GruppoMeta

Una breve storia dei Large Language Models e del Language Modeling

Michele Papucci

talia

Un Language Model è un ***modello*** del Linguaggio Naturale.

Assegna ad ogni possibile *sequenza di parole* nella lingua la *probabilità essere osservata*.

Esempio di una sequenza:
“Luca mangia la mela”

Notazione

$$\mathbf{X} = \{ \underset{Luca}{x_1}, \underset{mangia}{x_2}, \underset{la}{x_3}, \underset{mela}{x_4} \}$$

$$P(\mathbf{X}) = P(x_1, x_2, x_3, x_4) = P(Luca, mangia, la, mela)$$

Non tutte le sequenze sono *equiprobabili*

Un parlante italiano ha più probabilità di produrre la sequenza *“Luca mangia la mela”* piuttosto che *“mangia Luca mela la”*

Il Language Model deve quindi catturare la **sintassi** della lingua che vuole modellare.

“*Luca mangia la mela*” è una sequenza più probabile rispetto a *“la mela mangia Luca”*, nonostante anche questa sia *sintatticamente accettabile*.

Il Language Model deve quindi catturare la **semantica** della lingua che vuole modellare.

Questo è un problema difficile da modellare

Questo è un problema difficile da modellare

- La probabilità di osservare una certa parola **non è indipendente** dal contesto;

Questo è un problema difficile da modellare

- La probabilità di osservare una certa parola **non è indipendente** dal contesto;

- Il numero di parole che compongono una lingua (*Vocabolario*) non è fisso;

Questo è un problema difficile da modellare

- La probabilità di osservare una certa parola **non è indipendente** dal contesto;
- Il numero di parole che compongono una lingua (*Vocabolario*) non è fisso;
- La distribuzione delle parole (il loro uso) varia nel tempo e nello spazio.

Le soluzioni proposte inizialmente si basano sullo stimare le probabilità necessarie facendo alcune semplificazioni:

Le soluzioni proposte inizialmente si basano sullo stimare le probabilità necessarie facendo alcune semplificazioni:

- Si utilizza un campione della lingua (solitamente scritta) facendo uso di **corpora**;

Le soluzioni proposte inizialmente si basano sullo stimare le probabilità necessarie facendo alcune semplificazioni:

- Si utilizza un campione della lingua (solitamente scritta) facendo uso di **corpora**;

- Si calcola la probabilità di una sequenza (*probabilità congiunta delle sue parole*) utilizzando la **chain rule** che la trasforma in un prodotto di **probabilità condizionate**.

$$P(Luca, mangia, la, mela) =$$
$$= P(Luca)P(mangia|Luca)P(la|mangia, Luca)P(mela|la, mangia, Luca)$$

Per sequenze molto lunghe, stimare la probabilità condizionata di una parola, data una *sequenza molto lunga* è spesso impossibile.
Questo perché il linguaggio è **creativo** e potremmo avere troppi pochi esempi, o nessuno, per farlo.

Adottiamo un'ulteriore semplificazione con un'**Assunzione Markoviana**: possiamo predirre la prossima parola nella sequenza guardando ad un *contesto passato limitato*.

Esempio in **Bigrammi**:

$$P(mela|la, mangia, Luca) \approx P(mela|la)$$

La versione generale in *n-grammi* si ottiene contando le occorrenze degli *n-grammi* nel corpus e **normalizzandole**, ottenendo così la probabilità di osservare ogni n-gramma (**Maximum Likelihood Estimation**).

Storicamente, i modelli a n-grammi sono stati usati per risolvere vari tipi di **task**, ad esempio:

- Auto-completamento delle parole;
- Correzione automatica del testo
- Sequenziamento del DNA o di Proteine;
- Input a sistemi di Classificazione del testo;

Usi storici di modelli a **n-grammi**

talia

I modelli a n-grammi presentano una serie di problematiche per la quale oggi sono scarsamente in uso rispetto a nuove soluzioni al problema del Language Modeling:

I modelli a n-grammi presentano una serie di problematiche per la quale oggi sono scarsamente in uso rispetto a nuove soluzioni al problema del Language Modeling:

1. **Scarsità dei dati**: alcuni n-grammi potrebbero non apparire mai nei corpora, o apparire con una frequenza troppo bassa, ottenendo quindi una probabilità falsata vicino allo zero o nulla;

I modelli a n-grammi presentano una serie di problematiche per la quale oggi sono scarsamente in uso rispetto a nuove soluzioni al problema del Language Modeling:

1. **Scarsità dei dati**: alcuni n-grammi potrebbero non apparire mai nei corpora, o apparire con una frequenza troppo bassa, ottenendo quindi una probabilità falsata vicino allo zero o nulla;

2. **Vocabolario fisso**: alcune parole potrebbero non apparire nei corpora;

I modelli a n-grammi presentano una serie di problematiche per la quale oggi sono scarsamente in uso rispetto a nuove soluzioni al problema del Language Modeling:

1. **Scarsità dei dati**: alcuni n-grammi potrebbero non apparire mai nei corpora, o apparire con una frequenza troppo bassa, ottenendo quindi una probabilità falsata vicino allo zero o nulla;

2. **Vocabolario fisso**: alcune parole potrebbero non apparire nei corpora;

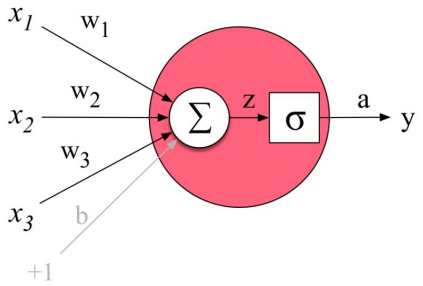
3. **Contesto e polisemanticità**: alcune parole possono avere una semantica *diversa* in base al contesto che, con n-grammi troppo piccoli, potremmo non catturare.
   Esempio: *“Le stelle brillano nel cielo”* e “*Ieri sera le stelle del cinema erano tutte agli Oscar”*. In queste due sequenze, il bigramma {*le, stelle}* ha una semantica completamente diversa.

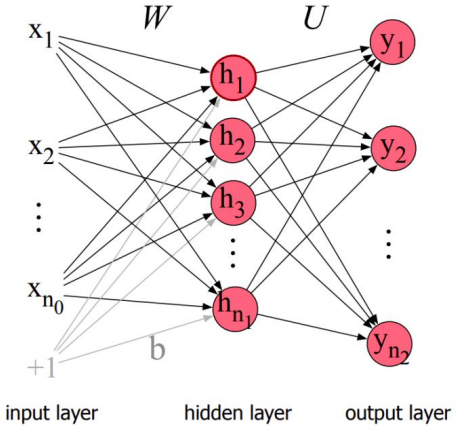
Una rete neurale è composta da singoli neuroni. Ogni neurone calcola una **somma pesata** dei suoi input ottenendo un *singolo valore*.

L'unione di più singoli neuroni crea una *rete neurale*. Il "*modello*" è l'insieme dei pesi della rete, che vengono imparati durante l'*addestramento*.

Nel caso del Language Modeling, ogni input è una **rappresentazione numerica** di una parola.

Come rappresentare le parole numericamente è il problema fondamentale (**Representation Learning**).





Per rappresentare delle parole o delle sequenze in maniera numerica, sono state sviluppate varie strategie:

Per rappresentare delle parole o delle sequenze in maniera numerica, sono state sviluppate varie strategie:

1. **Feature Esplicite**: È possibile rappresentare una parola o una frase con un insieme di valori numerici, ognuno dei quali rappresenta delle *caratteristiche linguistiche*. Ad esempio: la *categoria grammaticale*, l'appartenenza a *classi di interesse*, la *lunghezza in caratteri*, se inizia o meno con una *maiuscola*, etc. (Vettore **denso**);

Per rappresentare delle parole o delle sequenze in maniera numerica, sono state sviluppate varie strategie:

1. **Feature Esplicite**: È possibile rappresentare una parola o una frase con un insieme di valori numerici, ognuno dei quali rappresenta delle *caratteristiche linguistiche*. Ad esempio: la *categoria grammaticale*, l'appartenenza a *classi di interesse*, la *lunghezza in caratteri*, se inizia o meno con una *maiuscola*, etc. (Vettore **denso**);

2. **n-grammi**: È possibile rappresentare una *sequenza* utilizzando gli n-grammi: si contano tutti gli n-grammi che appaiono nel corpus, e si crea un vettore lungo quanto il totale. La frase è rappresentando mettendo 1 sulla posizione che corrisponde agli n-grammi presenti al suo interno, e 0 altrimenti (Vettore **sparso**).

3. **Word embeddings**. I word embeddings sono rappresentazioni *dense* delle parole. Ad ogni parola viene associato un vettore numerico di una lunghezza prefissata. Il *dizionario* che associa ad ogni parola la sua rappresentazione è costruito utilizzando un algoritmo chiamato **skip-gram**.

3. **Word embeddings**. I word embeddings sono rappresentazioni *dense* delle parole. Ad ogni parola viene associato un vettore numerico di una lunghezza prefissata. Il *dizionario* che associa ad ogni parola la sua rappresentazione è costruito utilizzando un algoritmo chiamato **skip-gram**.

   L'algoritmo è una rete neurale che a partire da una parola, prova a predirre la parola precedente e quella successiva.

   Finito l'addestramento, le *attivazioni* della rete rispetto ad una parola, diventano la sua rappresentazione. Passando ogni parola presente nel corpus, otteniamo il dizionario.

Con questo addestramento, la rete incapsula in quelle rappresentazioni informazioni **sintattiche** e **semantiche** relative alla parola. In alcuni casi, alcune di queste relazioni sono state trovate in maniera esplicita:

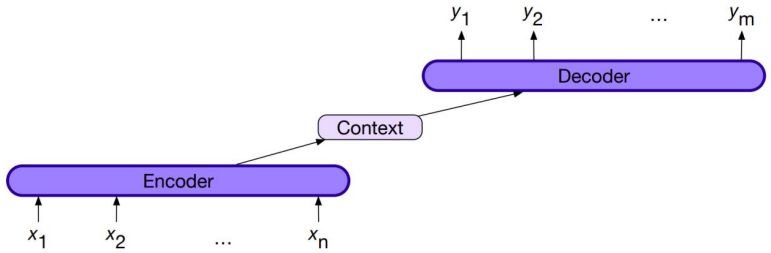
$$\overrightarrow{King} + \overrightarrow{Woman} \approx \overrightarrow{Queen} \qquad \overrightarrow{Paris} - \overrightarrow{France} + \overrightarrow{Italy} \approx \overrightarrow{Rome}$$

Addestrando **reti neurali** e **reti neurali ricorrenti**, utilizzando come input i word embeddings, si è raggiunto nuovi livelli nello stato dell'arte del language modeling e di tutti i task ad esso collegati, in particolare la *classificazione* del testo, la *machine translation*, sistemi di *information retrieval* come motori di ricerca, etc.

Nonostante questo, presentano ancora problemi legati all'avere un **vocabolario fisso** e al fatto che le rappresentazioni **non sono contestuali**.

Una prima evoluzione di queste reti è stata l'aggiunta del meccanismo di **attenzione** attraverso l'**Attention Module**.

Questo pesa quali parti della rete di input (encoder) usare per tradurre la parola corrente su cui lavorava la rete di output (decoder).

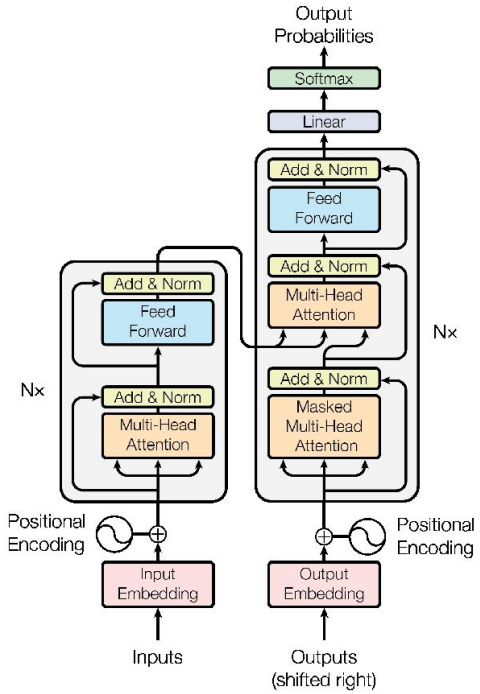


Nel 2017, in *Attention is all you need* (Vaswani et al.) viene presentata una nuova architettura neurale chiamata **Transformer**.

Come altre architetture precedenti, è costituita da due parti, un **encoder** che si occupa di costruire delle rappresentazioni **contestuali** delle parole, e un **decoder** che utilizza queste rappresentazioni per *generare* nuovo testo.

Si basa (quasi) esclusivamente su una variante del meccanismo di **attenzione.**

Supera il problema del **contesto** e quello del **vocabolario fisso**.



Il Transformer

talia

La prima introduzione essenziale che è diventata popolare grazie al Transformer è una nuova tecnica di **tokenizzazione**(Inizialmente introdotta in Wu et al., 2016)

Tokenizzazione Classica: *il, mio, hobby, è, l', automobilismo, .*

Vantaggi: Rappresentazione ricca per ogni token.
Svantaggi: I token possibili sono un numero fisso. Il Dizionario è piuttosto grande.

Tokenizzazione per caratteri: *i,l, m,i,o, h,o,b,b,y, è, l,', a,u,t,o,m,o,b,i,l,i,s,m,o,.*

Vantaggi: Tramite composizione, possiamo formare qualsiasi parola, anche mai vista.
Dizionario piccolo.
Svantaggi: La rappresentazione dei token è povera, e non può incapsulare **semantica** o **sintassi**.

Tokenizzazione per wordpiece: *il, mio, hobby, è, l', automobili-, -smo,.*

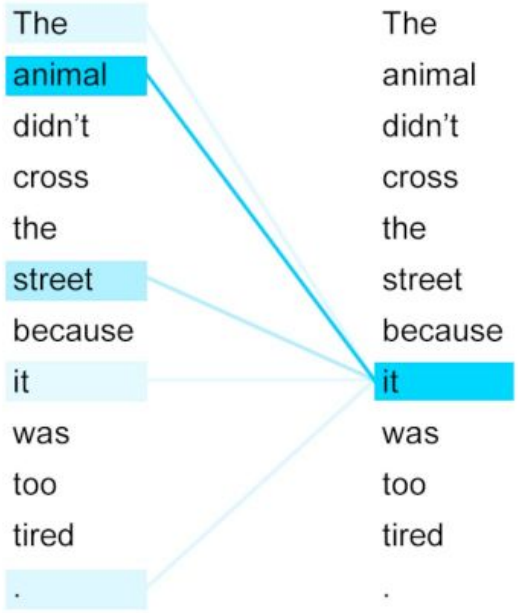
Vantaggi: Tramite composizione, possiamo formare qualsiasi parola, anche mai vista.
La rappresentazione è ricca per ogni token, mantenendo informazioni **semantiche** e **sintattiche**.
Svantaggi: Dizionario grande, ma gestibile.

Il meccanismo di **self-attention** serve a spostare informazione tra un token e l'altro, creando delle rappresentazioni **contestuali** della parola.

Per ogni token, si crea una rappresentazione *pesando le informazioni di ogni altro token nella frase*.

Queste rappresentazioni sono poi passate ad un **Multi Layer Perceptron** (Rete neurale, solitamente a 3 strati), che impara a combinare queste informazioni.

La struttura Self-Attention + Multi Layer Perceptron, costituisce un **transformer block**. Sia l'encoder, che il decoder, sono formati da vari strati di questi blocchi.



I Transformers sono addestrati in due fasi:

- **Pre-training**: Addestramento auto-supervisionato, il modello impara a predirre, data una sequenza, una parola mascherata al suo interno (Task di **Language modeling**, introdotto in *Bengio et al., 2003*). Permette al modello di imparare la *sintassi* e la *semantica* della lingua su cui è addestrato, sfruttando enormi corpora di testi.

I Transformers sono addestrati in due fasi:

- **Pre-training**: Addestramento auto-supervisionato, il modello impara a predirre, data una sequenza, una parola mascherata al suo interno (Task di **Language modeling**, introdotto in *Bengio et al., 2003*). Permette al modello di imparare la *sintassi* e la *semantica* della lingua su cui è addestrato, sfruttando enormi corpora di testi.

- **Fine-tuning**: Addestramento supervisionato, il modello impara il task di interesse utilizzando *pochi dati annotati*. Esempio di task: *sentiment analysis,* classificazione del testo, *machine translation*, *summarization*, semplificazione del testo.

I Transformers hanno dimostrato velocemente di essere la migliore architettura per la maggior parte dei task di elaborazione dei testi.
Su questa base sono nate delle architetture specifiche per risolvere certe categorie di task:

BERT (Bidirectional Encoder Representations from Transformers, *Devlin et al., 2018*) è una delle più famose architetture basate su transformer. È *encoder-only*, ed è pensato per generare ricche rappresentazioni testuali (contestuali) per task di classificazione.

GPT (Generative Pre-training Transformer, *Radford et al., 2018*) è un'architettura *decoder-only* pensato per task di generazione del testo. Ad oggi, la maggior parte dei Large Language Model generativi si basano su questa architettura.

Il Transformer ha **rivoluzionato** l'elaborazione automatica del linguaggio, migliorando lo stato dell'arte di praticamente ogni task legato al testo.

Ha risolto problemi storici di questa disciplina, ma alcuni sono irrisolti e altri sono comparsi, in particolare:

Il Transformer ha **rivoluzionato** l'elaborazione automatica del linguaggio, migliorando lo stato dell'arte di praticamente ogni task legato al testo.

Ha risolto problemi storici di questa disciplina, ma alcuni sono irrisolti e altri sono comparsi, in particolare:

- **Black Box Problem**: I Transformer sono *reti neurali profonde*. I loro pesi e le *"decisioni"* prese da questi modelli **non sono spiegabili**. Cosa che rende il loro utilizzo in sistemi ad alto rischio impossibile (medico, finanziario, etc.)

Il Transformer ha **rivoluzionato** l'elaborazione automatica del linguaggio, migliorando lo stato dell'arte di praticamente ogni task legato al testo.

Ha risolto problemi storici di questa disciplina, ma alcuni sono irrisolti e altri sono comparsi, in particolare:

- **Black Box Problem**: I Transformer sono *reti neurali profonde*. I loro pesi e le *"decisioni"* prese da questi modelli **non sono spiegabili**. Cosa che rende il loro utilizzo in sistemi ad alto rischio impossibile (medico, finanziario, etc.)

- **Lunghezza del contesto**: I Transformer, rispetto ai precedenti Language Model basati su Reti neurali ricorrenti hanno una dimensione di input **fissa** (anche se sempre più grande).

Il Transformer ha **rivoluzionato** l'elaborazione automatica del linguaggio, migliorando lo stato dell'arte di praticamente ogni task legato al testo.

Ha risolto problemi storici di questa disciplina, ma alcuni sono irrisolti e altri sono comparsi, in particolare:

- **Black Box Problem**: I Transformer sono *reti neurali profonde*. I loro pesi e le *"decisioni"* prese da questi modelli **non sono spiegabili**. Cosa che rende il loro utilizzo in sistemi ad alto rischio impossibile (medico, finanziario, etc.)

- **Lunghezza del contesto**: I Transformer, rispetto ai precedenti Language Model basati su Reti neurali ricorrenti hanno una dimensione di input **fissa** (anche se sempre più grande).

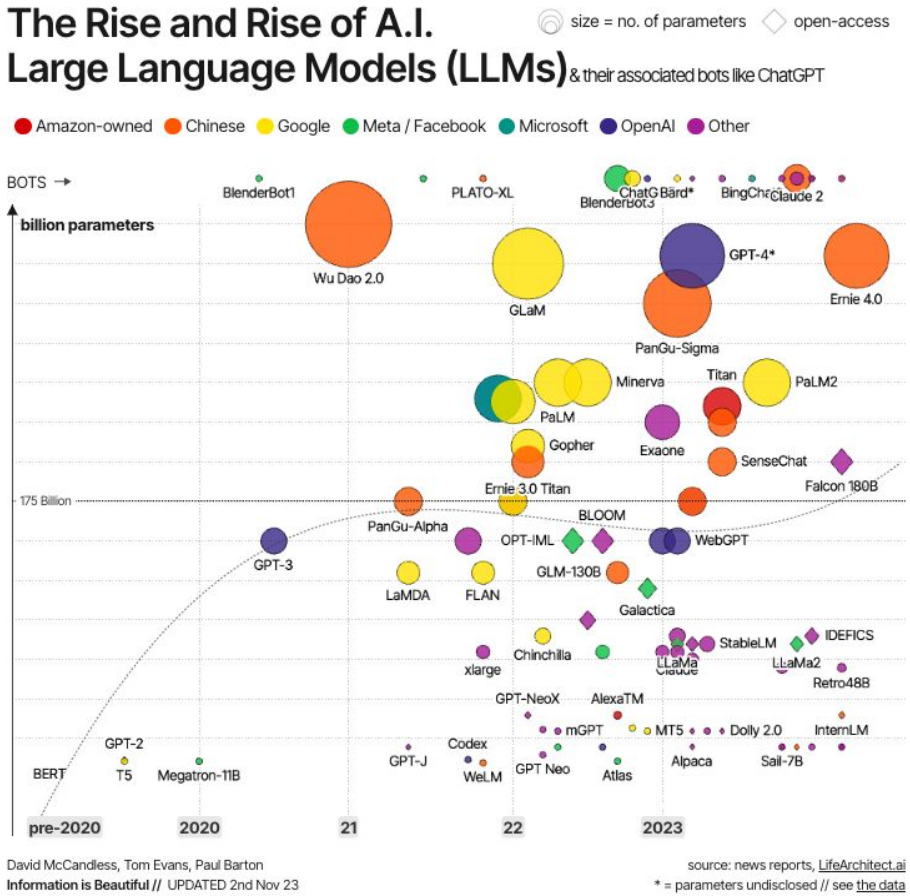
- **Allucinazioni**: Le generazioni testuali di modelli basati su Transformer non sono automaticamente controllabili o verificabili. Talvolta questi producono testi sintatticamente perfetti, ma **errati e presentati come fatti**.

I Language model stanno diventando sempre **più grandi**.

Il numero di parametri è direttamente correlato al **costo energetico** da pagare per l'addestramento e l'uso.

Questo crea una serie di problemi **etici** e di **accessibilità**.

Utilizzare un LLM "*in proprio*" è sempre più difficile



LLMs - **Dimensioni**

talia

Grazie per l’attenzione

Contacts

Mail: michele.papucci@talia.cloud
Twitter: @mpapucci_
Linkedin: linkedin.com/in/michelepapucci

talia