Alma Mater Studiorum · Università di Bologna

**SCUOLA DI SCIENZE**
**Corso di Laurea in Informatica per il Management**

# Verso una migliore comprensione dei modelli linguistici di grandi dimensioni: Spiegabilità attraverso XAI

**Relatore:**
**Chiar.mo Dott.**
**Davide Evangelista**

**Presentata da:**
**Alessandro Verri**

**II Sessione**
**Anno Accademico 2023-2024**

*Al Dott.* Davide Evangelista,
*per avermi introdotto allo studio*
*del tema dell'intelligenza artificiale.*

# Introduzione

Negli ultimi anni, l'avanzamento e la diffusione dell'intelligenza artificiale sono diventati sempre più evidenti. Tuttavia, insieme a questa crescita, emerge la necessità di comprendere come questi modelli funzionino e perché prendano determinate decisioni. Questa comprensione è fondamentale per motivi di sicurezza, per garantire una maggiore trasparenza nei risultati prodotti e per affrontare tematiche riguardanti l'etica. La spiegabilità dei modelli di intelligenza artificiale è essenziale per assicurare che le decisioni prese siano giuste, non discriminatorie e conformi a principi etici fondamentali. Senza una chiara comprensione del funzionamento interno dei modelli, risulta difficile identificare e correggere potenziali bias o comportamenti indesiderati. Inoltre una maggiore comprensione di ciò che influenza le scelta degli LLM (Large Language Models) potrebbe darci maggiore fiducia rispetto ai risultati stessi.

Grazie alle ricerche orientate al raggiungimento dell'AGI (Artificial General Intelligence), sono stati sviluppati i Transformer e gli LLM, che negli ultimi anni hanno sorpreso molti per le loro capacità di ragionamento. Sebbene tali capacità siano ancora relativamente semplici e oggetto di dibattiti, rappresentano un primo sguardo verso le future direzioni dell'intelligenza artificiale.

In questa tesi, attraverso l'utilizzo di vari strumenti di XAI (Explainable Artificial Intelligence), ci proponiamo di:

- Valutare l'efficacia degli strumenti di XAI attuali nel fornire spiegazioni comprensibili e affidabili.
- Esplorare i limiti degli approcci esistenti e proporre potenziali strategie alternative per migliorare la comprensione dei modelli LLM, anche di maggiori dimensioni.

Le principali domande di ricerca che questa tesi intende affrontare sono:

- Quali strumenti di XAI offrono le spiegazioni più chiare e utili per comprendere le decisioni degli LLM?
- Quali sono le principali sfide nell'applicazione di XAI ai modelli LLM di grandi dimensioni e come possono essere superate?

Attraverso questo lavoro, miriamo a contribuire a una migliore comprensione dei modelli linguistici di grandi dimensioni per supportare lo sviluppo di intelligenze artificiali più trasparenti e affidabili.

## Struttura del Lavoro

Questa tesi è organizzata nei seguenti capitoli. Nel **Capitolo 1**, introduciamo i concetti di base quali le reti neurali, i Transformer e gli LLM, fornendo una panoramica essenziale per comprendere gli argomenti trattati successivamente. Il **Capitolo 2** è dedicato all'introduzione generale all'XAI e ai tool utilizzati nel nostro esperimento. Nel **Capitolo 3**, è presente un elenco degl esperimenti che abbiamo effettuato. Il **Capitolo 4** si concentra sui risultati ottenuti dagli esperimenti effettuati. Infine, nel **Capitolo 5**, offriamo una riflessione sui risultati ottenuti, discutendo le implicazioni dei nostri risultati, le limitazioni dello studio e propendo possibili direzioni per ricerche future.

# Indice

# Capitolo 1

# LLM

In questo capitolo, introdurremo i **Large Language Models** (LLM) e le tecnologie fondamentali che ne hanno permesso lo sviluppo, con particolare attenzione alle reti neurali e all'architettura dei Transformer.

Gli LLM si sono diffusi enormemente in questi ultimi due anni, specialmente a seguito del rilascio pubblico di modelli come GPT-3.5 e dall'introduzione di prodotti quali ChatGPT verso la fine del 2022. Da allora, sono stati sviluppati diversi modelli di LLM con vari tipi di licenze. Alcuni di questi sono disponibili pubblicamente tramite licenze più o meno restrittive per l'utilizzo, rendendo disponibili i pesi del modello per l'esecuzione in locale. L'esempio più famoso da questo punto di vista è la famiglia di modelli di LLaMA (Paper della prima versione [1]) pubblicati da Meta (Per le versioni 1/2/Code/3/3.1/3.2).

## 1.1 Una Nuova Primavera dell'Intelligenza Artificiale

Negli ultimi anni, l'interesse pubblico per l'intelligenza artificiale (IA o AI in inglse e come usato in questa tesi) è aumentato esponenzialmente, grazie all'emergere di modelli avanzati di generazione di testo, immagini e video come ChatGPT, DALL·E e Sora. Questi modelli, basati sull'architettura dei Transformer introdotta nel 2017 con il paper *Attention is All You Need* [2], hanno rivoluzionato il modo in cui l'AI interagisce con gli esseri umani, rendendo le sue applicazioni più concrete ed impattanti rispetto al grande pubblico.

Questo nuovo periodio di forte crescita per questo settore si contrappone agli anni '90 e i primi anni 2000 in cui l'AI attraversò periodi noti come "inverni dell'AI", caratterizzati da scetticismo e mancanza di fiducia dovuti a promesse tecnologiche non mantenute. L'AI era spesso infatti confinata a settori accademici o di nicchia, con un interesse pubblico limitato e percepita come una tecnologia distante o deludente.

Il panorama è cambiato radicalmente con l'introduzione dei Transformer, che hanno permesso lo sviluppo di modelli più potenti e versatili.

Secondo una mia personale analisi, un fattore decisivo di questo nuovo boom dell'AI è stata la decisione di aziende come OpenAI e Midjourney di rendere disponibili al pubblico i loro modelli più avanzati. Questo ha permesso al grande pubblico di sperimentare personalmente le ultime innovazioni nell'intelligenza artificiale, trasformando l'AI in un argomento di discussione diffuso non solo tra gli esperti del settore ma anche tra il pubblico generale. Altrimenti, fino a quel momento, tali tecnologie sarebbero rimaste confinate in grandi aziende come DeepMind di Alphabet (Google), limitate a esperimenti e con risultati accessibili solamente tramite i dati limitati degli articoli scientifici rilasciati. Tuttavia, solo nei prossimi anni potremo valutare se questa decisione si rivelerà positiva. La situazione attuale, infatti, sta portando a un avanzamento più rapido e orientato al profitto, rispetto al periodo

precedente, caratterizzato da una maggiore attenzione alla ricerca e da un approccio forse più cauto.

## 1.2 Introduzione alle Reti Neurali

Le **reti neurali** sono lo strumento base su cui si basa l'architettura dei **Transformer** e successivamente degli **LLM**. Le reti neurali di base sono strutture ispirate al funzionamento dei neuroni biologici, sono composte da unità chiamate *neuroni artificiali*, organizzati in strati. Le connessioni tra i neuroni sono rappresentate da archi che simulano le sinapsi cerebrali. Ogni neurone artificiale riceve segnali provenienti dai neuroni dello strato precedente attraverso queste connessioni. Il "segnale" è rappresentato da un numero reale e l'output di ogni neurone viene calcolato mediante una funzione sui dati in input, chiamata funzione di attivazione.

### 1.2.1 La struttura

Le reti neurali sono composte da tre tipi principali di layer, come mostrato in un esempio semplificato in Figura 1.1:

- **Input**: In cui vengono passati i dati iniziali.
- **Hidden layer/s**: Che possono essere presenti o meno. Nel caso in cui fossero presenti due o più, la rete viene definita come Rete Neurale Profonda o Deep Neural Network (DNN).
- **Output layer**: Fornisce il risultato finale dell'elaborazione, ad esempio una classificazione o un valore continuo.

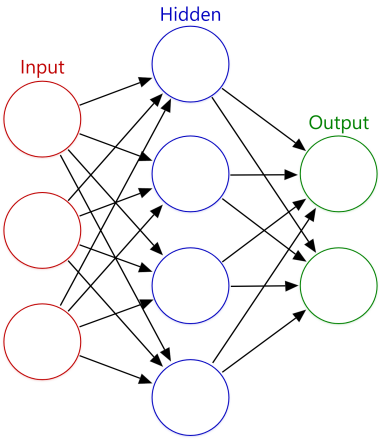
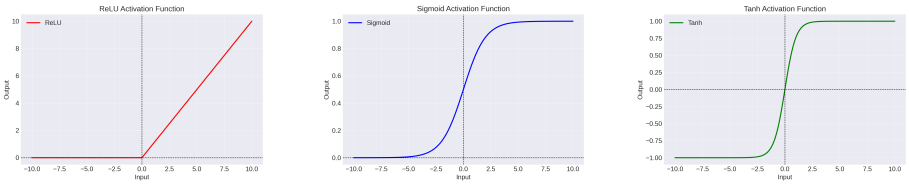



Figura 1.1: Schema di una rete neurale artificiale.

### 1.2.2 Funzione di attivazione

Ogni neurone artificiale funziona tramite una **funzione di attivazione** che introduce una non linearità nel modello, permettendo alla rete di apprendere relazioni complesse nei dati. Alcune delle funzioni di attivazione più comuni includono, come mostrato anche nei grafici in Figura 1.2:

- **ReLU (Rectified Linear Unit)**: $f(x) = \max(0, x)$
- **Sigmoid**: $f(x) = \frac{1}{1+e^{-x}}$
- **Tanh**: $f(x) = \tanh(x)$




(a) ReLU (b) Sigmoid (c) Tanh

Figura 1.2: Visualizzazione delle funzioni di attivazione

Per i layer finali delle reti neurali, invece, sono usate spesso anche altre funzioni di attivazione particolari e più specifiche, quali ad esempio nel caso di classificazione, la:

**Softmax**:

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

dove $K$ rappresenta il numero di classi possibili e che permette di normalizzare tra 0 e 1 gli output della rete.

### 1.2.3 Apprendimento delle Reti Neurali

Prima di essere utilizzate, le reti neurali devono passare attraverso un processo di *training* in cui apprendono dai dati di input regolando i pesi delle connessioni tra i neuroni per minimizzare l'errore nelle previsioni del modello. Questo processo avviene attraverso la combinazione di tecniche di calcolo del gradiente, come la *backpropagation* [3], e algoritmi di ottimizzazione, come lo *Stochastic Gradient Descent (SGD)* ([4], [5], [6]) o l'algoritmo *Adam* ([7]).

Per ulteriori dettagli sugli algoritmi di ottimizzazione, vedere le appendici A e B.

Durante l'addestramento, la rete neurale passa attraverso numerosi cicli (epoch) in cui viene esposta a diversi campioni di dati, migliorando progressivamente le sue prestazioni.

Il processo di backpropagation aggiorna iterativamente i pesi delle connessioni secondo la seguente formula:

$$w_{k+1} = w_k - \eta_k \frac{\partial \mathcal{L}_k}{\partial w_k}$$

dove:

- $w_k$: Rappresenta il peso della connessione nel modello di rete neurale al tempo `"k"`. I pesi determinano l'importanza delle diverse connessioni tra i neuroni.
- $\eta_k$ (eta): È il *tasso di apprendimento* (learning rate). Questo parametro controlla la dimensione del passo con cui i pesi vengono aggiornati

durante l'ottimizzazione. Un valore di $\eta_k$ troppo grande può causare oscillazioni e impedire la convergenza, mentre un valore troppo piccolo può rallentare significativamente il processo di apprendimento.

- $\frac{\partial \mathcal{L}_k}{\partial w_k}$: Rappresenta la *derivata parziale* della funzione di perdita $\mathcal{L}_k$ rispetto al peso $w_k$. Questa derivata misura quanto la funzione di perdita cambia al variare del peso $w_k$. In altre parole, indica la direzione e la magnitudine del cambiamento necessario per ridurre l'errore del modello.

**Approcci alternativi:** Sebbene la backpropagation e algoritmi come SGD e Adam siano ampiamente utilizzati, esistono metodologie alternative per l'addestramento delle reti neurali. Ad esempio, in alcuni modelli di apprendimento non supervisionato o in reti neurali evolutive, possono essere impiegati metodi di aggiornamento dei pesi diversi che non richiedono il calcolo esplicito dei gradienti.

## 1.3 Architettura Transformer

### 1.3.1 Introduzione

L'architettura **Transformer**, introdotta da Vaswani et al. nel 2017 [2], ha rappresentato una svolta significativa nel campo dell'elaborazione del linguaggio naturale (NLP, Natural language processing). Prima dei Transformer, le **Reti Neurali Ricorrenti** (RNN) e le **Long Short-Term Memory** (LSTM) [8] erano ampiamente utilizzate per gestire dati sequenziali. Le RNN elaborano le informazioni in sequenza, mantenendo uno stato interno che può teoricamente catturare dipendenze a lungo raggio. Tuttavia, tali modelli soffrono di alcune limitazioni come la difficoltà nel parallelizzare i calcoli e la gestione delle dipendenze a lungo raggio, problema mitigato successivamente dalle LSTM introducendo celle di memoria e meccanismi di gating.

Per una trattazione più dettagliata delle RNN e delle LSTM, si rimanda rispettivamente all'Appendice C e all'Appendice D.

### 1.3.2 Vantaggi rispetto ad altre architetture

Per cercare di superare le limitazioni di queste architetture sono stati introdotti i Transformer, che presentano diversi vantaggi:

- **Gestione delle Dipendenze a Lungo Termine**:
  - **RNN**: Le RNN tradizionali faticano a catturare dipendenze a lungo termine a causa del problema del *vanishing gradient*.
  - **LSTM**: Gli LSTM mitigano parzialmente questo problema grazie alle celle di memoria, ma possono comunque incontrare difficoltà con sequenze molto lunghe.
  - **Transformer**: Grazie al meccanismo di self-attention, i Transformer possono pesare l'importanza di ogni parola nella sequenza rispetto alle altre, indipendentemente dalla loro distanza, migliorando l'efficacia nel catturare relazioni a lungo termine.
- **Parallelizzazione, Scalabilità ed Efficienza Computazionale**:
  - **RNN e LSTM**: Queste architetture elaborano le sequenze in modo prevalentemente sequenziale, limitando la possibilità di sfruttare appieno la parallelizzazione. La natura ricorrente delle loro operazioni complica la scalabilità e riduce l'efficienza computazionale, rallentando sia l'addestramento che l'inferenza [1].
  - **Transformer**: Grazie al meccanismo di self-attention, i Transformer possono elaborare l'intera sequenza contemporaneamente, migliorando la parallelizzazione e velocizzando l'addestramento. Questo approccio semplifica anche la scalabilità, consentendo di aumentare il numero di strati e parametri senza i tipici limiti delle RNN/LSTM. Di conseguenza, l'elaborazione è più efficiente, facilitando il pieno utilizzo delle GPU o TPU e riducendo i tempi di addestramento e inferenza[1].

[1] L'inferenza è il processo di utilizzo di un modello addestrato per fare previsioni o prendere decisioni basate su nuovi dati.

### 1.3.3 Componenti Chiave

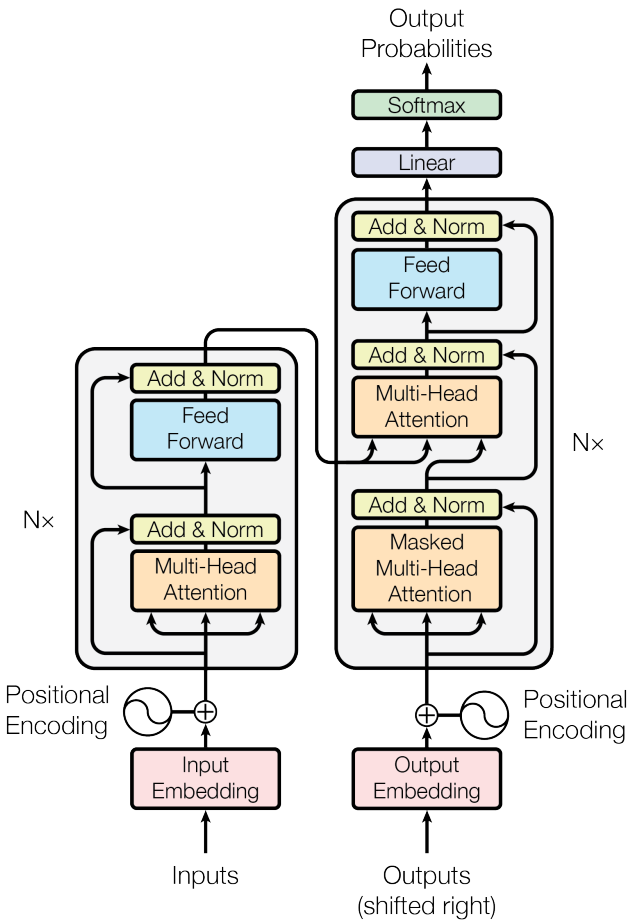



Figura 1.3: Schema dell'architettura del Transformer [2]

L'architettura del Transformer (Schema 1.3) è composta principalmente da due parti: l'**encoder** e il **decoder**, ciascuno formato da più strati identici. L'encoder elabora l'input e genera rappresentazioni intermedie, mentre il decoder utilizza queste rappresentazioni per generare l'output, ad esempio una traduzione in un'altra lingua.

Grazie alla sua flessibilità e potenza, l'architettura Transformer ha portato a significativi miglioramenti in numerosi compiti tra cui anche quelli di NLP, come la traduzione automatica, la generazione di testo e la comprensione del linguaggio. Inoltre, ha permesso lo sviluppo di modelli pre-addestrati

di grandi dimensioni, come BERT [9] e GPT [10], che hanno ulteriormente spinto avanti le frontiere dell'intelligenza artificiale nel trattamento del linguaggio naturale.

Per comprendere appieno il funzionamento dei Transformer, introdurremo brevemente i loro componenti chiave:

### Meccanismo di Self-Attention

Il cuore dell'architettura Transformer è il meccanismo di **self-attention** che permette al modello di assegnare pesi diversi alle diverse parti della sequenza di input. Questo consente di catturare le dipendenze tra parole che possono essere distanti tra loro nella sequenza, migliorando la comprensione del contesto e permettendo di focalizzarsi meglio sulle parti più rilevanti della sequenza di input.

Matematicamente, il meccanismo di attenzione è definito come:

$$\text{Self-Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dove:

- $Q$ (Query), $K$ (Key) e $V$ (Value) sono matrici ottenute trasformando l'input attraverso matrici di peso apprese.
- $d_k$ è la dimensione delle chiavi (e delle query).
- La funzione softmax normalizza i pesi di attenzione.

### Multi-Head Attention

Il meccanismo di **Multi-Head Attention** (Come mostrato anche nel grafico 1.4) estende il concetto di self-attention applicando diverse proiezioni lineari in parallelo, permettendo al modello di focalizzarsi su diverse parti della sequenza contemporaneamente:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

Dove ogni $\text{head}_i$ è un'attenzione singola definita come:

$$\text{head}_i = \text{Attenzione}(QW_i^Q, KW_i^K, VW_i^V)$$

Dove le proiezioni sono matrici di parametri:

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \quad W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}.$$

Questo approccio permette al Transformer di catturare una varietà di relazioni tra le parole, migliorando la capacità del modello di comprendere il contesto e la semantica del testo.

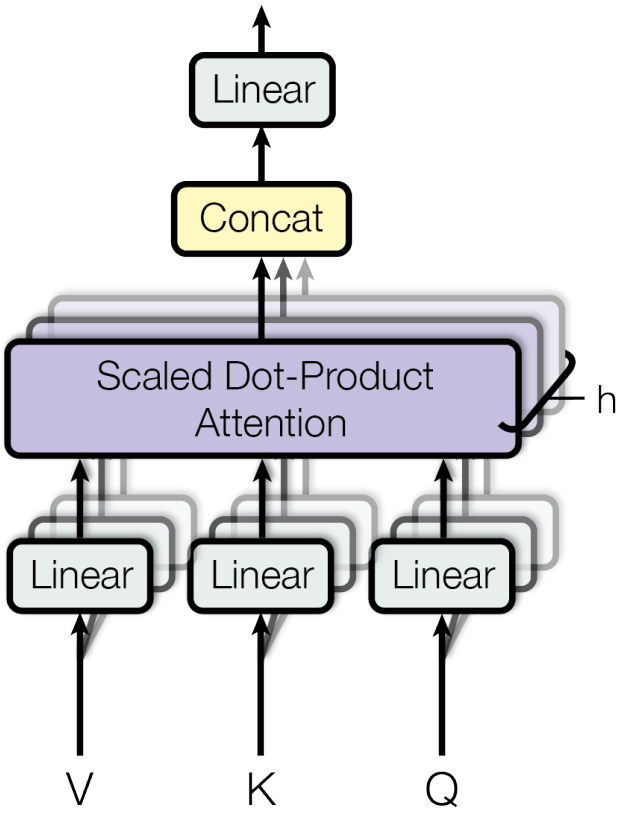



Figura 1.4: Schema architettura Multi-Head Attention [2]

### Positional Encoding

Poiché i Transformer non possiedono una struttura sequenziale intrinseca, il meccanismo di **Positional Encoding** aggiunge informazioni sulla posizione delle parole nella sequenza. Questo viene fatto tramite l'aggiunta di vettori di posizione agli embeddings delle parole, permettendo al modello di considerare l'ordine delle parole durante l'elaborazione.

Le posizioni vengono codificate usando funzioni sinusoidali:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Questi vettori vengono aggiunti agli embeddings delle parole, fornendo al modello informazioni sulla posizione relativa delle parole nella sequenza.

### Feed-Forward Networks

Le **Feed-Forward Networks** sono reti neurali completamente connesse applicate a ogni posizione della sequenza. La loro struttura tipica è:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

dove $W_1$ e $W_2$ sono matrici di peso e $b_1$, $b_2$ sono bias.

### Normalizzazione e Residual Connections

Le **Residual Connections** e la **Layer Normalization** migliorano la stabilità del training e facilitano la propagazione dei gradienti attraverso la rete. Le residual connections permettono ai segnali di bypassare alcuni strati, mentre la normalizzazione stabilizza l'attivazione delle reti neurali.

### 1.3.4 Encoder e Decoder

Tutti i componenti descritti precedentemente si uniscono poi per formare le due strutture principali dell'architettura:

- **Encoder**: Responsabile di leggere e comprendere l'input. È composto da più strati identici, ognuno dei quali contiene un meccanismo di self-attention e una rete neurale feed-forward. L'encoder trasforma l'input in rappresentazioni intermedie che catturano le caratteristiche semantiche della sequenza.

- **Decoder**: Utilizzato nei modelli di traduzione o generazione del testo, genera l'output basandosi sull'input elaborato dall'encoder e sui token precedentemente generati. Ogni strato del decoder include un sottostrato di self-attention, un sottostrato di attenzione sull'encoder e una rete neurale feed-forward. Questo permette al decoder di integrare informazioni dall'encoder e di generare sequenze di output coerenti.

Nei Large Language Models come GPT (Generative Pre-trained Transformer), viene utilizzata principalmente la parte di decoder dell'architettura Transformer per generare testo. Questo approccio permette al modello di prevedere il token successivo nella sequenza, basandosi sul contesto fornito dai token precedenti.

### 1.3.5 Applicazioni

L'architettura Transformer ha rivoluzionato diversi ambiti dell'intelligenza artificiale, estendendo il suo utilizzo ben oltre la traduzione automatica. Alcune delle sue applicazione si possono trovare ad esempio nella:

- **Generazione di Testo**: Modelli come GPT sono in grado di creare testi coerenti e contestualmente pertinenti, trovando impiego in chatbot e strumenti di scrittura automatica.

- **Comprensione del Linguaggio Naturale**: Modelli come BERT (Bidirectional Encoder Representations from Transformers) migliorano le prestazioni in compiti come l'analisi del sentiment, la risposta a domande e la classificazione del testo. Queste capacità sono infatti già state integrate in motori di ricerca quali Google, migliorando la pertinenza dei risultati forniti.

- **Sintesi e Riassunto Automatico del Testo**: Questi modelli sono capaci di condensare grandi quantità di testo mantenendo le informazioni essenziali, facilitando la creazione di riassunti automatici di articoli o documenti.

- **Riconoscimento Vocale e Traduzione Multimodale**: I Transformer vengono utilizzati per combinare informazioni da diverse modalità, come testo e audio, migliorando la precisione nel riconoscimento vocale e nella traduzione. Ad esempio, sistemi di traduzione simultanea multilingue beneficiano di queste tecniche per offrire traduzioni più accurate in tempo reale.

- **Analisi delle Sequenze Biologiche**: I Transformer sono stati applicati nello studio di sequenze biologiche, come il DNA e le proteine, contribuendo a scoperte nel campo della bioinformatica. Ad esempio, modelli come AlphaFold [11] hanno rivoluzionato la previsione della struttura proteica, accelerando la ricerca in biologia molecolare.

## 1.4 Large Language Models (LLM)

I **Large Language Models** (LLM) sono modelli di intelligenza artificiale basati sull'architettura Transformer, progettati per comprendere e generare linguaggio naturale.

### 1.4.1 Definizione

Un **Large Language Model** è un modello di AI basato sull'architettura del Transformer ed addestrato su enormi quantità di dati testuali per prevedere la parola successiva in una sequenza, consentendo così la generazione di testo di senso compiuto e contestualmente rilevante.

### 1.4.2 Capacità emergenti

Man mano che la dimensione e la complessità dei LLM aumentano, si osservano fenomeni noti come **capacità emergenti**. Queste sono abilità che i modelli non mostrano in forma embrionale nei modelli più piccoli, ma che iniziano a manifestarsi in modo evidente solo oltre una certa soglia di dimensioni e di dati di addestramento. Esempi di capacità emergenti possono includere la capacità di svolgere compiti complessi di ragionamento o di eseguire traduzioni più accurate tra diverse lingue.

### 1.4.3 Progresso e Sviluppo

L'evoluzione degli LLM ha seguito una traiettoria esponenziale in termini di dimensioni, complessità e capacità negli ultimi anni. Di seguito presentiamo una panoramica dei principali sviluppi, evidenziando le innovazioni chiave.

- **ELMo (Febbraio 2018):** Sviluppato da AllenNLP, ELMo ha introdotto rappresentazioni contestuali delle parole, permettendo una migliore comprensione del contesto rispetto ai modelli precedenti basati

su word embeddings statici. Questa innovazione ha migliorato significativamente le prestazioni in compiti di comprensione del linguaggio naturale.

- **GPT (Giugno 2018):** Il primo modello della serie *Generative Pre-trained Transformer* di OpenAI ha segnato un'importante svolta nella generazione del testo. Basato sull'architettura Transformer, GPT ha dimostrato la capacità di generare testi coerenti e fluidi, aprendo la strada a modelli più avanzati.

- **GPT-3.5 e ChatGPT (Novembre 2022):** GPT-3.5 ha introdotto miglioramenti nelle capacità di ragionamento e interattività, permettendo anche una prima riduzione dei costi che ha permesso poi il rilascio di ChatGPT. Questo chatbot è stato un passo fondamentale per dimostrare pubblicamente per la prima volta al grande pubblico una prima applicazione degli LLM.

- **LLaMA di Meta (Febbraio 2023):** Meta rilascia il primo modello "open source" [2] di LLM, una di una lunga serie riaprendo le speranze per modelli di grandi dimensioni pubblici dopo la scelta di OpenAI di non rilasciare i pesi dei modelli GPT-3 e successivi. Promuovendo l'open science e permettendo ulteriori sviluppi e ottimizzazioni da parte della comunità accademica e industriale.

- **GPT-4 (Marzo 2023):** OpenAI ha lanciato GPT-4 con capacità avanzate di comprensione e generazione del linguaggio. Sebbene i dettagli sul numero di parametri non siano stati divulgati, GPT-4 ha mostrato miglioramenti significativi nella gestione di contesti complessi e nella generazione di risposte più accurate e pertinenti. Mostrando

[2]Sebbene Meta abbia rilasciato LLaMA sotto una licenza che consente ampi utilizzi accademici e di ricerca, i suoi modelli non soddisfano più i criteri dell'Open Source Initiative (OSI) per essere considerati "open source" in senso stretto. Per ulteriori dettagli, si veda: `https://opensource.org/blog/metas-llama-2-license-is-not-open-source` e `https://opensource.org/ai/open-source-ai-definition`

anche i primi segnali di una maggiore comprensione del testo e una significativamente migliorata affidabilità rispetto ai modelli precedenti.

### Attori Principali nello Sviluppo

Attualmente, lo sviluppo degli LLM è dominato da alcune delle più grandi aziende tecnologiche, che investono massicciamente in ricerca e sviluppo per spingere oltre i limiti delle capacità linguistiche artificiali, lasciando quindi per il momento lo sviluppo abituale degli studi agli ambienti scientifici universitari e accademici a causa degli elevati costi della ricerca.

I principali attori attuali includono:

- **OpenAI e Microsoft:** Collaborazione strategica che ha portato allo sviluppo della serie GPT, inclusi i recenti modelli GPT-4o e o1 (settembre 2024).
- **Google:** Attraverso modelli come BERT, LaMDA, PaLM e la recente serie Gemini, Google continua a innovare nel campo degli LLM, integrando tecniche avanzate di comprensione e generazione del linguaggio.
- **Anthropic:** Con i modelli Claude offre soluzioni avanzate come Claude 3.5 Sonet e Claude 3.5 Opus.
- **Meta:** Con la serie LLaMA, di cui le piu recenti versioni la 3.1 e la 3.2 disponibili in varie dimensioni di parametri.

### Gli ultimo modelli più avanzati

Negli ultimi anni, le principali aziende hanno rilasciato modelli avanzati che rappresentano lo stato dell'arte nella tecnologia degli LLM. Tra questi, i più significativi includono:

- **o1 di OpenAI (Settembre 2024):** [12] o1 è progettato per migliorare il ragionamento e la risoluzione di problemi complessi, con applicazioni

avanzate in settori come la scienza e la programmazione. Questo modello incorpora tecniche di apprendimento più sofisticate per aumentare la precisione e l'affidabilità delle risposte.

- **Gemini 1.5 Pro di Google (Febbraio 2024):** [13] Questo modello sperimentale introduce miglioramenti significativi nelle prestazioni di codifica, nelle capacità di ragionamento e nella comprensione visiva. Ha raggiunto le prime posizioni nei benchmark di prestazioni per l'AI, superando i modelli precedenti.

- **LLaMA 3.1 e 3.2 di Meta (Settembre 2024):** La nuova famiglia di modelli LLaMA includono rispettivamente versioni da 8, 70 e 405 miliardi di parametri e 1, 3, 11 e 90 miliardi di parametri. La variante con 405 miliardi di parametri, rappresenta il più grande modello "open-source" realizzato finora, offrendo potenzialità avanzate per la ricerca e lo sviluppo.

- **Claude 3.5 Sonnet di Anthropic (Giugno 2024):** Questo modello rappresenta un significativo avanzamento nell'intelligenza artificiale, offrendo prestazioni eccezionali in vari ambiti. Claude 3.5 Sonnet eccelle nel ragionamento complesso, nella generazione di codice e nell'interpretazione visiva.

### 1.4.4 Architettura

Ogni LLM, soprattutto quelli più moderni hanno architetture diversificate, per semplicità, prenderemo in considerazione l'architettura del **GPT originale** (*Generative Pre-trained Transformer*), introdotto da OpenAI nel 2018 [10].

**GPT-1** utilizza esclusivamente la parte di **decoder** dell'architettura Transformer, ottimizzata per la generazione di testo in modo autoregressivo. L'architettura di GPT-1 è composta da (Come in figura 1.5):

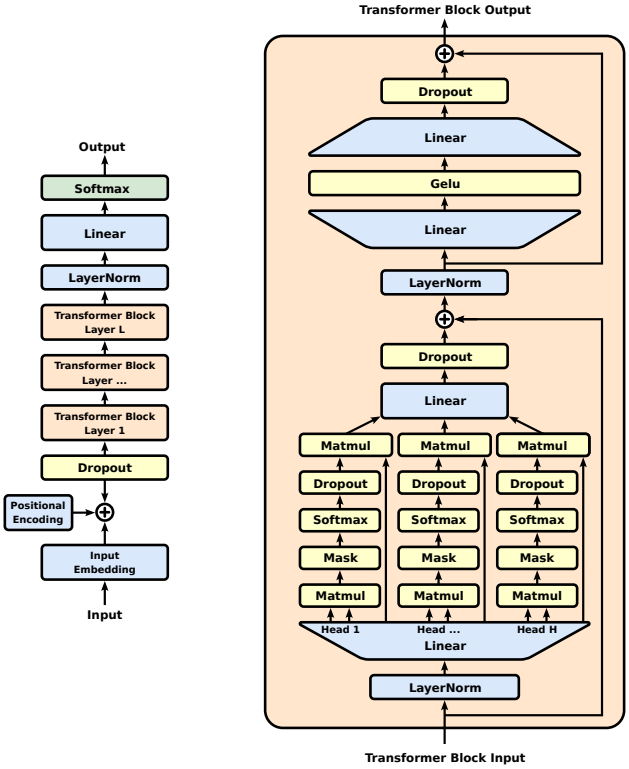



Figura 1.5: Schema dell'architettura del GPT originale basato su Transformer.

- **Strati di Decoder**: Il modello è formato da 12 strati (*layers*) di decoder Transformer. Ogni strato comprende un meccanismo di **self-attention mascherata** e una rete **feed-forward** completamente connessa.

- **Self-Attention Mascherata**: A differenza del self-attention standard, la self-attention mascherata impedisce al modello di attendere posizioni future nella sequenza, garantendo che la previsione di ogni token dipenda solo dai token precedenti.

- **Connessioni Residuali e Normalizzazione**: Ogni sottostrato è seguito da una connessione residuale e da una **Layer Normalization**, migliorando la propagazione del gradiente e la stabilità dell'addestramento.

- **Positional Encoding**: Poiché il Transformer non ha una struttura sequenziale intrinseca, vengono aggiunti **Positional Encodings** agli embeddings dei token per incorporare informazioni sulla posizione nella sequenza.

**Meccanismo di Self-Attention Mascherata** In GPT-1, il meccanismo di self-attention viene esteso introducendo una maschera di attenzione $M$ per garantire che il modello non possa accedere a informazioni future durante la previsione del token corrente. La funzione di attenzione mascherata è definita come:

$$\text{Attenzione}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V$$

Dove $M$ è una matrice di maschera che assegna valori molto negativi alle posizioni future, impedendo al modello di considerarle durante il calcolo dell'attenzione.

L'introduzione della maschera $M$ è fondamentale per mantenere la causalità nel modello, assicurando che le previsioni per un determinato token si basino solo sui token precedenti e non su quelli futuri.

**Obiettivo di Addestramento** GPT-1 è addestrato utilizzando un obiettivo di **modellazione del linguaggio** autoregressivo, che consiste nel massimizzare la probabilità dei token successivi dati i token precedenti. Formalmente, l'obiettivo di addestramento è minimizzare:

$$\mathcal{L} = -\sum_{t=1}^{N} \log P(x_t | x_{<t})$$

Dove:

- $x_t$ è il token al tempo $t$.
- $x_{<t}$ rappresenta la sequenza di token precedenti al tempo $t$.
- $N$ è la lunghezza della sequenza di addestramento.

**Dimensioni del Modello** Il GPT-1 originale ha circa 117 milioni di parametri, una dimensione considerevole per l'epoca, che ha permesso al modello di catturare complesse strutture linguistiche.

### Caratteristiche Principali

Le principali caratteristiche dell'architettura di GPT-1 includono:

- **Pre-addestramento Non Supervisionato**: Il modello viene pre-addestrato su grandi quantità di testo non etichettato, apprendendo le regolarità del linguaggio naturale.
- **Fine-tuning Supervisionato**: Dopo il pre-addestramento, GPT-1 viene adattato a specifici compiti di NLP attraverso il fine-tuning su dataset etichettati più piccoli.
- **Transfer Learning**: Questa combinazione di pre-addestramento e fine-tuning consente al modello di trasferire conoscenze apprese su dati non supervisionati a compiti specifici, migliorando le prestazioni.

### Importanza dell'Architettura di GPT-1

L'architettura di GPT-1 ha segnato un punto di svolta nel campo dell'elaborazione del linguaggio naturale, dimostrando l'efficacia dell'approccio basato su Transformer per la generazione di testo. Ha introdotto il concetto di pre-addestramento su larga scala seguito da fine-tuning, che è diventato uno standard per i modelli successivi.

### Limitazioni

Nonostante i successi, GPT-1 presentava alcune limitazioni:

- **Capacità Limitata**: Con 117 milioni di parametri, il modello aveva una capacità inferiore rispetto ai modelli successivi, limitando la complessità delle rappresentazioni che poteva apprendere.

- **Comprensione Contestuale Limitata**: Essendo un modello unidirezionale (left-to-right), GPT-1 non poteva utilizzare informazioni future nel contesto, limitando la sua capacità di comprensione rispetto a modelli bidirezionali come BERT.
- **Bias nei Dati**: Come molti modelli addestrati su grandi quantità di dati testuali, GPT-1 poteva ereditare bias presenti nel corpus di addestramento.
- **Limitata Scalabilità**: L'architettura iniziale non era ottimizzata per sfruttare pienamente le risorse computazionali disponibili, rendendo difficile l'addestramento di modelli su scale più grandi.

#### Evoluzione Successiva

L'architettura di GPT-1 ha posto le basi per lo sviluppo di modelli più avanzati, come GPT-2, GPT-3 e GPT-4, che hanno aumentato significativamente il numero di parametri e la complessità dell'architettura, migliorando le capacità di generazione e comprensione del linguaggio naturale. Questi modelli hanno introdotto tecniche come il **fine-tuning su istruzioni** e l'utilizzo di **dataset più diversificati**, ampliando ulteriormente le applicazioni degli LLM.

### 1.4.5 Applicazioni

I **Large Language Models** hanno trovato applicazione in una vasta gamma di settori, grazie alla loro capacità di comprendere e generare linguaggio naturale in modo sofisticato. Alcune delle principali applicazioni includono:

- **Assistenti Virtuali**: I modelli linguistici come GPT-3.5 e ChatGPT vengono utilizzati per sviluppare chatbot avanzati capaci di sostenere conversazioni naturali, migliorando l'esperienza del customer service e fornendo assistenza personalizzata. Inoltre, supportano la generazione automatica di contenuti testuali, facilitando la creazione di articoli, report e riassunti.

- **Supporto agli Sviluppatori**: Strumenti basati su LLM assistono gli sviluppatori nella scrittura e nel completamento del codice, suggerendo frammenti, completando funzioni e generando interi blocchi di codice a partire da descrizioni in linguaggio naturale, aumentando l'efficienza e la produttività nello sviluppo software.

- **Assistenza nello Studio**: Gli LLM facilitano lo studio di argomenti complessi supportando gli studenti grazie ad un processo di apprendimento personalizzato, adattando il materiale didattico alle esigenze individuali e semplificando la comprensione dei concetti più difficili.

Le potenzialità degli LLM inoltre continuano a espandersi man mano che vengono sviluppati modelli più avanzati e vengono esplorate nuove applicazioni in diversi settori.

# Capitolo 2

# XAI

## 2.1 Il problema dell'approccio Black Box

I modelli complessi di intelligenza artificiale, come anche gli LLM, sono spesso considerati *black box* a causa della difficoltà nel comprenderne il loro funzionamento interno. Sebbene sia possibile osservare gli input e gli output, i processi che portano ai risultati restano opachi, rendendo difficile spiegare le decisioni del modello e compromettendone l'affidabilità in contesti critici [14].

In particolare, gli LLM sono costituiti da milioni o miliardi di connessioni neuronali che interagiscono in modi complessi e non sempre interpretabili. Questa complessità rende difficile comprendere come un LLM arrivi a una determinata conclusione o generi una specifica risposta.

La comprensione, anche parziale, del funzionamento degli LLM sarebbe utile per diverse ragioni. In settori come la medicina, è essenziale sapere perché un sistema automatizzato ha emesso una determinata diagnosi o raccomandazione, al fine di garantire la sicurezza e l'affidabilità delle decisioni cliniche. Analogamente, nell'ambito delle risorse umane, un algoritmo utilizzato per la selezione dei candidati deve poter giustificare i criteri di valutazione adottati, assicurando l'assenza di bias etici o discriminazioni nel processo di selezione.

## 2.2 Introduzione all'Explainable AI (XAI)

Per affrontare le sfide poste dalla natura *black box* dei modelli di AI, è emersa la necessità di sviluppare metodi che ne permettano la comprensione. L'**Explainable AI (XAI)** ha l'obiettivo di rendere i modelli di AI più interpretabili e trasparenti, fornendo spiegazioni comprensibili delle decisioni prese dai modelli, aumentando la fiducia degli utenti nei sistemi di AI e facilitando l'identificazione di possibili errori o bias.

### 2.2.1 Obiettivi dell'XAI

Gli obiettivi principali dell'Explainable AI includono:

- **Trasparenza**: Fornire una visione chiara del funzionamento interno dei modelli di AI.
- **Affidabilità**: Aumentare la fiducia degli utenti nelle decisioni prese dai sistemi di AI.
- **Giustizia ed Equità**: Identificare e mitigare bias che possono portare a decisioni discriminatorie.
- **Conformità Normativa**: Soddisfare requisiti legali e regolamentari che richiederanno probabilmente una spiegabilità delle decisioni automatizzate.

### 2.2.2 Tecniche di XAI

Le tecniche di XAI possono essere classificate secondo vari criteri:

- **Intrinseche vs Post-hoc**: Tecniche intrinseche si basano su modelli interpretabili per natura (ad esempio gli alberi decisionali), mentre le tecniche post-hoc cercano di spiegare modelli complessi dopo l'addestramento.

- **Globali vs Locali**: Le spiegazioni globali descrivono il comportamento generale del modello, mentre quelle locali si focalizzano su singole predizioni.
- **Model-specific vs Model-agnostic**: Alcune tecniche sono progettate per specifici tipi di modelli, altre possono essere applicate indipendentemente dall'architettura.
- **Simboliche vs Statistiche**: Le tecniche simboliche utilizzano rappresentazioni logiche o regole, mentre quelle statistiche si basano su misure quantitative come l'importanza delle caratteristiche.
- **White-box vs Black-box**: Le tecniche white-box hanno accesso completo alla struttura interna del modello; le black-box trattano il modello come una scatola nera, basandosi solo su input e output.
- **Dominio-specifiche vs Generiche**: Alcune tecniche sono progettate per settori specifici (es. visione artificiale, elaborazione del linguaggio naturale), mentre altre sono applicabili trasversalmente.

### 2.2.3 Sfide nell'XAI

Nonostante i progressi, l'XAI affronta diverse sfide:

- **Complessità dei Modelli**: Modelli come gli LLM sono estremamente complessi, rendendo difficile fornire spiegazioni semplici e comprensibili.
- **Compromesso tra Accuratezza e Interpretabilità**: Spesso le spiegazioni dell'XAI più interpretabili sono anche meno accurate e viceversa.

Esiste anche una sottile differenza tra i termini *interpretability* ed *explainability* nel contesto dell'intelligenza artificiale.

- **Interpretability**: Livello di comprensione di come funziona la tecnologia (AI) sottostante.

- **Explainability**: Livello di comprensione di come il sistema basato sull'AI ha prodotto un dato risultato.

## 2.3 Stato dell'arte sulla spiegabilità degli LLM

La spiegabilità dei modelli di intelligenza artificiale (XAI) ha iniziato a svilupparsi con tecniche come LIME [15] e SHAP [16], fornendo un solido punto di partenza per comprendere il comportamento locale dei modelli. Tuttavia, con l'avvento dei Large Language Models (LLM), l'XAI ha affrontato nuove sfide a causa della complessità e delle capacità avanzate di questi modelli. Infatti nonostante il recente boom del settore dell'intelligenza artificiale, l'XAI non ha seguito lo stesso ritmo di evoluzione.

Gli LLM, caratterizzati da una natura subsimbolica e un numero enorme di parametri, rendono difficile interpretare il processo decisionale interno. Problematiche come bias nei dati di addestramento, allucinazioni nelle risposte e difficoltà nel verificare la coerenza delle risposte evidenziano l'urgenza di sviluppare tecniche di XAI adeguate [17]. Inoltre, è fondamentale adottare un approccio equilibrato che valorizzi l'interpretabilità al pari dei progressi funzionali [18].

Recenti approcci nell'XAI per LLM includono:

- **Tecniche di attribuzione e visualizzazione**: Strumenti come Grad-CAM e saliency maps [19] sono stati adattati per fornire spiegazioni visive dei contributi dei token d'ingresso alle predizioni, spesso arricchite con spiegazioni testuali generate dagli stessi LLM. Ad esempio, [20] introduce "XAIstories", che sfrutta gli LLM per fornire narrazioni sulle decisioni dell'AI.

- **Interpretabilità post-hoc**: Metodi come il prompting esplicativo [21] utilizzano i LLM stessi per generare narrazioni che spiegano i loro output, migliorando l'interpretabilità e l'usabilità delle spiegazioni [21].

- **Metodologie iterative e di auto-riflessione**: Tecniche come la "self-reflection" [22] consentono ai modelli di iterare sulle proprie risposte, verificandone coerenza, factualità e logica. Parallelamente, un approccio di valutazione human-centered degli LLM [23] sottolinea l'importanza di considerare i bisogni e le prospettive degli utenti nel processo di spiegazione e di interazione uomo-macchina.

- **Sistemi ibridi e spiegabilità causale**: La combinazione di LLM con modelli simbolici o l'uso di tecniche di inferenza causale può migliorare la spiegabilità. Ad esempio, [24] propone un approccio guidato dagli LLM per generare spiegazioni controfattuali per classificatori di testo considerati come "black-box" [25].

- **Coinvolgimento dell'utente e gamification**: Per rendere l'XAI accessibile anche a utenti non tecnici, sono stati proposti approcci che integrano elementi di gamification e interazioni narrative. [26] presenta un framework che utilizza gli LLM per creare esperienze interattive che facilitano l'esplorazione e la comprensione delle visualizzazioni AI da parte degli utenti. Inoltre, [18] sfida l'assunto tradizionale di "aprire la black-box" nell'era degli LLM, proponendo una prospettiva centrata sull'utente che si concentra sull'explainability al di fuori e attorno alla black-box.

Nonostante questi progressi, permangono sfide significative, come la necessità di sviluppare metriche standardizzate per valutare la qualità delle spiegazioni [27], bilanciare l'interpretabilità con le prestazioni del modello e adattare le tecniche XAI per applicazioni specifiche ad alto rischio [28]. Il futuro dell'XAI nell'era degli LLM richiede un'attenzione particolare alla trasparenza, all'affidabilità e all'interazione uomo-macchina [18]. L'adozione di approcci centrati sull'utente e la continua collaborazione tra modelli di apprendimento automatico e LLM saranno fondamentali per avanzare in questo campo in rapida evoluzione.

## 2.4 Gli strumenti scelti per gli esperimenti

Per affrontare il problema della spiegabilità negli LLM, utilizzeremo alcune librerie di XAI disponibli pubblicamente.

In particolare in questa tesi sperimentiamo due tipi di strumenti: `LIME` e `Captum`.

### 2.4.1 LIME

LIME (Local Interpretable Model-agnostic Explanations)[1] è una delle prime librerie sviluppate per creare spiegazioni interpretabili di modelli di intelligenza artificiale ed è ampiamente utilizzata nell'ambito della Explainable AI (XAI). Essendo uno strumento model-agnostic, LIME può essere applicato a modelli di qualsiasi tipo senza necessità di conoscerne la struttura interna. Trattando il modello come una *black box* e analizzando come piccoli cambiamenti negli input influenzano gli output, è infatti possibile fornire spiegazioni utili senza accesso alla struttura interna.

Per generare spiegazioni, LIME costruisce un modello interpretabile locale che approssima il comportamento del modello complesso solo nell'intorno di uno specifico input di interesse. Questa interpretabilità locale consente a LIME di spiegare un singolo output del modello complesso, rendendo chiaro il comportamento della predizione per quell'input specifico senza tentare di rappresentare l'intero modello.

Il processo si svolge attraverso la perturbazione dell'input originale, creando variazioni simili per osservare come il modello complesso risponde a ciascuna di esse. Per approssimare il comportamento locale, LIME minimizza una funzione obiettivo definita come:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

dove:

[1] LIME GitHub repository: `https://github.com/marcotcr/lime`

- $\xi(x)$ è il modello interpretabile locale scelto per spiegare la predizione $f(x)$.
- $\mathcal{L}(f, g, \pi_x)$ è una funzione di perdita che misura quanto bene il modello interpretabile $g$ approssima il modello complesso $f$ nell'intorno dell'input $x$.
- $\Omega(g)$ è un termine di regolarizzazione che penalizza la complessità del modello interpretabile $g$.
- $G$ rappresenta lo spazio dei modelli interpretabili.

La funzione di perdita $\mathcal{L}(f, g, \pi_x)$ è definita come:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in Z} \pi_x(z)\big(f(z) - g(z')\big)^2$$

dove:

- $\pi_x(z)$ assegna un peso maggiore ai campioni vicini all'input $x$, ad esempio in base a una distanza $D$, come la distanza coseno per il testo o la distanza L2 per le immagini.
- $f(z)$ rappresenta la predizione del modello complesso per l'input perturbato $z$.
- $g(z')$ rappresenta la predizione del modello interpretabile per lo stesso input perturbato $z'$.
- La somma viene calcolata sull'insieme di campioni perturbati $Z$.

Oltre alla costruzione del modello locale, LIME affronta il problema della selezione ottimale delle feature da includere nell'esplanazione. Questo viene formalizzato come un **Pick Problem**, in cui viene selezionato un sottoinsieme di feature $V$ tale da massimizzare la copertura dei dati, garantendo allo stesso tempo che il sottoinsieme mantenga una dimensione limitata per preservare l'interpretabilità.

La funzione obiettivo è definita come:

$$c(V, \mathcal{W}, \mathcal{I}) = \sum_{j=1}^{d'} \mathbf{1}\left[\exists i \in V : w_{ij} > 0\right] \mathcal{I}_j$$

dove:

- $V$ è il sottoinsieme di feature selezionate.
- $\mathcal{W}$ è la matrice dei pesi, dove $w_{ij} > 0$ indica che la feature $i \in V$ copre l'istanza $j$.
- $\mathcal{I}_j$ è un indicatore di importanza per l'istanza $j$.
- La funzione $\mathbf{1}[\cdot]$ è un indicatore che vale 1 se la condizione all'interno è vera.

Il problema di ottimizzazione è formulato come segue:

$$\text{Pick}(\mathcal{W}, \mathcal{I}) = \arg \max_{V, |V| \leq B} c(V, \mathcal{W}, \mathcal{I})$$

dove $|V| \leq B$ impone che il numero di feature selezionate sia limitato a una soglia $B$, evitando così spiegazioni eccessivamente complesse.

Questo approccio bilancia la copertura dei dati con la semplicità delle spiegazioni, assicurando che LIME produca risultati interpretabili e utili. In sintesi, LIME costruisce un modello interpretabile locale, come una regressione lineare o un albero decisionale, che approssima il comportamento del modello complesso nell'intorno dell'input, rendendo chiaro il comportamento della predizione per quell'input specifico.

### 2.4.2 Captum

**Captum** [2] è una libreria sviluppata da Meta AI per l'interpretabilità di modelli basati su **PyTorch**. Il suo scopo è fornire un ambiente unificato per applicare una vasta gamma di metodi interpretativi. Con l'aggiornamento alla versione 0.7, Captum ha introdotto nuove funzionalità per l'analisi dei **Large Language Models (LLM)**[29]. Questa versione permette di applicare l'**analisi di attribuzione** su modelli di linguaggio, consentendo di esplorare l'influenza di singoli token o segmenti di testo sul contenuto generato e migliorando la comprensione delle dinamiche interne dei modelli. Le nuove funzionalità includono algoritmi basati su **perturbazione** e **gradiente** per interpretare l'input testuale e la gestione di predizioni sequenziali. In particolare, Captum offre due metodi di attribuzione per LLM:

#### Metodi basati sulla perturbazione

Captum implementa metodi come **Feature Ablation** e **Shapley Values**. Questi metodi permettono di *rimuovere* token o segmenti di testo dal prompt [3] per studiarne l'impatto sul testo generato. Tale approccio fornisce una visione dettagliata dell'influenza di ogni parola o segmento sull'output del modello.

**Feature Ablation** Sostituisce ogni caratteristica dell'input con un valore di riferimento (baseline) e misura l'impatto sul risultato del modello. La formula è definita come segue:

$$\phi_i(f, X) = f(X) - f(X_{D\setminus\{i\}})$$

[2]Captum GitHub repository: `https://github.com/pytorch/captum`

[3]Nel contesto dei modelli LLM, per "prompt" si intende il testo fornito in input al modello, utilizzato per generare risposte o completamenti. Spesso viene preferito il termine "prompt" anziché "frase/testo di input" per specificare meglio il ruolo svolto nel processo di interazione con l'LLM

Dove:

- $f(X)$ è il risultato del modello per l'input completo.
- $f(X_{D\setminus\{i\}})$ è il risultato del modello quando la caratteristica $i$ viene rimossa o sostituita.

**Shapley Values** Derivato dalla teoria dei giochi cooperativi, assegna punteggi di attribuzione a ogni caratteristica, calcolandone il contributo marginale al risultato. La formula è:

$$\phi_i(f, X) = \sum_{S \subseteq D\setminus\{i\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!} \big(f(X_{S\cup\{i\}}) - f(X_S)\big)$$

Dove:

- $S$ è un sottoinsieme delle caratteristiche senza $i$.
- $f(X_{S\cup\{i\}})$ è il risultato del modello includendo la caratteristica $i$ nel sottoinsieme $S$.

### Metodi basati sul gradiente

Tra i metodi basati sul gradiente, Captum include **Saliency Maps** e **Integrated Gradients**. Questi metodi utilizzano la backpropagation per calcolare l'importanza delle caratteristiche in base alla sensibilità del modello.

**Saliency Maps** Il metodo Saliency si basa sui gradienti del modello calcolati nel punto di input, utilizzando questi gradienti come attribuzioni delle caratteristiche corrispondenti. Questo approccio può essere visto come un'approssimazione di primo ordine della funzione.

$$\phi_i(f, X) = f'(X)$$

**Integrated Gradients** Stima l'importanza di ogni caratteristica calcolando l'integrale dei gradienti del modello lungo un percorso tra il punto di riferimento (baseline) e l'input attuale. La formula è:

$$\phi_i(f, X) = (X_i - B_i) \times \int_{\alpha=0}^{1} \frac{\partial f(B + \alpha(X - B))}{\partial X_i} d\alpha$$

Dove:

- $X$ è l'input corrente.
- $B$ è il punto di riferimento o baseline.
- $\frac{\partial f}{\partial X_i}$ è il gradiente del modello rispetto alla caratteristica $i$.

### Visualizzazioni e Applicazioni

Captum utilizza queste formule per produrre visualizzazioni come heatmap e grafici di attribuzione per i token, evidenziando l'importanza di specifici segmenti di testo rispetto all'output generato. Questo è particolarmente utile per analizzare il comportamento dei modelli LLM, consentendo agli utenti di comprendere meglio le associazioni apprese dal modello e l'impatto delle diverse caratteristiche del prompt testuale.

## 2.5 Altri Strumenti e Tecniche di XAI

Per questa tesi abbiamo scelto di utilizzare LIME e Captum, ma esistono diversi altri strumenti di XAI che potrebbero essere applicati in future espansioni di questa ricerca. Di seguito abbiamo inserito una breve e sicuramente non esaustiva panoramica rispetto a altri strumenti di XAI sviluppati:

### Strumenti Sviluppati da Grandi Aziende

1. **AI Explainability 360 (AIX360)** [30]
    - **Descrizione**: Toolkit open-source sviluppato da IBM che offre una gamma completa di algoritmi per l'interpretabilità e la spiegazione dei modelli di machine learning.
    - **Vantaggi**: Fornisce tecniche per spiegare sia modelli "black-box" che "glass-box", facilitando la comprensione del processo decisionale dei modelli.
2. **What-If Tool (WIT)** [31]
    - **Descrizione**: Strumento open-source sviluppato da Google che consente agli utenti di analizzare modelli di machine learning attraverso scenari ipotetici e visualizzazioni interattive.
    - **Vantaggi**: Permette di testare le prestazioni in situazioni ipotetiche e valutare metriche di equità, il tutto senza necessità di codifica estensiva.
3. **OmniXAI** [32]
    - **Descrizione**: Libreria open-source proposta dai ricercatori di Salesforce che integra diverse tecniche di XAI in un'unica interfaccia.
    - **Vantaggi**: Offre una piattaforma semplice e unificata per generare spiegazioni e visualizzazioni per modelli basati su diversi tipi di dati.

## Strumenti Sviluppati da Organizzazioni Indipendenti o Comunità Open-Source

1. **SHAP (SHapley Additive exPlanations)** [16]

   - **Descrizione**: Utilizza i valori di Shapley dalla teoria dei giochi per assegnare un'importanza a ogni caratteristica. Fornisce spiegazioni sia globali che locali del modello.
   - **Vantaggi**: Supporta una varietà di modelli (alberi decisionali, modelli lineari, deep learning) e offre spiegazioni robuste e interpretabili.

2. **Anchors** [33]

   - **Descrizione**: Sviluppato dallo stesso autore di LIME, è basato su "ancore" (regole locali) per spiegazioni interpretabili del modello. Funziona bene con dati strutturati e modelli NLP.
   - **Vantaggi**: Focalizzato sull'accuratezza delle spiegazioni locali, particolarmente utile nell'elaborazione del linguaggio naturale.

3. **Alibi** [34]

   - **Descrizione**: Toolkit per l'XAI che include spiegazioni controfattuali, ancore e rilevamento di anomalie.
   - **Vantaggi**: Ampia gamma di tecniche per vari modelli, adatto sia per spiegazioni locali che controfattuali.

4. **TCAV (Testing with Concept Activation Vectors)** [35]

   - **Descrizione**: Sviluppato ufficialmente da Google per Tensorflow è una tecnica per comprendere come concetti specifici influenzino le decisioni del modello.
   - **Vantaggi**: Consente un'analisi basata su concetti di alto livello, particolarmente utile nell'analisi di immagini.

5. **iNNvestigate** [36]

   - **Descrizione**: Focalizzato su modelli di deep learning, offre metodi di spiegazione come LRP (Layer-wise Relevance Propagation) e mappe di salienza basate sul gradiente.
   - **Vantaggi**: Ideale per reti neurali profonde, supportando varie tecniche basate sul gradiente e LRP.

# Capitolo 3

# I nostri esperimenti e analisi sugli LLM

In questa tesi abbiamo progettato alcuni test per analizzare i risultati degli LLM e individuare gli strumenti di XAI più appropriati da adottare.

## 3.1 I modelli di LLM scelti

Per i nostri esperimenti abbiamo scelto di utilizzare modelli di dimensioni ridotte (Come mostrato in tabella 3.1), per la necessità di facilitare il testing e gestire le attuali limitazioni delle risorse GPU a nostra disposizione. L'uso di modelli più leggeri ci ha permesso inoltre di eseguire e sperimentare inizialmente su una serie di test più diversificata e significativa.

| **Nome** | **Dimensione** | **Anno di rilascio** | **Azienda** |
|---|---|---|---|
| SmolLM2-1.7B-Instruct [37] | 1.7B | 2024 | HuggingFace |
| gpt2-xl [38] | 1.5B | 2019 | OpenAI |
| bloomz-560m [39] | 560M | 2022 | BigScience |

Tabella 3.1: Modelli LLM scelti per i nostri esperimenti

## 3.2 L'hardware adottato

Per condurre questi esperimenti, abbiamo usufruito dei servizi gratuiti e non offerti da Google tramite Colab. Tramite questo servizio infatti soprattutto nei periodi iniziali abbiamo avuto l'accesso ad un ambiente pre configurato, facile da utilizzare e predisposto per l'uso di una GPU per avvelocizzare le operazioni di testing. In seguito abbiamo potuto aggiungere alle nostre risorse a disposizione anche le GPU offerte dal cluster HPC (High Performance Computing) del nostro dipartimento.

In particolare per quanto riguarda la GPU, in questa sezione vengono presentate le specifiche principali di quelle utilizzate per i nostri test: la Nvidia Tesla T4, L4 e L40 (Tabella 3.2).

## 3.3 Le nostre configurazione dei tool XAI

Per utilizzare LIME su un task di classificazione complesso come quello degli LLM abbiamo adottato una semplificazione.

In particolare, abbiamo configurato il tool per operare su un problema binario tramite due classi:

- **Token Atteso**: Il token che ci aspettiamo come corretto dal risultato dell'LLM.
- **Token Non Atteso**: Qualsiasi token che non corrisponde a quello previsto.

La trasformazione delle predizioni del modello viene eseguita utilizzando la formula:

$$\mathbf{P} = \begin{bmatrix} p \\ 1-p \end{bmatrix}$$

Dove:

- $p$ **(Predizione)**: Indica la probabilità che il modello assegna al token atteso. Questa probabilità riflette la fiducia del modello nella selezione del token specifico rispetto ad altri possibili token.
- $1-p$ **(Probabilità Complementare)**: Rappresenta la probabilità aggregata di tutti gli altri token non previsti. Essa aggrega le probabilità di tutti i token alternativi.

In questo modo le spiegazioni risultano semplici da interpretare, poiché si focalizzano su una decisione binaria chiara. Futuri sviluppi potrebbero esplorare configurazioni più complesse per arricchire ulteriormente la comprensione degli LLM sperimentando con LIME.

## 3.4 Esperimenti

Per esaminare il funzionamento degli strumenti di XAI attualmente disponibili, abbiamo progettato una serie di esperimenti. L'obiettivo è valutare quali strumenti forniscano risultati più comprensibili e quali, al contempo, producano risultati più "accurati", evidenziando inoltre gli eventuali limiti attuali di queste tecnologie. In questa tesi presenteremo e discuteremo i risultati più significativi ottenuti dagli esperimenti condotti.

## Nome delle capitali

In questo esperimento abbiamo focalizzato la nostra attenzione sul cercare di capire grazie agli strumenti di XAI scelti quali parti del prompt di input siano considerate più importanti nella predizione dell'output del modello.

### Dati utilizzati

Per il prompt, abbiamo scelto un ambito specifico: riconoscere il nome della capitale di uno stato europeo. Questa scelta mira a semplificare la richiesta, concentrandosi principalmente sulla capacità dei modelli di richiamare "fatti" e "conoscenze comuni" apprese durante la fase di training, senza dover quindi mettere alla prova o cercare di comprendere anche le loro capacità di ragionamento.

Il prompt di base utilizzato è stato:

```
"The name of the capital of France is"
```

Il token atteso è stato:

```
" Paris"
```

### Ipotesi

Per questo esperimento avanziamo alcune ipotesi per cui:

1. I token `"France"` e `"capital"` siano i più importanti per la predizione, poiché specificano il paese e l'entità da identificare, descrivendo meglio i requisiti del prompt.

2. I modelli di maggiori dimensioni potrebbero considerare importanti invece parti grammaticali della frase, come `"is"`, per indicare di dover continuare a predire una frase di senso compiuto.

### Metodo

Con questo primo esperimento proveremo a testare gli strumenti di XAI che abbiamo scelto per questi esperimenti, applicandoli ai vari modello per generare la predizioni e analizzando quindi i risultati ottenuti. In particolare applicheremo *LIME* e *Captum* nelle varianti Perturbation Based (Feature Ablation) e Gradient Based (Integrated Gradients).

### Riflessioni

I risultati di questo esperimento iniziale consentiranno di valutare l'efficacia degli strumenti di XAI in questo contesto specifico e di verificare se le nostre ipotesi sono supportate dai dati raccolti. Inoltre ci permetterà di iniziare ad identifichare le principali sfide nell'applicazione dell'XAI sugli LLM, come la gestione della complessità delle interazioni tra i token e l'interpretazione delle importanze assegnate.

Ad esempio, qualora i risultati evidenziassero una maggiore attenzione su `"capital"` e `"France"`, ciò confermerebbe la nostra prima ipotesi riguardo alla rilevanza semantica di questi token. Viceversa, se l'attenzione si concentrasse significativamente anche su elementi grammaticali, ciò sosterrebbe la nostra seconda ipotesi, indicando una maggiore influenza della struttura sintattica nel processo di predizione.

Tabella 3.2: Specifiche tecniche Tesla T4, L4, e L40

| Specification | Tesla T4 | L4 | L40 |
|---|---|---|---|
| **Graphics Processor** | TU104 | AD104 | AD102 |
| **Architecture** | Turing | Ada Lovelace | Ada Lovelace |
| **Process Size** | 12 nm | 5 nm | 5 nm |
| **Transistors** | 13.60 billion | 35.80 billion | 76.30 billion |
| **Cores (CUDA)** | 2560 | 7424 | 18176 |
| **TMUs** | 160 | 240 | 568 |
| **ROPs** | 64 | 80 | 192 |
| **Memory Size** | 16 GB | 24 GB | 48 GB |
| **Memory Type** | GDDR6 | GDDR6 | GDDR6 |
| **Memory Bus** | 256-bit | 192-bit | 384-bit |
| **Memory Bandwidth** | 320.0 GB/s | 300.1 GB/s | 864.0 GB/s |
| **Base Clock** | 585 MHz | 795 MHz | 735 MHz |
| **Boost Clock** | 1590 MHz | 2040 MHz | 2490 MHz |
| **TDP** | 70 W | 72 W | 300 W |
| **Tensor Cores** | 320 | 240 | 568 |
| **RT Cores** | 40 | 60 | 142 |
| **FP16 Performance** | 65.13 TFLOPS | 30.29 TFLOPS[†] | 90.52 TFLOPS[†] |
| **FP32 Performance** | 8.141 TFLOPS | 30.29 TFLOPS | 90.52 TFLOPS |
| **FP64 Performance** | 254.4 GFLOPS | 473.3 GFLOPS | 1.414 TFLOPS |
| **L2 Cache** | 4 MB | 48 MB | 96 MB |
| **Release Date** | September 13, 2018 | March 21, 2023 | October 13, 2022 |

[†]Le performance FP16 indicate per L4 e L40 probabilmente si riferiscono al throughput basato su core CUDA, non includendo l'accelerazione aggiuntiva fornita dai Tensor Core.

# Capitolo 4

# I risultati dei nostri esperimenti

## 4.1 Nome delle capitali

Per iniziare l'esperimento abbiamo valutato il livello di confidenza dei vari modelli scelti rispetto alla previsione del token `" Paris"` ottenendo (Tabella 4.1):

| Modello | Confidenza |
|---|---|
| **bloomz-560m** | 83.94% |
| **gpt2-xl** | 31.59% |
| **SmolLM2-1.7B-Instruct** | 40.80% |

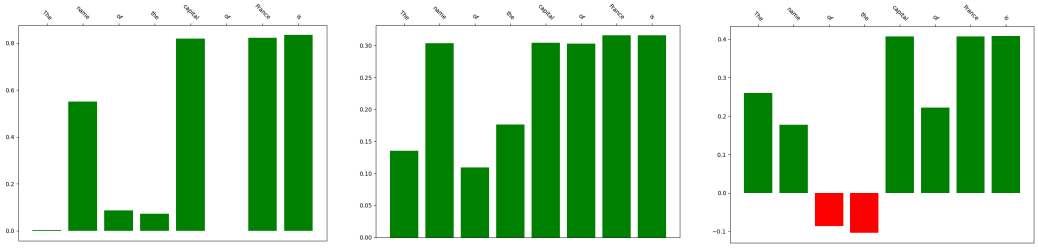
Tabella 4.1: Confidenza nella previsione del token `" Paris"`

Per avere una base di riferimento su cui poi poter confrontare i risultati dei vari strumenti di XAI abbiamo predisposto un primo esperimento provando a creare un grafico di base senza l'uso di strumenti particolari confrontando solamente la differenza di confidenza del modello nel prevedere il token atteso rimuovendo parti del prompt iniziale.

Così abbiamo ottenuto questi risultati :

- **bloomz-560m** (Tabella 4.2)
- **SmolLM2-1.7B-Instruct** (Tabella 4.3)
- **gpt2-xl** (Tabella 4.4)

Visualizzando questi risultati in un grafico a barre otteniamo (4.1):



(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.1: Risultati test Baseline

| Prompt | Parola Rimossa | Confidenza | Perdita di Confidenza |
|---|---|---|---|
| The name of the capital of France is | Nessuna | 0.8393 | - |
| name of the capital of France is | The | 0.8364 | 0.0029 |
| The of the capital of France is | name | 0.2890 | 0.5502 |
| The name the capital of France is | of | 0.7534 | 0.0859 |
| The name of capital of France is | the | 0.7670 | 0.0722 |
| The name of the of France is | capital | 0.0202 | 0.8190 |
| The name of the capital France is | of | 0.8408 | -0.0014 |
| The name of the capital of is | France | 0.0160 | 0.8233 |
| The name of the capital of France | is | 0.0047 | 0.8346 |

Tabella 4.2: **bloomz-560m** con parti del prompt rimosse

| Prompt | Parola Rimossa | Confidenza | Perdita di Confidenza |
|---|---|---|---|
| The name of the capital of France is | Nessuna | 0.4079 | - |
| name of the capital of France is | The | 0.1478 | 0.2601 |
| The of the capital of France is | name | 0.2310 | 0.1768 |
| The name the capital of France is | of | 0.4946 | -0.0866 |
| The name of capital of France is | the | 0.5122 | -0.1042 |
| The name of the of France is | capital | 0.0010 | 0.4069 |
| The name of the capital France is | of | 0.1860 | 0.2219 |
| The name of the capital of is | France | 0.0007 | 0.4071 |
| The name of the capital of France | is | 2.5212e-05 | 0.4079 |

Tabella 4.3: **SmolLM2-1.7B-Instruct** con parti del prompt rimosse

| Prompt | Parola Rimossa | Confidenza | Perdita di Confidenza |
|---|---|---|---|
| The name of the capital of France is | Nessuna | 0.3159 | - |
| name of the capital of France is | The | 0.1805 | 0.1353 |
| The of the capital of France is | name | 0.0122 | 0.3036 |
| The name the capital of France is | of | 0.2067 | 0.1091 |
| The name of capital of France is | the | 0.1394 | 0.1765 |
| The name of the of France is | capital | 0.0116 | 0.3042 |
| The name of the capital France is | of | 0.0132 | 0.3026 |
| The name of the capital of is | France | 0.0001 | 0.3157 |
| The name of the capital of France | is | 2.1934E-05 | 0.3158 |

Tabella 4.4: **gpt2-xl** con parti del prompt rimosse

Da questo primo testi di base possiamo già iniziare a cercare di avere una prima prospettiva rispetto ai token ritenuti più importanti dai vari modelli.

- **bloomz-560m**: Considera come parti fondamentali `"capital"` e `"France"` indicando che il modello comprende in modo efficace il contesto del prompt. Cosa che si riflette effettivamente sul livello alto di confidenza del modello nel predirre `" Paris"`. Inoltre considera importante anche il token finale `"is"` probabilmente per la grammatica della frase e `"name"` per avere maggiore certezza sul come la frase dovrebbe proseguire.

- **gpt2-xl**: Mostra una distribuzione più uniforme dell'importanza tra le parole, con una concentrazione minore su `"capital"` e `"France"` rispetto a **bloomz-560m** e **SmolLM2-1.7B-Instruct**. Ciò potrebbe essere dovuto dal fatto di essere uno dei modelli più vecchi. Predisponendolo forse per una minore capacità di generalizzare i concetti rispetto a modelli anche più piccoli tipo **bloomz-560m** ma più moderni. Questa incertezza nel distribuire in modo più o meno equilibrato l'importanza ai token in input effettivamente trova anche un riscontro nel livello di confidenza del modello, pari solamente al 31.59%, ovvero la più bassa tra tutti e 3.

- **SmolLM2-1.7B-Instruct**: I suoi risultati sono simili a quelli di **bloomz-560m**, mostrando un'importanza dei token abbastanza focalizzata su concetti più importanti quali `"capital"`, `"France"` e `"is"`. Notiamo anche che in modo particolare `"of"` e `"the"` calano stranamente la probabilità di avere il token atteso.

Tornando però al punto di partenza e rispetto alle nostre ipotesi, se consideriamo la rimozione dei token quali `"capital"` e `"France"` otteniamo effettivamente (Come da grafico 4.2 e tabella 4.5):

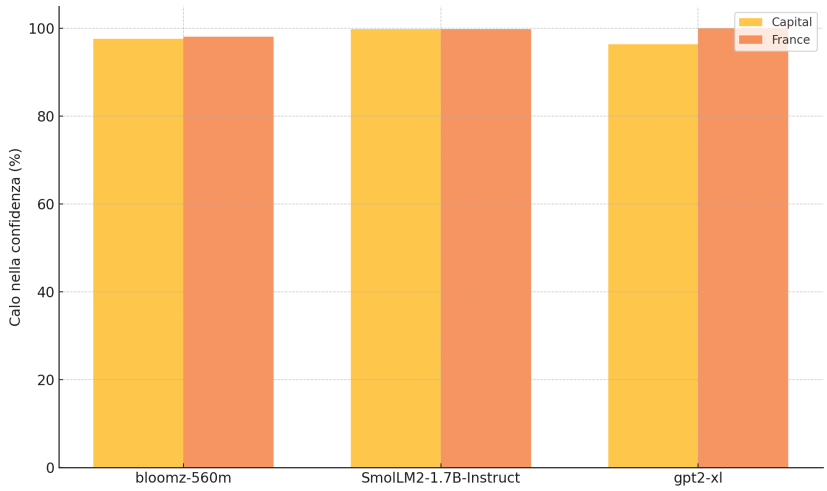



Figura 4.2: Percentuale di calo nella confidenza per i token `"capital"` e `"France"`.

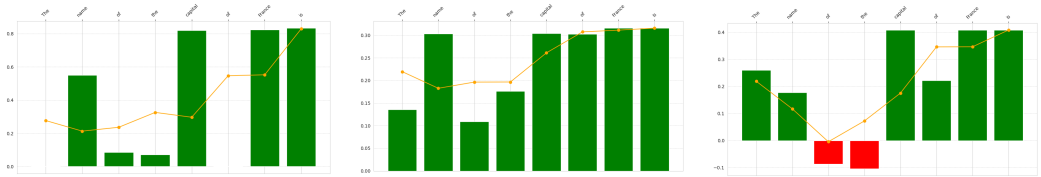
| Modello | Capital (% di calo) | France (% di calo) |
|---|---|---|
| **bloomz-560m** | 97.59% | 98.09% |
| **SmolLM2-1.7B-Instruct** | 99.74% | 99.81% |
| **gpt2-xl** | 96.32% | 99.94% |

Tabella 4.5: Percentuale di calo nella confidenza rimuovendo i token `"capital"` e `"France"`.

Questi primi risultati sono abbastanza incoraggianti e ci fanno capire che già un primo semplice test può dare risultati interessanti. Infatti nonostante le parole `"capital"` e `"France"` siano considerate importanti da tutti i modelli, presupponiamo che essendo la frase corta e semplice anche la maggior parte delle altre parole sembri essere considerata come molto importante per la predizione. Nonostante ciò notiamo che i modelli che si focalizzano di più sulle parole `"name"`, `"capital"` e `"France"` sono quelli anche con una confidenza maggiore nel predirre correttamente il token `" Paris"`. Ciò ci da fiducia che anche partendo da un primo test semplice di XAI, sia possibile ottenere risultati promettenti.

## Analisi dei trend

Per analizzare i cambiamenti di confidenza associati ai token rimossi, inoltre abbiamo anche provato a visualizzare i trend relativi al cambio di confidenza. A tal fine, è stata applicata una media mobile centrata con una finestra di 3, garantendo che il valore medio di un token consideri in modo equilibrato il token stesso, quello precedente e quello successivo. Questo metodo permette di smussare le fluttuazioni locali ed evidenziare un trend globale più rappresentativo. Abbiamo quindi ottenuto i seguenti risultati (Grafico 4.3).



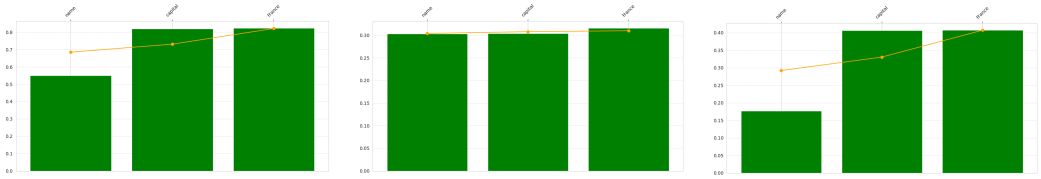
(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.3: Confronto Trend sul test Baseline

Dai risultati ottenuti emerge che, oltre a confermare parzialmente le ipotesi formulate in precedenza, tutti e tre i modelli sembrano seguire un trend comune: i token tendono ad aumentare di importanza man mano che ci si avvicina alla fine del prompt. Ipotizziamo che questo potrebbe essere dovuto al loro contributo progressivo al significato complessivo della frase.

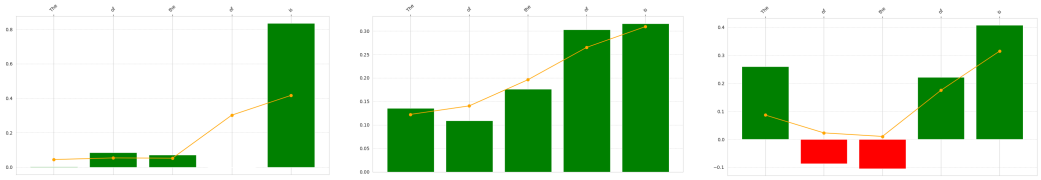
Per approfondire questa osservazione, i dati sono stati suddivisi in due set distinti: uno contenente i token più rilevanti, quali `"name"`, `"capital"` e `"France"`, e un altro comprendente gli altri token. Analizzando separatamente questi due set, la tendenza all'aumento dell'importanza dei token, se non per il modello `SmolLM2-1.7B-Instruct`, diventa ancora più evidente. I grafici 4.4 e 4.5 illustrano in modo ancora più evidente questi trend in aumento, rappresentando l'importanza dei token lungo il prompt, procedendo da sinistra a destra verso la fine.

Proviamo ora ad usare anche gli strumenti di XAI a nostra disposizione.



(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.4: Confronto Trend sul test Baseline per le parti più importanti

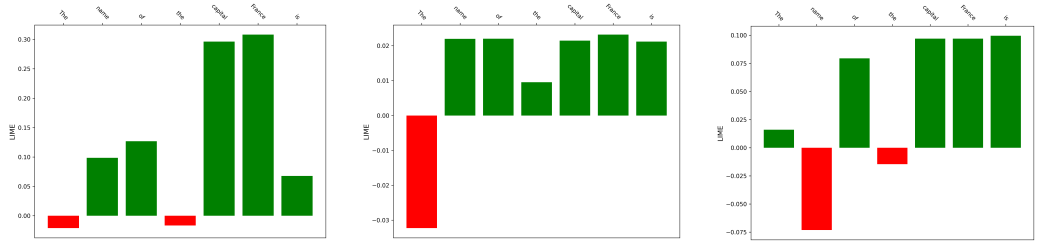


(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.5: Confronto Trend sul test Baseline per le parti meno importanti

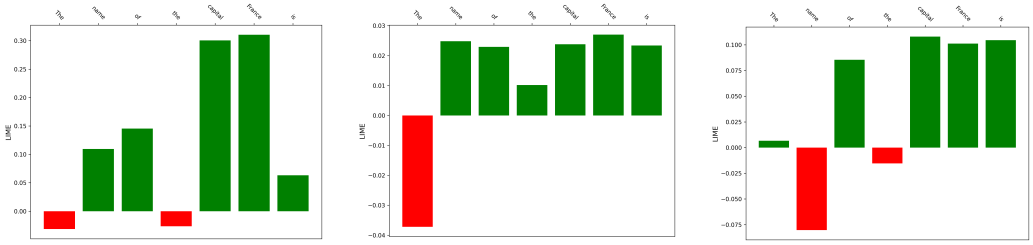
### 4.1.1 LIME

Partendo con LIME configurato con 1000 sample otteniamo il seguente grafico 4.6. Per testare meglio le possibilità offerte da questo tool abbiamo provato anche ad aumentare il numero di sample utilizzati per l'esperimento (Grafico 4.7) a 5000 (La configurazione di default di LIME). Data la poca differenza di risultati però abbiamo ritenuto sufficiente mantenere una configurazione di 1000 sample.



(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.6: LIME con 1000 sample



(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.7: LIME con 5000 sample

A primo impatto sembrerebbe che il modello con i risultati più comprensibili, riconfermando i risultati del test base, sia anche quello di dimensioni più ridotte, **bloomz-560m**, valorizzando maggiormente le parole "capital" e "France" per ottenere il risultato finale. Anche il modello **SmolLM2-1.7B-Instruct** sembra aver dato risultati promettenti, mentre **gpt2-xl** ha riportato un bilanciamento più generale dei risultati e del valore dei singoli token in input.

In particolare, rianalizzando i risultati abbiamo che:

- **bloomz-560m**: Assegna una rilevanza molto alta ai token `"capital"` e `"France"`, confermando i risultati ottenuti precedentemente dal test Baseline.

- **gpt2-xl**: Mostra una distribuzione più uniforme dell'importanza tra i token, con una concentrazione minore su `"capital"` e `"France"` rispetto a **bloomz-560m**. Inoltre, sembra dare un valore negativo all'inizio della frase per la parola `"The"`.

- **SmolLM2-1.7B-Instruct**: I suoi risultati sono simili a quelli di **gpt2-xl**, mostrando un'importanza dei token abbastanza distribuita e non specialmente concentrata sui token più importanti quali `"capital"` e `"France"`, ma comunque più specifica rispetto a gpt2-xl. Anche in questo caso però sembra che alcuni token (`"name"` e `"the"`) siano considerati stranamente negati.

Questi primi risultati di LIME sono abbastanza allineati se non completamente con i risutlati mostrati dal primo test di base almeno per le parti principali.

Ricontrollando dai valori del test base, possiamo inoltre verificare i risultati ottenuti con LIME. Provando infatti a rimuovere alcuni token dal prompt di base otteniamo:

**Rimuovendo i token più importanti.** Provando a questo punto a rimuovere i token considerti più importanti da LIME otteniamo:

- **bloomz-560m**: Rimuovendo "France" - 1.60%
- **gpt2-xl**: Rimuovendo "France" - 0.02%
- **SmolLM2-1.7B-Instruct**: Rimuovendo "capital" - 0.10%

**Rimuovendo i secondi token più importanti** Provando a rimuovere invece i secondi token considerati più importanti da LIME otteniamo:

- **bloomz-560m**: Rimuovendo "capital" - 2.03%
- **gpt2-xl**: Rimuovendo "name" - 1.22%
- **SmolLM2-1.7B-Instruct**: Rimuovendo "is" - 0.00%

**Provando a rimuovere i token che influiscono negativamente** Provando a rimuovere invece i token che secondo i risultati di LIME sembrano influire negativamente sul risultato otteniamo:

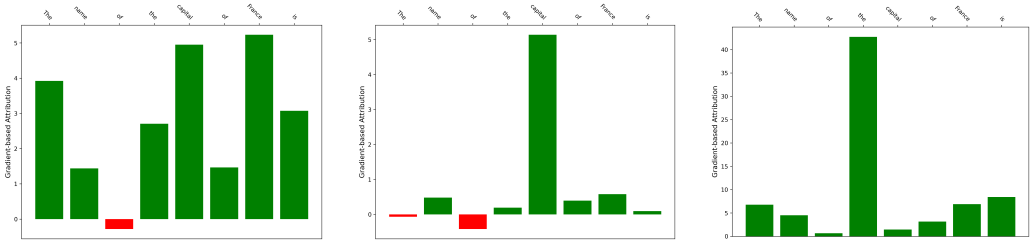
- **bloomz-560m**: Rimuovendo "The" - 83.64%
- **gpt2-xl**: Rimuovendo "The" - 18.05%
- **SmolLM2-1.7B-Instruct**: Rimuovendo "name" - 23.11%

Se i risultati per i token più importanti sembrano dare risultati comprensibili, quelli rimuovendo i token che sembravano meno importanti risultano invece inaspettati. Infatti rimuovendo questi token considerati peggiorativi da LIME otteniamo invece un calo della confidenza del modello nel predirre il token " Paris". Mostrando già i primi segnali dei limiti della configurazione usata in questo test per l'explainability degli LLM. Un fattore che potrebbe avere influenzato questo tipo di risultato supponiamo potrebbe essere dovuto da come il problema di classificazione è stato configurato inizialmente. Potrebbe essere che la formula semplice usata per calcolare le due classi da predirre con LIME non sia sufficientemente adeguata per analizzare anche quali siano i token che rimossi influenzano positivamente la confidenza del modello. In questo caso infatti il test baseline semplice sembra aver dato risultati più significativi.

### 4.1.2 Captum

Abbiamo provato quindi ad utilizzare Captum per l'interpretabilità di questo caso specifico e abbiamo ottenuto i seguenti due risultati.

Partendo con un test **Gradient Based** (Figura 4.8):



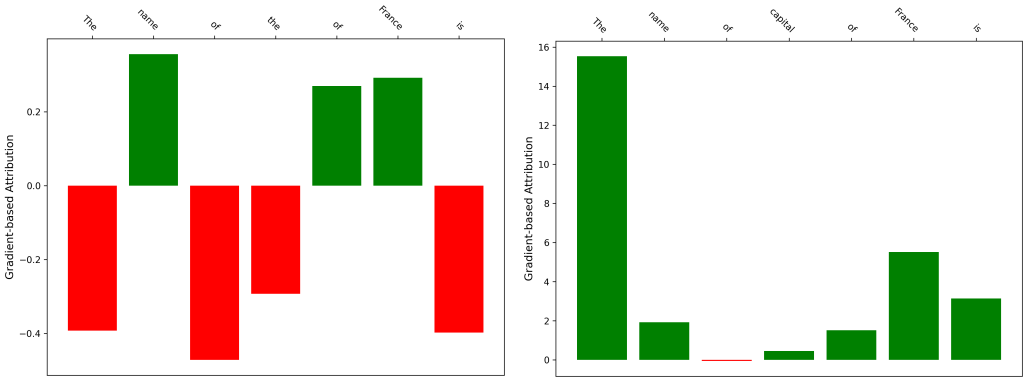

(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.8: Risultati test Gradient Based

Da questi risultati sembra che:

- **bloomz-560m**: Assegni abbastza correttemente un importanza molto alta ai token `"capital"` e `"France"` come ci aspettavamo e anche a `"is"`. Anche se rispetto al test base sembra considerare come abbastanza importanti anche i due `"the"` presenti, rendendo l'importanza delle parti principali meno evidente.
- **gpt2-xl**: Sembra in questo caso dare risultati non molto attendibili, dando giustamente maggiore rilevana a `"capital"` come parte fondamentale, ma sicuramente sovrastimandone l'importanza rispetto ad altri token importanti della frase, confrontando questo risultato con il nostro test di base.
- **SmolLM2-1.7B-Instruct**: Anche in questo caso il test bassato sul gradiente mostra un risultato abbastanza non attendibile rilevando il secondo `"the"` come token molto più importante rispetto a tutti gli altri.

Abbiamo a questo punto provato a rimuovere questi due outlier per gli utlimi due modelli testati e abbiamo ottenuto (Come dal grafico 4.9):




(a) gpt2-xl (b) SmolLM2-1.7B-Instruct

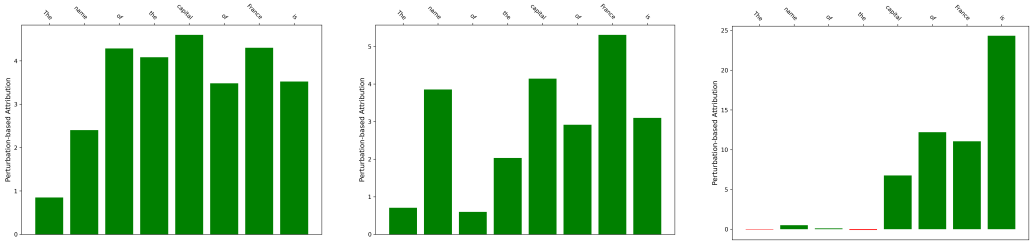
Figura 4.9: Risultati test Gradient Based aggiornato

Con una probabilità per il token `" Paris"` dello:

- **gpt2-xl**: 1.15%
- **SmolLM2-1.7B-Instruct**: 49.54%

Anche in questo caso abbiamo ottenuto risultati contrastanti. Da un lato la spiegazione della tecninca Gradient Based ha funzionato per gpt2-xl riducendo la fiducia del modello nell'output `" Paris"` dal 31.59% al 1.15%. D'altro canto il modello SmolLM2-1.7B-Instruct ha incrementato inaspettatamente la sua fiducia nel suo output dal 40.80% al 49.54%.

Provando ora con una soluzione **Perturbation Based** (Più simile a LIME) otteniamo i risultati come mostrato nel grafico 4.10:

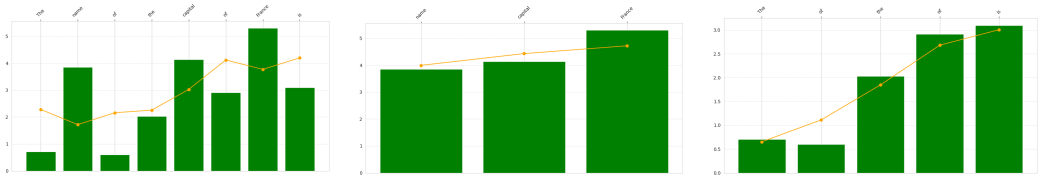



(a) bloomz-560m (b) gpt2-xl (c) SmolLM2-1.7B-Instruct

Figura 4.10: Risultati test Perturbation Based

Da questi test emerge una tendenza a avvicinarsi di più come anche LIME al risultato del test baseline. Questo probabilmente anche dato dal fatto di basarsi sulla perturbazione del prompt iniziale come dal test base.

Infatti inoltre se analizziamo nello specifico il risultato di gpt2-xl, dove questo aspetto è più visibile, come dal grafico 4.11 possiamo ritrovare lo stesso tipo di trend di importanza dei token analizzato in precedenza.



(a) Completo (b) Parti più importanti (c) Parti meno importanti

Figura 4.11: Visualizzazione trend per i risultati di Captum per gpt2-xl

Questo suggerisce che le tecniche basate sulla perturbazione sia più adatta a catturare le dipendenze contestuali negli LLM utilizzati. Al contrario, la tecnica *Gradient Based* di `Captum` ha presentato limitazioni nell'affidabilità delle spiegazioni, come evidenziato dalla distribuzione non coerente dell'importanza dei token. Questo potrebbe essere attribuito alla natura intrinseca dei gradienti, che possono essere influenzati da rumore o da piccole variazioni nei pesi del modello, portando a interpretazioni meno stabili.

# Capitolo 5

# Riflessioni sui risultati

## 5.1 I nostri risultati

Dopo aver visionato i risultati dei nostri esperimenti abbiamo capito che serviranno ulteriori analisi per comprendere meglio quali siano gli strumenti più adatti ad analizzare gli LLM essendo l'argomento così ampio.

Emergono comunque diverse considerazioni da poter trarre:

**Metodi XAI più Funzionali**

I risultati suggeriscono che i metodi più funzionali in questo ambito siano `LIME` e `Captum` con la versione Perturbation Based. Mentre il metodo Gradient Based sembra dare risultati meno promettenti almeno nelle nostre condizioni rispetto agli esperimenti effettuati, ovvero:

- Utilizzando modelli LLM di dimensioni "ridotte".
- Impiegando un prompt relativamente semplice come input.
- Utilizzando un task relativo al testing delle conoscenza apprese dagli LLM

**Dimensioni e complessità dei Modelli**

Proponiamo l'ipotesi che le dimensioni dei modelli possano influire significativamente sull'efficacia degli strumenti di XAI. In particolare, riteniamo

che alcune tecniche potrebbero essere più adatte a spiegare modelli di piccole dimensioni, offrendo spiegazioni più chiare e dettagliate, mentre altre tecniche potrebbero fornire spiegazioni migliori per modelli di grandi dimensioni, gestendo meglio la complessità e la struttura di tali modelli. Pertanto, l'efficacia di uno strumento XAI potrebbe variare in base alla dimensione del modello, suggerendo la necessità di selezionare o adattare gli strumenti in base alle caratteristiche specifiche del modello in esame.

Tuttavia, riconosciamo che al momento disponiamo di dati limitati per convalidare questa ipotesi.

**Semplicità del Prompt**: La natura semplice e breve del prompt utilizzato ha probabilmente influito positivamente sull'efficacia degli strumenti XAI, consentendo una migliore identificazione dei token rilevanti. Tuttavia, questa scelta comporta anche delle limitazioni, poiché l'utilizzo di frasi semplici potrebbe non riflettere la complessità e le sfumature presenti in contesti reali. Di conseguenza, le spiegazioni generate potrebbero non essere completamente rappresentative delle dinamiche che emergono in prompt più complessi e articolati.

## 5.2 Limiti e Sfide dell'XAI

Nonostante i risultati promettenti, l'applicazione degli strumenti XAI ai LLM presenta diverse sfide:

- **Affidabilità delle Spiegazioni**: Alcuni metodi, come il Gradient Based di Captum, hanno mostrato limitazioni nell'interpretare correttamente l'importanza dei token, evidenziando la necessità di ulteriori perfezionamenti.

- **Complessità dei Modelli**: L'aumentare delle dimensioni e della complessità dei modelli LLM può rendere più difficile l'applicazione efficace degli strumenti XAI, richiedendo approcci più sofisticati.

- **Configurazione degli Strumenti**: La configurazione ottimale degli strumenti XAI, come il numero di campioni in LIME, risulta cruciale per ottenere spiegazioni accurate e affidabili.

## 5.3 Potenziali sviluppi futuri

### Esperimenti più diversificati

In questo primo test più semplice, abbiamo applicato strumenti di XAI agli LLM per valutarne la capacità di ricordare "fatti" di conoscenza comune. Tuttavia, considerando l'ampiezza delle capacità degli LLM, sarebbe necessario condurre una serie di test diversificati per analizzare in modo più approfondito le loro altre capacità emergenti.

Per esempio sviluppi futuri sull'XAI per gli LLM potrebbero ampliare la ricerca con nuovi esperimenti, utilizzando anche le metodologie proposte in questa tesi per analizzare come e quali parti del prompt vengano considerate più importanti in task che richiedano un ragionamento. Sarebbe interessante notare le differenze soprattuto tra i modelli di più piccoli dimensioni e quelli di grandi dimensioni per vedere quali parti del prompt ritengano più utili per la risposta finale.

### Creazione di standard per la spiegabilità degli LLM

Pensiamo inoltre che in futuro potrebbe essere utile sviluppare test standard per poter avere dei report su come un particolare modello appena rilasciato si comporta e perché prenda determinate scelte. Questo sarà un processo lungo, ma aziende come OpenAI hanno già iniziato a mostrare un primo approccio al problema rilasciando per esmepio le "System Card" per i loro ultimi modelli rilasciati (gpt4o [1] e o1 [2]).

[1] GPT-4o System Card: `https://openai.com/index/gpt-4o-system-card/`
[2] o1 System Card: `https://openai.com/index/openai-o1-system-card/`

## Dunning-Kruger effect

Un possibile sviluppo interessante sarebbe dato dal primo risultato iniziale della tabella 4.1. Analizzando i risultati, abbiamo notato che il modello di dimensioni più piccole (*bloomz-560m*) mostra una confidenza maggiore nella previsione rispetto ai modelli più grandi. Questo è interessante, poiché generalmente si potrebbe assumere che modelli con un numero maggiore di parametri possano avere performance migliori e quindi una confidenza più alta nelle loro previsioni.

Una possibile spiegazione potrebbe risiedere nel fatto che modelli più piccoli, avendo una rappresentazione meno complessa del linguaggio, potrebbero sovrastimare la propria "confidenza" nelle previsioni. Al contrario, modelli più grandi, con una comprensione più approfondita e dettagliata, potrebbero essere più "cauti" nelle loro previsioni a causa della maggiore consapevolezza delle ambiguità e delle variabilità presenti nel linguaggio.

Questo comportamento richiama, in modo molto generale, il Dunning-Kruger effect descritto in [40], dove individui con minori competenze tendono a sovrastimare le proprie abilità, mentre quelli più competenti possono sottostimarle. Sebbene applicare direttamente questo concetto psicologico ai modelli di intelligenza artificiale sia speculativo, potrebbe rappresentare un interessante punto di partenza per futuri studi.

Recenti studi infatti hanno iniziato a esplorare analogie simili nei modelli linguistici. Il recente paper [41] ha investigato se i LLMs manifestano bias cognitivi umani come l'effetto Dunning-Kruger. I risultati mostrano che GPT-4 mantiene una buona corrispondenza tra fiducia e accuratezza, mentre modelli come LLaMA e Claude tendono a sovrastimare le proprie capacità, riflettendo un comportamento simile al Dunning-Kruger. Nello studio [42], è stato osservato che la fiducia dei LLM diminuisce con l'aumento della complessità dei compiti. Questo comportamento suggerisce una maggiore consapevolezza delle proprie limitazioni nei modelli di dimensioni maggiori.

Un ulteriore studio significativo è [43], che analizza i conflitti di conoscenza nei modelli linguistici aumentati da recupero (RALMs, Retrieval-

augmented language models). I risultati evidenziano che i RALMs spesso persistono nella fiducia verso una memoria interna errata, anche di fronte a evidenze esterne corrette, mostrando un comportamento analogo al Dunning-Kruger. Inoltre, i RALMs manifestano bias di disponibilità e confirmation bias, preferendo evidenze coerenti con la propria memoria interna. Per affrontare questi problemi, gli autori propongono il metodo Conflict-Disentangle Contrastive Decoding (CD2), che migliora la calibrazione della fiducia e risolve efficacemente i conflitti di conoscenza.

## Domande Aperte

Vorremmo inoltre sollevare alcune domande e ulteriori proposte che richiederanno ulteriori indagini:

- Come si comporterebbero questi strumenti su modelli di dimensioni maggiori o addestrati su dataset più ampi?
- In che misura strumenti di XAI diversi potrebbero complementarsi per fornire spiegazioni più coerenti e complete?
- In quali altre modalità si potrebbe configurare LIME per l'interpretabilità dei risultati degli LLM?
- Come la lingua scelta per il prompt può influenzare le capacità del modello nella previsione del risultato atteso?

Con queste domande vorremo proporre altre opportunità non ancora esplorate per future ricerche nel campo.

# Conclusioni

In conclusione, possiamo affermare che le prospettive sull'Explainable AI (XAI) applicata ai Large Language Models (LLM) siano promettenti, almeno in parte o per modelli di dimensioni più ridotte. I nostri esperimenti infatti hanno dimostrato che alcuni strumenti come `LIME` e `Captum` (nella sua versione Perturbation Based) risultano particolarmente efficaci nel fornire spiegazioni comprensibili e affidabili per questi modelli. Anche se sarà necessario ulteriore lavoro di ricerca per capire se queste tecniche si potranno poi scalare ed applicare anche per modelli più complessi.

Inoltre la semplicità del prompt utilizzato nei nostri esperimenti ha facilitato l'identificazione dei token rilevanti, evidenziando come la natura dell'input possa influenzare l'efficacia degli strumenti di XAI. Questo sottolinea l'importanza di considerare la formulazione del prompt nella progettazione di esperimenti di explainability.

Nonostante i limiti riscontrati, l'ampia varietà di strumenti di XAI disponibili offre una solida base per iniziare a sviluppare una prima prospettiva di explainability per gli LLM. Le principali sfide attuali includono l'affidabilità delle spiegazioni fornite, la gestione della complessità dei modelli e la configurazione ottimale degli strumenti stessi. Questi aspetti richiederanno ulteriori ricerche e perfezionamenti per garantire spiegazioni sempre più accurate e affidabili.

Per il futuro, si auspica di esplorare test più approfonditi delle capacità degli LLM, sviluppare standard di testing per valutare in modo coerente i nuovi modelli e indagare fenomeni come l'effetto Dunning-Kruger nei mo-

delli di intelligenza artificiale. Inoltre, l'analisi della comprensione dei task di ragionamento e l'integrazione di diverse tecniche di XAI potrebbero contribuire a migliorare ulteriormente la trasparenza e l'affidabilità dei modelli linguistici di grandi dimensioni.

Attraverso questo lavoro, abbiamo supportato le basi per una migliore comprensione degli LLM, favorendo lo sviluppo di intelligenze artificiali più trasparenti e affidabili. Sebbene siano necessari ulteriori studi per affrontare le sfide emerse, le nostre conclusioni indicano una direzione positiva per l'integrazione dell'XAI nei modelli linguistici, favorendo una maggiore fiducia e sicurezza nell'uso di tali tecnologie.

# Appendice A

# Stochastic Gradient Descent (SGD)

In questa appendice viene fornita una descrizione dettagliata dell'algoritmo **Stochastic Gradient Descent (SGD)**, inclusa la sua formula matematica e le sue varianti.

## A.1 Formula di Aggiornamento

La formula di aggiornamento per i pesi $\theta$ è la seguente:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta L(\theta_t; x_t, y_t)$$

### A.1.1 Descrizione dei Termini

- $\theta_t$: Vettore dei pesi al passo $t$.
- $\eta$: Tasso di apprendimento (learning rate).
- $\nabla_\theta L(\theta_t; x_t, y_t)$: Gradiente della funzione di perdita $L$ rispetto ai pesi $\theta$, calcolato sul campione $(x_t, y_t)$.

# Appendice B

# Adam (Adaptive Moment Estimation)

Questa appendice approfondisce l'algoritmo **Adam**, illustrandone le formule di aggiornamento e le sue caratteristiche principali.

## B.1 Formule di Aggiornamento

Le formule di aggiornamento di Adam sono le seguenti:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
\theta_{t+1} &= \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}
$$

### B.1.1 Descrizione dei Termini

- $m_t$: Stima del primo momento (media) al passo $t$.
- $v_t$: Stima del secondo momento (varianza) al passo $t$.

- $\beta_1$: Fattore di decadimento per il primo momento (tipicamente vicino a 0.9).
- $\beta_2$: Fattore di decadimento per il secondo momento (tipicamente vicino a 0.999).
- $g_t$: Gradiente della funzione di perdita rispetto ai pesi al passo $t$.
- $\hat{m}_t$: Stima corretta del primo momento.
- $\hat{v}_t$: Stima corretta del secondo momento.
- $\eta$: Tasso di apprendimento.
- $\epsilon$: Termine di stabilizzazione (tipicamente $10^{-8}$) per evitare divisioni per zero.

# Appendice C

# Reti Neurali Ricorrenti (RNN)

## C.1 Introduzione

Le **Reti Neurali Ricorrenti (RNN)** rappresentano una classe di reti neurali progettate per elaborare dati sequenziali o temporali. A differenza delle reti neurali feedforward tradizionali, le RNN possiedono connessioni ricorrenti che permettono di mantenere uno stato interno, consentendo così di memorizzare informazioni sulle sequenze precedenti.

### C.1.1 Architettura

L'architettura di base di una RNN può essere descritta come segue:

- **Input Sequenziale**: La rete riceve una sequenza di input $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, dove $T$ è la lunghezza della sequenza.
- **Stati Nascosti**: Per ogni passo temporale $t$, la rete aggiorna il suo stato nascosto $\mathbf{h}_t$ in base all'input corrente $x_t$ e allo stato nascosto precedente $\mathbf{h}_{t-1}$.

  $$\mathbf{h}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

  dove:

- $\mathbf{W}_{hx}$ è la matrice dei pesi per l'input.
- $\mathbf{W}_{hh}$ è la matrice dei pesi per le connessioni ricorrenti.
- $\mathbf{b}_h$ è il bias.
- tanh è una funzione di attivazione non lineare.

- **Output**: L'output $\mathbf{y}_t$ al passo $t$ può essere calcolato come:

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y$$

dove $\mathbf{W}_{hy}$ è la matrice dei pesi per l'output e $\mathbf{b}_y$ è il bias dell'output.

### C.1.2 Vantaggi

- **Memorizzazione di Dipendenze Temporali**: Le RNN possono teoricamente catturare dipendenze a lungo termine nelle sequenze, grazie alla loro struttura ricorrente.
- **Modellazione di Dati Sequenziali**: Sono particolarmente adatte per compiti come la traduzione automatica, il riconoscimento vocale e la generazione di testo.

### C.1.3 Limitazioni

- **Problemi di Gradiente**: Durante l'addestramento tramite retropropagazione attraverso il tempo (BPTT), le RNN soffrono spesso di problemi di gradiente che esplode o svanisce, rendendo difficile l'apprendimento di dipendenze a lungo raggio.
- **Difficoltà di Parallelizzazione**: La natura sequenziale delle RNN rende difficile la parallelizzazione dei calcoli, limitando l'efficienza computazionale.
- **Complessità nella Gestione delle Dipendenze a Lungo Termine**: Sebbene teoricamente capaci di catturare dipendenze a lungo termine, in pratica le RNN standard faticano a farlo a causa dei problemi di gradiente.

# Appendice D

# Long Short-Term Memory (LSTM)

## D.1 Introduzione

Per superare le limitazioni delle RNN tradizionali, le **Long Short-Term Memory (LSTM)** sono state introdotte da [8]. Le LSTM sono un tipo di RNN progettate specificamente per catturare dipendenze a lungo termine e mitigare i problemi di gradiente.

### D.1.1 Architettura

Le LSTM introducono celle di memoria e meccanismi di gating che regolano il flusso delle informazioni attraverso la rete. Un'unità LSTM standard è composta da tre porte principali:

- **Porta di Input ($i_t$)**: Decide quali informazioni aggiornare nella cella di memoria.

$$i_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

- **Porta di Dimenticanza ($f_t$)**: Determina quali informazioni della cella di memoria precedente devono essere mantenute o dimenticate.

$$f_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

- **Porta di Output** ($o_t$): Stabilisce quali informazioni della cella di memoria devono essere utilizzate per l'output.

$$o_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

Oltre alle porte, le LSTM mantengono una **cellula di memoria** $\mathbf{C}_t$ che viene aggiornata come segue:

$$\mathbf{C}_t = f_t \odot \mathbf{C}_{t-1} + i_t \odot \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)$$

dove $\odot$ rappresenta il prodotto elemento per elemento.

L'output nascosto $\mathbf{h}_t$ è quindi calcolato come:

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{C}_t)$$

### D.1.2 Vantaggi

- **Gestione delle Dipendenze a Lungo Termine**: Grazie ai meccanismi di gating, le LSTM possono mantenere informazioni rilevanti per lunghi periodi, superando i limiti delle RNN tradizionali.

- **Stabilità del Gradiente**: Le LSTM mitigano i problemi di gradiente che esplode o svanisce, facilitando l'addestramento su sequenze più lunghe.

- **Flessibilità**: Le LSTM possono essere utilizzate in diverse varianti, come le **Gated Recurrent Units (GRU)**, che semplificano l'architettura mantenendo performance simili.

### D.1.3 Limitazioni

- **Complessità Computazionale**: Le LSTM hanno una struttura più complessa rispetto alle RNN tradizionali, comportando un aumento dei costi computazionali.

- **Difficoltà di Addestramento**: Nonostante migliorino la gestione delle dipendenze a lungo termine, le LSTM possono ancora essere difficili da addestrare su sequenze molto lunghe.

# Bibliografia


[1] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[2] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[4] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[5] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[6] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.

[11] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[12] Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.

[13] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[14] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of

interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[16] Scott Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[17] Zihan Ji, Tianyu Yu, Yong Xu, et al. Towards mitigating llm hallucination via self-reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843. Association for Computational Linguistics, 2023.

[18] Upol Ehsan and Mark Riedl. Explainable ai reloaded: Challenging the xai status quo in the era of large language models. In *Proceedings of the Halfway to the Future Symposium*, pages 1–8, 2024.

[19] Marco Polignano, Cataldo Musto, Roberto Pellungrini, Erasmo Purificato, Giovanni Semeraro, and Mattia Setzu. Xai. it 2024: An overview on the future of ai in the era of large language models. 2024.

[20] David Martens, James Hinns, Camille Dams, Mark Vergouwen, and Theodoros Evgeniou. Tell me a story! narrative-driven xai with large language models. *arXiv preprint arXiv:2309.17057*, 2023.

[21] Alexandra Zytek, Sara Pidò, and Kalyan Veeramachaneni. Llms for xai: Future directions for explaining explanations. *arXiv preprint arXiv:2405.06064*, 2024.

[22] Alexandre Piché, Aristides Milios, Dzmitry Bahdanau, and Chris Pal. LLMs can learn self-restraint through iterative self-reflection. *arXiv preprint arXiv:2405.13022*, 2024.

[23] Teresa Datta and John P Dickerson. Who's thinking? a push for human-centered evaluation of llms using the xai playbook. *arXiv preprint arXiv:2303.06223*, 2023.

[24] Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. Llms as counterfactual explanation modules: Can chatgpt explain black-box text classifiers? *arXiv preprint arXiv:2309.13340*, 2023.

[25] Walid S. Saba. Stochastic LLMs Do Not Understand Language: Towards Symbolic, Explainable, and Ontologically Based LLMs. In *Conceptual Modeling: 42nd International Conference, ER 2023, Lisbon, Portugal, November 6–9, 2023, Proceedings*, pages 3–19, Cham, 2023. Springer.

[26] Yuzhe You and Jian Zhao. Gamifying xai: Enhancing ai explainability for non-technical users through llm-powered narrative gamifications. *arXiv preprint arXiv:2410.04035*, 2024.

[27] Zichen Chen, Jianda Chen, Mitali Gaidhani, Ambuj Singh, and Misha Sra. Xplainllm: A qa explanation dataset for understanding llm decision-making. *arXiv preprint arXiv:2311.08614*, 2023.

[28] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[29] Vivek Miglani, Aobo Yang, Aram H Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. Using captum to explain generative language models. *arXiv preprint arXiv:2312.05491*, 2023.

[30] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. Ai explainability 360: Impact and design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12651–12657, 2022.

[31] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020.

[32] Wenzhuo Yang, Hung Le, Tanmay Laud, Silvio Savarese, and Steven CH Hoi. Omnixai: A library for explainable ai. *arXiv preprint arXiv:2206.01612*, 2022.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[34] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal of Machine Learning Research*, 22(181):1–7, June 2021.

[35] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[36] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.

[37] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Lewis Tunstall, Agustín Piqueres, Andres Marafioti, Cyril Zakka, Leandro von Werra, and Thomas Wolf. Smollm2 - with great data, comes great performance, 2024.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[39] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.

[40] Justin Kruger and David Dunning. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology*, 77(6):1121, 1999.

[41] Aniket Kumar Singh, Bishal Lamichhane, Suman Devkota, Uttam Dhakal, and Chandra Dhakal. Do large language models show human-like biases? exploring confidence—competence gap in ai. *Information*, 15(2):92, 2024.

[42] Aniket Kumar Singh, Suman Devkota, Bishal Lamichhane, Uttam Dhakal, and Chandra Dhakal. The confidence-competence gap in large language models: A cognitive study. *arXiv preprint arXiv:2309.16145*, 2023.

[43] Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, Xiaojian Jiang, Jiexin Xu, Qiuxia Li, and Jun Zhao. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. *arXiv preprint arXiv:2402.14409*, 2024.

# Ringraziamenti

Vorrei ringraziare la mia famiglia per il loro sostegno incondizionato e la loro costante presenza durante tutto il mio percorso di studi.

Un ringraziamento speciale va inoltre al mio relatore di tesi, il Dott. Davide Evangelista per il prezioso supporto, la guida e la disponibilità offerta durante tutto il percorso di svolgimento della tesi, nonché per il suo sostegno nei momenti di maggiore difficoltà.

Ringrazio inoltre Google per avermi fornito accesso gratuito alle loro GPU tramite il servizio Colab, e il nostro dipartimento per l'accesso al cluster HPC, strumenti fondamentali che hanno reso possibili i miei esperimenti e le mie ricerche.

Vorrei infine ringraziare i miei vecchi compagni Alberico, Antonio e Nicola che, anche se ci siamo successivamente divisi per strade diverse, mi hanno supportato durante i primi anni di università.

Per concludere vorrei dire che anche se negli ultimi anni il mio percorso universitario è stato un po' travagliato, riuscire a concludere questa tappa è stato per me importante e una sfida che, sotto diversi aspetti, mi ha permesso di crescere.