COME E PERCHÉ FUNZIONA CHAT GPT

ALESSANDRO TOSCANO[1,2,3]

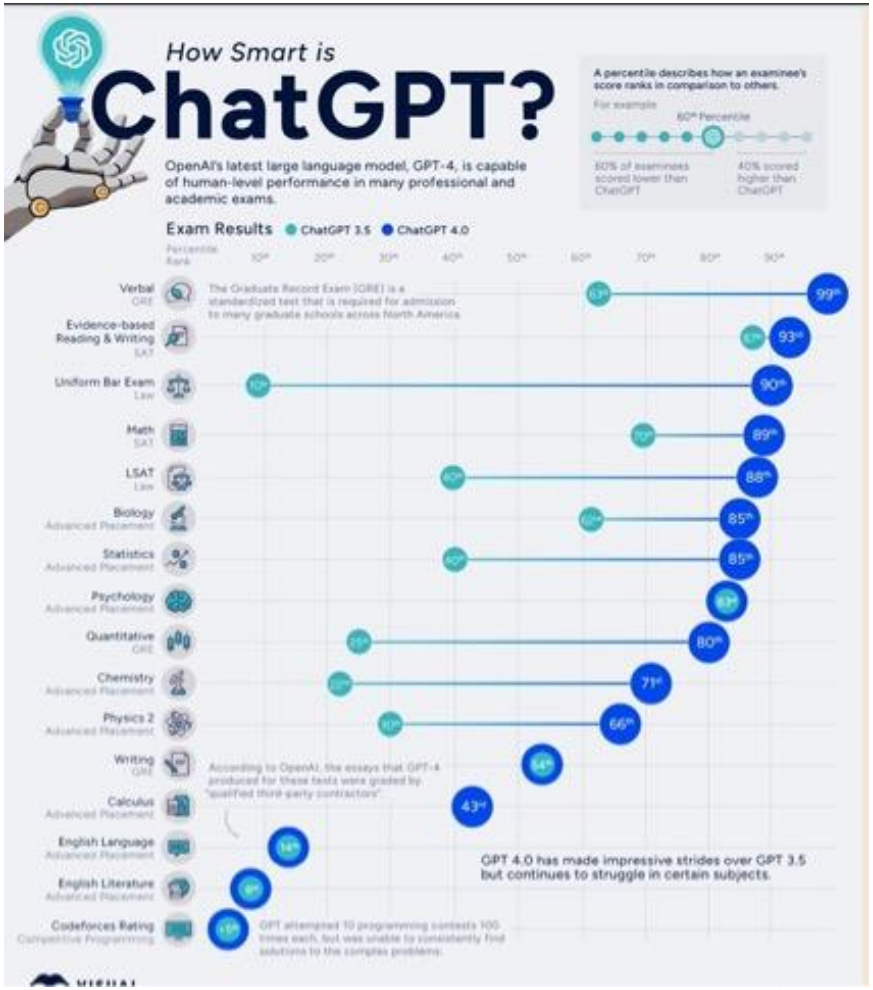
1. **Università degli Studi Roma Tre**
2. **Italian Observatory of Blockchain at Cnel**
3. **LabChain**

09 maggio 2023

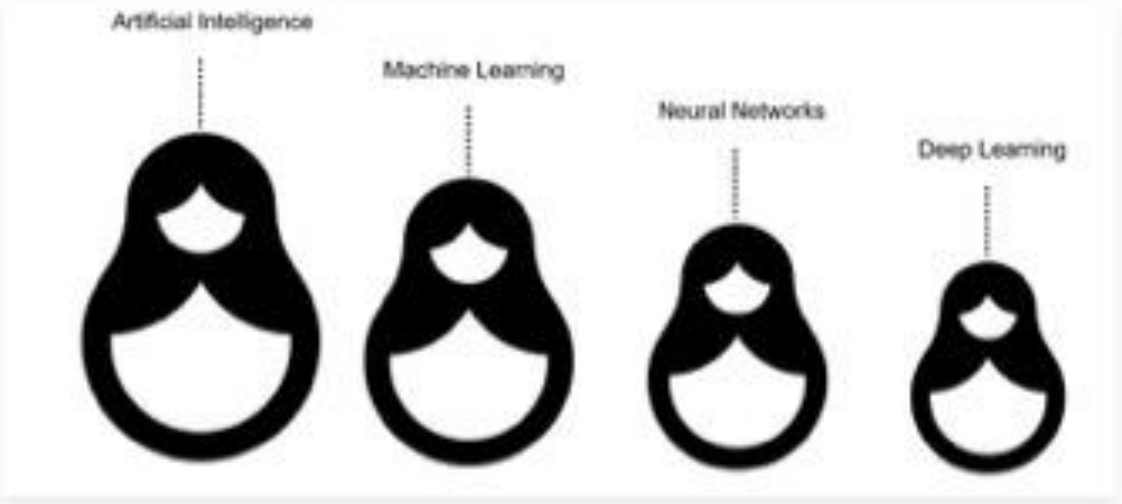
Intelligenza artificiale e lavoro

COS'È CHAT GPT?

COS'È CHAT GPT?



INTELLIGENZA ARTIFICIALE, MACHINE LEARNING, RETI NEURALI E DEEP LEARNING



Il Deep learning è un tipo di rete neurale che a sua volta è una delle tecniche per implementare il machine learning (o apprendimento automatico) che infine è una forma, non l'unica, di intelligenza artificiale.

INTELLIGENZA ARTIFICIALE, MACHINE LEARNING, **RETI NEURALI E DEEP LEARNING**

- Tendenzialmente, si parla erroneamente di deep learning e reti neurali in modo interscambiabile.
- L’aggettivo qualificativo "profondo" (deep) in deep learning si riferisce esclusivamente alla profondità dei livelli, vale a dire al numero di livelli, in una rete neurale.
- Una rete neurale che consiste in più di tre livelli - comprensivi del livello di ingresso e di uscita - può essere considerata un algoritmo di deep learning.
- Una rete neurale che ha solo due o tre livelli è definita come una rete neurale di base.

INTELLIGENZA ARTIFICIALE, **MACHINE LEARNING**, RETI NEURALI E DEEP LEARNING

- Il deep learning e il machine learning **non sono** la stessa cosa.
- Il machine learning riguarda gli algoritmi di apprendimento automatico di una macchina.
- In generale gli algoritmi di machine learning analizzano un insieme di dati per costruire un modello previsionale in grado di classificare gli oggetti in autonomia e rispondere correttamente alle domande su un particolare dominio della conoscenza.
- Possono basarsi su tecniche di apprendimento supervisionato, non supervisionato o con rinforzo.

INTELLIGENZA ARTIFICIALE, **MACHINE LEARNING**, RETI NEURALI E DEEP LEARNING

Le tecniche più utilizzate di apprendimento automatico (machine learning) sono:

- Alberi di decisione;
- Reti neurali;
- Apprendimento Bayesiano;
- Apprendimento con rinforzo;
- Algoritmi genetici;
- Programmazione logica induttiva.

INTELLIGENZA ARTIFICIALE, **MACHINE LEARNING**, RETI NEURALI E **DEEP LEARNING**

- La discriminante tra deep learning e le altre tecniche di machine learning è la modalità di apprendimento di ciascun algoritmo.
- **Il deep learning automatizza** gran parte della parte del processo relativo all'estrazione di caratteristiche, eliminando parte dell'intervento umano e consentendo l'uso di dataset più grandi.
- Può inserire i **dati non strutturati** nel loro formato non elaborato (ad es. testo, immagini) e può determinare automaticamente la gerarchia delle caratteristiche che distingue tra loro le diverse categorie di dati.
- **Il machine learning classico**, o "non profondo", dipende in misura maggiore dall'**intervento umano** per apprendere. Esperti umani determinano la gerarchia delle caratteristiche per comprendere le differenze tra gli input di dati, richiedendo dati più strutturati per l'apprendimento.

INTELLIGENZA ARTIFICIALE, MACHINE LEARNING, RETI NEURALI E DEEP LEARNING

- L’intelligenza artificiale (AI) è il termine più ampio utilizzato per classificare le macchine che imitano l'intelligenza umana. Viene utilizzato per prevedere, automatizzare e ottimizzare le attività proprie degli esseri umani, come il riconoscimento vocale e facciale, i processi decisionali e la traduzione di testi.
- Include il machine learning (ML) e molto altro ancora (sistemi esperti, teoria dei segnali, ricerca operativa, logica, ontologie, ecc.).

INTELLIGENZA ARTIFICIALE CONVERSAZIONALE

- L'intelligenza artificiale conversazionale ha fatto molta strada negli ultimi anni, con numerosi modelli e piattaforme sviluppati per consentire alle macchine di comprendere e rispondere agli input del linguaggio naturale.
- Il Natural Language Generation (NLG) è una sottocategoria del NLP (Natural Language Processing) ed è finalizzato a generare del testo in maniera automatica a partire da dati strutturati.
- Nel corso degli anni sono state individuate diverse tecniche: Catena di Markov, RNN, **Transformers**.

INTELLIGENZA ARTIFICIALE CONVERSAZIONALE: **TRANSFORMERS**

- Il funzionamento delle **reti neurali Transformer** è basato su una struttura chiamata "trasformatore" che permette di elaborare e comprendere i dati in modo più efficiente rispetto ai metodi tradizionali.
- Il trasformatore è composto da due componenti principali:
  - il modulo di attenzione
  - il modulo di codifica

INTELLIGENZA ARTIFICIALE CONVERSAZIONALE: **TRANSFORMERS**

- Il modulo di attenzione permette alla rete neurale di «CONCENTRARSI» solo sui dati rilevanti per risolvere un determinato problema, ignorando gli altri. Questo permette di gestire grandi quantità di dati in modo più efficiente e di ottenere risultati più accurati.
- Il modulo di codifica, invece, si occupa di trasformare i dati in uscita dal modulo di attenzione in un formato che la rete neurale può comprendere e utilizzare per prendere decisioni.

COS'È CHAT GPT?

- Chat GPT, acronimo di Generative Pretrained Transformer, è uno strumento di NLG potente e versatile che utilizza algoritmi avanzati di apprendimento automatico per generare risposte simili a quelle umane all'interno di un discorso.
- Realizzata da OpenAI (organizzazione no profit per la ricerca sull'intelligenza artificiale) con l'obiettivo di ottimizzare la conversazione e facilitare l'utilizzo da parte degli utenti.

CHAT GPT: STORIA

- GPT-2 (2019): Modello con 1.5 miliardi di parametri, addestramento da circa 8 milioni di pagine web.
- GPT-3 (2020): Modello con 175 miliardi di parametri che richiede 800 GB di RAM per l'addestramento.

Dati di addestramento di GPT-3:

Set di dati	Numero di Token	Peso nel mix di addestramento
Common Crawl	410 miliardi	60%
WebText2	19 miliardi	22%
Books1	12 miliardi	8%
Books2	55 miliardi	8%
Wikipedia	3 miliardi	3%

CHAT GPT: STORIA

- GPT-4 (14 Marzo 2023): Attualmente disponibile tramite l'utilizzo di chiamate API e per gli utenti di ChatGPT Plus.
- Come altri "transformers", GPT-4 è stato **pre-addestrato a prevedere il prossimo token** utilizzando sia dati pubblici che "dati concessi in licenza da fornitori di terze parti", ed è stato **poi perfezionato con l'apprendimento di rinforzo dal feedback umano**.
- A differenza delle versioni precedenti, GPT-4 può accettare un prompt di testo e immagini. Genera **output testuali** (linguaggio naturale, codice, ecc.) con **input costituiti da testo e immagini intercalati**.
- **Non sono disponibili** dati ufficiali confermati da OpenAI riguardo al numero di parametri utilizzati dal modello neurale e nemmeno dettagli sull'infrastruttura di calcolo e l'architettura usata per eseguire GPT-4.

CHAT GPT-4: PERFORMANCE

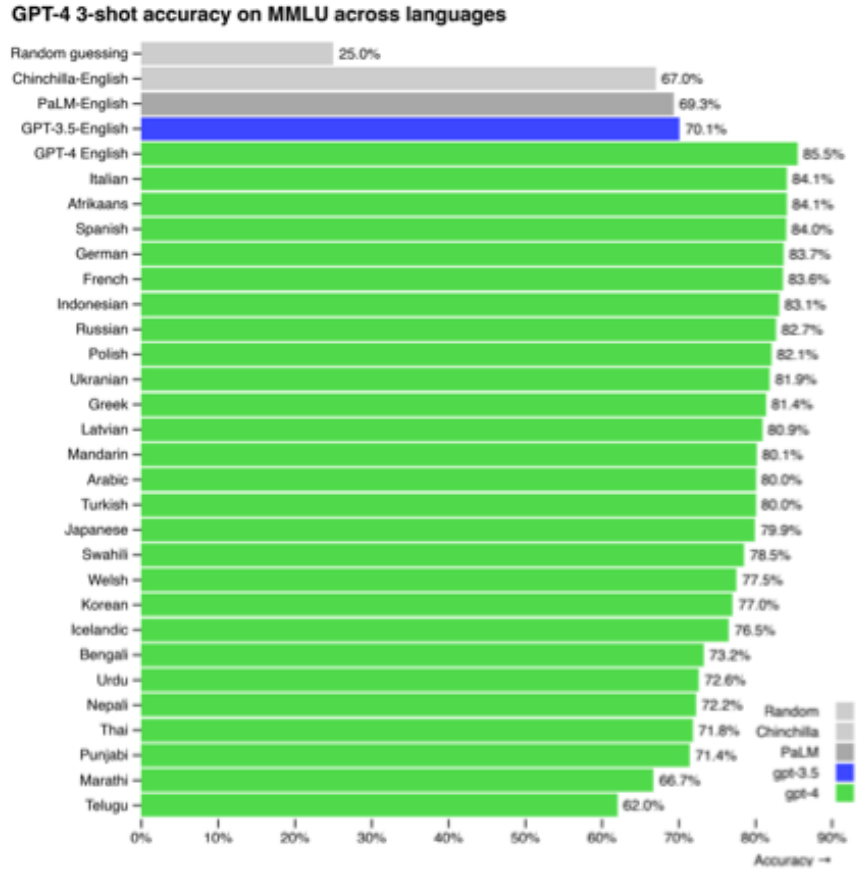


Figure 5. Performance of GPT-4 in a variety of languages compared to prior models in English on MMLU. GPT-4 outperforms the English-language performance of existing language models [2, 3] for the vast majority of languages tested, including low-resource languages such as Latvian, Welsh, and Swahili.

CHAT GPT-4: PERFORMANCE

Reinforcement
Learning with
Human
Feedback

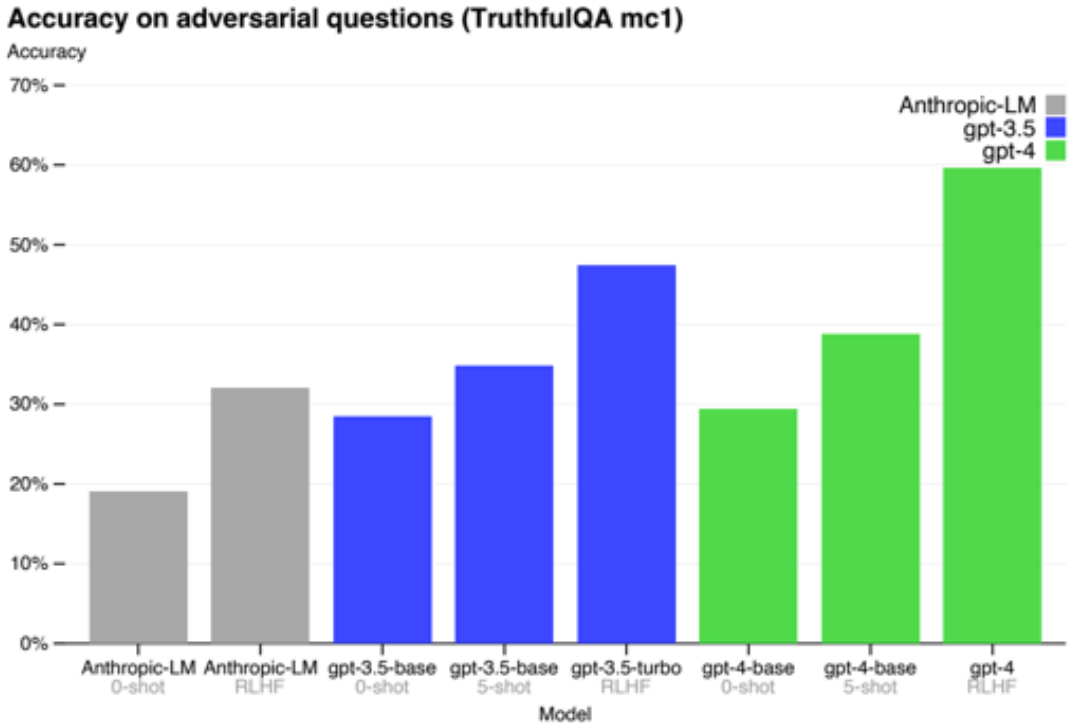


Figure 7. Performance of GPT-4 on TruthfulQA. Accuracy is shown on the y-axis, higher is better. We compare GPT-4 under zero-shot prompting, few-shot prompting, and after RLHF fine-tuning. GPT-4 significantly outperforms both GPT-3.5 and Anthropic-LM from Bai et al. [67].

COME FUNZIONA CHAT GPT: ALGORITMO

1. **Generative**
   a. Tokenizzazione
   b. N-grammi

2. **Pretrained**
   a. Modello
   b. Parametrizzazione

3. **Transformer**
   a. Concentrazione
   b. RLHF

COME FUNZIONA CHAT GPT: TOKENIZZAZIONE

ChatGPT produce una “continuazione ragionevole” del testo che ha prodotto finora, dove per “ragionevole” intendiamo “quello che ci si aspetterebbe che qualcuno scriva dopo aver visto ciò che le persone hanno scritto su miliardi di pagine web, libri scritti, etc.”

```
In[1]:= model = NetModel[{"GPT2 Transformer Trained on WebText Data", "Task" → "LanguageModeling"}]
```

Out[1]= NetChain [Input port: string / Output port: class]

```
In[2]:= model["The best thing about AI is its ability to", {"TopProbabilities", 5}];
        Dataset[ReverseSort[Association[%]], ItemDisplayFunction → (PercentForm[#, 2] &)]
```

Out[3]=

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

COME FUNZIONA CHAT GPT: TOKENIZZAZIONE

La cosa notevole è che ChatGPT fondamentalmente risponde ripetutamente alla stessa domanda: “dato il testo finora, quale dovrebbe essere la prossima parola?” - e aggiunge ogni volta una parola. (Più precisamente sta aggiungendo un “token”.)

```
NestList[StringJoin[#, model[#, "Decision"]] &, "The best thing about AI is its ability to", 7] // Dataset
```

The best thing about AI is its ability to
The best thing about AI is its ability to learn
The best thing about AI is its ability to learn from
The best thing about AI is its ability to learn from experience
The best thing about AI is its ability to learn from experience.
The best thing about AI is its ability to learn from experience. It
The best thing about AI is its ability to learn from experience. It's
The best thing about AI is its ability to learn from experience. It's not

COME FUNZIONA CHAT GPT: TOKENIZZAZIONE

Ok, ad ogni passo ottiene un elenco di parole con le relative probabilità.
Ma quale parola dovrebbe effettivamente scegliere da aggiungere al testo che sta scrivendo?
Si potrebbe pensare che dovrebbe essere la parola alla quale è stata assegnata la "probabilità" più alta.
Ma qui è dove inizia ad entrare un po' di magia.
Perché per qualche motivo - che forse un giorno capiremo scientificamente - se scegliamo sempre la parola con la probabilità più alta, otterremo tipicamente un saggio molto "piatto", che non sembra "mostrare alcuna creatività".

```
StringReplace[Nest[StringJoin[#, model[#, "Decision"]] &, "The best thing about AI is its ability to", 60], "\n" .. → " "]

The best thing about AI is its ability to learn from experience. It's not just a
  matter of learning from experience, it's learning from the world around you. The AI is a very good example
  of this. It's a very good example of how to use AI to improve your life. It's a very good example of
```

COME FUNZIONA CHAT GPT: TOKENIZZAZIONE

Mentre se a volte (a caso) scegliamo parole con punteggio più basso, otteniamo un testo "più interessante".
L'introduzione di questa scelta casuale fa si che se usiamo lo stesso input più volte, otterremo testi diversi ogni volta.
E, in linea con l'aura magica che circonda ChatGPT, è stato introdotto un particolare parametro chiamato "**temperatura**" che determina quanto spesso verranno utilizzate parole con punteggio più basso.
Si è scoperto che una "temperatura" di 0,8 è quella che funziona meglio.
È importante sottolineare che non viene utilizzata alcuna "teoria" per la determinazione della **temperatura ottima:** è il valore che si è visto **funzionare meglio nella pratica**.

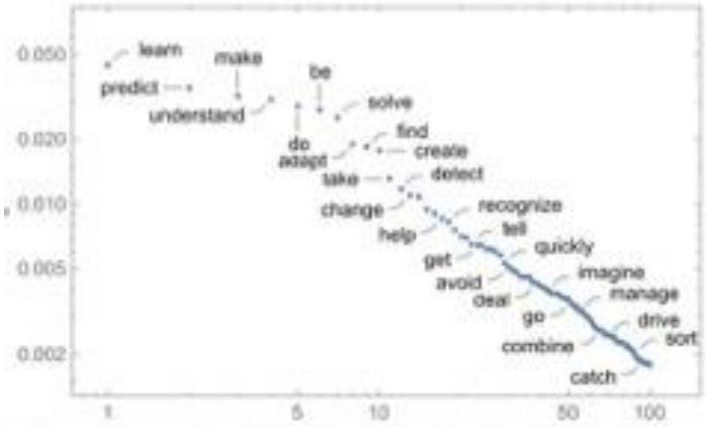
```
StringReplace[Nest[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &, "The best thing about AI is its ability to", 100],
 "\n" .. → " "]
```

```
The best thing about AI is its ability to choose its own goals. I really think it's only a matter of time before the AI-powered gaming
  industry is quite capable of developing concepts that will drive him toward a competitive gaming platform. And that's just something
  we can only hope comes from the development of a completely new game engine. The game engine will be the vehicle of choice. I
  think that this engine is literally the engine of the next generation. There's this whole idea that our next generation of mobile
```

COME FUNZIONA CHAT GPT: TOKENIZZAZIONE

Vale la pena sottolineare che già alla prima iterazione ci sono molte possibili "parole successive" tra cui scegliere (a temperatura 0,8), anche se le loro probabilità diminuiscono abbastanza rapidamente (la linea retta in questo grafico log-log corrisponde a un andamento come n-1, come previsto dalla **teoria statistica generale del linguaggio**, secondo la quale **l'n-esima parola più comune in un campione molto grande di parole si presenta con una frequenza di 1/n**)

```
ListLogLogPlot[Take[ReverseSort[model["The best thing about AI is its ability to", "Probabilities"]], 100], Frame → True]
```



COME FUNZIONA CHAT GPT: N-GRAMMI

Gli N-Grammi sono sequenze di N parole nel corpus della lingua.
Consideriamo ad esempio il seguente corpus di 3 frasi sul quale costruire un modello con tri-grammi:
Oggi vado al mare
Al mare oggi piove
Se piove vado al parco

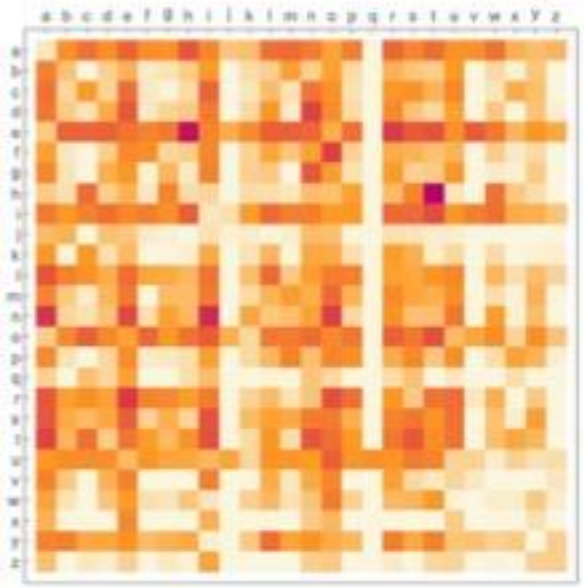
$C(w_i \mid w_{i-1})$	< d >	*oggi*	*vado*	*al*	*mare*	*piove*	*se*	*parco*
< d >	0	1	0	1	0	0	1	0
oggi	0	0	1	0	0	1	0	0
vado	0	0	0	2	0	0	0	0
al	0	0	0	0	2	0	0	1
mare	1	1	0	0	0	0	0	0
piove	1	0	1	0	0	0	0	0
se	0	0	0	0	0	1	0	0
parco	1	0	0	0	0	0	0	0

$P(w_i \mid w_{i-1})$	< d >	*oggi*	*vado*	*al*	*mare*	*piove*	*se*	*parco*
< d >	0	0,33	0	0,33	0	0	0,33	0
oggi	0	0	0,50	0	0	0,50	0	0
vado	0	0	0	1	0	0	0	0
al	0	0	0	0	0,67	0	0	0,33
mare	0,50	0,50	0	0	0	0	0	0
piove	0,50	0	0,50	0	0	0	0	0
se	0	0	0	0	0	1	0	0
parco	1	0	0	0	0	0	0	0

COME FUNZIONA CHAT GPT: N-GRAMMI

Esempio: Di-Gramma composto da due vocali in lingua inglese

```
MatrixPlot[Transpose@With[{data = ...},
  Outer[Lookup[Lookup[data, Key@(#1)], Key@(#2), 0] &, Alphabet[], Alphabet[]]],
 FrameTicks → {{#, None}, {None, #}} &[Table[{i, FromLetterNumber[i]}, {i, 26}]],
 ColorFunction → ChatTechColors["Orange"]]
```



COME FUNZIONA CHAT GPT: MODELLO

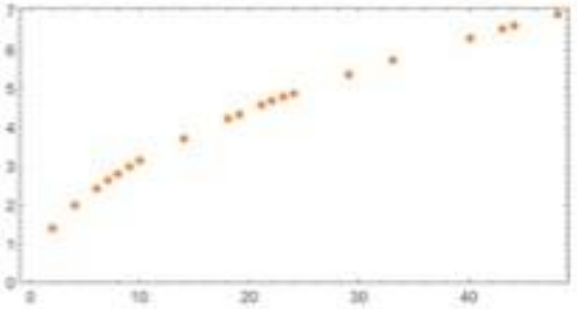
Immaginiamo di volere sapere (come fece Galileo alla fine del 1500) quanto tempo impieghi una palla di cannone lanciata da un qualunque piano della Torre di Pisa per colpire il suolo.

Il modo più semplice è quello di **misurare il tempo** impiegato per ogni piano e **memorizzare i risultati** in una tabella.

Oppure si potrebbe fare ciò che è l'essenza della scienza teorica: **creare un modello** che fornisca una sorta di procedura per calcolare la risposta una volta per tutte piuttosto che misurare e memorizzare ogni evenienza possibile.

Immaginiamo di conoscere il tempo impiegato dalla palla di cannone per ciascun piano della Torre di Pisa:

```
SeedRandom[34535]; ListPlot[{#, Sqrt[#]} & /@ Sort[
   RandomSample[Range[50], 20]], Frame → True, PlotRange → {0, Sqrt[50]},
 PlotStyle → {ChatTechColors["Data"]["Orange"],
   PointSize[.015]}, AspectRatio → 1 / 2]
```



COME FUNZIONA CHAT GPT: PARAMETRIZZAZIONE

Supponiamo di **non conoscere le leggi fisiche** sottostanti e che tutto quello che abbiamo sono i dati misurati.

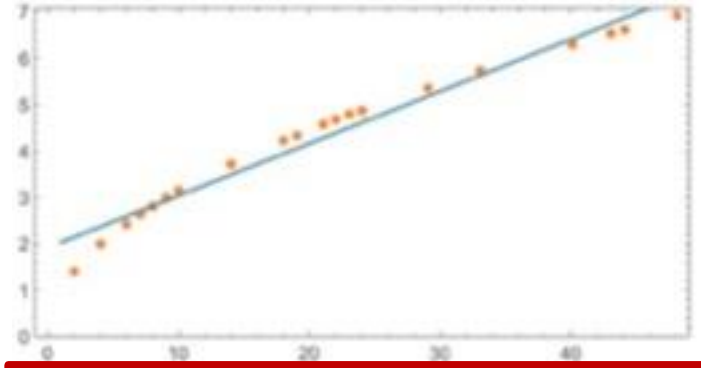
In questo caso **il metodo scientifico procede per ipotesi**.

Ipotizziamo in prima approssimazione di usare l'equazione di una linea retta come modello (y=a+b x).

Per definire il modello bisogna poi **definire i due parametri a e b incogniti**.

Tra tutte le linee rette possibili abbiamo scelto quella che in media è più vicina ai dati disponibili. E con questa linea retta siamo in grado di stimare il tempo di caduta da qualsiasi piano. Ecco che abbiamo creato un modello.

```
With[{data = (SeedRandom[34535];
    {#, Sqrt[#]} & /@ Sort[RandomSample[Range[50], 20]])},
 Show[ListPlot[data, Frame → True, PlotRange → {0, Sqrt[50]},
   PlotStyle → {ChatTechColors["Data"]["Orange"],
     PointSize[.015]}, AspectRatio → 1 / 2],
  Plot[Evaluate[Fit[data, {1, x}, x]], {x, 1, 50},
   PlotRange → {0, Sqrt[50]},
   PlotStyle → ChatTechColors["Data"]["Teal"]]]]
```



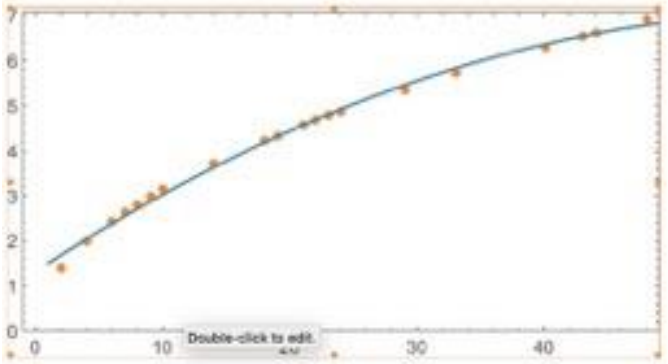
```
Evaluate[Fit[data, {1, x}, x]]

1.93329 + 0.112544 x
```

COME FUNZIONA CHAT GPT: PARAMETRIZZAZIONE

Il modello sembra funzionare bene.
Ma perché abbiamo scelto la retta come modello?
Lo abbiamo fatto perché è qualcosa di matematicamente semplice e perché sappiamo che in molti casi i dati misurati sono descritti da funzioni matematicamente semplici.
Vediamo cosa succede se proviamo ad utilizzare qualcosa di matematicamente più complicato - diciamo una funzione del tipo a+bx+cx^2.
In questo caso il modello è a tre parametri e sembra funzionare meglio.

```
With[{data = (SeedRandom[34535];
    {#, Sqrt[#]} & /@ Sort[RandomSample[Range[50], 20]])},
 Show[ListPlot[data, Frame → True, PlotRange → {0, Sqrt[50]},
   PlotStyle → {ChatTechColors["Data"]["Orange"],
     PointSize[.015]}, AspectRatio → 1 / 2],
  Plot[Evaluate[Fit[data, {1, x, x^2}, x]], {x, 1, 50},
   PlotRange → {0, Sqrt[50]},
   PlotStyle → ChatTechColors["Data"]["Teal"]
  ]]]
```



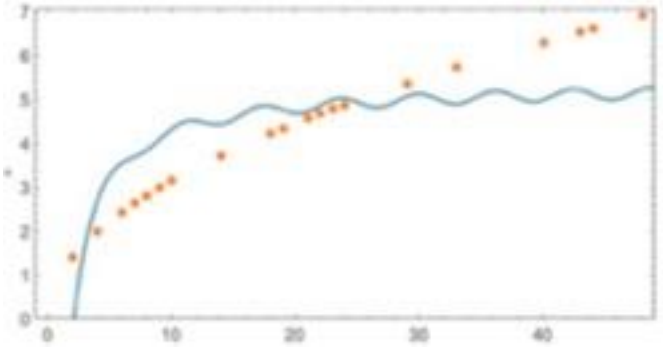

```
Evaluate[Fit[data, {1, x, x^2}, x]]

1.33479 + 0.186767 x - 0.00151404 x^2
```

COME FUNZIONA CHAT GPT: PARAMETRIZZAZIONE

Non è detto però che complicando il modello automaticamente il risultato sia migliore.
Le cose, infatti, possono andare anche molto peggio.
Di seguito il meglio che possiamo fare utilizzando la funzione a+b/x+c sinx:

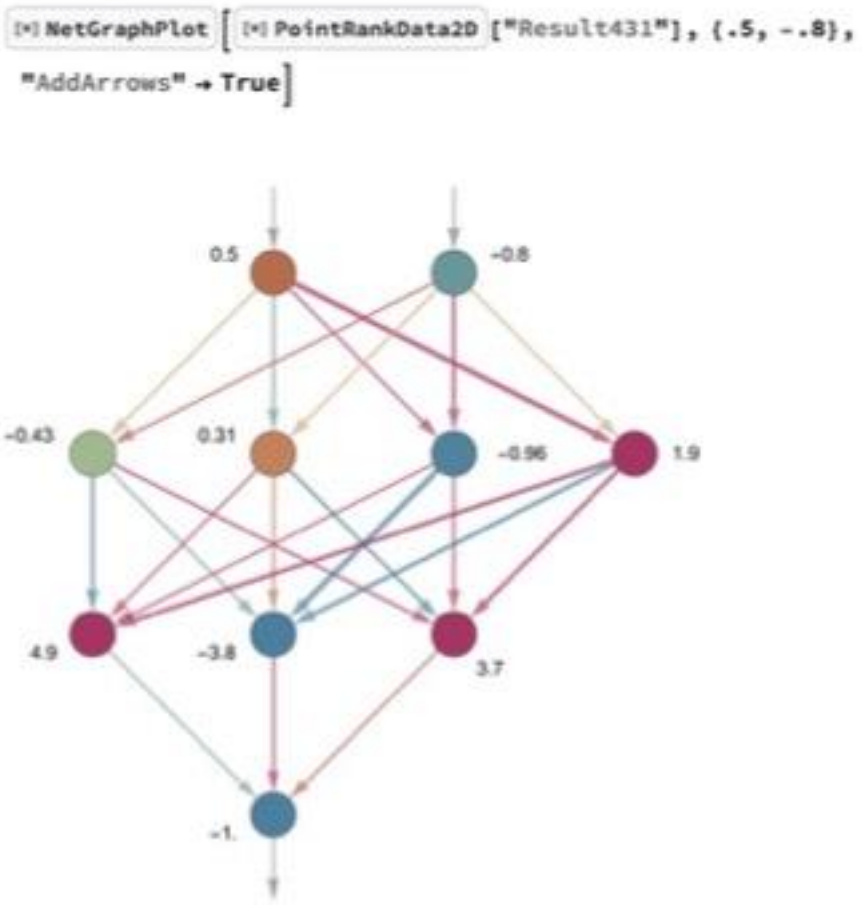
```
With[{data = (SeedRandom[34535];
    (#, Sqrt[#]) & /@ Sort[RandomSample[Range[50], 20]])},
  Show[ListPlot[data, Frame → True, PlotRange → {0, Sqrt[50]},
    PlotStyle → {ChatTechColors["Data"]["Orange"],
      PointSize[.015]}, AspectRatio → 1 / 2],
   Plot[
    Evaluate[(a + b / x + c Sin[x]) /.
      FindFit[data, a + b / x + c Sin[x], {a, b, c}, x]], {x, 1, 50},
    PlotRange → {0, Sqrt[50]},
    PlotStyle → ChatTechColors["Data"]["Teal"]
   ]]]
```



COME FUNZIONA CHAT GPT: TRANSFORMER

Data la struttura di una rete neurale biologica, si è tentato di riprodurre – in modo semplificato – un sistema simile che potesse funzionare altrettanto bene per l'elaborazione di informazioni.

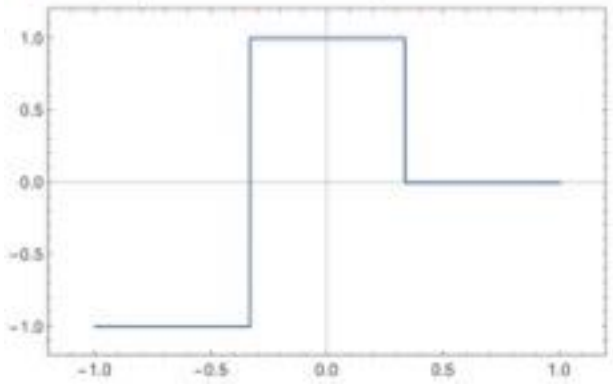
La rete neurale artificiale è composta da **nodi** o neuroni formali che sono delle unità computazionali fondamentali e che, essendo **interconnessi**, formano un **grafo** composto almeno da **uno strato di ingresso** e **uno strato di uscita**. Anche questa rete ha come obiettivo quello di elaborare dati e informazioni e funziona con un input (che possiamo paragonare all'impulso elettrico per il neurone biologico) che arriva al nodo dello strato d'ingresso.



COME FUNZIONA CHAT GPT: TRANSFORMER

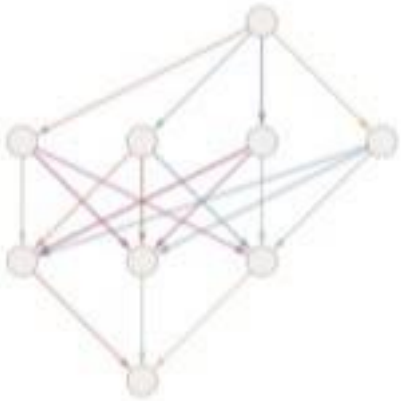
Alleniamo, vale a dire determiniamone i parametri costitutivi, una rete neurale per riprodurre una funzione del tipo seguente.

```
Plot[Piecewise[{{-1, x < -1/3}, {1, -1/3 < x < 1/3}, {0, x > 1/3}}],
 (x, -1, 1), Exclusions → None, Frame → True, PlotRangePadding → .2]
```



Cosa succede se forniamo valori casuali ai pesi?

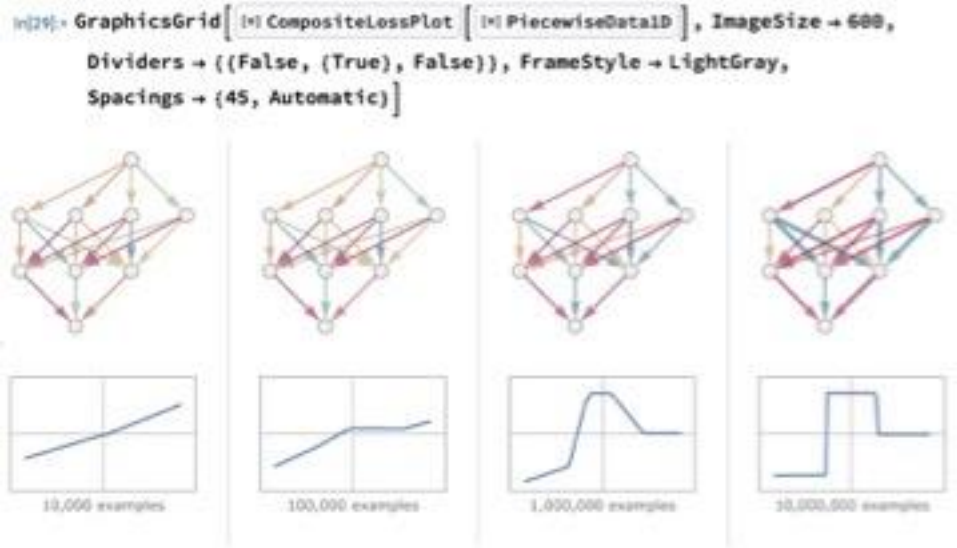
```
GraphicsGrid[
 With[{net = NetInitialize[ PiecewiseNet [{4, 3, 1}], RandomSeeding → #,
       Method → {"Random", "Biases" → NormalDistribution[0, .2]}]},
  {NetGraphPlot [net, ImageSize → {140}],
   Plot[net[x], {x, -1, 1}, Frame → True, PlotRange → {-1, 1},
    FrameTicks → None, ImageSize → {140}]} & /@ (10) // Transpose,
 ImageSize → 600, Alignment → Center, Dividers → {{False, {True}, False}},
 FrameStyle → LightGray, Spacings → {50, -20}]
```



COME FUNZIONA CHAT GPT: TRANSFORMER

Quindi, come possiamo trovare i pesi che riprodurranno la funzione?

L'idea di base è quella di fornire molti esempi di coppie "input → output" da «imparare» per determinare i pesi che riprodurranno al meglio l'uscita in funzione dell'ingresso.
Di seguito il risultato di ciò che si ottiene utilizzando progressivamente un numero maggiore di coppie ingresso/uscita di allenamento.



PERCHÉ FUNZIONA CHAT GPT

Il calcolo della probabilità di una generica frase si scontra con **un primo limite obiettivo**, ovvero che è calcolabile ed è non nulla solo se nel corpus sono presenti tutti gli N-Grammi coinvolti nel calcolo.

Ciò implicherebbe un corpus enorme dove compaiono tutte le possibili frasi da valutare.

Nel web sono presenti alcune centinaia di miliardi di parole; nei libri digitalizzati sono presenti altre centinaia di miliardi di parole.

Ma con 40.000 parole comuni, persino il numero di **possibili bigrammi** è già pari a **1,6 miliardi** - e il numero di **possibili trigrammi** è di **60 trilioni**.

Quindi non c'è modo di stimare tutte le probabilità di un trigramma da tutte le parole disponibili.

E quando arriviamo a frasi di 20 parole, il numero di possibili **20-grammi** è **maggiore del numero di particelle nell'universo**!

PERCHÉ FUNZIONA CHAT GPT

Quindi, cosa possiamo fare?

La grande idea è quella di creare un «Large Language Model» che ci consenta di stimare le probabilità con cui le sequenze dovrebbero occorrere - anche se non abbiamo mai visto esplicitamente quelle sequenze nel corpus di testo che abbiamo esaminato/che è disponibile.

E al centro di ChatGPT c'è proprio un LLM che è stato costruito per fare un buon lavoro nel calcolare queste probabilità.