AI e Large Language Models

Da Turing ai Transformers

Valerio Vaccaro

Satoshi Spritz Connect

24 Marzo 2026

Valerio Vaccaro 👀

- 💻 Sviluppatore Bitcoin ed Esperto Hardware
- 🔥 Contributore a progetti Bitcoin open source
- ⚠️ Appassionato di hardware fai-da-te (DIY)
- Ingegnere Bitcoin e Liquid presso Blockstream

Social

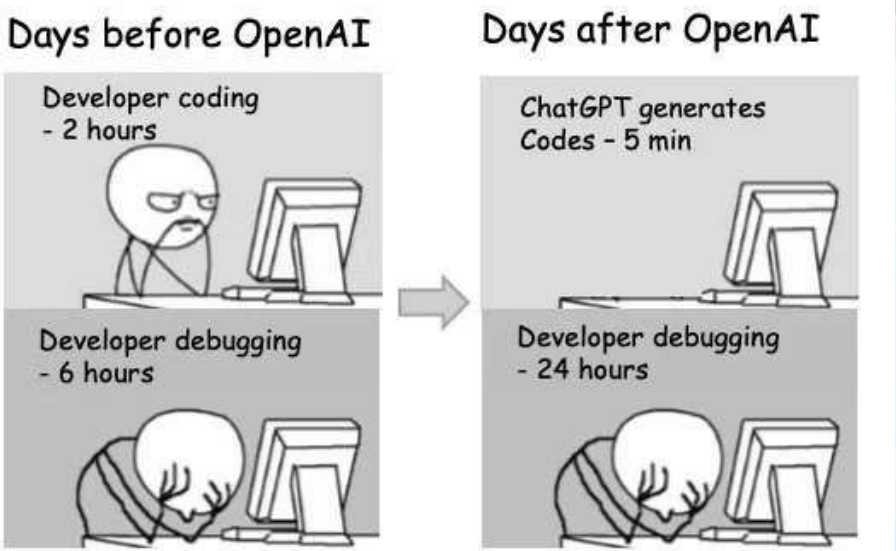
- 👨‍💼 **LinkedIn** linkedin.com/in/valeriovaccaro
- 🐙 **Github** github.com/valerio-vaccaro
- **Telegram** t.me/valeriovaccaro

Licenza

Questa presentazione è distribuita sotto la licenza Creative Commons CC BY-SA 4.0.

Le immagini utilizzate in questa presentazione sono proprietà dei rispettivi autori e sono incluse solo a fini educativi e illustrativi. Se usi questa presentazione, anche in parte, ricordati di citare l'autore originale.

May this presentation inspire you to become more self-sovereign!


Days before OpenAI
Developer coding
- 2 hours
Developer debugging
- 6 hours
Days after OpenAI
ChatGPT generates
Codes - 5 min
Developer debugging
- 24 hours

Sommario

- 📜 Breve storia dell'Intelligenza Artificiale
- 🤖 Cosa sono i Large Language Models (LLM)
- ⚙️ Come funzionano: token, embedding, attenzione
- 💻 Costruire un LLM minimale in Python
- 📈 Vantaggi dei LLM rispetto ad altri modelli AI

Breve storia dell'AI

Le origini (1940–1960)

- **1943** – McCulloch & Pitts: primo modello matematico di un neurone
- **1950** – Alan Turing pubblica *Computing Machinery and Intelligence* (Test di Turing)
- **1956** – Conferenza di Dartmouth: nasce il termine **Intelligenza Artificiale**
- **1958** – Frank Rosenblatt costruisce il **Perceptron**, la prima rete neurale addestrabile

Storia dell'AI: inverni e primavere

Cicli di entusiasmo e delusione

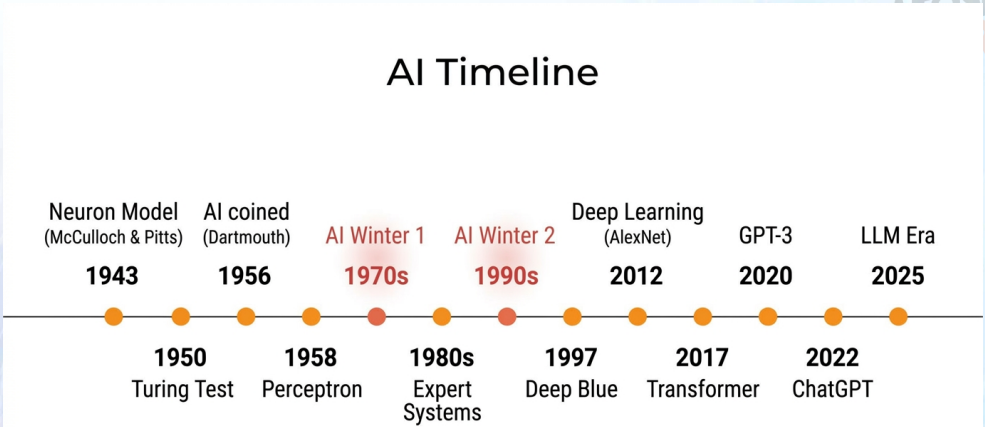
- 🔴 **Anni '70** – Primo **Inverno dell'AI**: potenza di calcolo limitata, promesse non mantenute
- 🚀 **Anni '80** – Boom dei Sistemi Esperti (AI basata su regole)
- 🔴 **Anni '90** – Secondo Inverno dell'AI: i sistemi esperti non scalano
- ✅ **1997** – IBM Deep Blue sconfigge Kasparov a scacchi
- 📈 **2012** – Rivoluzione del Deep Learning: AlexNet vince ImageNet

Storia dell'AI: l'era moderna

Dal deep learning agli LLM

- 🔥 **2014** – GAN (Generative Adversarial Networks) di Ian Goodfellow
- 🔑 **2017** – Google pubblica *Attention Is All You Need* (architettura Transformer)
- 🤖 **2018** – OpenAI rilascia GPT-1 (117M parametri)
- 🚀 **2020** – GPT-3 (175B parametri): emerge il few-shot learning
- 📈 **2022–2025** – ChatGPT, Claude, Llama, Gemini, DeepSeek: gli LLM diventano mainstream

Timeline dell'AI



Cosa sono gli LLM?

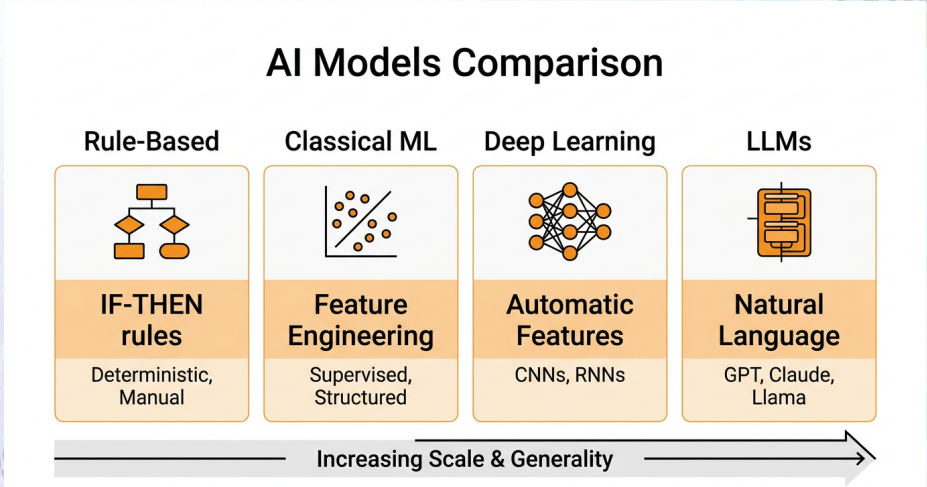
Large Language Models

- **Reti neurali** addestrate su enormi corpora di testo
- Obiettivo: predire il **prossimo token** in una sequenza
- Apprendono pattern del linguaggio: grammatica, fatti, ragionamento
- La scala conta: più parametri $\rightarrow$ più capacità

Intuizione chiave

*Un LLM è una **distribuzione di probabilità su sequenze di token**. Dato un prompt, genera testo campionando ripetutamente il token più probabile.*

Tipi di modelli AI



Come funzionano gli LLM: tokenizzazione

Dal testo ai numeri

- ⚙ Il testo viene diviso in **token** (parole, sotto-parole o caratteri)
- 🔑 Ogni token viene mappato a un ID intero tramite un **vocabolario**
- 📜 Esempio: `"Bitcoin è libertà"` → `[15496, 350, 8765]`

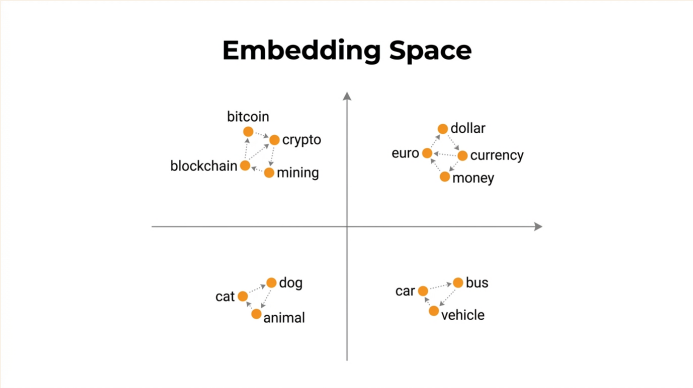
Tokenizzatori comuni

Tokenizzatore	Usato da	Dimensione vocabolario
BPE	GPT, LLaMA	50k–100k
WordPiece	BERT	30k
SentencePiece	T5, Gemma	32k–256k

Come funzionano gli LLM: embedding

Da token a vettore

- Ogni ID token viene mappato a un **vettore denso** (es. 768 o 4096 dimensioni)
- La **codifica posizionale** aggiunge informazioni sull'ordine dei token
- Parole semanticamente simili finiscono **vicine** nello spazio vettoriale



Come funzionano gli LLM: il Transformer

Meccanismo di attenzione

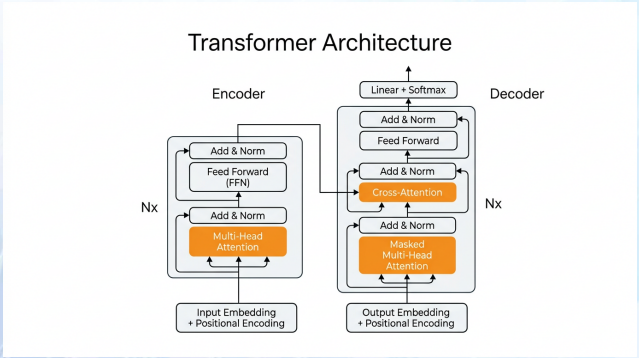
- **Self-attention**: ogni token presta attenzione a tutti gli altri
- Calcola matrici Query, Key, Value dall'input
- Punteggio di attenzione: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$

Blocco Transformer

1. Multi-Head Self-Attention
2. Add & Normalize
3. Feed-Forward Network (MLP)
4. Add & Normalize

Impilato 12–96+ volte a seconda della dimensione del modello.

Architettura Transformer



Come gli LLM generano testo

Generazione autoregressiva

- ⚙ Dati i token del prompt $[t_1, t_2, ..., t_n]$
- 🔑 Il modello predice la distribuzione di probabilità sul vocabolario per t_{n+1}
- 🔥 Campiona o sceglie il token più probabile
- 📜 Appende t_{n+1} alla sequenza e ripete

Strategie di campionamento

- **Greedy**: sceglie sempre la probabilità più alta
- **Temperature**: scala i logit per controllare la casualità
- **Top-k**: campiona dai k token più probabili
- **Top-p (nucleus)**: campiona dal più piccolo insieme con probabilità cumulativa $\geq p$

Costruire un LLM minimale in Python

Un modello bigramma a livello di carattere

Il modello linguistico più semplice: predice il prossimo carattere basandosi solo su quello corrente.

```
import random

# Corpus di addestramento
corpus = "bitcoin è libertà e la libertà è bitcoin"

# Costruisci tabella di transizione bigramma
bigrams = {}
for i in range(len(corpus) - 1):
    c1, c2 = corpus[i], corpus[i + 1]
    bigrams.setdefault(c1, []).append(c2)
```

LLM minimale: generazione

Campionamento dal modello bigramma

```python
def generate(bigrams, start, length=60):
    result = start
    current = start
    for _ in range(length):
        if current not in bigrams:
            break
        nxt = random.choice(bigrams[current])
        result += nxt
        current = nxt
    return result

print(generate(bigrams, "b"))
# Esempio: "bitcoin è libertà e la libertà è bi..."
```

LLM minimale: generazione

Campionamento dal modello bigramma

⚠ Questo è un modello a **2 stati** (carattere corrente $\rightarrow$ prossimo carattere). I veri LLM usano lo stesso principio ma con miliardi di parametri e finestre di contesto complete.

LLM minimale: codice completo

Esempio funzionante completo (20 righe)

```
import random

corpus = "bitcoin è libertà e la libertà è bitcoin"

# Apprendimento: conta le transizioni
bigrams = {}
for i in range(len(corpus) - 1):
    c1, c2 = corpus[i], corpus[i + 1]
    bigrams.setdefault(c1, []).append(c2)
```

LLM minimale: codice completo

Esempio funzionante completo (20 righe)

```python
# Generazione: campiona il prossimo carattere
def generate(bigrams, start, length=60):
    result, current = start, start
    for _ in range(length):
        if current not in bigrams:
            break
        current = random.choice(bigrams[current])
        result += current
    return result

for _ in range(3):
    print(generate(bigrams, "b"))
```

Da bigramma a Transformer

Scalare la stessa idea

Caratteristica	Modello bigramma	Classe GPT-4
Contesto	1 carattere	128k+ token
Parametri	circa 50	circa 1.8 trilioni
Dati di addestramento	1 frase	trilioni di token
Architettura	Tabella di lookup	Stack Transformer
Generazione	Scelta casuale	Softmax + campionamento

L'**idea fondamentale è identica**: apprendere pattern statistici e predire l'elemento successivo. La differenza è la scala e l'architettura.

Vantaggi dei LLM

Perché gli LLM superano l'AI tradizionale

- ☑ **General purpose**: un modello gestisce traduzione, codice, ragionamento, riassunti
- ☑ **Few-shot / zero-shot**: eseguono compiti con pochi o nessun esempio
- ☑ **Interfaccia naturale**: interazione in linguaggio naturale
- ☑ **Transfer learning**: addestramento unico, adattamento a molti compiti

LLM vs approcci tradizionali

Sistemi basati su regole

- ● Richiedono ingegneria manuale delle regole
- ● Fragili: falliscono sui casi limite
- ☑ Deterministici e spiegabili

ML classico (Random Forest, SVM, ecc.)

- ● Richiedono ingegneria delle feature
- ● Un modello per ogni compito
- ☑ Efficienti su dati strutturati piccoli

LLM

- ☑ Nessuna ingegneria delle feature necessaria
- ☑ Un modello, molti compiti
- ⚠ Richiedono grandi risorse di calcolo e dati

Confronto modelli AI

LLMs vs Traditional AI

	Flexibility	Scalability	Explainability	Data needed
Rule-Based Systems	Flexibility: 1/5	Scalability: 1/5	Explainability: 5/5	Data needed: Low
Classical ML (SVM, RF)	Flexibility: 3/5	Scalability: 3/5	Explainability: 3/5	Data needed: Medium
LLMs (GPT, Claude)	Flexibility: 5/5	Scalability: 5/5	Explainability: 1/5	Data needed: Very High

Quando NON usare un LLM

Gli LLM non sono sempre la scelta migliore

- 🔴 **Dati tabulari/strutturati**: XGBoost, Random Forest vincono ancora
- 🔴 **Bassa latenza in tempo reale**: l'inferenza LLM è lenta
- 🔴 **Logica deterministica**: usare algoritmi tradizionali
- 🔴 **Dataset piccoli**: il ML classico è più efficiente con pochi dati
- ⚠️ **Allucinazioni**: gli LLM possono generare affermazioni plausibili ma false

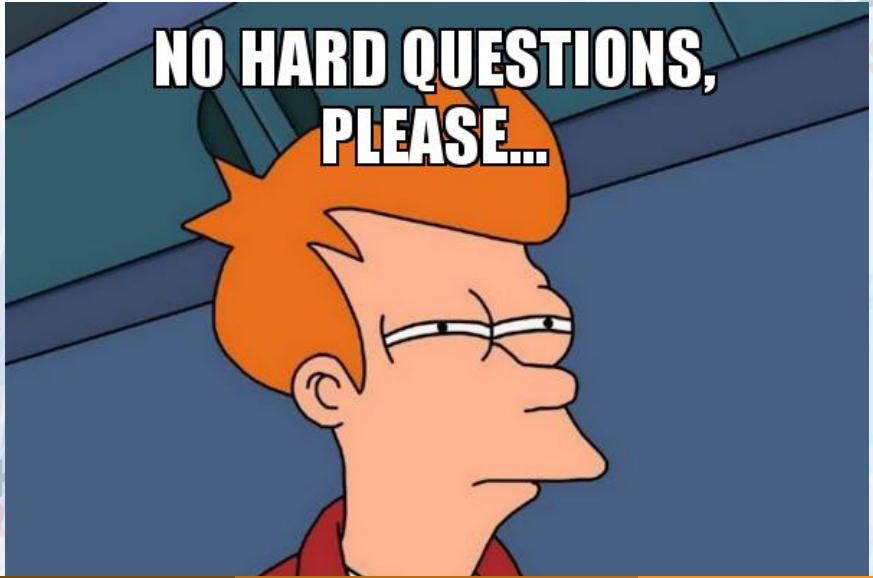
Regola pratica

*Usa un LLM quando il compito riguarda la **comprensione o generazione di linguaggio naturale**. Usa modelli tradizionali per tutto il resto.*

Conclusioni

- L'AI ha una lunga storia di cicli; siamo in una grande **primavera**
- Gli LLM sono reti neurali che predicono il prossimo token
- L'architettura Transformer (attenzione) è la svolta fondamentale
- Anche un modello bigramma di 20 righe cattura il principio base
- Gli LLM eccellono nei compiti linguistici ma non sono sempre lo strumento giusto
- Valutare sempre: **LLM vs ML classico vs sistemi a regole**

? Domande



Riferimenti

- Vaswani et al., *Attention Is All You Need* (2017)
- Radford et al., *Language Models are Unsupervised Multitask Learners* (GPT-2, 2019)
- Brown et al., *Language Models are Few-Shot Learners* (GPT-3, 2020)
- Turing, *Computing Machinery and Intelligence* (1950)
- Andrej Karpathy, *Let's build GPT from scratch* (YouTube, 2023)

Risorse Online

- arxiv.org/abs/1706.03762 – Paper Transformer
- github.com/karpathy/nanoGPT – Implementazione minimale di GPT
- huggingface.co – Modelli e dataset

Progetto Satoshi Spritz

- Federazione di gruppi locali di Bitcoiner
- Eventi gratuiti e privacy oriented
- BITCOIN ONLY
- Satoshi Spritz Connect online settimanale
- Orientato all'apprendimento della self-sovereign
- Tutte le settimane un evento online → Satoshi Spritz Connect

Links

- satoshispritz.it
- t.me/SatoshiSpritzConnect

Officine Bitcoin

- Comunità Italiana di Bitcoiners, totalmente gratuita
- BITCOIN ONLY
- Focus su educazione e sviluppo di progetti
- Progetti:
  - Sviluppo nodi Bitcoin
  - Uso di Hardware Wallet
  - Filosofia open source
  - Installazione di Debian
  - Mnemoniche & Dadi
  - . . . e molto altro

Links

- officinebitcoin.it

Bitcoinissimo

- Podcast Bitcoin e statistiche
- Episodi in italiano, inglese, ungherese, cinese, russo, spagnolo, francese
- Statistiche rete in tempo reale, dati di mercato, block explorer

Links

- bitcoinissimo.it