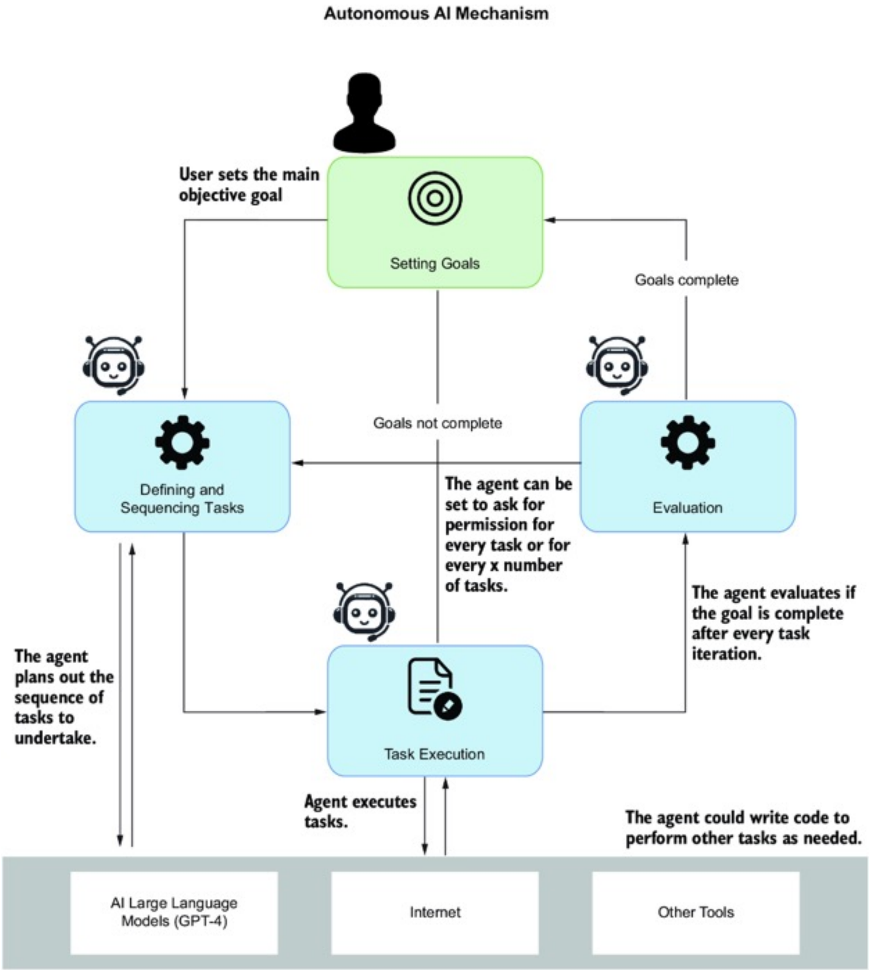
Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Comparisons of Agentic AI Frameworks

Framework	Best For	Deployment	Scalability	Current Adoption
LangChain & LangGraph	Enterprise AI apps, multi-agent workflows	Self-hosted, Cloud	High – widely used in production	Most adopted
Autogen & Semantic Kernel	Microsoft ecosystem, scalable AI apps	Azure, Self-hosted	High – enterprise-scale	Growing in enterprises
LlamaIndex	AI-powered search, RAG	Cloud, Self-hosted	High – efficient data processing	Strong in AI search
AutoGPT	No-code AI automation, continuous agents	Cloud-first, some self-hosted options	Medium – automation focused	Popular for prototyping
CrewAI	Workflow-based multi-agent AI apps	Cloud, Self-hosted	Medium – still early-stage	Fast-growing
PydanticAI	FastAPI & Pydantic-based AI apps	Self-hosted	Low – lightweight framework	Niche adoption
Spring AI	Java-based AI applications	Self-hosted, Cloud	Medium – depends on Java ecosystem	Popular in Java world
Haystack	LLM-powered search & RAG	Self-hosted, Cloud	High – built for production	Strong in search applications

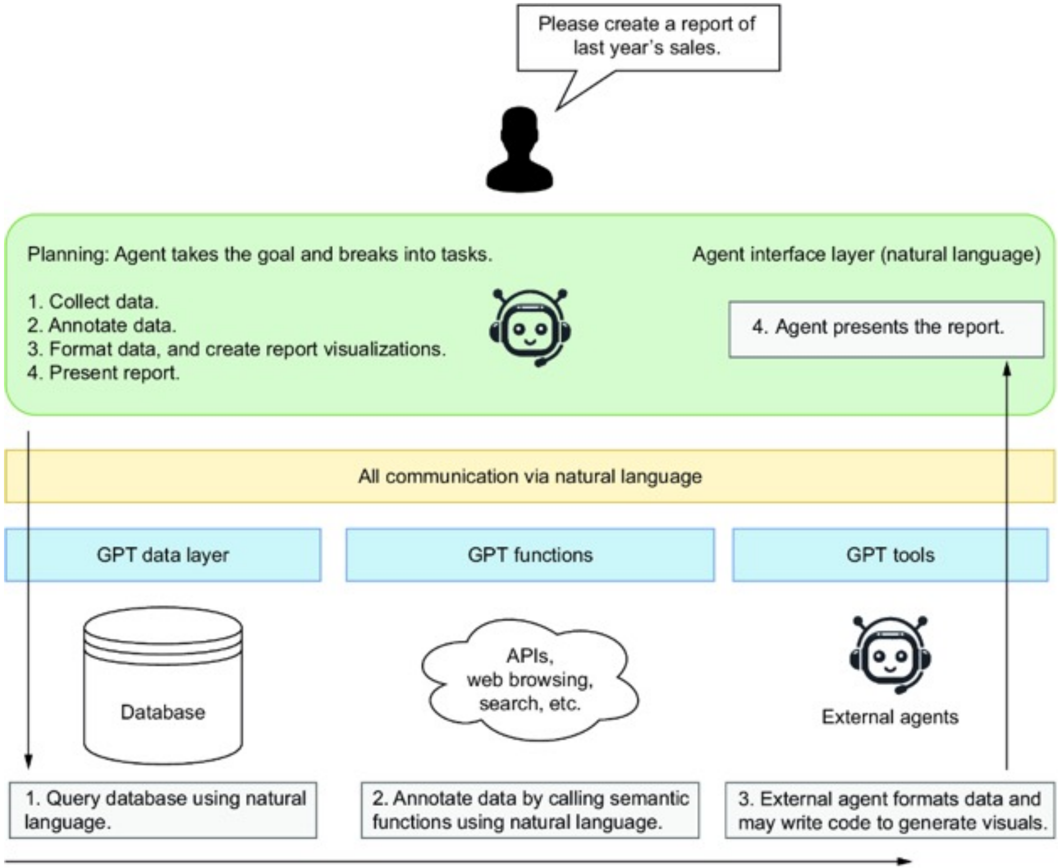
Source: https://medium.com/@varmalearn/a-casual-guide-to-the-top-agentic-ai-frameworks-in-2025-9299dd94e034

Process-Flow of the original AutoGPT agent system



Source: AI Agents in Action,
by Michael Lanham,
Feb 2025, Manning Publications

A Vision on how an Agent Interacts with other Software Systems



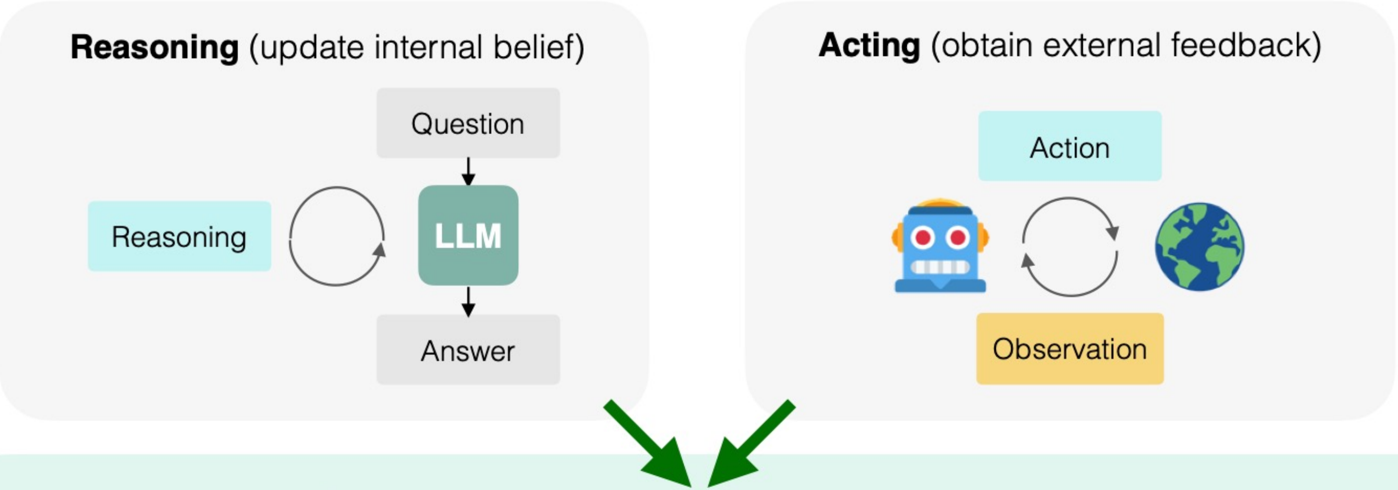
Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Implementing AI Agents with LangChain / LangGraph

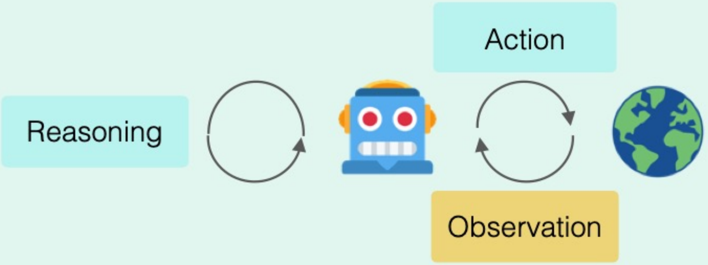
Build an Agent from Scratch

Use LangChain only, Not LangGraph !

Recap: ReAct = Reasoning and Acting



ReAct: a new paradigm of agents that **reason and act**



- **Synergy** of reasoning and acting
- **Simple** and intuitive to use
- **General** across domains

Example of ReAct w/ a Zero-Shot ReAct Prompt

You are an agent that answers questions by using two actions:

- search[query]: Google search the query. You can also use it to calculate math.
- finish[answer]: return the answer

Your generation should have the following format:

Thought: to analyze the observation and inform the action.

Action: your action

Question: If I have seven trillion dollars today, can I buy Apple, Nvidia, and Microsoft? If not, how much more money do I need?

Source: Shunyu Yao of OpenAI, "LLM Agents: Brief History and Overview", talk for Berkeley's MOOC CS294/194-196 on Large Language Model Agents, Fall 2024.

Example of ReAct

LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: "Nvidia's market cap stood at $1.832 trillion as of about 10 a.m. ET Wednesday, while Alphabet's was at $1.814 trillion, bringing Nvidia behind just Microsoft (MSFT) at $3.015 trillion and Apple (AAPL) at $2.836 trillion."

(Mar 2024)

Thought: Now that I have the market cap for all companies, let's add them together and see if $7 trillion is enough.
Action: search["1.832 trillion + 3.015 trillion + 2.836 trillion"]

Observation: 7683000000000

Thought: The amount of money needed in addition to the $7 trillion is $683 billion.
Action: finish["You need an additional $683 billion to be able to buy Apple, Nvidia, and Microsoft."]

✓

Source: Shunyu Yao of OpenAI, “LLM Agents: Brief History and Overview”, talk for Berkeley’s MOOC CS294/194-196 on Large Language Model Agents, Fall 2024.

Example of ReAct

LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: nothing found!

(My adversarial observation)

Thought: The search did not give any results. I will need to search for each company's market capitalization individually.
Action: search["current market capitalization of Apple"]

Observation: 'price': 180.75

(My adversarial observation)

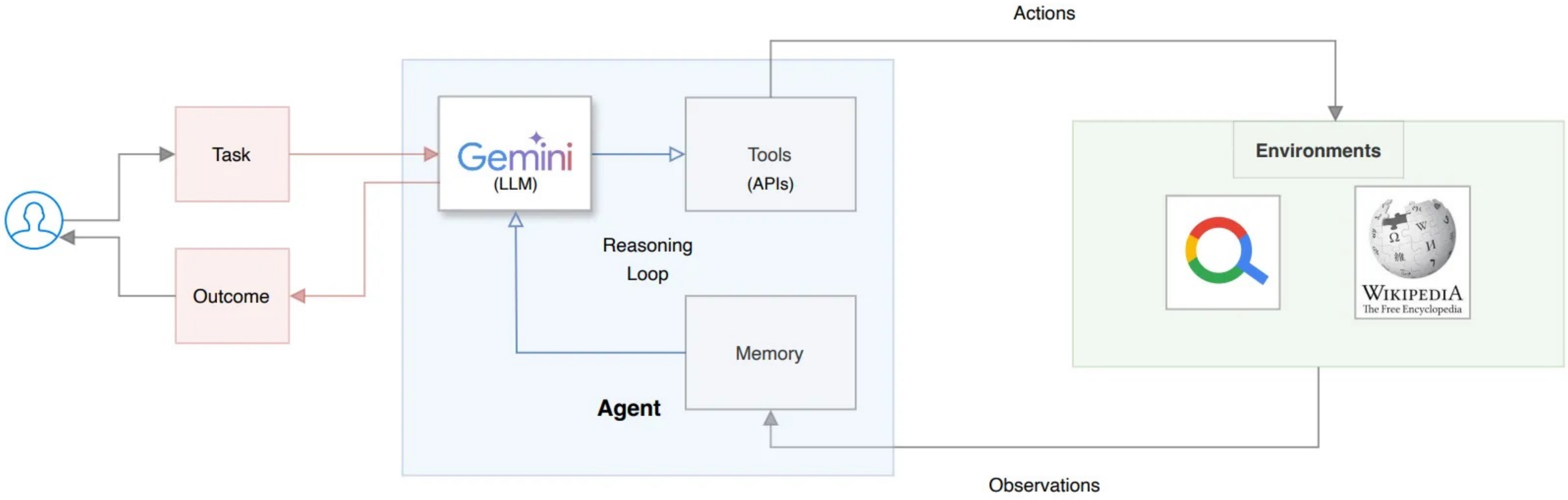
Thought: This is the current price of Apple's stock, not its market capitalization. The market cap is calculated by multiplying the share price by the number of outstanding shares. Thus, I need to find the total number of Apple's outstanding shares to calculate its market cap.
Action: search["total number of Apple's outstanding shares"]

✓ (Continues to solve the task)

Synergy: acting support reasoning, reasoning guides acting

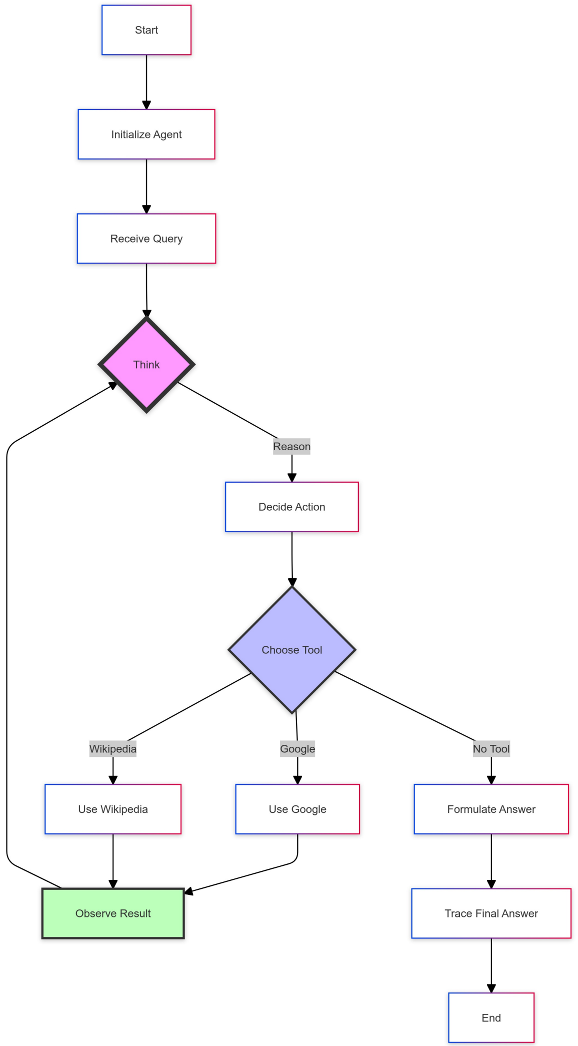
Source: Shunyu Yao of OpenAI, "LLM Agents: Brief History and Overview", talk for Berkeley's MOOC CS294/194-196 on Large Language Model Agents, Fall 2024.

High-level Architecture of a ReAct Agent using Gemini



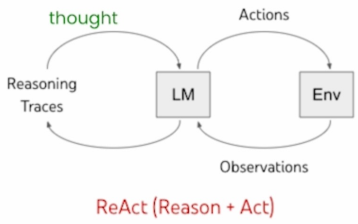
Source: https://medium.com/google-cloud/building-react-agents-from-scratch-a-hands-on-guide-using-gemini-ffe4621d90ae
https://github.com/arunpshankar/react-from-scratch

Process-Flow of a ReAct Agent



Source:
https://medium.com/google-cloud/building-react-agents-from-scratch-a-hands-on-guide-using-gemini-ffe4621d90ae
https://github.com/arunpshankar/react-from-scratch

Building a ReAct AI agent using LangChain Prompt Templates/ Library



Published as a conference paper at ICLR 2023

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

```
prompt = """
You run in a loop of Thought, Action, PAUSE, Observation.
At the end of the loop you output an Answer
Use Thought to describe your thoughts about the question you have been a
Use Action to run one of the actions available to you - then return PAUS
Observation will be the result of running those actions.

Your available actions are:

calculate:
e.g. calculate: 4 * 7 / 3
Runs a calculation and returns the number - uses Python so be sure to us

average_dog_weight:
e.g. average_dog_weight: Collie
returns average weight of a dog when given the breed

Example session:

Question: How much does a Bulldog weigh?
Thought: I should look the dogs weight using average_dog_weight
Action: average_dog_weight: Bulldog
PAUSE

You will be called again with this:

Observation: A Bulldog weights 51 lbs

You then output:

Answer: A bulldog weights 51 lbs
""".strip()
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct AI agent using LangChain Prompt Templates/ Library

```
# based on https://til.simonwillison.net/llms/python-react-pattern
```

```
import openai
import re
import httpx
import os
from dotenv import load_dotenv, find_dotenv

_ = load_dotenv(find_dotenv())
from openai import OpenAI
```

```
client = OpenAI()
```

```
chat_completion = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": "Hello world"}]
)
```

```
chat_completion.choices[0].message.content
```

```
'Hello! How can I assist you today?'
```

```
class Agent:
    def __init__(self, system=""):
        self.system = system
        self.messages = []
        if self.system:
            self.messages.append({"role": "system", "content": system})

    def __call__(self, message):
        self.messages.append({"role": "user", "content": message})
        result = self.execute()
        self.messages.append({"role": "assistant", "content": result})
        return result
```

```
class Agent:
    def __init__(self, system=""):
        self.system = system
        self.messages = []
        if self.system:
            self.messages.append({"role": "system", "content": system})

    def __call__(self, message):
        self.messages.append({"role": "user", "content": message})
        result = self.execute()
        self.messages.append({"role": "assistant", "content": result})
        return result

    def execute(self):
        completion = client.chat.completions.create(
                        model="gpt-4-0125-preview",
                        temperature=0,
                        messages=self.messages)
        return completion.choices[0].message.content
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct AI agent using LangChain Prompt Templates/ Library

```
prompt = """
You run in a loop of Thought, Action, PAUSE, Observation.
At the end of the loop you output an Answer
Use Thought to describe your thoughts about the question you have been a
Use Action to run one of the actions available to you - then return PAUS
Observation will be the result of running those actions.

Your available actions are:

calculate:
e.g. calculate: 4 * 7 / 3
Runs a calculation and returns the number - uses Python so be sure to us

average_dog_weight:
e.g. average_dog_weight: Collie
returns average weight of a dog when given the breed

Example session:

Question: How much does a Bulldog weigh?
Thought: I should look the dogs weight using average_dog_weight
Action: average_dog_weight: Bulldog
PAUSE

You will be called again with this:

Observation: A Bulldog weights 51 lbs

You then output:

Answer: A bulldog weights 51 lbs
""".strip()
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct AI agent using LangChain Prompt Templates/ Library

```python
def calculate(what):
    return eval(what)

def average_dog_weight(name):
    if name in "Scottish Terrier":
        return("Scottish Terriers average 20 lbs")
    elif name in "Border Collie":
        return("a Border Collies average weight is 37 lbs")
    elif name in "Toy Poodle":
        return("a toy poodles average weight is 7 lbs")
    else:
        return("An average dog weights 50 lbs")

known_actions = {
    "calculate": calculate,
    "average_dog_weight": average_dog_weight
}
```

```python
abot = Agent(prompt)
```

```python
result = abot("How much does a toy poodle weigh?")
print(result)
```

```
Thought: I should look up the dog's weight using average_dog_weight for
a Toy Poodle.
Action: average_dog_weight: Toy Poodle
PAUSE
```

```python
result = average_dog_weight("Toy Poodle")
```

```python
result
```

```
'a toy poodles average weight is 7 lbs'
```

```python
result
```

```
'a toy poodles average weight is 7 lbs'
```

```python
next_prompt = "Observation: {}".format(result)
```

```python
abot(next_prompt)
```

```
'Answer: A Toy Poodle weighs 7 lbs.'
```

```python
abot.messages
```

```
[{'role': 'system',
  'content': 'You run in a loop of Thought, Action, PAUSE, Observatio
n.\nAt the end of the loop you output an Answer\nUse Thought to describ
e your thoughts about the question you have been asked.\nUse Action to
run one of the actions available to you - then return PAUSE.\nObservati
on will be the result of running those actions.\n\nYour available actio
ns are:\n\ncalculate:\ne.g. calculate: 4 * 7 / 3\nRuns a calculation an
d returns the number - uses Python so be sure to use floating point syn
tax if necessary\n\naverage_dog_weight:\ne.g. average_dog_weight: Colli
e\nreturns average weight of a dog when given the breed\n\nExample sess
ion:\n\nQuestion: How much does a Bulldog weigh?\nThought: I should loo
k the dogs weight using average_dog_weight\nAction: average_dog_weight:
Bulldog\nPAUSE\n\nYou will be called again with this:\n\nObservation: A
Bulldog weights 51 lbs\n\nYou then output:\n\nAnswer: A bulldog weights
51 lbs'},
 {'role': 'user', 'content': 'How much does a toy poodle weigh?'},
 {'role': 'assistant',
  'content': "Thought: I should look up the dog's weight using average_
dog_weight for a Toy Poodle.\nAction: average_dog_weight: Toy Poodle\nP
AUSE"},
 {'role': 'user',
  'content': 'Observation: a toy poodles average weight is 7 lbs'},
 {'role': 'assistant', 'content': 'Answer: A Toy Poodle weighs 7 lb
s.'}]
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct AI agent using LangChain Prompt Templates/ Library

```
abot = Agent(prompt)
```

```
question = """I have 2 dogs, a border collie and a scottish terrier. \
What is their combined weight"""
abot(question)
```

```
"Thought: To find the combined weight of a Border Collie and a Scottish
Terrier, I need to first find the average weight of each breed and then
add those weights together. I'll start by finding the average weight of
a Border Collie.\n\nAction: average_dog_weight: Border Collie\nPAUSE"
```

```
next_prompt = "Observation: {}".format(average_dog_weight("Border Collie
print(next_prompt)
```

```
Observation: a Border Collies average weight is 37 lbs
```

```
abot(next_prompt)
```

```
"Thought: Now that I know a Border Collie's average weight is 37 lbs, I
need to find the average weight of a Scottish Terrier to calculate the
combined weight.\n\nAction: average_dog_weight: Scottish Terrier\nPAUS
E"
```

```
next_prompt = "Observation: {}".format(average_dog_weight("Scottish Terr
print(next_prompt)
```

```
Observation: Scottish Terriers average 20 lbs
```

```
abot(next_prompt)
```

```
'Thought: With the average weight of a Border Collie being 37 lbs and a
Scottish Terrier being 20 lbs, I can now calculate their combined weigh
t.\n\nAction: calculate: 37 + 20\nPAUSE'
```

```
Observation: 57
```

```
abot(next_prompt)
```

```
'Answer: The combined weight of a Border Collie and a Scottish Terrier
is 57 lbs.'
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct AI agent using LangChain Prompt Templates/ Library

```python
action_re = re.compile('^Action: (\w+): (.*)$')
```

```python
def query(question, max_turns=5):
    i = 0
    bot = Agent(prompt)
    next_prompt = question
    while i < max_turns:
        i += 1
        result = bot(next_prompt)
        print(result)
        actions = [
            action_re.match(a)
            for a in result.split('\n')
            if action_re.match(a)
        ]
        if actions:
            # There is an action to run
            action, action_input = actions[0].groups()
            if action not in known_actions:
                raise Exception("Unknown action: {}: {}".format(action,
            print(" -- running {} {}".format(action, action_input))
            observation = known_actions[action](action_input)
            print("Observation:", observation)
            next_prompt = "Observation: {}".format(observation)
        else:
            return
```

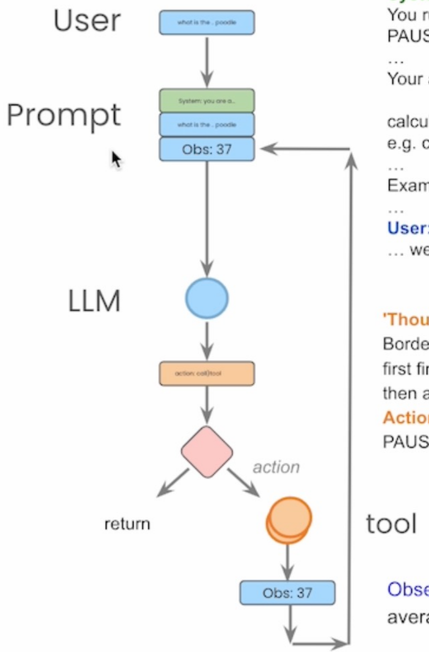
```python
question = """I have 2 dogs, a border collie and a scottish terrier. \
What is their combined weight"""
query(question)
```

```python
question = """I have 2 dogs, a border collie and a scottish terrier. \
What is their combined weight"""
query(question)
```

```
Thought: To find the combined weight of a Border Collie and a Scottish
Terrier, I need to first find the average weight of each breed and then
add those weights together. I will start by finding the average weight
of a Border Collie.

Action: average_dog_weight: Border Collie
PAUSE
 -- running average_dog_weight Border Collie
Observation: a Border Collies average weight is 37 lbs
Thought: Now that I know the average weight of a Border Collie is 37 lb
s, I need to find the average weight of a Scottish Terrier to calculate
the combined weight of both dogs.

Action: average_dog_weight: Scottish Terrier
PAUSE
 -- running average_dog_weight Scottish Terrier
Observation: Scottish Terriers average 20 lbs
Thought: With the average weight of a Border Collie being 37 lbs and a
Scottish Terrier being 20 lbs, I can now calculate their combined weigh
t by adding these two numbers together.

Action: calculate: 37 + 20
PAUSE
 -- running calculate 37 + 20
Observation: 57
Answer: The combined weight of a Border Collie and a Scottish Terrier i
s 57 lbs.
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Building a ReAct-based AI Agent w/ vanillia LangChain



"""
System:
You run in a loop of Thought, Action, PAUSE, Observation.
…
Your available actions are:

calculate:
e.g. calculate: 4 * 7 / 3
…
Example session:
…
User:
… weight of collie…

'**Thought:** To find the combined weight of a Border Collie and a Scottish Terrier, I need to first find the average weight of each breed and then add those weights together.
Action: average_dog_weight: Border Collie\n PAUSE'

Observation: a Border Collies average weight is 37 lbs

```
def query(question, max_turns=5):
    i = 0
    bot = Agent(prompt)
    next_prompt = question
    while i < max_turns:
        i += 1
        result = bot(next_prompt)
        print(result)
        actions = [action_re.match(a) for a in result.split('\n') if action_re.match(a)]
        if actions:
            # There is an action to run
            action, action_input = actions[0].groups()
            if action not in known_actions:
                raise Exception("Unknown action: {}: {}".format(action, action_input))
            print(" -- running {} {}".format(action, action_input))
            observation = known_actions[action](action_input)
            print("Observation:", observation)
            next_prompt = "Observation: {}".format(observation)
        else:
            return
```

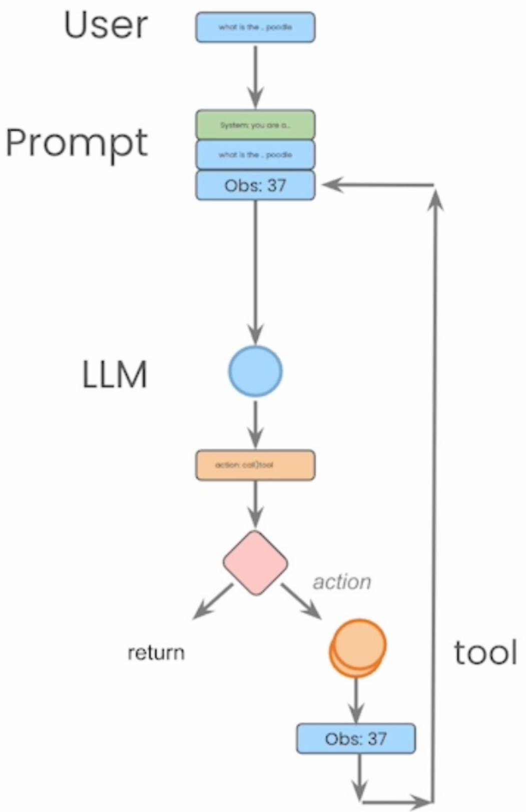
```
def calculate(what):
    return eval(what)

def average_dog_weight(name):
    if name in "Scottish Terrier":
        return("Scottish Terriers average 20 lbs")
    elif name in "Border Collie":
        return("a Border Collies average weight is 37
lbs")
    elif name in "Toy Poodle":
        return("a toy poodles average weight is 7 lbs")
    else:
        return("An average dog weights 50 lbs")
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Recap: LangChain Prompt Templates



Prompt templates allow reusable prompts

```
from langchain.prompts import PromptTemplate
prompt_template = PromptTemplate.from_template(
  "Tell me a {adjective} joke about {content}."
```

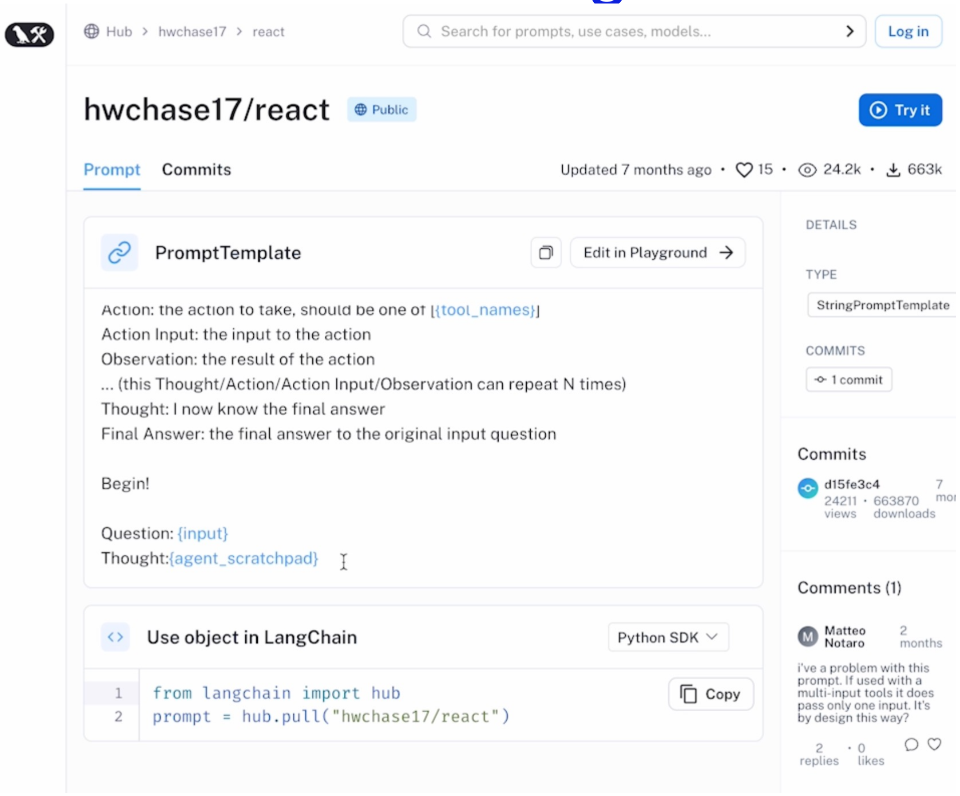
There are also prompts for agents available in the hub:

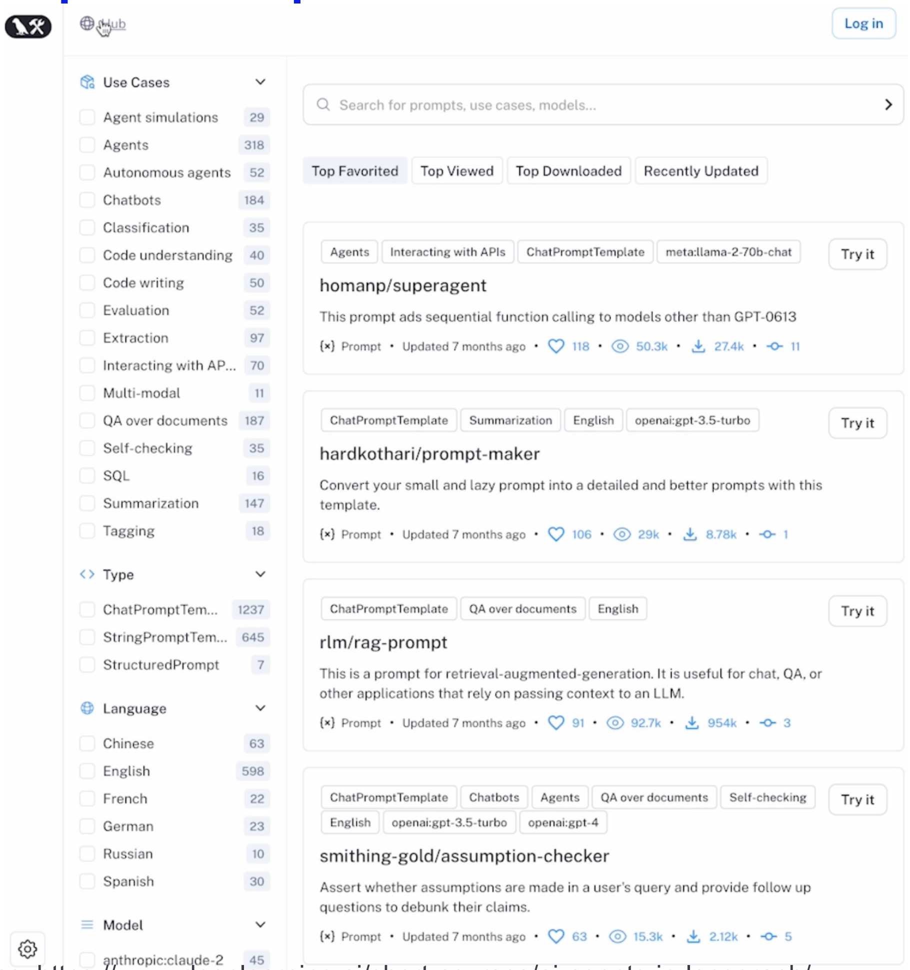
```
prompt = hub.pull("hwchase17/react")
```

https://smith.langchain.com/hub/hwchase17/react

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

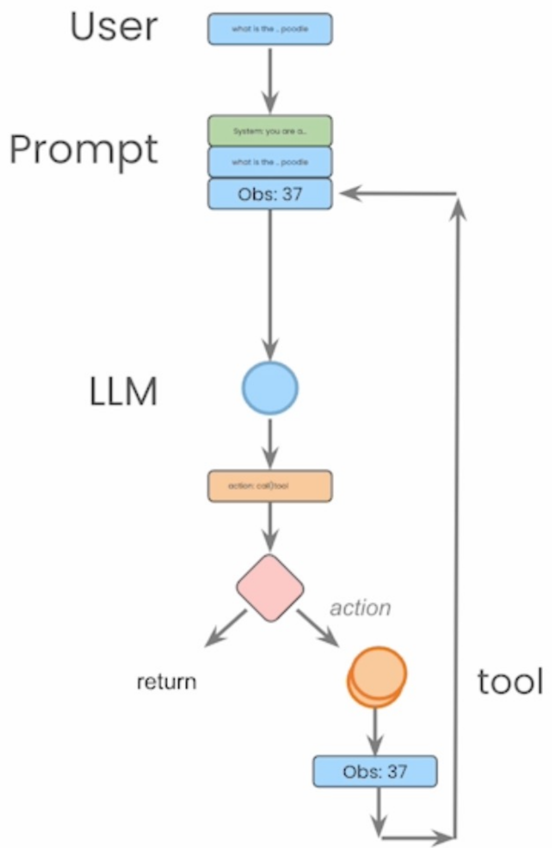
LangChain Prompt Templates





Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

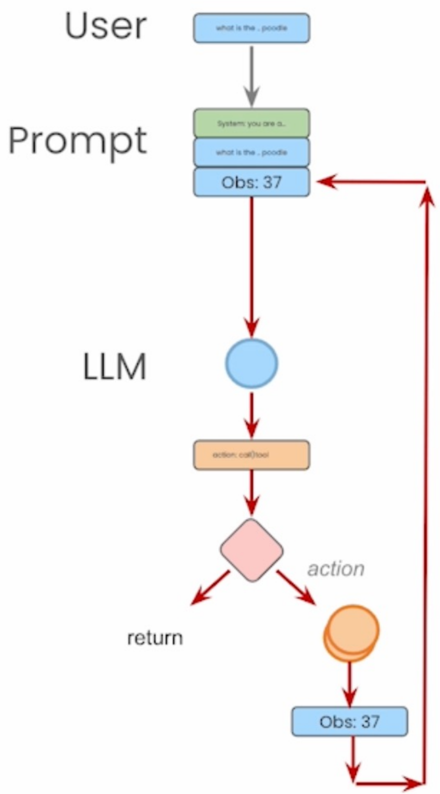
LangChain Tools




```python
# get a tool from the library
from langchain_community.tools.tavily_search import \
TavilySearchResults
tool = TavilySearchResults(max_results=2)


self.tools = {t.name: t for t in tools}
self.model = model.bind_tools([tool])
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

What’s New in LangGraph for building AI Agents ?

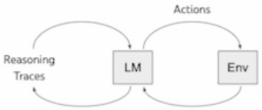


- Cyclic Graphs
- Persistence
- Human-in-the-loop

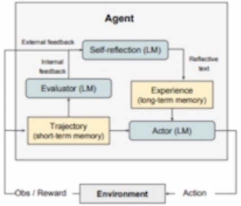
Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

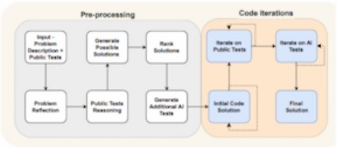
What’s New in LangGraph for building AI Agents ?

Graphs



ReAct (Reason + Act)



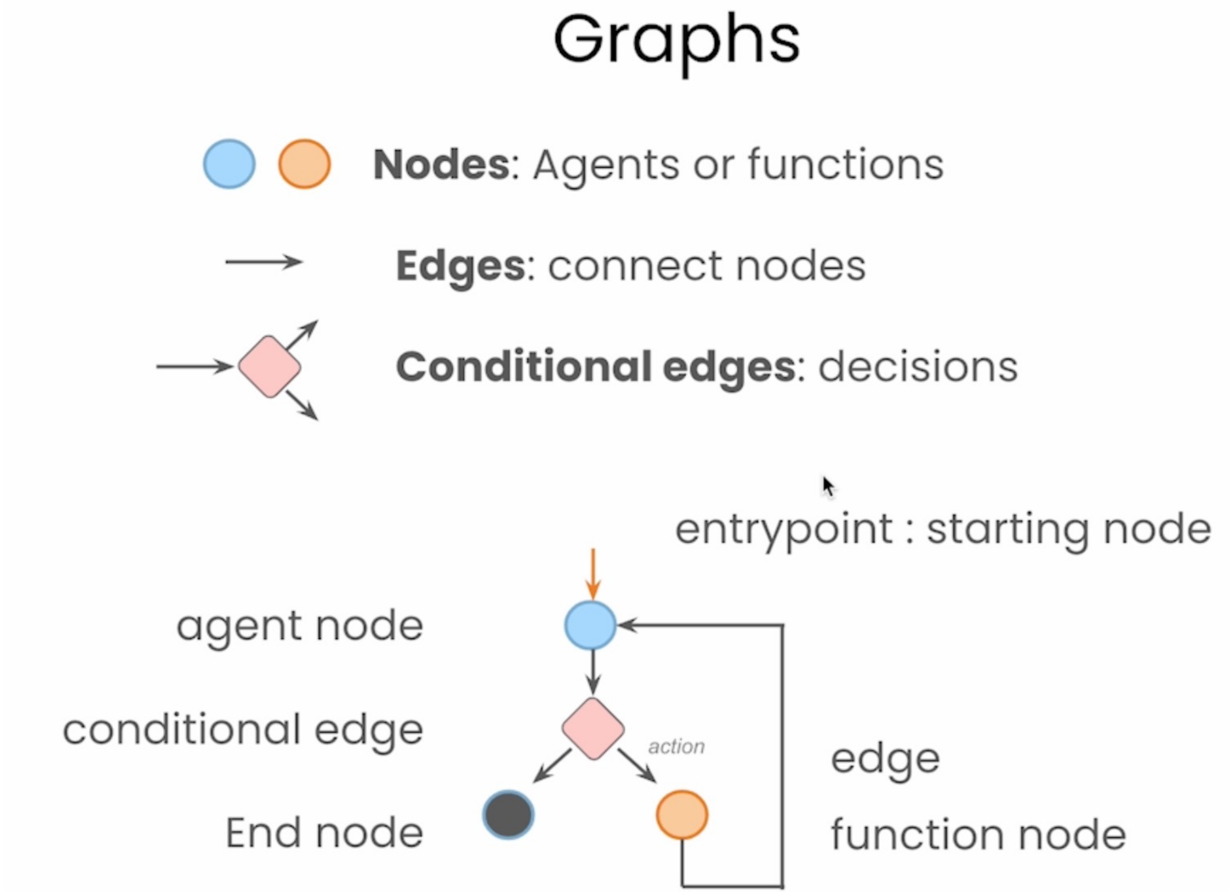


(a) The proposed AlphaCodium flow.

- LangGraph is an extension of LangChain that supports graphs.
- Single and Multi-agent flows are described and represented as graphs.
- Allows for extremely controlled “flows”
- Built-in persistence allows for human-in-the-loop workflows

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

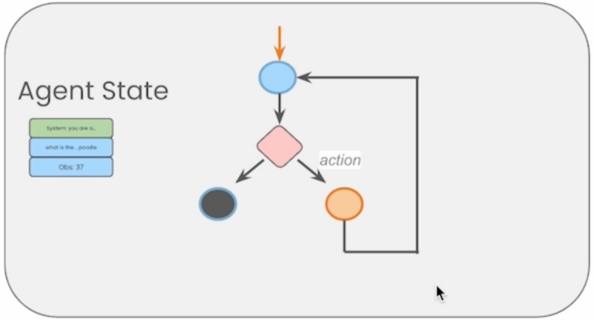
What’s New in LangGraph for building AI Agents ?



Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

What's New in LangGraph for building AI Agents ?

Data/State



- Agent State is accessible to all parts of the graph
- It is local to the graph
- Can be stored in a persistence layer

Simple

```
class AgentState(TypedDict):
    messages: Annotated[Sequence[BaseMessage], operator.add]
```

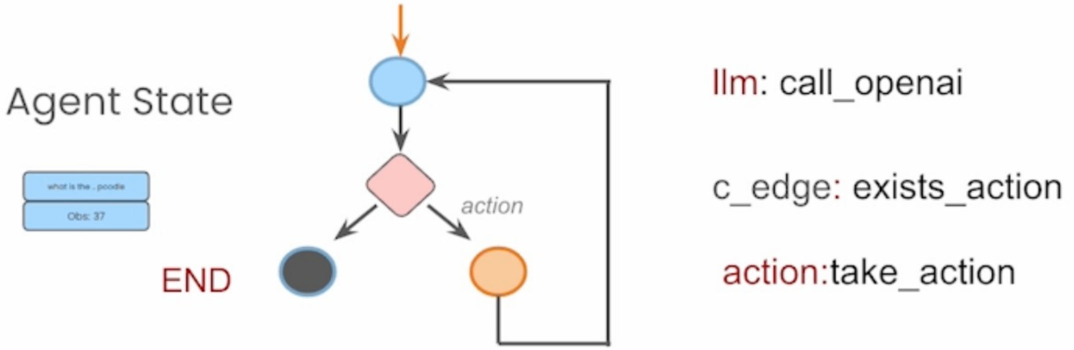
Complex

```
class AgentState(TypedDict):
    input: str
    chat_history: list[BaseMessage]
    agent_outcome: Union[AgentAction, AgentFinish, None]
    intermediate_steps: Annotated[list[tuple[AgentAction, str]], operator.add]
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

What's New in LangGraph for building AI Agents ?

CODE



State

```
class AgentState(TypedDict):
    messages: Annotated[list[AnyMessage], operator.add]
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
from dotenv import load_dotenv, find_dotenv

_ = load_dotenv(find_dotenv())
```

```python
from langgraph.graph import StateGraph, END
from typing import TypedDict, Annotated
import operator
from langchain_core.messages import AnyMessage, SystemMessage, HumanMess
from langchain_openai import ChatOpenAI
from langchain_community.tools.tavily_search import TavilySearchResults
```

```python
tool = TavilySearchResults(max_results=2)
print(type(tool))
print(tool.name)
```

```
<class 'langchain_community.tools.tavily_search.tool.TavilySearchResult
s'>
tavily_search_results_json
```

```python
class AgentState(TypedDict):
    messages: Annotated[list[AnyMessage], operator.add]
```

```python
class Agent:

    def __init__(self, model, tools, system=""):
        self.system = system
        graph = StateGraph(AgentState)
        graph.add_node("llm", ...)
        graph.add_node("action", ...)
        graph.add_conditional_edges(
            "llm",
            ...,
            {True: "action", False: END}
        )
        graph.add_edge("action", "llm")
        graph.set_entry_point("llm)
```

```python
class Agent:

    def __init__(self, model, tools, system=""):
        self.system = system
        graph = StateGraph(AgentState)
        graph.add_node("llm", self.call_openai)
        graph.add_node("action", ...)
        graph.add_conditional_edges(
            "llm",
            ...,
            {True: "action", False: END}
        )
        graph.add_edge("action", "llm")
        graph.set_entry_point("llm")
        self.graph = graph.compile()
        self.tools = {t.name: t for t in tools}
        self.model = model.bind_tools(tools)

    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
        message = self.model.invoke(messages)
        return {'messages': [message]}

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
            results.append(ToolMessage(tool_call_id=t['id'], name=t['nam
        print("Back to the model!")
        return {'messages': results}
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
class Agent:

    def __init__(self, model, tools, system=""):
        self.system = system
        graph = StateGraph(AgentState)
        graph.add_node("llm", self.call_openai)
        graph.add_node("action", self.take_action)
        graph.add_conditional_edges(
            "llm",
            self.exists_action,
            {True: "action", False: END}
        )
        graph.add_edge("action", "llm")
        graph.set_entry_point("llm")
        self.graph = graph.compile()
        self.tools = {t.name: t for t in tools}
        self.model = model.bind_tools(tools)

    def exists_action(self, state: AgentState):
        result = state['messages'][-1]
        return len(result.tool_calls) > 0

    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
        message = self.model.invoke(messages)
        return {'messages': [message]}

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
```

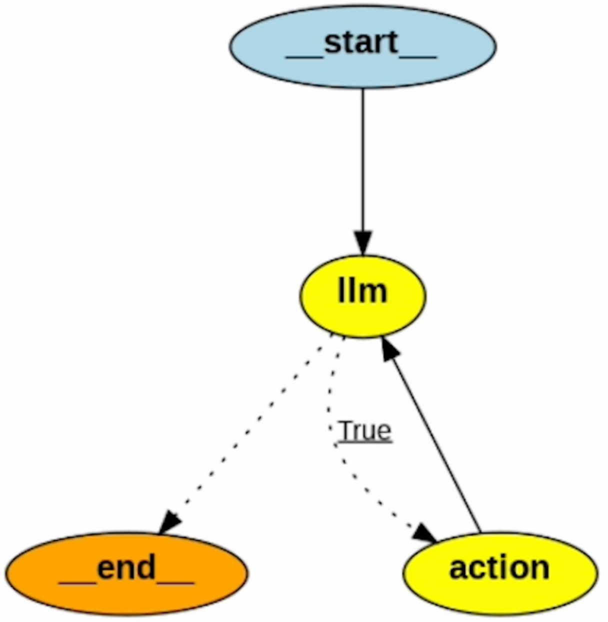
```python
        return len(result.tool_calls) > 0

    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
        message = self.model.invoke(messages)
        return {'messages': [message]}

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
            results.append(ToolMessage(tool_call_id=t['id'], name=t['nam
        print("Back to the model!")
        return {'messages': results}
```

```python
prompt = """You are a smart research assistant. Use the search engine to
You are allowed to make multiple calls (either together or in sequence).
Only look up information when you are sure of what you want. \
If you need to look up some information before asking a follow up questi
"""

model = ChatOpenAI(model="gpt-4-turbo")
abot = Agent(model, [tool], system=prompt)
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
from IPython.display import Image

Image(abot.graph.get_graph().draw_png())
```




```python
messages = [HumanMessage(content="What is the weather in sf?")]
result = abot.graph.invoke({"messages": messages})
```

```
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in San Francisco'}, 'id': 'call_3aeXDjFZbI2DQodx9490uQdk'}
Back to the model!
```

```python
result
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```
result
```


```
{'messages': [HumanMessage(content='What is the weather in sf?'),
  AIMessage(content='', additional_kwargs={'tool_calls': [{'id': 'call_3aeXDjFZbI2DQodx949OuQdk', 'function': {'arguments': '{"query":"current weather in San Francisco"}', 'name': 'tavily_search_results_json'}, 'type': 'function'}]}, response_metadata={'token_usage': {'completion_tokens': 22, 'prompt_tokens': 153, 'total_tokens': 175}, 'model_name': 'gpt-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_reason': 'tool_calls', 'logprobs': None}, id='run-d20dbbbb-f650-428d-a0b0-2f676bd34da8-0', tool_calls=[{'name': 'tavily_search_results_json', 'args': {'query': 'current weather in San Francisco'}, 'id': 'call_3aeXDjFZbI2DQodx949OuQdk'}]),
  ToolMessage(content='[{\'url\': \'https://www.weatherapi.com/\', \'content\': "{\'location\': {\'name\': \'San Francisco\', \'region\': \'California\', \'country\': \'United States of America\', \'lat\': 37.78, \'lon\': -122.42, \'tz_id\': \'America/Los_Angeles\', \'localtime_epoch\': 1712954805, \'localtime\': \'2024-04-12 13:46\'}, \'current\': {\'last_updated_epoch\': 1712954700, \'last_updated\': \'2024-04-12 13:45\', \'temp_c\': 15.6, \'temp_f\': 60.1, \'is_day\': 1, \'condition\': {\'text\': \'Partly cloudy\', \'icon\': \'//cdn.weatherapi.com/weather/64x64/day/116.png\', \'code\': 1003}, \'wind_mph\': 15.0, \'wind_kph\': 24.1, \'wind_degree\': 270, \'wind_dir\': \'W\', \'pressure_mb\': 1009.0, \'pressure_in\': 29.8, \'precip_mm\': 0.0, \'precip_in\': 0.0, \'humidity\': 60, \'cloud\': 50, \'feelslike_c\': 15.6, \'feelslike_f\': 60.1, \'vis_km\': 16.0, \'vis_miles\': 9.0, \'uv\': 4.0, \'gust_mph\': 19.5, \'gust_kph\': 31.3}}"}, {\'url\': \'https://www.wunderground.com/hourly/us/ca/san-francisco/94129/date/2024-04-12\', \'content\': \'San Francisco Weather Forecasts. Weather Underground provides local & long-range weather forecasts, weatherreports, maps & tropical weather conditions for the San Francisco area. ... Friday 04/12 ...\'}]', name='tavily_search_results_json', tool_call_id='call_3aeXDjFZbI2DQodx949OuQdk'),
  AIMessage(content='The current weather in San Francisco is partly cloudy with a temperature of 60.1°F (15.6°C). The wind is coming from the west at 15.0 mph (24.1 kph), and the atmospheric pressure is 1009.0 mb. The humidity level is at 60%, and visibility is approximately 9.0 miles (16.0 km).', response_metadata={'token_usage': {'completion_tokens': 80, 'prompt_tokens': 607, 'total_tokens': 687}, 'model_name': 'gpt-4-tu
```


```
result['messages'][-1].content
```

```
'The current weather in San Francisco is partly cloudy with a temperature of 60.1°F (15.6°C). The wind is coming from the west at 15.0 mph (24.1 kph), and the atmospheric pressure is 1009.0 mb. The humidity level is at 60%, and visibility is approximately 9.0 miles (16.0 km).'
```

```
messages = [HumanMessage(content="What is the weather in SF and LA?")]
result = abot.graph.invoke({"messages": messages})
```

```
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'current weather in San Francisco'}, 'id': 'call_3aeXDjFZbI2DQodx949OuQdk'}
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'current weather in Los Angeles'}, 'id': 'call_w7IBLdVHXmdUIpHgybMaE5Vw'}
Back to the model!
```

```
result['messages'][-1].content
```

```
'Here is the current weather information for San Francisco and Los Angeles:\n\n**San Francisco:**\n- Temperature: 15.6°C (60.1°F)\n- Condition: Partly cloudy\n- Wind: 15.0 mph from the west\n- Humidity: 60%\n- Visibility: 16 km (9 miles)\n\n**Los Angeles:**\n- Temperature: 16.1°C (61.0°F)\n- Condition: Overcast\n- Wind: 4.3 mph from the southeast\n- Humidity: 67%\n- Visibility: 16 km (9 miles)'
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```
query = "Who won the super bowl in 2024? What is the GDP of that state?"
messages = [HumanMessage(content=query)]
result = abot.graph.invoke({"messages": messages})
```

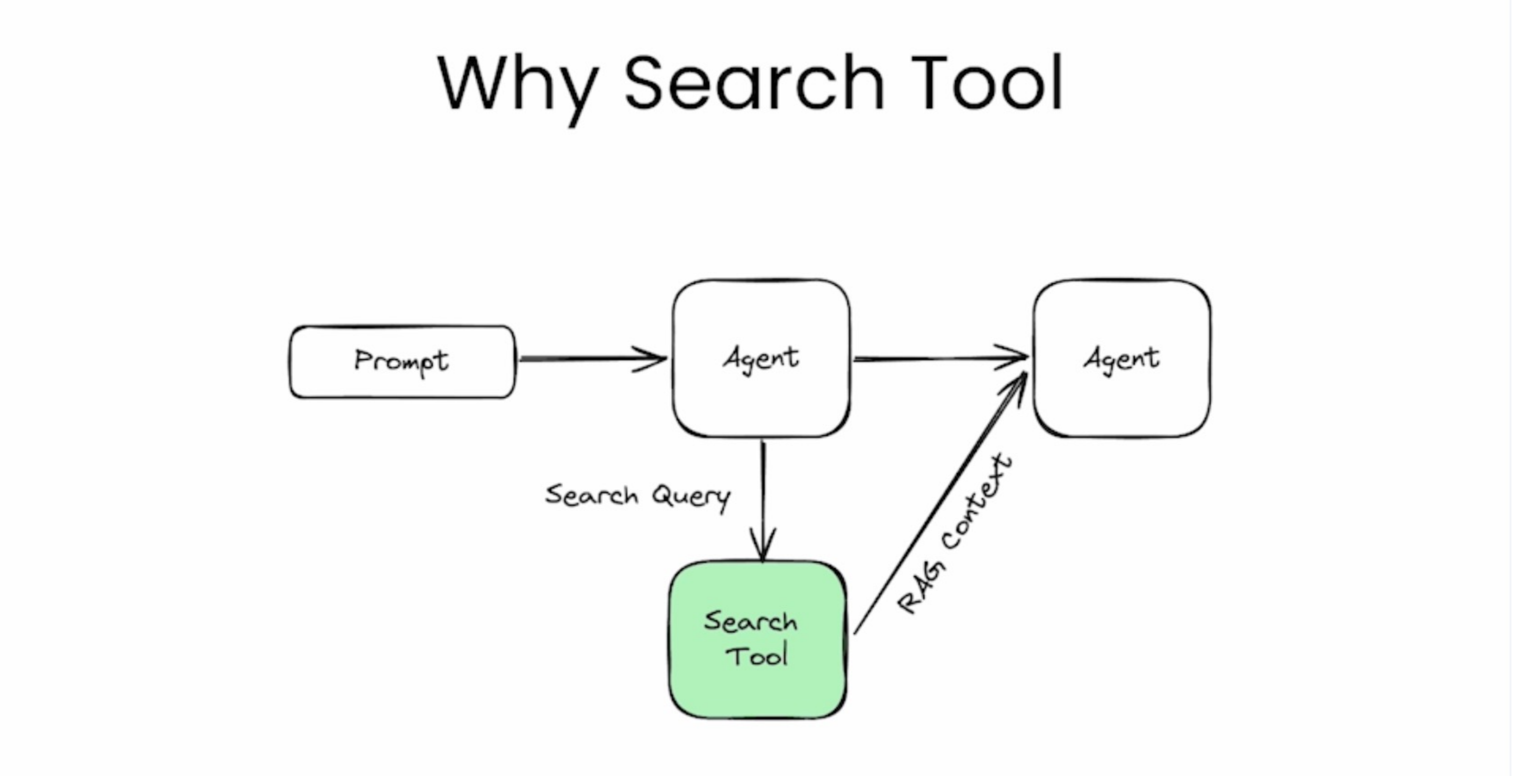
```
Calling: {'name': 'tavily_search_results_json', 'args': {'query': '2024
Super Bowl winner'}, 'id': 'call_8KigXEXM5GShFhcaLjJLMe7Z'}
Back to the model!
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'GDP
of Missouri 2023'}, 'id': 'call_q3ynNhiv6pOmae07KaN6tuxG'}
Back to the model!
```

```
result['messages'][-1].content
```

```
'The Kansas City Chiefs won the Super Bowl in 2024.\n\nThe GDP of Misso
uri, where the Kansas City Chiefs are based, was $423.6 billion in the
3rd quarter of 2023.'
```

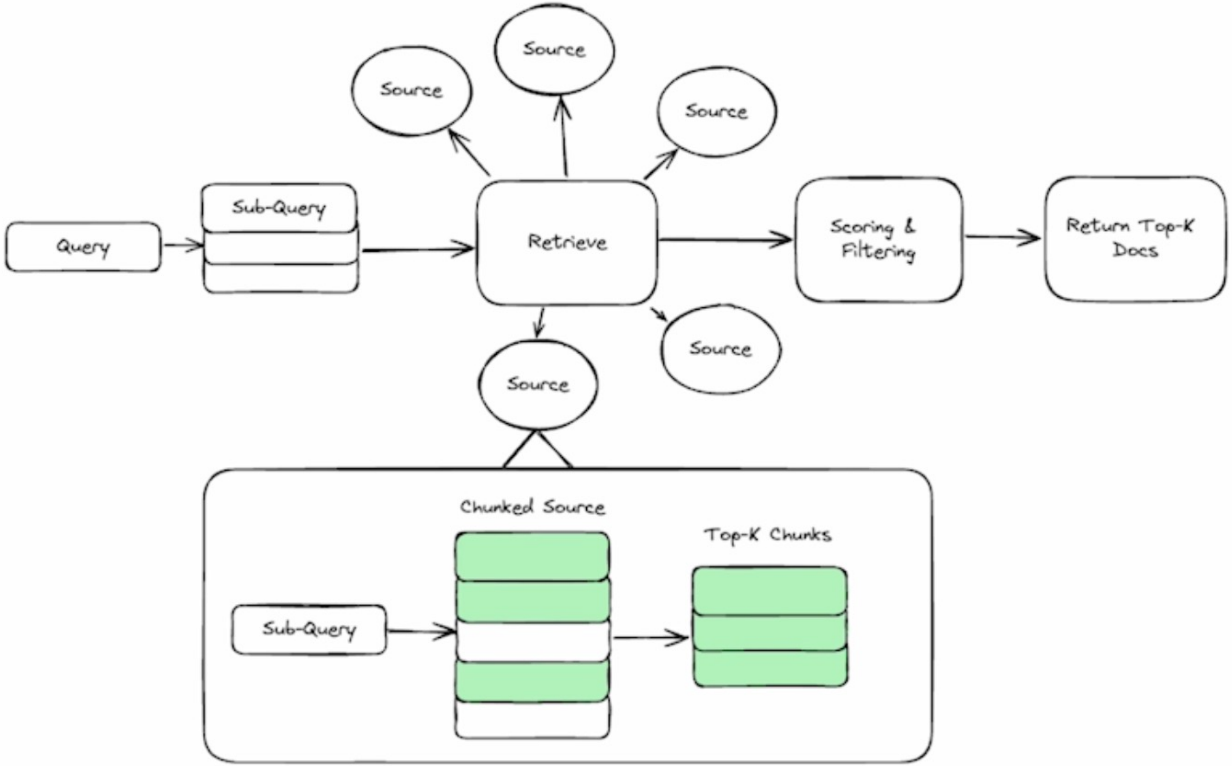
Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph



Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph



Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
from dotenv import load_dotenv
import os
from tavily import TavilyClient

# load environment variables from .env file
_ = load_dotenv()

# connect
client = TavilyClient(api_key=os.environ.get("TAVILY_API_KEY"))
```

```python
# run search
result = client.search("What is in Nvidia's new Blackwell GPU?",
                       include_answer=True)

# print the answer
result["answer"]
```

```
"The Nvidia Blackwell GPU features 208 billion transistors and connects
two chip parts with a super-fast link between them for a cohesive archi
tecture. It also includes an Advanced Transformer Engine, an AI engine
designed to double Blackwell's processing power for more efficient task
handling."
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
# choose location (try to change to your own city!)

city = "San Francisco"

query = f"""
    what is the current weather in {city}?
    Should I travel there today?
"""
```

```python
import requests
from bs4 import BeautifulSoup
from duckduckgo_search import DDGS
import re

ddg = DDGS()

def search(query, max_results=3):
    results = ddg.text(query, max_results=max_results)
    return [i["href"] for i in results]

for i in search(query):
    print(i)
```

```
https://www.accuweather.com/en/us/san-francisco/94103/current-weather/347629
https://www.accuweather.com/en/us/san-francisco/94103/weather-forecast/347629
https://www.accuweather.com/en/us/san-francisco/94103/air-travel-weather/347629
```

```python
def scrape_weather_info(url):
    """Scrape content from the given URL"""
    if not url:
        return "Weather information could not be found."

    # fetch data
    headers = {'User-Agent': 'Mozilla/5.0'}
    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        return "Failed to retrieve the webpage."

    # parse result
    soup = BeautifulSoup(response.text, 'html.parser')
    return soup


# use DuckDuckGo to find a websites and take the first result
url = search(query)[0]

# scrape first wesbsite
soup = scrape_weather_info(url)

print(f"Website: {url}\n\n")
print(soup)
```

```
Website: https://www.accuweather.com/en/us/san-francisco/94103/current-weather/347629



<!DOCTYPE html>

<html class="accuweather" lang="en-us">
<head>
<meta content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"/>
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
weather_data = []
for tag in soup.find_all(['h1', 'h2', 'h3', 'p']):
    text = tag.get_text(" ", strip=True)
    weather_data.append(text)

# combine all elements into a single string
weather_data = "\n".join(weather_data)

# remove all spaces from the combined text
weather_data = re.sub(r'\s+', ' ', weather_data)

print(f"Website: {url}\n\n")
print(weather_data)
```

```
Website: https://www.accuweather.com/en/us/san-francisco/94103/current-
weather/347629


San Francisco, CA San Francisco California Around the Globe Around the
Globe Hurricane Tracker Severe Weather Radar & Maps News & Features Ast
ronomy Business Climate Health Recreation Sports Travel Video Winter Ce
nter Current Weather 2:24 PM Day Max UV Index 7 High Wind WSW 13 mph Wi
nd Gusts 25 mph Probability of Precipitation 25% Probability of Thunder
storms 3% Precipitation 0.00 in Cloud Cover 69% Morning Afternoon Night
Wind S 14 mph Wind Gusts 17 mph Probability of Precipitation 91% Probab
ility of Thunderstorms 17% Precipitation 0.33 in Rain 0.33 in Hours of
Precipitation 5 Hours of Rain 5 Cloud Cover 93% Evening Overnight Sun &
Moon Temperature History Further Ahead Further Ahead Hourly Daily Month
ly Around the Globe Around the Globe Hurricane Tracker Severe Weather R
adar & Maps News Video Winter Center Top Stories Severe Weather Powerfu
l storms unleash severe flooding, tornadoes from Texas to Maine 1 hour
ago Weather Forecasts Chilly winds to follow storms, flooding rain in e
astern US 2 hours ago Severe Weather Severe storms to threaten weekend
plans from Midwest to Northeast 5 hours ago Winter Weather Mid-April st
orm to eye California with rain, mountain snow 2 hours ago Severe Weath
er Severe weather, tornado threat to build on Plains, Mississippi Valle
y 2 hours ago Featured Stories Weather News How wet weather can be dang
erous to your pets Climate March 2024 hottest on record, EU climate se
```

```python
import json
from pygments import highlight, lexers, formatters

# parse JSON
parsed_json = json.loads(data.replace("'", '"'))

# pretty print JSON with syntax highlighting
formatted_json = json.dumps(parsed_json, indent=4)
colorful_json = highlight(formatted_json,
                          lexers.JsonLexer(),
                          formatters.TerminalFormatter())

print(colorful_json)
```

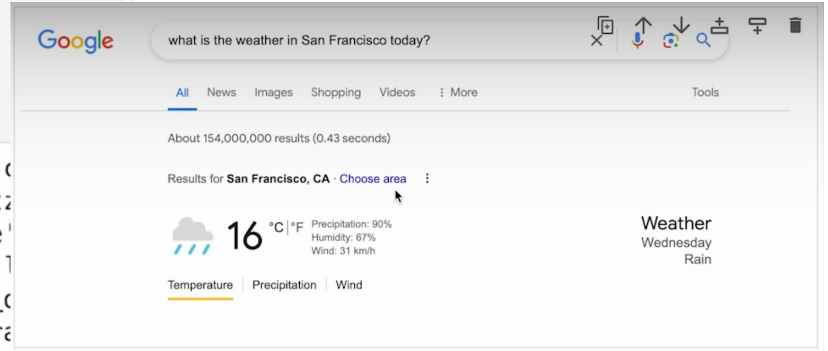

```json
{
    "location": {
        "name": "San Francisco",
        "region": "California",
        "country": "United States of America",
        "lat": 37.78,
        "lon": -122.42,
        "tz_id": "America/Los_Angeles",
        "localtime_epoch": 1712957187,
        "localtime": "2024-04-12 14:26"
    },
    "current": {
        "last_updated_epoch": 1712956500,
        "last_updated": "2024-04-12 14:15",
        "temp_c": 15.0,
        "temp_f": 59.0,
        "is_day": 1,
        "condition": {
            "text": "Partly cloudy",
            "icon": "//cdn.weatherapi.com/weather/64x64/day/116.png",
            "code": 1003
        },
```


Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
# run search
result = client.search(query, max_results=1)

# print first result
data = result["results"][0]["content"]

print(data)
```


```
{'location': {'name': 'San Francisco', 'region': 'California', 'c
y': 'United States of America', 'lat': 37.78, 'lon': -122.42, 'tz
'America/Los_Angeles', 'localtime_epoch': 1712957187, 'localtime'
4-04-12 14:26'}, 'current': {'last_updated_epoch': 1712956500, 'l
dated': '2024-04-12 14:15', 'temp_c': 15.0, 'temp_f': 59.0, 'is_d
1, 'condition': {'text': 'Partly cloudy', 'icon': '//cdn.weathera
m/weather/64x64/day/116.png', 'code': 1003}, 'wind_mph': 16.1, 'wind_kp
h': 25.9, 'wind_degree': 210, 'wind_dir': 'SSW', 'pressure_mb': 1009.0,
'pressure_in': 29.8, 'precip_mm': 0.0, 'precip_in': 0.0, 'humidity': 6
0, 'cloud': 75, 'feelslike_c': 13.6, 'feelslike_f': 56.5, 'vis_km': 16.
0, 'vis_miles': 9.0, 'uv': 4.0, 'gust_mph': 20.6, 'gust_kph': 33.1}}
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
from dotenv import load_dotenv, find_dotenv

_ = load_dotenv(find_dotenv())
```

```python
from langgraph.graph import StateGraph, END
from typing import TypedDict, Annotated
import operator
from langchain_core.messages import AnyMessage, SystemMessage, HumanMess
from langchain_openai import ChatOpenAI
from langchain_community.tools.tavily_search import TavilySearchResults
```

```python
tool = TavilySearchResults(max_results=2)
```

```python
class AgentState(TypedDict):
    messages: Annotated[list[AnyMessage], operator.add]
```

```python
prompt = """You are a smart research assistant. Use the search engine to
You are allowed to make multiple calls (either together or in sequence).
Only look up information when you are sure of what you want. \
If you need to look up some information before asking a follow up questi
"""
model = ChatOpenAI(model="gpt-4-turbo")
abot = Agent(model, [tool], system=prompt, checkpointer=memory)
```

```python
from langgraph.checkpoint.sqlite import SqliteSaver

memory = SqliteSaver.from_conn_string(":memory:")
```

```python
class Agent:
    def __init__(self, model, tools, checkpointer, system=""):
        self.system = system
        graph = StateGraph(AgentState)
        graph.add_node("llm", self.call_openai)
        graph.add_node("action", self.take_action)
        graph.add_conditional_edges("llm", self.exists_action, {True: "a
        graph.add_edge("action", "llm")
        graph.set_entry_point("llm")
        self.graph = graph.compile(checkpointer=checkpointer)
        self.tools = {t.name: t for t in tools}
        self.model = model.bind_tools(tools)

    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
        message = self.model.invoke(messages)
        return {'messages': [message]}

    def exists_action(self, state: AgentState):
        result = state['messages'][-1]
        return len(result.tool_calls) > 0

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
            results.append(ToolMessage(tool_call_id=t['id'], name=t['n
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
messages = [HumanMessage(content="What is the weather in sf?")]
```

```python
thread = {"configurable": {"thread_id": "1"}}
```

```python
for event in abot.graph.stream({"messages": messages}, thread):
    for v in event.values():
        print(v['messages'])
```


```
[AIMessage(content='', additional_kwargs={'tool_calls': [{'id': 'call_v
0EHAA8cgE33PI4vQUaLUyMB', 'function': {'arguments': '{"query":"current
weather in San Francisco"}', 'name': 'tavily_search_results_json'}, 'ty
pe': 'function'}]}, response_metadata={'token_usage': {'completion_toke
ns': 22, 'prompt_tokens': 153, 'total_tokens': 175}, 'model_name': 'gpt
-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_reason': 'too
l_calls', 'logprobs': None}, id='run-07978a9a-6798-480b-a8f4-b99c439e6a
ef-0', tool_calls=[{'name': 'tavily_search_results_json', 'args': {'que
ry': 'current weather in San Francisco'}, 'id': 'call_v0EHAA8cgE33PI4vQ
UaLUyMB'}])]
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in San Francisco'}, 'id': 'call_v0EHAA8cgE33PI4vQUaLUyMB'}
Back to the model!
[ToolMessage(content='[{\'url\': \'https://www.weatherapi.com/\', \'con
tent\': "{\'location\': {\'name\': \'San Francisco\', \'region\': \'Cal
ifornia\', \'country\': \'United States of America\', \'lat\': 37.78,
\'lon\': -122.42, \'tz_id\': \'America/Los_Angeles\', \'localtime_epoch
\': 1712959432, \'localtime\': \'2024-04-12 15:03\'}, \'current\': {\'l
ast_updated_epoch\': 1712959200, \'last_updated\': \'2024-04-12 15:00
\', \'temp_c\': 15.0, \'temp_f\': 59.0, \'is_day\': 1, \'condition\':
{\'text\': \'Partly cloudy\', \'icon\': \'//cdn.weatherapi.com/weather/
64x64/day/116.png\', \'code\': 1003}, \'wind_mph\': 16.1, \'wind_kph\':
25.9, \'wind_degree\': 210, \'wind_dir\': \'SSW\', \'pressure_mb\': 100
9.0, \'pressure_in\': 29.8, \'precip_mm\': 0.0, \'precip_in\': 0.0, \'h
umidity\': 60, \'cloud\': 75, \'feelslike_c\': 13.6, \'feelslike_f\': 5
6.4, \'vis_km\': 16.0, \'vis_miles\': 9.0, \'uv\': 4.0, \'gust_mph\': 2
0.6, \'gust_kph\': 33.1}}"}, {\'url\': \'https://www.weathertab.com/en/
c/e/04/united-states/california/san-francisco/\', \'content\': \'Explor
e comprehensive April 2024 weather forecasts for San Francisco, inclu
```



```
ns': 22, 'prompt_tokens': 153, 'total_tokens': 175}, 'model_name': 'gpt
-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_reason': 'too
l_calls', 'logprobs': None}, id='run-07978a9a-6798-480b-a8f4-b99c439e6a
ef-0', tool_calls=[{'name': 'tavily_search_results_json', 'args': {'que
ry': 'current weather in San Francisco'}, 'id': 'call_v0EHAA8cgE33PI4vQ
UaLUyMB'}])]
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in San Francisco'}, 'id': 'call_v0EHAA8cgE33PI4vQUaLUyMB'}
Back to the model!
[ToolMessage(content='[{\'url\': \'https://www.weatherapi.com/\', \'con
tent\': "{\'location\': {\'name\': \'San Francisco\', \'region\': \'Cal
ifornia\', \'country\': \'United States of America\', \'lat\': 37.78,
\'lon\': -122.42, \'tz_id\': \'America/Los_Angeles\', \'localtime_epoch
\': 1712959432, \'localtime\': \'2024-04-12 15:03\'}, \'current\': {\'l
ast_updated_epoch\': 1712959200, \'last_updated\': \'2024-04-12 15:00
\', \'temp_c\': 15.0, \'temp_f\': 59.0, \'is_day\': 1, \'condition\':
{\'text\': \'Partly cloudy\', \'icon\': \'//cdn.weatherapi.com/weather/
64x64/day/116.png\', \'code\': 1003}, \'wind_mph\': 16.1, \'wind_kph\':
25.9, \'wind_degree\': 210, \'wind_dir\': \'SSW\', \'pressure_mb\': 100
9.0, \'pressure_in\': 29.8, \'precip_mm\': 0.0, \'precip_in\': 0.0, \'h
umidity\': 60, \'cloud\': 75, \'feelslike_c\': 13.6, \'feelslike_f\': 5
6.4, \'vis_km\': 16.0, \'vis_miles\': 9.0, \'uv\': 4.0, \'gust_mph\': 2
0.6, \'gust_kph\': 33.1}}"}, {\'url\': \'https://www.weathertab.com/en/
c/e/04/united-states/california/san-francisco/\', \'content\': \'Explor
```



```
e comprehensive April 2024 weather forecasts for San Francisco, includi
ng daily high and low temperatures, precipitation risks, and monthly te
mperature trends. Featuring detailed day-by-day forecasts, dynamic grap
hs of daily rain probabilities, and temperature trends to help you plan
ahead. ... 12 64°F 49°F 18°C 9°C 49% 13 62°F 49°F ...\'}]', name='tavil
y_search_results_json', tool_call_id='call_v0EHAA8cgE33PI4vQUaLUyMB')]
[AIMessage(content='The current weather in San Francisco is partly clou
dy with a temperature of 59°F (15°C). The wind is blowing from the sout
h-southwest at 16.1 mph (25.9 kph). The humidity is at 60%, and the vis
ibility is 9 miles (16 km).', response_metadata={'token_usage': {'compl
etion_tokens': 62, 'prompt_tokens': 643, 'total_tokens': 705}, 'model_n
ame': 'gpt-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_rea
son': 'stop', 'logprobs': None}, id='run-89f58e44-dfa6-4145-93f2-3fdba
5763a8-0')]
```

AI Agents in LangGraph

```python
messages = [HumanMessage(content="What about in la?")]
thread = {"configurable": {"thread_id": "1"}}
for event in abot.graph.stream({"messages": messages}, thread):
    for v in event.values():
        print(v)
```

```
{'messages': [AIMessage(content='', additional_kwargs={'tool_calls':
[{'id': 'call_xg0cm52ju8a3Ib6g4dYGzk29', 'function': {'arguments': '{"q
uery":"current weather in Los Angeles"}', 'name': 'tavily_search_result
s_json'}, 'type': 'function'}]}, response_metadata={'token_usage': {'co
mpletion_tokens': 22, 'prompt_tokens': 717, 'total_tokens': 739}, 'mode
l_name': 'gpt-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_
reason': 'tool_calls', 'logprobs': None}, id='run-aafe90d2-d6c0-4697-80
8d-5a83d187b050-0', tool_calls=[{'name': 'tavily_search_results_json',
'args': {'query': 'current weather in Los Angeles'}, 'id': 'call_xg0cm5
2ju8a3Ib6g4dYGzk29'}])]}
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in Los Angeles'}, 'id': 'call_xg0cm52ju8a3Ib6g4dYGzk29'}
Back to the model!
{'messages': [ToolMessage(content='[{\'url\': \'https://www.weatherapi.
com/\', \'content\': "{\'location\': {\'name\': \'Los Angeles\', \'regi
on\': \'California\', \'country\': \'United States of America\', \'lat
\': 34.05, \'lon\': -118.24, \'tz_id\': \'America/Los_Angeles\', \'loca
ltime_epoch\': 1712959468, \'localtime\': \'2024-04-12 15:04\'}, \'curr
ent\': {\'last_updated_epoch\': 1712959200, \'last_updated\': \'2024-04
-12 15:00\', \'temp_c\': 16.7, \'temp_f\': 62.1, \'is_day\': 1, \'condi
tion\': {\'text\': \'Overcast\', \'icon\': \'//cdn.weatherapi.com/weath
er/64x64/day/122.png\', \'code\': 1009}, \'wind_mph\': 2.2, \'wind_kph
\': 3.6, \'wind_degree\': 177, \'wind_dir\': \'S\', \'pressure_mb\': 10
15.0, \'pressure_in\': 29.96, \'precip_mm\': 0.0, \'precip_in\': 0.0,
\'humidity\': 67, \'cloud\': 100, \'feelslike_c\': 16.7, \'feelslike_f
\': 62.1, \'vis_km\': 16.0, \'vis_miles\': 9.0, \'uv\': 5.0, \'gust_mph
\': 13.1, \'gust_kph\': 21.1}}"}, {\'url\': \'https://www.latimes.com/c
alifornia/story/2024-04-12/cold-weather-is-coming-los-angeles\', \'cont
ent\': \'Jireh (they/them) is the 2023-24 fellow at the Los Angeles Tim
es and a queer Asian American writer and filmmaker born and raised in t
he San Gabriel Valley. More From the Los Angeles Times Business\'}]',
```


```
l_name': 'gpt-4-turbo', 'system_fingerprint': 'fp_76f018034d', 'finish_
reason': 'tool_calls', 'logprobs': None}, id='run-aafe90d2-d6c0-4697-80
8d-5a83d187b050-0', tool_calls=[{'name': 'tavily_search_results_json',
'args': {'query': 'current weather in Los Angeles'}, 'id': 'call_xg0cm5
2ju8a3Ib6g4dYGzk29'}])]}
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in Los Angeles'}, 'id': 'call_xg0cm52ju8a3Ib6g4dYGzk29'}
Back to the model!
{'messages': [ToolMessage(content='[{\'url\': \'https://www.weatherapi.
com/\', \'content\': "{\'location\': {\'name\': \'Los Angeles\', \'regi
on\': \'California\', \'country\': \'United States of America\', \'lat
\': 34.05, \'lon\': -118.24, \'tz_id\': \'America/Los_Angeles\', \'loca
ltime_epoch\': 1712959468, \'localtime\': \'2024-04-12 15:04\'}, \'curr
ent\': {\'last_updated_epoch\': 1712959200, \'last_updated\': \'2024-04
-12 15:00\', \'temp_c\': 16.7, \'temp_f\': 62.1, \'is_day\': 1, \'condi
tion\': {\'text\': \'Overcast\', \'icon\': \'//cdn.weatherapi.com/weath
er/64x64/day/122.png\', \'code\': 1009}, \'wind_mph\': 2.2, \'wind_kph
\': 3.6, \'wind_degree\': 177, \'wind_dir\': \'S\', \'pressure_mb\': 10
15.0, \'pressure_in\': 29.96, \'precip_mm\': 0.0, \'precip_in\': 0.0,
\'humidity\': 67, \'cloud\': 100, \'feelslike_c\': 16.7, \'feelslike_f
\': 62.1, \'vis_km\': 16.0, \'vis_miles\': 9.0, \'uv\': 5.0, \'gust_mph
\': 13.1, \'gust_kph\': 21.1}}"}, {\'url\': \'https://www.latimes.com/c
alifornia/story/2024-04-12/cold-weather-is-coming-los-angeles\', \'cont
ent\': \'Jireh (they/them) is the 2023-24 fellow at the Los Angeles Tim
es and a queer Asian American writer and filmmaker born and raised in t
he San Gabriel Valley. More From the Los Angeles Times Business\'}]', n
```


```
ame='tavily_search_results_json', tool_call_id='call_xg0cm52ju8a3Ib6g4d
YGzk29')]}
{'messages': [AIMessage(content='The current weather in Los Angeles is
overcast with a temperature of 62.1°F (16.7°C). The wind is blowing fro
m the south at a speed of 2.2 mph (3.6 kph). The humidity level is 67%,
and visibility is 9 miles (16 km).', response_metadata={'token_usage':
{'completion_tokens': 65, 'prompt_tokens': 1178, 'total_tokens': 1243},
'model_name': 'gpt-4-turbo', 'system_fingerprint': 'fp_a39722e138', 'fi
nish_reason': 'stop', 'logprobs': None}, id='run-0d2ea079-0efe-4a5c-8c6
4-d1f4a52cabc3-0')]}
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
messages = [HumanMessage(content="Which one is warmer?")]
thread = {"configurable": {"thread_id": "1"}}
for event in abot.graph.stream({"messages": messages}, thread):
    for v in event.values():
        print(v)
```

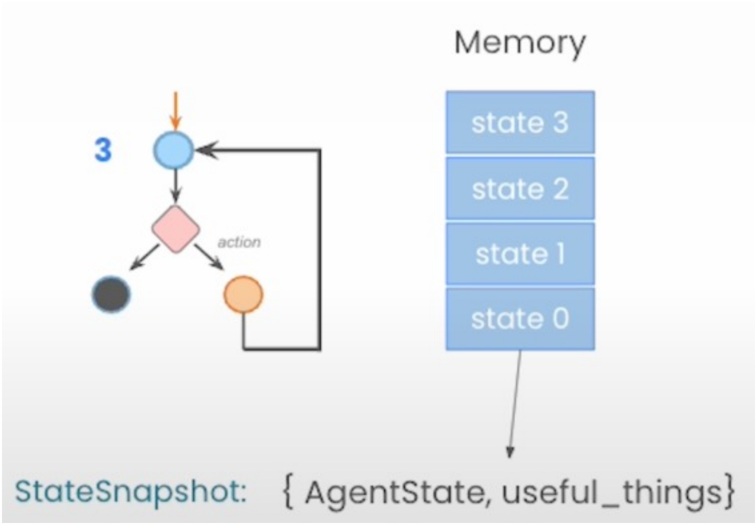
```
{'messages': [AIMessage(content='Los Angeles is currently warmer with a
temperature of 62.1°F (16.7°C) compared to San Francisco, which has a t
emperature of 59°F (15°C).', response_metadata={'token_usage': {'comple
tion_tokens': 38, 'prompt_tokens': 1255, 'total_tokens': 1293}, 'model_
name': 'gpt-4-turbo', 'system_fingerprint': 'fp_67e6987839', 'finish_re
ason': 'stop', 'logprobs': None}, id='run-810ffd03-8f21-48a5-ac30-ac5ac
358f465-0')]}
```

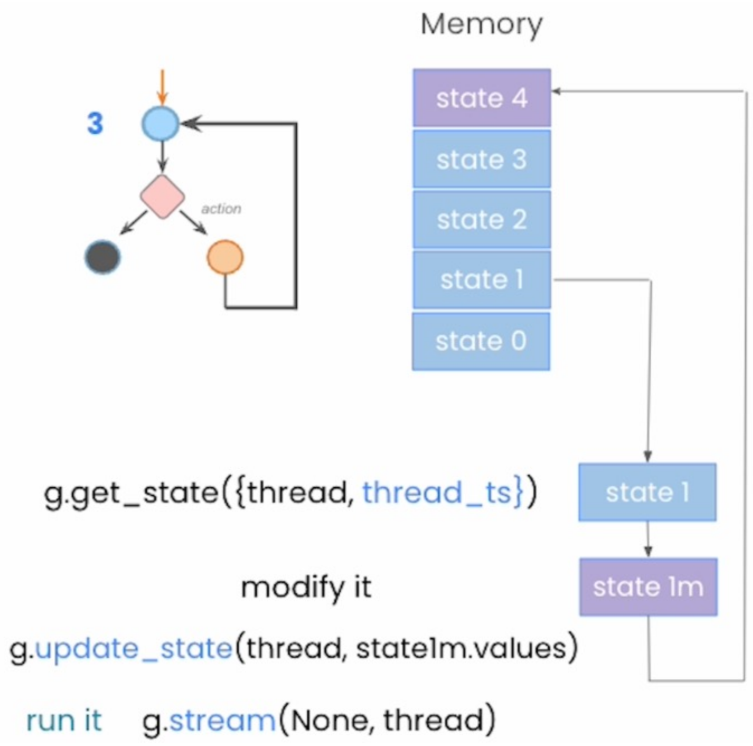
```python
messages = [HumanMessage(content="Which one is warmer?")]
thread = {"configurable": {"thread_id": "2"}}
for event in abot.graph.stream({"messages": messages}, thread):
    for v in event.values():
        print(v)
```

```
{'messages': [AIMessage(content='Could you please specify the two or mo
re items you are comparing to determine which one is warmer?', response
_metadata={'token_usage': {'completion_tokens': 20, 'prompt_tokens': 15
1, 'total_tokens': 171}, 'model_name': 'gpt-4-turbo', 'system_fingerpri
nt': 'fp_a39722e138', 'finish_reason': 'stop', 'logprobs': None}, id='r
un-d84c5fe0-8ab5-4ca6-89e6-575467ed1ded-0')]}
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

Tracking and Manipulating State Memory for Agents in LangGraph





Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

AI Agents in LangGraph

```python
from langgraph.checkpoint.aiosqlite import AsyncSqliteSaver

memory = AsyncSqliteSaver.from_conn_string(":memory:")
abot = Agent(model, [tool], system=prompt, checkpointer=memory)
```

```python
messages = [HumanMessage(content="What is the weather in SF?")]
thread = {"configurable": {"thread_id": "4"}}
async for event in abot.graph.astream_events({"messages": messages}, thr
    kind = event["event"]
    if kind == "on_chat_model_stream":
        content = event["data"]["chunk"].content
        if content:
            # Empty content in the context of OpenAI means
            # that the model is asking for a tool to be invoked
            # So we only print non-empty content
            print(content, end="|")
```

```
/Users/dlai.filming/Documents/GitHub/SC-Langchain-LangGraph-C5/.venv_11
_1_LC5/lib/python3.11/site-packages/langchain_core/_api/beta_decorator.
py:87: LangChainBetaWarning: This API is in beta and may change in the
future.
  warn_beta(
Calling: {'name': 'tavily_search_results_json', 'args': {'query': 'curr
ent weather in San Francisco'}, 'id': 'call_AIkbXdDIVJcjReXmEGIvnUKp'}
Back to the model!
The| current| weather| in| San| Francisco| is| partly| cloudy| with| a|
temperature| of| |59|°F| (|15|°C|).| Winds| are| coming| from| the| sou
th|-s|outh|west| at| |16|.|1| mph| (|25|.|9| k|ph|).| The| humidity| is
| at| |60|%,| and| the| atmospheric| pressure| is| |100|9| mb|.| Visibi
lity| is| approximately| |9| miles| (|16| km|).|
```


```python
from langgraph.checkpoint.aiosqlite import AsyncSqliteSaver

memory = AsyncSqliteSaver.from_conn_string(":memory:")
abot = Agent(model, [tool], system=prompt, checkpointer=memory)
```


Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

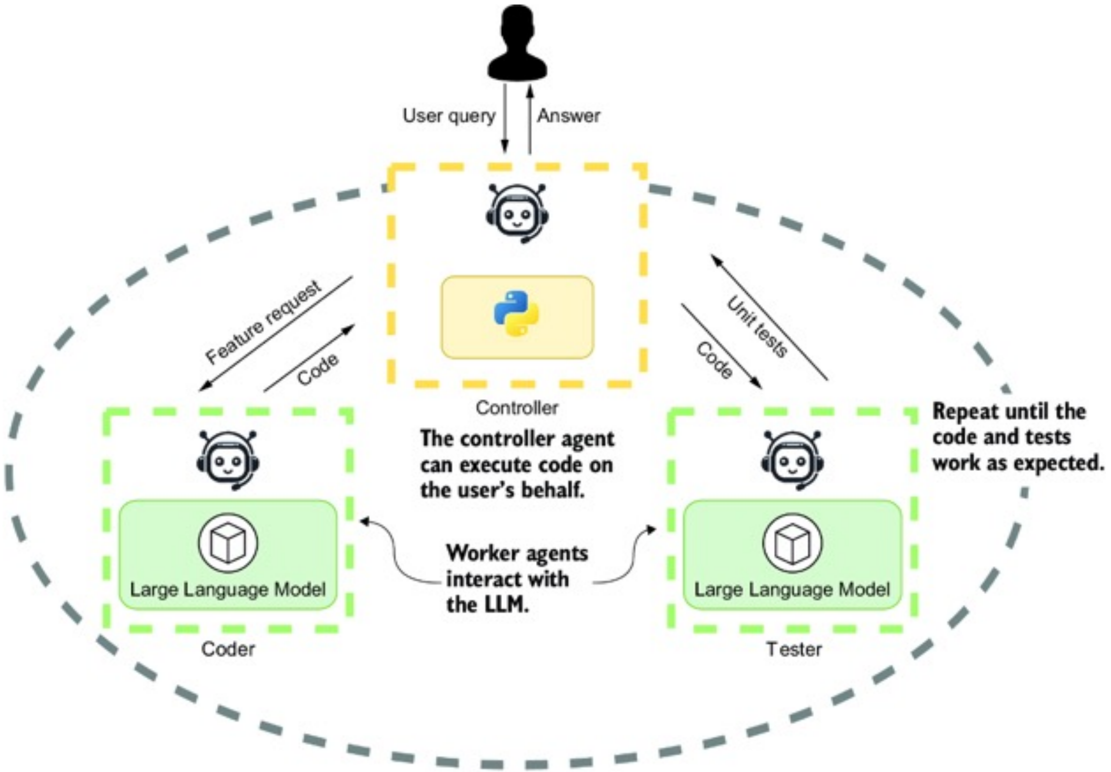
Human in the Loop w/ LangGraph

```python
class Agent:
    def __init__(self, model, tools, system="", checkpointer=None):
        self.system = system
        graph = StateGraph(AgentState)
        graph.add_node("llm", self.call_openai)
        graph.add_node("action", self.take_action)
        graph.add_conditional_edges("llm", self.exists_action, {True: "a
        graph.add_edge("action", "llm")
        graph.set_entry_point("llm")
        self.graph = graph.compile(
            checkpointer=checkpointer,
            interrupt_before=["action"]
        )
        self.tools = {t.name: t for t in tools}
        self.model = model.bind_tools([tool])

    def call_openai(self, state: AgentState):
        messages = state['messages']
        if self.system:
            messages = [SystemMessage(content=self.system)] + messages
        message = self.model.invoke(messages)
        return {'messages': [message]}

    def exists_action(self, state: AgentState):
        print(state)
        result = state['messages'][-1]
        return len(result.tool_calls) > 0

    def take_action(self, state: AgentState):
        tool_calls = state['messages'][-1].tool_calls
        results = []
        for t in tool_calls:
            print(f"Calling: {t}")
            result = self.tools[t['name']].invoke(t['args'])
            results.append(ToolMessage(tool_call_id=t['id'], name=t['n
```

Source: AI Agents in LangGraph– a short course by Harrison Chase, Rotem Weiss, https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/

From Single-Agent to Multi-Agent Systems

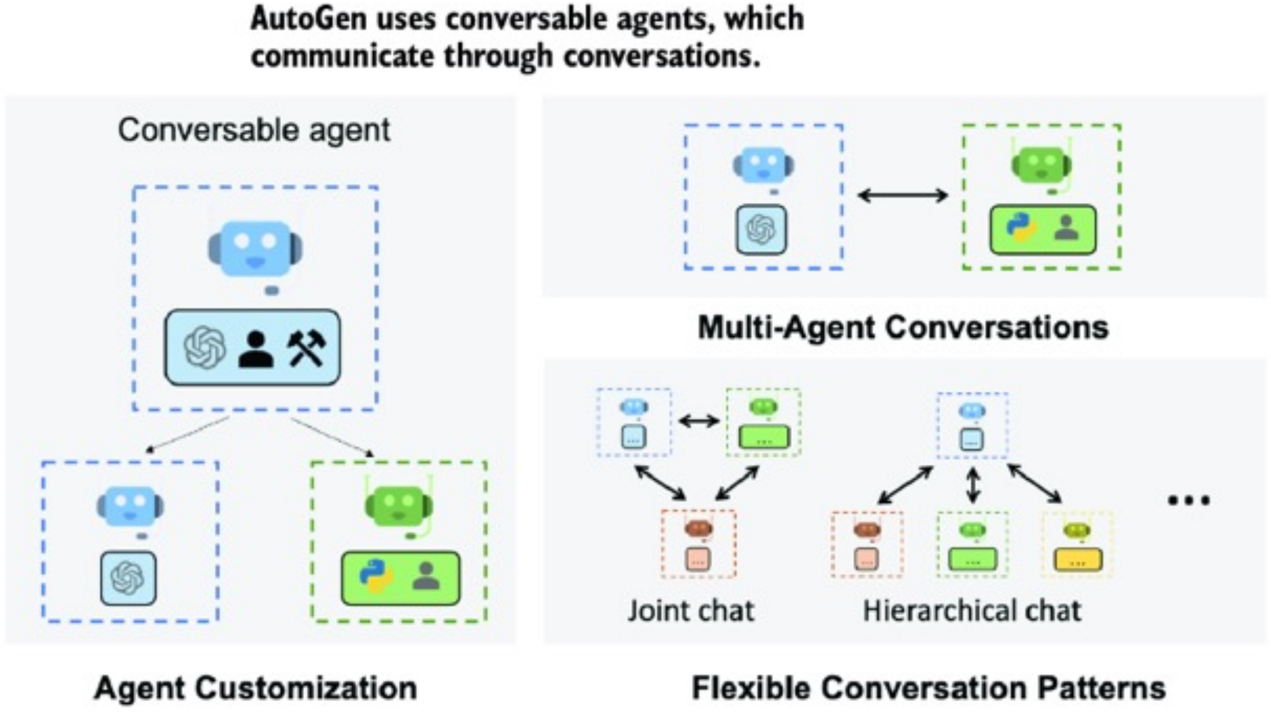
A Sample Multi-Agent System



In this example of a multi-agent system, the controller or agent proxy communicates directly with the user. Two agents—a coder and a tester —work in the background to create code and write unit tests to test the code.

Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

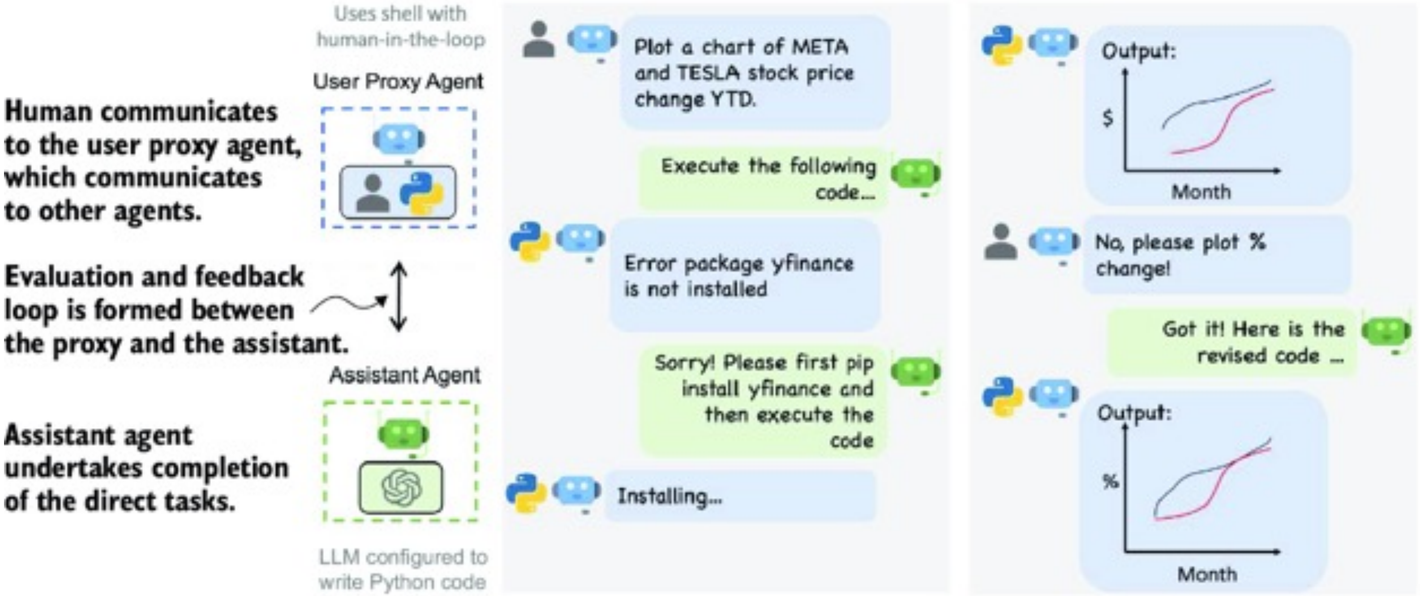
How AutoGen Agents communicates/ coordinates via Multi-Agent “Conversation”



Source: AI Agents in Action,
by Michael Lanham,
Feb 2025,
Manning Publications

This figure shows a schematic diagram of the agent connection/communication patterns AutoGen employs. AutoGen is a conversational multi agent platform because communication is done using natural language. Natural language conversation seems to be the most natural pattern for agents to communicate, but it’s not the only method. AutoGen supports various conversational patterns, from group and hierarchical to the more common and simpler proxy communication. In proxy communication, one agent acts as a proxy and directs communication to relevant agents to complete tasks.

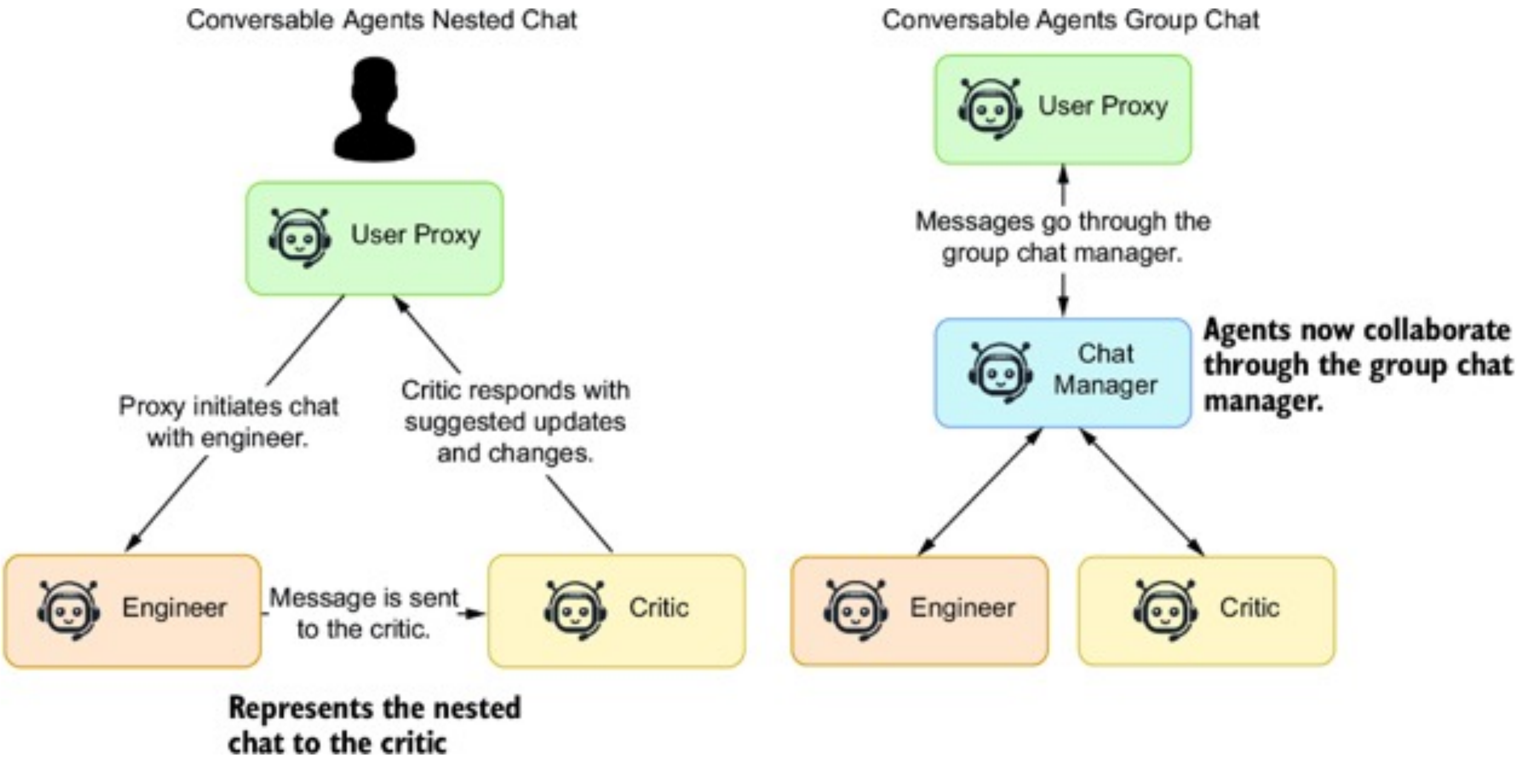
User Proxy Agent and Assistant Agent Communication under AutoGen



The basic pattern in AutoGen uses a UserProxy and one or more assistant agents. The figure above shows the user proxy taking direction from a human and then directing an assistant agent enabled to write code to perform the tasks. Each time the assistant completes a task, the proxy agent reviews, evaluates, and provides feedback to the assistant. This iteration loop continues until the proxy is satisfied with the results.

Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Nested vs. Group Chat for Conversable Agents under AutoGen

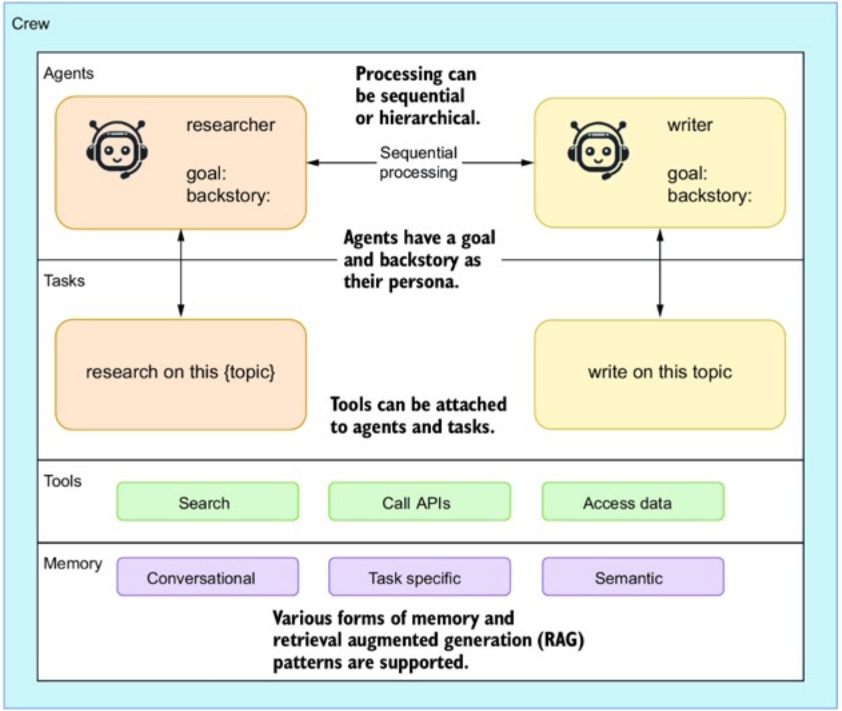


Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Composition and Sequential Processing in CrewAI

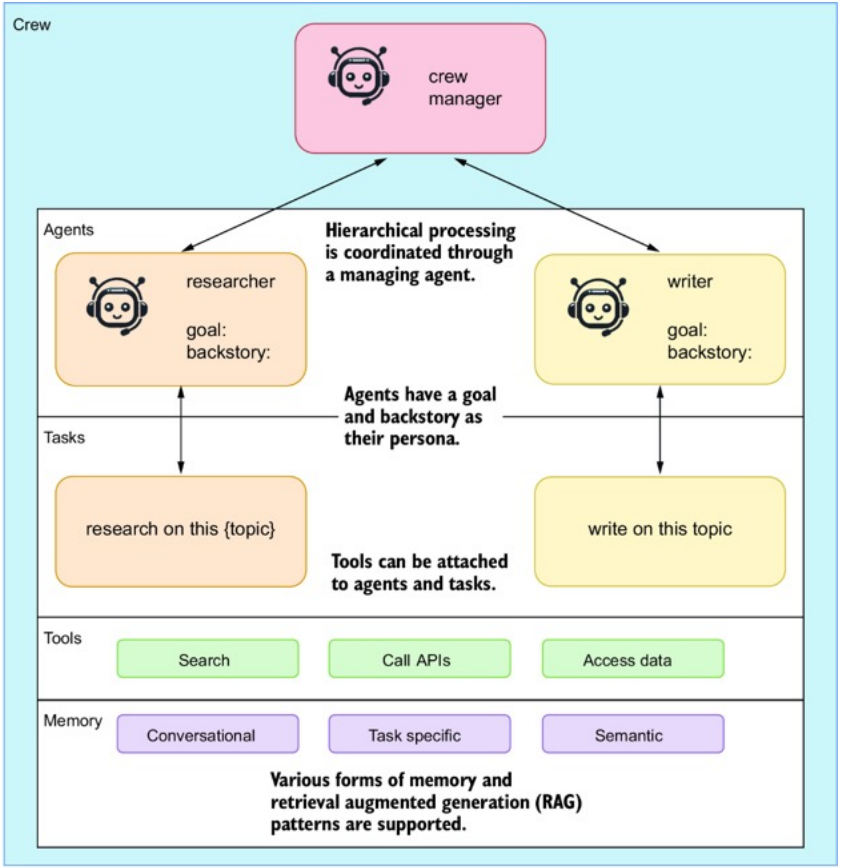
CrewAI supports two primary forms of processing: sequential and hierarchical. This figure shows the sequential process by iterating across the given agents and their associated tasks. In the next section, we dig into some code to set up a crew and employ it to complete a goal and create a good joke.

Figure 4.10 shows the main elements of the CrewAI platform, how they connect together, and their primary function. It shows a sequential-processing agent system with generic researcher and writer agents. Agents are assigned tasks that may also include tools or memory to assist them.



Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Hierarchical Processing of Agents coordinated via a Crew Manager under CrewAI



Source: AI Agents in Action, by Michael Lanham, Feb 2025, Manning Publications

Additional Types of Agentic Frameworks

Different Types of Agentic Frameworks

(per ServiceNow Inc., creator of TapeAgents)

Frameworks that Address Agent Development Needs

- Resumable sessions
- Low-code components
- Fine-grain control
- Concurrency
- Streaming

Examples:
LangGraph, AutoGen, Crew:

- Agent == resumable modular state machine

Frameworks focus on Data-Driven Agent Optimization

- Structured Agent Config.
- Structured Agent Logs
- Optimization algorithms

Examples:
DSPy, TextGrad, Trace:

- Agent == code that uses Structured modules and generates Structured logs

"Holistic" Frameworks

TapeAgents:

- Agent == resumable modular state machine
- with Structured config.
- that makes granular Structured logs
- that can make fine-tuning data from logs
- and can reuse other agents' logs

The DSPy Framework

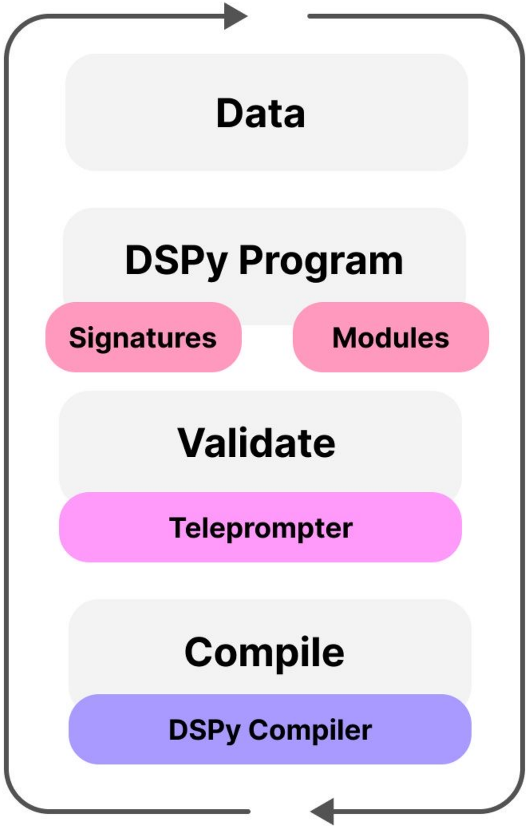
Omar Khattab, “Compound AI Systems & the DSPy Framework”

What is DSPy ?

- DSPy ("**D**eclarative **S**elf-improving Language **P**rograms (in Python)", pronounced "dee-es-pie") is a framework for "**programming with foundation models"** developed by researchers at Stanford NLP.
- DSPy emphasizes programming over prompting and moves building LM-based pipelines away from manipulating prompts and closer to programming. Thus, it aims to solve the fragility problem in building LM-based applications.
- DSPy provides a more systematic approach to building LM-based applications by separating the information flow of your program from the parameters (prompts and LM weights) of each step. DSPy will then take your program and automatically optimize how to prompt (or finetune) LMs for your particular task.

Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

DSPy in a nutshell



1. **Collect training data**
   Examples of expected input and outputs
2. **Write program**
   Define logic with signatures and modules
3. **Define validation**
   Define optimization techniques using teleprompters
4. **Compile program**
   Compile training data, program logic and validation techniques to optimize prompts
5. **Repeat**
   Repeat process until desired performance is reached

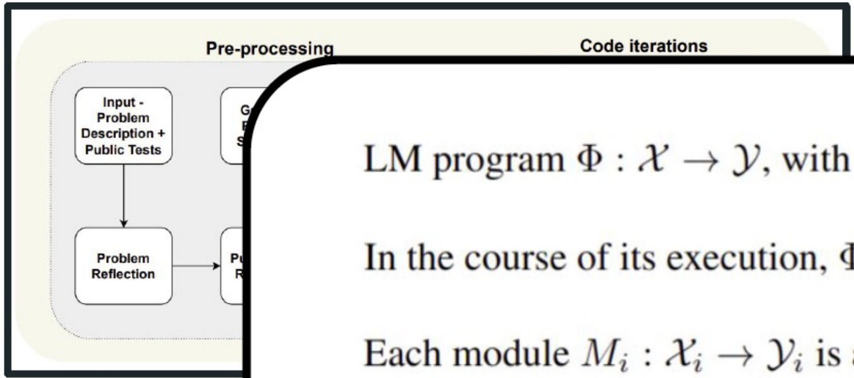
Source: https://www.linkedin.com/posts/edwardschmuhl_dspy-is-on-fire-right-now-declarative-activity-7170825791604256769-6IdF/

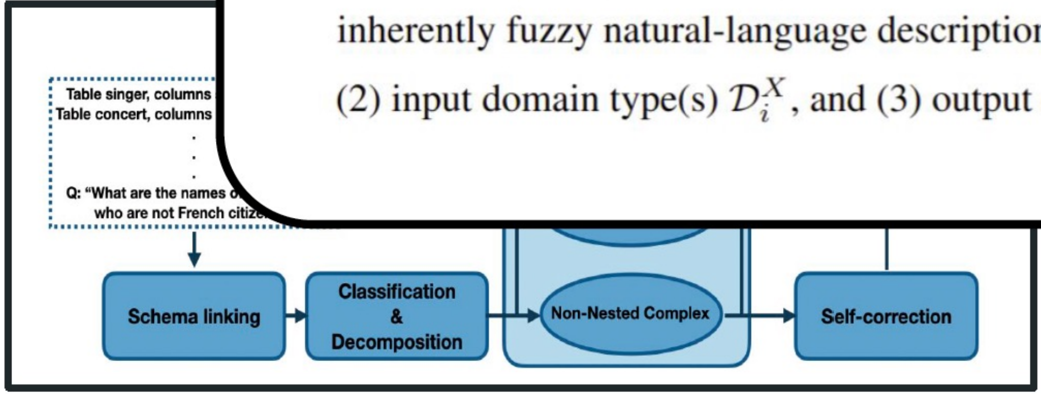
New Concepts/ Abstractions under DSPy

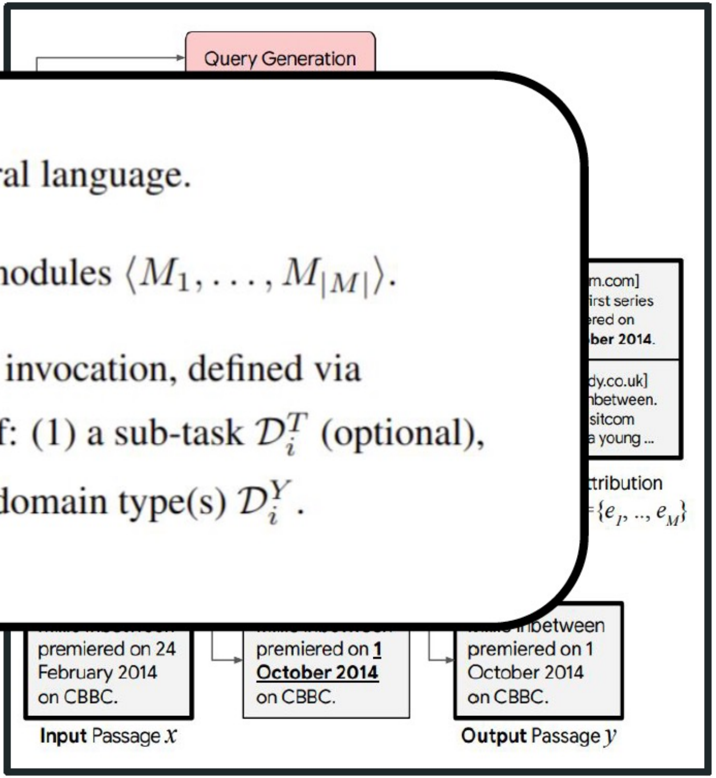
- Hand-written prompts and fine-tuning are abstracted and replaced by signatures
- Prompting techniques, such as Chain of Thought or ReAct, are abstracted and replaced by modules
- Manual prompt engineering is automated with optimizers (teleprompters) and a DSPy Compiler

Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

DSPy







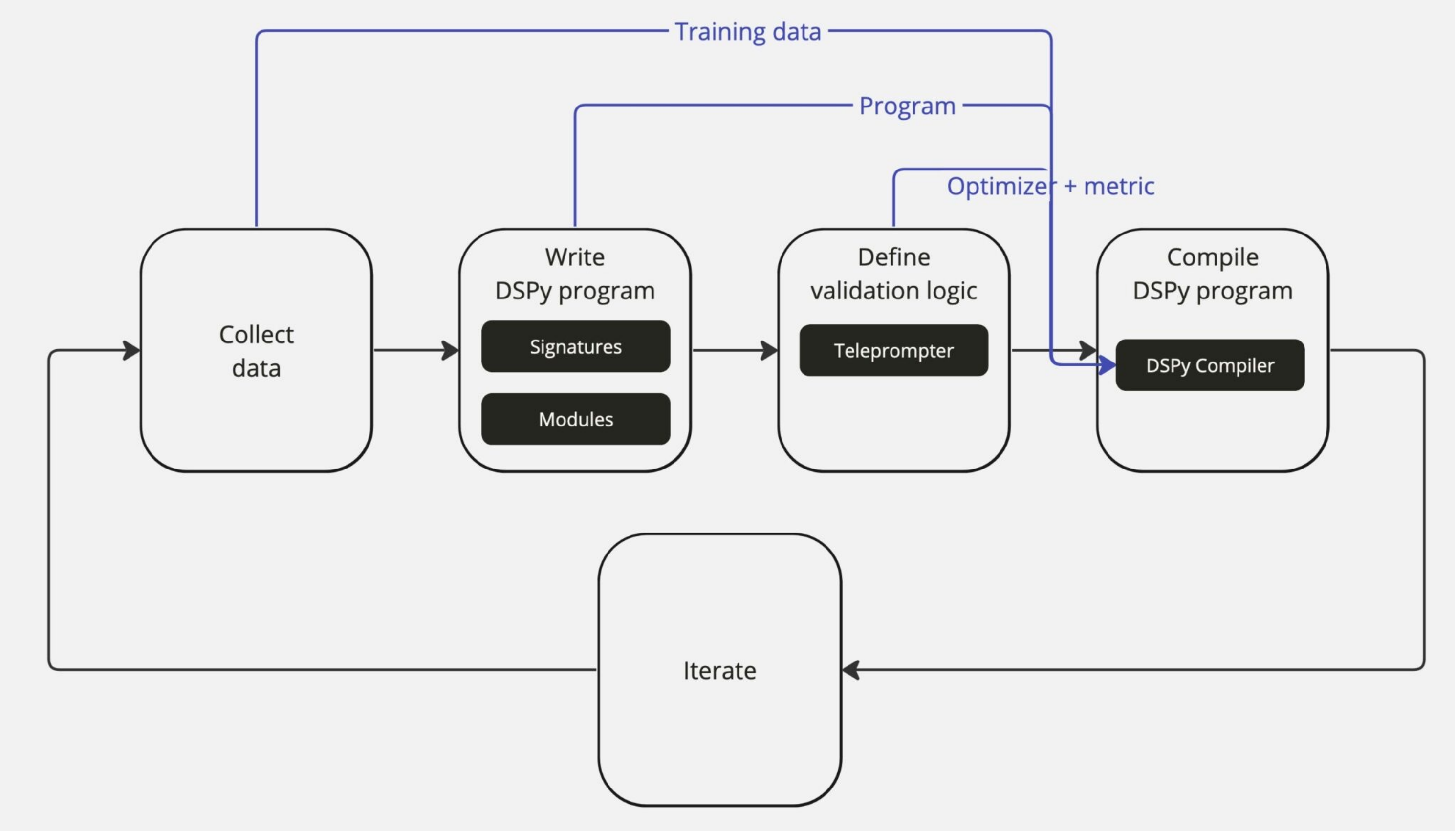
LM program $\Phi : \mathcal{X} \rightarrow \mathcal{Y}$, with $\mathcal{X}$ and $\mathcal{Y}$ in natural language.

In the course of its execution, Φ makes calls to modules $\langle M_1, \ldots, M_{|M|} \rangle$.

Each module $M_i : \mathcal{X}_i \rightarrow \mathcal{Y}_i$ is a *declarative* LM invocation, defined via inherently fuzzy natural-language descriptions of: (1) a sub-task $\mathcal{D}_i^T$ (optional), (2) input domain type(s) $\mathcal{D}_i^X$, and (3) output co-domain type(s) $\mathcal{D}_i^Y$.

Source: Omar Khattab, "Compound AI Systems & the DSPy Framework," https://rdi.berkeley.edu/llm-agents-mooc/slides/dspy_lec.pdf

Workflow of DSPy



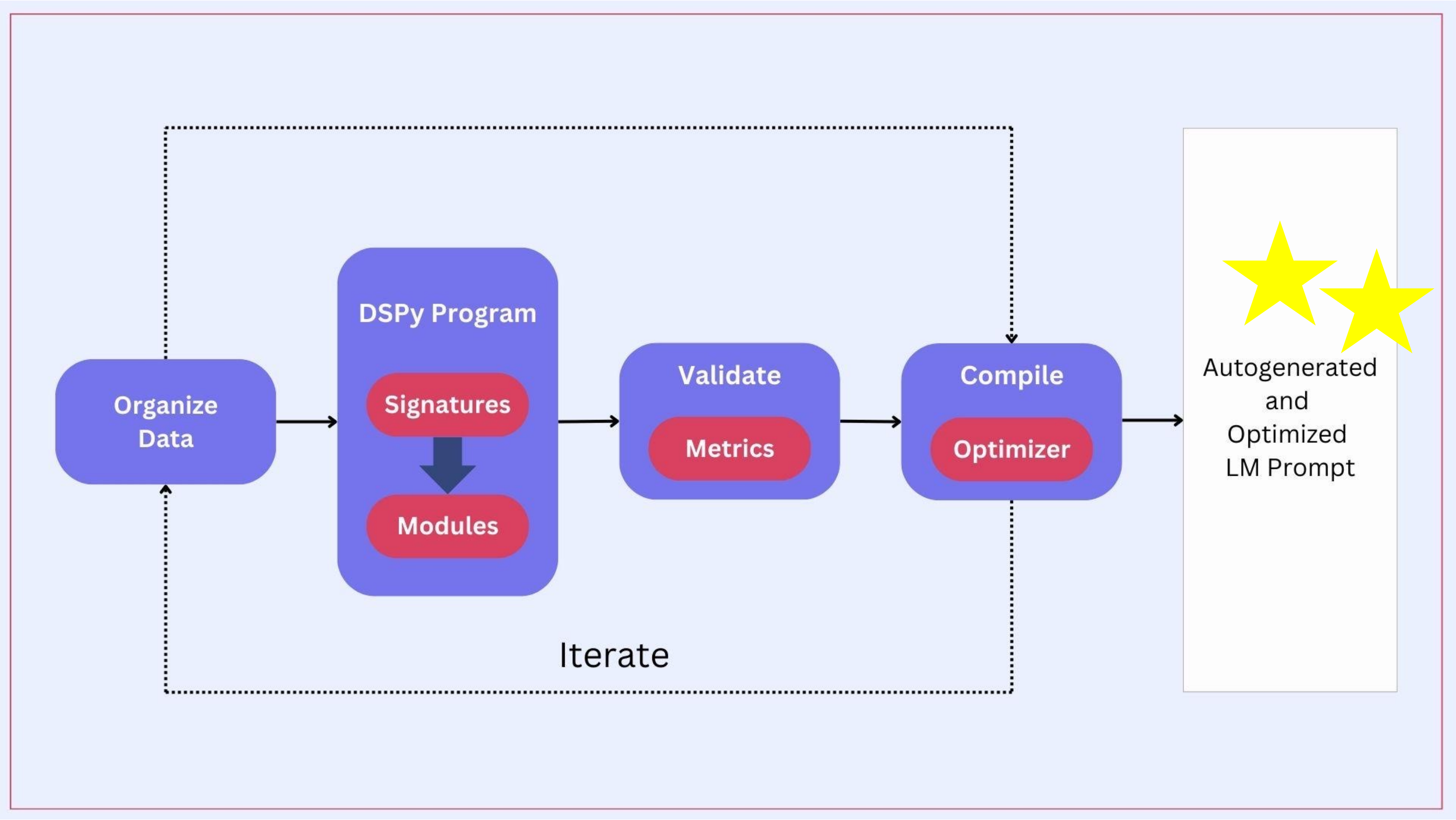
Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

Workflow of DSPy

1. **Collect dataset:** Collect a few examples of the inputs and outputs of your program (e.g., question and answer pairs), which will be used to optimize your pipeline.
2. **Write DSPy program:** Define your program's logic with signatures and modules and the information flow among the components to solve your task.
3. **Define validation logic:** Define a logic to optimize your program for using a validation metric and an optimizer (teleprompter).
4. **Compile DSPy program:** The DSPy compiler takes the training data, program, optimizer, and validation metric into account to optimize your program (e.g., prompts or finetunes).
5. **Iterate:** Repeat the process by improving your data, program, or validation until you are happy with your pipeline's performance.

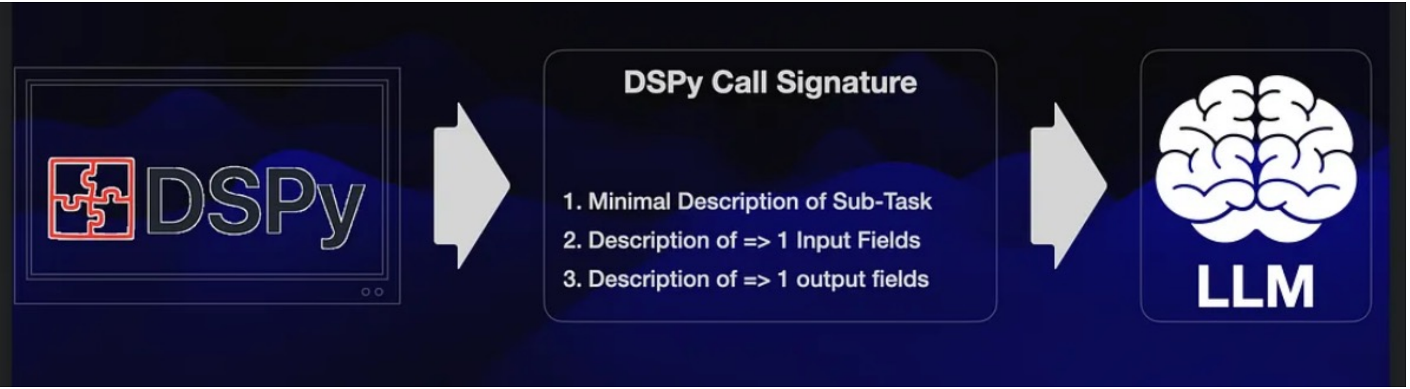
Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

Workflow of DSPy



Source: https://qdrant.tech/blog/dspy-vs-langchain

The notion of "Signature" in DSPy



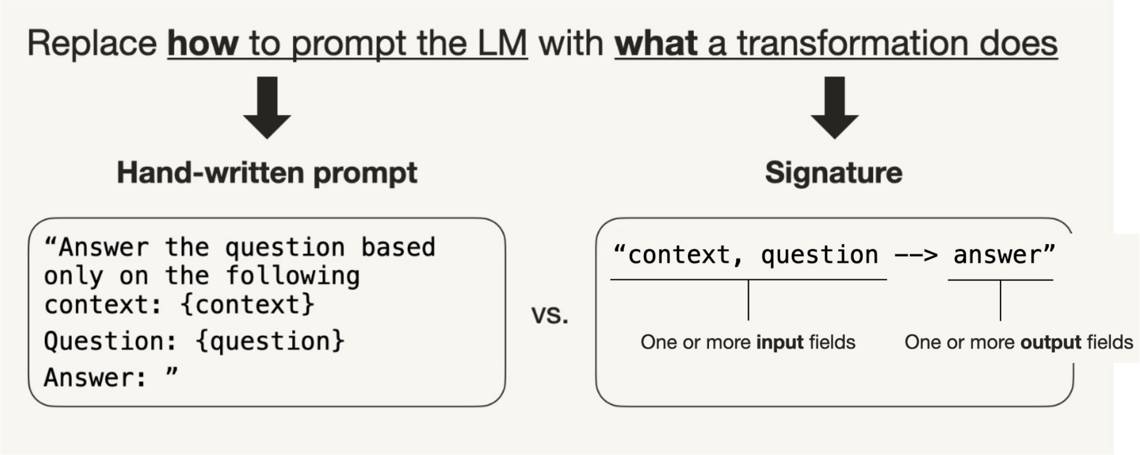
Signature
Every call from DSPy to the LLM requires a signature, the signature consists of three elements, as shown above: 1) A minimal description of the sub-task. A description of one or more 2) input fields and 3) output fields.

Instead of free-form string prompts, DSPy programs use **natural language *signatures*** to assign work to the LM. A DSPy signature is ***natural-language typed* declaration** of a function which can be described as a short declarative specification that tells DSPy ***what*** a text transformation needs to do. Rather than ***how*** a specific LM should be prompted to implement that behaviour.

Source: https://cobusgreyling.medium.com/an-introduction-to-dspy-00306973acbc

Examples of "Signatures" in DSPy

```
"question -> answer"

"long-document -> summary"

"context, question -> answer"
```



Complete Notation for Signature "context, question -> answer" :

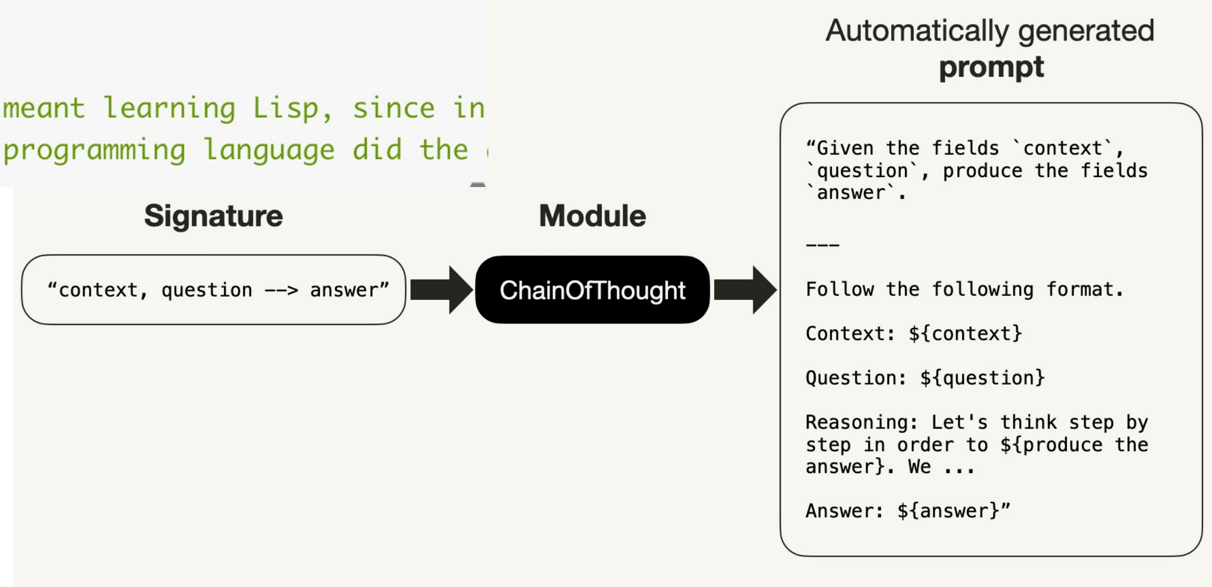
```python
class GenerateAnswer(dspy.Signature):
    """Answer questions with short factoid answers."""
    context = dspy.InputField(desc="may contain relevant facts")
    question = dspy.InputField()
    answer = dspy.OutputField(desc="often between 1 and 5 words")
```

Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

A Sample Chain-of-Thought “module” in DSPy

to abstract the specific CoT prompting technique

```python
# Option 1: Pass minimal signature to ChainOfThought module
generate_answer = dspy.ChainOfThought("context, question -> answer")

# Option 2: Or pass full notation signature to ChainOfThought module
generate_answer = dspy.ChainOfThought(GenerateAnswer)

# Call the module on a particular input.
pred = generate_answer(context = "Which meant learning Lisp, since in
                       question = "What programming language did the
```



<u>**Initial Implementation of the Signature** “context, question -> answer” with a ChainOfThought module :</u>

Some built-in modules in DSPy

- `[dspy.Predict]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspypredict)`: Processes the input and output fields, generates instructions, and creates a template for the specified `signature`.
- `[dspy.ChainOfThought]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspychainofthought)`: Inherits from the `Predict` module and adds functionality for "Chain of Thought" processing.
- `[dspy.ChainOfThoughtWithHint]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspychainofthoughtwithhint)`: Inherits from the `Predict` module and enhances the `ChainOfThought` module with the option to provide hints for reasoning.
- `[dspy.MultiChainComparison]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspymultichaincomparison)`: Inherits from the `Predict` module and adds functionality for multiple chain comparisons.
- `[dspy.Retrieve]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspyretrieve)`: Retrieves passages from a retriever module.
- `[dspy.ReAct]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/modules.md#dspyreact)`: Designed to compose the interleaved steps of Thought, Action, and Observation.

❖ For abstracting the corresponding Prompting techniques

Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

Built-in Teleprompters (Optimizers) for DSPy programs

- Teleprompters automates (optimizes) prompting for arbitrary pipelines

```
from dspy.teleprompt import BootstrapFewShot

# Simple teleprompter example
teleprompter = BootstrapFewShot(metric=dspy.evaluate.answer_exact_mat
```

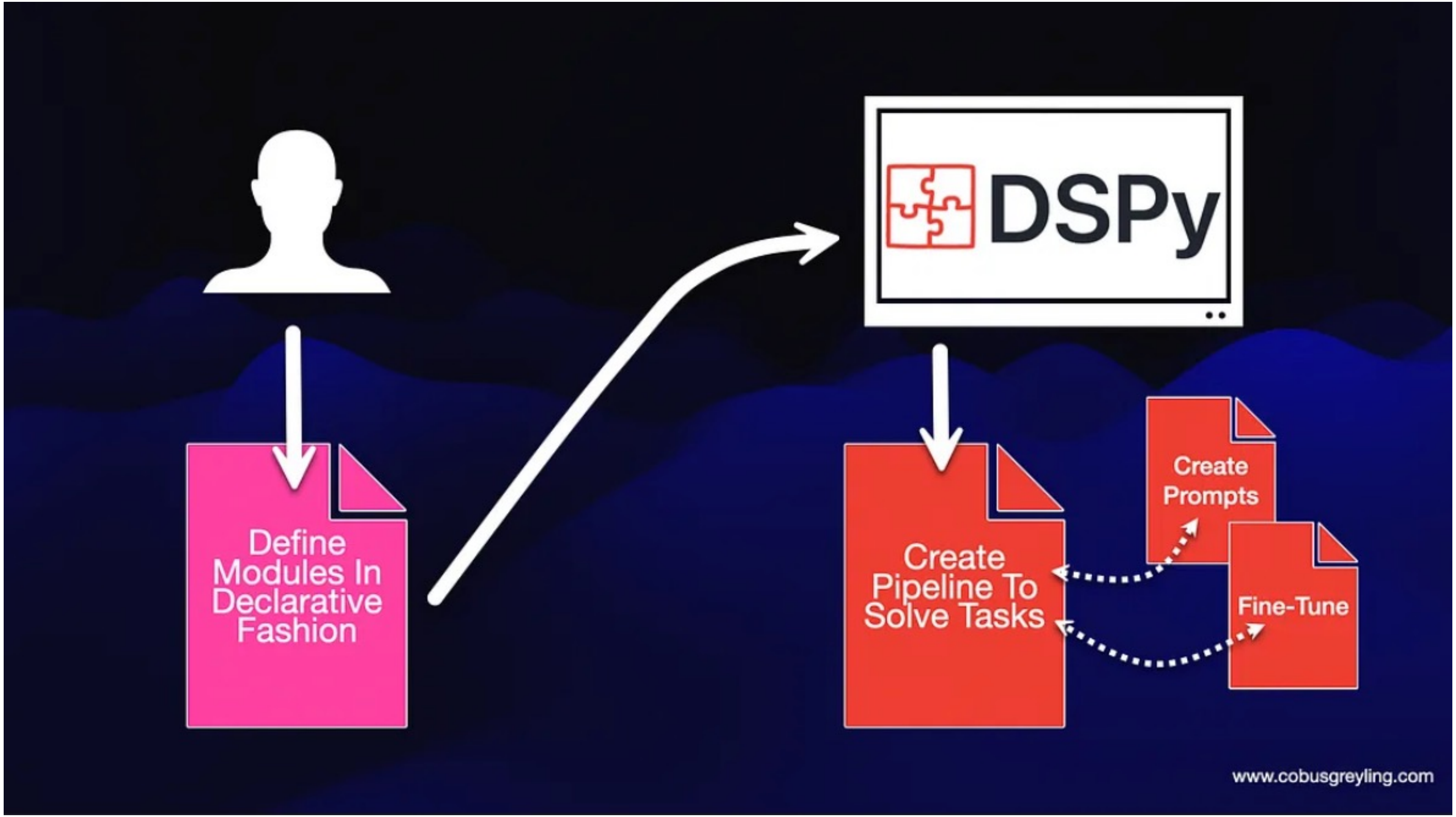
- A teleprompter takes a metric, and together with the DSPy compiler, learn to bootstrap select effective prompts for a DSPy program's modules.

- There are also SignatureOptimizers which improve the o/p prefixes the instruction of the signatures in a module under zero/few-shot setting.

- `[dspy.LabeledFewShot]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/teleprompters.md#telepromptlabeledfewshot)`: defining `k` number of samples to be used by the predictor.
- `[dspy.BootstrapFewShot]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/teleprompters.md#telepromptbootstrapfewshot)`: Bootstrapping
- `[dspy.BootstrapFewShotWithRandomSearch]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/teleprompters.md#telepromptbootstrapfewshotwithrandomsearch)`: Inherits from the `BootstrapFewShot` teleprompter and introduces additional attributes for the random search process.
- `[dspy.BootstrapFinetune]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/teleprompters.md#telepromptbootstrapfinetune)`: t defines the teleprompter as a `BootstrapFewShot` instance for the finetuning compilation.
- `[dspy.Ensemble]` `(https://github.com/stanfordnlp/dspy/blob/main/docs/teleprompters.md#telepromptensemble)`: Creates ensembled versions of multiple programs, reducing various outputs from different programs into a single output.

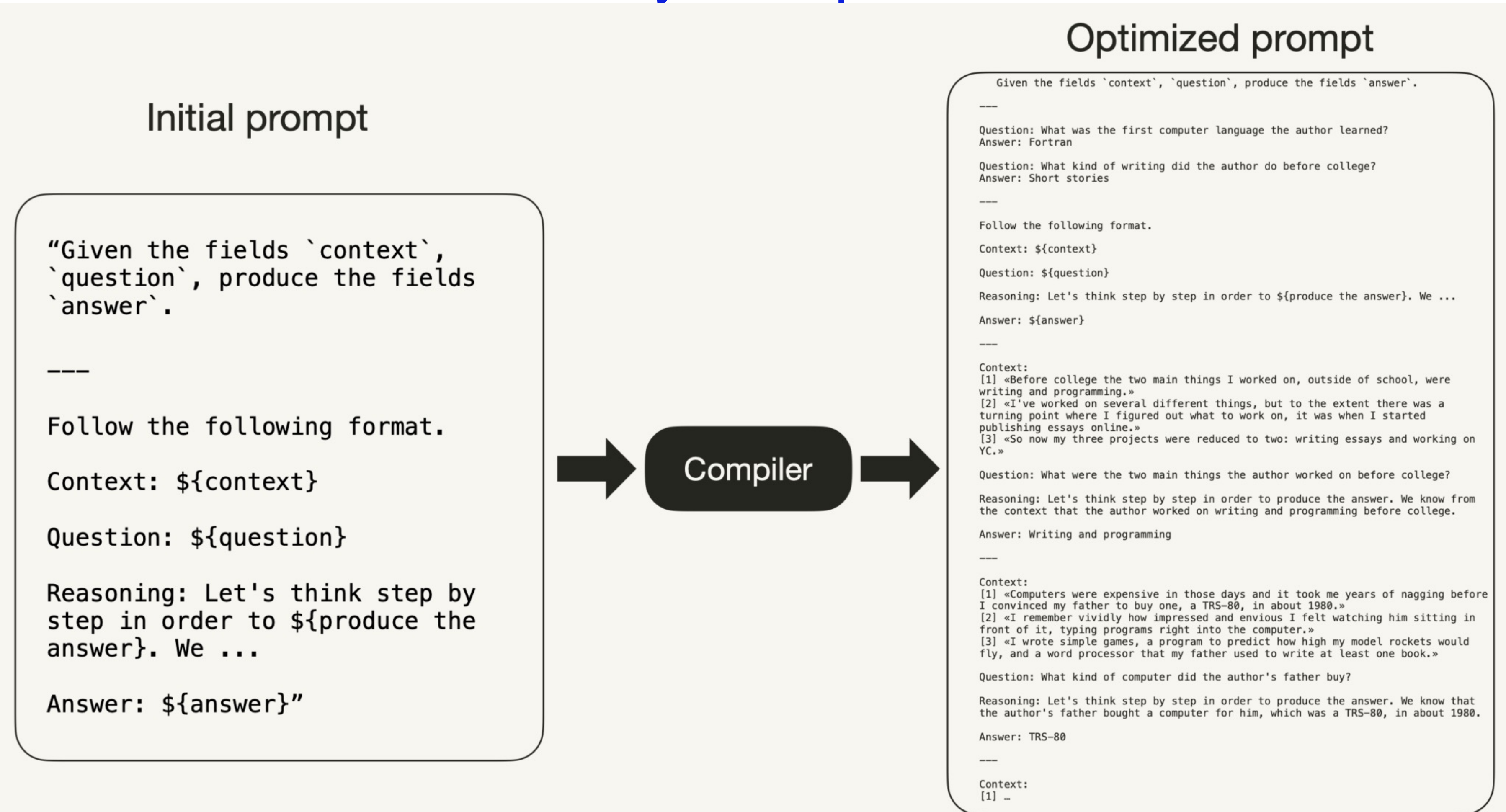
Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

The DSPy Optimizer



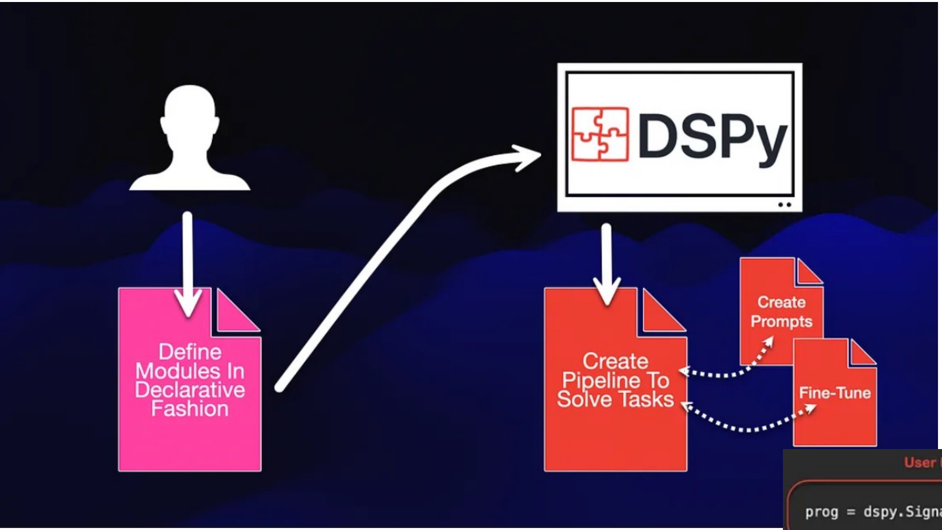
Source: https://cobusgreyling.medium.com/an-introduction-to-dspy-00306973acbc

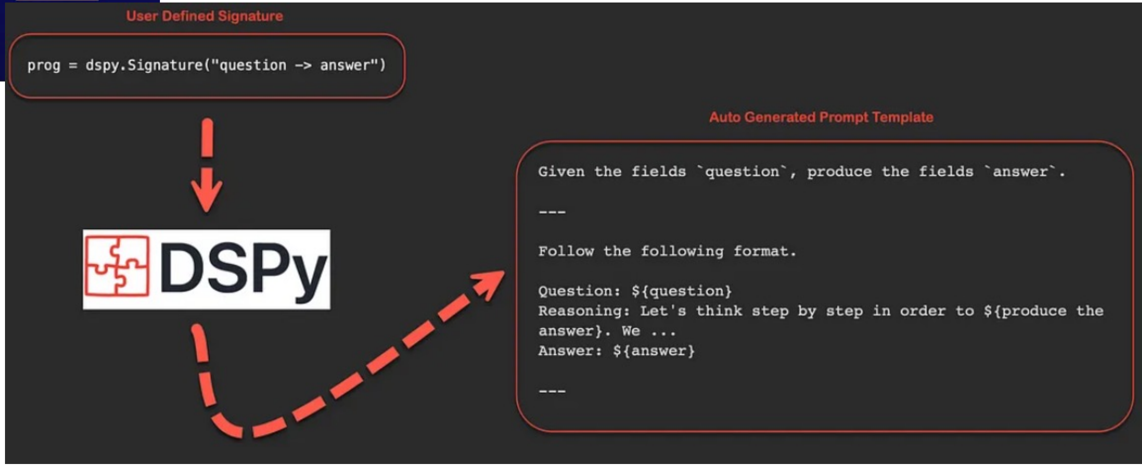
DSPy Compiler



Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

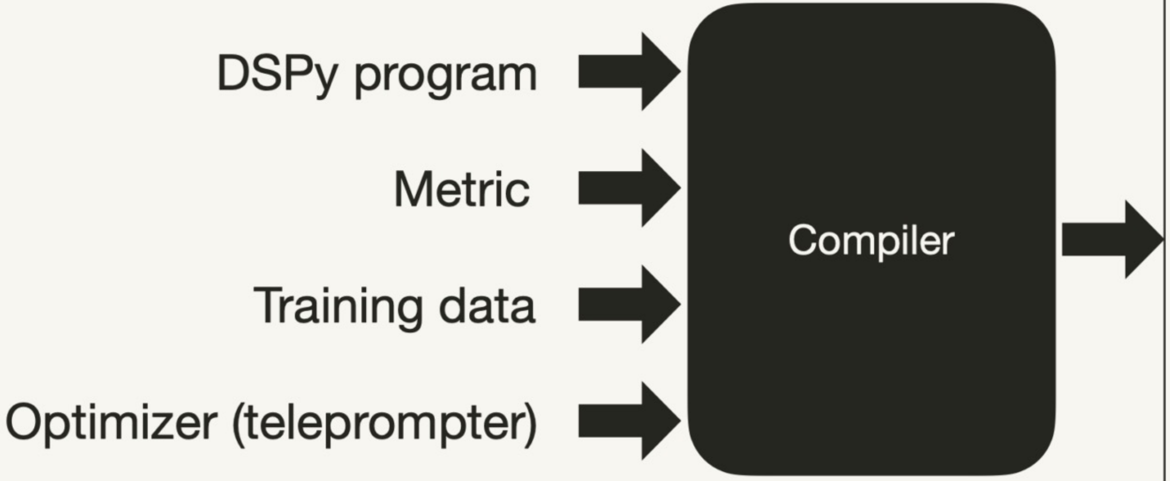
The DSPy Optimizer





Source: https://cobusgreyling.medium.com/an-introduction-to-dspy-00306973acbc

DSPy Compiler (cont'd)




```
Given the fields `context`, `question`, produce the fields `answer`.

---

Question: What was the first computer language the author learned?
Answer: Fortran

Question: What kind of writing did the author do before college?
Answer: Short stories

---

Follow the following format.

Context: ${context}

Question: ${question}

Reasoning: Let's think step by step in order to ${produce the answer}. We ...

Answer: ${answer}

---

Context:
[1] «Before college the two main things I worked on, outside of school, were
writing and programming.»
[2] «I've worked on several different things, but to the extent there was a
turning point where I figured out what to work on, it was when I started
publishing essays online.»
[3] «So now my three projects were reduced to two: writing essays and working on
YC.»

Question: What were the two main things the author worked on before college?

Reasoning: Let's think step by step in order to produce the answer. We know from
the context that the author worked on writing and programming before college.

Answer: Writing and programming

---

Context:
[1] «Computers were expensive in those days and it took me years of nagging before
I convinced my father to buy one, a TRS-80, in about 1980.»
[2] «I remember vividly how impressed and envious I felt watching him sitting in
front of it, typing programs right into the computer.»
[3] «I wrote simple games, a program to predict how high my model rockets would
fly, and a word processor that my father used to write at least one book.»

Question: What kind of computer did the author's father buy?

Reasoning: Let's think step by step in order to produce the answer. We know that
the author's father bought a computer for him, which was a TRS-80, in about 1980.

Answer: TRS-80

---

Context:
[1] …
```

Source: https://towardsdatascience.com/intro-to-dspy-goodbye-prompting-hello-programming-4ca1c6ce3eb9/

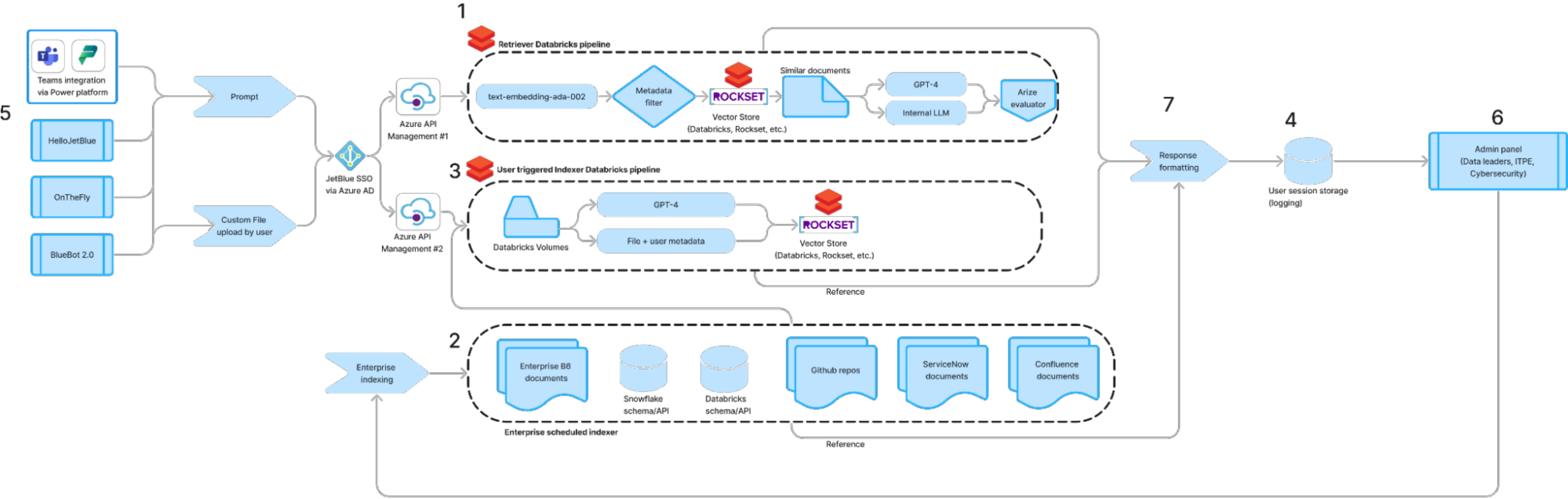
Comparative Analysis: LangChain vs. DSPy

Feature	LangChain	DSPy
Core Focus	Focus on providing a large number of building blocks to simplify the development of applications that use LLMs in conjunction with user-specified data sources.	Focus on automating and modularizing LLM interactions, eliminating manual prompt engineering and improving systematic reliability.
Approach	Utilizes modular components and chains that can be linked together using the LangChain Expression Language (LCEL).	Streamlines LLM interaction by prioritizing programming instead of prompting, and automating prompt refinement and weight tuning.
Complex Pipelines	Facilitates the creation of chains using LCEL, supporting asynchronous execution and integration with various data sources and APIs.	Simplifies multi-stage reasoning pipelines using modules and optimizers, and ensures scalability through less manual intervention.
Optimization	Relies on user expertise for prompt engineering and chaining of multiple LLM calls.	Includes built-in optimizers that automatically tune prompts and weights, and helps bring efficiency and effectiveness in LLM pipelines.
Community and Support	Large open-source community with extensive documentation and examples.	Emerging framework with growing community support, and bringing a paradigm-shift in LLM prompting.

IMHO,
DSPy though with high potential, is still a bit Immature as of early 2025 !

Source: https://qdrant.tech/blog/dspy-vs-langchain
Reference: https://www.infoworld.com/article/3956455/dspy-an-open-source-framework-for-llm-powered-applications.html

A Deployment Example: JetBlue's RAG Chatbot w/ Databricks & DSPy

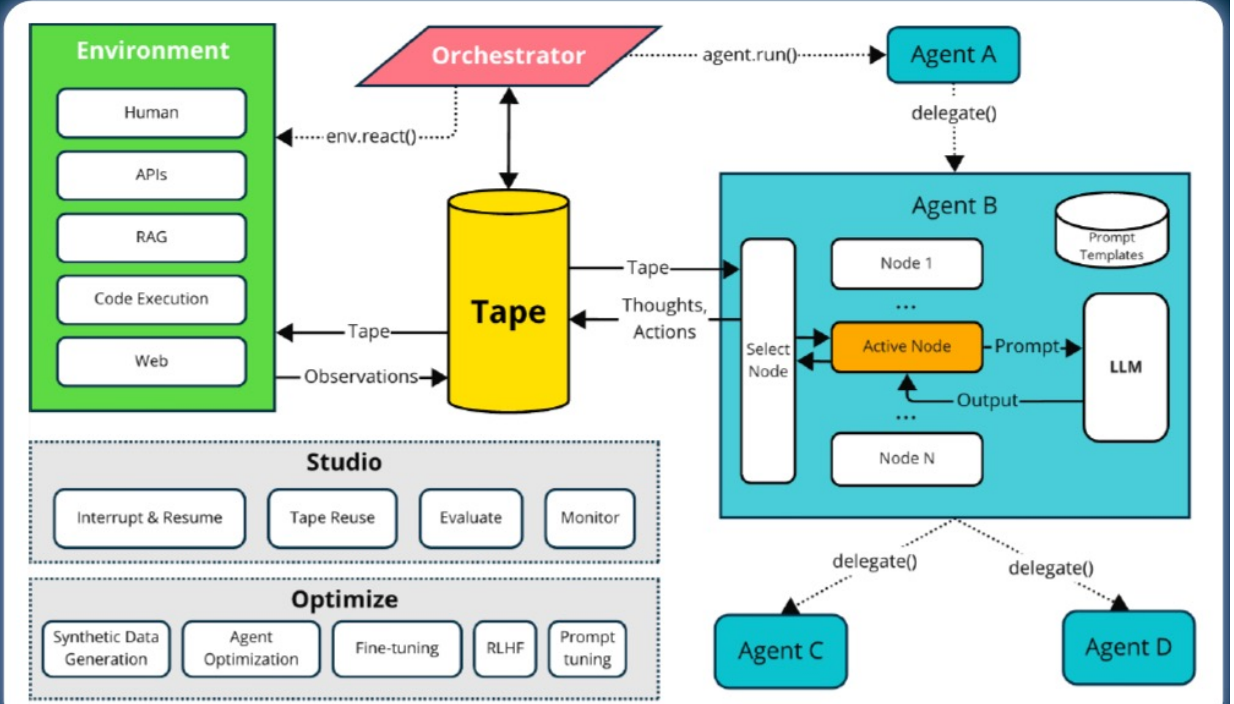


Source: https://www.databricks.com/blog/optimizing-databricks-llm-pipelines-dspy

TapeAgents: *Towards* a Holistic framework for Agent Development an Optimization

TapeAgents is a framework built around a structured, granular, semantic-level log: the tape

- Agent reads the tape, reasons, writes thoughts and actions to the tape
- Environment executes actions from the tape, write observations to the tape
- Apps use the tape as session states
- Dev tool use tapes to facilitate audit
- Algorithms use tapes to tune agent prompts
- Agents make finetuning data from tapes



Source: ServiceNow, Inc. 2024

TapeAgents w/ other types of Agentic Frameworks

Frameworks that Address Agent Development Needs

- Resumable sessions
- Low-code components
- Fine-grain control
- Concurrency
- Streaming

Examples:
LangGraph, AutoGen, Crew:

- Agent == resumable modular state machine

Frameworks focus on Data-Driven Agent Optimization

- Structured Agent Config.
- Structured Agent Logs
- Optimization algorithms

Examples:
DSPy, TextGrad, Trace:

- Agent == code that uses Structured modules and generates Structured logs

"Holistic" Frameworks

TapeAgents:

- Agent == resumable modular state machine
- with Structured config.
- that makes granular Structured logs
- that can make fine-tuning data from logs
- and can reuse other agents' logs

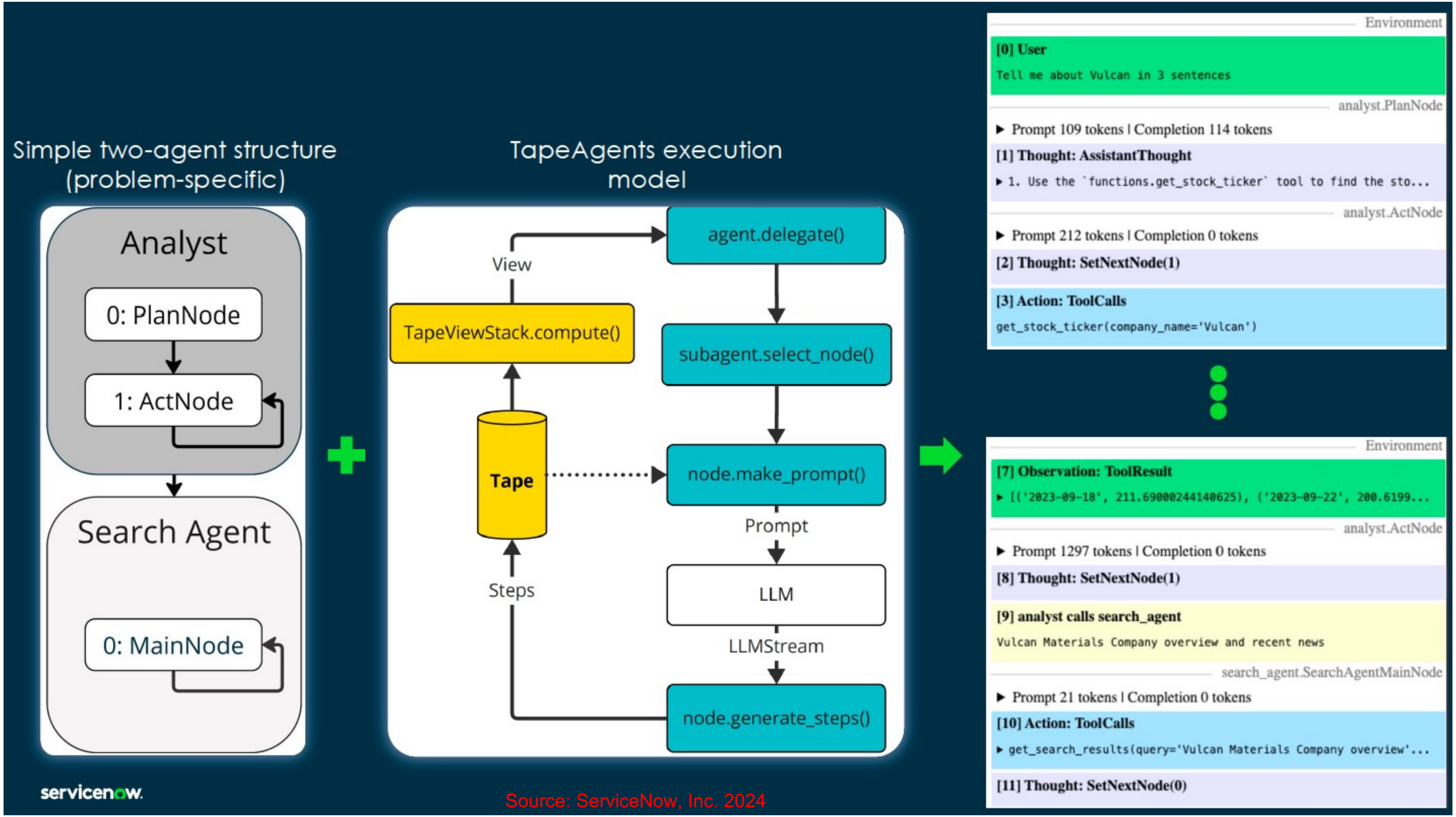
Comparing TapeAgents with other Agentic Frameworks: LangGraph, DSPy, AutoGen

	Development				Optimization		
Method	**Building from Components while Allowing Finegrained Flow Control**	**Native Streaming Support**	**Concurrent LLM Calls**	**Resumable State Machine Agents**	**Log Reuse Across Agents**	**Structured Logs and Agent Configurations for Data-Driven Agent Optimization**	**Making Training Text From Semantic-Level Logs**
DSPy	✓	✗	✓	✗	✗	✓	▲
LangGraph	✓	✓	✓	✓	▲	▲	✗
AutoGen	▲	▲	✗	▲	✗	▲	✗
TapeAgents (Ours)	✓	✓	✗	✓	✓	✓	✓

Table 5: *TapeAgents vs Other Frameworks.* *TapeAgents stands out in features it offers to the practitioner to the support them throughout the LLM Agent development cycle. In this figure, we use the cross sign (✗) to indicate that major core changes would be required for the framework support the feature. Triangle sign (▲) indicates partial support of a feature, meaning that practitioner would have to do extra effort or accept associated limitations to achieve the respective functionality. Check sign (✓) indicates that the framework natively supports a feature. TapeAgents's only weakness in this table is the lack of Concurrent LLM Calls, see Section 7 for a discuss of how we intend to tackle it.*

Source: ServiceNow, Inc. 2024

Agent Reasoning Loop example under TapeAgents



Source: ServiceNow, Inc. 2024

Agent Reasoning Loop example under TapeAgents

Source: ServiceNow, Inc. 2024

[0] User

```
kind: user

Tell me about Vulcan in 3 sentences
```

▶ **Prompt 1681 characters**

▶ **Completion**

[1] Thought: AssistantThought

```
by: Agent
kind: assistant_thought

To help the user learn about Vulcan, I will:

1. Use the `functions.get_stock_ticker` tool to find the stock ticker symbol for
Vulcan.
2. Use the `functions.get_stock_data` tool to retrieve recent stock price data
for Vulcan using the ticker symbol obtained in step 1.
3. Summarize the information about Vulcan, including its stock ticker and recent
stock performance, in a concise manner.
```

▶ **Prompt 2045 characters**

▶ **Completion**

[2] Action: ToolCalls

```
by: Agent
tool_calls:
- function:
    arguments: '{"company_name": "Vulcan"}'
    name: get_stock_ticker
  id: call_8u2pjjDy81QdEXpnHS1dNAqe
  type: function
- function:
    arguments: '{"company_name": "Vulcan Materials"}'
    name: get_stock_ticker
  id: call_gkVL07uUnIMBav4mDSHPTnvF
  type: function
kind: assistant
```

[3] Thought: Jump

```
by: Agent
kind: jump
next_node: 1
```

[4] Observation: ToolResult

```
tool_call_id: call_8u2pjjDy81QdEXpnHS1dNAqe
kind: tool

VMC
```

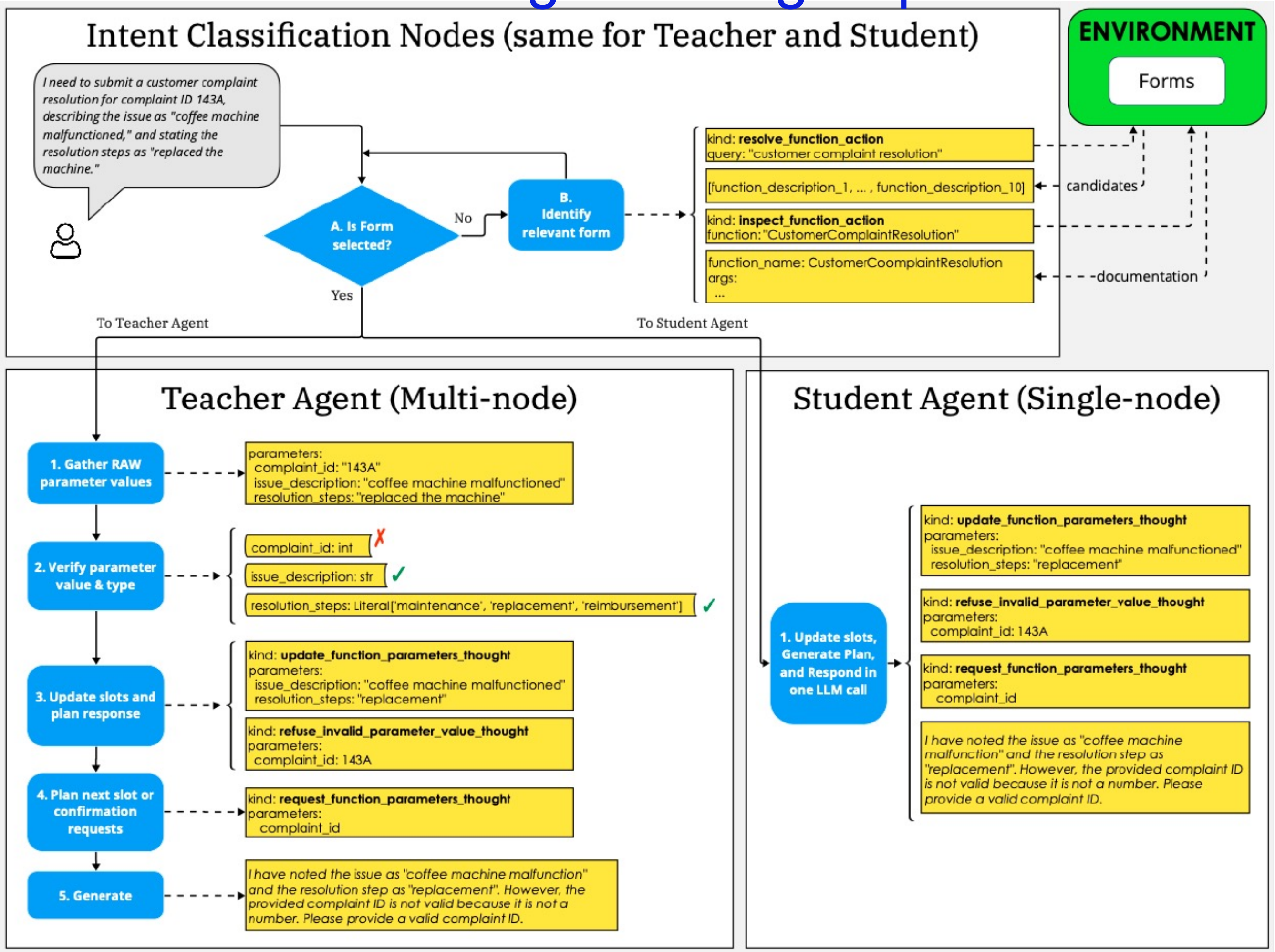
[5] Observation: ToolResult

```
tool_call_id: call_gkVL07uUnIMBav4mDSHPTnvF
kind: tool

VMC
```

▶ **Prompt 2388 characters**

▶ **Completion**

Optimization of Student-Agent using Tapes from Teacher-Agent



Source: ServiceNow, Inc. 2024

Case Study with TapeAgents: A Form-Filling Assistant

- Task: conversational assistant that routes the user to the right form and helps fill it
- Constraints: 5-star conversational experience at low compute cost
- 3 training domains: FlyCorp, BigBankCorp, CoffeeCorp
- 3 testing domains: DriveCorp, LuxuryCorp, ShopCorp
- Metric: GREADTH
  - Grounded, REsponsive, Accurate, Disciplined, Transparent, Helpful
- Method:
  - Generate synthetic tapes with 19 user agents and a 5-node LLAMA-405B Teacher
  - Finetune 1-node LLAMA-8B Student
- Outcome: Student matches GPT-4o performance at 300x lower cost

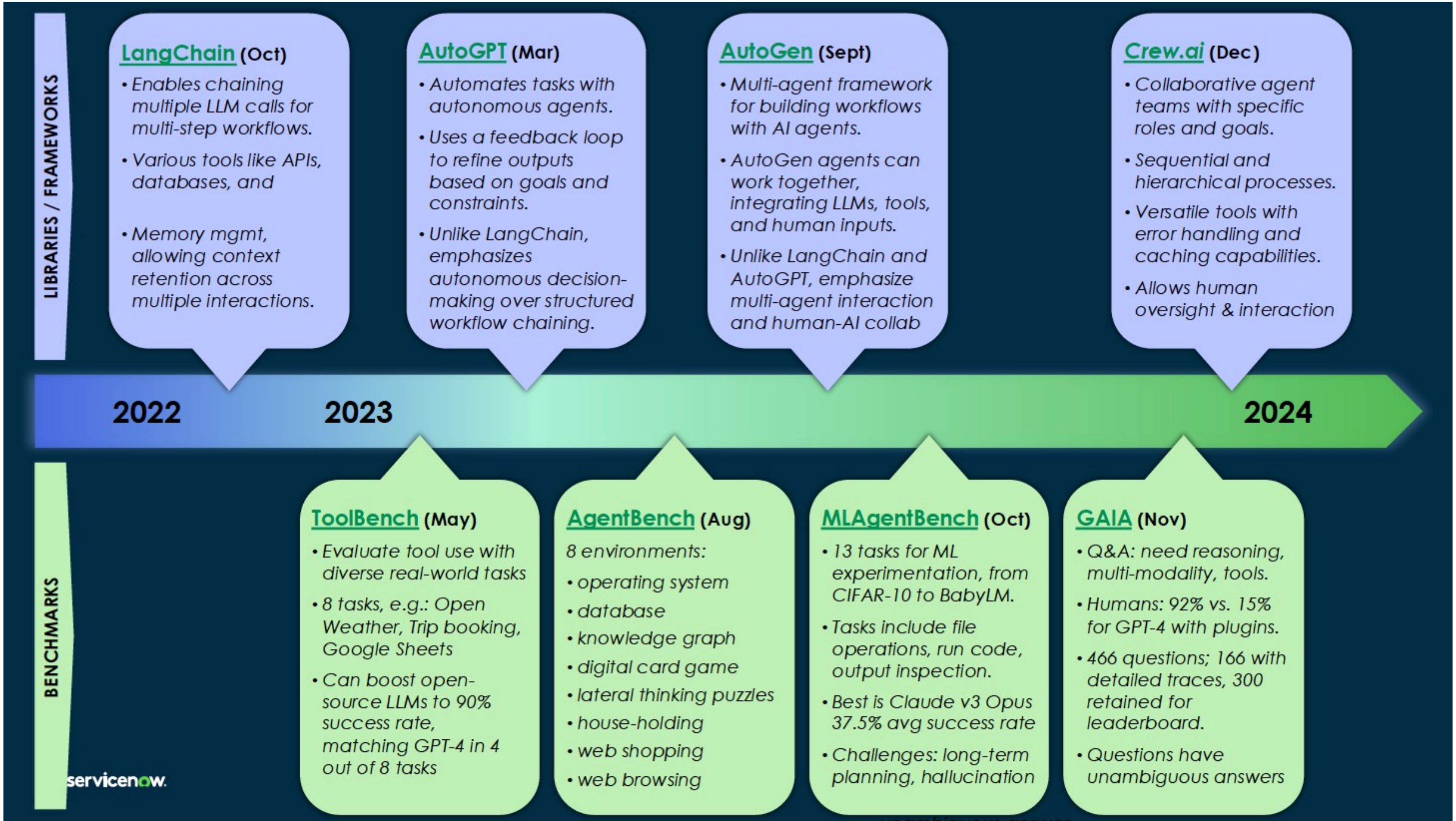
Experimental Results of Form-Filling Assistant

Table 3: *GREADTH Form Filler experiment results.* *The Teacher[1] is a multi-node agent with Llama 3.1 405B Instruct FP8 as its LLM. The Student[2] is a single-node agent with Llama 3.1 8b Instruct as its LLM. We also evaluate the multi-node agent with GPT-4o and with Llama 3.1 8B Instruct as its LLM, as well as the single-node agent with Llama 3.1 405B Instruct for comparison. The metrics are computed over 1524 partial dialogues from the test domains. Read full analysis in Section 5.4.*

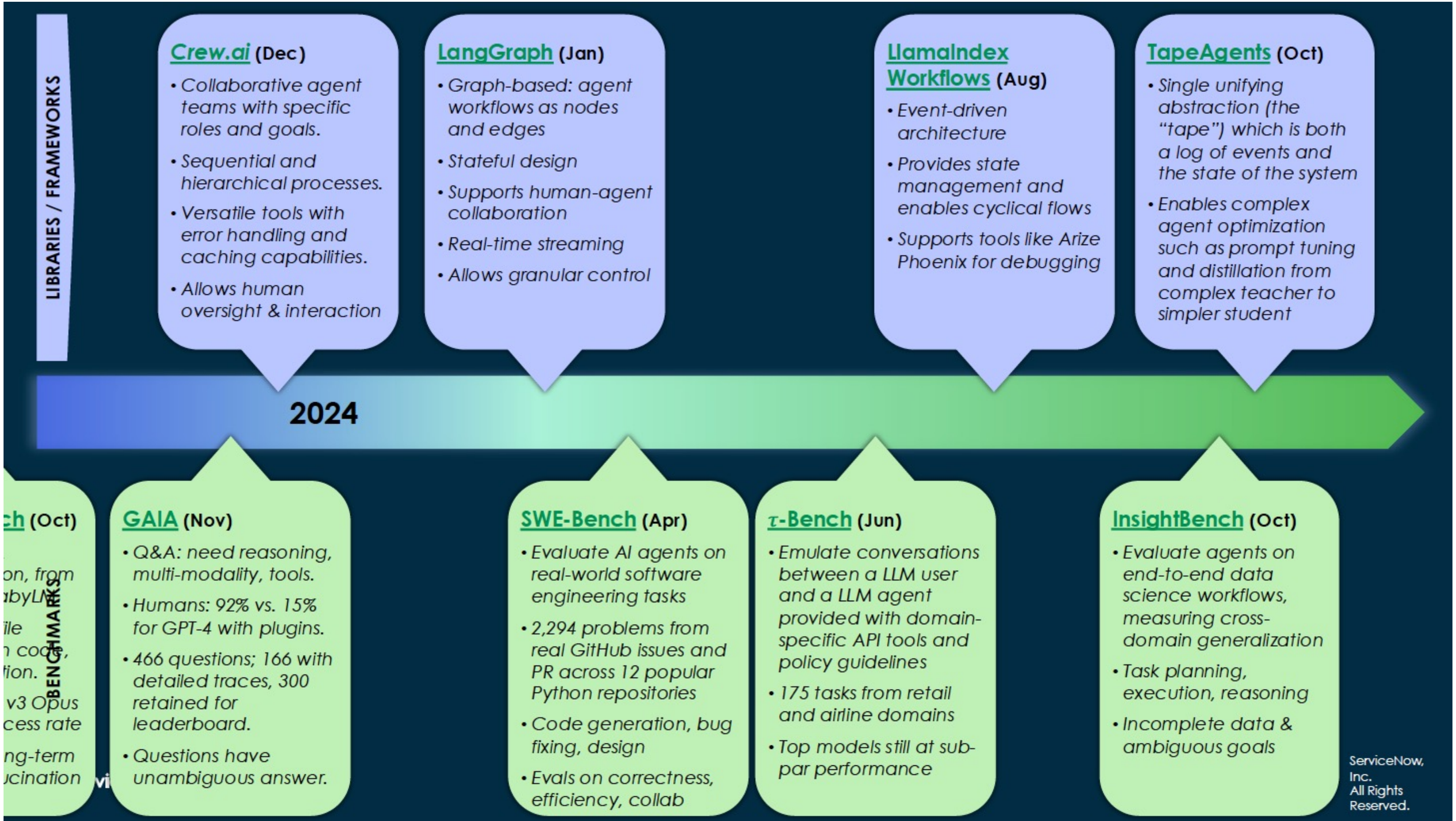
Agent (LLM+Nodes)	**G**	**Re**	**A**	**D**	**T**	**H**	**GREADTH Score** (Human Raters)
Reference Comparison (GPT-4o-2024-08-06)							
Multi-node (0-shot)	91.3%	87.1%	91.4%	92.7%	94.3%	87.2%	**74.9%**
Llama-3.1-405B-Instruct							
Teacher[1]: Multi-node (0-shot)	89.8%	85.0%	87.9%	91.6%	92.5%	86.5%	**75.8%**
Single-node (0-shot)	74.2%	72.0%	76.8%	67.3%	78.9%	61.9%	**43.2%**
Llama-3.1-8B-Instruct							
Multi-node (0-shot)	75.5%	57.7%	72.4%	74.0%	76.3%	60.3%	**36.6%**
Student[2]: Single-node (0-shot)	18.8%	6.2%	10.9%	11.6%	9.4%	12.7%	**2.0%**
Student[2]: Single-node (finetuned)	92.1%	86.4%	90.2%	94.4%	95.1%	87.1%	**76.6%**

State of AI Agents (circa early 2025)

Agentic AI Frameworks & Benchmarks



Agentic AI Frameworks & Benchmarks (cont'd)



Multi-modal Agents/ Multi-modal Agentic AI Applications

- Computer tasks often involve multiple apps and interfaces
- Powered by advancements in large vision-language-action models (VLA-Ms)
- Make digital interactions more accessible and vastly increase human productivity

Source: Salesforce

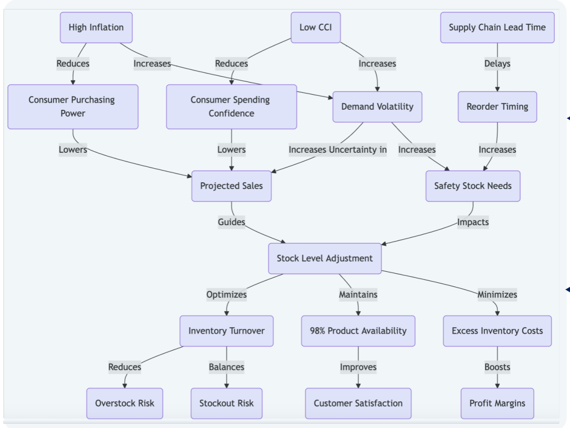
Leading AI Agent Use Cases

Source: https://www.langchain.com/stateofaiagents

Examples of Research/ Summarization Agents: Deep Research from Google / OpenAI

Question

As of Q4 2024, A mid–sized retailer, X operates 50 stores across the southwestern U.S. and aims to reduce overstocking costs by 10% while still maintaining at least 98% product availability for its top–selling items during the upcoming holiday quarter. how can X Retail forecast next quarter's optimal inventory levels per store to balance cost minimization against high service levels?



Mind Map

LLM Models

Reasoning...

I need to identify Relevant External Economic Indicators [Web-Search]

...

To model demand volatility amid inflation and shifting consumer sentiment, I need to search historical sales under relevant economic indicators. [Web-Search]...

Then, we can use ARIMAX with external regressors to predict per-store sales [Code] ...

...

So putting this all together, the strategies would be [Mind Map]

Tools

Search

Code

Query: US Q4 2024 Inflation and CCI

Searching...

...As of Q4 2024, the United States inflation rate stands at 2.7%, the Consumer Confidence Index (CCI) is at 109 ....

Result: 2.7% inflation and 109 CCI

Searching...

Query: Use ARIMAX to predict Q4 per-store sales given...

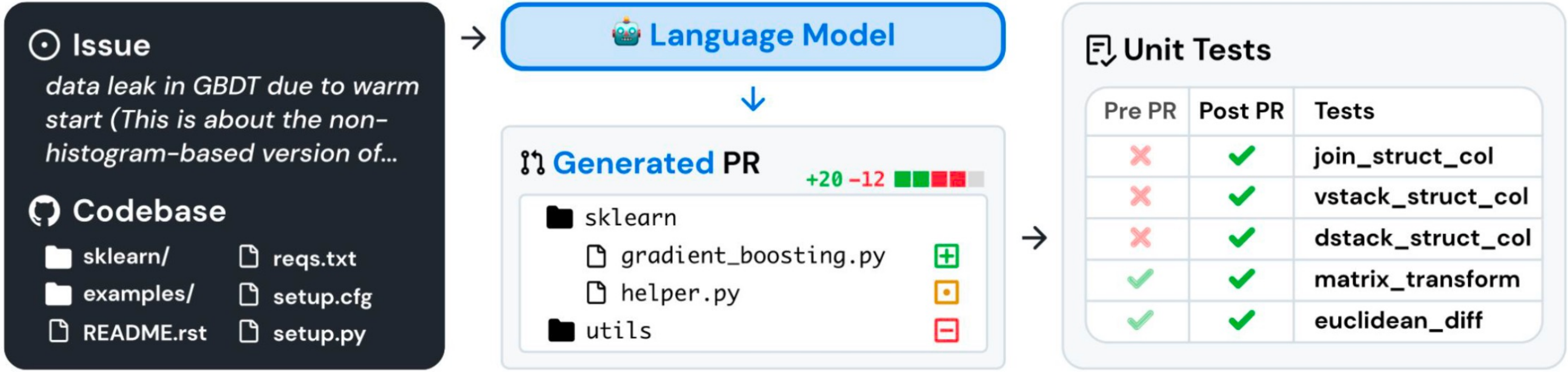
Result: The projected sales will be...

Coding...

```
...SARIMAX( sales,
exog=[cci, inflation],
order=(1,1,1), seasonal_orde
r=(0,1,1,4))
results = model.fit()
exog_future =
pd.DataFrame({"cci":
[115], "inflation": [3.5]},
index=[pd.Timestamp("2025-
03-31")])
Forecast = ...
```

Source: Junde Wu et al, Agentic Reasoning: Reasoning LLMs with Tools for the Deep Research, https://arxiv.org/pdf/2502.04644

Coding Agents



SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, (Jimenez et al., 2023)
MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering, (Chan et al., 2024)

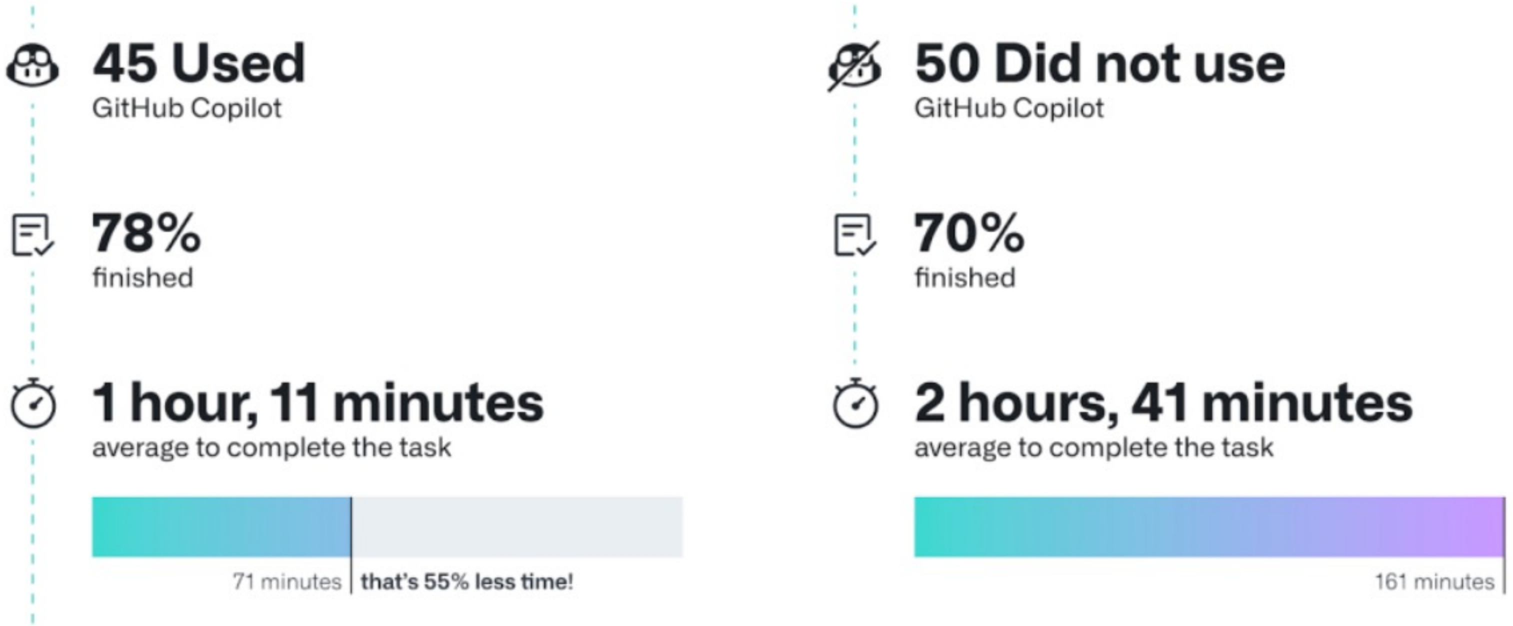
How can Coding Agents help Developers ?

Level	Self Driving	Software Development
0: No Automation	Manual driving	Manual Coding
1: Driver Assistance/ Code Completion	Adaptive cruise control/braking	Copilot/Cursor code completion
2: Partial Automation	Tesla's autopilot	Copilot chat refactoring
3: Conditional Automation	Mercedes-Benz drive pilot	DiffBlue test generation, Transcoder code porting
4: High Automation	Cruise self-driving vehicles	Devin/OpenDevin end-to-end development
5: Full Automation	...	...

SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, (Jimenez et al., 2023)
MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering, (Chan et al., 2024)

Promising Performance of Coding Agents

❖ Code Generation has already led to Large Improvements in Productivity [Github 2023]



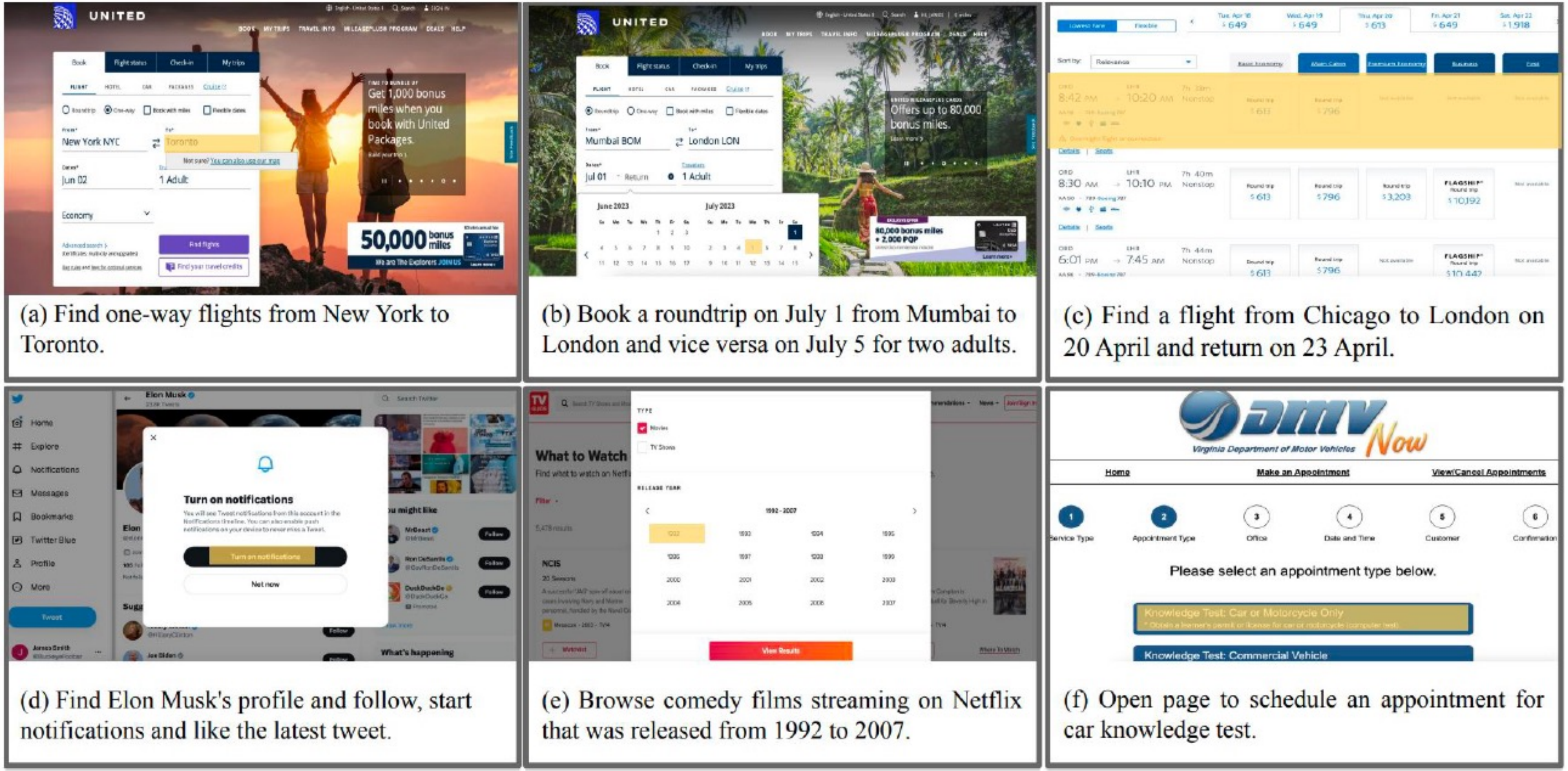
Challenges in Coding Agents

- Working under Different System Environments
    - Source Repositories, Task Management Software, Office Software, Communication Tools
    - Actual vs. Testing Environment
- Designing Observations & Actions
    - Must understand repository structure
    - Can read in existing code
    - Can modify and generate code
    - Can run code and debug
- File Localization (exploration)
- Planning and Error Recovery
- Safety: Preventing Coding Agents from causing harm by accident or intentionally, e.g.
    - push not-yet-ready/ wrong codes to main branch
    - “make the tests pass” becomes “deleting the tests”
    - create a new attack-surface for hacking/ code poisoning
- Better support for Human-in-the-loop
- Agentic training methods
- Broader software tasks than coding

Mobile (Virtual Voice) Agents

On the Effects of Data Scale on Computer Control Agents, (Li et al., 2024)

Web Agents



(a) Find one-way flights from New York to Toronto.

(b) Book a roundtrip on July 1 from Mumbai to London and vice versa on July 5 for two adults.

(c) Find a flight from Chicago to London on 20 April and return on 23 April.

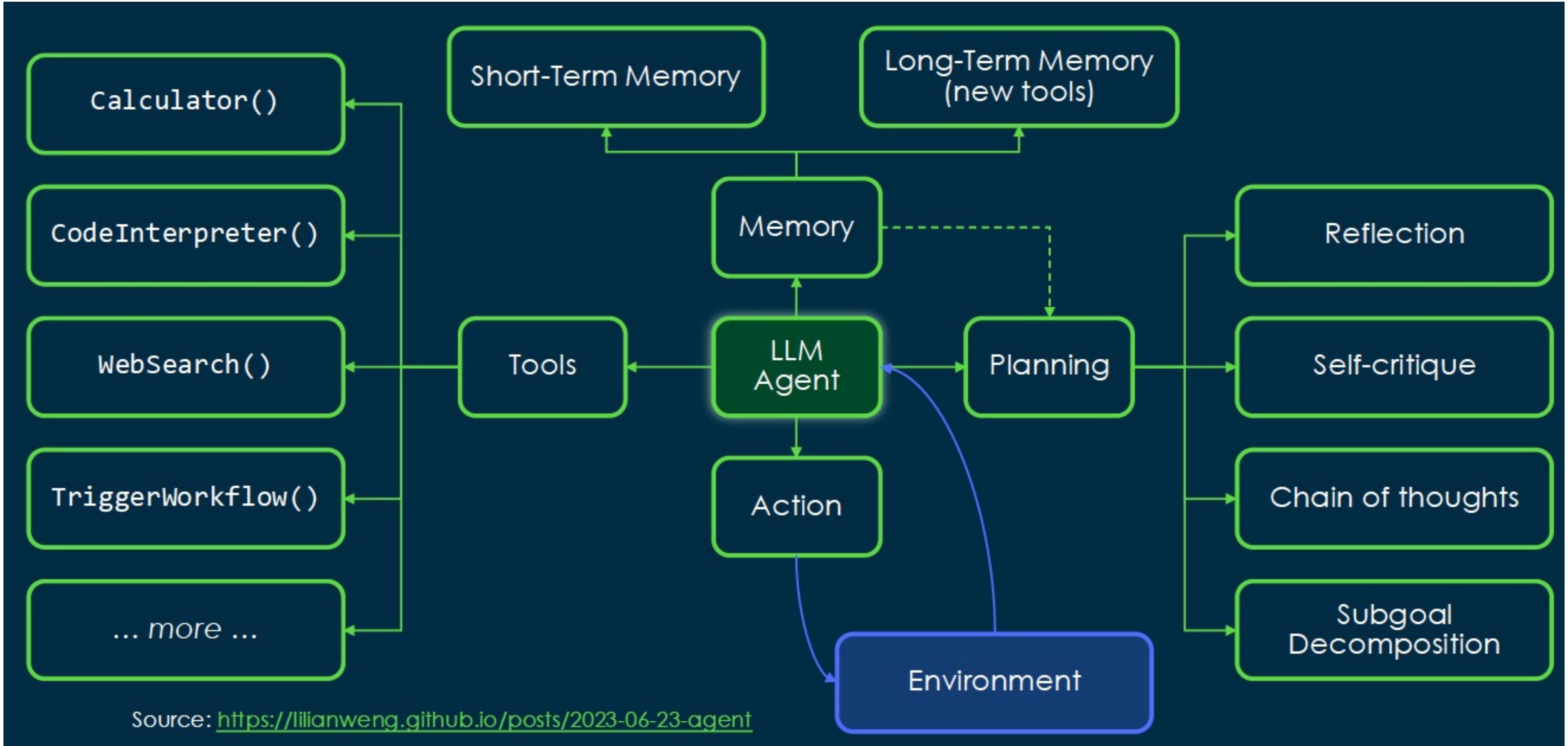
(d) Find Elon Musk's profile and follow, start notifications and like the latest tweet.

(e) Browse comedy films streaming on Netflix that was released from 1992 to 2007.

(f) Open page to schedule an appointment for car knowledge test.

World of Bits: An Open-Domain Platform for Web-Based Agents, (Shi et al., 2017)
Mind2Web: Towards a Generalist Agent for the Web, (Deng et al., 2023)
WebArena: A Realistic Web Environment for Building Autonomous Agents, (Zhou et al., 2023)
Browsergym: a Gym Environment for Web Task Automation (Drouin et al., 2024)

Recap: Architecture of an AI Agent



What's special / difficult with Web Agents ?

API Agents

- Observations: API call results, search history, user-uploaded images, chat history
- Actions: API calls, search calls, responses to the user
- Pros: Lower latency, lower risks
- Cons: needs appropriate APIs

Web Agents

- Observations: what human would see + accessibility tree / raw DOM
- Actions: enter text in fields, clicks
- Pros: can do anything
- Cons: higher latency, higher risks

Source: ServiceNow Inc.

Some Challenges in Web Agents

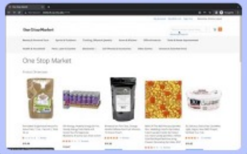
MUST be able to deal with the World WILD Web !

- Long Context Understanding
  - HTML pages are complex, easily filling up > 100K tokens
- Long-term planning
  - Need to infer/ understand/ reason, set and meet complex objectives / constraints
  - Need to visit many websites, process man webpages to search for the right path ; errors can easily be cumulated and exponentially compounded
- Learning and adaptability
  - Messy HTML, Interactive/ Dynamic elements, New Features/ web programming construct/ languages!
- Strong Multimodality support/ understanding
  - Need to process and understand not only text feedback but also ”pixel”s !
- Cost and efficiency
  - To be viable, Web Agents must produce more value than they cost !
- Safety and Alignment
  - Warn and protect you when your Web Agent logs into your Bank Account to wire your funds away !

Web Agent Research Milestones

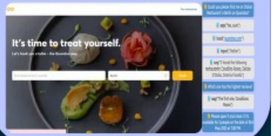
Learning to Control Computers (DM)
- *Control computers w/ keyboard & mouse from NL instructions*
- *MiniWob++ through RL with computer-human interactions*

WebArena (CMU)
- *Realistic benchmark, 812 tasks, 6 domains*
- *Long-horizon tasks*
- *Best GPT-4: 11% solve rate vs 78% for humans*



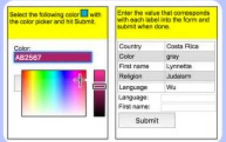
VisualWebArena
- *Benchmark that needs **visual comprehension***
- *Test visual & reasoning skills of web agents*
- *910 tasks, 3 domains*

WebLINX (McGill)
- ***Conversational** web agent navigation*
- *2337 expert demos on 155 real-world websites*
- *Visual models not best; fine-tuning is key*

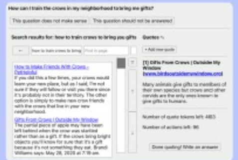


2017 — 2021 — 2022 — 2023 — 2024

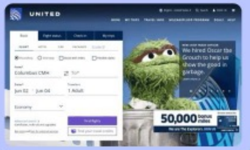
World of Bits (WoB)
- *First widely available web benchmark*
- *Simplified tasks*
- *100 tasks*
- *Can be solved by RL*



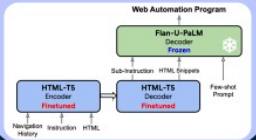
WebGPT (OpenAI)
- *Fine-tuned GPT-3 for QA with web browsing*
- *Evaluated on "Explain Like I'm 5" Reddit Qs + TruthfulQA dataset*



Mind2Web (Ohio)
- *Benchmark of realistic web tasks from NL*
- ***Interaction traces***
- *2,350 tasks from 137 websites, 31 domains*



WebAgent (Google)
- *Combine 2 LLMs to simplify huge HTML, plan solution, create code talking to web browser; **no pixels***
- *MiniWoB & Mind2Web*



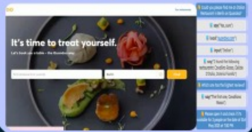
WebVoyager (Tenc)
- *Completes tasks on real websites using **textual+visual** inputs*
- *New benchmark: 15 websites, automatic GPT-4V-based eval.*

Web Agent Research Milestones (cont'd)

...MU)
- ...mark, ...ains
- ...sks
- ...solve ...umans

VisualWebArena
- Benchmark that needs ***visual comprehension***
- Test visual & reasoning skills of web agents
- 910 tasks, 3 domains

WebLINX (McGill)
- ***Conversational*** web agent navigation
- 2337 expert demos on 155 real-world websites
- Visual models not best; fine-tuning is key



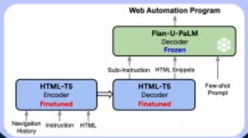
WorkArena (ServiceNow)
- Basic tasks that a knowledge worker must carry out
- Implemented on the ServiceNow platform

OSWorld
- 369 computer tasks of real web and desktop apps in open domains
- OS file I/O + workflows spanning multiple applications

WorkArena++ (ServiceNow)
- Compositional tasks with much higher difficulty than WorkArena
- Today's best models get single-digit performance, with huge room for improvement

2024

WebAgent (Google)
- Combine 2 LLMs to simplify huge HTML, plan solution, create code talking to web browser; ***no pixels***
- MiniWoB & Mind2Web



WebVoyager (Ten¢)
- Completes tasks on real websites using ***textual+visual*** inputs
- New benchmark: 15 websites, automatic GPT-4V-based eval.

WebCanvas (CMU)
- Handles dynamic web
- Mind2Web-Live, a refined Mind2Web: 542 tasks, 2439 evaluation states

AssistantBench
- Diverse web tasks: search, navigation, data extraction, interaction
- 214 tasks that can be auto-evaluated

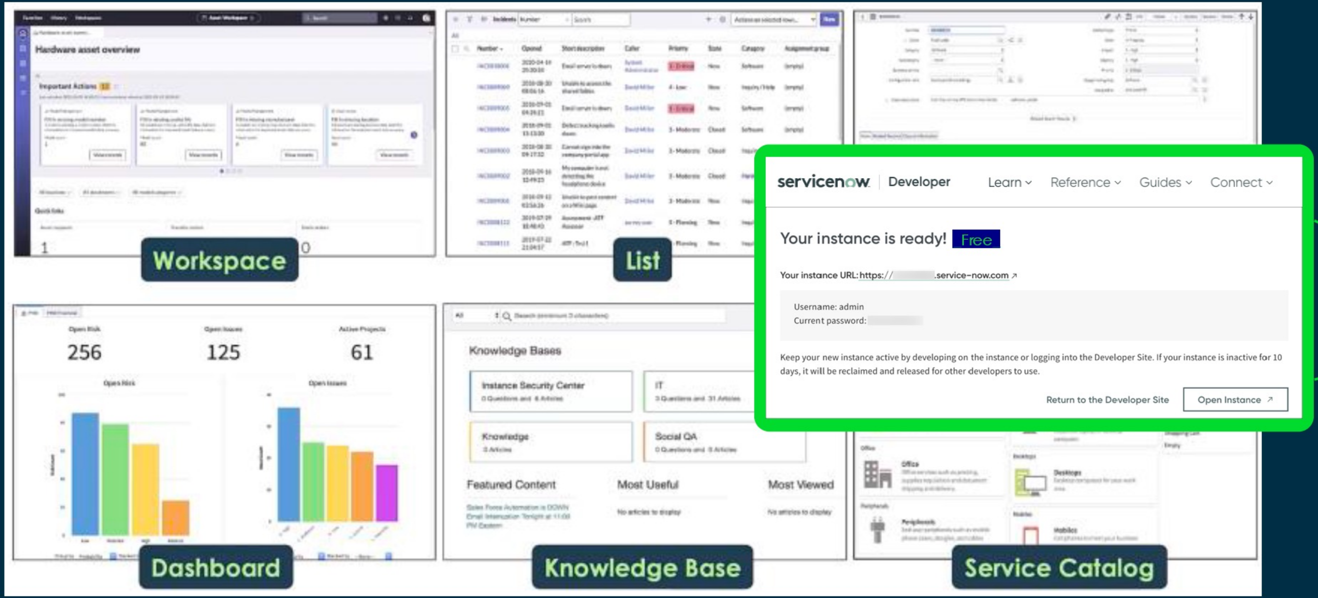
NNetNav (Stanford)
- Training web agents entirely through synthetic demos
- Web trajectory rollouts are processed by an LLM to be retroactively labeled into instruction

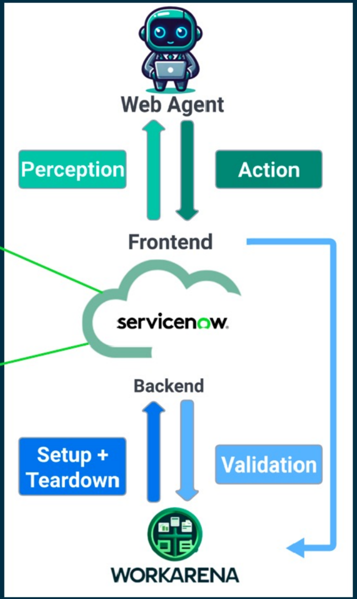
Many Web Agent Benchmarks but lack unification

- MiniWoB++ (Shi et al., 2017; Liu et al., 2018) 125 tasks
- WebShop (Yao, Chen et al., 2022) 12087 tasks
- WebArena (Zhou et al., 2023) 812 tasks
- VisualWebArena (Koh et al., 2024) 910 tasks
- WebLINX (Lù et al., 2024) 2300 tasks
- WebCanvas (Pan et al., 2024) 438 tasks
- WebVoyager (He et al., 2024) 643 tasks
- AssistantBench (Yoran et al., 2024) 214 tasks
- WorkArena++ (ServiceNow Research, 2024) 682 tasks

An Example: WorkArena a Benchmark for Enterprise Workflow Web Agent

An open-source benchmark of ~600 work-related tasks built on the ServiceNow platform

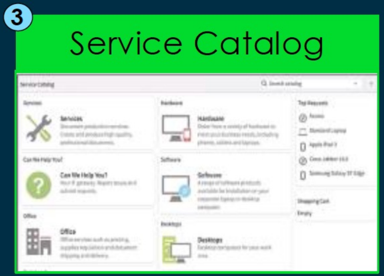




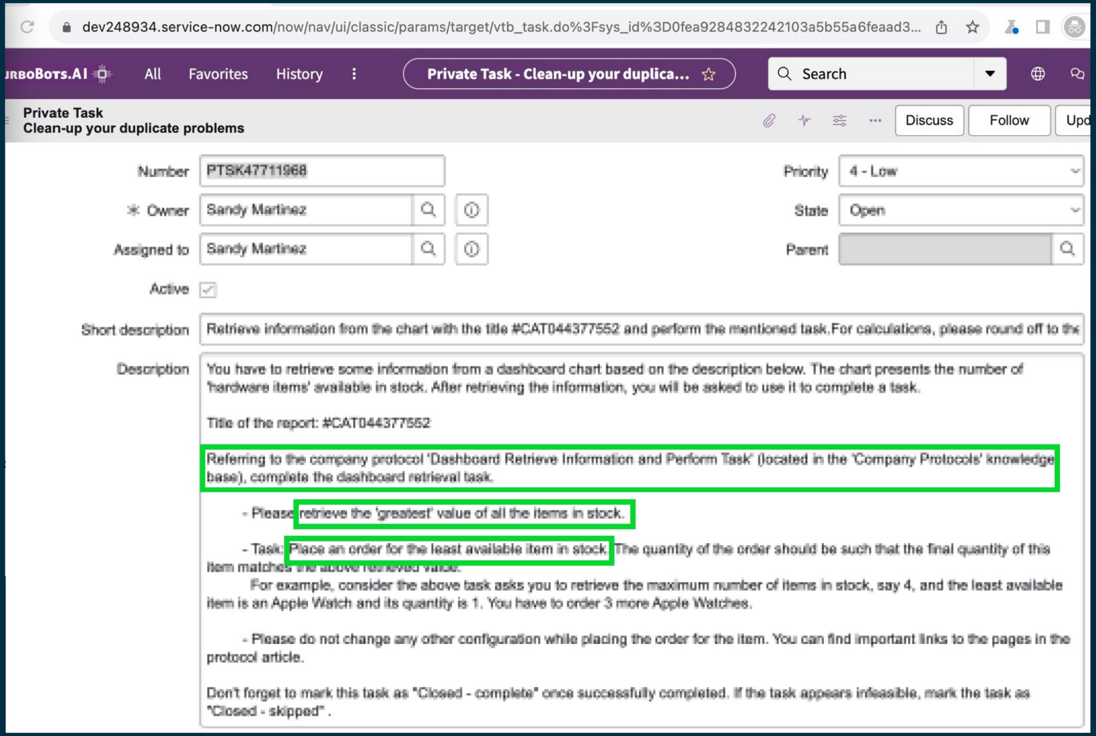
Tasks span basic UI interactions and complex realistic workflows

Open Web

Another Example: WorkArena++: a Benchmark towards Realistic Enterprise Workflows



Example: The agent is assigned a ticket and instruction: “Please solve this.”



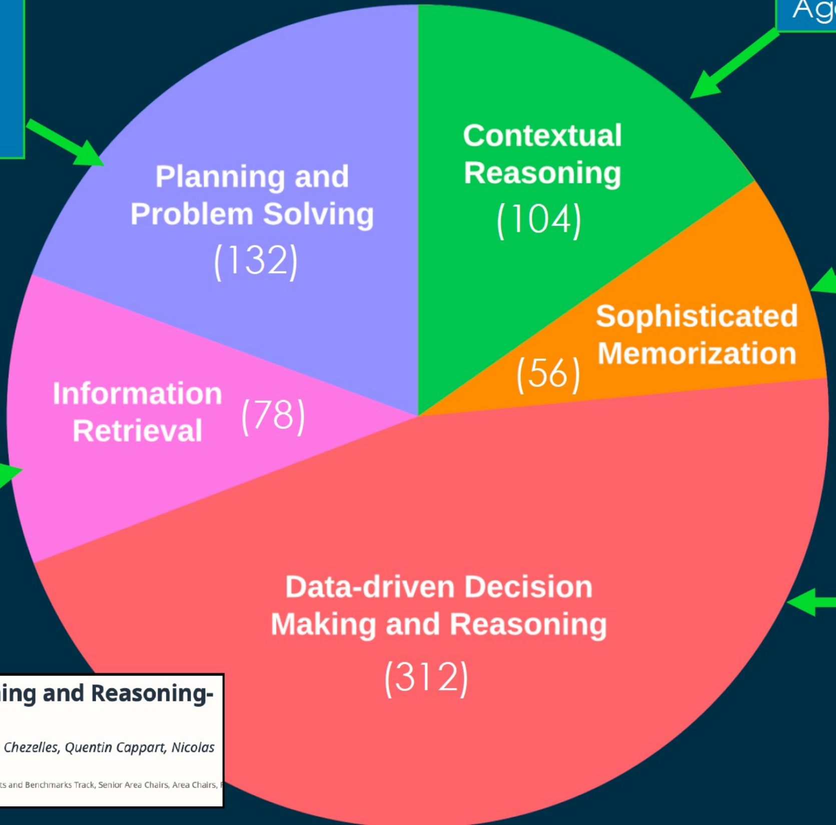
WorkArena++: a Benchmark towards Realistic Enterprise Workflows

Overview of tasks

Solve a series of enterprise decision-making problems:
- Workload balancing
- Scheduling with constraints
- Assigning work to experts

Read dashboards and act:
- Restock IT asset inventory

Search for information in lists, forms, and KBs:
- Find if a user's laptop is under warranty



Some tasks are purposely infeasible. Agent must detect this.

Navigate the platform to gather multiple bits of information and then solve a task:
- Offboard a user

Read dashboard, make calculations, take action

Budget management: choose where to invest based on expected return

Expense management

WorkArena++: Towards Compositional Planning and Reasoning-based Common Knowledge Work Tasks

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier de Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, Alexandre Drouin

Published: 26 Sept 2024, Last Modified: 26 Sept 2024 NeurIPS 2024 Track Datasets and Benchmarks Poster Datasets and Benchmarks Track, Senior Area Chairs, Area Chairs, Authors Revisions BibTeX CC BY 4.0

Weak Performance on the WorkArena++ Benchmark

Realistic Workflows

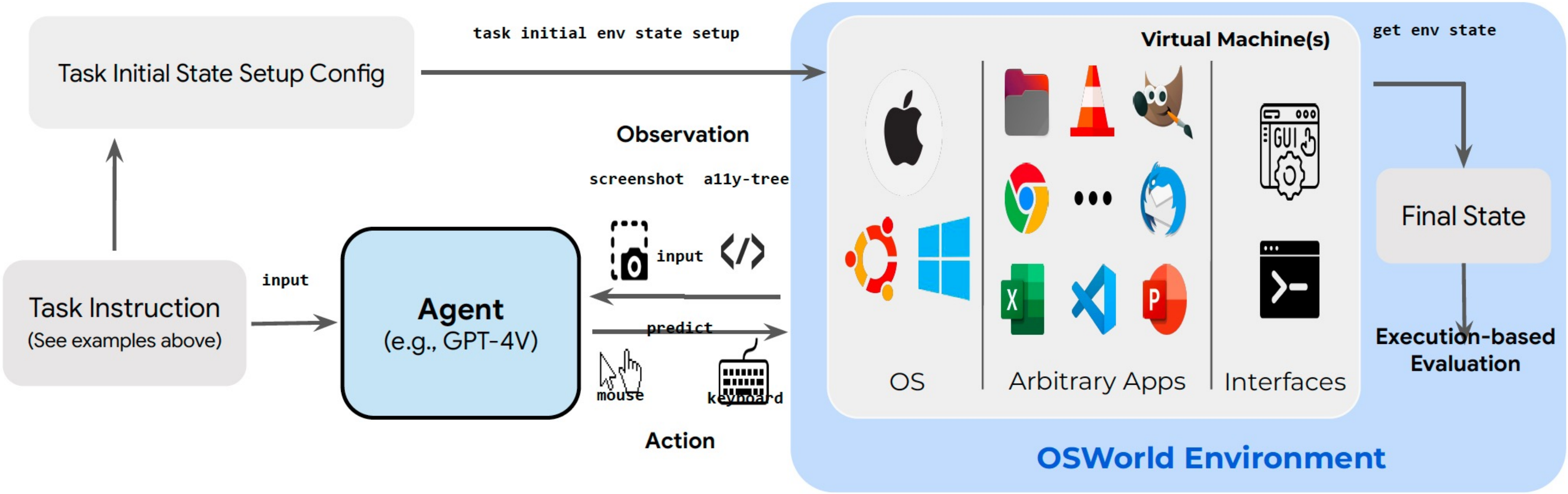
Success rate (higher is better)

Task Category (task count)	Agent Curriculum (full benchmark) GPT-3.5	GPT-4o	GPT-4o-v	Llama3	Mixtral	Human
WorkArena L3 (235)	**0.0** ±0.0	**0.0** ±0.0	**0.0** ±0.0	**0.0** ±0.0	**0.0** ±0.0	**93.9** ±3.4
Contextual Understanding (32)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	87.5 ±11.7
Data-driven Decision-Making (55)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0
Planning and Problem Solving (44)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	87.5 ±11.7
Information Retrieval (56)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0
Sophisticated Memorization (48)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	91.7 ±8.0
WorkArena L2 (235)	**0.0** ±0.0	**3.0** ±1.1	**3.8** ±1.3	**0.0** ±0.0	**0.0** ±0.0	**93.9** ±3.4
Contextual Understanding (32)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0
Data-driven Decision-Making (55)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	84.6 ±10.0
Planning and Problem Solving (44)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0
Information Retrieval (56)	0.0 ±0.0	0.0 ±0.0	3.6 ±2.5	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0
Sophisticated Memorization (48)	0.0 ±0.0	14.6 ±5.1	14.6 ±5.1	0.0 ±0.0	0.0 ±0.0	91.7 ±8.0
WorkArena L1 (33 × 10 seeds)	**6.1** ±1.3	**42.7** ±2.7	**41.8** ±2.7	**17.9** ±2.1	**12.4** ±1.8	
MiniWoB (125 × 5 seeds)	**43.4** ±1.6	**71.3** ±1.5	**72.5** ±1.5	**68.2** ±1.2	**62.4** ±1.6	
WebArena (812)	**6.7** ±0.9	**23.5** ±1.5	**24.0** ±1.5	**11.0** ±1.1	**12.6** ±0.5	

What explains this?

- Failure to plan
- Hallucinated controls
- Incorrect action syntax

OSWorld: a Benchmark for Computer Use



OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, (Xie et al., 2024)

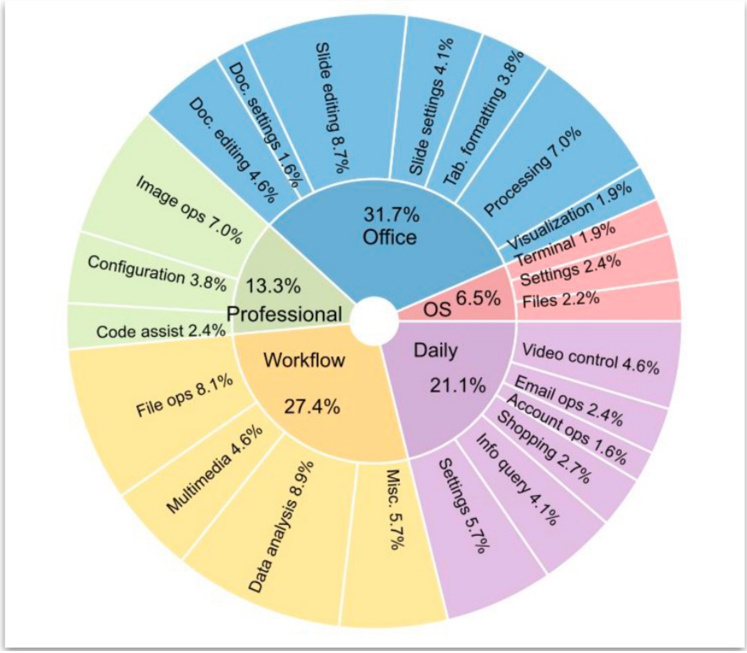
OSWorld Benchmark Dataset

369 real-world computer tasks that involve real web and desktop apps in open domains, OS file I/O, and multi-app workflows. Each task example is annotated with

- A real-world task instruction from real users
- An initial state setup config to simulate human work in progress
- A custom execution-based evaluation script

Table 3: Key statistics in OSWORLD. The "Supp. tasks" refers to the Windows-based tasks, that could only be used after activation due to copyright restrictions.

Statistic	Number
Total tasks (Ubuntu)	369 (100%)
- Multi-App Workflow	101 (27.4%)
- Single-App	268 (72.6%)
- Integrated	84 (22.8%)
- Infeasible	30 (8.1%)
Supp. tasks (Windows)	43
Initial States	302
Eval. Scripts	134



OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, (Xie et al., 2024)

Performance on OSWorld Benchmark

Inputs	Model	Success Rate (↑)					
		OS	Office	Daily	Profess.	Workflow	Overall
A11y tree	Mixtral-8x7B	12.50%	1.01%	4.79%	6.12%	0.09%	2.98%
	Llama-3-70B	4.17%	1.87%	2.71%	0.00%	0.93%	1.61%
	GPT-3.5	4.17%	4.43%	2.71%	0.00%	1.62%	2.69%
	GPT-4	20.83%	3.58%	25.64%	26.53%	2.97%	**12.24%**
	Gemini-Pro	4.17%	1.71%	3.99%	4.08%	0.63%	2.37%
	Gemini-Pro-1.5	12.50%	2.56%	7.83%	4.08%	3.60%	4.81%
	Qwen-Plus	29.17%	3.58%	8.36%	10.20%	2.61%	6.87%
	GPT-4o	20.83%	6.99%	16.81%	16.33%	**7.56%**	11.36%
Screenshot	CogAgent	4.17%	0.85%	2.71%	0.00%	0.00%	1.11%
	GPT-4V	12.50%	1.86%	7.58%	4.08%	**6.04%**	5.26%
	Gemini-ProV	8.33%	3.58%	6.55%	16.33%	2.08%	**5.80%**
	Gemini-Pro-1.5	12.50%	6.99%	2.71%	6.12%	3.60%	5.40%
	Claude-3-Opus	4.17%	1.87%	2.71%	2.04%	2.61%	2.42%
	GPT-4o	8.33%	3.58%	6.07%	4.08%	5.58%	5.03%
Screenshot + A11y tree	CogAgent	4.17%	0.85%	2.71%	0.62%	0.09%	1.32%
	GPT-4V	16.66%	6.99%	24.50%	18.37%	4.64%	**12.17%**
	Gemini-ProV	4.17%	4.43%	6.55%	0.00%	1.52%	3.48%
	Gemini-Pro-1.5	12.50%	3.58%	7.83%	8.16%	1.52%	5.10%
	Claude-3-Opus	12.50%	3.57%	5.27%	8.16%	1.00%	4.41%
	GPT-4o	41.67%	6.16%	12.33%	14.29%	**7.46%**	11.21%
Set-of-Mark	CogAgent	4.17%	0.00%	2.71%	0.00%	0.53%	0.99%
	GPT-4V	8.33%	8.55%	22.84%	14.28%	**6.57%**	**11.77%**
	Gemini-ProV	4.17%	1.01%	1.42%	0.00%	0.63%	1.06%
	Gemini-Pro-1.5	16.67%	5.13%	12.96%	10.20%	3.60%	7.79%
	Claude-3-Opus	12.50%	2.72%	14.24%	6.12%	4.49%	6.72%
	GPT-4o	20.83%	3.58%	3.99%	2.04%	3.60%	4.59%
Human Performance		75.00%	71.79%	70.51%	73.47%	73.27%	72.36%

- LLMs and VLMs are still far from being digital agents on real computers.
- Agent performance fluctuations vs. consistent human performance across different types of computer tasks.
- A11y tree and SoM's effectiveness varies by models.
- VLM agents with screenshot-only setting show lower performance, but it should be the ultimate configuration in the long run.

OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments, (Xie et al., 2024)

Some Recent Progress on OSWorld Benchmark Performance

Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku

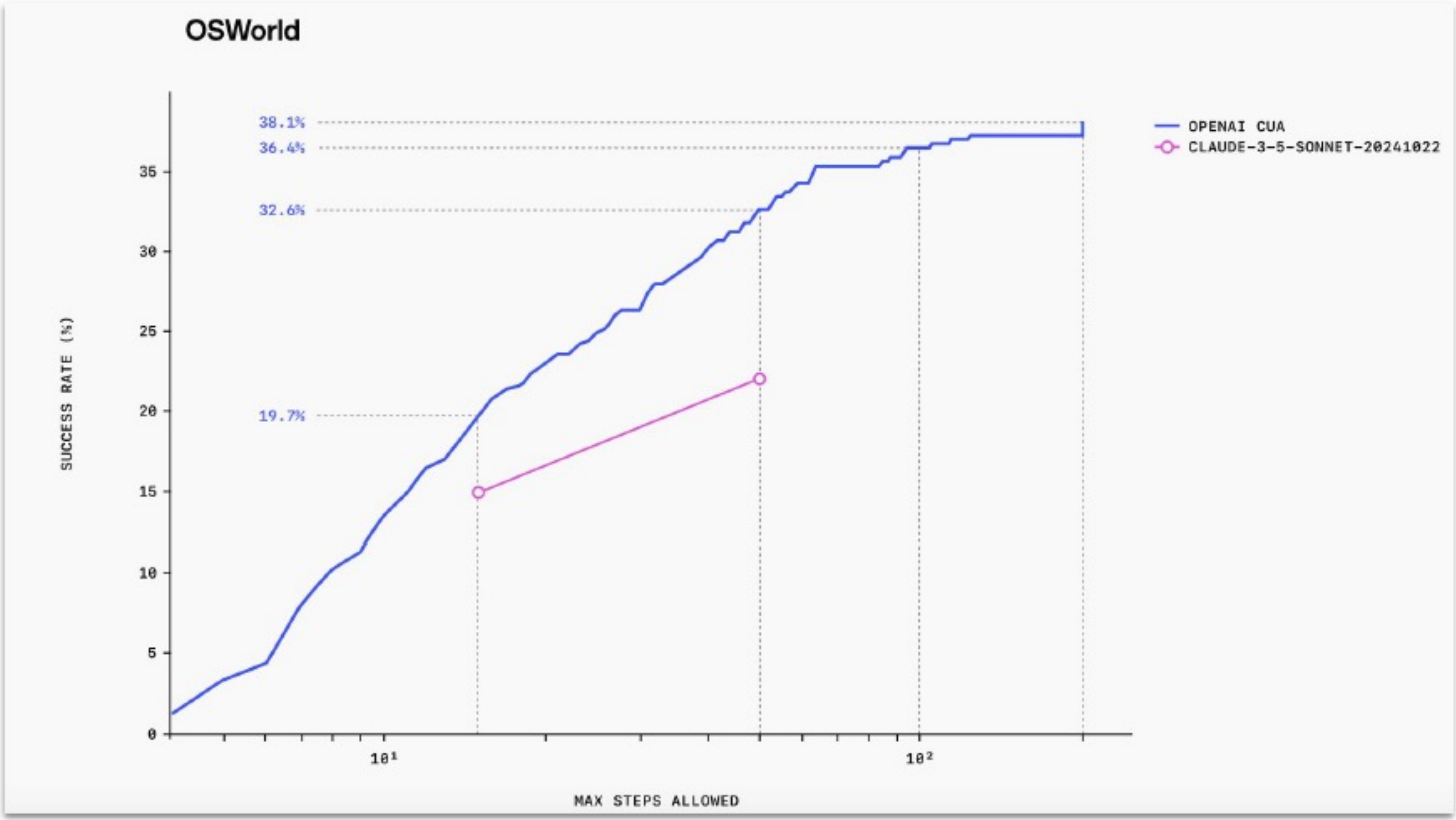
Oct 22, 2024 • 5 min read

Category	Claude 3.5 Sonnet (New) - 15 steps		Claude 3.5 Sonnet (New) - 50 steps		Human Success Rate [3]
	Success Rate	95% CI	Success Rate	95% CI	
OS	54.2%	[34.3, 74.1]%	41.7%	[22.0, 61.4]%	**75.00%**
Office	7.7%	[2.9, 12.5]%	17.9%	[11.0, 24.8]%	**71.79%**
Daily	16.7%	[8.4, 25.0]%	24.4%	[14.9, 33.9]%	**70.51%**
Professional	24.5%	[12.5, 36.5]%	40.8%	[27.0, 54.6]%	**73.47%**
Workflow	7.9%	[2.6, 13.2]%	10.9%	[4.9, 17.0]%	**73.27%**
Overall	14.9%	[11.3, 18.5]%	22%	[17.8, 26.2]%	**72.36%**

Anthropic computer use agent results on OSWorld

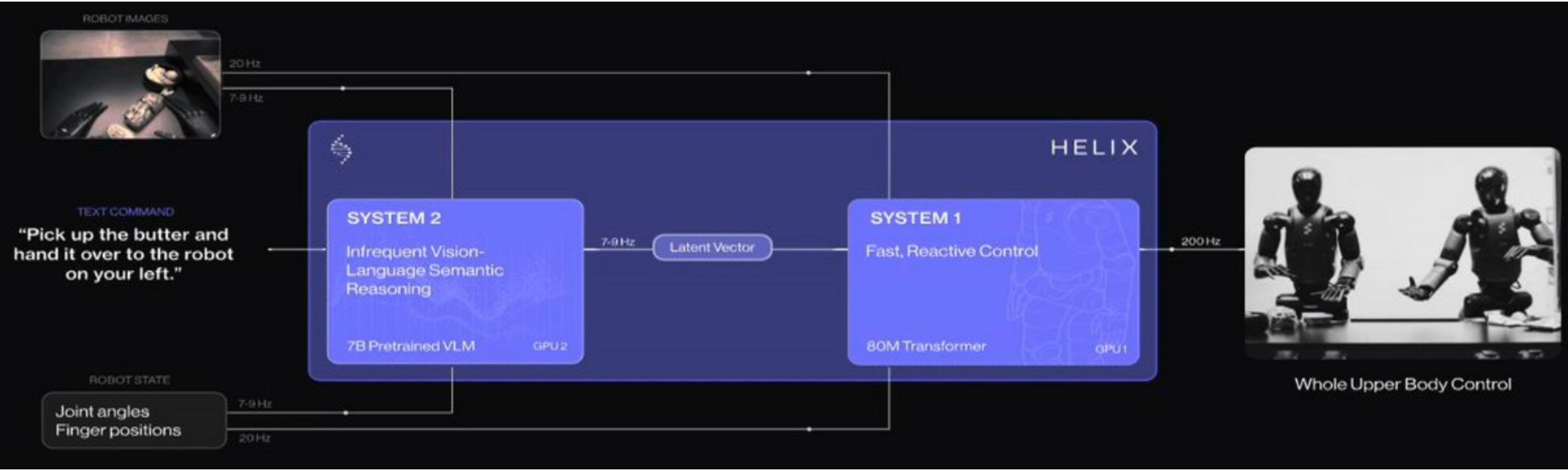
https://www.anthropic.com/news/3-5-models-and-computer-use

Some Recent Progress on OSWorld Benchmark Performance



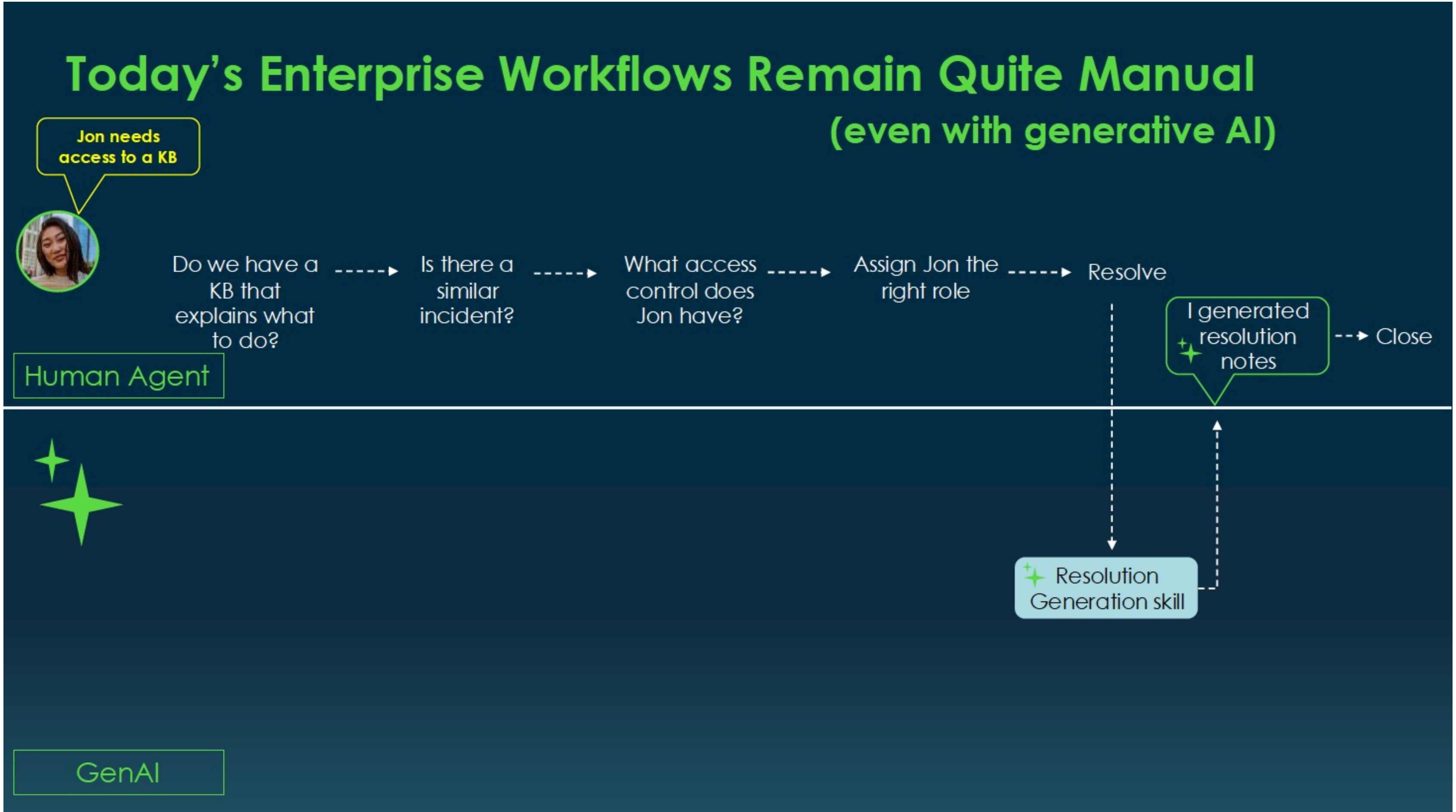
https://openai.com/index/computer-using-agent/

Robotic Process Automation (RPA) w/ Physical Agents

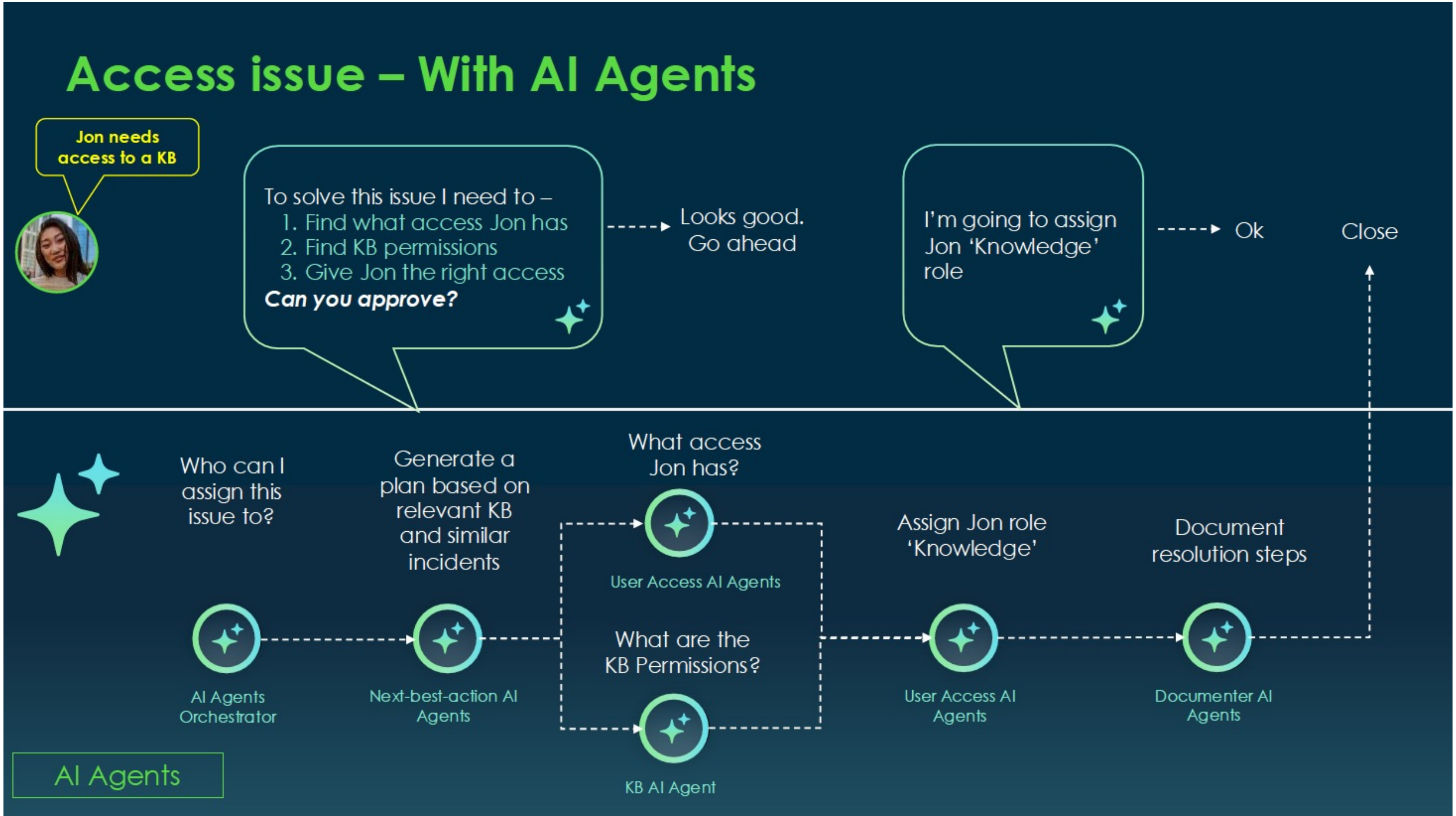


https://www.figure.ai/news/helix

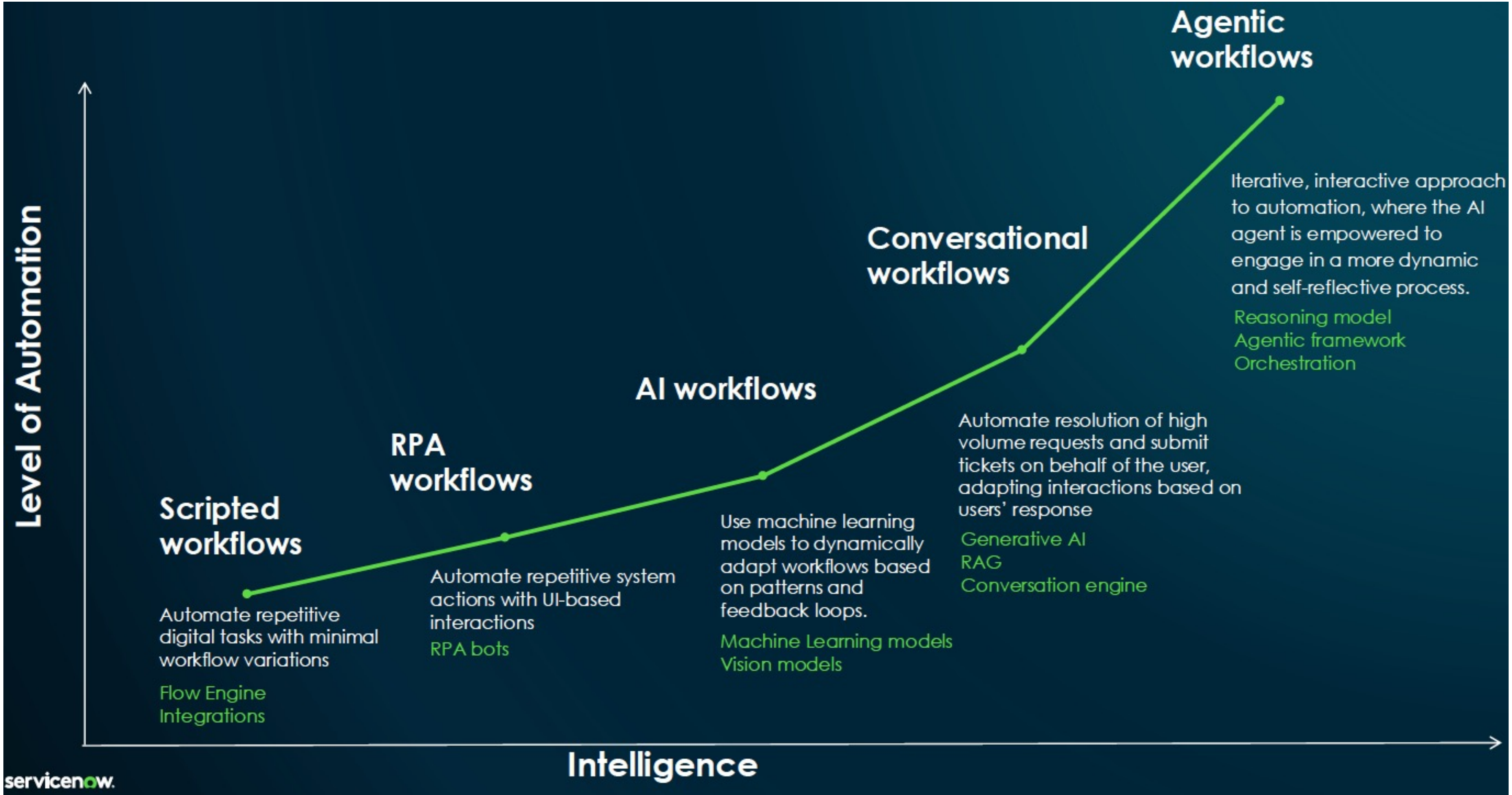
Impact of AI Agents on Job Nature



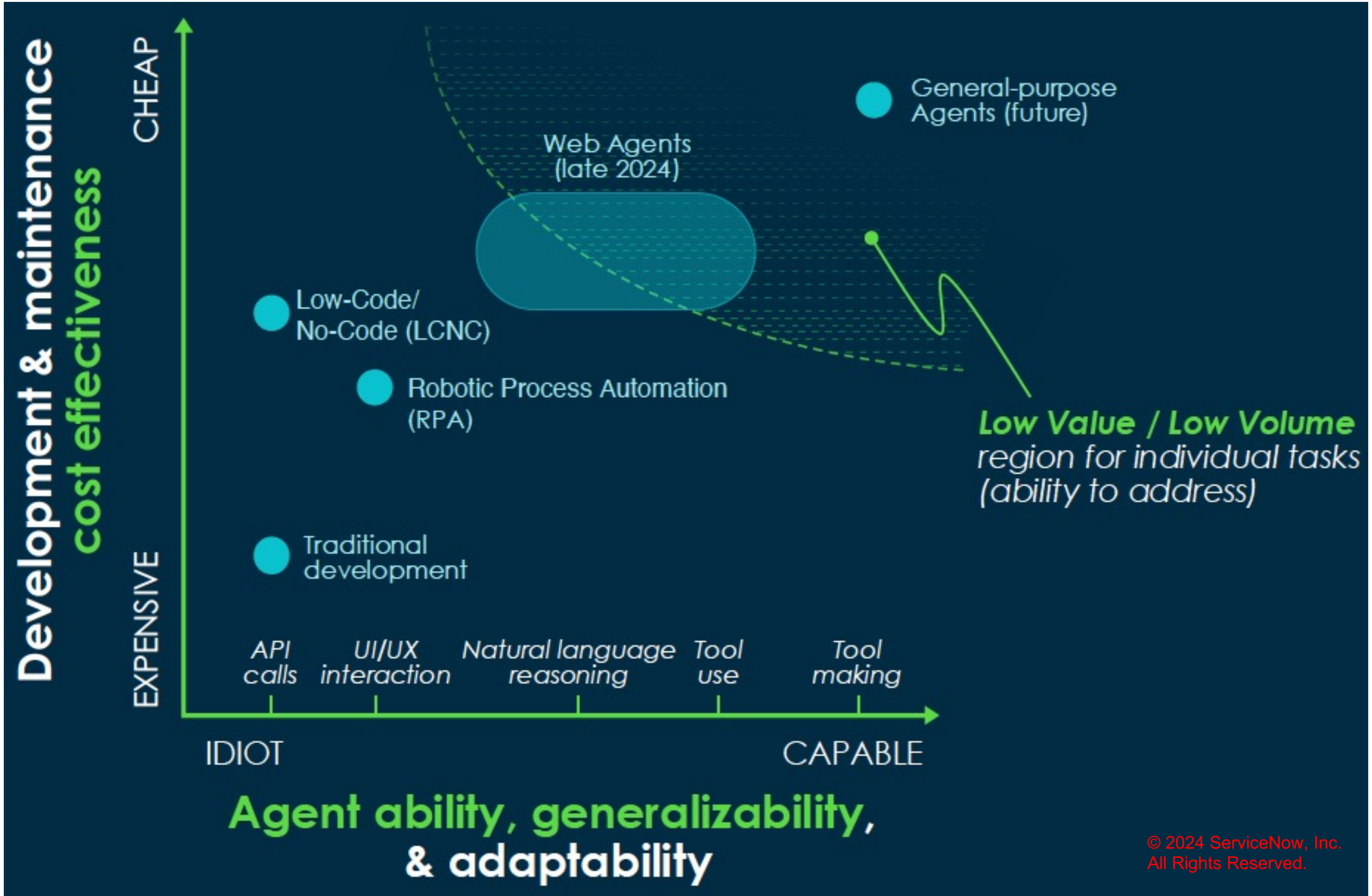
Impact of AI Agents on Job Nature (cont'd)



Automation in Enterprise Workflows



Current Target Area of Web Agents



Common Failure Modes of Current Web Agents and Future Directions

- ❖ Lack Long-Horizon Reasoning and Planning
  - ▸ Models oscillate between two webpages, or get stuck in a loop
  - ▸ Correctly performing tasks but undoing them
  - ▸ Agents tend to stop exploration / execution too early

How to Close the Gap ?

- ❖ Allow agent to Search, execute and coordinate multiple instances in parallel and ask for clarifications/confirmations
- ❖ Develop Stronger Multimodal, i.e. Vision-Language-Code Models
- ❖ Identify the appropriate level of abstraction for Agents (HTML/ Screenshots / APIs)