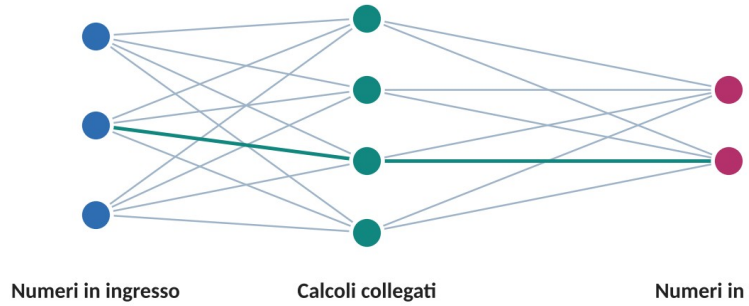
L’idea di base e la rete

Il modello elabora numeri appresi dai testi e genera una continuazione, un token alla volta.

Un **modello linguistico** è un programma che, dopo aver letto moltissimi testi, ha imparato a prevedere quale pezzo di testo (token) può venire dopo quelli che riceve. Quelli molto grandi si chiamano LLM, dall’inglese Large Language Model. Impara regolarità dai testi: non consulta una biblioteca completa a ogni risposta, ma può anche memorizzare e riprodurre brani specifici. Le parole nuove sono spiegate nella pagina seguente, «Gli attrezzi».

Una rete di calcoli (rete neurale)



Ogni cerchio rappresenta un’unità che esegue semplici calcoli. Le frecce indicano come i numeri passano da un’unità all’altra, vengono elaborati e contribuiscono a produrre il risultato finale. Il disegno mostra una versione semplificata di una rete neurale.

Durante l’apprendimento

Il modello confronta i risultati con quelli attesi e, quando sbaglia, modifica i propri calcoli per migliorare. Alla fine i pesi vengono salvati nel file del modello.

Durante la risposta

I pesi restano fissi; cambiano i numeri calcolati sul testo.

Il nostro obiettivo

Capire come un modello, ricevendo l’inizio di frase «Domani vado al», possa arrivare a scegliere come parola successiva «mare», rifacendo a mano tutti i calcoli.

Il percorso che seguiremo

Dalle parole ai numeri
preparare la frase
passi 1–3

→

Numeri e relazioni
elaborare il contesto
passi 4–11

→

La parola successiva
scegliere il token
passo 12

Nei passi che seguono i numeri sono inventati e piccoli, per poter fare i conti a mano. Il corso segue un’altra frase («Il vecchio orologio della piazza si fermò quando») con i numeri misurati su un modello vero: Minerva. Due strade verso le stesse idee. Le tavole animate: www.cultor.net/corso-ia/percorso-animato/.

NEL CORSO
Modulo 10 · passo 1 «Una frase, ancora muta»
Modulo 12 · passo 12 «Imparare prevedendo» (l’addestramento)

Gli attrezzi

Prima di cominciare vediamo gli strumenti che il modello usa per svolgere il compito. Nei passi li ritroveremo uno per uno.

Modello
Il «cervello» dell'IA, per usare un'immagine: un programma che, dopo aver letto moltissimi testi, ha imparato regolarità del linguaggio e prevede il token successivo. Ogni assistente (ChatGPT, Claude, Gemini…) ne usa uno. Nel corso usiamo Minerva, creato all'Università Sapienza di Roma, per italiano e inglese.
→ passo 0

Programma
L'applicazione che usa il modello: prepara il testo da dargli, aggiunge i token scelti e decide quando fermarsi.
→ passi 1 e 12

Token
Un pezzo di testo: una parola, una parte di parola o un segno.
→ passo 1

Vocabolario
L'elenco dei token che il modello conosce, salvato in un file a parte. A ogni token corrisponde un numero intero, l'ID. Ogni modello ha il suo vocabolario.
→ passo 1

Tabella degli embedding (Embedding Matrix)
Per ogni ID, una riga di numeri: il vettore di partenza del token. Token usati in contesti simili hanno righe simili, ma non per forza lo stesso significato.
→ passo 2

Vettore e dimensione
Un vettore è una fila ordinata di numeri. La dimensione è quanti numeri ha il vettore di ogni token: in Minerva-350M sono 1 152.
→ passo 2

Mappa delle somiglianze
I vettori visti come punti: vicini se i token compaiono in contesti simili. Su carta si disegna in due dimensioni, perdendo parte delle relazioni.
→ passo 2

Posizione
Dice al modello dove si trova ogni token: con un vettore sommato all'embedding o, nei modelli recenti, dentro il confronto fra Q e K.
→ passo 3

Pesi
Tutti i numeri imparati durante l'addestramento e salvati nel file del modello: le righe delle tabelle, WQ, WK, WV, WO, W1, W2 e gli altri. Durante una risposta restano fissi.
→ passi 0 e 4

Tabelle WQ, WK, WV
Trasformano il vettore di un token in tre file di numeri: la domanda Q (che cosa cerco), l'etichetta K (come posso essere trovato) e il contenuto V (che cosa trasmetto).
→ passo 4

Prodotto scalare
Si moltiplicano i numeri nello stesso posto e si sommano i risultati: un solo numero che dice quanto due vettori sono compatibili.
→ passo 5

Maschera causale
Impedisce a ogni token di usare quelli che vengono dopo: i loro punteggi diventano $-\infty$ e, dopo la softmax, zero.
→ passo 6

Softmax e numero e
Trasforma i punteggi in quote positive che sommano al 100%, usando il numero e (circa 2.718, una costante come π).
→ passi 7 e 12

Attenzione e teste
L'attenzione fa arrivare a ogni token le informazioni dei token precedenti. Una testa è un gruppo di questi calcoli; in ogni strato ne lavorano diverse insieme, e la tabella WO ne unisce i risultati. Alcune mostrano schemi riconoscibili, molte no.
→ passi 5–9

Rete in avanti (Feed Forward)
Rielabora ogni token da solo, senza guardare gli altri, con le tabelle W1 e W2 e la soglia ReLU.
→ passo 10

Normalizzazione e connessione residua
La prima regola la scala dei numeri. La seconda somma i risultati al vettore di partenza, invece di sostituirlo.
→ passo 11

Strato (layer)
Un blocco fatto di attenzione e rete in avanti. Si ripete più volte, ogni volta con pesi diversi: 16 in Minerva-350M.
→ passo 11

Transformer
Il tipo di architettura, il «motore», costruito con questi strati. Nei modelli che scrivono, ogni token può usare solo sé stesso e quelli che lo precedono. Minerva è un transformer.
→ passi 4–11

Tabella di uscita (logit)
Dà un punteggio a ogni token del vocabolario; la softmax lo trasforma in probabilità.
→ passo 12

Temperatura
Regola quanto è varia la scelta del token: più prudente o più creativa.
→ passo 12

Il calcolo, passo per passo

Una frase, tutti i numeri: ogni operazione si può rifare a mano.

Versione didattica semplificata usando la frase:
"Domani vado al mare"
(4 parole, così i calcoli restano leggibili.)
Nei calcoli usiamo il punto per i decimali, come sulle calcolatrici (0.51 vale 0,51); i numeri grandi sono scritti con uno spazio fra le migliaia (50 257).

---

0. Prima di cominciare: da dove arrivano i numeri

Tutti i numeri di questo esempio sono inventati, scelti piccoli per poter fare i conti a mano. Nei modelli veri (Claude, ChatGPT, Gemini, Minerva ecc.) i numeri sono di due tipi, ed è utile distinguerli fin dall'inizio:

- **Numeri appresi** (detti anche **pesi** o **parametri**): sono fissati una volta per tutte durante l'addestramento e salvati nel **file del modello** (un file a sé, spesso di molti gigabyte, per esempio con estensione .safetensors). Le matrici Embedding, WQ, WK, WV, WO, W1, W2 e i valori b1, b2 stanno tutti lì.
- **Numeri calcolati al momento**: nascono mentre il modello elabora la frase (Q, K, V, punteggi, probabilità…) e vengono buttati via quando ha finito.
- Il **Vocabolario** è un terzo file, separato dal file del modello: contiene solo l'elenco «pezzo di testo → numero ID».

Come si «imparano» i pesi? All'inizio dell'addestramento sono numeri casuali. Il modello legge miliardi di frasi, prova a indovinare ogni parola successiva, misura quanto ha sbagliato e corregge di pochissimo tutti i pesi nella direzione che riduce l'errore. Ripetuto un numero enorme di volte, questo processo produce i pesi definitivi (quelli utilizzati dal modello che si sta usando). Il corso approfondisce questa fase; qui ci serve solo sapere che, quando il modello lavora, i pesi sono già pronti e non cambiano più.

---

1. Tokenizzazione

Serve a
Trasformare le parole in identificatori numerici presenti nel vocabolario del modello.

Da dove vengono i numeri?
Dal **Vocabolario** (Vocabulary).

Token	ID
Domani	123
vado	34
al	5
mare	72

Quindi:

Domani	vado	al	mare
123	34	5	72

Non c'è alcun calcolo.
Si effettua una ricerca nel vocabolario:
"Domani" → 123
Gli ID sono sempre numeri interi da 0 in su, senza segno meno né virgola: indicano solo la posizione della parola nell'elenco, come il numero di una pagina. I numeri negativi e con la virgola compaiono dal passo 2.

> Nella realtà un token non è sempre una parola intera: le parole rare vengono spezzate in più pezzi (per esempio «tokenizzazione» → «token» + «izzazione»). Un vocabolario reale contiene decine di migliaia di pezzi: 50 257 in GPT-2, 51 200 in Minerva-7B.

---

2. Embedding

Serve a
Trasformare ogni ID in un vettore numerico che esprima il significato della parola.

Da dove arrivano i numeri?
Da una gigantesca tabella chiamata **Embedding Matrix**: una riga per ogni token del vocabolario. L'ID dice quale riga leggere (ID 123 → riga 123). La tabella è fatta di numeri appresi e sta nel file del modello.
Nella realtà un vettore può avere 768, 1 536, 4 096 o 12 288 dimensioni (cioè contenere altrettanti numeri).
Noi ne usiamo solo 4.

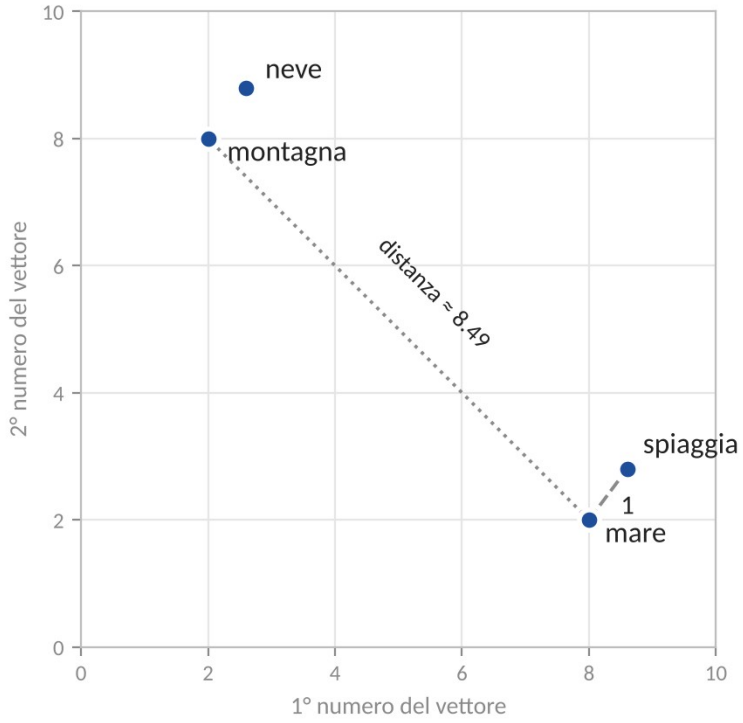
Esempio

ID	Embedding
123	[0.8, 0.1, -0.4, 0.5]
34	[0.3, 0.9, 0.2, -0.2]
5	[0.1, -0.5, 0.6, 0.7]
72	[0.9, 0.8, -0.1, 0.3]

I numeri possono essere positivi o negativi (anche nell'esempio ce ne sono) e in genere sono piccoli, vicini allo zero. Anche qui non c'è calcolo: si legge una riga della tabella. I singoli numeri non hanno un'etichetta leggibile (non esiste «la casella del colore»): è l'insieme del vettore a collocare la parola vicino alle parole di significato simile.

Mappa delle somiglianze
Che cosa vuol dire «vicino»? Ogni vettore si può pensare come un punto su una mappa: i suoi numeri sono le coordinate, come latitudine e longitudine su una carta geografica. Con 2 numeri la mappa è un foglio; con 4 o con migliaia di numeri non si può più disegnare, ma si calcola nello stesso modo.
Esempio con vettori inventati di 2 numeri:

Parola	Vettore
mare	[8, 2]
spiaggia	[8.6, 2.8]
montagna	[2, 8]
neve	[2.6, 8.8]



Le parole che compaiono in contesti simili finiscono vicine: mare e spiaggia da una parte, montagna e neve dall'altra.

Operazione
Distanza tra due punti (teorema di Pitagora): si fa la differenza tra i numeri corrispondenti, si eleva ciascuna al quadrato, si sommano i quadrati e si estrae la radice quadrata.

```
mare–spiaggia: √((8.6–8)² + (2.8–2)²) = √(0.36 + 0.64) = √1 = 1
mare–montagna: √((2–8)² + (8–2)²) = √(36 + 36) = √72 ≈ 8.49
```

Distanza piccola = parole simili; distanza grande = parole lontane.

> **Chi decide le vicinanze?** Nessuno: durante l'addestramento le parole che compaiono in frasi simili («andare al mare», «andare in spiaggia») ricevono vettori simili. Per questo la vicinanza misura il «comparire negli stessi contesti», non sempre l'identità di significato: «caldo» e «freddo» sono contrari, eppure finiscono vicini, perché si usano nelle stesse frasi («oggi fa …»).
> **Nei modelli reali** la somiglianza si misura più spesso con il prodotto scalare (o con una sua variante, la «similarità del coseno»): la stessa operazione che useremo al passo 5. Le mappe a due dimensioni che si vedono nei libri sono semplificazioni: per disegnare migliaia di dimensioni su un foglio bisogna «schiacciarle» in due, perdendo parte delle informazioni.

3. Informatore della posizione (Positional Encoding)

Serve a
Far capire al modello dove si trova ogni parola nella frase.
Infatti «cane morde uomo» non è uguale a «uomo morde cane»: stesse parole, stessi vettori, ma ordine diverso.

Vettori posizione

posizione	vettore
1	[0.1, 0.0, 0.1, 0.0]
2	[0.2, 0.0, 0.2, 0.0]
3	[0.3, 0.0, 0.3, 0.0]
4	[0.4, 0.0, 0.4, 0.0]

(Numeri inventati, scelti per far vedere la regolarità. Nei modelli reali i vettori di posizione si ottengono con una formula fissa, oppure sono appresi, oppure — nei modelli più recenti — la posizione viene inserita più avanti, «ruotando» i vettori Q e K.)

Operazione
Somma elemento per elemento: il 1° numero col 1° numero, il 2° col 2°, e così via.
Per "Domani":

```
Embedding: [0.8, 0.1, -0.4, 0.5]
Posizione: [0.1, 0.0,  0.1, 0.0]
Somma:     [0.9, 0.1, -0.3, 0.5]
```

Esercizio — risultati per tutte le parole (controllabili a mano):

Parola	Embedding + posizione	Risultato
Domani (pos. 1)	[0.8, 0.1, -0.4, 0.5] + [0.1, 0, 0.1, 0]	[0.9, 0.1, -0.3, 0.5]
vado (pos. 2)	[0.3, 0.9, 0.2, -0.2] + [0.2, 0, 0.2, 0]	[0.5, 0.9, 0.4, -0.2]
al (pos. 3)	[0.1, -0.5, 0.6, 0.7] + [0.3, 0, 0.3, 0]	[0.4, -0.5, 0.9, 0.7]
mare (pos. 4)	[0.9, 0.8, -0.1, 0.3] + [0.4, 0, 0.4, 0]	[1.3, 0.8, 0.3, 0.3]

---

4. Calcolo di Q, K e V

Serve a
Preparare i dati per l'attenzione.
Ogni parola genera tre vettori, con tre ruoli diversi:

- Query (Q) — «che cosa sto cercando?»
- Key (K) — «che cosa offro, come mi presento?»
- Value (V) — «che informazione porto, se vengo scelta?»

Da dove arrivano?
Da tre matrici **apprese**, cioè fissate durante l'addestramento: WQ, WK, WV.

> **Dove si trovano?** Non nel Vocabolario: nel file del modello, insieme alla Embedding Matrix e a tutti gli altri pesi (vedi passo 0). Ogni strato (passo 11) e ogni testa (passo 9) ha le sue WQ, WK, WV.

Attenzione: da qui in avanti usiamo vettori di sole 2 dimensioni, perché i conti con 4 numeri diventerebbero lunghi. I numeri sono quindi nuovi e indipendenti da quelli dei passi precedenti.
Supponiamo che il vettore della parola «al» (già sommato alla sua posizione) sia:

```
X(al) = [1, 2]
```

e

```
WQ = [ 0.5  0.2 ]
     [ 0.3  0.4 ]
```

Operazione
Moltiplicazione vettore × matrice (riga per colonna).
`Q = X · WQ`
Come si fa: ogni numero del risultato si ottiene prendendo il vettore X e una colonna della matrice, moltiplicando i numeri a coppie e sommando.
`1° numero (colonna 1): 1×0.5 + 2×0.3 = 1.1`
`2° numero (colonna 2): 1×0.2 + 2×0.4 = 1.0`
Risultato:
`Q(al) = [1.1, 1.0]`
Lo stesso viene fatto per le altre due matrici:
`K = X · WK`
`V = X · WV`
e per ogni parola della frase. Alla fine ogni parola ha il suo Q, il suo K e il suo V.

---

5. Attention Score

Serve a
Misurare quanto una parola è correlata alle altre.
Prendiamo la parola «al» e confrontiamo la sua Query con la Key di ogni parola della frase (compresa sé stessa). Scegliamo «al» e non «Domani» perché «Domani» è la prima parola: per la regola del passo 6 potrebbe guardare solo sé stessa e l'esempio non mostrerebbe nulla.
Per esempio:
`Q(al) = [1.1, 1.0]`
e
`K(Domani) = [0.7, 0.5]`

Operazione
Prodotto scalare: si moltiplicano i numeri a coppie (1° con 1°, 2° con 2°) e si sommano i risultati.
`1.1×0.7 + 1.0×0.5 = 0.77 + 0.5 = 1.27`
Questo è il punteggio di attenzione di «al» verso «Domani». Lo stesso calcolo con le altre Key:

Key (inventate)	Calcolo	Punteggio
K(Domani) = [0.7, 0.5]	1.1×0.7 + 1.0×0.5	1.27
K(vado) = [0.5, 0.25]	1.1×0.5 + 1.0×0.25	0.80
K(al) = [0.2, 0.0]	1.1×0.2 + 1.0×0.0	0.22
K(mare) = [0.9, 0.6]	1.1×0.9 + 1.0×0.6	1.59

Più il punteggio è alto, più le due parole «si interessano» l'una all'altra (è una misura di somiglianza, come la distanza nella mappa del passo 2).

> Nei modelli reali il punteggio viene anche diviso per la radice quadrata della dimensione dei vettori (qui √2 ≈ 1.41), per evitare numeri troppo grandi. Per tenere i conti semplici lo omettiamo.

---

6. Causal Mask (maschera causale)

Serve a
Impedire a una parola di guardare il futuro, cioè le parole che vengono dopo di lei.
Perché serve? Durante l'addestramento il modello riceve la frase intera e deve imparare a indovinare ogni parola a partire solo da quelle precedenti. Se «al» potesse vedere «mare»,

copierebbe la risposta invece di imparare a prevederla. Per questo il punteggio di «mare» (1.59, il più alto!) va cancellato.
(Quando poi il modello genera testo, «Domani vado al …», la parola «mare» non esiste ancora: la maschera garantisce che il modello lavori sempre nelle stesse condizioni in cui è stato addestrato.)

Matrice

	D	V	A	M
D	0	-∞	-∞	-∞
V	0	0	-∞	-∞
A	0	0	0	-∞
M	0	0	0	0

(In grigio le caselle vietate: il «futuro».)
Come si legge: ogni riga è la parola che guarda, ogni colonna la parola guardata. 0 = permesso, -∞ («meno infinito») = vietato. Riga A («al»): può guardare D, V e sé stessa; M («mare») è vietata. Si vede che sopra la diagonale ci sono solo -∞: è il «futuro».

Operazione
La riga della maschera si **somma** ai punteggi della parola:

```
punteggi di «al»:  [1.27, 0.80, 0.22, 1.59]
maschera riga A:   [0,    0,    0,    -∞  ]
somma:             [1.27, 0.80, 0.22, -∞  ]
```

Sommare 0 lascia il numero invariato; sommare -∞ lo rende infinitamente negativo.
Durante la Softmax (passo 7) si calcola e elevato a ciascun punteggio (e è un numero fisso, circa 2.718, come π: vedi il riquadro «Che numero è e?» al passo 7), e:

```
e^-∞ = 0
```

quindi i valori futuri spariscono.

> **Perché e^-∞ = 0?** Un esponente negativo significa «uno diviso»: e^-1 = 1/e ≈ 0.37; e^-2 = 1/e² ≈ 0.135; e^-10 ≈ 0.000045. Più l'esponente è negativo, più il risultato si avvicina a zero; con -∞ diventa zero. (In pratica i computer non usano il vero infinito ma un numero negativo enorme, per esempio -1 000 000 000: il risultato è comunque zero.) Che cos'è e e è spiegato al passo 7.

---

7. Softmax

Serve a
Trasformare i punteggi in probabilità: numeri tra 0 e 1 che sommati fanno 1 (cioè percentuali che sommate fanno 100%).
Partiamo dai punteggi di «al» dopo la maschera:

```
[1.27, 0.80, 0.22, -∞]
```

Operazione
1) Esponenziale (e elevato a ciascun punteggio):

```
e^1.27 = 3.56
e^0.80 = 2.23
e^0.22 = 1.25
e^-∞   = 0
```

> **Che numero è e?** Sì, è una costante fissa come π (pi greco): e = 2.71828… (numero di Nepero, o di Eulero), con infinite cifre decimali. Così come π salta fuori ogni volta che c'è di mezzo un cerchio, e salta fuori ogni volta che c'è di mezzo una crescita proporzionale (interessi composti, popolazioni, decadimento radioattivo).

> **Come si calcola?** Con una calcolatrice scientifica: tasto e^x (o «exp»). Per esempio e^1.27: si digita 1.27 e si preme e^x → 3.56. In alternativa: 2.71828 elevato a 1.27.
> **Perché si usa?** Per tre ragioni: (1) il risultato è sempre positivo, anche se il punteggio è negativo, e una probabilità non può essere negativa; (2) cresce in fretta, quindi accentua le differenze: il punteggio più alto «stacca» gli altri; (3) ha proprietà matematiche comode durante l'addestramento. Con un'altra base (per esempio 2 o 10) l'idea funzionerebbe lo stesso, ma e rende i calcoli dell'addestramento più semplici.

2) Somma:

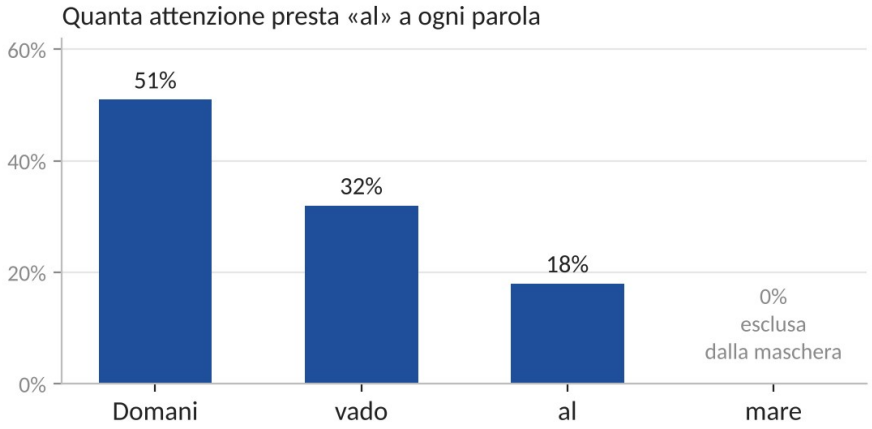
```
3.56 + 2.23 + 1.25 + 0 = 7.04
```

3) Divisione di ciascun valore per la somma:

```
3.56 / 7.04 = 0.51
2.23 / 7.04 = 0.32
1.25 / 7.04 = 0.18
0    / 7.04 = 0
```

Risultato:

```
[0.51, 0.32, 0.18, 0]
```



Leggendo: «al» presta il 51% dell'attenzione a «Domani», il 32% a «vado», il 18% a sé stessa e lo 0% a «mare». (La somma fa 1.01 invece di 1 solo per via degli arrotondamenti.)

---

8. Somma pesata dei Value

Serve a

Creare una nuova rappresentazione della parola «al», arricchita dalle informazioni delle parole che ha «guardato».

Supponiamo:

```
V1 = V(Domani) = [2, 1]
V2 = V(vado)   = [3, 4]
V3 = V(al)     = [1, 2]
```

> **Da dove arrivano V1, V2, V3?** Dal passo 4: sono i vettori Value di ciascuna parola, calcolati con V = X · WV. Qui li inventiamo per brevità. Il Value di «mare» c'è, ma il suo peso è 0 e quindi non contribuisce.

e pesi (dal passo 7):

```
0.51   0.32   0.18
```

Operazione

```
0.51·V1 + 0.32·V2 + 0.18·V3
```

(Ogni vettore viene moltiplicato per il suo peso, poi si sommano componente per componente.)

Prima componente:
`0.51×2 + 0.32×3 + 0.18×1 = 1.02 + 0.96 + 0.18 = 2.16`
Seconda componente:
`0.51×1 + 0.32×4 + 0.18×2 = 0.51 + 1.28 + 0.36 = 2.15`
Nuovo vettore di «al»:
`[2.16, 2.15]`

---

9. Multi-Head Attention

Serve a
Osservare la frase da punti di vista diversi.
Ad esempio:

- testa 1 → grammatica
- testa 2 → soggetto-verbo
- testa 3 → significato

(Esempi indicativi: nessuno assegna un compito alle teste. Le specializzazioni emergono da sole durante l'addestramento e spesso non sono così nette né facili da etichettare.)

> **Da dove arrivano le teste?** Una testa non è un dato salvato da qualche parte: è una ripetizione dei passi 4–8 fatta con un altro gruppo di tre matrici WQ, WK, WV. Ogni testa ha il suo gruppo, appreso durante l'addestramento e salvato nel file del modello (non nel Vocabolario). Tutte le teste lavorano in parallelo sulla stessa frase.

Se abbiamo 2 teste:

```
Head1 → [2.16, 2.15]   (quella calcolata ai passi 4–8)
Head2 → [1, 3]         (inventata)
```

Concatenazione (si mettono i vettori uno dopo l'altro):
`[2.16, 2.15, 1, 3]`
Così si torna a 4 numeri, la dimensione di partenza: nei modelli reali le teste si dividono la dimensione totale (GPT-2 Small: 12 teste × 64 = 768; GPT-3: 96 teste × 128 = 12 288). Il vettore concatenato viene poi moltiplicato per un'ulteriore matrice appresa, WO, che mescola i risultati delle teste.

---

10. Feed Forward Network

Serve a
Rielaborare l'informazione prodotta dall'attenzione, una parola alla volta.
Si applicano altre matrici (W1, W2) e due vettori di correzione (b1, b2, detti «bias»).

> **Da dove arrivano?** Come WQ, WK, WV: sono pesi appresi durante l'addestramento e salvati nel file del modello. Ogni strato ha i suoi.

`H = X·W1 + b1`
poi
`ReLU(H)`
poi
`Y = ReLU(H)·W2 + b2`
ReLU è la regola più semplice possibile: se un numero è negativo diventa 0, altrimenti resta com'è. Serve a introdurre una «soglia», senza la quale tutte le moltiplicazioni in fila equivarrebbero a una sola.

Esempio

```
X = [2, 1]   (numeri inventati)
```

```
W1 = [0.5  0.2]
     [0.1  0.4]
```

1) X·W1 (riga per colonna, come al passo 4):

```
2×0.5 + 1×0.1 = 1.1     2×0.2 + 1×0.4 = 0.8
```

Risultato:

```
[1.1, 0.8]
```

2) Si somma b1 = [0.1, -1.0]:

```
H = [1.1+0.1, 0.8-1.0] = [1.2, -0.2]
```

3) ReLU: il negativo diventa 0

```
ReLU(H) = [1.2, 0]
```

4) Si moltiplica per W2 = [0.3 0.6] / [0.9 0.2] e si somma b2 = [0.1, 0.1]:

```
1.2×0.3 + 0×0.9 = 0.36     1.2×0.6 + 0×0.2 = 0.72
Y = [0.36+0.1, 0.72+0.1] = [0.46, 0.82]
```

Nei modelli reali il vettore intermedio H è circa 4 volte più lungo dell'ingresso (GPT-2 Small: da 768 a 3 072 numeri e ritorno), e al posto di ReLU si usano varianti più morbide (GELU, SwiGLU).

---

11. Strati (Layers)

Serve a

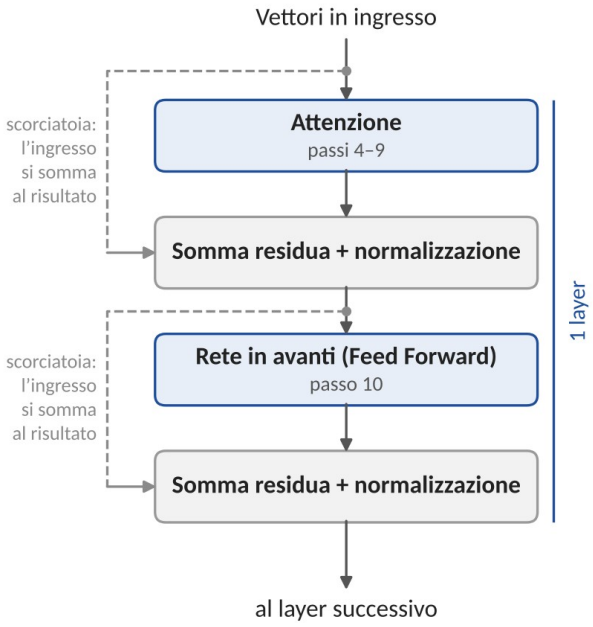
Ripetere tutto il procedimento più volte, raffinando la rappresentazione di ogni parola.

Un layer esegue:

- Attention, cioè i passi 4–9: calcolo di Q, K, V, punteggi, maschera, softmax, somma pesata, più teste, matrice WO.
- Feed Forward, cioè il passo 10.

Dopo ciascuno dei due blocchi ci sono due operazioni di servizio:

- **somma residua**: il risultato del blocco viene sommato al vettore che vi era entrato (somma elemento per elemento, come al passo 3), così l'informazione originale non si perde;
- **normalizzazione**: i numeri vengono riportati su una scala standard, per evitare che crescano o si riducano troppo da uno strato all'altro.



L'uscita del layer 1 diventa l'ingresso del layer 2, e così via. Ogni layer ha le sue matrici (WQ, WK, WV, WO, W1, W2…): 24 layer significano 24 serie di pesi diverse.
Numero tipico:

Modello	Layer
GPT-2 Small	12
GPT-3	96
Minerva-7B (Sapienza)	32
GPT-4	non pubblico

Se ne avessimo 24:
`Token → Layer 1 → Layer 2 → … → Layer 24`

---

12. Punteggio finale (Logits)

Serve a
Dare un punteggio a ogni possibile parola successiva, da trasformare poi in probabilità.
Da dove arrivano? Il vettore dell'ultima parola («al»), uscito dall'ultimo layer, viene moltiplicato per un'ultima matrice appresa (spesso la stessa Embedding Matrix «letta al contrario»). Il risultato è un punteggio per ogni token del vocabolario: 50 257 punteggi in GPT-2. Qui ne mostriamo solo 3.
Output finale:

Parola	Logit
sole	3.2
mare	7.1
montagna	1.5

Softmax finale
Stesso procedimento del passo 7 (e = 2.71828…, la costante spiegata lì):

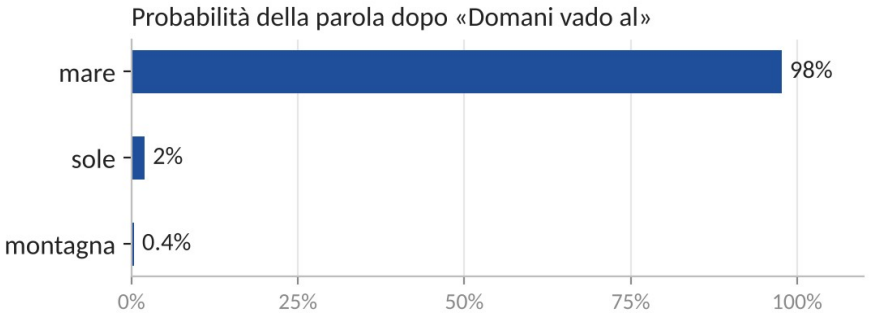
```
e^3.2 = 24.53
e^7.1 = 1212
e^1.5 = 4.48
```

Somma:

```
24.53 + 1212 + 4.48 = 1241 (circa)
```

Probabilità:

```
sole      24.53 / 1241 = 2%
mare      1212 / 1241 = 98%
montagna  4.48 / 1241 = 0.4%
```



(Una probabilità calcolata con e non è mai esattamente zero: «montagna» è molto improbabile, non impossibile.)
Il modello sceglie: **mare**
Di solito la scelta non è «sempre la più probabile»: il modello estrae a sorte rispettando le probabilità (98 volte su 100 «mare»). Un'impostazione chiamata «temperatura» rende la scelta più prudente o più creativa. La parola scelta viene aggiunta alla frase e tutto il procedimento ricomincia per la parola successiva.

Quante operazioni per questa frase?

Con 4 token, vettori di dimensione 4, una sola testa, un solo layer e uno strato Feed Forward intermedio di 16 numeri (4 volte la dimensione), otteniamo circa:

Fase	Operazioni
Positional Encoding	16 somme
Q, K, V	192 moltiplicazioni + 144 somme
Attention Score	10 prodotti scalari (40 moltiplicazioni + 30 somme): con la maschera servono solo 1+2+3+4 confronti
Softmax	circa 26 operazioni
Somma pesata	circa 64 operazioni
Feed Forward	circa 1 000 operazioni (512 moltiplicazioni + altrettante somme)

Totale:
circa 1 600 operazioni elementari per un solo layer (senza contare WO e la normalizzazione). Con 24 layer si superano le 38 000. Il solo passo finale (logits), con un vocabolario di 50 000 token, richiede da solo 200 000 moltiplicazioni.

Regola pratica per i modelli reali: per ogni token elaborato servono circa 2 operazioni per ogni peso del modello. Un modello da 7 miliardi di pesi fa quindi circa 14 miliardi di operazioni per ogni singola parola.

Quanti gruppi di numeri entrano in gioco?

Numeri appresi — salvati nei file del modello

Gruppo	Che cosa fa	Dove sta
Vocabolario	parola → ID	file del vocabolario (tokenizer)
Embedding Matrix	ID → vettore semantico	file del modello
Positional Encoding	posizione → vettore posizione	formula fissa o file del modello
WQ, WK, WV	producono Query, Key, Value	file del modello (un gruppo di tre per testa e per layer)
WO	mescola i risultati delle teste	file del modello
Feed Forward Weights	W1, b1, W2, b2	file del modello (per ogni layer)

Numeri calcolati al momento — spariscono a fine elaborazione

Gruppo	Che cosa sono
Q, K, V	vettori di ogni parola (passo 4)
Attention Scores	confronti Q·K
Attention Weights	softmax dei punteggi
Somma pesata dei Value	informazione trasportata
Logits finali	punteggi per tutto il vocabolario

Le teste di attenzione non sono un gruppo di numeri a sé: sono gruppi distinti di tre matrici WQ, WK, WV (non copie), già compresi sopra.
In un modello reale moderno i numeri appresi sono da alcuni miliardi a centinaia di miliardi, mentre per la nostra frase l'elaborazione passa attraverso poche decine di vettori e matrici.
Per una presentazione didattica usiamo vettori a 4 dimensioni fino al passo 3 e a 2 dimensioni dal passo 4 in poi, così il pubblico può seguire ogni operazione a mano.

Il problema e l'attrezzo

Ogni attrezzo esiste perché risolve un problema. Conoscere il problema aiuta a capire la soluzione.

Attrezzo	Perché serve	Passo
Token e vocabolario	Il modello lavora con un elenco finito di pezzi, non con un testo qualsiasi.	1
ID	Numero identificativo di ogni token: permette al modello di trovare la sua riga nella tabella degli embedding.	1
Embedding	Trasforma ogni token in una serie di numeri che il modello può utilizzare per elaborare il testo.	2
Mappa delle somiglianze	Serve un modo per vedere e misurare quando due vettori si somigliano.	2
Posizione	Le stesse parole in un ordine diverso dicono cose diverse.	3
Q, K e V	Da ogni vettore il modello ricava tre serie di numeri: una per cercare informazioni, una per essere individuato e una per contribuire alla risposta.	4
Punteggio di attenzione	Bisogna stabilire quali token precedenti sono utili a quello che si sta elaborando.	5
Maschera causale	Chi scrive un token alla volta non può usare ciò che viene dopo.	6
Softmax	I punteggi sono numeri qualsiasi: servono quote positive che sommano al 100%.	7, 12
Somma pesata	Ogni token deve ricevere le informazioni dei precedenti, in proporzione alla loro utilità.	8
Più teste	Una sola testa coglie un solo tipo di relazione alla volta.	9
Rete in avanti	Dopo aver raccolto le informazioni degli altri, ogni token deve rielaborarle per conto suo.	10
Connessione residua e normalizzazione	L'informazione di partenza non deve perdersi e i numeri non devono crescere o ridursi troppo.	11
Strati	Un solo passaggio non basta a combinare tutte le informazioni.	11
Tabella di uscita	Dal vettore finale bisogna arrivare a un voto per ogni token possibile.	12
Temperatura	Controllare quanto è varia la scelta del token.	12
Ripetizione	Una risposta è fatta di molti token, da produrre uno alla volta.	12

Gli stessi attrezzi, in modelli diversi

Ogni modello ha i suoi numeri. Per i modelli aperti sono pubblicati; per gli assistenti commerciali più diffusi no.

Modello	Chi lo ha fatto	Voci del vocabolario	Dimensione dei vettori	Strati	Teste per strato	Tabella degli embedding	WQ per strato
Minerva-350M	Sapienza NLP, 2024	32 768	1 152	16	16	32 768 × 1 152	1 152 × 1 152
Minerva-7B	Sapienza NLP, 2024	51 200	4 096	32	32	51 200 × 4 096	4 096 × 4 096
Mistral 7B	Mistral AI, 2023	32 000	4 096	32	32	32 000 × 4 096	4 096 × 4 096
Llama 3 8B	Meta, 2024	128 256	4 096	32	32	128 256 × 4 096	4 096 × 4 096
ChatGPT, Claude, Gemini	OpenAI, Anthropic, Google	non pubblicato	non pubblicato	non pubbl.	non pubbl.	non pubblicata	non pubblicata

Le tabelle WK e WV sono più piccole di WQ quando etichette e contenuti sono condivisi fra gruppi di teste: 1 152 × 288 in Minerva-350M, 4 096 × 1 024 negli altri tre modelli aperti. Fonti: file di configurazione pubblicati con i modelli Minerva; Jiang et al. (2023), Mistral 7B, arxiv.org/abs/2310.06825; Grattafiori et al. (2024), The Llama 3 Herd of Models, arxiv.org/abs/2407.21783.