PRISMA[1]: Disentanglement e scomposizione in feature monosemantiche delle rappresentazioni latenti nei LLM tramite Sparse Autoencoders

Edoardo Tedesco

1 Introduzione e Problema

L'avvento dei Large Language Models (LLM) ha segnato un punto di svolta nell'elaborazione del linguaggio naturale (**devlin2018bert**), consentendo alle macchine di apprendere complesse sfumature semantiche celate nei simboli testuali. Tuttavia, questa straordinaria capacità predittiva ha un costo significativo: la perdita di interpretabilità. I modelli operano proiettando le parole in spazi vettoriali ad altissima dimensionalità (embedding densi), i cui assi non possiedono un significato intrinseco umanamente comprensibile. In uno scenario ideale, per garantire trasparenza e sicurezza, sarebbe auspicabile che i singoli neuroni della rete corrispondessero a concetti isolati. Ad esempio, vorremmo poter identificare un singolo neurone che si attiva linearmente solo in presenza di concetti interpretabili come "colore rosso" o "muso di cane". In termini formali, l'obiettivo è ottenere una rappresentazione in cui il numero di dimensioni (neuroni) corrisponda esattamente ai **fattori di variazione** dei dati sottostanti (**wang2024disentangledrepresentationlearning**). Empiricamente, tuttavia, si osserva raramente tale corrispondenza. La discrepanza tra i concetti umani e le attivazioni neurali è alla base del **problema dell'allineamento** (**elhage2022toy**): diventa arduo fidarsi di un sistema se non se ne comprendono i meccanismi decisionali interni. Una risposta teorica a questo fenomeno è fornita dalla **Superposition Hypothesis** (**elhage2022toy**). Secondo questa ipotesi, le reti neurali sfruttano il fatto che vi sono concetti mutuamente esclusivi per rappresentare un numero di feature molto superiore al numero di neuroni fisici disponibili (e.g. il concetto di "meccanica quantistica" raramente si attiva in presenza di "torta al cioccolato"). I modelli simulano una rete molto più grande "comprimendo" le informazioni tramite interferenza controllata. Sebbene efficiente, questo meccanismo genera *polisemanticità*, rendendo la rete una "black box". Il presente lavoro di tesi propone l'implementazione di un metodo per invertire questo processo attraverso il **disentanglement** delle rappresentazioni. L'obiettivo è proiettare gli embedding densi in nuovi spazi semantici sparsi, dotati di assi interpretabili (**oneill2024disentangling**). A tale scopo è stata sviluppata **Prisma**, un'applicazione che, analogamente a un prisma ottico che scompone la luce bianca nel suo spettro, isola i concetti atomici costitutivi del testo.

2 Prisma: Metodologia

2.1 Autoencoders Classici

Un Autoencoder (AE) è un'architettura di rete neurale non supervisionata addestrata per apprendere una funzione identità $h_{w,b}(x) \approx x$. Un AE è composto tipicamente da due moduli: un *encoder* che comprime l'input in una rappresentazione latente z di dimensione inferiore (*bottleneck*), e un *decoder* che ricostruisce l'input a partire da z.

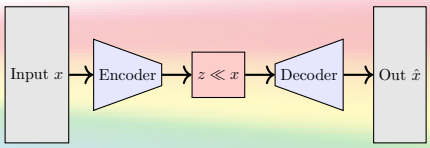

Figura 1: Architettura di un Autoencoder classico: il bottleneck forza la compressione informativa.

Gli AE classici possono estrarre fattori principali, ma non garantiscono che le dimensioni latenti siano monosemantiche o regolarizzate. L'assenza di vincoli espliciti sulla struttura di z crea uno spazio latente "discontinuo".

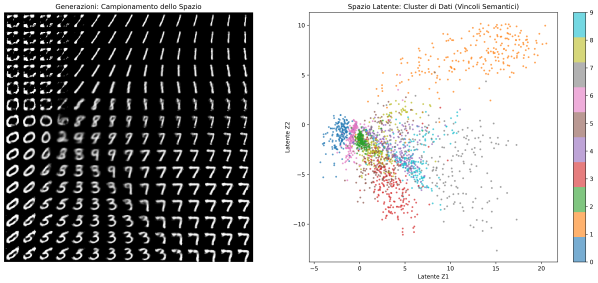

Figura 2: Analisi della semantica nell'AE. A destra: lo spazio latente mostra cluster isolati (vincoli imposti dai dati). A sinistra: le generazioni mostrano che nelle zone vuote tra i cluster, dove mancano i vincoli, la semantica scompare lasciando spazio a ricostruzioni incoerenti.

Come mostrato in Figura 2, la semantica emerge come sottomanifold indotto dai dati; al di fuori di questi vincoli, il modello non ha capacità generativa coerente.

2.2 Sparse Autoencoders (SAE)

Per ottenere *disentanglement* e feature più interpretabili, Prisma utilizza uno **Sparse Autoencoder** (**oneill2024disentangling**). A differenza degli AE classici, un SAE usa una rappresentazione latente *overcomplete*: la dimensione latente n è maggiore della dimensione dell'input d, definendo l'expansion factor $\rho = \frac{n}{d}$ con tipicamente $\rho \in [2, 9]$ (**oneill2024disentangling**). Contestualmente all'overcompletezza, si impone un forte vincolo di sparsità, costringendo solo un numero esiguo di neuroni ad attivarsi per ogni input. L'intuizione è che l'overcompletezza fornisca molte "direzioni" potenzialmente disponibili, mentre la sparsità forza il modello a selezionarne poche per rappresentare un dato input, favorendo feature più stabili e interpretabili.

2.3 Architettura Matematica

Seguiamo il formalismo di (**oneill2024disentangling**). Sia $x \in \mathbb{R}^d$ l'input (embedding) e $h \in \mathbb{R}^n$ la rappresentazione latente nascosta, con $n > d$. L'architettura è definita da due mappe parametrizzate:

$$\theta = \{W_e, b_e\}, \qquad \phi = \{W_d, b_d\}.$$

[1] Acronimo per ***P**rojection of **R**epresentations for **I**nterpretability via **S**parse **M**onosemantic **A**utoencoders.*

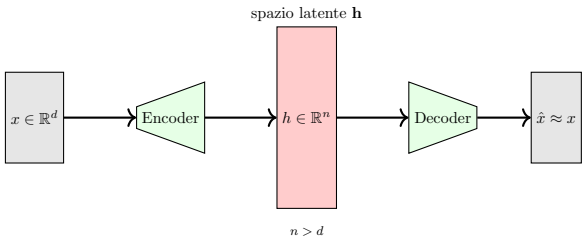

Figura 3: Sparse Autoencoder: spazio latente overcomplete con attivazioni sparse.

(i) **Encoder:** mappa l'input nella rappresentazione nascosta.

$$h = f_\theta(x) = \sigma(W_e x + b_e) \tag{1}$$

dove $W_e \in \mathbb{R}^{n \times d}$, $b_e \in \mathbb{R}^n$ e $\sigma(\cdot)$ è una non-linearità (ReLU).

(ii) **Decoder:** ricostruisce l'input dalla rappresentazione latente.

$$\hat{x} = g_\phi(h) = W_d h + b_d \tag{2}$$

dove $W_d \in \mathbb{R}^{d \times n}$ e $b_d \in \mathbb{R}^d$. Il decoder è **lineare**: di conseguenza la ricostruzione è una *combinazione lineare* dei vettori-feature (le colonne di W_d) pesata dalle attivazioni in h, rendendo ciascuna feature interpretabile come una *direzione* nello spazio degli embedding (**oneill2024disentangling**). *Nota:* la linearità non implica che tali "concetti" siano indipendenti o ortogonali; in uno spazio con più feature che dimensioni l'ortogonalità perfetta è impossibile, coerentemente con l'idea di **superposizione** discussa in introduzione (**elhage2022toy**).

2.4 Training e Loss Function

L'obiettivo di addestramento combina: (i) una perdita di ricostruzione, (ii) un vincolo di sparsità, e (iii) una perdita ausiliaria per mitigare il problema dei *dead latents* (**oneill2024disentangling**). La funzione di costo complessiva è:

$$\mathcal{L} = \underbrace{\frac{1}{d}\|x - \hat{x}\|_2^2}_{\text{Reconstruction}} + \lambda \mathcal{L}_{sparse}(h) + \alpha \mathcal{L}_{aux}(x, \hat{x}) \tag{3}$$

dove $\lambda > 0$ e $\alpha > 0$ controllano il trade-off tra fedeltà, sparsità e recupero delle feature inattive.

2.4.1 Vincolo di Sparsità (Top-K / k-sparse)

Il termine $\mathcal{L}_{sparse}(h)$ è implementato tramite un vincolo **Top-K**: durante il forward pass, vengono mantenute solo le k attivazioni maggiori in h, mentre le restanti sono poste a zero (**oneill2024disentangling**). Questo differisce dalla regolarizzazione L_1, che può introdurre *shrinkage* sistematico delle attivazioni e quindi rappresentazioni meno fedeli.

2.4.2 Auxiliary Loss (AuxK, ispirata ai Ghost Grads)

Per mitigare il problema dei *dead latents* (feature che restano inattive su scala di training, ad esempio non attivandosi per un intero epoch), si introduce una perdita ausiliaria ispirata alla tecnica dei *ghost grads* (**oneill2024disentangling**). Definito l'errore di ricostruzione $e = x - \hat{x}$, la perdita ausiliaria è:

$$\mathcal{L}_{aux}(x, \hat{x}) = \|e - \hat{e}\|_2^2 \tag{4}$$

dove $\hat{e}$ è una ricostruzione dell'errore ottenuta usando un piccolo sottoinsieme di latenti selezionati (AuxK), tipicamente con $k_{aux} \approx 2k$ (**oneill2024disentangling**). Questo termine forza il modello a sfruttare anche feature raramente attive per spiegare l'errore residuo, aumentando l'utilizzo effettivo della capacità latente.

2.5 Rappresentazione Latente

Applicando l'encoder a un batch di documenti si ottiene la matrice delle attivazioni sparse H. Le attivazioni risultano puntiformi e strutturate, coerentemente con l'obiettivo di ottenere feature (più) monosemantiche e interpretabili.

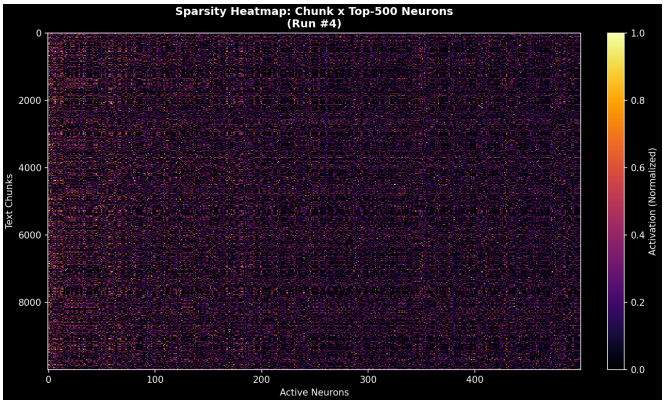

Figura 4: Visualizzazione della matrice sparsa H (primi 500 neuroni). Le attivazioni sono puntiformi e verticali.

2.6 Interpretabilità degli assi

Una volta addestrato il SAE, ogni feature può essere interpretata associandole una descrizione testuale. Nel paper, questa fase avviene tramite un LLM *Interpreter* che, osservando esempi che massimizzano l'attivazione di una feature e contro-esempi che non la attivano, produce un'etichetta semantica (topic/concetto) per quella direzione latente (**oneill2024disentangling**). Nel contesto di Prisma viene implementata questa sola fase di labelling (senza il successivo step di *Predictor*).

3 Analisi della Dimensionalità: Effective Rank

La matrice delle attivazioni sparse $H \in \mathbb{R}^{N \times n}$ (documenti $\times$ feature) non codifica feature perfettamente indipendenti: in domini reali esistono co-attivazioni sistematiche tra concetti (es. *febbre* $\leftrightarrow$ *polmonite*), che riducono i gradi di

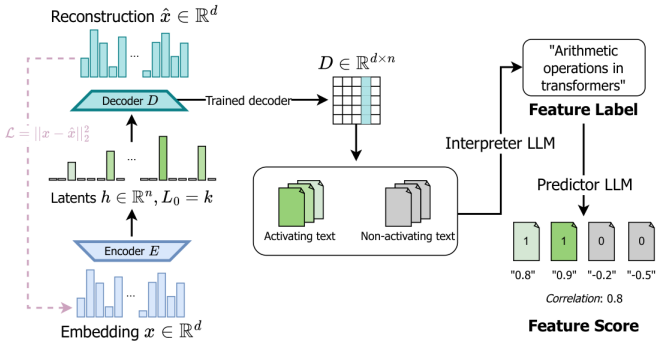

Figura 5: Processo di training e feature labelling del SAE: interpretazione delle feature tramite LLM (**oneill2024disentangling**).

libertà effettivi del codice. Per quantificare questa *dimensione effettiva* in modo continuo e robusto (a differenza del rango classico, instabile in presenza di rumore), adottiamo l'**Effective Rank** (**roy2007effective**), definito tramite l'entropia dello spettro singolare normalizzato:

$$ER(H) \;=\; \exp\left(-\sum_i p_i \log p_i\right), \qquad p_i \;=\; \frac{\sigma_i}{\|\sigma\|_1}, \quad (5)$$

dove $\{\sigma_i\}$ sono i valori singolari di H. Intuitivamente, $ER(H)$ è alto se la varianza è distribuita su molte direzioni (codice più *isotropo*), ed è basso se concentrata in poche direzioni (codice più *compresso*).

Stimiamo $ER(H)$ al variare dell'overcompletezza del SAE (expansion factor, quindi $n = d_{\text{latent}}$) e confrontiamo i dati reali con una *ipotesi nulla* ottenuta addestrando lo stesso SAE su input casuali non strutturati. Per sintetizzare l'effetto introduciamo il **Semantic Compression Ratio** (SCR), che misura la compressione relativa rispetto al rumore:

$$SCR(\%) \;=\; 100 \cdot \frac{ER_{\text{null}} - ER_{\text{real}}}{ER_{\text{null}}}. \quad (6)$$

Empiricamente osserviamo che, aumentando la capacità latente, ER cresce in entrambi i casi, ma cresce *sistematicamente meno* sui dati reali, indicando che la semantica impone vincoli globali di co-attivazione che riducono la dimensionalità effettiva del codice.

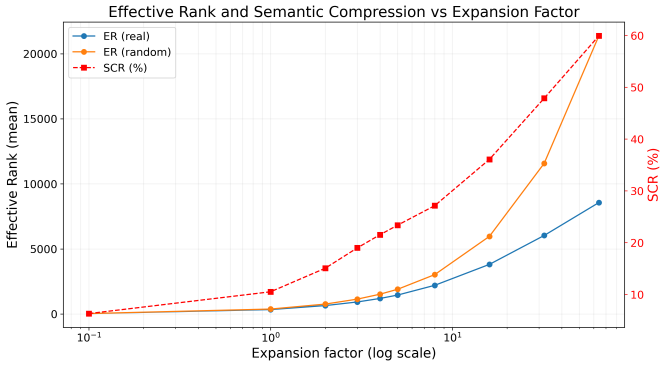

Figura 6: Effective Rank vs expansion factor (real vs ipotesi nulla) e Semantic Compression Ratio (SCR%).

4 Conclusioni

Il lavoro svolto con Prisma dimostra che l'interpretabilità dei LLM può essere recuperata invertendo il processo di *superposition*. Proiettando le attivazioni in uno spazio overcomplete e forzando la sparsità, i SAE agiscono come filtri che isolano feature monosemantiche. La tesi centrale è che la **semantica** non sia un'entità astratta, ma l'espressione di un insieme di **vincoli** che strutturano lo spazio latente. Senza vincoli (come nell'AE classico in zone non mappate), lo spazio è caotico e privo di informazione. Al contrario, nei dati strutturati, la semantica costringe le attivazioni su varietà a bassa dimensionalità. L'analogia fisica più calzante è quella dello **spazio delle fasi** di un sistema dinamico (Figura 7). Un pendolo libero di muoversi senza leggi fisiche potrebbe occupare qualunque punto (θ, ω). Tuttavia, la conservazione dell'energia (il vincolo) lo costringe a evolvere solo lungo orbite specifiche.

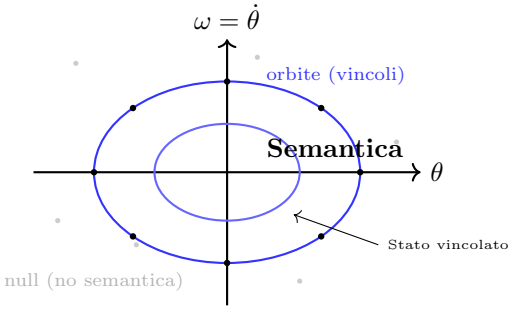

Figura 7: Analogia: nello spazio delle fasi (θ, ω), la fisica vincola il sistema su orbite precise. Analogamente, nei modelli Prisma, la semantica agisce come il vincolo che modella la distribuzione delle attivazioni latenti, separandole dal rumore dell'ipotesi nulla.