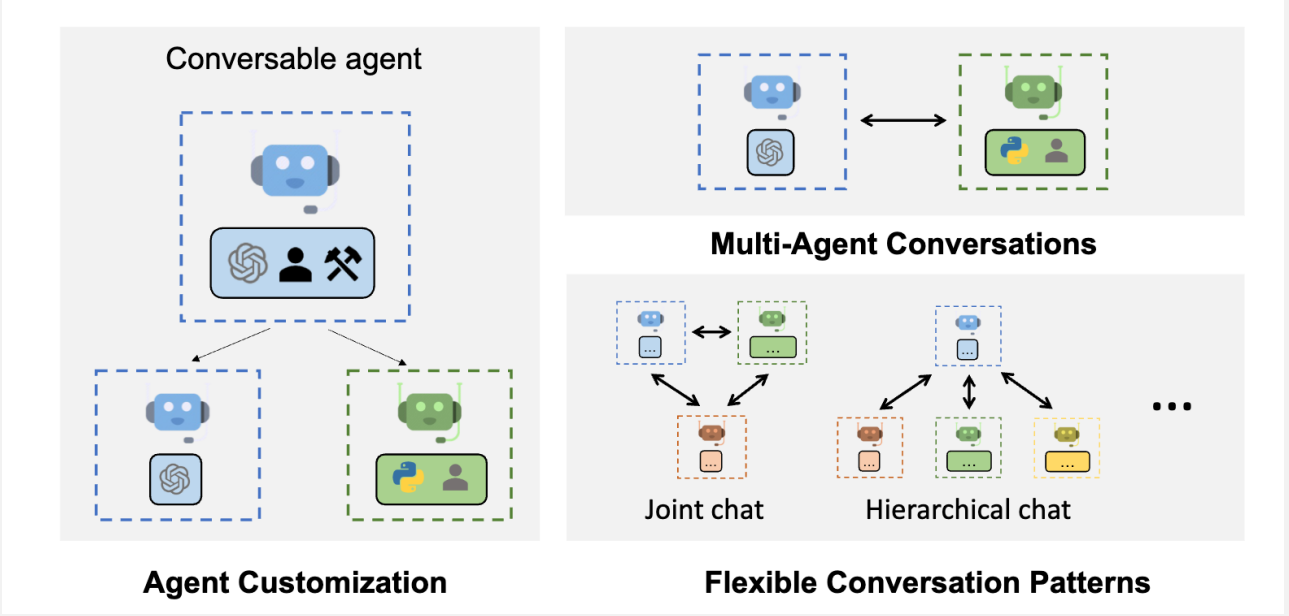
SEMINARIO SU

AUTOGEN

Massimo Belfiore X81001087 ~ DMI Unict
Seminario di Introduzione al Data Mining

Cos'è Autogen?

- Open-source programming framework
- Multi-Agent conversations



Autogen concetti chiave 

- Conversable Agents
- Conversation programming

Esempio di configurazione di un agente

- Settaggio dei parametri
  - human_input_mode (NEVER, ALWAYS, TERMINATE)
  - max_consecutive_auto_reply
- Human in the loop oppure conversazione tra agenti completamente autonoma.

```
agent = ConversableAgent(
  name="librarian",
  system_message=SYSTEM_PROMPT,
  human_input_mode="NEVER",
  llm_config={
    "config_list": config_list,
    "timeout": 180,
    "temperature": 0.2}
```

```
user = RetrieveUserProxyAgent(
    name="user",
    human_input_mode="NEVER",
    max_consecutive_auto_reply=3,
    retrieve_config={
        "task": "qa",
        "docs_path": None,
        "vector_db": chroma,
        "embedding_model": "all-MiniLM-L6-v2",
        "get_or_create": False,
    },
    code_execution_config=False,
)
```

```
import autogen

config_list = autogen.config_list_from_json(
    "OAI_CONFIG_LIST",
    filter_dict={
        "model": {
            "gpt-4",
            "gpt4",
            "gpt-4-32k",
            "gpt-4-32k-0314",
            "gpt-4-32k-v0314",
            "gpt-3.5-turbo",
            "gpt-3.5-turbo-16k",
            "gpt-3.5-turbo-0301",
            "chatgpt-35-turbo-0301",
            "gpt-35-turbo-v0301",
            "gpt",
        }
    }
)
```

Alcune funzionalità offerte da Autogen

- Autogen offre diversi pattern di conversazione sia base che avanzati (two-agent chat, sequential chat, group chat, nested, mix, constraints);
- Definizione di tools eseguiti dall'utente umano o dagli agenti (se configurati adeguatamente);
- Possibilità per un agente di eseguire il codice autogenerato dagli agenti creati tramite Autogen, oppure fornitogli da un utente umano;
- Autogen facilita l'uso dei vari LLMs. Inoltre è possibile associare una configurazione llm diversa per ogni agente se vogliamo.

Applicazioni multi-agent

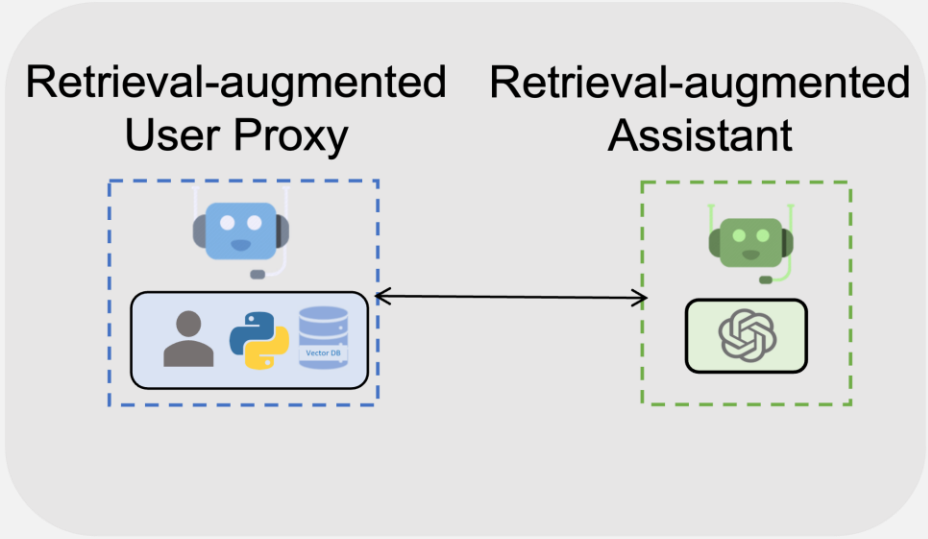
- Un approccio intuitivo adottato da Autogen per aumentare la potenza degli agenti è quello di utilizzare più agenti che cooperano;

- Autogen arrichisce le nostre soluzioni in Python, Java, o in qualsiasi altro linguaggio con il supporto multi-agent, facilitando lo sviluppo di LLM applications;

- Di fatti Autogen risulta molto efficace in applicazioni di dominio matematico, programmazione (per esempio leetcood), intrattenimento, question-answering, ricerca operativa, ecc…

Applicazioni multi-agent esempio #1 (Retrieval-Augmented Code Generation and Question Answering)

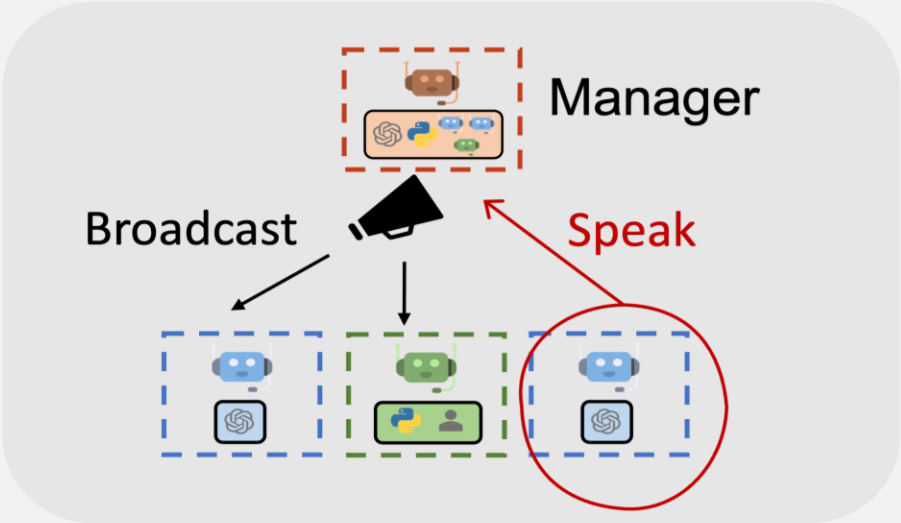
- Two-agent chat
- RAG system



A2. Retrieval-augmented Chat

Applicazioni multi-agent esempio #2 (Dynamic Group Chat)

- Multi-agent dynamic group chat with role-play speaker selection policy.
- 4-agent group chat system (admin, engineer, executor, critic).



A5. Dynamic Group Chat

Advanced features

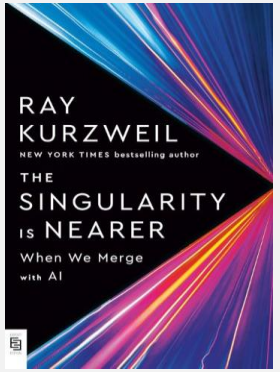
- RAG agents
- Teachable Agents
- Templating
- Logging
- Tune inference parameters

Confronto con alcuni sistemi single-agent e multi-agent famosi

- **Autogen vs AutoGPT**
- **Autogen vs Langchain Agents**
- **Autogen vs BabyAGI**
- **Autogen vs CAMEL**

AutoGen differisce dai sistemi a single-agent citati supportando multi-agent LLM applications.

BabyAGI e CAMEL supportano soltanto modelli di conversazione statica.



Summary dei vantaggi e svantaggi

Pro:

- Allow human involvement;
- Modularity;
- Both static and dynamic patterns;
- Programmability;
- Powerful and easy;
- Collaborative/adversarial agent interactions.

Contro:

- Ethical considerations.
- Cybersecurity threats.

Installazione di Autogen su Linux, Windows e MacOS

- Tramite conda environment

  AutoGen requires **Python version >= 3.8, < 3.13**.

  - Create and activate:
    ```
    conda create -n autogen python=3.11
    conda activate autogen
    ```
  - To deactivate later, run:
    ```
    conda deactivate
    ```
  - **`pip install pyautogen`**

- Tramite virtual environment: vedi https://microsoft.github.io/autogen/0.2/docs/installation/

Concetti e definizioni

- Agente: componente di AI specializzato in un compito su cui è stato istruito attraverso indicazioni testuali (di tipo tecnico o comportamentale) dette prompt. Esso ha accesso ai tools, può inviare e ricevere messaggi, e può anche generare una risposta utilizzando modelli(e.g. un LLM), human inputs, tools, o una combinazione di questi.

- Tools: funzioni scritte da un utente umano in un linguaggio di programmazione, che gli agenti possono utilizzare per eseguire tasks.

- LLMs: modelli di AI solitamente (ma non sempre) derivati dall' architettura transformer, progettati per interpretare o generare codice, linguaggio, ecc… Gli LLMs sono language models che possono essere o autoregressive, o autoencoding, oppure una combinazione fra questi (autoregressive+autoencoding).

- Vector Database: tipo di database pensato per memorizzare e recuperare dati rappresentati da vettori in modo rapido. Esso è molto utile per la memorizzazione degli embeddings generati da un LLM.

Sitografia/riferimenti

- https://microsoft.github.io/autogen/0.2/docs/Getting-Started -- documentazione ufficiale Autogen.
- https://www.databricks.com/it/product/machine-learning/large-language-models
- https://blog.dragonscale.ai/architectures-for-ai-agents/
- https://aws.amazon.com/it/what-is/retrieval-augmented-generation/
- https://huggingface.co/docs/transformers/v4.47.1/agents
- https://huggingface.co/docs/transformers/v4.47.0/agents
- https://docs.agpt.co/ -- documentazione ufficiale AutoGPT.
- https://github.com/yoheinakajima/babyagi
- https://github.com/chroma-core/chroma
- https://microsoft.github.io/autogen/0.2/docs/Examples#automated-multi-agent-chat
- https://stackoverflow.blog/2024/06/06/breaking-up-is-hard-to-do-chunking-in-rag-applications/
- https://www.youtube.com/watch?v=Nbjci9CV87M

Grazie
per l'attenzione