Weekly Update

Red Team

Agenda

1. Weekly Code Update
2. Plan for This Week
3. Future Update Discussions
4. Why Autogen?
5. Insight: Prompt Evolution

Weekly Code Update

Update Note: v3.1.1a

- Removed the “language” feature
- Changed an algorithm in the Cosine Similarity (TFIDF →BERT)
- Modified the benchmark_summary format
- Fixed some bugs

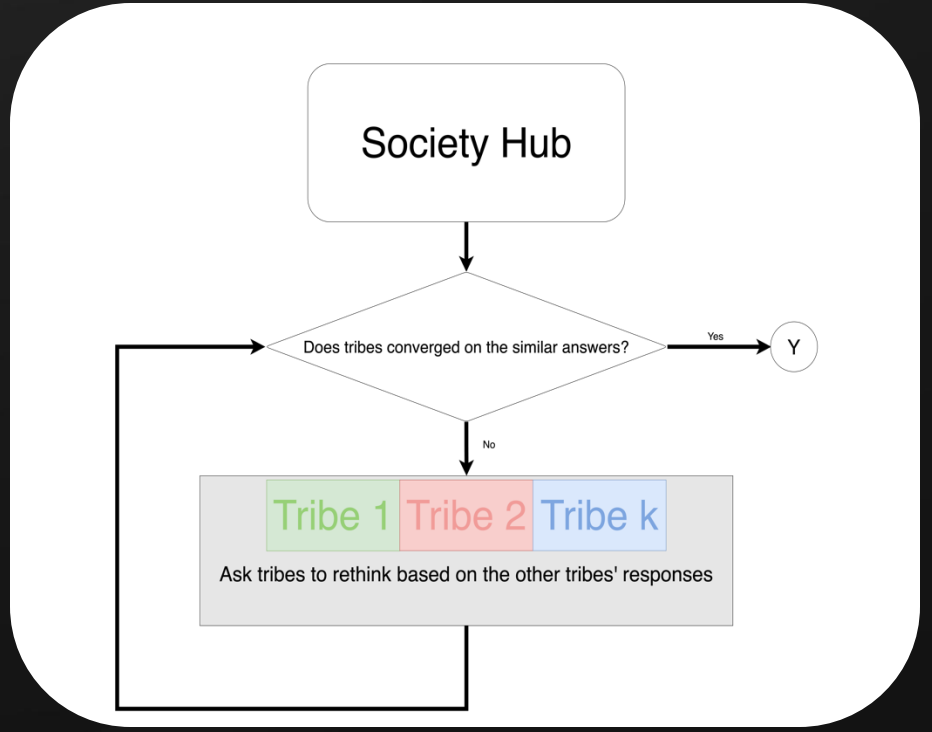
Update Note: v3.1.2

- Added a new functionality:

Leader of the group of agents (Planet) debates in Society Hub

New Class: Delegate

New Class: InterplanetaryCommittee



Update Note: v3.1.2

- New Class: Delegate

Each planet sends one delegate to the committee.

Delegates present their planet’s consensus answers in S-Hub.

- New Class: InterplanetaryCommittee

Planets now communicate with each other through delegates.

Make a Consensus Answer in Planet Scale

Consensus Achievement and Delegate Actions

- Achieved:

The Planet provides the consensus answer to the I-Committee

- Not Achieved:

The Planet skips the submission to the I-Committee

Majority Vote in the I-Committee

- Delegates vote on the consensus answers presented.
- A majority vote determines the final answer.

- When the vote differ

Re-debate the issue in Planet Level

- Planet: Consensus

Receive the other planet's answers and debate

- Planet: Not Consensus

Just debate without the other planet's answers

Plan for This Week

Future Update Discussion

Why Autogen?

Why Autogen?

Autogen is an agentic AI Framework

≒ OpenAI/Swarm, CrewAI, LangChain

- Active Community
- Highly Detailed Documents/Cookbooks
- Complex Agent Function
- Complex Multi-Agent Orchestration (GroupChat, SocietyOfMind)
- Automatic Handoff/Termination Based on Prompt/Context of Chat
- Tool (Python Code Executor)
- Local LLM with vLLM/Ollama
- Advanced Logging

Complex but Simple

Active Community

Autogen has one of the strongest community

Actively Developed

Preview v0.4

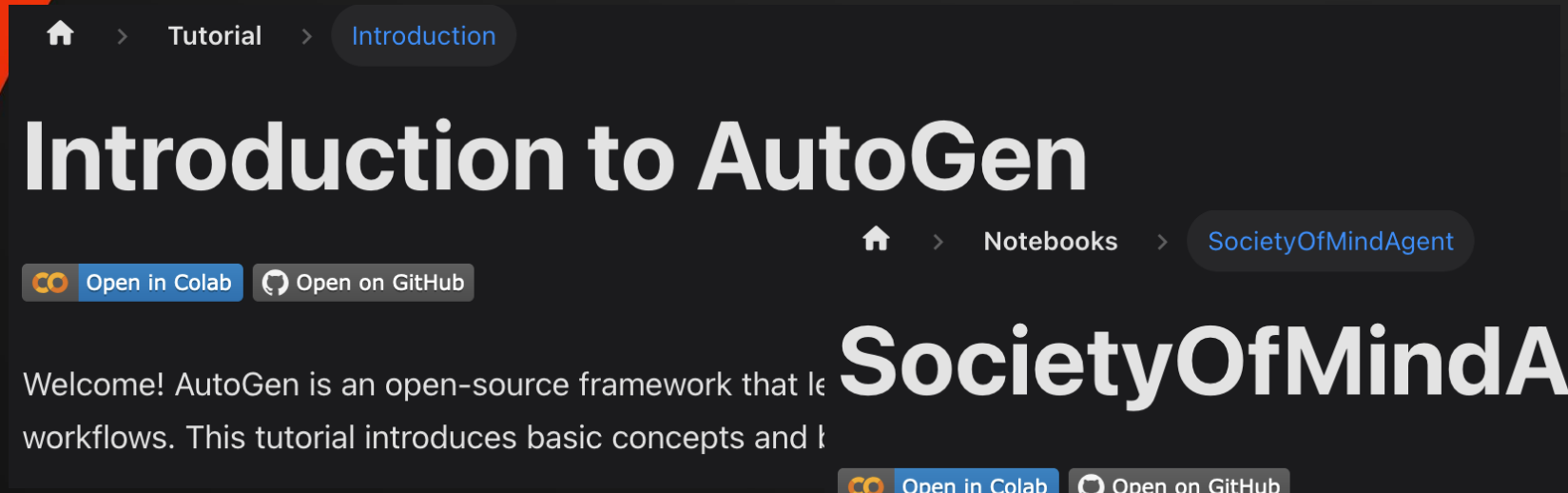
A new event driven, asynchronous architecture for AutoGen

New version under construction

Autogen can dateback to 2020

Highly Detailed Documents/Cookbooks

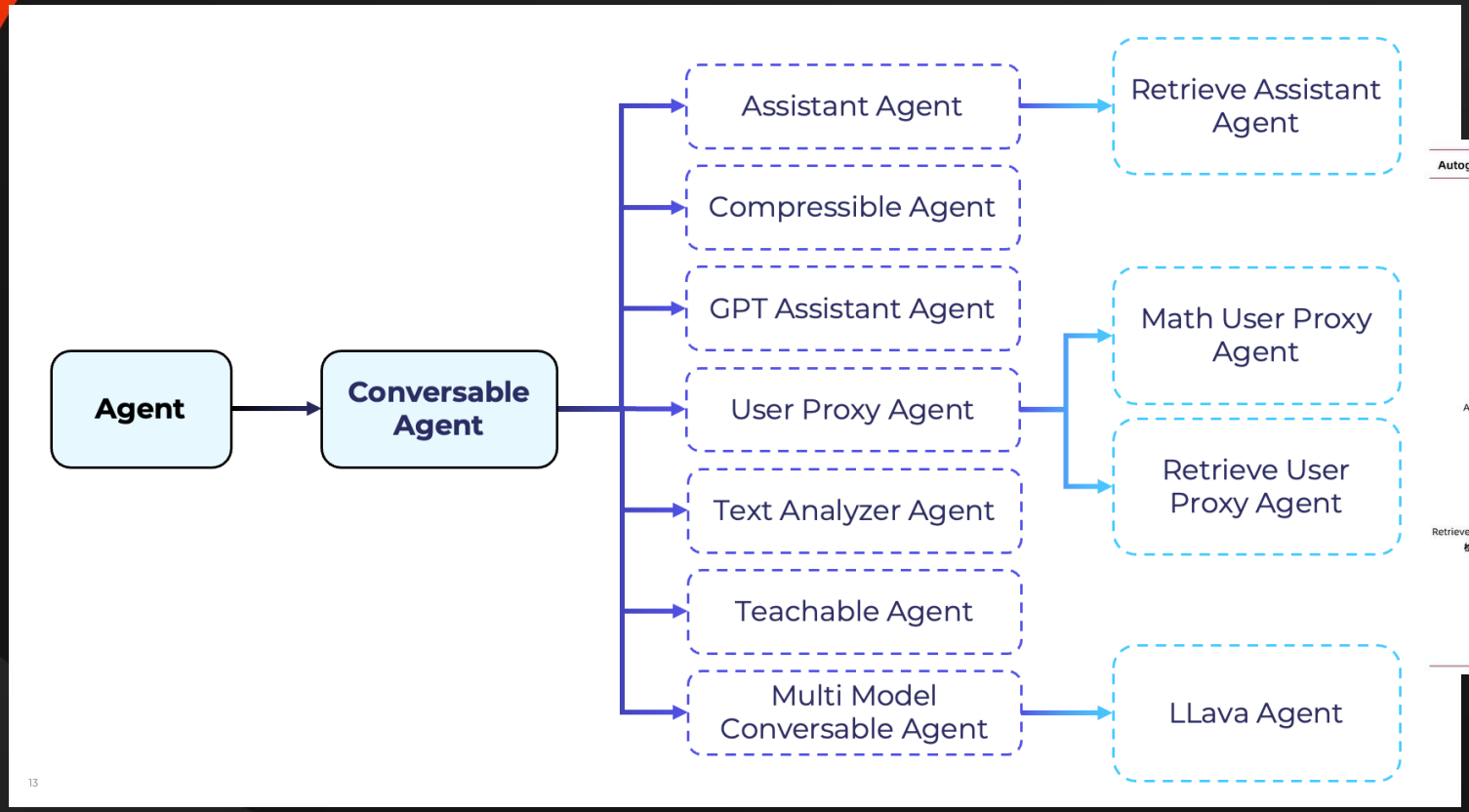
Autogen Offers Highly Complex API = Hard

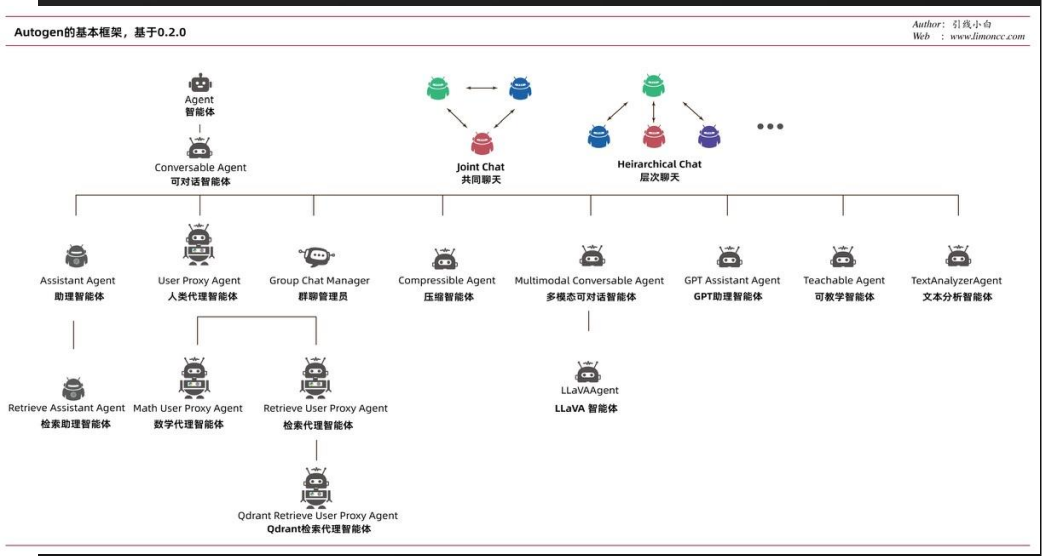


Very Detailed Document/Cookbook with Notebook (.ipynb)

Complex Agent Function

Types of Agents by Purpose





Multi-Agent Orchestration - GroupChat

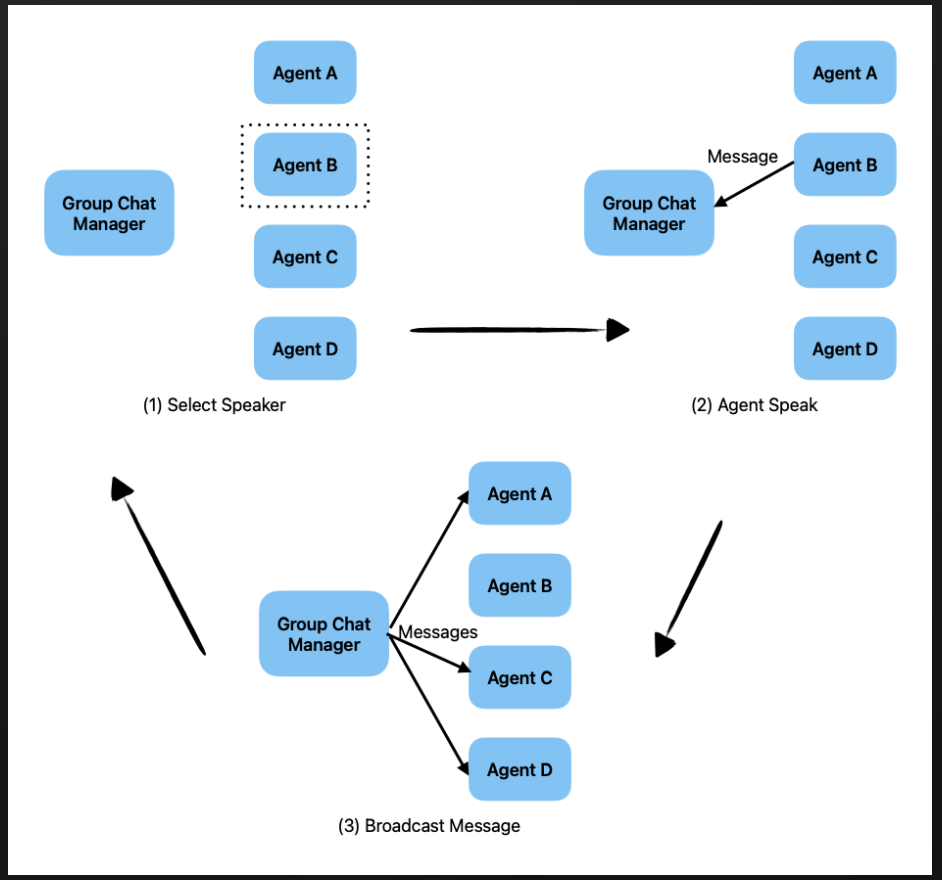
Multi-Agent Management by Default

1. Spawn Agents
2. Assign the Agents into a GroupChat

- Automatically Broadcast

GroupChatManager: LLM Agent

= Act according to the prompt



Multi-Agent Orchestration - SocietyOfMindAgent

SOMA = SocietyOfMindAgent

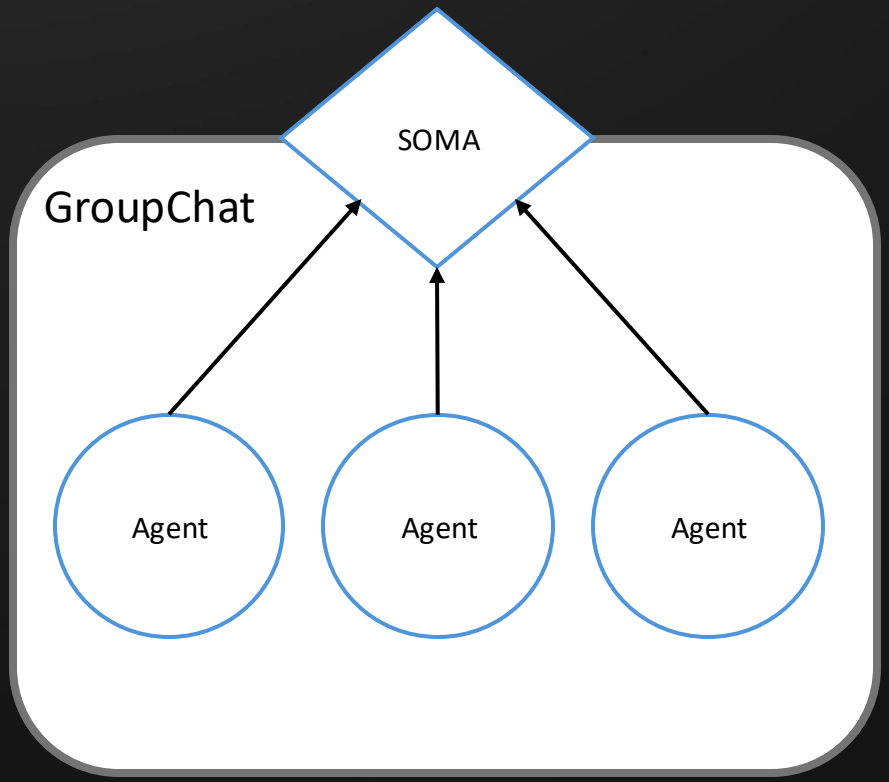
Wrap the GroupChat as an ConversableAgent

SOMA behave like a simple agent

GroupChat acts as inner monologues

Output is a "*Consensus Answer*"

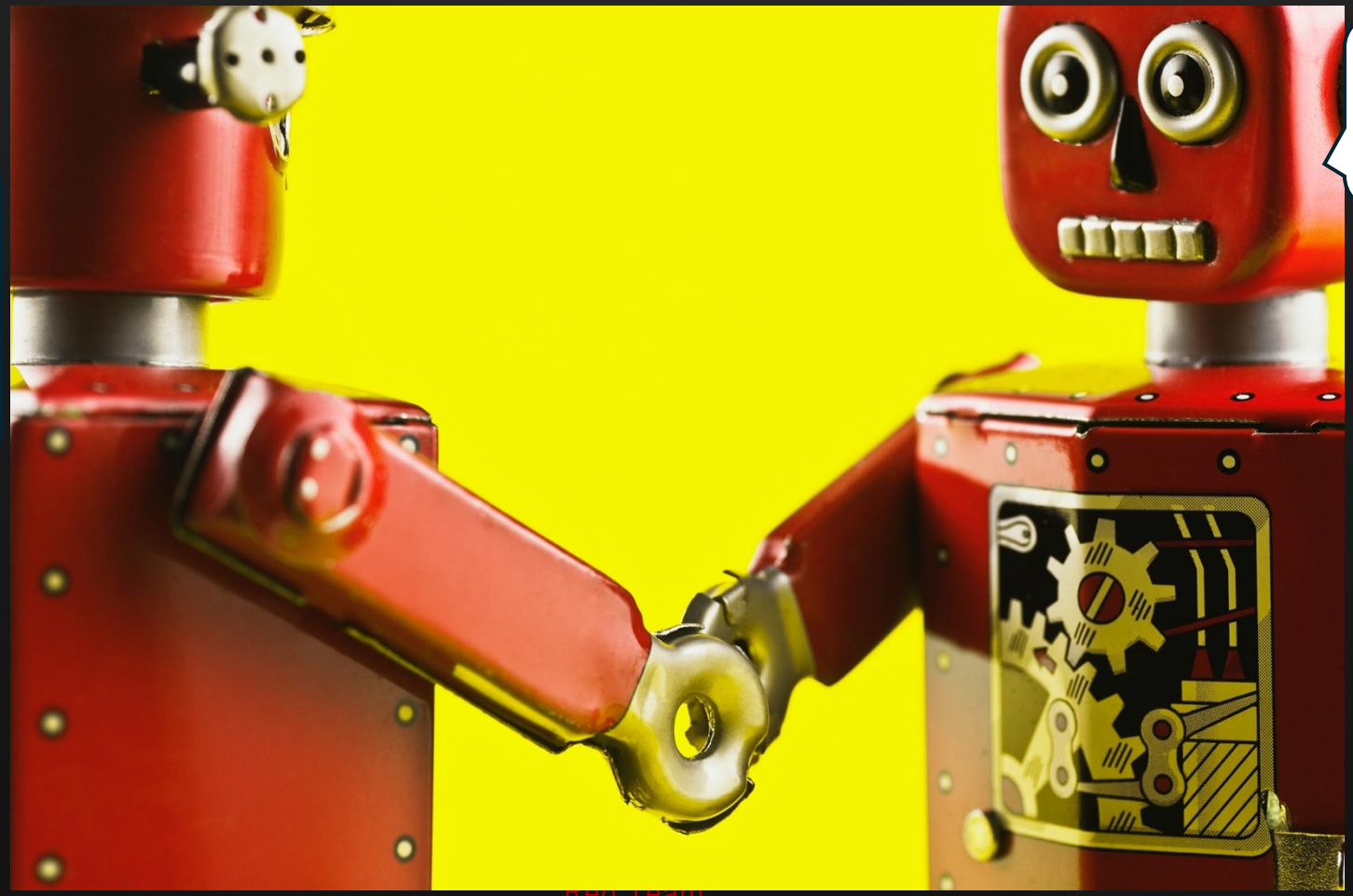
Delegate would be it



Automatic Handoff/Termination Based on AgentName/Prompt/Context

Handoff can be done by Agents

I think it is converged:
It's your turn Aggregator



k

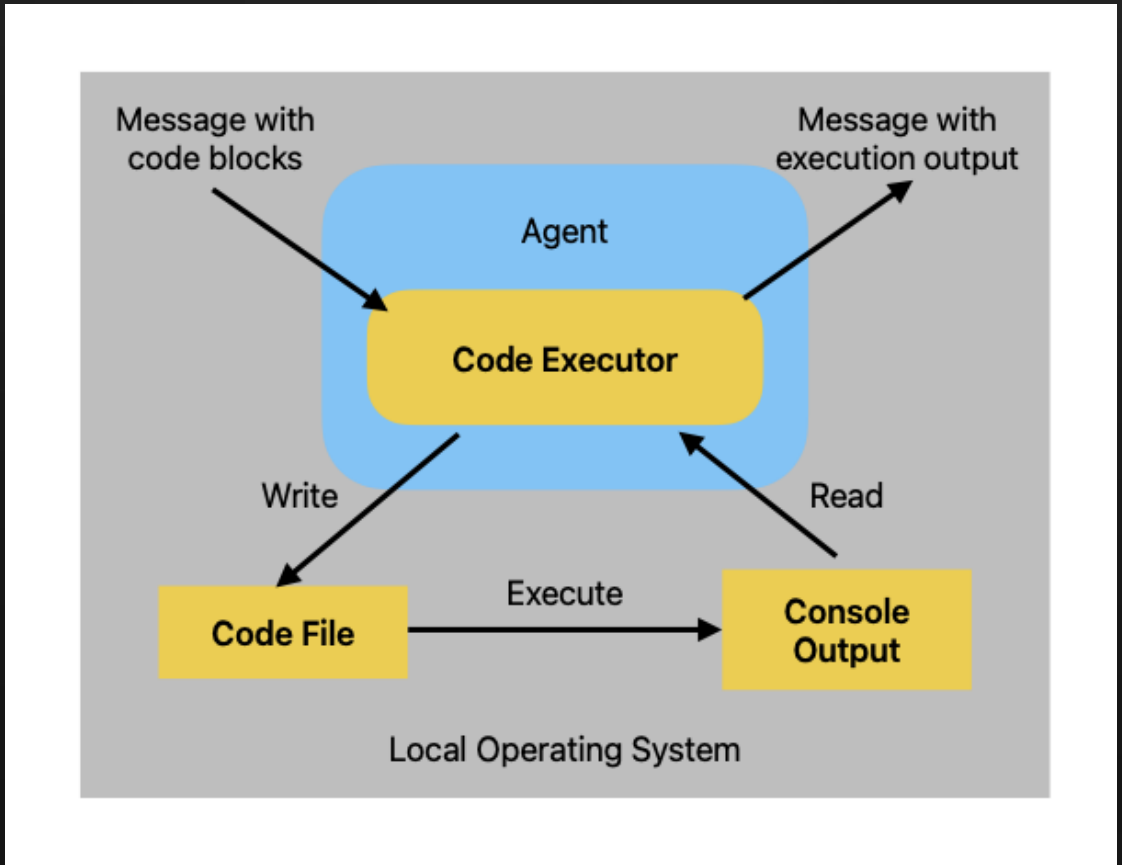
Red Team

Termination of chat can be done by Agents

The response is enough to answer the query

Tool (Python Code Executor)

Powerful Code Execution



Why Autogen?

Tool (Python Co :or)

DANGER

Executing LLM-generated code poses a

Local LLM with vLLM/Ollama

vLLM and Ollama serve as local LLM API Endpoint according to the OpenAI API Rule

Ollama

133 Model Family

User-Friendly Interface

= Personal Use

HF Models with GGUF File

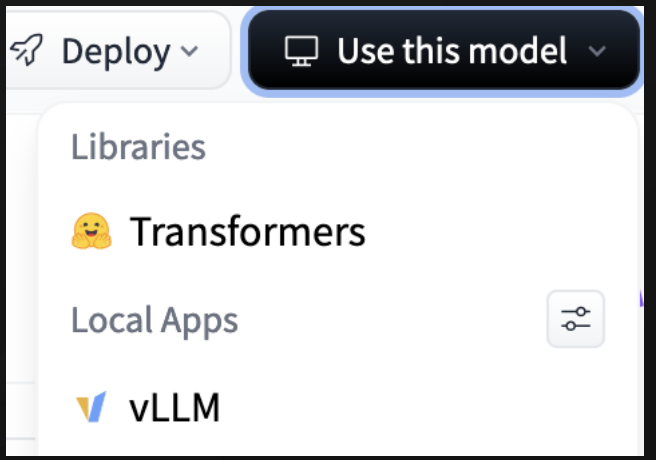
vLLM

59 Officially Support+ HF Models

Fast, Handle Large Request

= Enterprise Use

Natural Integration with HuggingFace

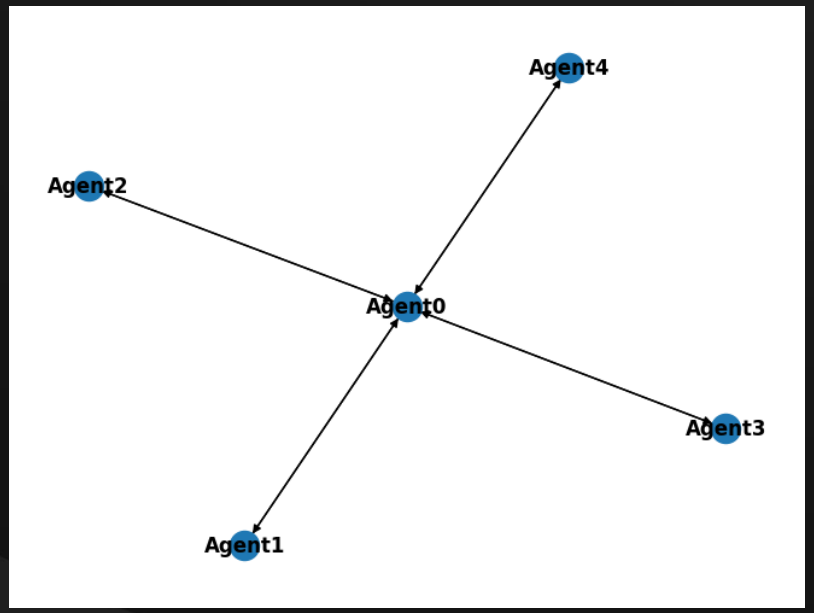


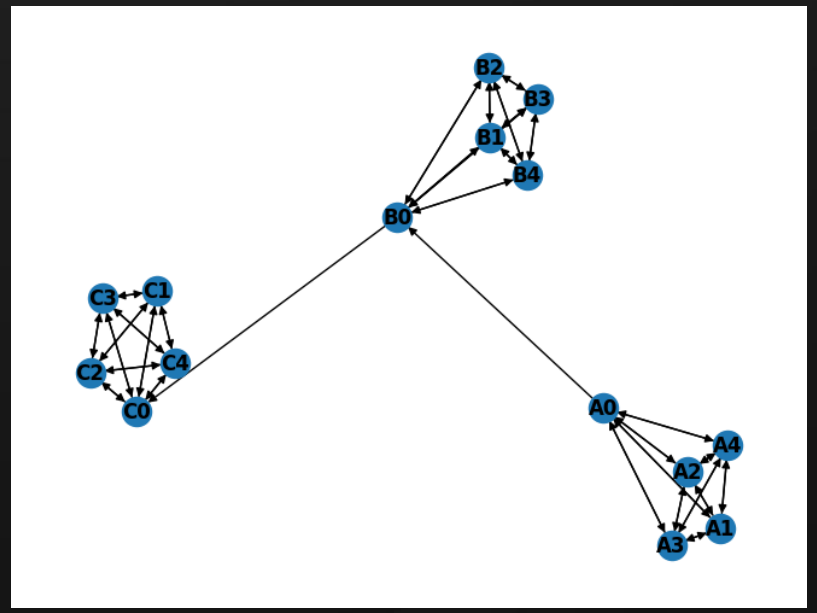
Advanced Logging

- “Runtime Logging” with .log File/SQLite

More Efficient Debugging, Examination

- visualize_speaker_transitions_dict visualize the topology





Caviat of Autogen

- Learning Cost

It definitely harder than Swarm

- More Complex

It offers many built-in classes/functions and customizability

- Automation

Some of the behavior by Agent is automated

= It may result in unexpected and unwanted

Prompting is more important than ever

Insight: Prompt Evolution

Quality of Output (Case of Zero-Shot)

LLM Output depends on the factors:

Model Capabilities: Number of Parameter, Architecture, Dataset, Weight

Prompt Design: System Prompt, User Prompt

Additional Data: RAG, Finetuning

Decoding Strategy: Temperature, Top-K, Top-P, Search Methods, Seed

Randomness: Random Internal State of Model

Environmental Factor: Hardware, Software, Network

Bias: Language Bias, Cultural Bias

Temporal Factor: Time Factor (Nowness of training datasets)

Quality of Output (Case of Zero-Shot)

$$Q = F(M, P, D, S, R, E, B, T)$$

Notation:

Q = LLM Output Quality, M = Model Capabilities, P = Prompt Design, D = Additional Data,

S = Decoding Strategy, R = Randomness, E = Environmental Factor, B = Bias,

T = Temporal Factor

Quality of Output (Case of Zero-Shot)

$$Q = F(\bar{M}, P, D, S, \bar{R}, \bar{E}, \bar{B}, \bar{T})$$

Notation:

Q = LLM Output Quality, M = Model Capabilities, P = Prompt Design, D = Additional Data,

S = Decoding Strategy, R = Randomness, E = Environmental Factor, B = Bias,

T = Temporal Factor

Prompt Breeder

Google DeepMind

PROMPTBREEDER:
SELF-REFERENTIAL SELF-IMPROVEMENT
VIA PROMPT EVOLUTION

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, Tim Rocktäschel

Google DeepMind
{chrisantha,dylski,henrykm,osindero,rocktaschel}@google.com

$$\mathrm{P} = \tau + p + M$$

Mutate Using LLM

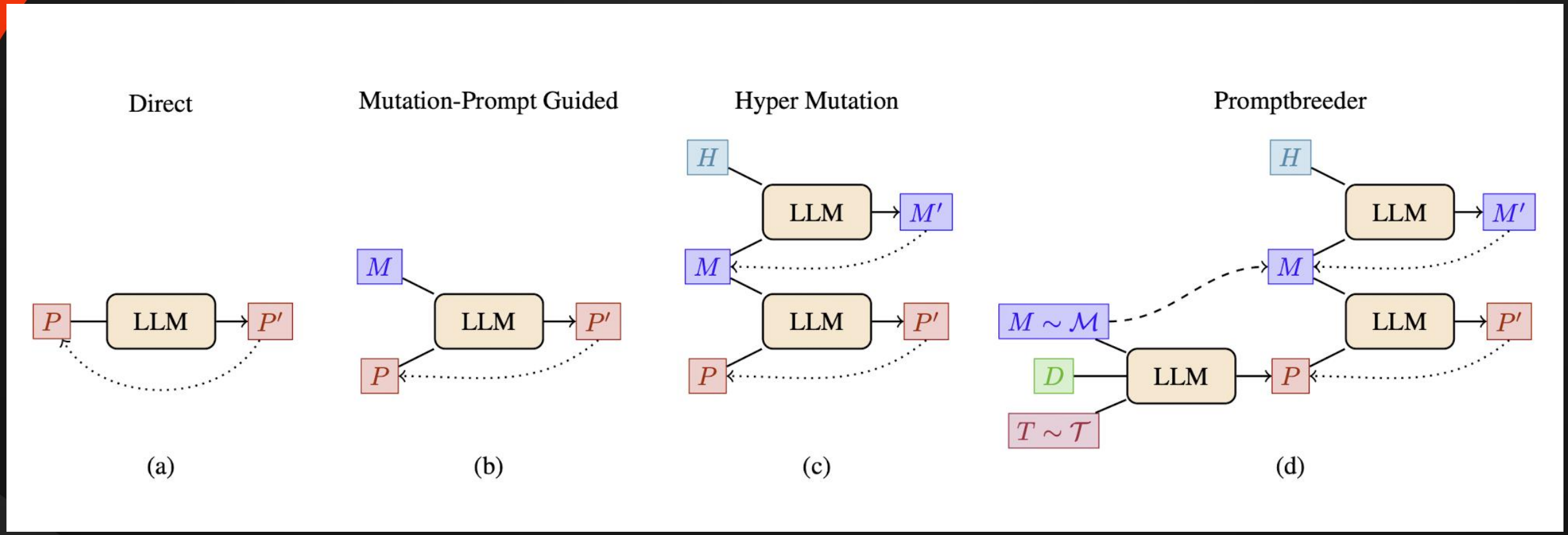
$$\mathrm{P}' = LLM(\tau + P + M)$$

τ=Thinking Style (i.e.: Chain of Thought)

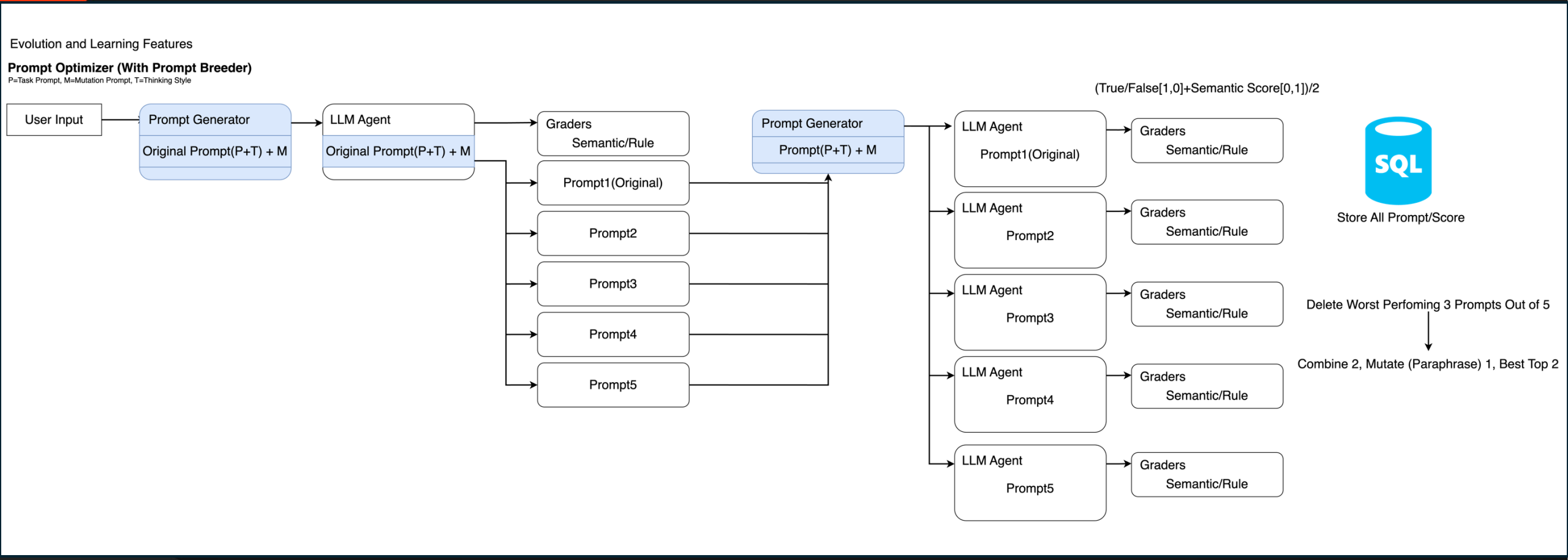
p=Task Prompt, P=Prompt,

M=Mutation Prompt (Defines the way to mutate P)

Prompt Mutation



Prompt Optimizer Overview (Alpha)



How to Calculate the Score

$$Score = \frac{Semantic + Rule}{2}$$

$$Score = [0,1]$$

Notation:

$Semantic$ is Score of Semantic Grading (Transformed Cosine Similarity Label/Output)

Cosine Similarity [-1,1] -> Linear Transformed Cosine Similarity [0,1]

$Rule$ is Boolean Value: B= {0, 1} $\begin{cases} if\ True: B = 1 \\ otherwise: B = 0 \end{cases}$

That‘s all for now