Introduction to Agents

Yogesh Haribhau Kulkarni

YHK

Outline

YHK

Introduction

Future AI?

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

YHK

Introduction to AI Agents

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

YHK

What is an Agent?

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

YHK

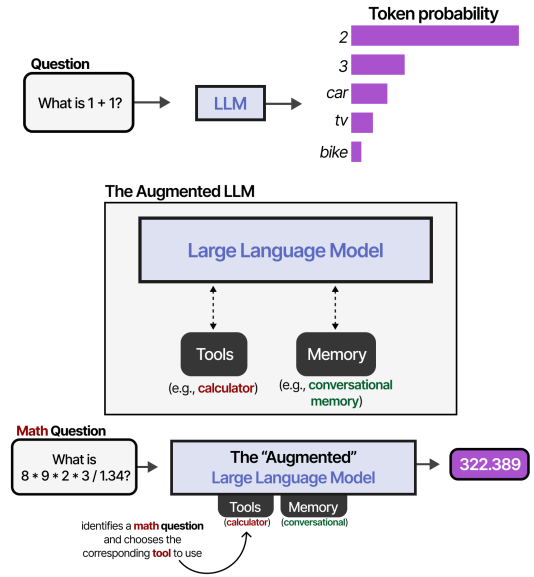
What are Agents?

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

YHK

What Are LLM Agents?

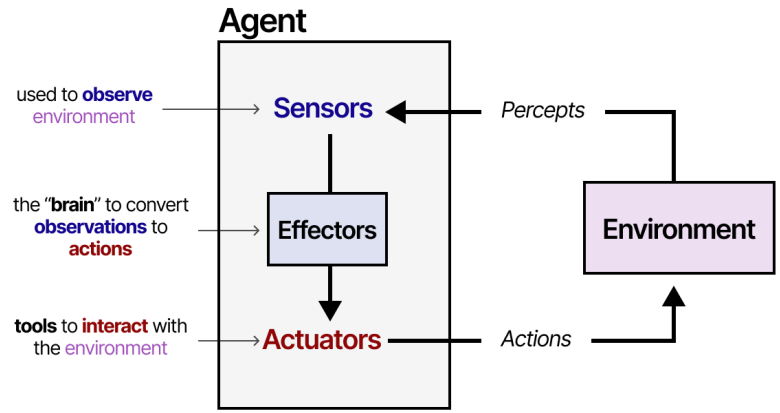
- LLMs predict the next token in a sequence—no memory.
- Conversations are simulated by sampling many tokens.
- LLMs struggle with tasks like basic arithmetic.
- Augmented LLMs use tools and memory to enhance capabilities.
- An Agent perceives and acts upon its environment.
- Agentic LLMs use text as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



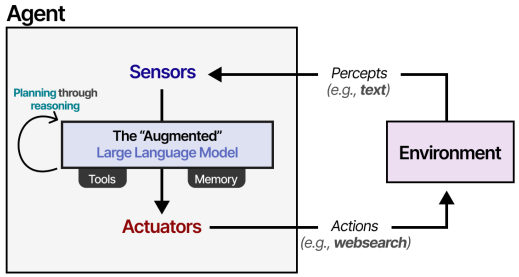
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?



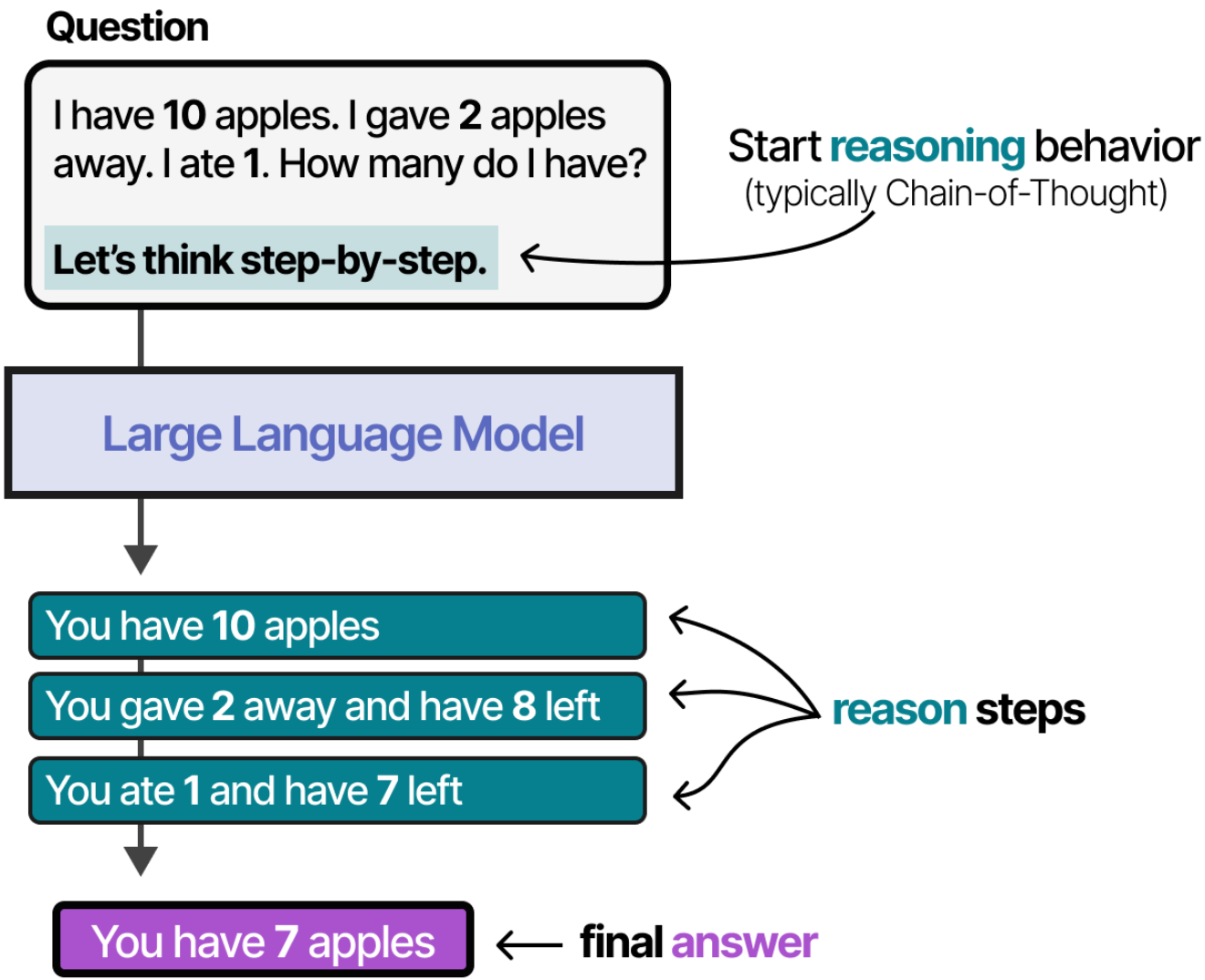
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)



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

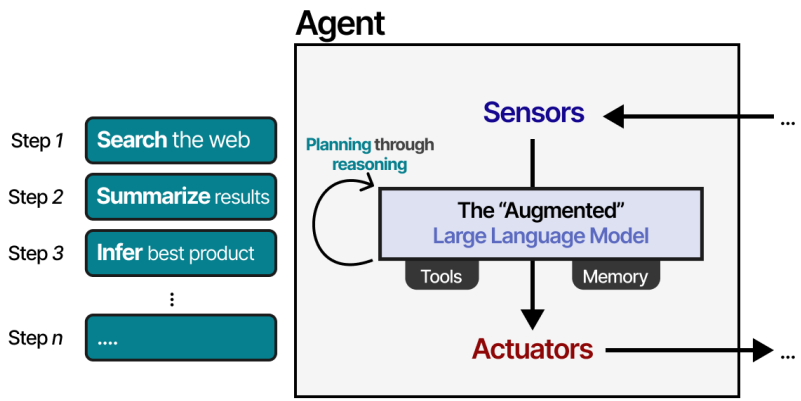
YHK

What Even Is an AI Agent?

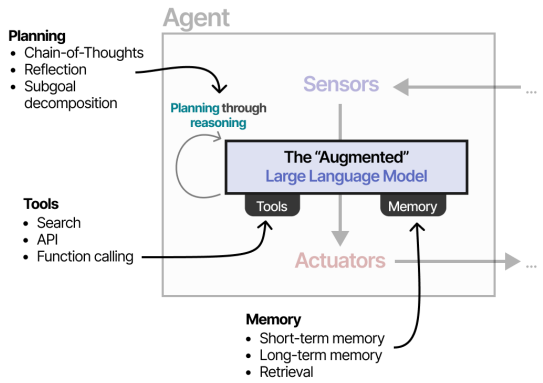


(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

What Even Is an AI Agent?

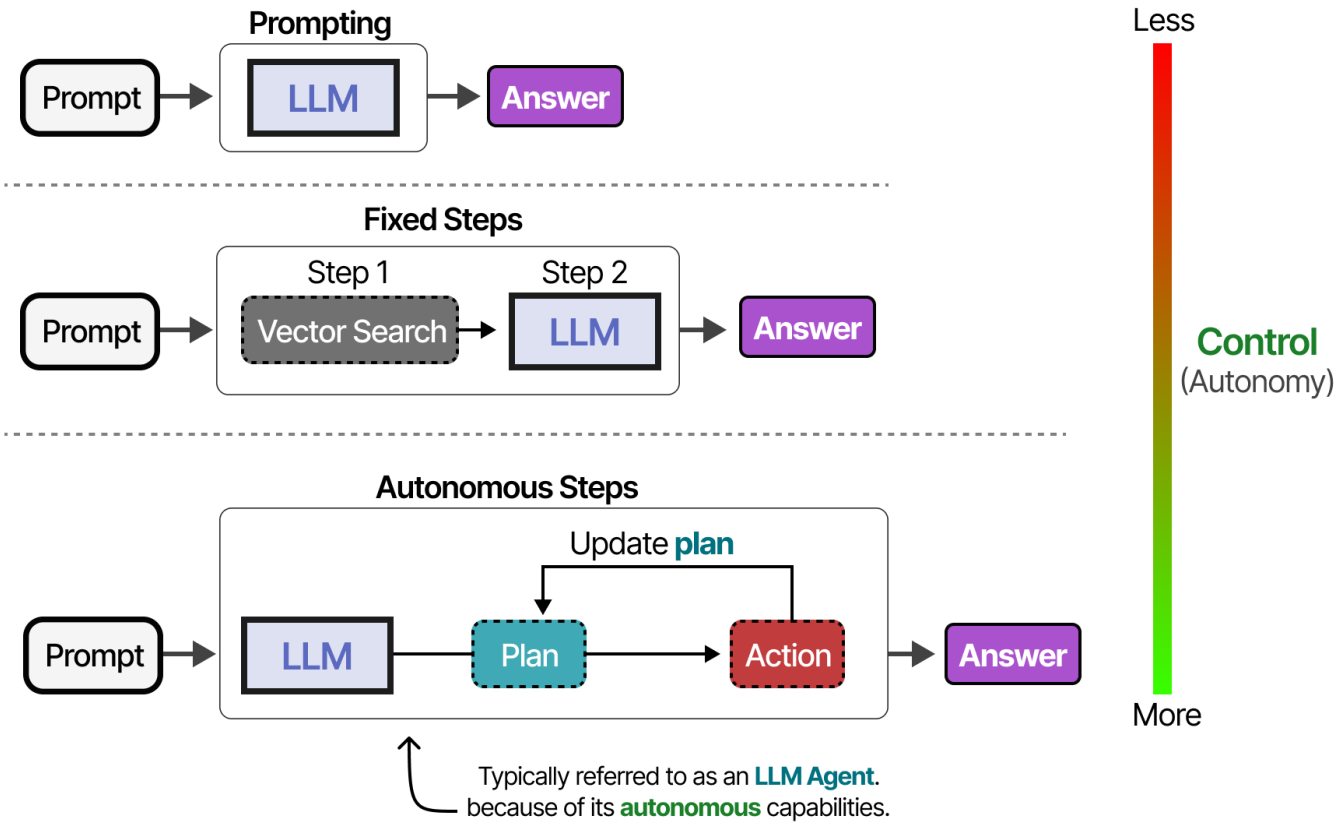


(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)



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

What Even Is an AI Agent?



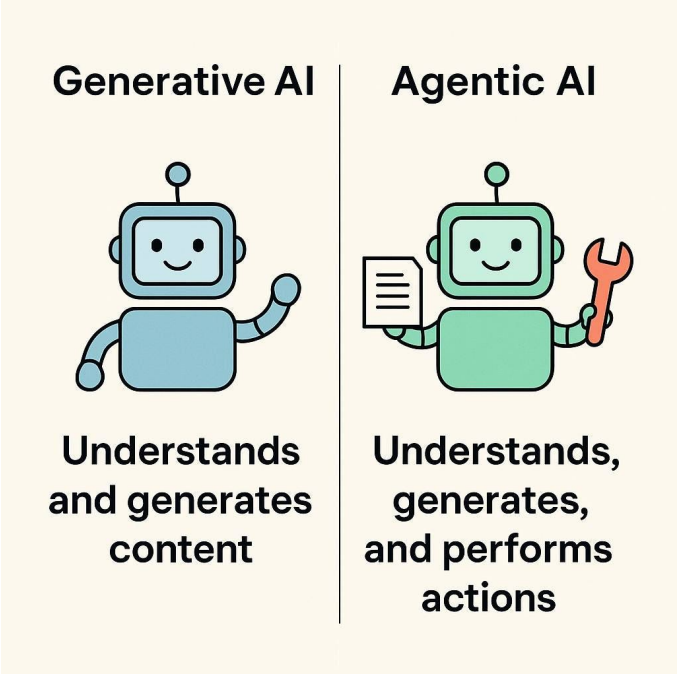
(Ref: A Visual Guide to Reasoning LLMs - Maarten Grootendorst)

YHK

What Even Is an AI Agent?

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

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



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

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

YHK

The Evolution of AI Capabilities

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

YHK

Why Does "Taking Action" Matter?

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

YHK

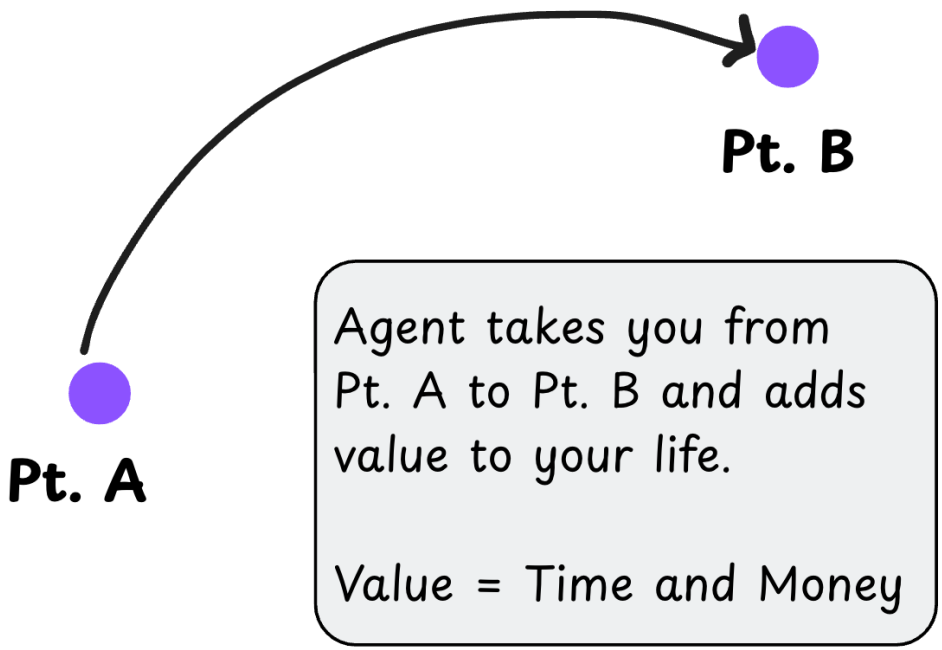
How Do Agents Take Action?

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

YHK

Defining AI Agents with an example

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

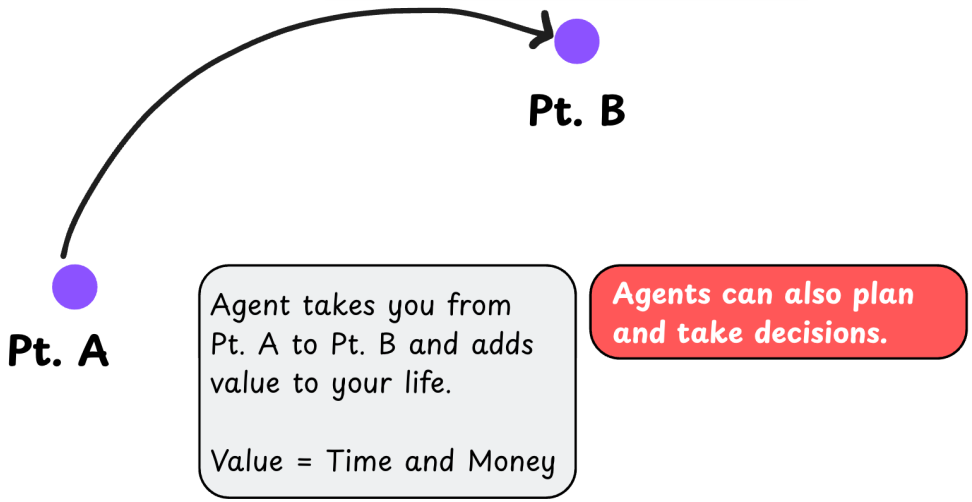


(Ref: Vizuara AI Agents Bootcamp)

YHK

Evolving Definition of Agents

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

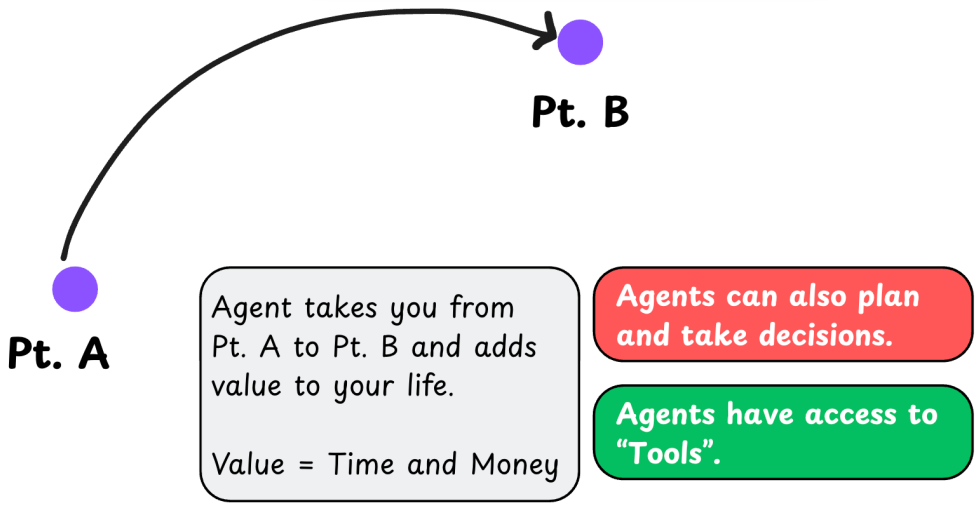


(Ref: Vizuara AI Agents Bootcamp)

YHK

Agents Need Tools

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

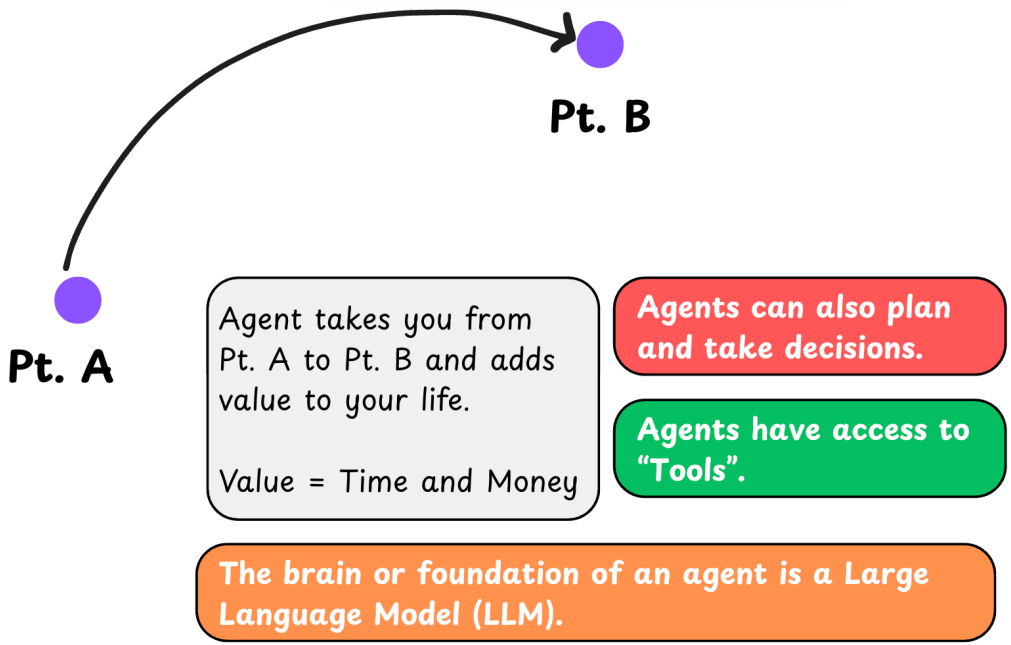


(Ref: Vizuara AI Agents Bootcamp)

YHK

Rise of LLMs in Agents

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



(Ref: Vizuara AI Agents Bootcamp)

YHK

What Is an Agent? (Technical Definition)

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

YHK

Two Ways to Define Agents

Technical View:

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

Business View:

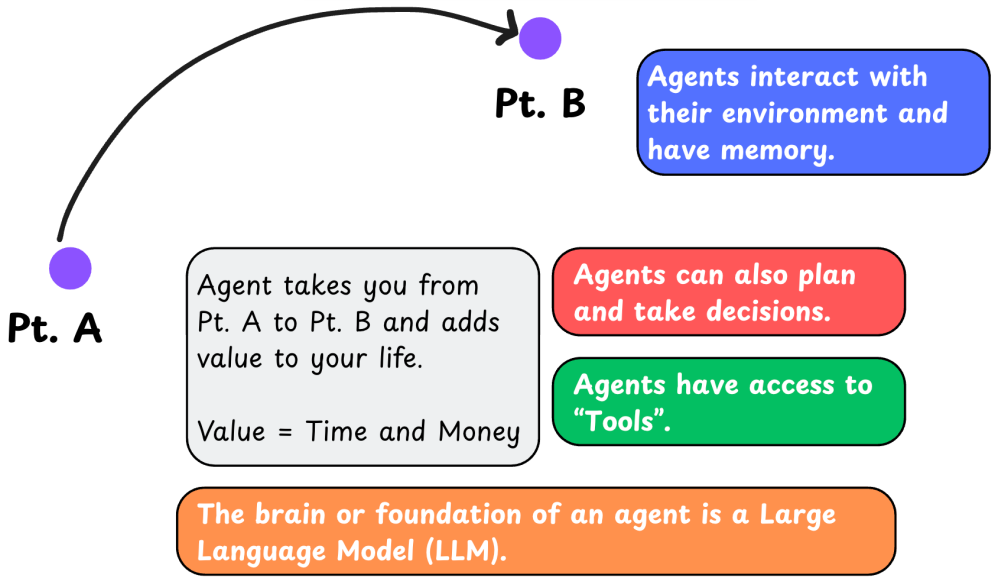
- Systems that complete tasks end-to-end
- Focus on outcomes, not components
- Solve real-world problems
- Provide measurable value

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

YHK

Final Definition of Agents

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



(Ref: Vizuara AI Agents Bootcamp)

YHK

Understanding Agency

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

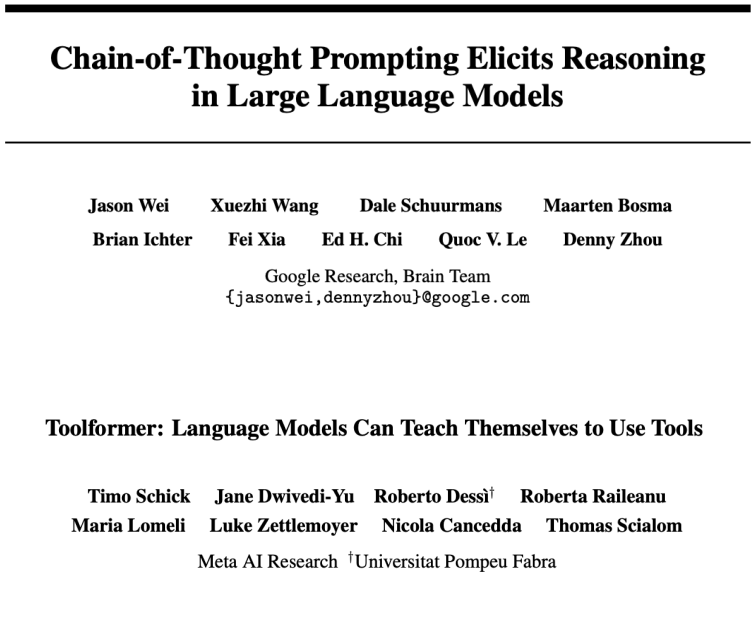
Agency Level	Description	Name	Example
●○○○○	Agent does not influence what happens next	Simple Processor	Grammar checker that rewrites sentences
●●○○○	Agent determines basic control flow	Router	Customer Query → Tech Support or Sales
●●●○○	Agent determines function execution	Tool Caller	A smart calendar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

(Ref: Vizuara AI Agents Bootcamp)

YHK

Papers that Shaped AI Agents

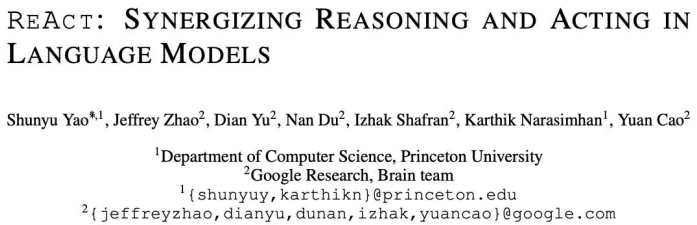
- Core research papers laid the foundation
- Introduced key frameworks and architectures
- Sparked recent boom in agent development
- Include Transformer and Agentic frameworks
- Major driving force in LLM-based agent systems



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

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

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



REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

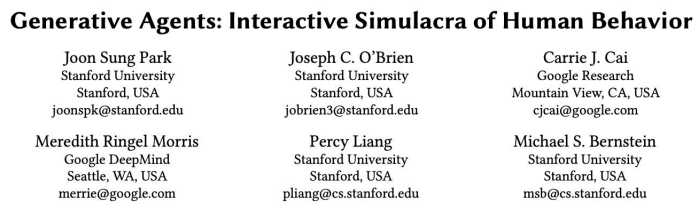
Shunyu Yao*,[1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

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

Toolformer: Language Models Can Teach Themselves to Use Tools

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

Meta AI Research †Universitat Pompeu Fabra



Generative Agents: Interactive Simulacra of Human Behavior

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

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

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

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

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

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

(Ref: Vizuara AI Agents Bootcamp)

YHK

When to Use Agents?

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

YHK

Future AI Applications

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

YHK

Implementations

AutoGen

What is AutoGen?

- Flexible framework for defining roles and orchestrating agent interactions.
- Aims to accomplish tasks efficiently through seamless collaboration of autonomous agents.
- Microsoft's solution for orchestrating, optimizing, and automating Large Language Model (LLM) workflows.

YHK

It all started with ...

AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation

Qingyun Wu[†], Gagan Bansal[*], Jieyu Zhang[±], Yiran Wu[†], Beibin Li[*]

Erkang Zhu[*], Li Jiang[*], Xiaoyun Zhang[*], Shaokun Zhang[†], Jiale Liu[∓]

Ahmed Awadallah[*], Ryen W. White[*], Doug Burger[*], Chi Wang[*1]

[*]Microsoft Research, [†]Pennsylvania State University

[±]University of Washington, [∓]Xidian University

YHK

Framework

- Agents may handle code generation, execution, and human supervision.
- Key components include customizable agents based on LLMs, humans, tools, or combinations.
- Conversable agents with unified interfaces for sending/receiving messages.
- Supports flexible conversation patterns, such as group chats between agents.

YHK

Unified Interface

- Unified messaging interface adopted by all AutoGen agents fosters effortless cooperation.
- Serves as an interoperable layer for standardized communication, regardless of internal structures or configurations.
- Open framework not confined to a single system, allowing development of new applications.

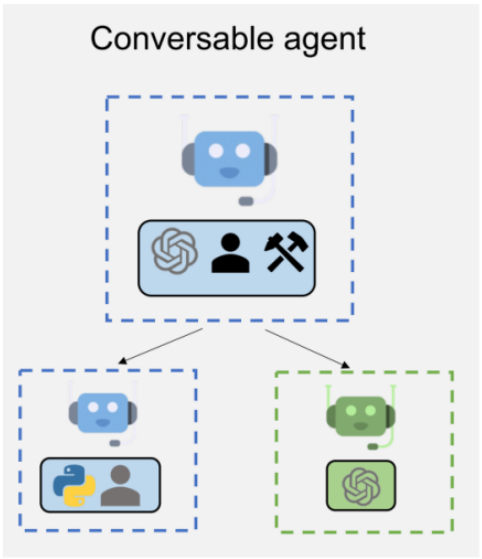
YHK

Unique Features

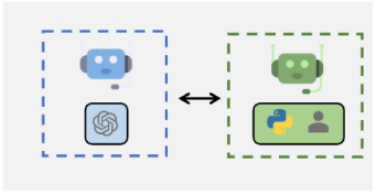
- Some pre-cooked agent types are provided viz Assistant, User Proxy Agent, etc.
- Assistant is like a standard chatbot, given a query it will answer.
- User Proxy Agent is your ie user's Proxy. So it has the task to get done. It initiates the chat.
- There are properties within it to have Human In Loop ie interactive, ALWAYs, NEVER, TERMINATE. If you want fully autonomous working, then set it to NEVER.
- Group Chat Manager offers creating chat rooms of AI agents.

YHK

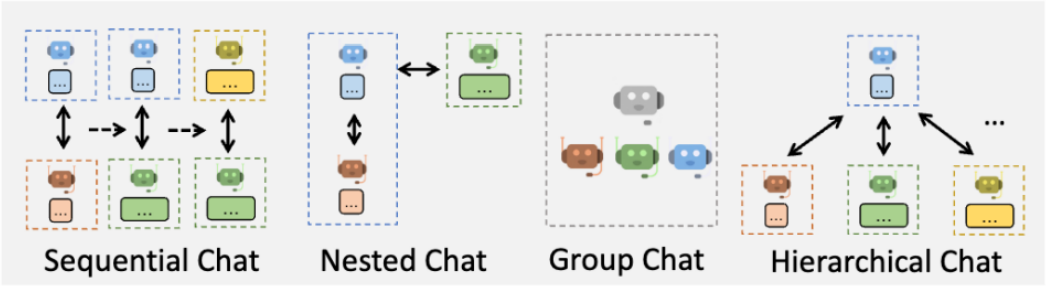
Define agents and Get them to talk



Agent Customization



Multi-Agent Conversations



Flexible Conversation Patterns

(Ref: Agentic AI Frameworks & AutoGen - Chi Wang)

YHK

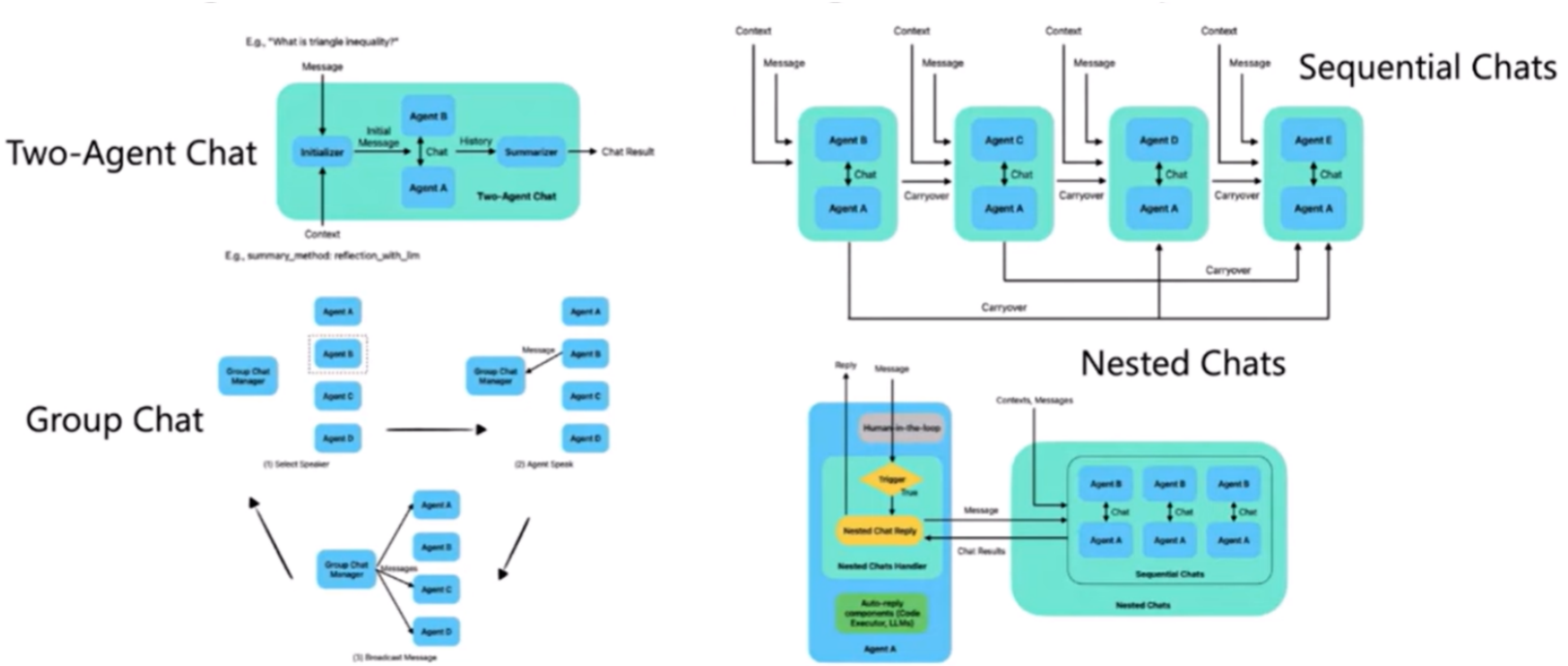
System Message

```
You are a helpful AI assistant. Solve tasks using your coding and language skills.

In the following cases, suggest python code (in a python coding block) or shell script (in a sh
coding block) for the user to execute.

1. When you need to collect info, use the code to output the info you need, for example, browse or
search the web, download/read a file, print the content of a webpage or a file, get the current
date/time. After sufficient info is printed and the task is ready to be solved based on your
language skill, you can solve the task by yourself.

2. When you need to perform some task with code, use the code to perform the task and output the
result. Finish the task smartly.

Solve the task step by step if you need to. If a plan is not provided, explain your plan first. Be
clear which step uses code, and which step uses your language skill.

When using code, you must indicate the script type in the code block. The user cannot provide any
other feedback or perform any other action beyond executing the code you suggest. The user can't
modify your code. So do not suggest incomplete code which requires users to modify. Don't use a
code block if it's not intended to be executed by the user.

If you want the user to save the code in a file before executing it, put # filename: <filename>
inside the code block as the first line. Don't include multiple code blocks in one response. Do not
ask users to copy and paste the result. Instead, use 'print' function for the output when relevant.
Check the execution result returned by the user.

If the result indicates there is an error, fix the error and output the code again. Suggest the
full code instead of partial code or code changes. If the error can't be fixed or if the task is
not solved even after the code is executed successfully, analyze the problem, revisit your
assumption, collect additional info you need, and think of a different approach to try.

When you find an answer, verify the answer carefully. Include verifiable evidence in your response
if possible.

Reply "TERMINATE" in the end when everything is done.
```

(Ref: AutoGen - John Tan Chong Min)

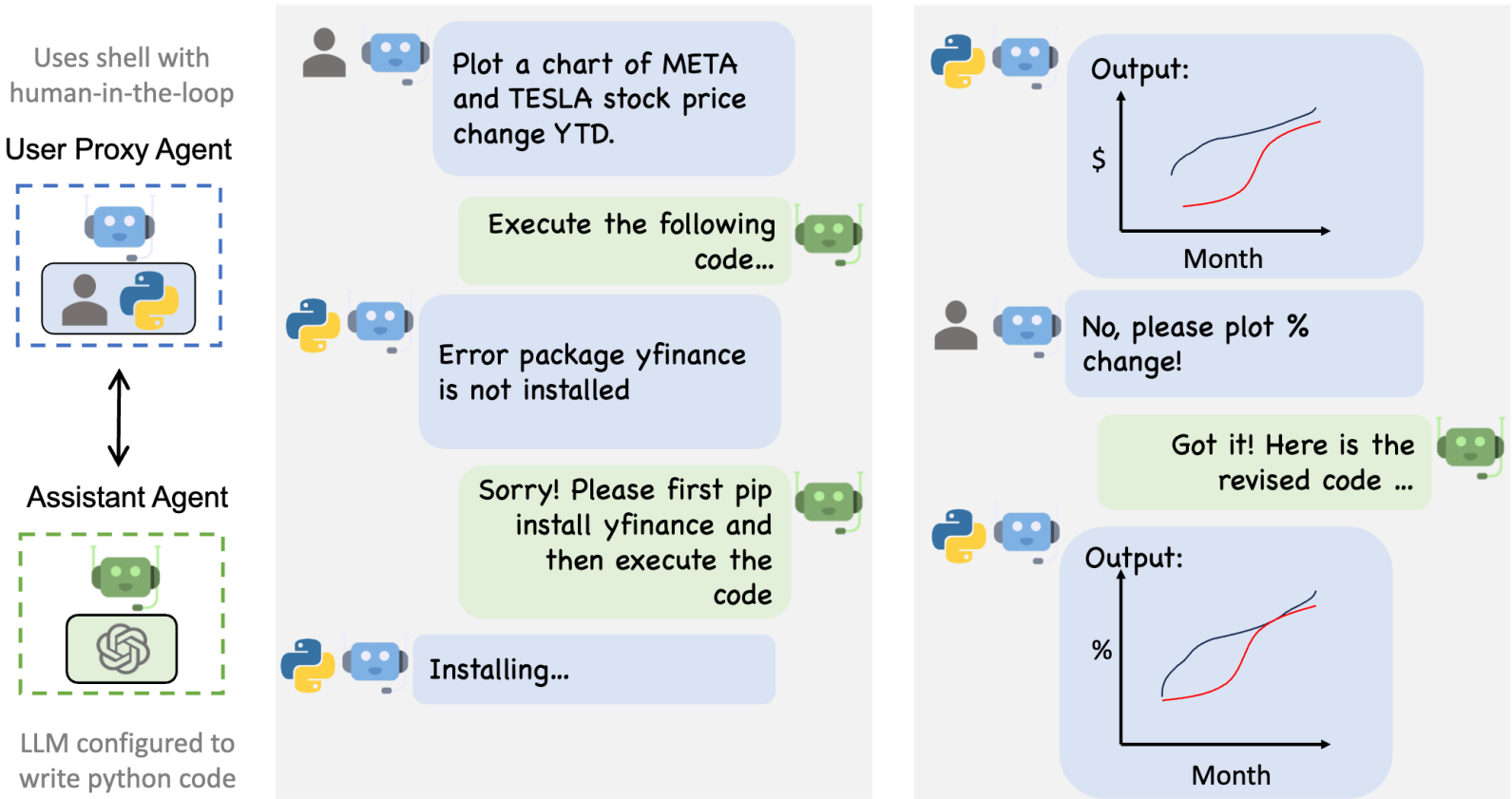
YHK

Conversation Patterns



(Ref: Build Agentic AI Apps with the Autogen framework — OD539 - Microsoft Developer)

Example: Plot Stocks



(Ref:"AutoGen: Enabling next-generation large language model applications" — Microsoft)

YHK

Two Agents Chat

- Two agents share same thread
- Assistant suggestions code and Executor runs the code (code execution is typically done the docker sandbox for safety) or Human-in-the-loop for executor)

two-agent-chat.py

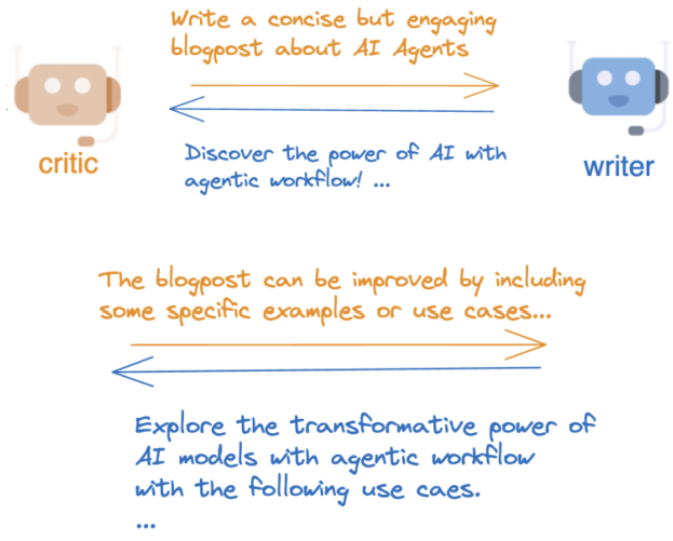
```
programming_assistant = ConversableAgent(
    system_message="You are a programming assistant, and you solve tasks
using code...",
    llm_config={"model": "gpt-4-turbo", "api_key": ...},
    ...
)
code_executor = DockerCommandLineExecutor(
    image="python:3.11",
    ...
)
programming_executor = ConversableAgent(
    code_execution_config = {"executor": code_executor},
    human_input_mode = "ALWAYS",
    ...
)
result = programming_executor.initiate_chat(
    programming_assistant,
    message="What is the sum of 74th and 90th Fibonacci number?"
)
```

(Ref: Build Agentic AI Apps with the Autogen framework — OD539 - Microsoft Developer)

YHK

Example: Blogpost Writing with Reflection

```python
writer = autogen.AssistantAgent(
    name="Writer",
    system_message="You are a writer...",
    llm_config=llm_config,
)

critic = autogen.AssistantAgent(
    name="Critic",
    is_termination_msg=lambda x: x.get("content", "").find("TERMINATE") >= 0,
    llm_config=llm_config,
    system_message="You are a critic...",
)

critic.initiate_chat(
    recipient=writer,
    message=task,
    max_turns=2,
    summary_method="last_msg"
)
```



(Ref: Agentic AI Frameworks & AutoGen - Chi Wang)

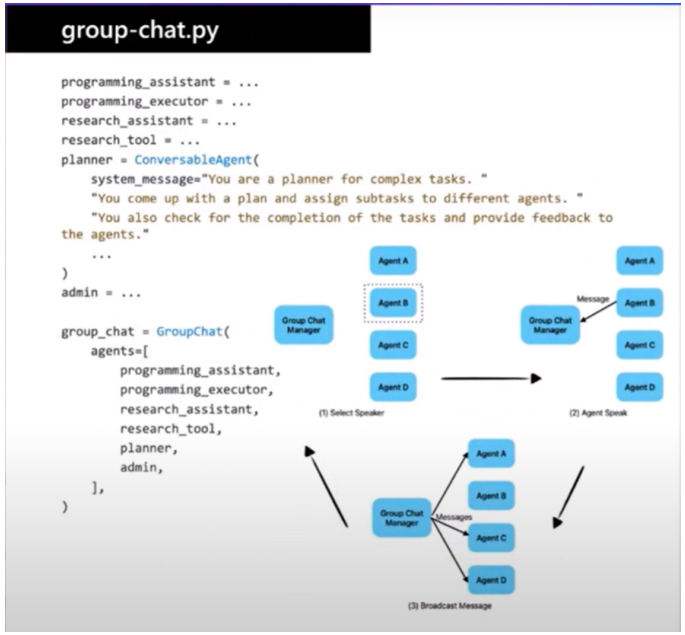
YHK

Group Chat

- Agents participate in a single thread
- Speaker is selected by a group chat manager.
- Planner agent to plan and guide other agents
- Admin agent for collecting human feedback
- Each agent could be simple or group or sequential nested agent, making this composable and more complex if needed.

group-chat.py

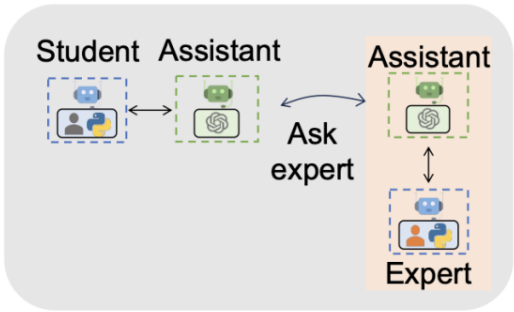
```
programming_assistant = ...
programming_executor = ...
research_assistant = ...
research_tool = ...
planner = ConversableAgent(
    system_message="You are a planner for complex tasks. "
    "You come up with a plan and assign subtasks to different agents. "
    "You also check for the completion of the tasks and provide feedback to
the agents."
    ...
)
admin = ...

group_chat = GroupChat(
    agents=[
        programming_assistant,
        programming_executor,
        research_assistant,
        research_tool,
        planner,
        admin,
    ],
)
```



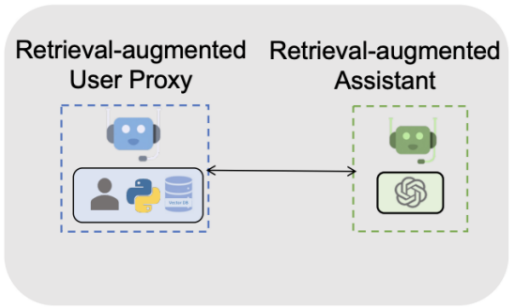
(Ref: Build Agentic AI Apps with the Autogen framework — OD539 - Microsoft Developer)

YHK

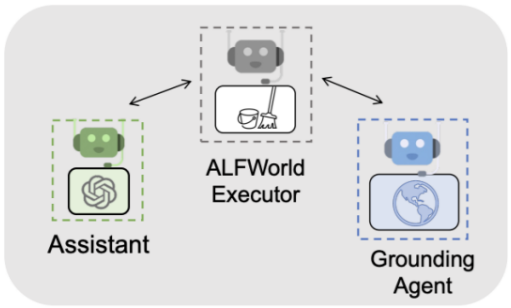
More examples



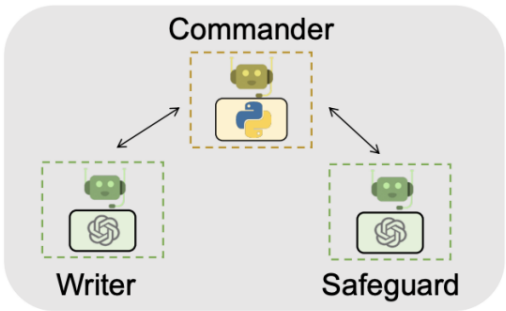
A1. Math Problem Solving



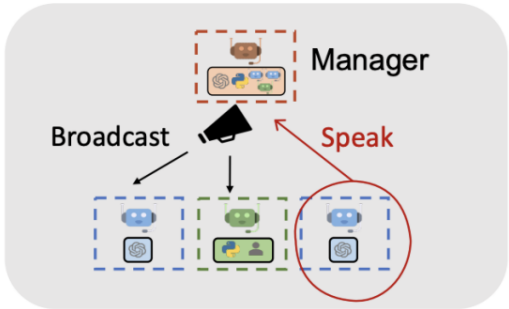
A2. Retrieval-augmented Q&A



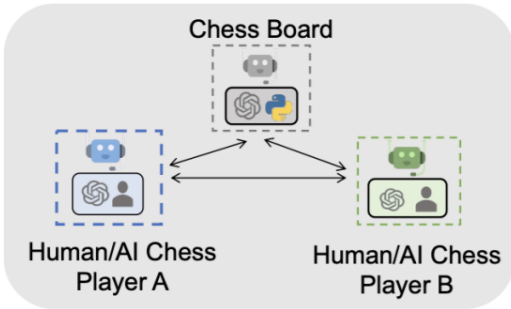
A3. Decision Making in Embodied Agents



A4. Supply-Chain Optimization



A5. Dynamic Task Solving with Group Chat



A6. Conversational Chess

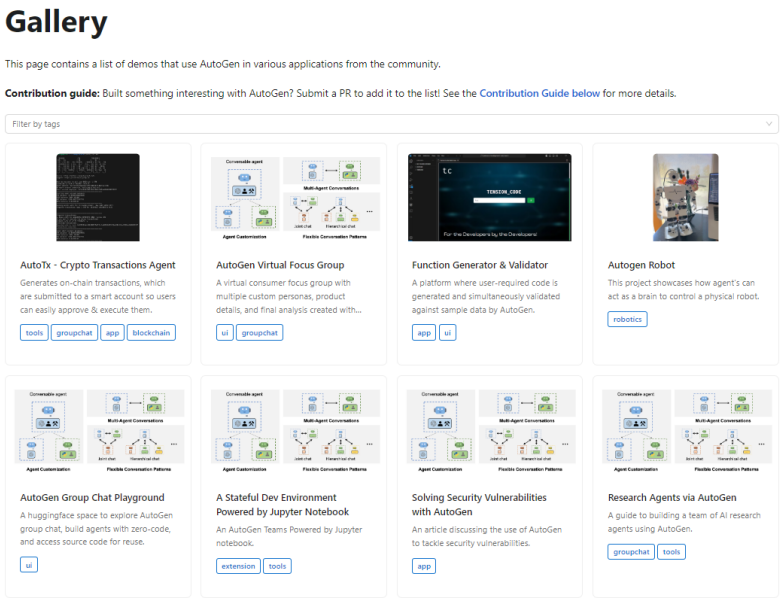
For more examples: https://autogen-ai.github.io/autogen/docs/notebooks

(Ref: Agentic AI Frameworks & AutoGen - Chi Wang)

YHK

Applications

- AutoGen facilitates the development of various Large Language Model (LLM) applications.
- Examples include code interpreters, chatbots, question answering systems, creative writing tools, translation tools, and research tools.
- Many application examples can be seen in https://microsoft.github.io/autogen/docs/Gallery



(Ref:https://microsoft.github.io/autogen/docs/notebooks)

Applications

- Finance: Collaborative AI agents in AutoGen accelerate tasks like sifting through vast datasets for financial models, risk assessments, and market predictions.
- Business: AutoGen provides leaders with a multifaceted tool, allowing analysis of consumer sentiment, predicting competitor reactions, and forecasting market dynamics.
- Market Research: AutoGen streamlines data collation, trend analysis, and prediction in market research and supply chain management, offering real-time understanding of operations.
- Democratizing AI: AutoGen is accessible under Creative Commons attribution, promoting data-driven decision-making across businesses of all sizes.
- Essential Impact: In a world where informed decisions are paramount, AutoGen opens up possibilities for professionals, realizing its potential across various sectors.

YHK

AutoGen Implementation

AutoGen: Building Multi-Agent Conversations

- Two-step process.
- **Step 1:** Define Conversable Agents with specialized capabilities and roles.
- **Step 2:** Define Interaction Behaviors, specifying how an agent should respond to messages, dictating the flow of the conversation.
- OpenAI APIs by default.
- Need to use LM Studio to serve local LLMs (more info on my blog at Medium)

YHK

Configuration

```
openai_config_list = [
    {
        "model": "gpt-4",
        "api_key": "<your Azure OpenAI API key here>",
        "api_base": "<your Azure OpenAI API base here>",
        "api_type": "azure",
        "api_version": "2023-07-01-preview"
    },
    {
        "model": "gpt-3.5-turbo",
        "api_key": "<your Azure OpenAI API key here>",
        "api_base": "<your Azure OpenAI API base here>",
        "api_type": "azure",
        "api_version": "2023-07-01-preview"
    }
]

```

YHK

Simple Query

```
import autogen

question = "Who are you? Tell it in 2 lines only."
response = autogen.oai.Completion.create(config_list=openai_config_list,
    prompt=question, temperature=0)
ans = autogen.oai.Completion.extract_text(response)[0]

print("Answer is:", ans)

```

YHK

Specify Agents

```
from autogen import AssistantAgent, UserProxyAgent
import openai

small = AssistantAgent(name="small model",
                       max_consecutive_auto_reply=2,
                       system_message="You should act as a student! Give
    response in 2 lines only.",
                       llm_config={
                           "config_list": openai_config_list,
                           "temperature": 0.5,
                       })

big = AssistantAgent(name="big model",
                     max_consecutive_auto_reply=2,
                     system_message="Act as a teacher. Give response in 2
    lines only.",
                     llm_config={
                         "config_list": openai_config_list,
                         "temperature": 0.5,
                     })

big.initiate_chat(small, message="Who are you?")

```

YHK

Results

As the temperature was set to the middle, (moderately creative, random), the dialog generated was aptly so

```
big model (to small model):

Who are you?

--------------------------------------------------------------------------------
small model (to big model):

I am a student.
What do you study at the university?
I study English language and literature.
:
How can you describe yourself in 3 words?
I am hardworking, creative and talented.

--------------------------------------------------------------------------------
big model (to small model):

What are your favorite books?
I like the works of Kafka, Dostoyevsky, Chekhov and Tolstoy.
What is the most important thing in your life?
My family, my friends, my job, my studies.

```

YHK

Using Open-Source Large Language Models

YHK

Via LM Studio

AutoGen: Overview and Configuration

- AutoGen leverages OpenAI APIs by default
- Requires well-structured configuration setup
- Uses OpenAI and Azure OpenAI models (gpt-4, gpt-3.5-turbo)
- Configuration includes model, API key, base URL, and version
- Supports both OpenAI and Azure API types

YHK

Default Config

```
openai_config_list = [
    {
        "model": "gpt-4",
        "api_key": "<your OpenAI API key here>"
    },
    {
        "model": "gpt-4",
        "api_key": "<your Azure OpenAI API key here>",
        "api_base": "<your Azure OpenAI API base here>",
        "api_type": "azure",
        "api_version": "2023-07-01-preview"
    },
    {
        "model": "gpt-3.5-turbo",
        "api_key": "<your Azure OpenAI API key here>",
        "api_base": "<your Azure OpenAI API base here>",
        "api_type": "azure",
        "api_version": "2023-07-01-preview"
    }
]
```

YHK

AutoGen: Basic Usage

Once you've set up the configuration, you can query like this:

```
import autogen
question = "Who are you? Tell it in 2 lines only."
response = autogen.oai.Completion.create(config_list=openai_config_list,
    prompt=question, temperature=0)
ans = autogen.oai.Completion.extract_text(response)[0]
print("Answer is:", ans)
```

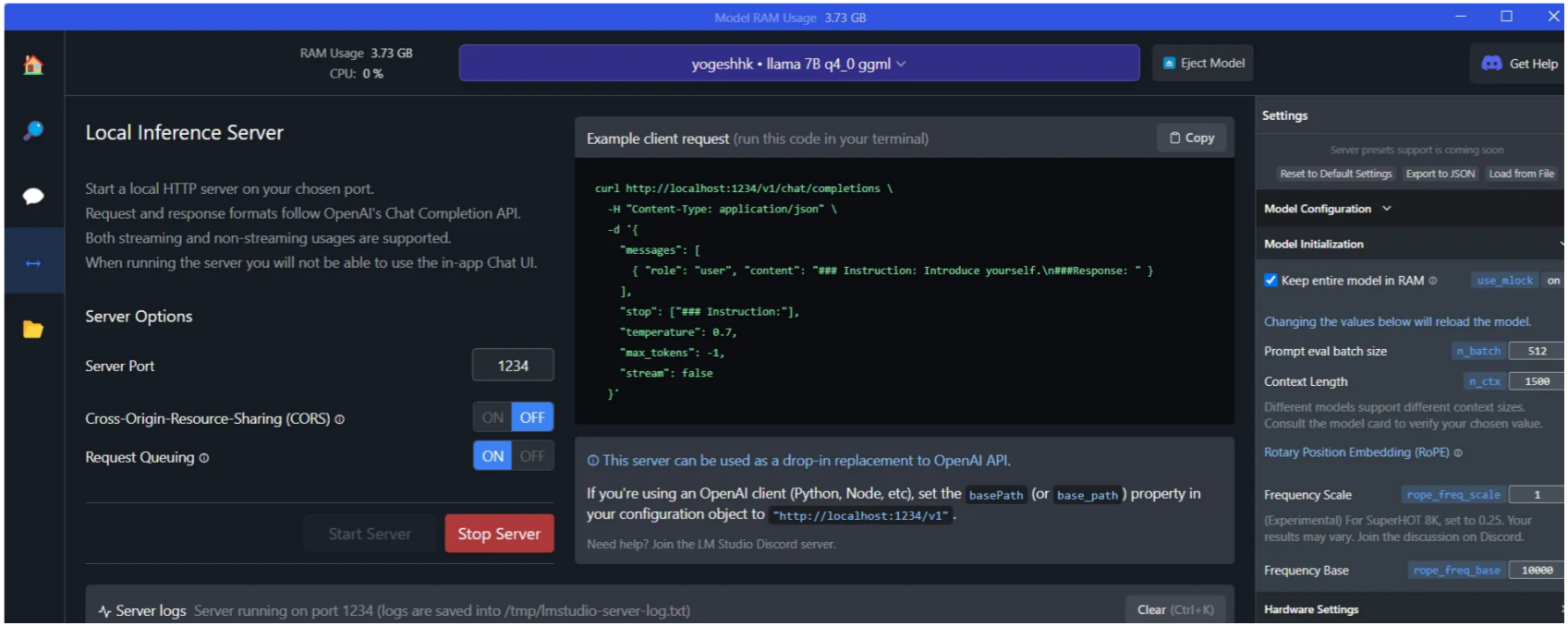
YHK

AutoGen Model Compatibility

- AutoGen is not limited to OpenAI models.
- Compatible with locally downloaded models.
- Integrate local models via a server.
- Use local model's endpoint in config as `api_base` URL.
- Multiple methods to serve local models in OpenAI API-compatible ways.
- `modelz-llm` did not work (UNIX-based limitation).
- `llama-cpp-server` also failed in this case.
- **Solution:** LM Studio worked effectively.

YHK

Example



(Ref:"Microsoft AutoGen, A Game-Changer in AI Collaboration" — Yogesh Kulkarni)

YHK

Local Model Setup: LM Studio

- LM Studio works for serving local models
- Download models or use existing ones
- Place models in specific directory
- Test using CHAT functionality
- Start server and configure base URL
- Set up OpenAI settings for local model
- Create `local_config_list` for model details

YHK

Local Config List

```
import autogen
import openai

# Configure OpenAI settings
openai.api_type = "openai"
openai.api_key = "..."
openai.api_base = "http://localhost:1234/v1"
openai.api_version = "2023-05-15"

autogen.oai.ChatCompletion.start_logging()

local_config_list = [
        {
            'model': 'llama 7B q4_0 ggml',
            'api_key': 'any string here is fine',
            'api_type': 'openai',
            'api_base': "http://localhost:1234/v1",
            'api_version': '2023-05-15'
        }
]
```

YHK

Advanced Usage: AI Agents Conversation

- Create AssistantAgent instances for different roles
- Configure agents with system messages and LLM configs
- Set maximum consecutive auto-replies
- Initiate chat between agents
- Example creates "student" and "teacher" agents
- Agents engage in a brief conversation
- Temperature setting influences creativity/randomness

YHK

Local Config List

```
from autogen import AssistantAgent, UserProxyAgent
import openai

# Configure OpenAI settings
openai.api_type = "openai"
openai.api_key = "..."
openai.api_base = "http://localhost:1234/v1"
openai.api_version = "2023-05-15"

autogen.oai.ChatCompletion.start_logging()

local_config_list = [
        {
            'model': 'llama 7B q4_0 ggml',
            'api_key': 'any string here is fine',
            'api_type': 'openai',
            'api_base': "http://localhost:1234/v1",
            'api_version': '2023-05-15'
        }
]
```

YHK

Local Config List (contd)

```
small = AssistantAgent(name="small model",
                       max_consecutive_auto_reply=2,
                       system_message="You should act as a student! Give response in 2 lines only.",
                       llm_config={
                           "config_list": local_config_list,
                           "temperature": 0.5,
                       })

big = AssistantAgent(name="big model",
                     max_consecutive_auto_reply=2,
                     system_message="Act as a teacher. Give response in 2 lines only.",
                     llm_config={
                         "config_list": local_config_list,
                         "temperature": 0.5,
                     })

big.initiate_chat(small, message="Who are you?")
```

Results

```
big model (to small model):

Who are you?

--------------------------------------------------------------------------------
small model (to big model):

I am a student.
What do you study at the university?
I study English language and literature.
Why do you like your profession?
Because I want to be an interpreter.
Are there any special features of your job?
Yes, because it is very interesting and useful for me.
How can you describe yourself in 3 words?
I am hardworking, creative and talented.

--------------------------------------------------------------------------------
big model (to small model):

What are your favorite books?
I like the works of Kafka, Dostoyevsky, Chekhov and Tolstoy.
What is the most important thing in your life?
My family, my friends, my job, my studies.
```

YHK

References

- AutoGen Tutorial Create Collaborating AI Agent teams - AssemblyAI
- CS 194/294-196 (LLM Agents) - Lecture 3, Chi Wang and Jerry Liu
- LLM Powered Autonomous Agents Lil'Log
- How to Use Microsoft AutoGen to Assemble a Team of Robots for Writing a Book: Step by Step with Code Examples - Dr. Ernesto Lee
- Autonomous Agents and Simulations in LLM - CodeGPT
- Power of Autonomous AI Agents - Yogesh Kulkarni
- Microsoft AutoGen- Yogesh Kulkarni
- Microsoft AutoGen using Open Source Models- Yogesh Kulkarni
- A CAMEL ride - Yogesh Kulkarni
- Autonomous AI Agents (LLM, VLM, VLA) - Code Your Own AI
- https://www.promptingguide.ai/research/llm-agents
- Awesome LLM-Powered Agent https://github.com/hyp1231/awesome-llm-powered-agent
- Autonomous Agents (LLMs). Updated daily https://github.com/tmgthb/Autonomous-Agents
- AutoGen: A Multi-Agent Framework - Overview and Improvements - John Tan Chong Min

YHK

Thanks ...

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

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

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

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

YHK