global AI bootcamp

16 March 2024

Introduction to Autonomous Agents

Yogesh Haribhau Kulkarni

MVP (AI)

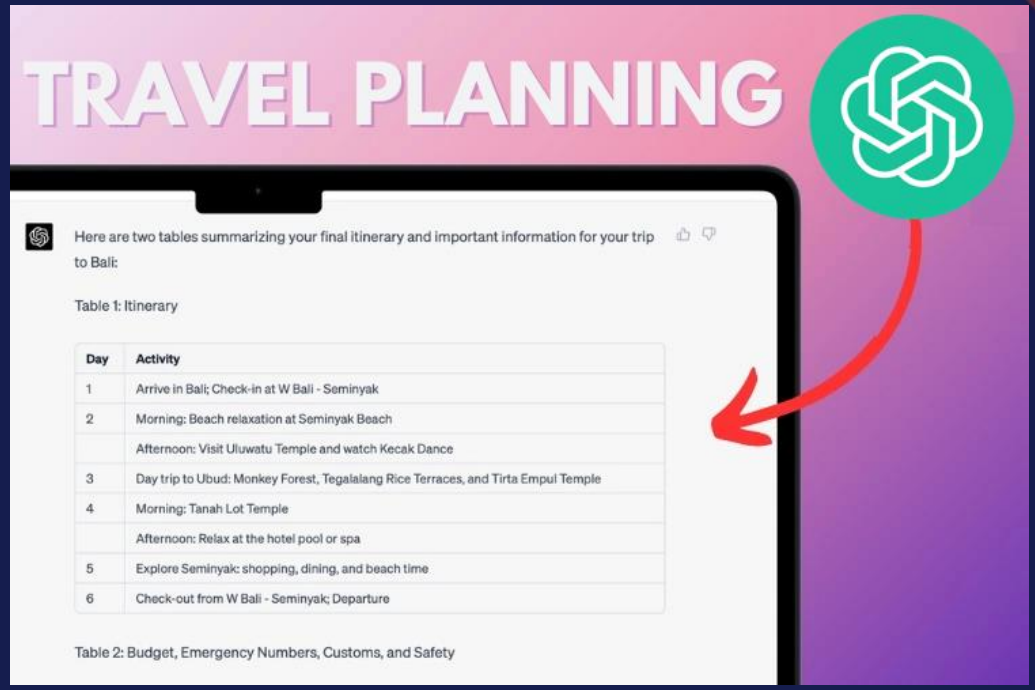
Agenda

- Introduction
- Implementation (AutoGen)
- Conclusions

Introduction

What Problem are your trying to solve?

- Imagine you are planning a trip
- Obviously, you ask chatgpt



source

But then ...

- Who is going to do booking?
- Who is going to check availability?
- Bring costing
- See if it's the budget

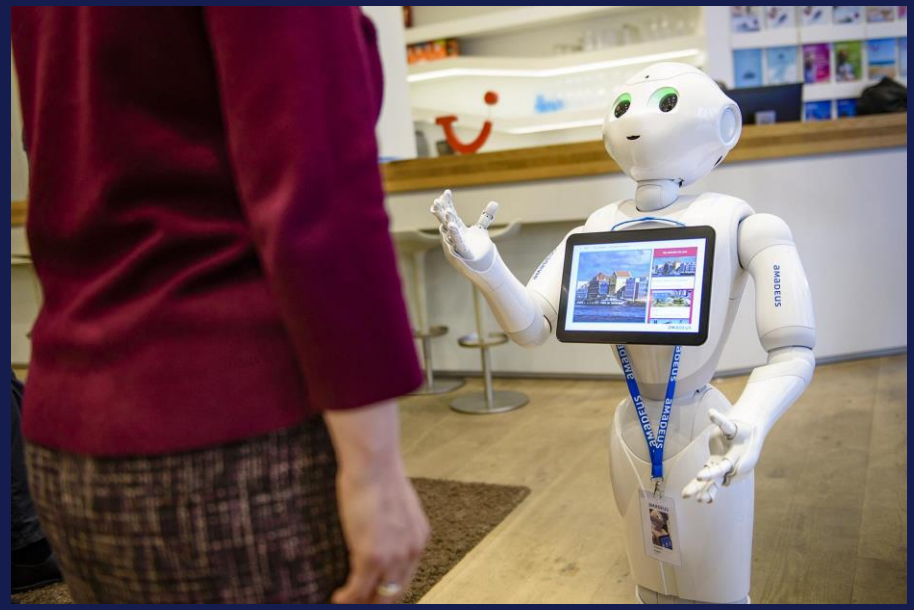
- That's, You!!

source

Oh No!! But, but, but ...

- Can someone do it for me?
- Just check for approval
- Or even do it fully on its own
- Autonomously...

- That's' Agents
- LLM based Agents



source

What are Autonomous AI Agents?

- Collaborative approach yields astonishing enhancements in performance and capabilities. Contrasted with using a single AI, such as ChatGPT, in isolation.
- Ability to assume distinct roles within a team. Like professionals in various fields.
- Each agent contributes specialized expertise to the conversation.

The Blueprint

- **Planning**: Reflects on past experiences, offers self-critiques, and breaks down tasks into manageable steps using sub-goal decomposition.
- **Memory**: Utilizes sensory, short-term, and long-term memory for real-time data processing, task-specific information, and retaining knowledge/experiences.
- **Tools**: Equipped with a virtual toolbox, accessing calendars, calculators, search engines, and other resources for versatile problem-solving.

Flow: The Symphony

- Task Decomposition
- Model (LLM) Selection
- Task Execution leveraging planning, memory, and tools
- Response Generation

Popular Agentic Frameworks

- **BabyAGI**: Pioneering AI learning system.
- **AutoGPT**: Automates content generation.
- **GPT Engineer**: Assists in coding and software development.
- **AutoGen**: Dialog based planning and execution

Implementation AutoGen by Microsoft

What is AutoGen?

- Microsoft's Flexible framework for defining roles and orchestrating agent interactions.
- Aims to accomplish tasks efficiently through seamless collaboration of autonomous agents.

What is AutoGen?

- Microsoft's Flexible framework for defining roles and orchestrating agent interactions.
- Created by a team at Microsoft Research, Pennsylvania State University, the University of Washington, and Xidian University.
- Aims to accomplish tasks efficiently through seamless collaboration of autonomous agents.

Components

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

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 (but can use Open Source LLMs also)
- Need to use LM studio to serve local LLMs (more info on my blog at Medium)

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"
    }
]
```

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)
```

Agent Types

- **UserProxyAgent** : a proxy for a human in the agent-agent conversations. It can be set up to solicit human input or it can be set to execute python code
- **AssistantAgent** : can be configured to play different roles. For example, to be a “Critic” of code written by others. You configure and create an agent is to instantiate it with a special “system_message”
- Each Agent has a “send” and a “receive” method.

Agent Types

- E.g. One UserProxyAgent is paired with one Assistant agent.
- The communication begins with

user_proxy.initiate_chat(Assistant, message = "the text of the message to the assistant")

- The user_proxy generates a "send" message to the Assistant.
- Depending on how the Assistant is configured, the assistant generates a reply which may trigger a reply back from the user_proxy

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?")
```

Results

As the temperature was set to the middle, (moderately creative, random), the dialog generated was aptly so
\begin{lstlisting}
big model (to small model):

Who are you?

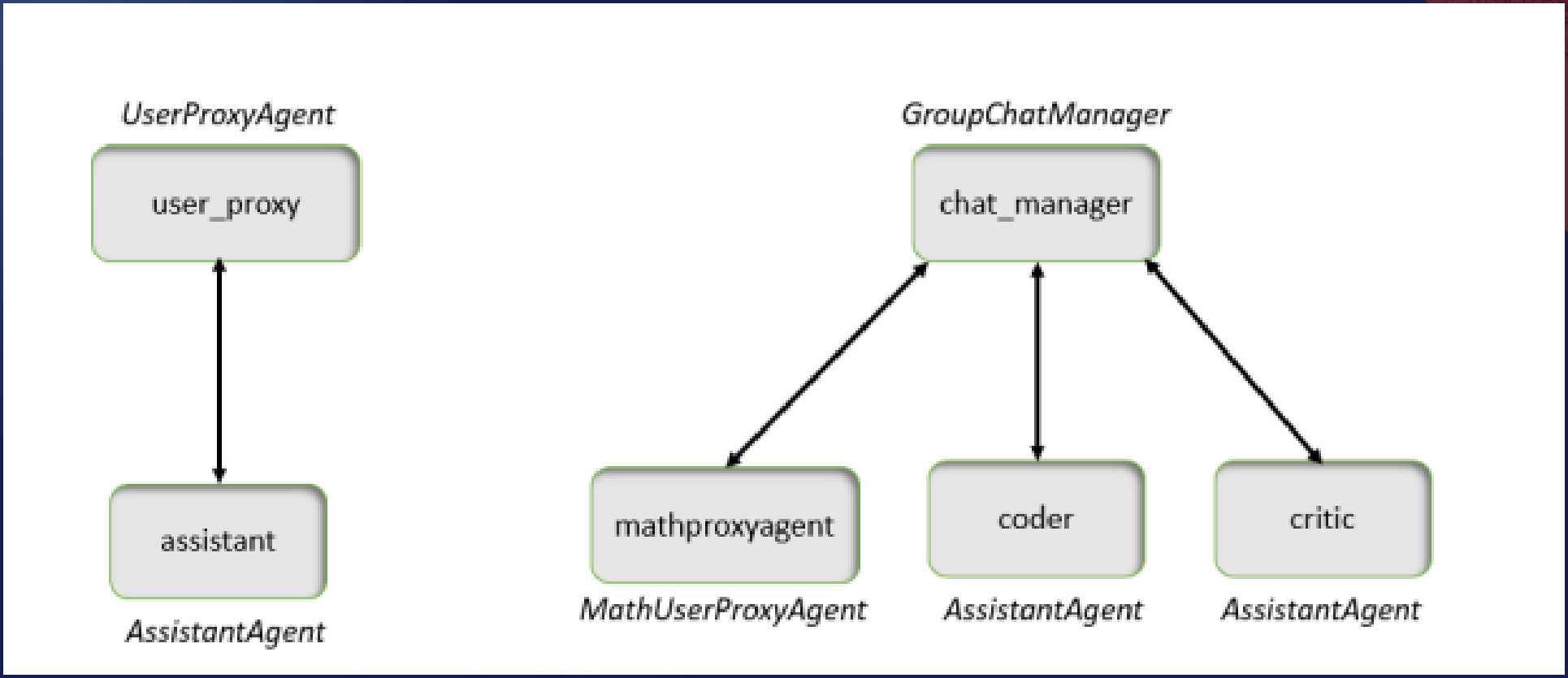
---
small model (to big model):

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):

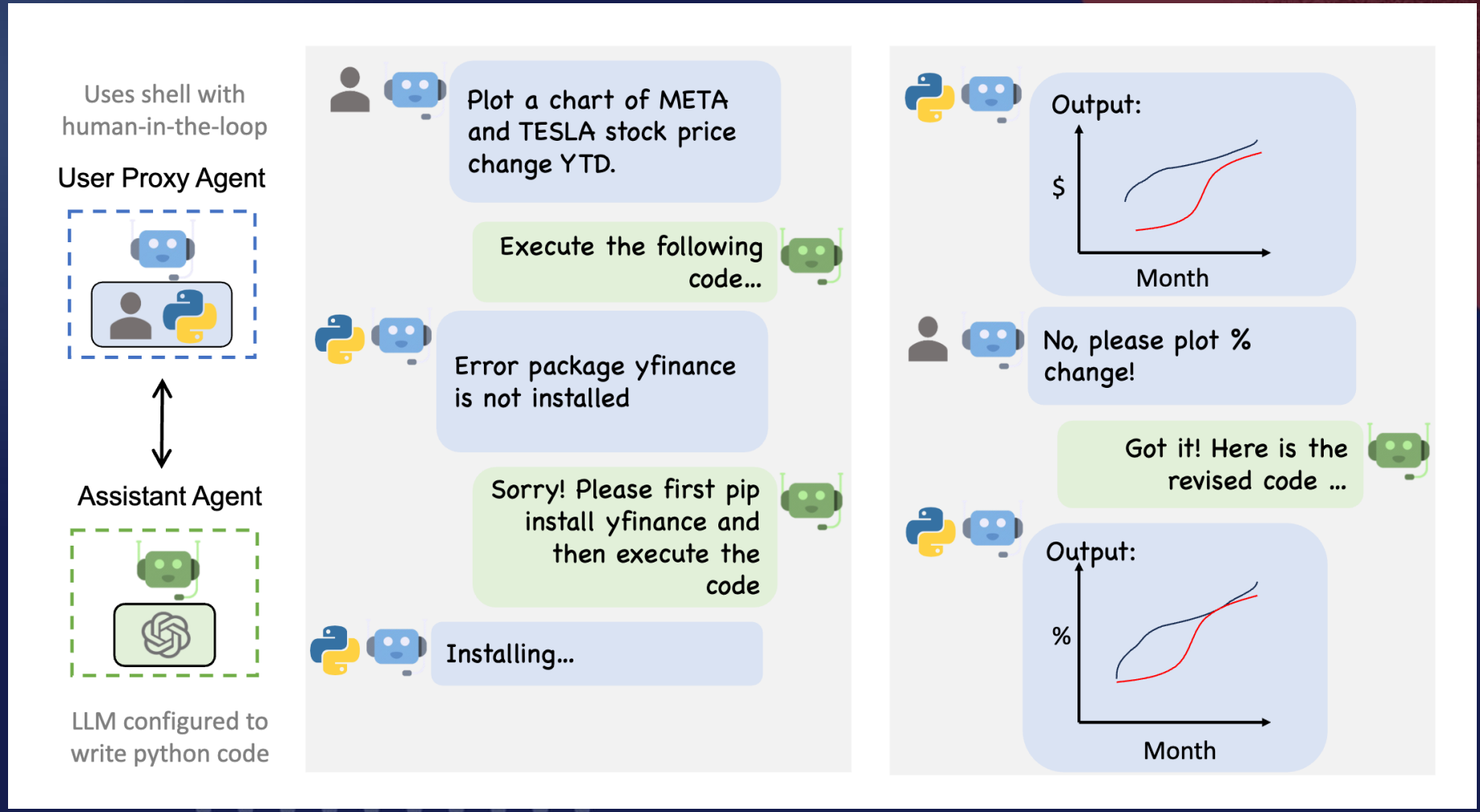
What are your favorite books?
I like the works of Kafka, Dostoyevsky, Chekhov and Tolstoy.
What is the most important thing in your life?
My family, my friends, my job, my studies.

Example Patterns



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

Applications



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

Agent Types

- **GroupChatManager** : to engage more than one Autogen Agent in a conversation
- A group chat usually begins with a UserProxyAgent instance sending a message to the group chat manager to start the discussion.
- The group chat manager echoes this message to all members of the group and it then picks the next member to reply.
- There are several ways this selection may happen, even RANDOM or in round-one-by-one, etc But still how??

Selecting next player

- The default and most interesting way the next speaker is selected is to let the large language model do it.
- To do this, the group chat manager sends the following request to the LLM:

"Read the above conversation. Then select the next role from [list of agents in the group] to play. Only return the role."

- As we shall see this works surprisingly well.

Example

```
from autogen import AssistantAgent, UserProxyAgent, config_list_from_json
config_list = config_list_from_json(env_or_file="OAI_CONFIG_LIST")
analyst = AssistantAgent(name =“analyst",
                         system_message="You are a Senior Financial Analyst  who fetches stock prices...",
                         llm_config={"config_list": config_list})
coder = AssistantAgent(name =“coder",
                         system_message="You are a  Programmer who  can plot graphs ...",
                         llm_config={"config_list": config_list})
user_proxy = UserProxyAgent(name=“admin",
                         system_message=“Gets analyst to fetch price and coder to plot it ...",
                         llm_config={"config_list": config_list})

groupchat = autogen.GroupChat( agents=[user_proxy, analyst, coder], messages=[], max_round=20)
manager = autogen.GroupChatManager(groupchat=groupchat, llm_config=llm_config)

user_proxy.initiate_chat(manager, message="Plot a chart of NVDA and TESLA stock price change YTD.")
```

(Ref: AutoGen - Stocks Prices demo | retkowsky/Azure-OpenAI-demosAutogen/AutoGen - stock prices demo.ipynb)

Interaction

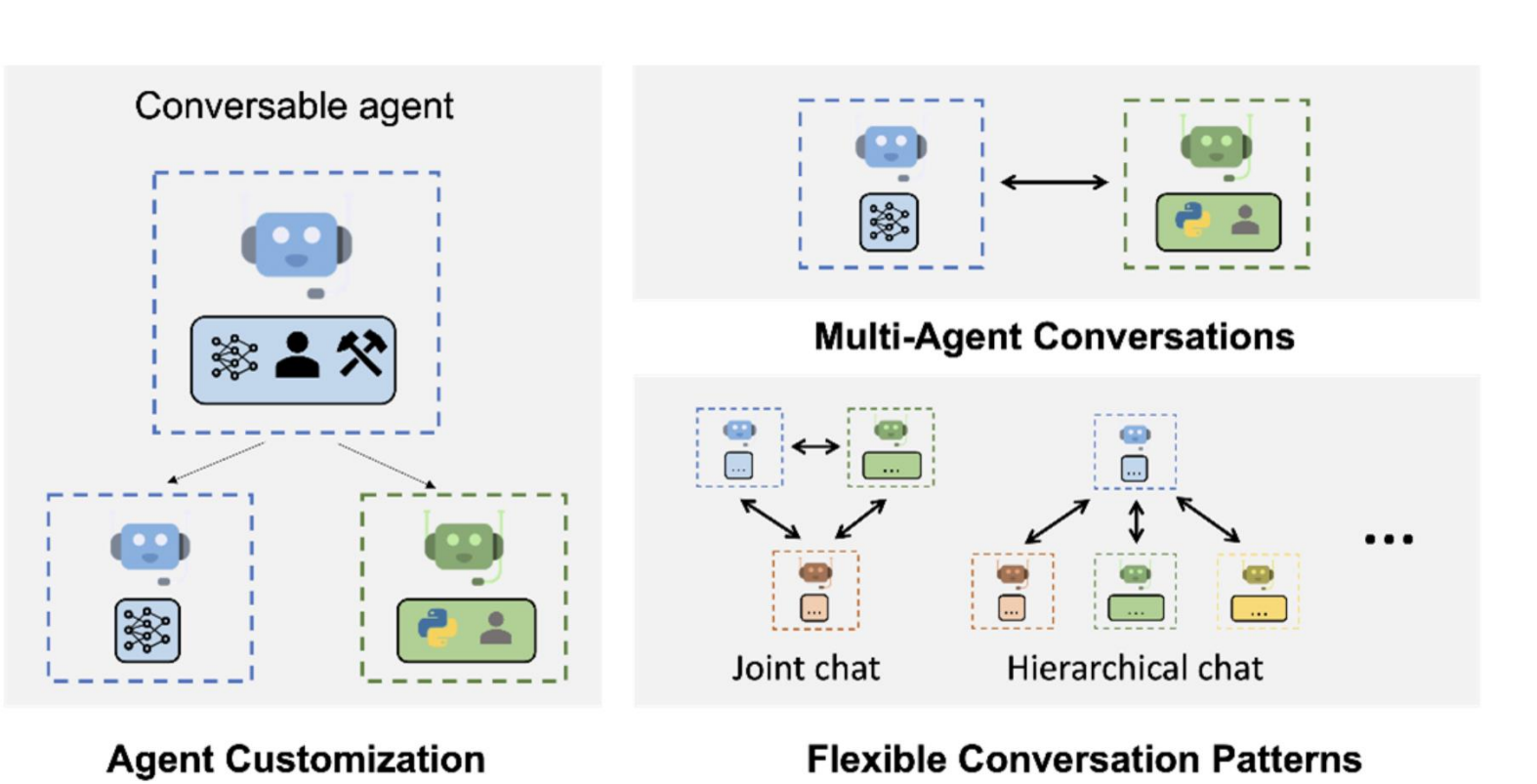


Figure 1. AutoGen enables complex LLM-based workflows using multi-agent conversations. (Left) AutoGen agents are customizable and can be based on LLMs, tools, humans, and even a combination of them. (Top-right) Agents can converse to solve tasks. (Bottom-right) The framework supports many additional complex conversation patterns.

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

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.

Features

- Innovative features include the User Proxy Agent for human intervention (human in the loop).
- Group Chat Manager offers flexibility in creating chat rooms of AI agents.
- Surpasses existing solutions like ChatDev, empowering developers to design dynamic conversational structures.

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.

Applications

- Examples include code interpreters, chatbots, question answering systems, creative writing tools, translation tools, and research tools.
- 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.

Conclusions

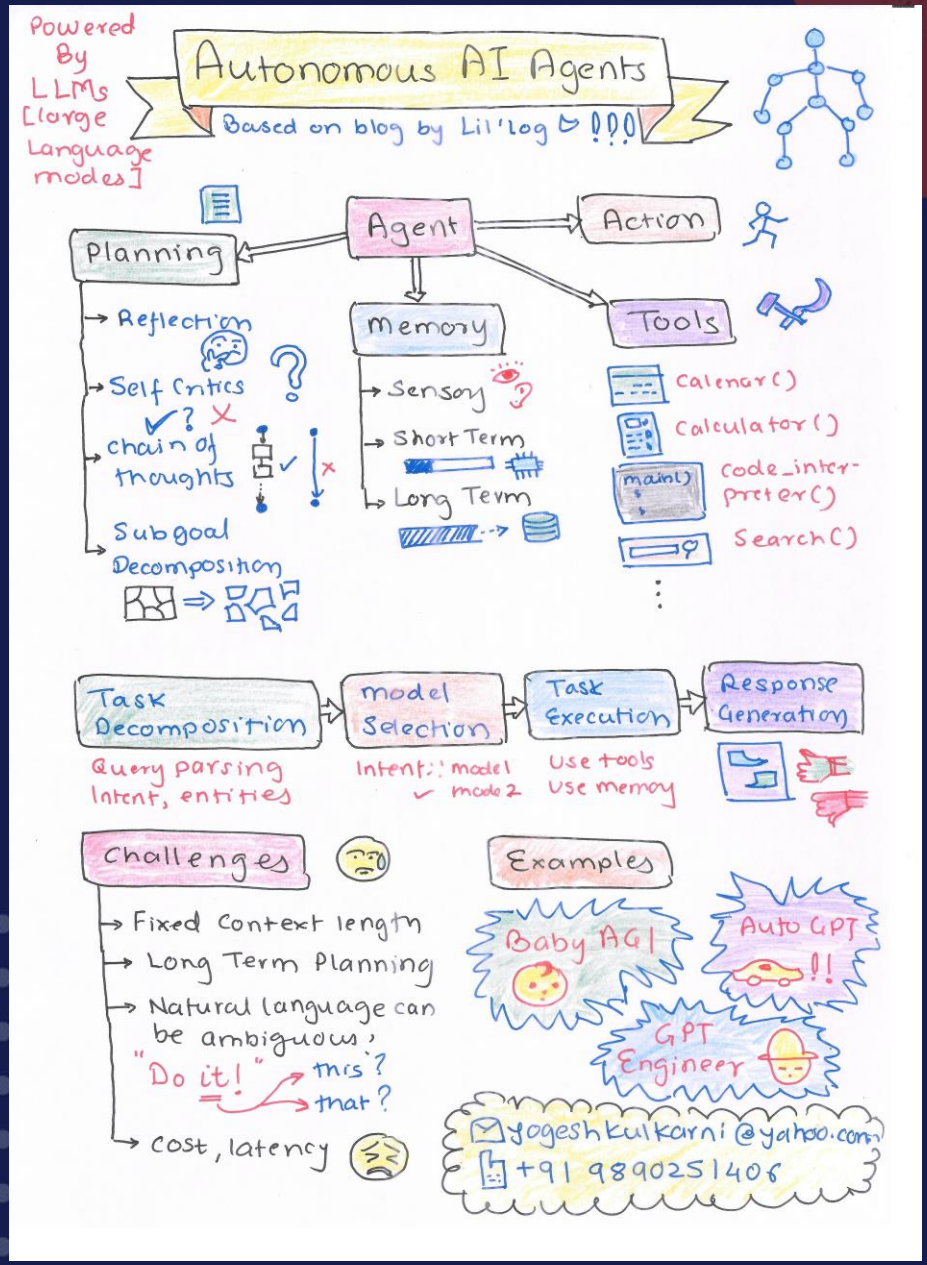
In General

- Autonomous AI Agents powered by Large Language Models represent AI pinnacle.
- Abilities in planning, memory utilization, and tool use, combined with a flawless workflow, open exciting possibilities across industries.
- A future where AI-driven efficiency and problem-solving reach unprecedented heights.
- Machines that think, remember, and adapt — a revolution in AI.

Challenges

- **Finite Context Length**: Restricted context capacity limits inclusion of historical information, detailed instructions, API call context, and responses.
- **Long-term planning and task decomposition**: LLMs struggle to adjust plans when faced with unexpected errors.
- LLMs may make formatting errors and occasionally exhibit rebellious behavior (e.g., refuse to follow an instruction).
- Less robust compared to humans who learn from trial and error

My Sketchnote



The Future

- Transformative era in AI collaboration is on the horizon.
- Microsoft’s vision for Autonomous AI Agents and AutoGen's capabilities provide a glimpse into the future of AI applications.
- Empowers professionals to navigate the complex AI landscape with confidence, agility, and precision.

Towards AGI

- Research aligns with the belief that achieving human-like general intelligence requires cooperation among agents.
- Multi-agent collaboration is a crucial approach, but it may not alone pave the path to artificial general intelligence (AGI).
- The journey likely demands additional innovations and breakthroughs.
- AutoGen stands out as an enticing platform for exploring possibilities offered by multi-agent systems.

References

- LLM Powered Autonomous Agents Lil'Log
- 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
- Prompt Guide
- Awesome LLM-Powered Agent
- Autonomous Agents (LLMs). Updated daily

Thanks

yogeshkulkarni@yahoo.com