Marzo 2026

La geometria del rifiuto nei modelli linguistici

Steering e rappresentazione nello spazio latente

AGID | Agenzia per l'Italia Digitale

CERT-AGID

La geometria del rifiuto nei modelli linguistici

Steering e rappresentazione nello spazio latente

Abstract

Questo lavoro approfondisce l'analisi presentata nello studio *Analisi del rifiuto nei modelli linguistici attraverso tecniche di Steering* (2025)[1], in cui è stato osservato il comportamento di *refusal* dei modelli linguistici analizzando le attivazioni interne *layer-by-layer* e dimostrando la possibilità di modificarlo tramite tecniche di *activation steering*.

In questo studio estendiamo l'analisi adottando una prospettiva geometrica sulle rappresentazioni latenti del modello. In particolare, analizziamo se la separazione tra prompt *harmful* e *benign* nello spazio latente possa essere descritta attraverso una direzione di *steering* e valutiamo se metodi più generali, come **Recursive Feature Machines (RFM)**, offrano vantaggi rispetto ad approcci più semplici come il **Consensus vector** o l'analisi tramite **Singular Value Decomposition (SVD)**.

L'analisi è condotta sul modello *open-weight* **Mistral-7B-Instruct-v0.2**, un'architettura *transformer* composta da 32 *layer* e circa 7 miliardi di parametri. La pipeline sperimentale si articola in quattro fasi principali: estrazione delle attivazioni interne, analisi della separazione tra prompt *harmful* e *benign* nei diversi *layer*, apprendimento di una direzione di *steering* nello spazio latente e valutazione su benchmark bilanciati.

L'obiettivo non è semplicemente forzare il modello a rispondere a richieste *harmful*, ma verificare se sia possibile **ridurre i rifiuti a prompt malevoli mantenendo stabilità sui prompt *benign***, evitando allo stesso tempo effetti di degenerazione o deriva semantica.

Oltre ai risultati tecnici, il lavoro evidenzia come l'utilizzo di modelli *open-weight* consenta di analizzare direttamente le rappresentazioni interne dei modelli e di sperimentare tecniche di controllo del comportamento senza trasferire dati sensibili verso servizi esterni, un aspetto particolarmente rilevante per contesti istituzionali e per la sicurezza dei dati.

Introduzione

I large language model spesso rifiutano di rispondere a richieste considerate pericolose o non conformi alle policy. Questo comportamento, chiamato *refusal*, viene generalmente interpretato come il risultato di regole apprese durante l'addestramento.

[1] Analisi del rifiuto nei modelli linguistici attraverso tecniche di Steering (2025) https://www.agid.gov.it/sites/agid/files/2025-12/Analisi_rifiuto_modelli_linguistici.pdf

Una possibile interpretazione alternativa è che il *refusal* emerga come **una struttura geometrica nello spazio latente del modello**. Quando il modello elabora un prompt, infatti, il testo viene trasformato in una rappresentazione numerica interna composta da migliaia di valori. Questa rappresentazione può essere vista come un punto nello spazio latente.

Se osserviamo molte richieste diverse, emerge spesso una struttura:

- i prompt *benign* tendono a raggrupparsi in una regione dello spazio
- i prompt associati al *refusal* tendono a concentrarsi in un'altra regione

Se questa separazione esistesse, allora dovrebbe essere possibile identificare **una direzione nello spazio latente** che separi le due regioni.

Una volta trovata questa direzione, possiamo modificare leggermente le attivazioni interne del modello durante la generazione per spostare la rappresentazione di un prompt. Questa tecnica è chiamata **activation steering**.

Nel lavoro precedente l'attenzione era concentrata sul **Consensus vector**, una direzione stimata come differenza tra le attivazioni medie dei prompt *harmful* e *benign*. Sebbene questo approccio permettesse di ottenere una risposta alla richiesta *harmful* analizzata, introduceva effetti di degrado nel comportamento del modello sui prompt *benign*.

Nel presente lavoro introduciamo un metodo più generale, basato su **Recursive Feature Machines**, e lo confrontiamo direttamente con il metodo precedente (*Consensus vector*) e con il metodo **SVD**.

La domanda centrale diventa quindi:

- esiste una direzione che controlla il comportamento di refusal?
- il metodo RFM apprende una direzione diversa e più efficace rispetto al consensus vector?
- è possibile ridurre il *refusal* sui prompt *harmful* **senza compromettere la coerenza semantica in risposta ai prompt benign**?

Schema della pipeline sperimentale

L'analisi segue una pipeline composta da diverse fasi sequenziali che permettono di passare dall'osservazione delle rappresentazioni interne del modello alla valutazione empirica dello steering sul comportamento di *refusal*.

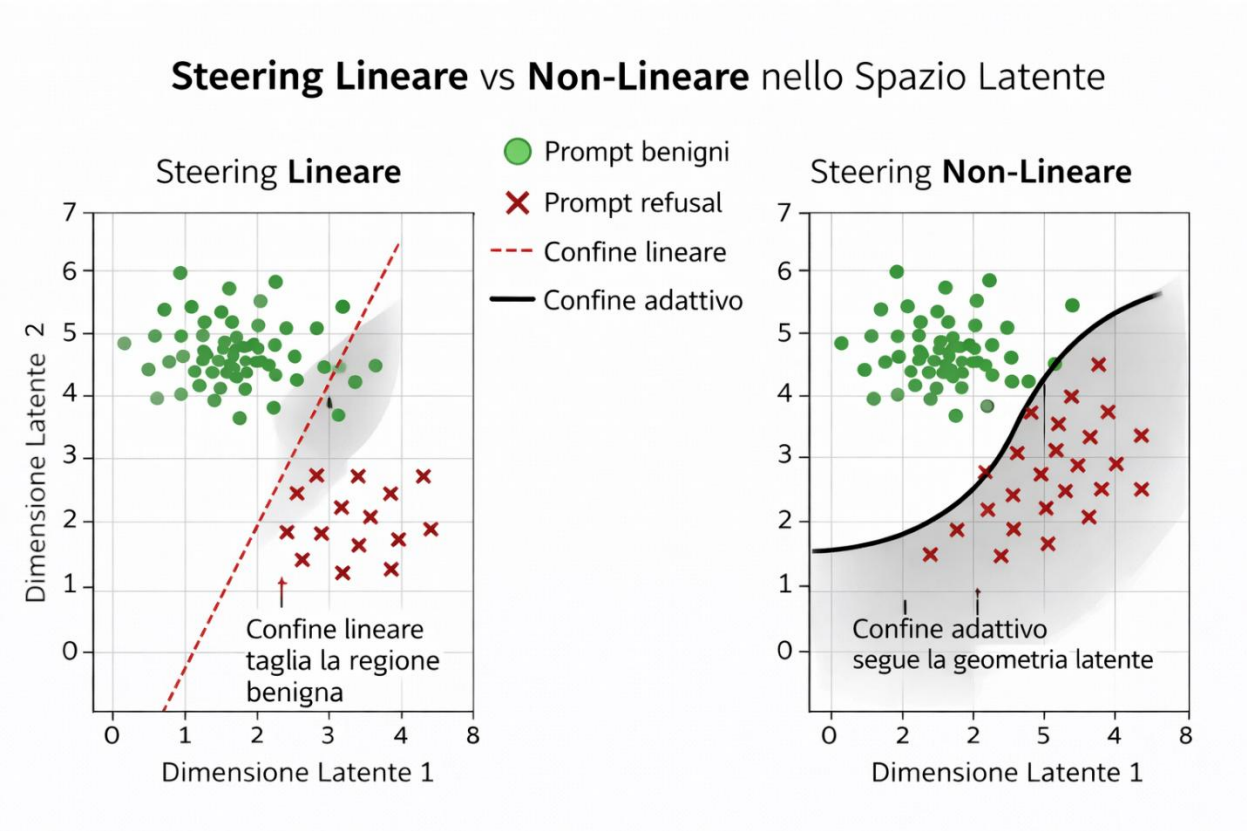
Si parte dall'estrazione delle attivazioni latenti del modello per un insieme di prompt *harmful* e *benign*. Queste rappresentazioni vengono analizzate per individuare i layer in cui la separazione tra le due classi è più evidente. Su questi layer candidati viene quindi stimata una direzione di steering nello spazio latente utilizzando diversi metodi (Consensus, RFM e SVD).

La direzione ottenuta viene applicata durante la generazione modificando leggermente le attivazioni interne del modello attraverso il parametro di intensità ***alpha***, che controlla quanto forte sia lo spostamento lungo la direzione di steering. Le diverse configurazioni vengono infine valutate su benchmark bilanciati e replicate su un secondo corpus per verificare la stabilità dei risultati.

Lo schema seguente riassume le principali fasi della pipeline.

```
prompt
↓
estrazione attivazioni latenti
↓
analisi separazione harmful / benign
↓
selezione layer candidati
↓
apprendimento direzione di steering
↓
iniezione dello steering
↓
valutazione su benchmark
↓
replica su corpus esterno
```

Geometria del refusal nello spazio latente



La figura illustra il principio geometrico alla base dello steering nello spazio latente. I metodi di steering lineare, come il **Consensus vector**, assumono implicitamente che queste due regioni possano essere separate da un confine lineare. In questa ipotesi, lo steering consiste nell'aggiungere una perturbazione lungo una singola direzione globale nello spazio latente.

Tuttavia, se la separazione reale tra le due distribuzioni non è perfettamente lineare, una perturbazione di questo tipo può attraversare anche la regione benigna, producendo effetti collaterali o deriva semantica. L'approccio basato su **Recursive Feature Machines (RFM)** mira invece ad apprendere una frontiera di separazione che segue la geometria effettiva delle rappresentazioni latenti. In questo caso lo steering non impone una direzione rigida ma si allinea alla forma della frontiera tra le due regioni, consentendo in principio di modificare il comportamento del modello vicino alla zona critica di separazione senza compromettere la stabilità delle risposte sui prompt *benign*. Questa interpretazione geometrica costituisce il punto di partenza dell'analisi sperimentale presentata nel resto del lavoro.

Metodo

Dataset di prompt

Il problema viene formulato come contrasto tra due tipi di richieste:

1. 50 prompt **harmful**
2. 50 prompt **benign**

Per verificare che i risultati non dipendano da una singola collezione di prompt, costruiamo anche un secondo benchmark di replica 50/50 utilizzando un corpus esterno.

Estrazione delle attivazioni

Per ogni prompt eseguiamo una sola esecuzione del modello e registriamo le attivazioni interne dell'ultimo token per ogni layer del trasformatore.

Cosa succede quando diamo un prompt al modello

Quando inseriamo un prompt, il modello lo elabora token per token attraversando tutti i layer del trasformatore. Ad ogni layer viene prodotta una rappresentazione interna del testo. Questa rappresentazione è un vettore numerico molto grande (la rappresentazione latente).

Quindi per ogni prompt otteniamo qualcosa del tipo:

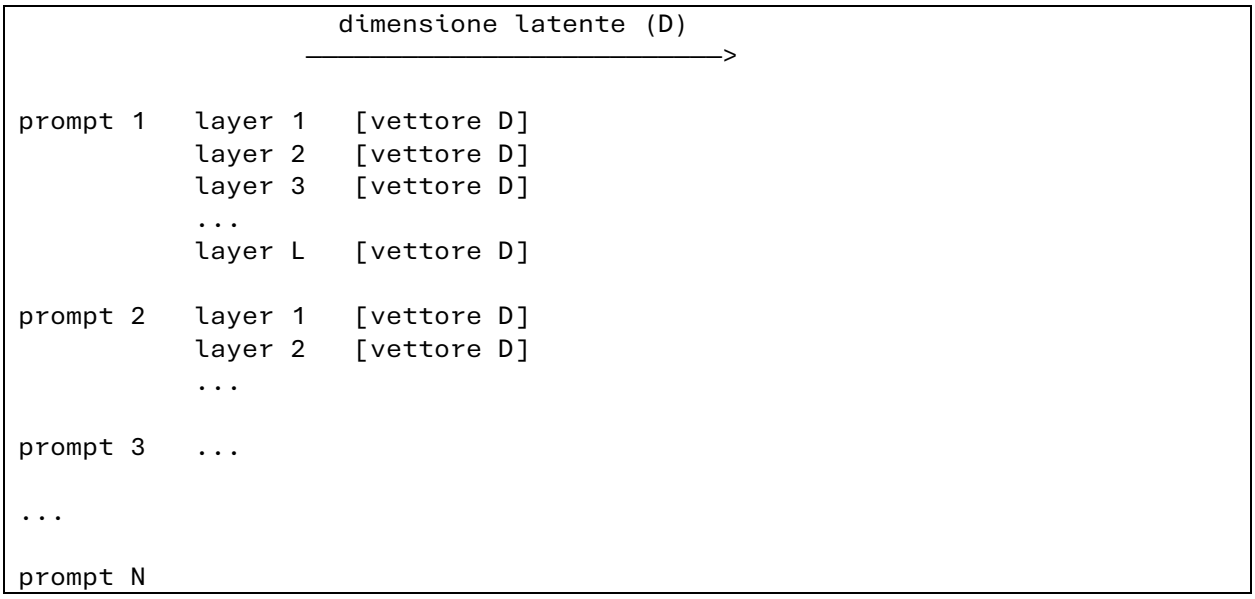
```
prompt → layer 1 → vettore
       → layer 2 → vettore
       → layer 3 → vettore
       ...
       → layer L → vettore
```

Noi registriamo **solo il vettore dell'ultimo token**, perché contiene il riassunto dello stato del modello in quel punto.

Quali dati otteniamo

Essendo in possesso del numero di prompt, del numero di layer, e delle dimensioni latenti, otteniamo una struttura di dati tridimensionale.

Possiamo immaginarla come un **blocco di dati**.



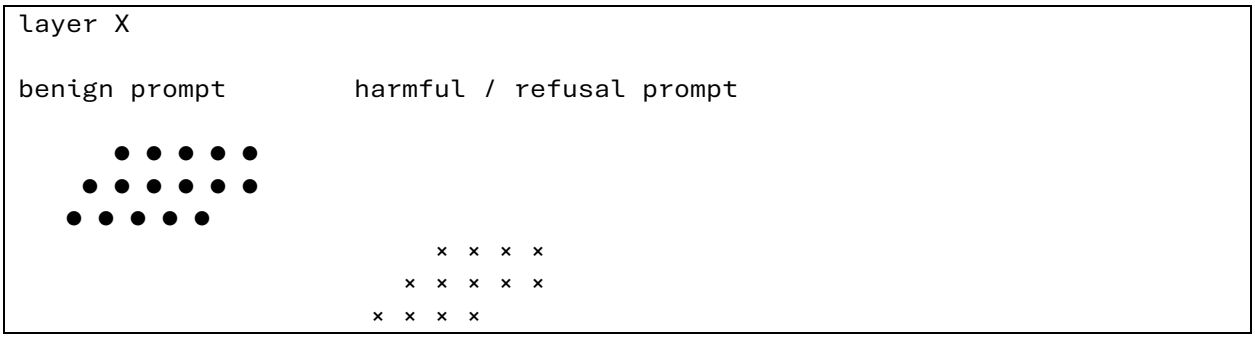
Otteniamo quindi una **matrice delle attivazioni H** dove:

```
dimensioni(H) = numero_prompt × numero_layer × dimensione_latente
```

Come gia' anticipato, ogni prompt viene eseguito una sola volta nel modello e viene registrato lo stato latente dell'ultimo token per tutti i layer del trasformatore. Le attivazioni vengono salvate in cache e utilizzate per tutte le analisi successive senza ulteriori esecuzioni del modello.

Intuizione geometrica

Ogni vettore può essere visto come **un punto in questo spazio**. Quindi per ogni layer otteniamo una nuvola di punti:



Questo è esattamente lo spazio dove poi viene misurato **quanto i due gruppi sono separati** e dove applichiamo una **direzione di steering.**

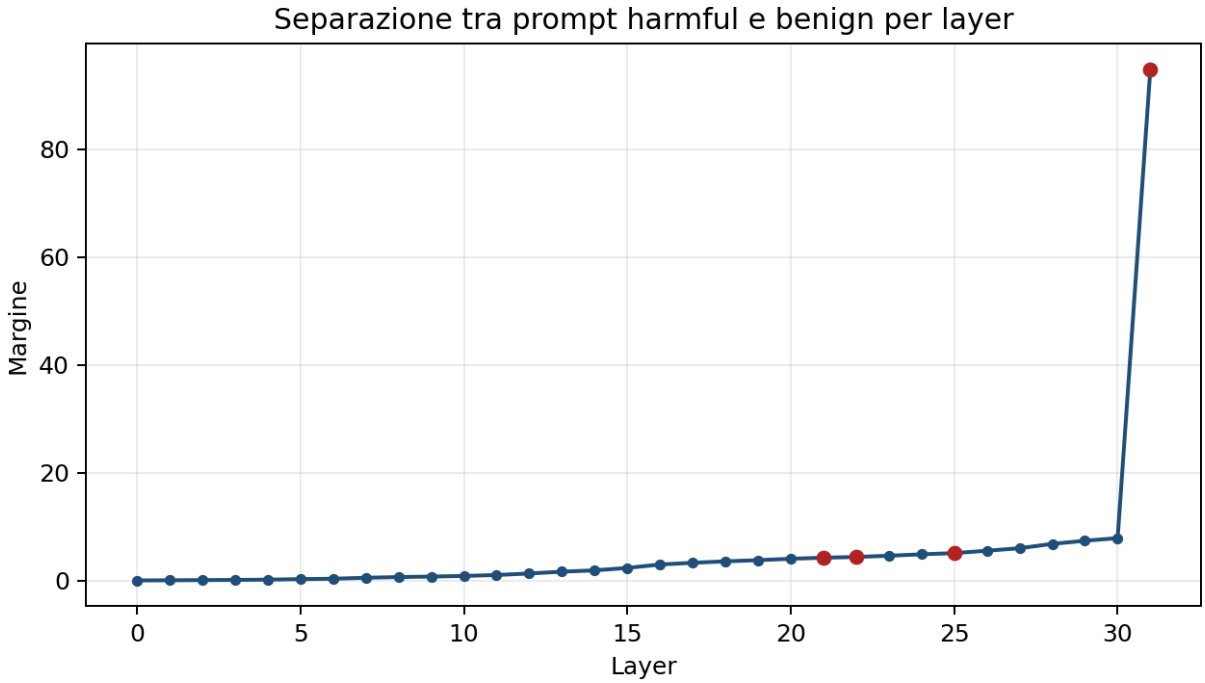
Analisi della separazione tra classi

Prima di applicare lo steering, analizziamo quanto i prompt *harmful* e *benign* risultino separati nei vari layer del modello e per ogni layer calcoliamo una misura chiamata **margin** (margine di separazione), definita come:

```
margin_layer = media(proiez_att_harmful) - media(proiez_att_benign)
```

Per ogni layer proiettiamo le attivazioni dei prompt *harmful* e *benign* lungo una direzione di contrasto definita dalla differenza tra le loro medie. Il margine di separazione del layer è definito come la differenza tra la media delle proiezioni *harmful* e la media delle proiezioni *benign*. Questo valore misura quanto le due distribuzioni risultino separate nello spazio.

Il risultato di questa analisi è mostrato nella figura a seguire, che riporta la separazione tra prompt *harmful* e *benign* per ciascun layer per il modello **Mistral-7B-Instruct-v0.2**.

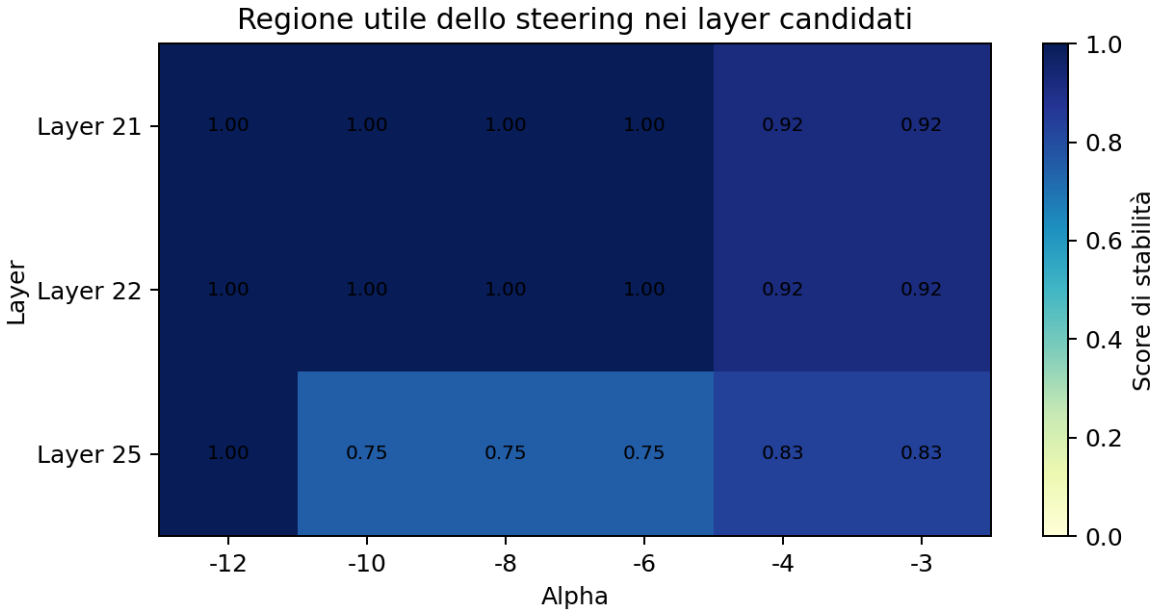


Si osserva che la separazione cresce progressivamente nei layer profondi del modello.

La scansione del margine di separazione individua il **layer 31**, cioè l'ultimo layer del modello (con numerazione a partire da 0), come massimo geometrico globale. Tuttavia, questo ranking non determina da solo il layer finale di steering.

Nella fase successiva abbiamo eseguito *sweep* (esplorazione dei parametri) controllati su layer candidati per verificare il comportamento in generazione. In questa fase i **layer 21, 22** e **25** sono emersi come i più promettenti dal punto di vista operativo, con **21** e **22** come candidati finali.

Il grafico a seguire (*heatmap*) presenta informazioni preziose riguardo ai risultati dell'analisi. Sull'asse delle x sono rappresentati i valori di **alpha**, mentre sull'asse delle y si trovano i **layer candidati**, identificati dai numeri 21, 22 e 25.



Ogni cella all'interno di questo grafico contiene uno **score di stabilità**, un indicatore che riflette le prestazioni del sistema. Lo score viene assegnato da un altro modello pipiccolo (*phi3:mini*), usato per la valutazione, che analizza automaticamente le risposte generate. In particolare, lo score viene considerato alto quando tre condizioni chiave si verificano:

1. Il valore di **harmful_refusal** è basso, suggerendo che il sistema è meno incline a rifiuti dannosi.
2. Il valore di **benign_refusal** è basso, il che implica che ci sono pochi rifiuti non dannosi.
3. Il valore di **degeneration** è basso, segnalando che ci sono minime perdite di performance.

In sintesi, le celle situate più in alto nel grafico indicano le **regioni dove lo steering funziona meglio**, suggerendo una maggiore stabilità e utilità. Al contrario, le celle più basse evidenziano le **regioni meno stabili o meno efficaci**, dove le prestazioni risultano inferiori. La heatmap offre quindi una rappresentazione immediata del comportamento del modello in relazione ai diversi layer candidati e ai valori di **alpha**. Nell'analisi ci concentriamo sui valori negativi di *alpha*, poiché gli sweep preliminari con valori positivi non avevano prodotto configurazioni stabili.

Direzione di steering

Dopo aver identificato i layer in cui la separazione tra prompt *harmful* e *benign* è più evidente, il passo successivo consiste nel determinare la **direzione di steering**.

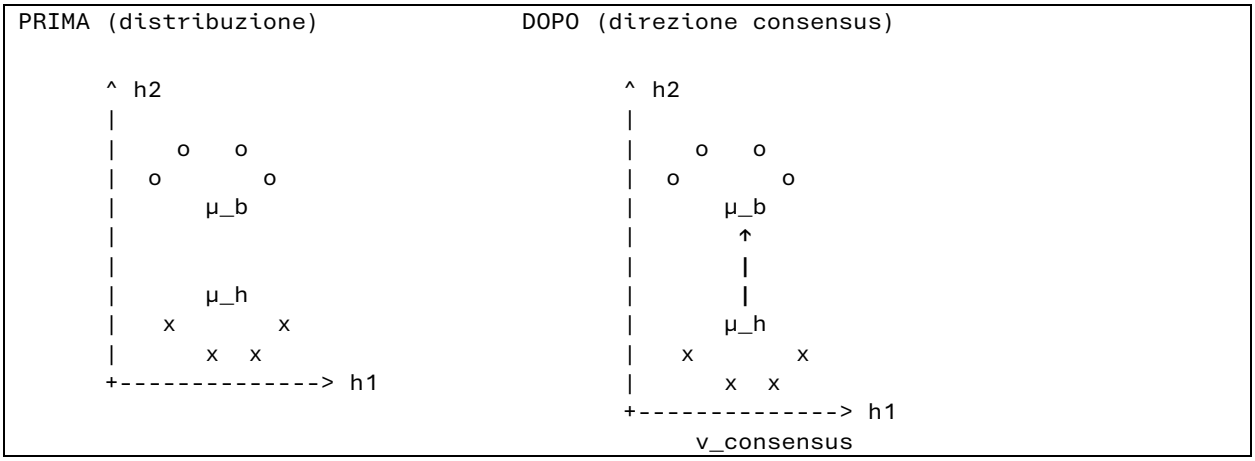
L'idea è che, se i prompt *harmful* e *benign* occupano regioni diverse, allora dovrebbe esistere una direzione che collega una regione all'altra. Muovendosi lungo questa direzione è possibile spostare leggermente la rappresentazione interna di un prompt e modificare il comportamento del modello.

Di seguito riportiamo tre modi diversi di leggere **la geometria dello stesso spazio latente**. In spazi da migliaia di dimensioni questi tre metodi spesso divergono parecchio.

Metodo	**Idea geometrica**
Consensus	direzione tra le medie delle due classi
RFM	direzione perpendicolare alla frontiera tra le classi
SVD	asse principale della nuvola di attivazioni

Direzione di steering nel metodo Consensus vector

Nel metodo **Consensus**, la direzione viene stimata nel modo più semplice possibile: si calcola la differenza tra le attivazioni medie dei prompt *harmful* e quelle dei prompt *benign*.



Questo vettore unisce i centri delle due distribuzioni nello spazio latente e definisce la direzione lungo cui viene applicato lo steering.

Direzione di steering con Recursive Feature Machines (RFM)

Il metodo **Recursive Feature Machines (RFM)**[2] utilizza un approccio più generale.

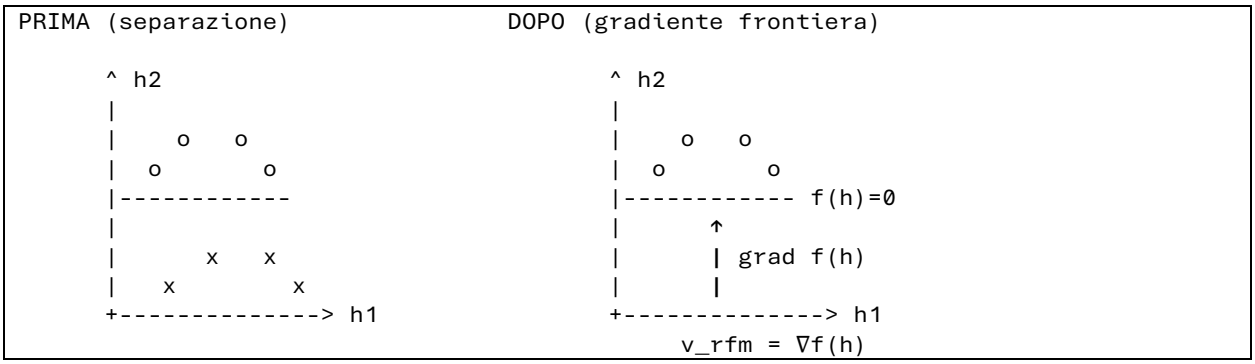
Invece di limitarsi alla differenza tra le due medie, RFM cerca una direzione che tenga conto **della struttura locale delle rappresentazioni nello spazio**.

Il procedimento parte comunque da una stima iniziale simile al consensus vector, ma applica un processo iterativo che raffina la direzione considerando come i punti *harmful* e *benign* sono distribuiti nello spazio latente.

RFM apprende prima una **funzione di separazione** tra le due classi `f(h)` dove **h** è il vettore di attivazione latente.

La frontiera tra le due regioni è definita dal punto in cui la funzione vale zero: `f(h) = 0`.

Dalla funzione f(h) possiamo osservare **come cambia nello spazio** e la direzione in cui cambia più velocemente. Questa direzione è il gradiente, che è perpendicolare alla superficie definita da `f(h) = 0`.



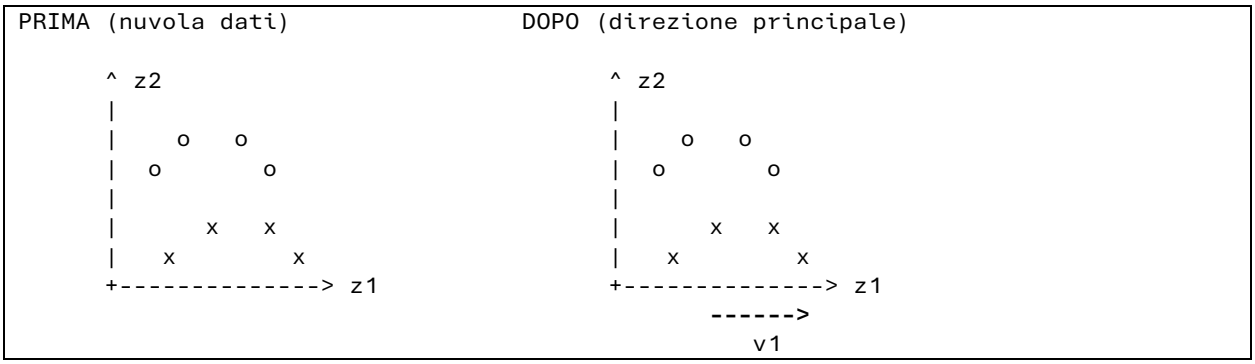
Immaginiamo una collina, dove la funzione f(h) rappresenta l'altezza del terreno. La linea f(h) = 0 segna il livello del mare. Il gradiente indica la direzione in cui la salita è più ripida. Se si seguono queste indicazioni, si sale rapidamente sopra il livello del mare, attraversando il confine tra regioni pianeggianti e quelle elevate. Questo movimento simboleggia come le variazioni di altitudine influenzino il percorso, proprio come accade nella vita reale.

[2] Toward universal steering and monitoring of AI models (febbraio 2026)
https://arxiv.org/pdf/2502.03708

Il risultato finale è comunque una direzione latente **v_rfm** che può essere utilizzata per steering.

Direzione di steering con SVD

Nel metodo basato su **SVD** (Singular Value Decomposition)[3], la direzione di steering viene stimata analizzando la struttura statistica delle attivazioni nello spazio latente.



Immaginiamo lo spazio latente del modello come una stanza piena di nebbia, dove i punti rappresentano le attivazioni prodotte dai prompt *harmful* e *benign*. Prima di analizzare questa distribuzione applichiamo una trasformazione di **whitening**. Questa operazione serve a rendere lo spazio più regolare.

Nelle attivazioni di un modello alcune dimensioni possono avere valori molto più grandi di altre, oppure muoversi insieme perché sono correlate. Questo crea distorsioni nello spazio. Alcune direzioni, ad esempio, sembrano importanti solo perché hanno numeri più grandi.

Il whitening corregge questo problema. Ridimensiona le dimensioni e rimuove le correlazioni tra di esse, in modo che tutte contribuiscano allo stesso modo. Dopo questa trasformazione lo spazio diventa più uniforme e le direzioni che emergono dall'analisi riflettono meglio la vera struttura dei dati, non semplici differenze di scala. In questo modo nessuna direzione risulta dominante solo per ragioni numeriche.

Una volta normalizzato lo spazio, applichiamo la **Singular Value Decomposition** (SVD) alla matrice X che contiene le attivazioni *whitened*. Questa decomposizione analizza come

[3] Keep Calm and Avoid Harmful Content: Concept Alignment and Latent Manipulation Towards Safer Answers

https://arxiv.org/html/2510.12672v1

i punti sono distribuiti nello spazio latente e identifica le direzioni principali lungo cui i dati variano di più.

Questa direzione viene quindi utilizzata come vettore di steering: `v_svd = v1.` La nuvola è allungata lungo l'asse z1.

Muoversi lungo questa direzione significa spostare le rappresentazioni latenti lungo l'asse statistico che meglio distingue i comportamenti del modello.

Iniezione di steering

Durante la generazione, il modello costruisce la risposta passando attraverso una sequenza di rappresentazioni interne, cioè vettori nascosti che descrivono lo stato del ragionamento in ciascun layer. L'idea dello steering è intervenire leggermente su una di queste rappresentazioni interne, spostandola in una direzione scelta in anticipo.

Nel nostro caso, questa direzione è il vettore di steering. Possiamo immaginarlo come una freccia nello spazio latente: invece di lasciare il punto esattamente dove il modello lo avrebbe portato da solo, lo spostiamo lungo questa freccia.

La modifica segue una regola molto semplice:

```
nuova_attivazione = attivazione_originale + alpha × vettore_di_steering
```

Il parametro **alpha** controlla quanto forte sia lo spostamento. Se alpha è piccolo, l'intervento è lieve. Se alpha è grande, il cambiamento è più marcato.

In questo modo non cambiamo direttamente il testo finale, ma modifichiamo leggermente il percorso interno con cui il modello arriva alla risposta. L'effetto osservabile è che, in certe configurazioni, il modello riduce il *refusal* sui prompt *harmful* senza compromettere il comportamento sui prompt *benign*.

Metriche di valutazione

La valutazione non si basa solo sulla percentuale complessiva di rifiuto. Considerare soltanto quante volte il modello rifiuta sarebbe troppo riduttivo, perché non permetterebbe di distinguere tra un rifiuto netto, una risposta parziale o una generazione degradata.

Per questo motivo, per ogni configurazione di steering osserviamo diversi aspetti del comportamento del modello. In particolare distinguiamo:

- **refusal_net**: casi in cui il modello rifiuta in modo esplicito e diretto la richiesta;
- **refusal_soft**: casi in cui il modello non esegue pienamente la richiesta, ma risponde in modo attenuato o sostitutivo, ad esempio spostandosi su spiegazioni generali o alternative sicure;
- **partial_useful**: casi in cui la risposta contiene comunque elementi utili o parzialmente utilizzabili rispetto alla richiesta;
- **degeneration**: casi in cui il testo perde qualità, ad esempio diventando ripetitivo, incoerente o frammentario;
- **recontextualization**: casi in cui il modello sposta il contenuto in un altro contesto, ad esempio trasformando la richiesta in una spiegazione teorica, educativa o astratta.

A partire da queste osservazioni ricaviamo due indicatori sintetici.

Il primo è **harmful_answer_rate**, che misura quanto spesso il modello smette di rifiutare i prompt harmful e produce effettivamente una risposta. Il secondo è **benign_stability**, che misura quanto il comportamento del modello resta stabile sui prompt *benign*, cioè quanto evita di introdurre rifiuti o distorsioni dove non dovrebbero esserci.

Una configurazione di steering viene considerata utile solo se soddisfa contemporaneamente tre condizioni:

1. aumenta la capacità del modello di rispondere ai prompt *harmful* del benchmark;
2. non introduce rifiuti sui prompt *benign*;
3. non genera degrado o instabilita'.

In questo modo la valutazione non misura solo "quanto il modello cambia", ma soprattutto se cambia nella direzione desiderata senza compromettere la qualità generale del comportamento.

Protocollo sperimentale

Tutti gli esperimenti sono stati condotti utilizzando il modello **Mistral-7B-Instruct-v0.2**.

L'intero studio segue un protocollo progressivo, in cui ogni passaggio prepara il successivo. L'idea di fondo è evitare scelte arbitrarie e arrivare alla configurazione finale attraverso una sequenza di verifiche guidate dai dati.

Il workflow sperimentale è composto da sette fasi.

1. Estrazione e caching delle attivazioni

Nella prima fase estraiamo le attivazioni interne del modello e le salviamo in cache. Questo ci permette di lavorare sempre sugli stessi dati latenti senza dover rieseguire ogni volta nuove *forward pass* del modello.

2. Analisi della separazione tra classi nei layer

Nella seconda fase analizziamo quanto bene i diversi layer separino i prompt *harmful* dai prompt *benign*. In altre parole, osserviamo in quali punti del modello emerge con maggiore chiarezza la differenza tra le due classi.

3. Selezione dei layer candidati

Nella terza fase, sulla base di questa analisi, selezioniamo un piccolo insieme di layer candidati. Questa selezione non viene fatta a priori, ma deriva direttamente dalle evidenze emerse nella fase precedente.

4. Sweep dei valori di alpha

Nella quarta fase eseguiamo sweep controllati sui valori di alpha, cioè sull'intensità dello steering. Per ogni layer candidato testiamo diversi valori e osserviamo in che modo cambia il comportamento del modello. La valutazione delle risposte è affidata al modello *phi3:mini*.

5. Valutazione su benchmark bilanciati

Nella quinta fase valutiamo ogni configurazione su benchmark bilanciati, composti da prompt *harmful* e prompt *benign*. Questo passaggio è fondamentale perché permette di verificare contemporaneamente due aspetti: la capacità di ridurre il *refusal* sui prompt *harmful* e la capacità di non compromettere il comportamento sui prompt *benign*.

6. Confronto tra metodi

Nella sesta fase confrontiamo il metodo RFM con altre direzioni di steering, in particolare Consensus e SVD, in modo da capire se il comportamento osservato sia specifico di RFM oppure condiviso anche da metodi più semplici o alternativi.

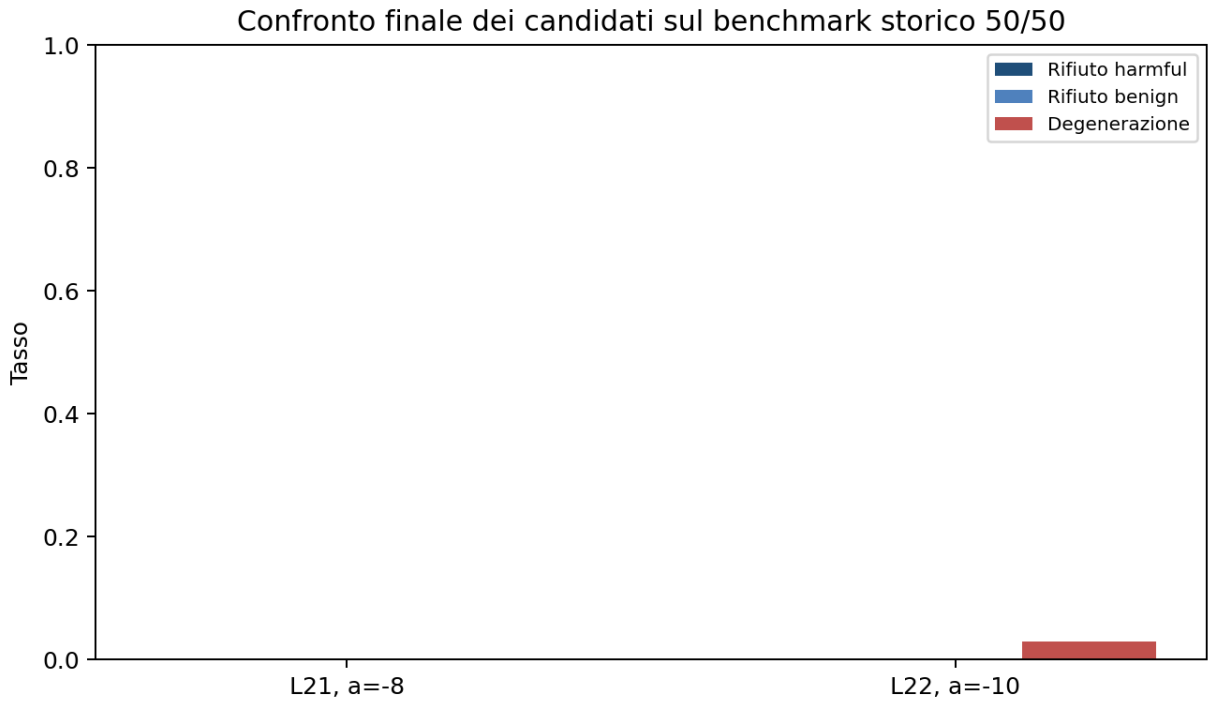
7. Replica su un secondo corpus

Infine, nella settima fase, replichiamo il benchmark finale su un secondo corpus di prompt, distinto da quello storico usato nel progetto. Questo serve a verificare se il risultato ottenuto sia stabile anche fuori dal dataset iniziale.

Risultati

Identificazione della configurazione di steering più stabile

La shortlist finale si concentra quindi sui layer 21, 22 e 25, che hanno mostrato comportamento più promettente negli sweep di steering. Il benchmark finale sul dataset storico confronta due candidati 21 e 22.



I due candidati sono equivalenti sui refusal. Il layer 22 introduce un 3% di degenerazione. Quando due configurazioni fanno lo stesso lavoro sul comportamento target, prevale quella con meno effetti collaterali. In questo caso **layer 21** con **alpha = -8**.

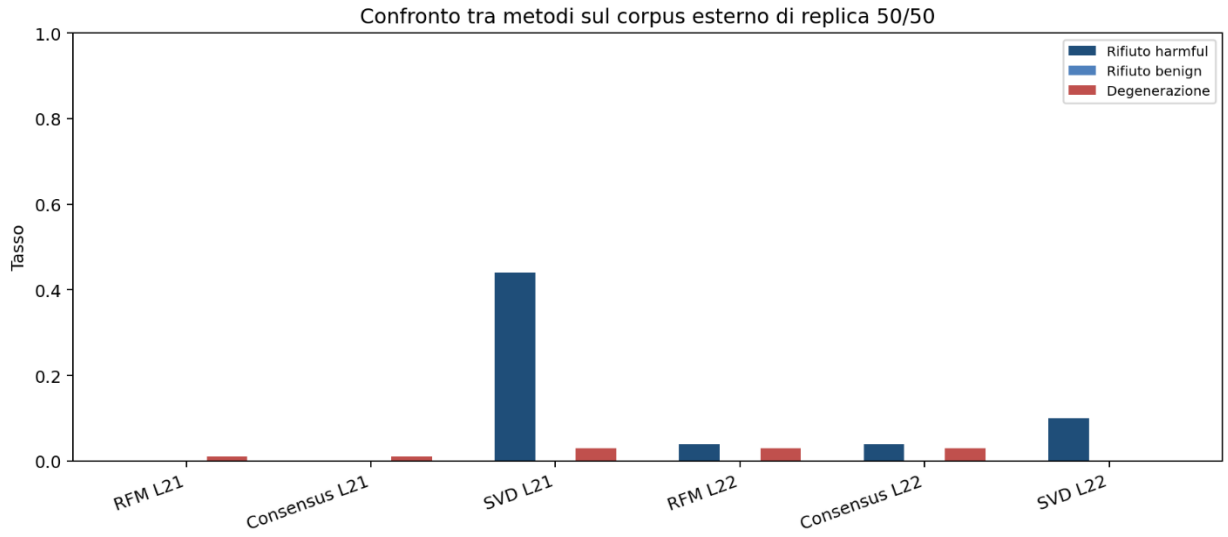
Tabella riassuntiva dei risultati del benchmark finale.

Layer	Alpha	Refusal	Benign Refusal	Degeneration
21	**-8**	0	0	0
22	**-10**	0	0	0.03

Per questo motivo selezioniamo **layer 21, alpha -8** come configurazione finale.

Replica su un corpus esterno

Per verificare che il risultato non dipenda dal dataset storico interno, ripetiamo il confronto su un secondo benchmark 50/50 costruito da un corpus esterno[4] di replica.



Di seguito la tabella riepilogativa che mette a confronto layer, metodo e metriche.

Layer	Alpha	Metodo	Harmful Refusal	Benign Refusal	Degeneration
21	**-8**	**RFM**	**0**	**0**	**0.01**
21	**-8**	**Consensus**	**0**	**0**	**0.01**
21	-8	Whitened SVD	0.44	0	0.03
22	-10	RFM	0.04	0	0.03
22	-10	Consensus	0.04	0	0.03
22	-10	Whitened SVD	0.1	0	0

[4] Dataset HuggingFace
https://huggingface.co/datasets/simonycl/aya-23-8B_advbench_jailbreak

Consensus e RFM stanno probabilmente trovando **quasi la stessa direzione**. Questo succede spesso quando la separazione tra classi è abbastanza lineare. In quel caso la differenza tra media e frontiera diventa minima.

SVD invece guarda **la varianza globale dei dati**, non la separazione tra classi. Se la direzione di massima varianza non coincide con la direzione, lo steering spinge il modello nella direzione sbagliata e il modello torna a rifiutare molte query *harmful*.

Confronto geometrico tra metodi

Misuriamo anche la similarità coseno tra le direzioni apprese. Nei layer principali, **RFM** e **Consensus** risultano quasi paralleli:

- layer 21: `cos(theta) = 0.999775`
- layer 22: `cos(theta) = 0.999698`

Questo dato geometrico è coerente con il comportamento osservato nei benchmark. Nel regime testato, **RFM** non definisce una direzione di steering sostanzialmente diversa dal **Consensus**. Al contrario, **SVD** è geometricamente diversa, ma questa differenza non si traduce in migliori prestazioni sul target principale.

Conclusione

In questo lavoro abbiamo analizzato il comportamento di *refusal* dei large language model da una prospettiva geometrica, studiando come le rappresentazioni interne dei prompt *harmful* e *benign* si distribuiscono nello spazio latente del modello.

Attraverso una pipeline sperimentale progressiva abbiamo identificato una configurazione di steering stabile nel modello **Mistral-7B-Instruct-v0.2** (layer 21, $\alpha = -8$), capace di ridurre completamente i *refusal* sui prompt *harmful* del benchmark senza introdurre rifiuti sui prompt *benign* e senza generare degenerazione significativa.

Il risultato si replica sia sul dataset storico del progetto sia su un corpus esterno, suggerendo che il comportamento osservato non dipende da una singola collezione di prompt ma riflette una struttura più generale delle rappresentazioni del modello.

Il confronto tra metodi mostra inoltre che la direzione appresa tramite **Recursive Feature Machines** è quasi perfettamente allineata con quella ottenuta dalla semplice differenza tra le medie delle rappresentazioni (**Consensus vector**). La similarità coseno tra le due direzioni è prossima a 1 nei layer principali, indicando che nel regime analizzato la

separazione tra prompt *harmful* e *benign* è approssimativamente lineare nello spazio latente.

Questo risultato suggerisce che, almeno in questo modello e per questo tipo di comportamento, il *refusal* possa essere interpretato come una proprietà geometrica relativamente semplice delle rappresentazioni interne. In tali condizioni, metodi complessi di apprendimento della frontiera non producono vantaggi sostanziali rispetto a direzioni di steering lineari più semplici.

L'analisi mostra che il comportamento dei modelli può essere studiato e modificato direttamente nello spazio delle loro rappresentazioni interne, rendendo possibile una forma di controllo e verifica più trasparente rispetto a sistemi completamente opachi.

In questo contesto, l'utilizzo di modelli open-weight assume un ruolo particolarmente rilevante. A differenza dei servizi basati su API proprietarie, i modelli eseguibili in locale permettono di analizzare le rappresentazioni interne, applicare tecniche di steering e svolgere verifiche indipendenti senza trasferire dati sensibili a servizi esterni.

Questo aspetto è particolarmente importante per contesti istituzionali in cui la protezione dei dati e la possibilità di audit tecnico sono requisiti fondamentali. L'approccio presentato in questo lavoro suggerisce quindi una possibile direzione per lo sviluppo di sistemi di AI più controllabili, verificabili e compatibili con le esigenze di sicurezza della Pubblica Amministrazione.