Introduction to Python programming

Large Language Models

- Armando Camerlingo

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani

Introduzione alla Programmazione in Python

Armando Camerlingo

23/06/2025 - Lezione 6

AI Generativa (GenAI)

ITINERIS

- Un sottogruppo dell'AI che usa modelli per produrre testo immagini, video o altro a partire da dati già disponibili;
- Esempi di applicazioni GenAI sono ChatGPT, Gemini, DeepSeek (Large Language Model, LLM), Midjourney, DALL-E (text-to-image), Sora (text-to-video)....;
- Altamente customizzabili: i modelli possono essere trainati in modo da essere specializzati in un certo ambito;
- Per trainarli ci vuole una grandissima capacità di calcolo e tantissimi dati -> Riservato solo a poche realtà;

GenAI: Storia

ITINERIS

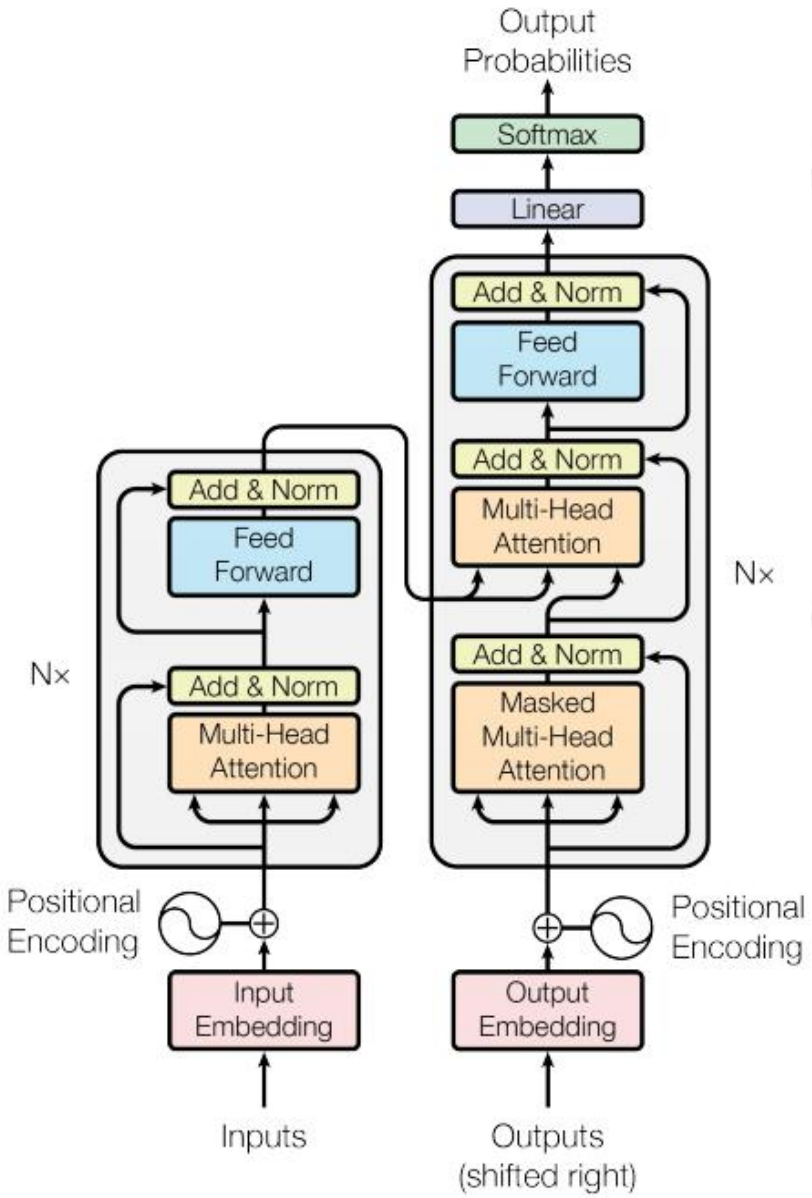
- Il primissimo algoritmo di GenAI risale all'inizio del 1900, ovvero la catena di Markov;
- Nei primi anni 70, viene creato AARON, un programma capace di creare immagini a partire da un set di istruzioni predefinite ;
- Nel 2014 dei primi modelli di Deep Learning vengono sviluppati per generare dei contenuti;
- 2020: viene pubblicato l'articolo "Attention is all you need" in cui viene presentato il Transformer;

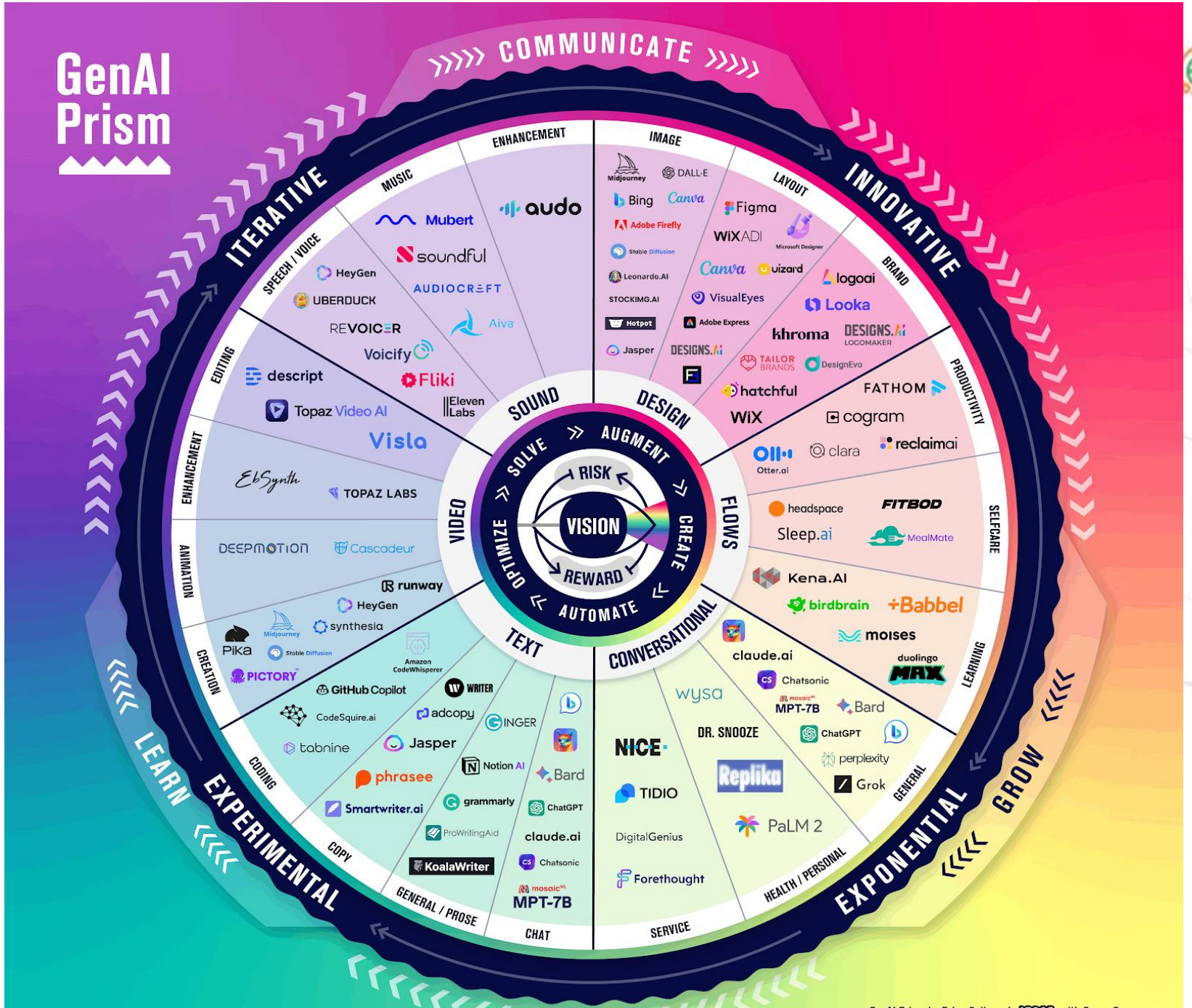
Attention is all you need!

ITINERIS

- La versione moderna del Transformer venne presentato da Google in questo articolo;
- Basato sul concetto di **tokenization**, ovvero il testo in input viene diviso in pezzi e convertito in rappresentazioni numeriche chiamate **token** ;
- Questi token vengono contestualizzati tra di loro tramite un meccanismo di **attenzione**, ovvero viene valutata la loro **importanza** nel testo dato in input;
- Sviluppato in primo luogo per la traduzione automatica, questo modello si è dimostrato eccellente anche in altri ambiti (testo, audio , immagini, robotica,...);

Attention is all you need!




GenAI Prism
COMMUNICATE
INNOVATIVE
EXPONENTIAL
GROW
EXPERIMENTAL
LEARN
ITERATIVE
IMAGE
LAYOUT
BRAND
PRODUCTIVITY
SELFCARE
LEARNING
GENERAL
HEALTH / PERSONAL
SERVICE
CHAT
GENERAL / PROSE
COPY
CODING
CREATION
ANIMATION
ENHANCEMENT
EDITING
SPEECH / VOICE
MUSIC
ENHANCEMENT
DESIGN
FLOWS
CONVERSATIONAL
TEXT
VIDEO
SOUND
AUGMENT
CREATE
AUTOMATE
OPTIMIZE
SOLVE
RISK
VISION
REWARD
Midjourney
DALL·E
Bing
Canva
Adobe Firefly
Stable Diffusion
Leonardo.AI
STOCKIMG.AI
Hotpot
Jasper
Figma
WIXADI
Microsoft Designer
Canva
vizard
VisualEyes
Adobe Express
DESIGNS.AI
TAILOR BRANDS
DesignEvo
hatchful
WIX
logoai
Looka
khroma
DESIGNS.AI LOGOMAKER
FATHOM
cogram
clara
reclaimai
Otter.ai
headspace
FITBOD
Sleep.ai
MealMate
Kena.AI
birdbrain
+Babbel
moises
duolingo MAX
claude.ai
Chatsonic
MPT-7B
Bard
ChatGPT
perplexity
Grok
wysa
DR. SNOOZE
Replika
PaLM 2
NICE
TIDIO
DigitalGenius
Forethought
audo
Mubert
soundful
AUDIOCRAFT
Aiva
HeyGen
UBERDUCK
REVOICER
Voicify
Fliki
Eleven Labs
descript
Topaz Video AI
Visla
EbSynth
TOPAZ LABS
DEEPMOTION
Cascadeur
runway
HeyGen
synthesia
Pika
Midjourney
Stable Diffusion
PICTORY
Amazon CodeWhisperer
GitHub Copilot
WRITER
CodeSquire.ai
adcopy
tabnine
Jasper
phrasee
Smartwriter.ai
INGER
Notion AI
grammarly
ProWritingAid
KoalaWriter
Bard
ChatGPT
claude.ai
Chatsonic
MPT-7B
GenAI Prism by Brian Solis and JESS3 with Conor Grennan

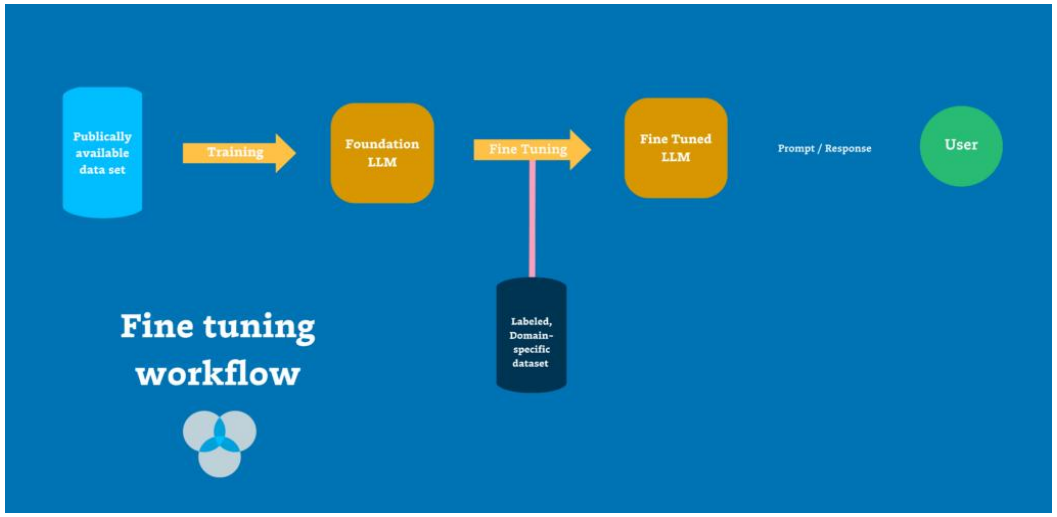
Come utilizzare la GenAI

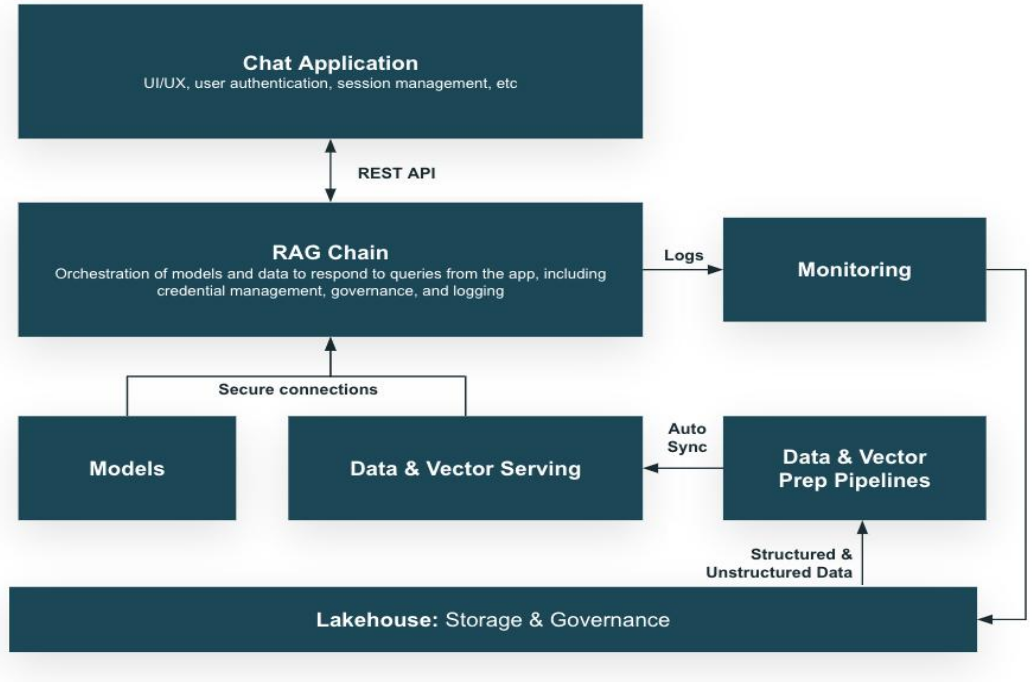
ITINERIS

- Nella maggior parte dei casi, l'utente non ha bisogno di far girare i modelli sui propri dispositivi. Si collega tramite API a modelli disponibili tramite internet;
- Le richieste dell'utente vengono inviate per essere processate da un modello per uso **generico** e l'output viene restituito;
- In molti strumenti si può definire un **contesto** tramite il prompt di input stesso o fornendo accesso a documenti aggiuntivi che possono essere d'aiuto per la risposta;
- A volte questo non può essere possibile o non sufficiente, per esempio documenti privati, grande mole di documenti o documenti troppo complessi;

Training vs Fine Tuning vs RAG

- Per avere un modello adattato completamente allo scopo voluto, l'ideale sarebbe ri-effettuare il training. Però sono necessarie moltissime risorse computazionali e non solo ;
- Ci sono altri metodi per rendere il modello più specifico, ovvero il Fine Tuning o la Retrieval Augmented Generation;





Fine Tuning

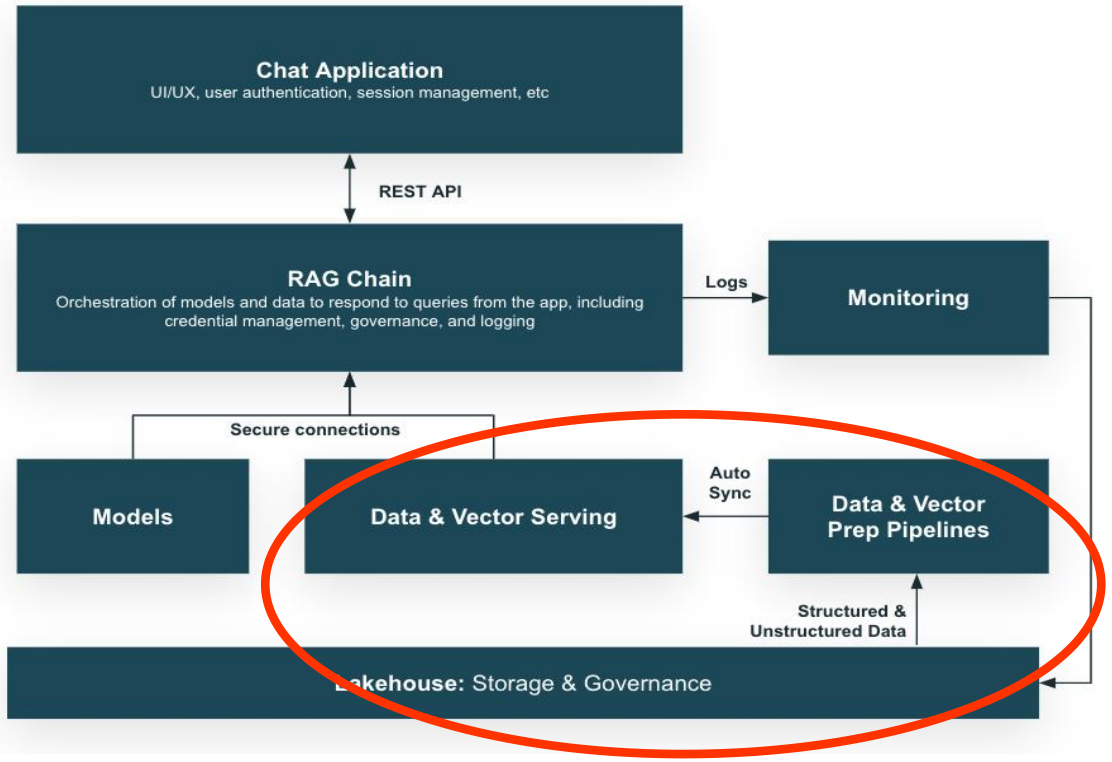
Viene effettuato un training su un dataset più piccolo e specializzato per regolare i parametri del modello generale

- Si ottengono risultati e performance migliori;
- Ci vogliono risorse anche in questo caso per effettuare il tuning;

Retrieval Augmented Generation

ITINERIS

Si dà la possibilità al modello di accedere ad un database contenente informazioni curate e aggiornate riguardo alle casistiche che vogliamo analizzare.



- Non c'è effettivo re-training del modello -> Meno risorse;
- Più specificità per le richieste dell'utente;
- Sicurezza e privacy del dato;
- Però è un processo complesso, composto da più fasi da pianificare e eseguire correttamente

LLM in Python

ITINERIS

Anche se sono disponibili moltissimi strumenti per utilizzare la genAI in situazioni no code/low code, sono presenti anche molteplici pacchetti per utilizzarli in script di codice, come Python.

Oggi vedremo un esempio di un modello relativamente semplice, il TinyLlama-1.1B di Meta, e come utilizzarlo.

TinyLlama è uno dei modelli più piccoli con 1.1 miliardo di parametri, mantenendo comunque una discreta precisione.

Meta

Hugging Face

LLM in Python

ITINERIS

Installiamo gli strumenti e i pacchetti necessari

```
!CMAKE_ARGS="-DLLAMA_BLAS=ON -DLLAMA_BLAS_VENDOR=OpenBLAS" pip install llama-cpp-python
```

```
!pip install huggingface-hub
```

```
!huggingface-cli download TheBloke/TinyLlama-1.1B-Chat-v1.0-GGUF tinyllama-1.1b-chat-v1.0.Q4_K_M.gguf --local-dir . --local-dir-use-symlinks False
```

LLM in Python

Configuriamo la nostra LLM

```
from llama_cpp import Llama
```

```
llm = Llama(model_path="./tinyllama-1.1b-chat-
v1.0.Q4_K_M.gguf",       #modello che vogliamo usare
            n_ctx=2048, #lunghezza della sequenza di tokens da
                        # usare, più è lunga più risorse ci
                        # vogliono
            n_threads=8, #numero di CPU threads
            n_gpu_layers=35 #se è presente una GPU numero di
                            #  layers da derogare
            )
```

LLM in Python

Creiamo un prompt

ITINERIS

```
llm.create_chat_completion(
      messages = [
        {
          "role": "system",
          "content": "You are story writing assistant"

        },
        {
          "role": "user",
          "content": "Write an extensive story about Life as a
young Adult"
        }
      ]
)
```

LLM in Python

Anche PyTorch offre opzioni per implementare LLM

```
!pip -qqq install bitsandbytes accelerate #per evitare messaggi di warning
```

```
import torch
from transformers import pipeline
```

```
pipe = pipeline("text-generation", model="TinyLlama/TinyLlama-1.1B-Chat-v1.0",
torch_dtype=torch.bfloat16, device_map="auto")
```

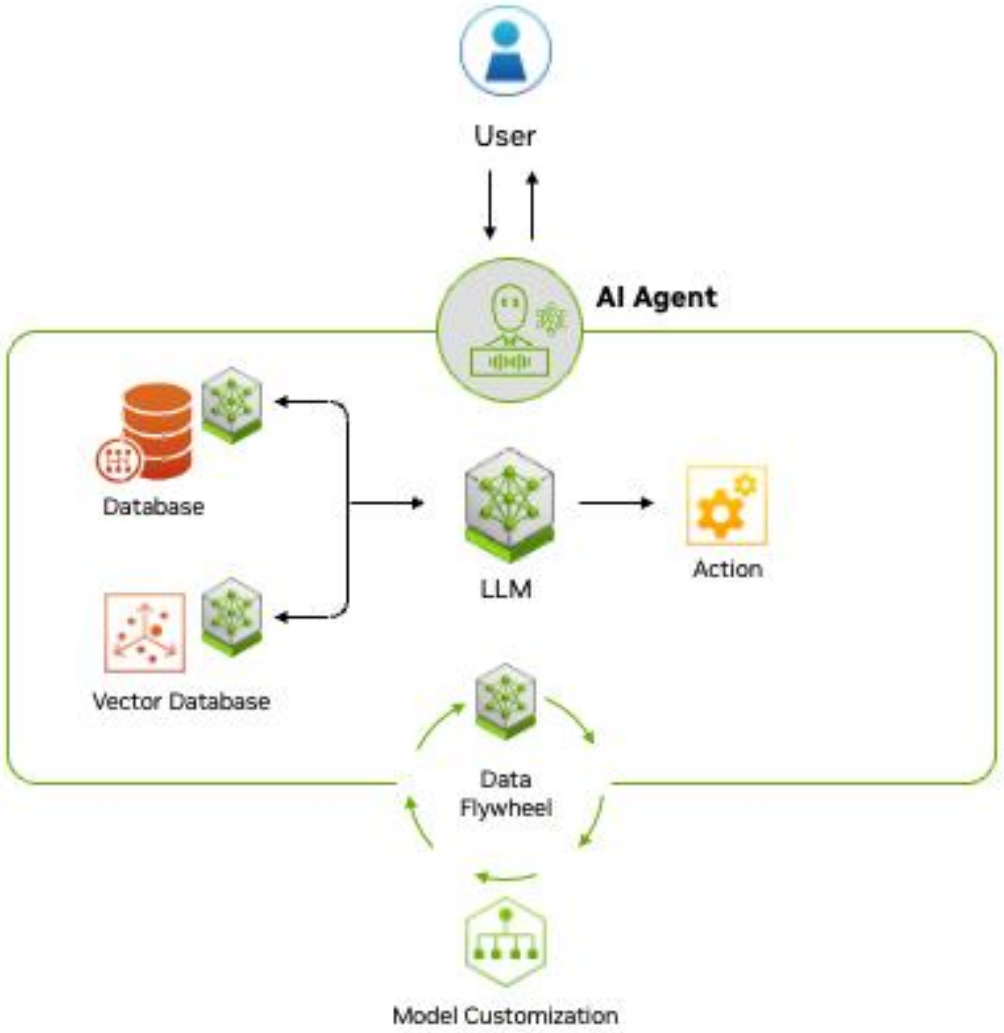
```
messages = [{
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",},
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]
prompt = pipe.tokenizer.apply_chat_template(messages, tokenize=False,
add_generation_prompt=True)
outputs = pipe(prompt, max_new_tokens=256, do_sample=True, temperature=0.7, top_k=50,
top_p=0.95)
print(outputs[0]["generated_text"])
```

Agenti AI

ITINERIS

- Le LLM forniscono un output che può essere utilizzato per prendere delle decisioni ed eseguire eventuali azioni;
- Si può delegare tutte queste operazioni ad un modello AI? La risposta sono gli **Agenti AI;**
- Ogni agente AI ha un compito specifico per cui è stato allenato ed ha capacità di eseguire azioni autonomamente (sotto supervisione limitata)

Agenti AI


User
AI Agent
Database
LLM
Action
Vector Database
Data
Flywheel
Model Customization

Conda

- Conda è uno strumento compatibile con Python per la gestione di pacchetti e ambienti;
- In Python un ambiente (virtuale, virtual environment o v-env) è uno spazio isolato sulla macchina in cui far girare script Python;
- Nell'ambiente si possono specificare diverse opzioni che non vanno ad interferire con l'installazione principale di Python o altri environment:
  - Si può scegliere che versione dell'interprete Python usare;
  - Si possono installare insiemi di pacchetti in modo separato dall'installazione generale. Ciò è utile se si desidera utilizzare versioni specifiche dei pacchetti per questioni di dipendenza e features deprecate;
- Utile per testing, organizzazione dei progetti e riproducibilità;
- Utilizzato molto in cluster a cui hanno accesso molti utenti in modo che ognuno abbia il proprio ambiente;
- Utile anche sul proprio pc per separare progetti diversi;

Conda

- Si può installare tramite la distribuzione Anaconda o le installazioni minimali MiniConda e MiniForge su Windows, Linux e MacOS
- Si può usare semplicemente con un’installazione Python classica o con IDE come VScode, Spyder, PyCharm....;
- Accessibile da linea di comando (terminal per Linux/MacOS, powershell o cmd prompt per windows;

Conda

Da terminale possiamo usare conda:

```
conda create -n <nome_ambiente> <versione Python> <pacchetti>
```

Per esempio, creiamo un ambiente con numpy e pandas

```
conda create -n mioambiente python numpy pandas
```

Per ottenere una lista degli ambienti disponibili:

```
conda info --envs
```

Per cambiare l'env corrente in uno nuovo:

```
conda activate
```

Conda su Colab

Su google colab si può utilizzare il pacchetto condacolab

```
!pip install -q condacolab
```

```
import condacolab
condacolab.install()
condacolab.check()
```

Su google colab non si possono creare ambienti, ma si possono installare:

```
!mamba install -q openmm
```

N.B. mamba è un altro gestore di pacchetti incluso in conda che lavora in parallelo;

Conda

Per installare nuovi pacchetti in un ambiente si può procedere in due modi:

1.
```
conda activate mioambiente
conda install matplotlib
```

2.
```
conda install --name mioambiente matplotlib
```

N.B. il ! ci vuole solo se si lavora in un notebook come Google Colab

Conda

- Un ambiente si può creare anche a partire da un file yaml in modo da avere un file di configurazione di riferimento:

```
name: mioambiente2
dependencies:
  - numpy
  - pandas
  - seaborn
  - scikit-learn
```

- Se il file si chiama env.yaml possiamo creare l'ambiente in questo modo:

```
conda env create -f env.yaml
```

Conda: altri comandi

- Con `compare` si possono confrontare i pacchetti di un ambiente con quelli di un file di configurazione:

```
conda compare -n mioambiente ./env.yaml
```

- Possiamo tornare all'installazione base con il comando `deactivate`

```
conda deactivate
```

- Si può anche esportare un ambiente in un file:

```
conda export --file <nome_file>
```

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani