Modulo 3: Accesso a conoscenza esterna tramite RAG

Alessandro Petruzzelli

Università degli Studi di Bari Aldo Moro

Per iniziare...

Vi dico due parole. Cosa vi viene in mente?

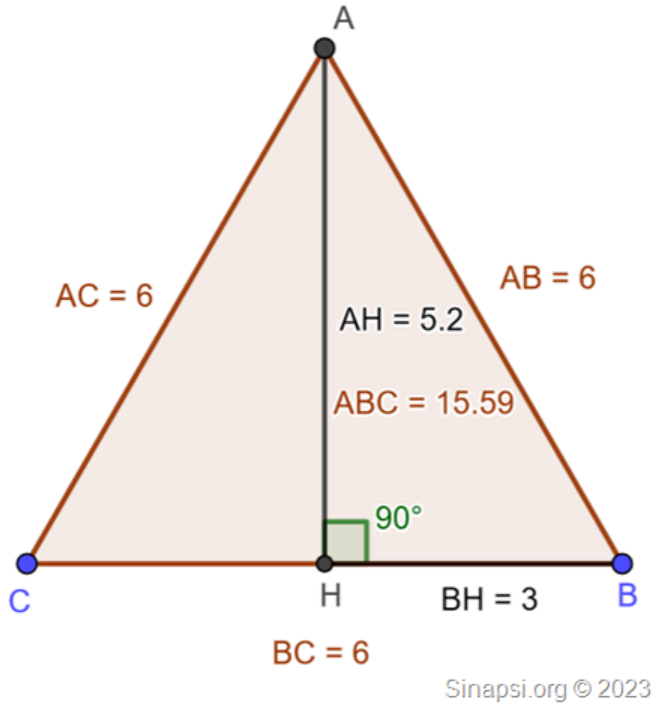
"Considerato" e "Triangolo"

(Prendetevi 10 secondi per pensare...)

Scenario A

> "L'area del triangolo, **considerato** il **triangolo** rettangolo..."

🧠 **Contesto**: Matematica / Geometria.



Scenario B

> "Il **triangolo** no, non l'avevo **considerato**..."
> 🎵

🧠 **Contesto**: Renato Zero (Musica Italiana).

La Lezione di Oggi

Senza **Contesto**, la stessa informazione è ambigua.

Gli LLM sono come noi:

- Se non diamo loro il contesto giusto, tirano a indovinare.

La Lezione di Oggi

Senza **Contesto**, la stessa informazione è ambigua.

Gli LLM sono come noi:

- Se non diamo loro il contesto giusto, tirano a indovinare.

Come fornire il contesto perfetto ad un LLM?

Agenda della Lezione

1. **Introduzione e Naïve RAG**
    - Limiti degli LLM
    - Workflow & Indexing Deep Dive
2. Chunking, Embeddings e Vector Databases
3. Valutazione e Strategie

Agenda della Lezione

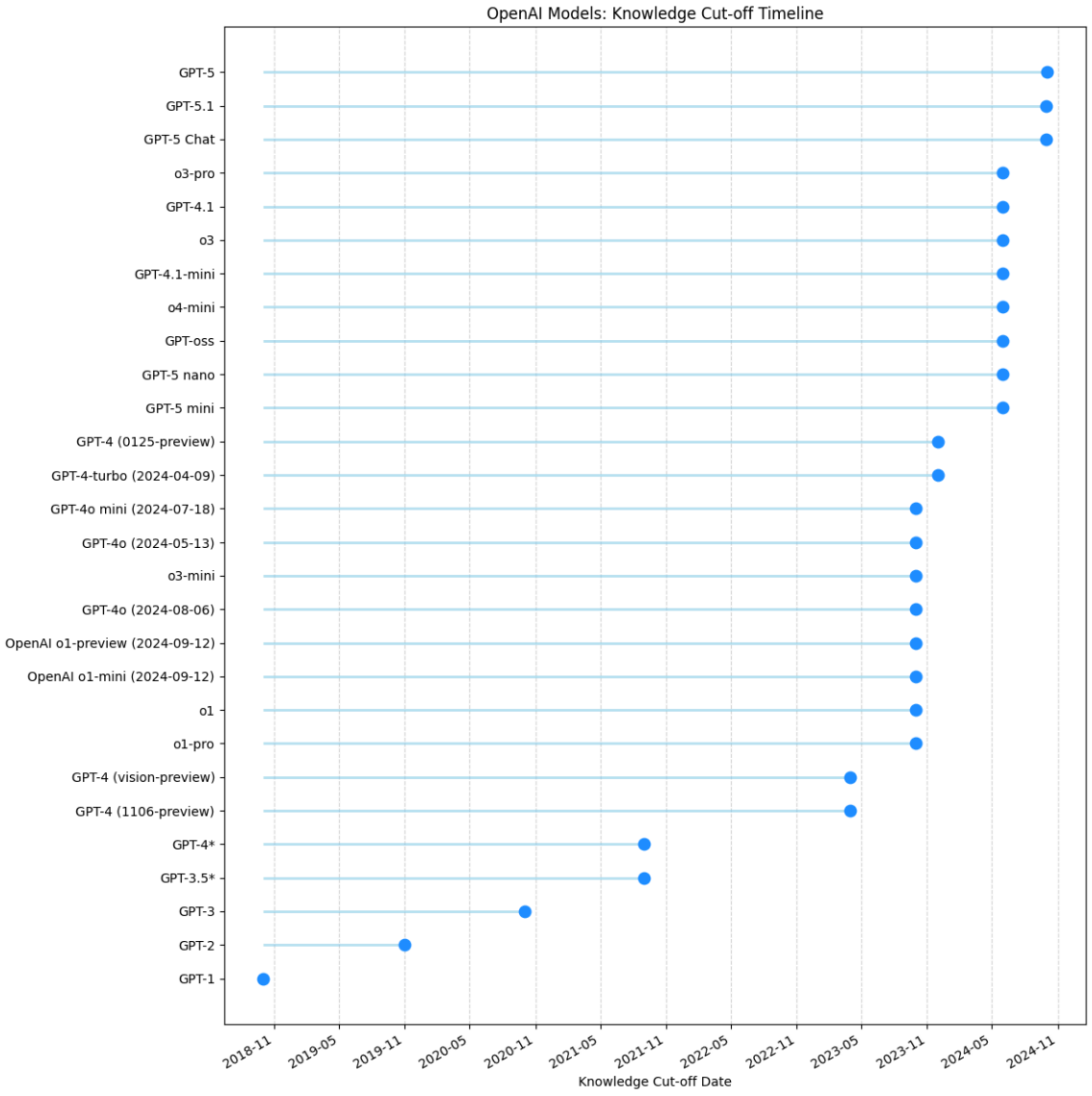
1. Introduzione e Naïve RAG
2. **Chunking, Embeddings e Vector Databases**
    - Strategie di Chunking Avanzate
    - Sparse (BM25) vs Dense (BERT)
    - Bi-Encoders vs Cross-Encoders
    - Tassonomia Vector DB & Algoritmi (HNSW)
3. Valutazione e Strategie

Agenda della Lezione

1. Introduzione e Naïve RAG
2. Chunking, Embeddings e Vector Databases
3. **Valutazione e Strategie**
    - Metriche di Retrieval (Recall@K, MRR)
    - Framework RAGAS
    - RAG vs Fine-tuning

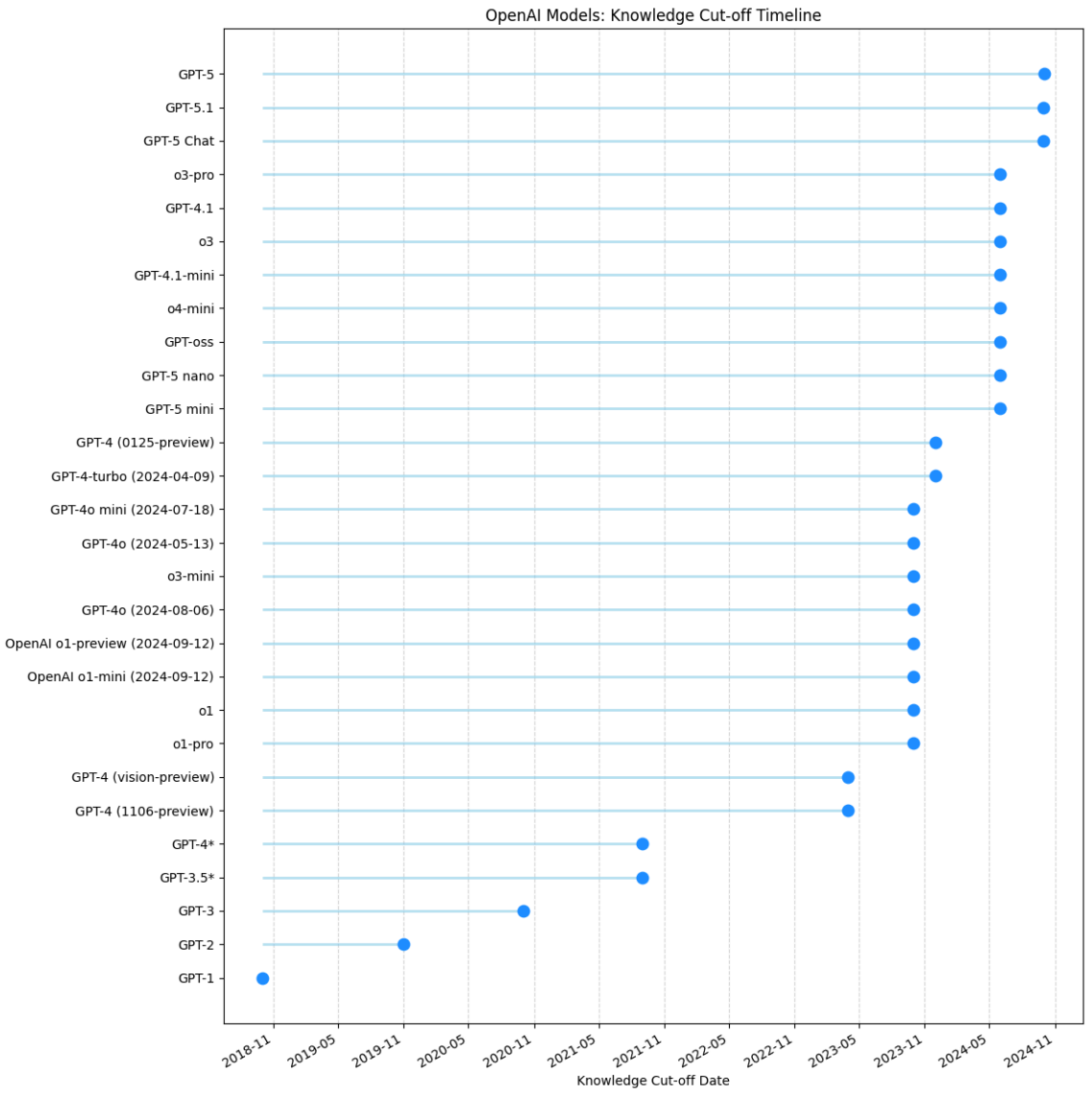
Modulo 1

Introduzione e Architettura Naïve RAG



Conoscenza "Cristallizzata"

Un LLM possiede una vasta conoscenza, ma questa è limitata al momento del suo training.

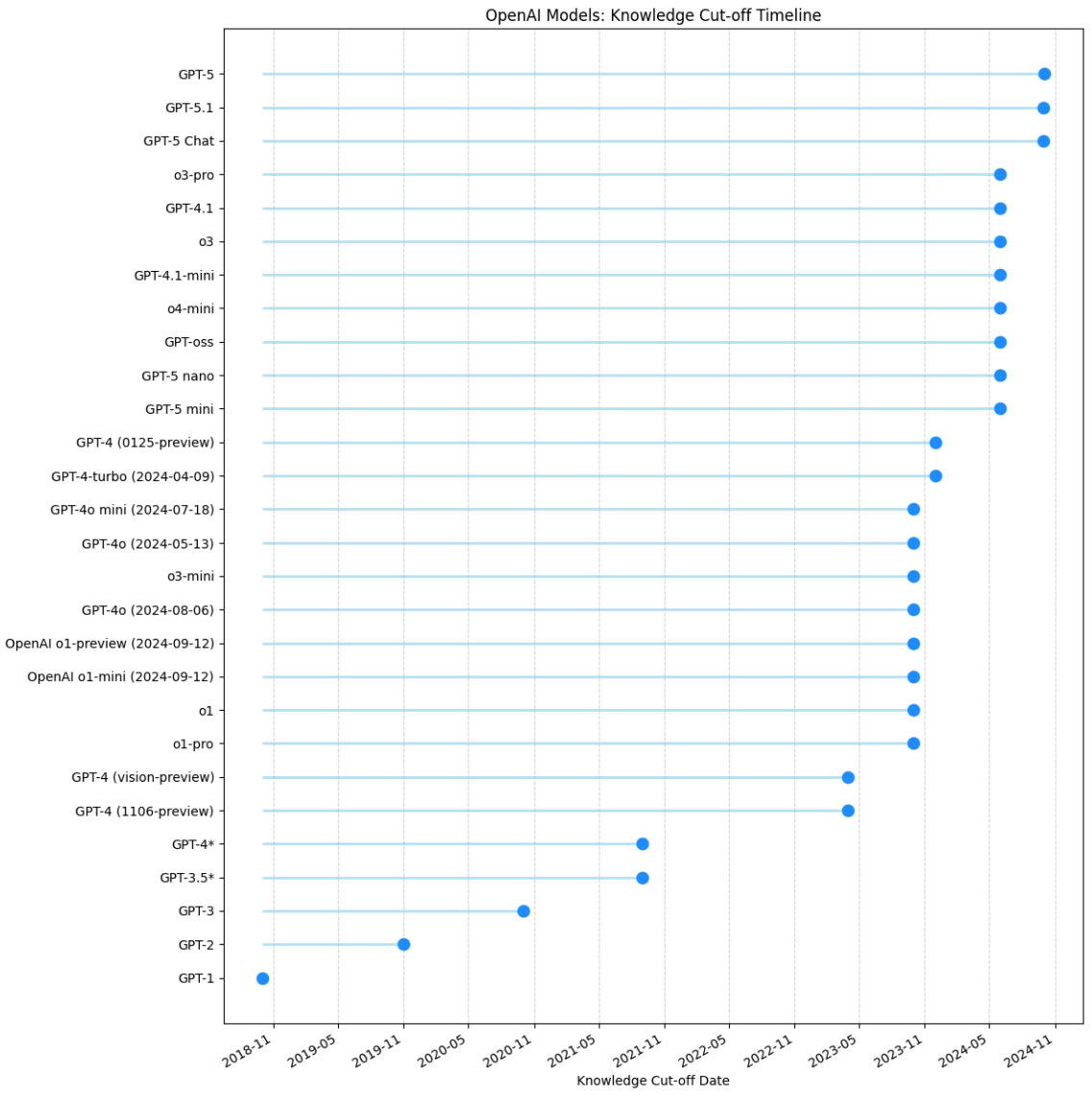


Conoscenza "Cristallizzata"

Un LLM possiede una vasta conoscenza, ma questa è limitata al momento del suo training.

- **Conoscenza Parametrica**:
  - È la conoscenza interna del modello, immagazzinata nei suoi pesi.
  - È statica e difficile da aggiornare.

SWAP researchgroup



Conoscenza "Cristallizzata"

Un LLM possiede una vasta conoscenza, ma questa è limitata al momento del suo training.

- **Conoscenza Parametrica**
- **Limitazioni**:
  - Training cut-off.
  - Costo elevato per il re-training continuo.

SWAP researchgroup

Il "Gap" della Conoscenza Aziendale

Un LLM (es. GPT-4 base) è un genio con **cultura generale**, ma un incompetente con i **dati aziendali** o **proprietari**.

- **Cosa sa**: Storia, Coding, Letteratura, Grammatica.
- **Cosa NON sa**:
  - "Quanto abbiamo fatturato ieri?"
  - "Cosa dice la policy HR aggiornata?"
  - "Dettagli del progetto Top-Secret X".

> **Risultato**: In azienda, un LLM senza dati proprietari è spesso inutile o pericoloso.

Il Problema degli LLM: Allucinazioni

Le allucinazioni sono un difetto intrinseco della natura probabilistica degli LLM.

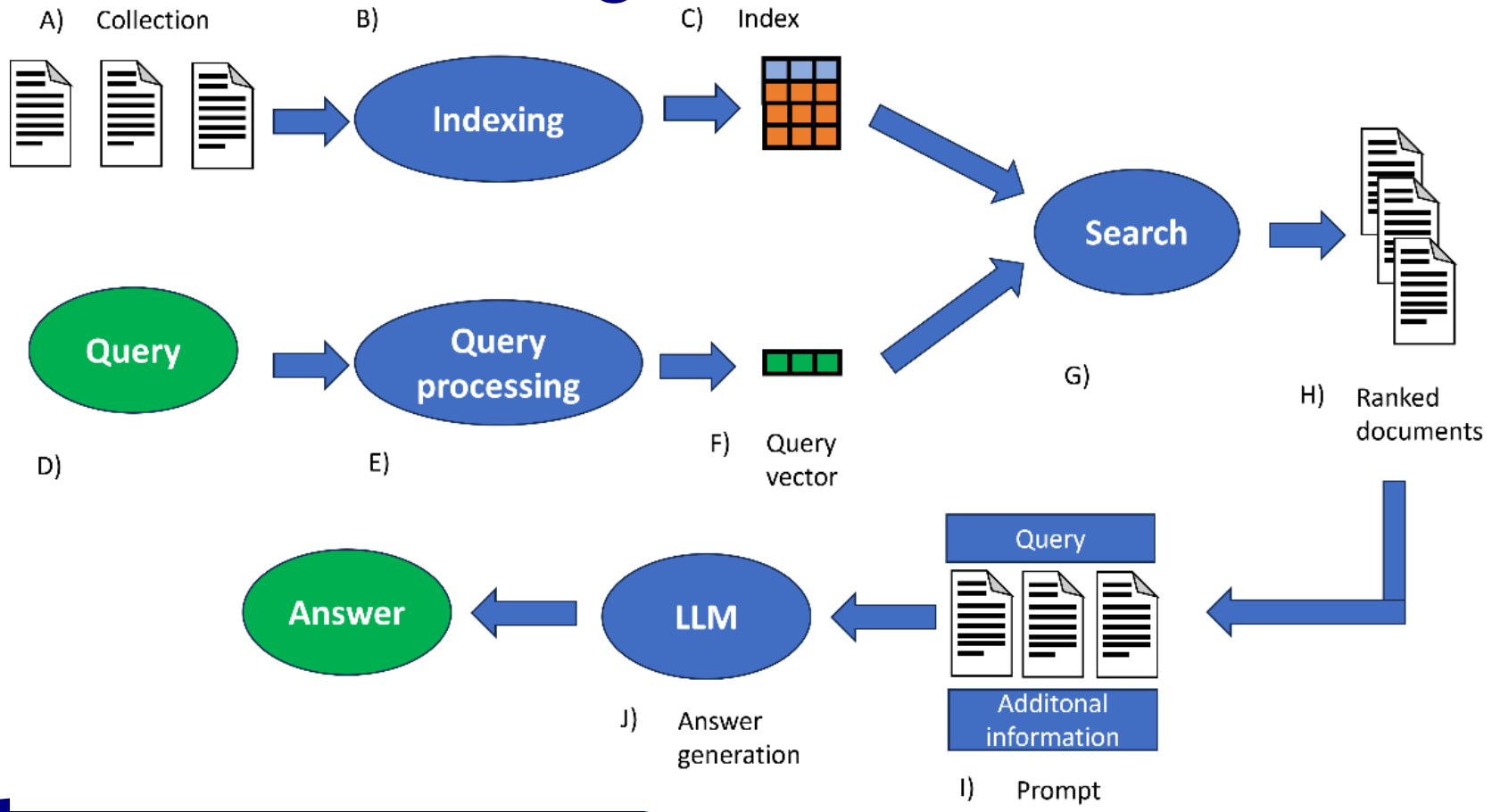
- **Generazione Probabilistica**:
  - Il modello predice il prossimo token basandosi su probabilità statistiche.
  - Favorisce risposte plausibili "grammaticalmente" piuttosto che "fattualmente".

Il Problema degli LLM: Allucinazioni

Perché avvengono le allucinazioni?

- **Mancanza di fonti**: Il modello inventa fatti per colmare lacune.
- **Overfitting/Bias**: Pattern ripetuti nel training set possono forzare risposte errate.
- **Temperature**: Alta "creatività" aumenta il rischio di errori fattuali.

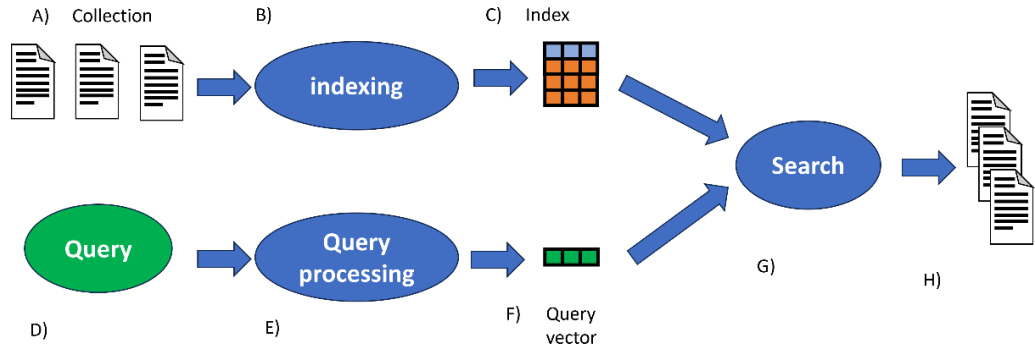
Soluzione: Retrieval-Augmented Generation



Retrieval-Augmented Generation

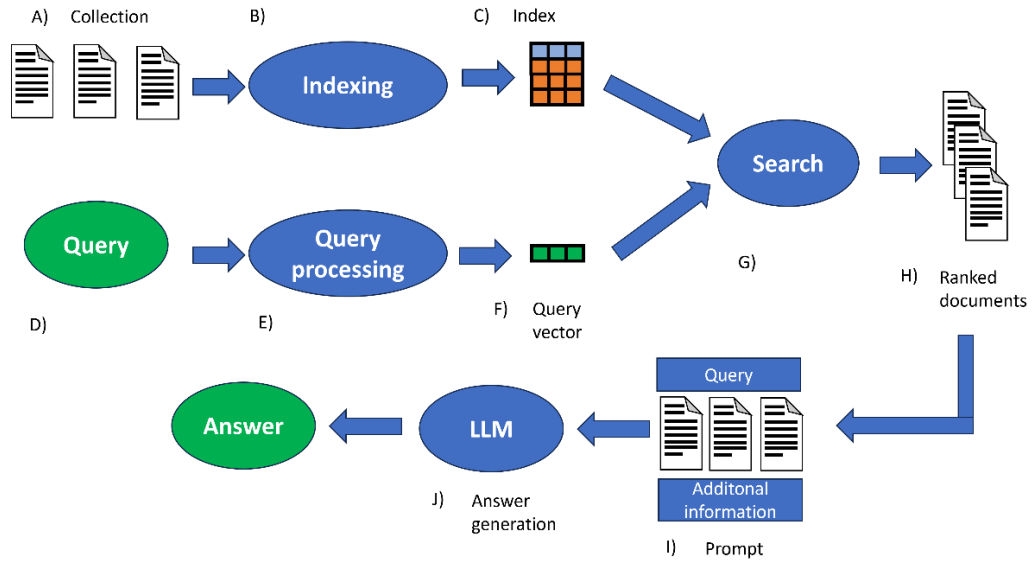
Il RAG unisce la potenza linguistica dell'LLM con una base di conoscenza esterna.

- **Memoria Parametrica**: L'LLM.
- **Memoria Non-Parametrica**: Vector DB.



Workflow Naïve RAG: Panoramica

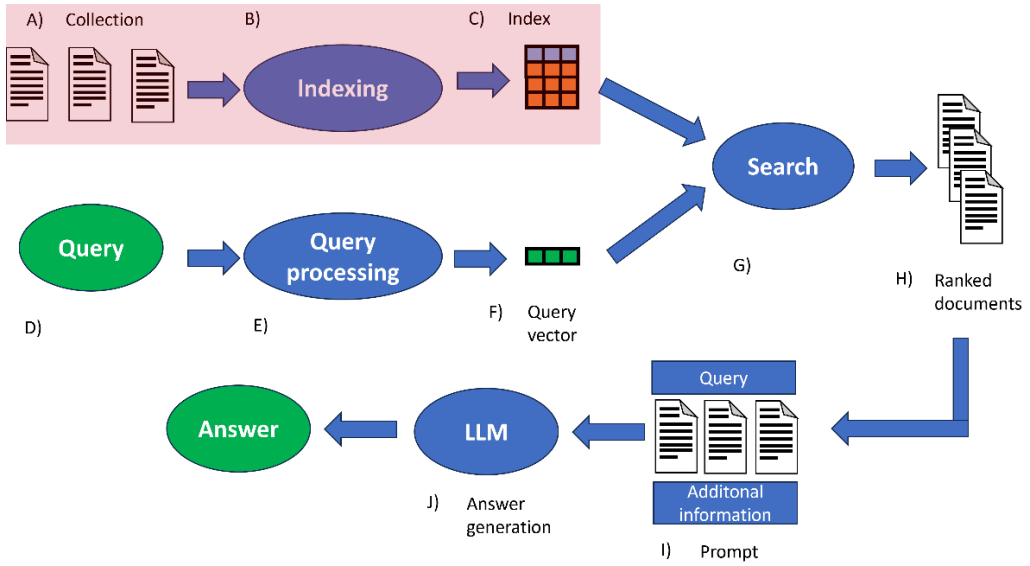
Analizziamo i tre pilastri fondamentali.



Workflow Naïve RAG: Panoramica

Analizziamo i tre pilastri fondamentali.

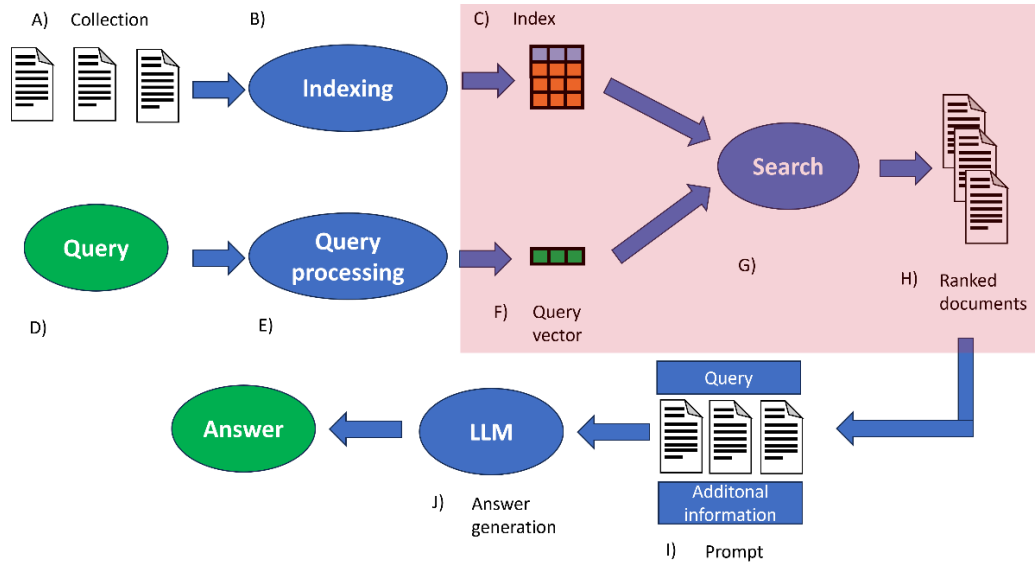
1. **Indexing**: Preparazione dei dati.



Workflow Naïve RAG: Panoramica

Analizziamo i tre pilastri fondamentali.

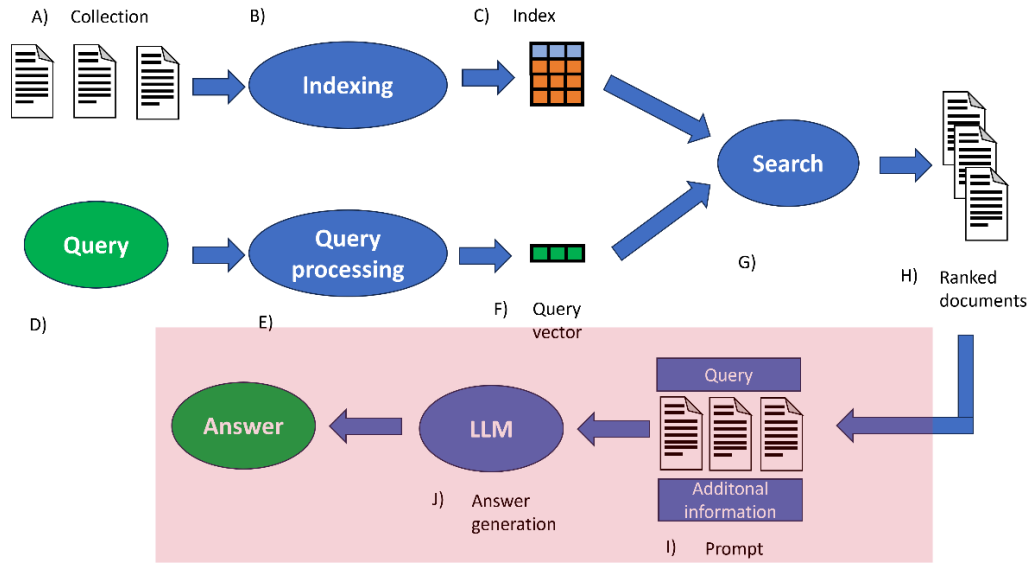
1. **Indexing**: Preparazione dei dati.
2. **Retrieval**: Recupero informazioni.

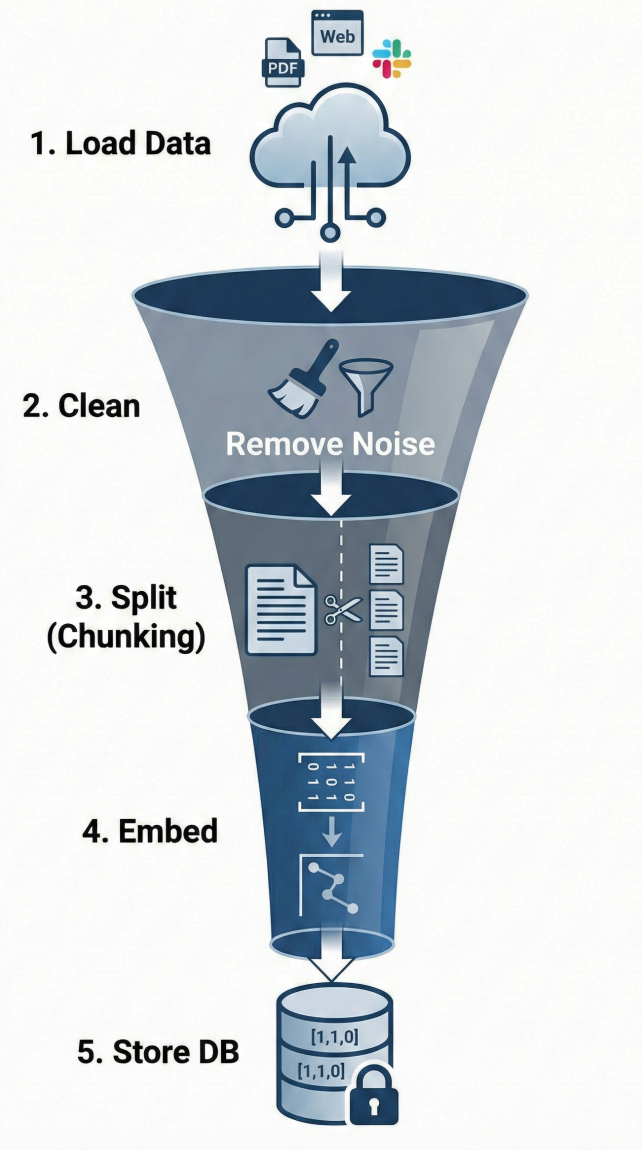


Workflow Naïve RAG: Panoramica

Analizziamo i tre pilastri fondamentali.

1. **Indexing**: Preparazione dei dati.
2. **Retrieval**: Recupero informazioni.
3. **Generation**: Sintesi della risposta.





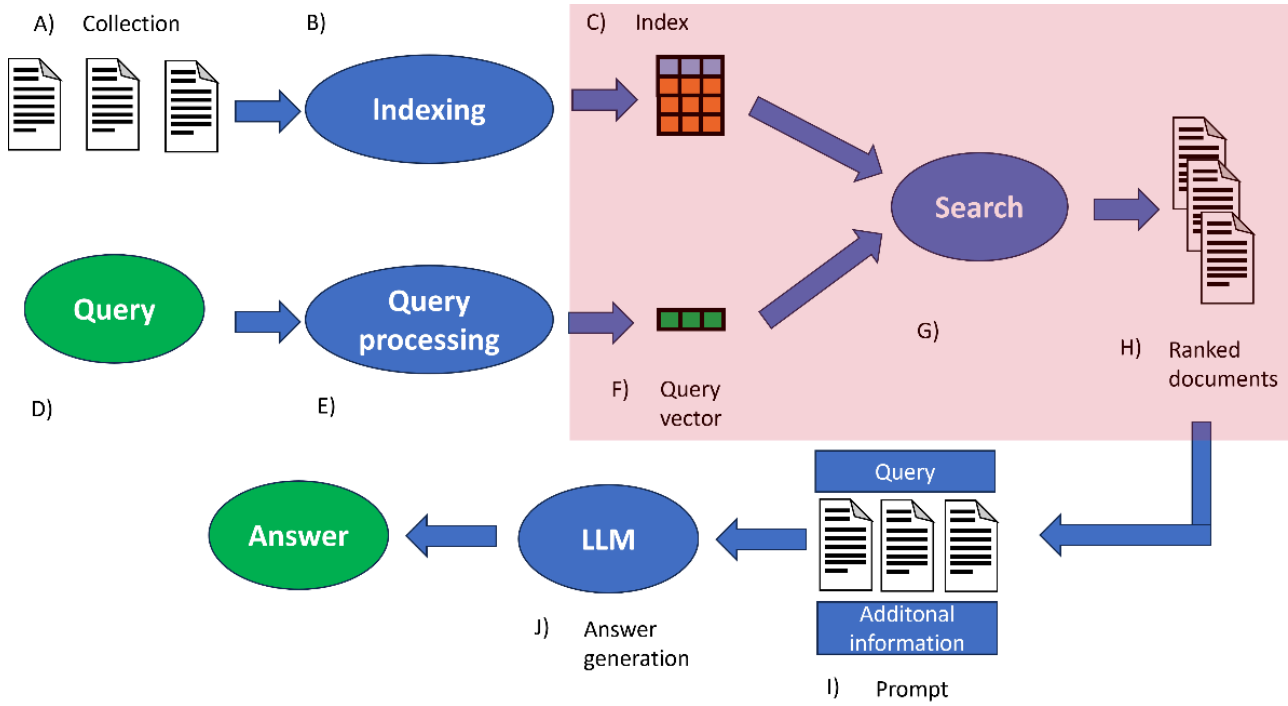
Pilastro 1: Indexing

Prima di poter cercare, dobbiamo costruire il nostro indice.

1. **Load**: Caricamento dati da sorgenti eterogenee.
2. **Clean**: Rimozione rumore.
3. **Split (Chunking)**: Suddivisione in unità logiche.
4. **Embed**: Conversione in vettori.
5. **Store**: Salvataggio nel DB.

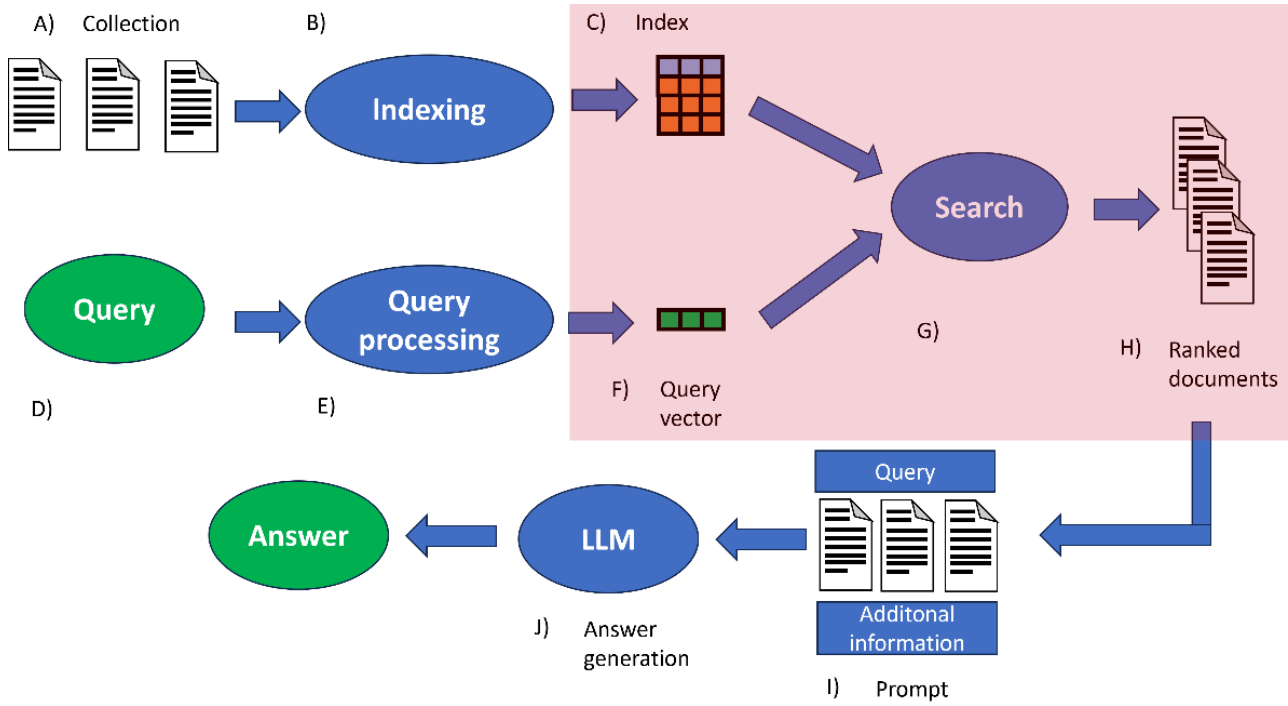
Pilastro 2: Retrieval

- La **Query dell'utente** viene convertita in un vettore (stesso modello di embedding).



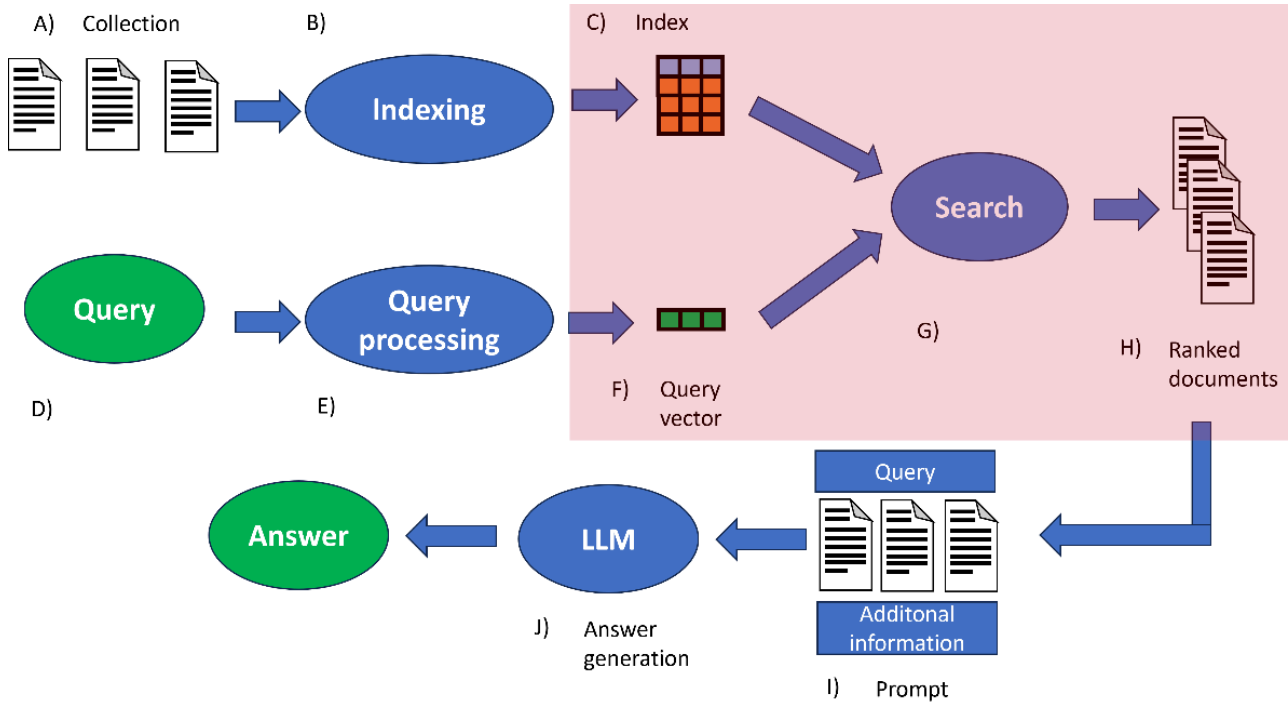
Pilastro 2: Retrieval

- La **Query dell'utente** viene convertita in un vettore (stesso modello di embedding).
- **Similarity Search**: Confronto matematico (es. Cosine Similarity) tra vettore query e vettori documenti.



Pilastro 2: Retrieval

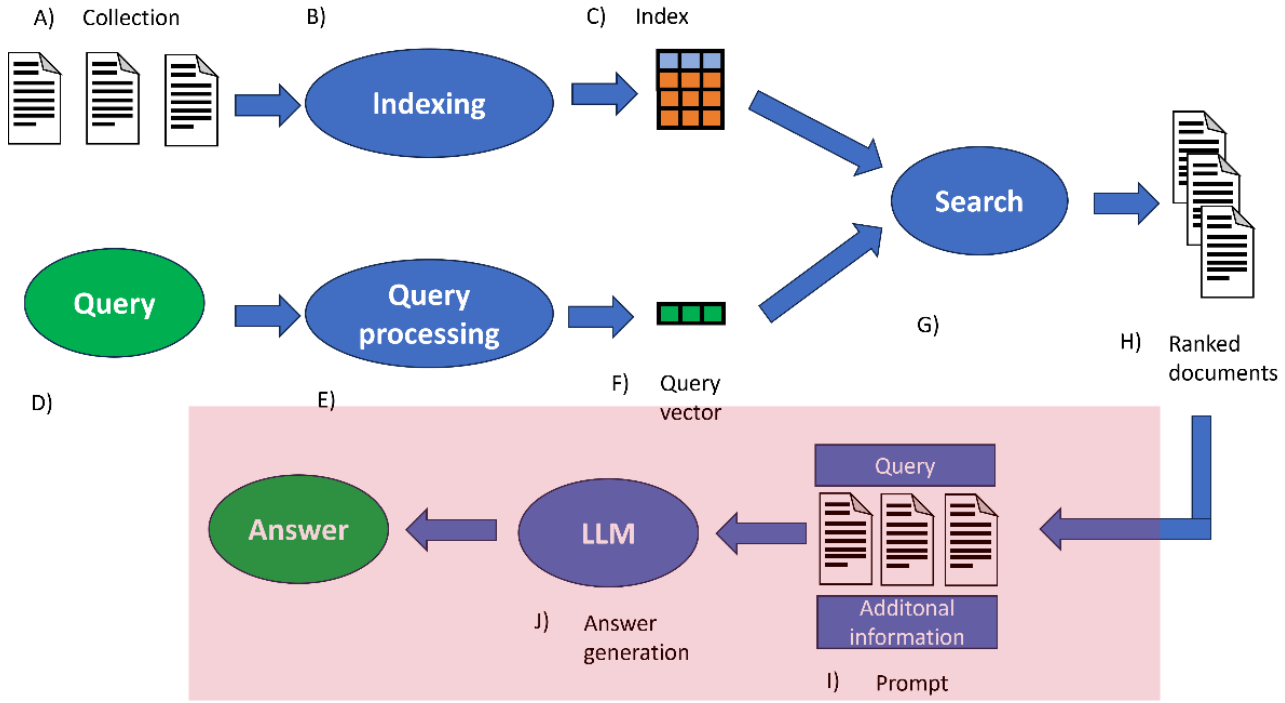
- La **Query dell'utente** viene convertita in un vettore (stesso modello di embedding).
- **Similarity Search**: Confronto matematico (es. Cosine Similarity) tra vettore query e vettori documenti.
- **Top-K**: Selezione dei K chunk più simili.



Pilastro 3: Generation

Sintetizzare la risposta.

- **Context Injection**: I chunk recuperati vengono inseriti nel Prompt.
- **Risposta**: L'LLM genera la risposta basandosi sui fatti forniti.



Visualizzazione Matematica

Come cambia la generazione?

LLM Standard:

$$P(y|x)$$

Probabilità della risposta y data la domanda x.

RAG:

$$P(y|x, z)$$

Probabilità della risposta y data la domanda x E il contesto z.

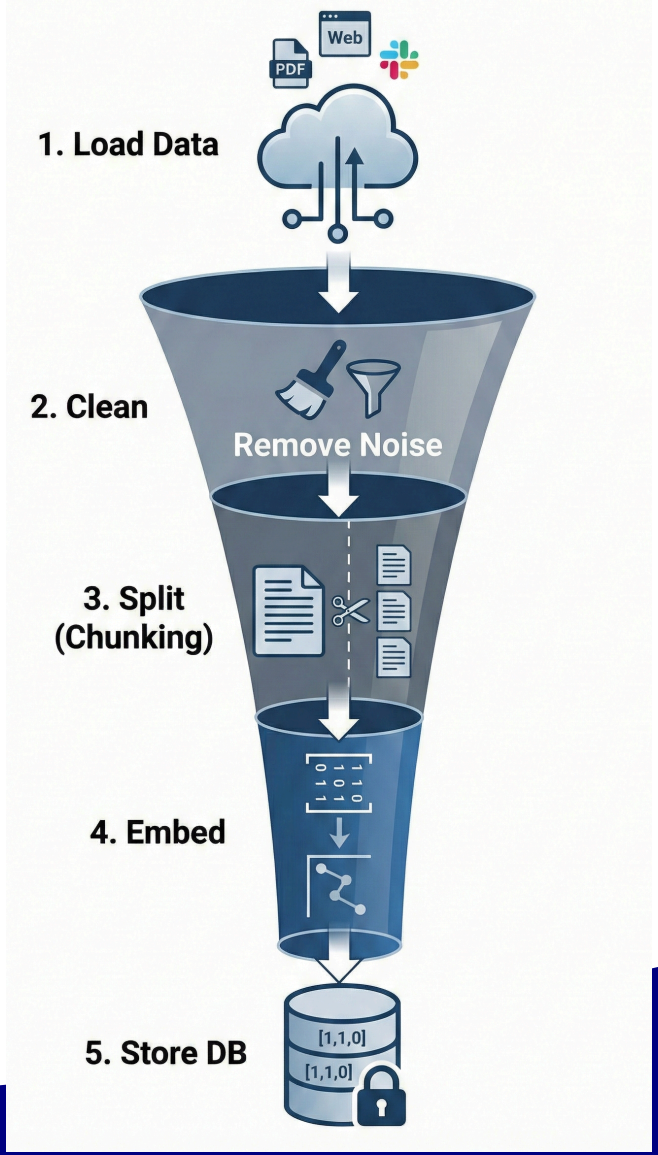
Modulo 2

Il Cuore Tecnico: Chunking, Embeddings e Vector Databases

Deep Dive: Strategie di Chunking

Perché è importante?

Il chunking non è banale "taglio del testo".



Deep Dive: Strategie di Chunking

Perché è importante?

- **Troppo Corto**:
  - Perdo contesto. Una frase senza il paragrafo precedente potrebbe non avere senso.
- **Troppo Lungo**:
  - Rumore per l'LLM (troppe info inutili nel prompt).

Deep Dive: Strategie di Chunking

1. Fixed-Size Chunking

La strategia più semplice. Taglio ogni N token.

- **Parametri**:
  - `chunk_size`: Es. 500 token.
- **Pro**: Facile, veloce.
- **Contro**: Spezza concetti a metà (es. taglia una frase o una tabella).

Deep Dive: Strategie di Chunking

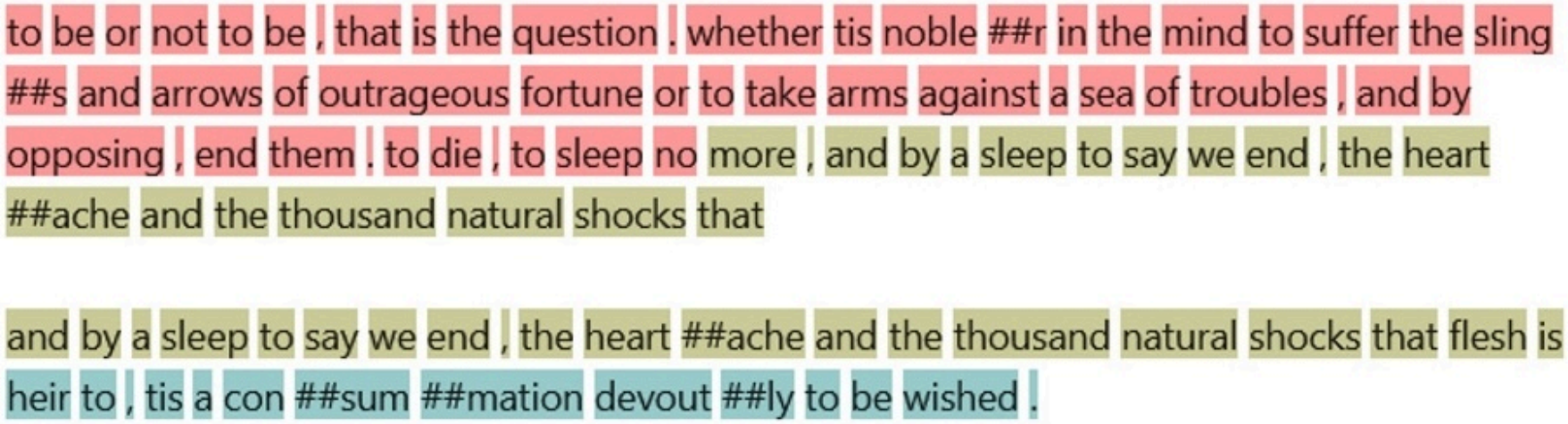
1. Fixed-Size Chunking

To be or not to be, that is the question. Whether tis nobler in the mind to suffer The slings and arrows of outrageous fortune Or to take arms against a sea of troubles, And by opposing, end them. To die, to sleep No more, and by a sleep to say we end, The heartache and the thousand natural shocks That

flesh is heir to, tis a consummation Devoutly to be wished.

1. Fixed-Size Chunking con Overlap

`chunk_overlap` : Es. 50 token (per mantenere continuità alle frontiere).



Deep Dive: Strategie di Chunking

2. Recursive Character Chunking

Un approccio più intelligente che rispetta la sintassi.

- Prova a tagliare su separatori logici in ordine di priorità:
    i. Paragrafo (`\n`)
- I chunk vengono uniti fino a un certo numero di token.
- **Risultato**: I paragrafi tendono a rimanere interi. Preserva meglio il contesto semantico base.

Deep Dive: Strategie di Chunking

3. Hierarchical

Sfrutta la struttura del documento.

- Se il documento è Markdown/HTML/Latex, usiamo gli Header (`#` , `##`) come confini.
- **Metadati**: Ogni chunk eredita il titolo della sezione in cui si trova.
  - *Esempio*: Un chunk che parla di "Prezzi" eredita `Section: Tariffe 2024` .
- **Pro**: Ottimo per RAG su documentazione tecnica.

Deep Dive: Strategie di Chunking

4. Semantic Chunking

Il metodo più avanzato. Non taglia su regole fisse, ma sul **significato**.

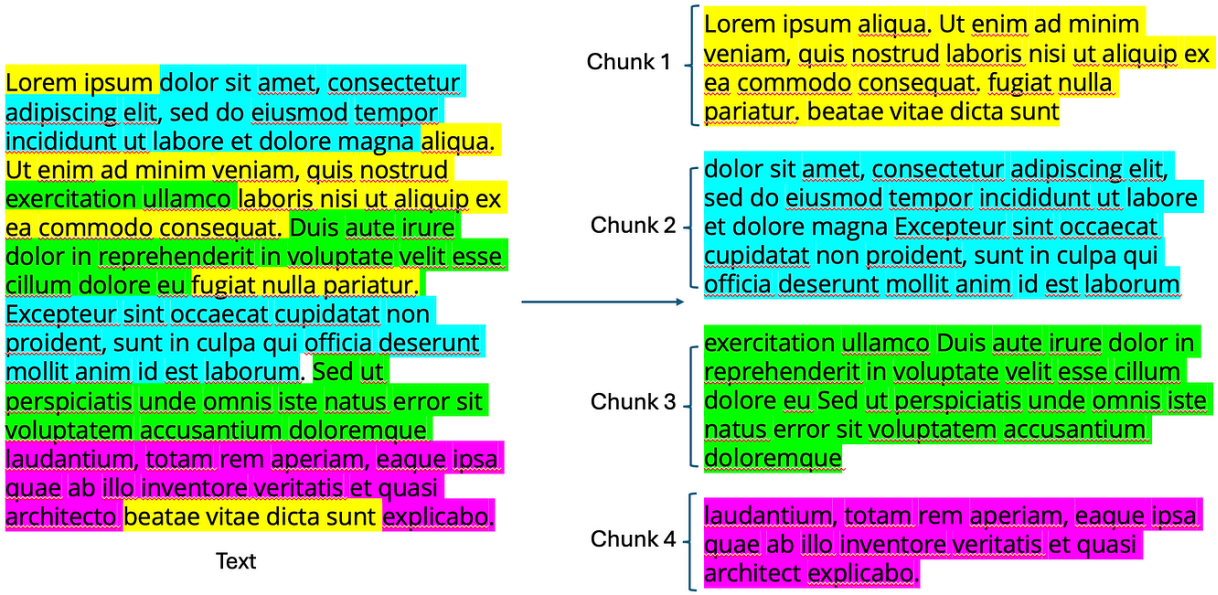
1. Calcola l'embedding per ogni frase.
2. Confronta la similarità tra frasi consecutive.
3. Se la similarità crolla (cambio di argomento), crea un nuovo chunk.

Deep Dive: Strategie di Chunking

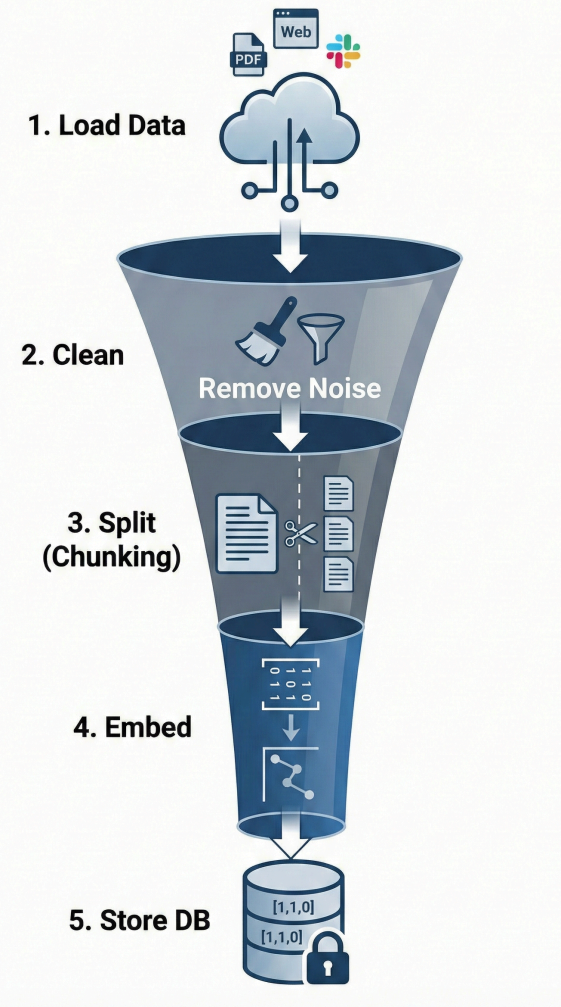
4. Semantic Chunking

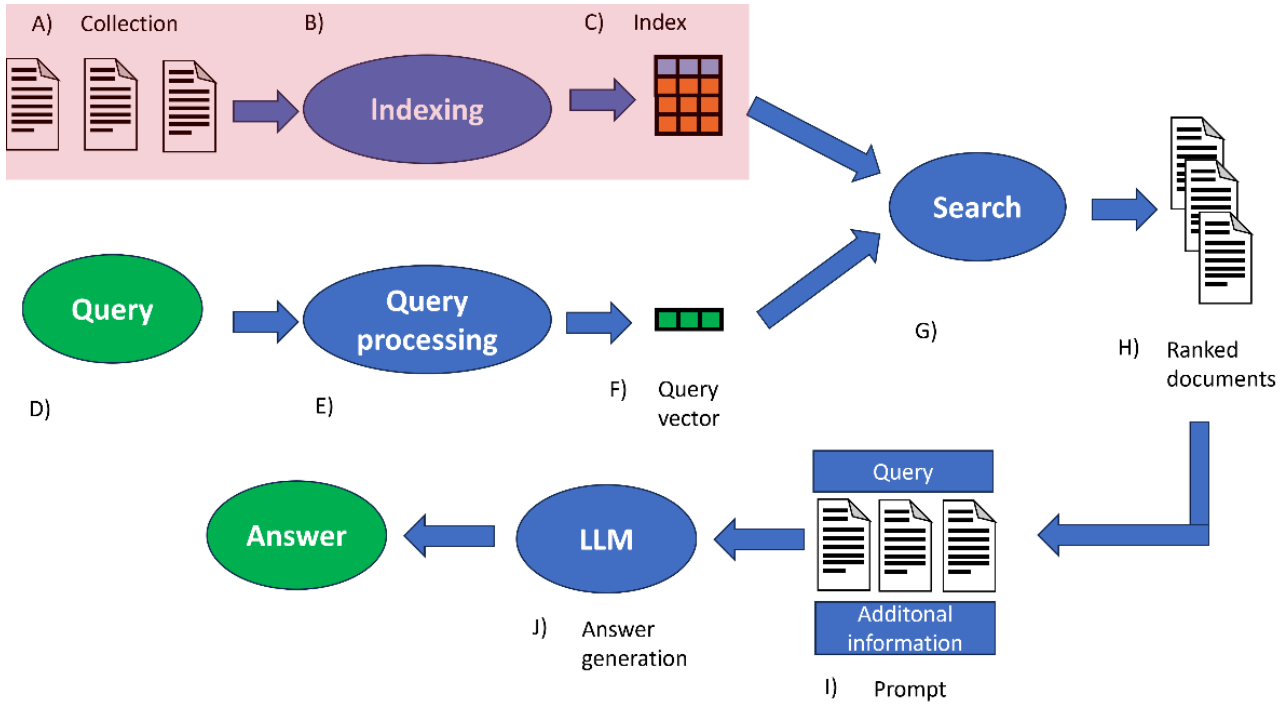
Il metodo più avanzato. Non taglia su regole fisse, ma sul **significato**.

- **Pro**: I chunk sono topic-coesi.
- **Contro**: Lento e costoso (richiede inferenza per ogni frase).



Embed them all

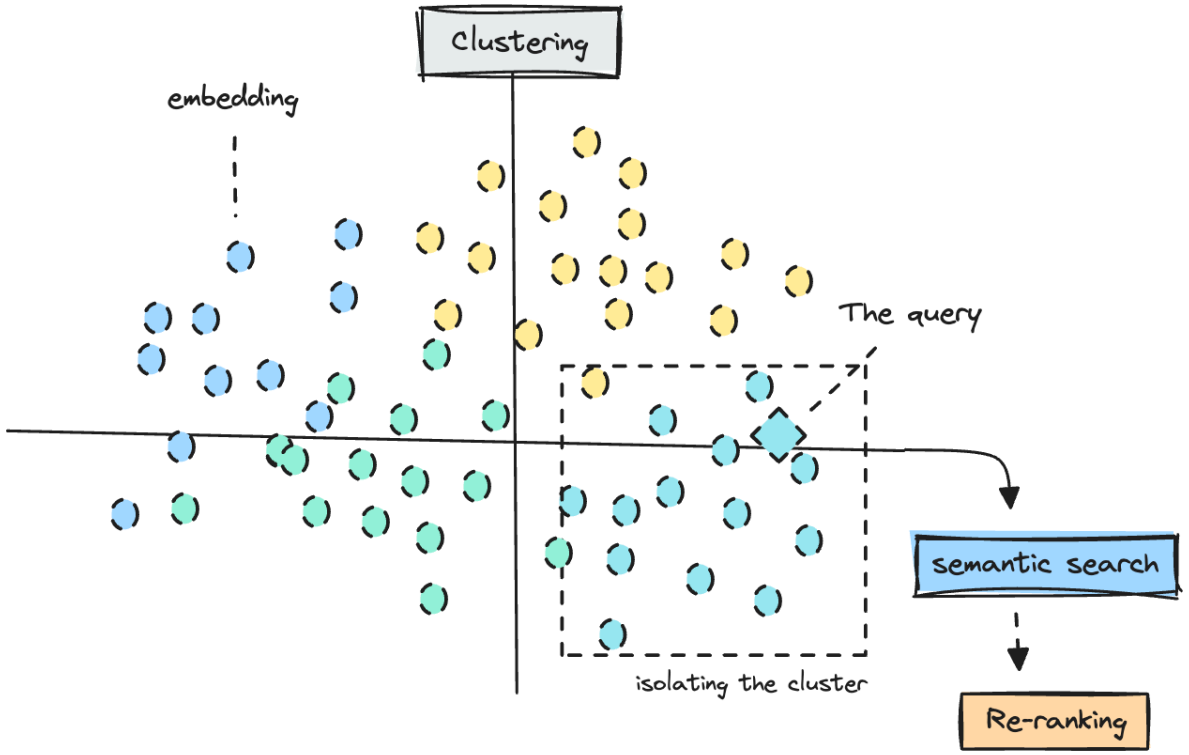




Cosa sono gli Embeddings?

Non possiamo calcolare la similarità tra stringhe di testo grezze in modo efficace. Dobbiamo trasformarle in numeri.

- **Input**: "Il gatto è sul divano"
- **Output**: `[0.12, -0.98, 0.45, ...]`



Sparse Vectors (Keyword-based)

L'approccio classico (es. TF-IDF, BM25).

- Il vettore ha la dimensione del vocabolario (es. 50.000 parole).
- La maggior parte dei valori è `0` (sparse).
- Se una parola c'è, il valore è > 0.

Index →	0	1	2	3	4	5	6	7	8
	and	**document**	**first**	**is**	**one**	**second**	**the**	**third**	**this**
"This is the first document."	0	0.46979139	0.58028582	0.38408524	0	0	0.38408524	0	0.38408524
"This document is the second document."	0	0.6876236	0	0.28108867	0	0.53864762	0.28108867	0	0.28108867
"And this is the third one."	0.51184851	0	0	0.26710379	0.51184851	0	0.26710379	0.51184851	0.26710379
"Is this the first document?"	0	0.46979139	0.58028582	0.38408524	0	0	0.38408524	0	0.38408524

Deep Dive: BM25 (Best Matching 25)

L'evoluzione del TF-IDF. È lo standard de-facto per la ricerca a parole chiave.

$$\text{Score}(D, Q) = \sum \text{IDF} \cdot \frac{\text{TF} \cdot (k + 1)}{\text{TF} + k \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

- **Term Frequency (TF)**: Quanto spesso appare la parola nel documento?
- **Inverse Document Frequency (IDF)**: La parola è rara in tutti i documenti?
- **Field Length**: Il documento è breve? (Se la parola appare in un tweet, vale più che in un libro).

Limiti del BM25

BM25 è potente, ma ha dei limiti strutturali:

1. **Vocabulary Mismatch**: Se query e documento non condividono le *esatte* parole, il match fallisce (es. "PC" vs "Computer").
2. **Polisemia**: Non distingue i significati multipli di una parola (es. "Fiera" campionaria vs "Fiera" bestia).
3. **Nessun Contesto**: Tratta le parole come sacchi di termini indipendenti (Bag-of-Words), ignorando la sintassi.

> **Soluzione**: Passare ai **Dense Vectors** che catturano il significato.

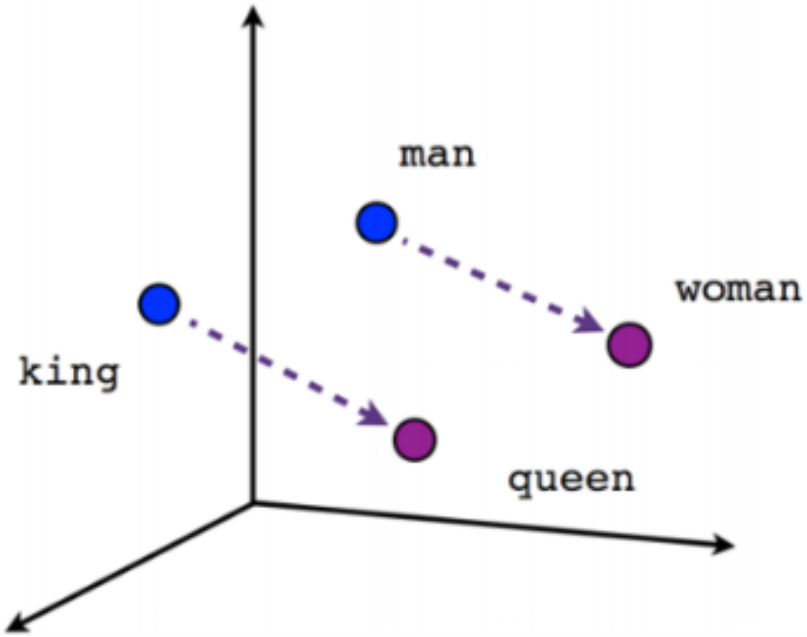
Tipologie di Rappresentazione

Dense Vectors (Semantic)

L'approccio moderno (BERT, OpenAI Ada).

- Cattura il **significato**, non solo la sintassi.
- **Analogy**: Riusciamo a fare matematica con i concetti.

$$\text{King} - \text{Man} + \text{Woman} \approx \text{Queen}$$



Problema Tecnico: Anisotropia (Rogue Dimensions)

I vettori densi non sono perfetti. Soffrono di **Anisotropia**.

- I vettori tendono a raggrupparsi in uno stretto cono dello spazio, rendendo la *Cosine Similarity* artificialmente alta per tutti.
- **Causa**: Alcune dimensioni hanno valori molto alti e dominano il calcolo.
- **Correlazione**: Spesso queste dimensioni tracciano la frequenza delle parole o la punteggiatura, non il significato.
- **Fix**: Normalizzazione (Z-Score) o post-processing.

Come si addestrano gli Embeddings?

Contrastive Learning

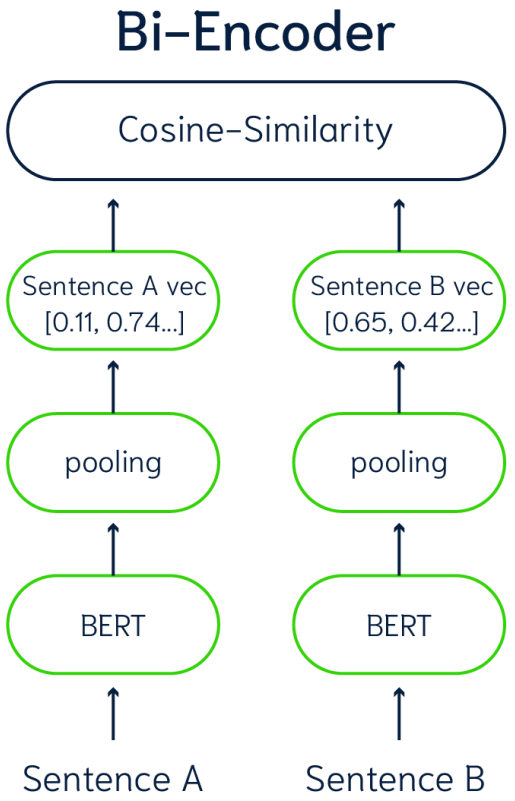
Non usiamo "etichette" classiche. Usiamo coppie di frasi.

- **Dataset (es. MultiNLI)**:
  - **Coppia Positiva (Entailment)**: "Il gatto dorme" $\leftrightarrow$ "Il felino riposa".
  - **Coppia Negativa (Contradiction)**: "Il gatto dorme" $\leftrightarrow$ "Il cane corre".
- **Obiettivo**: Avvicinare i vettori positivi, allontanare i negativi.

Bi-Encoder (Retrieval)

Il cavallo da lavoro.

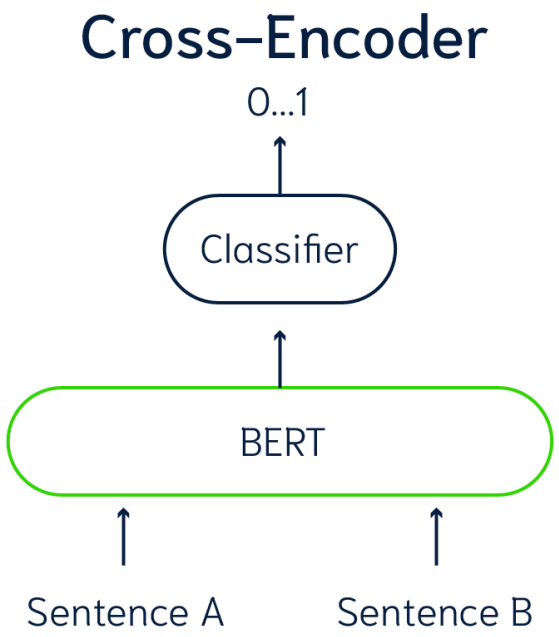
1. **Indicizzazione**: Calcolo tutti i vettori dei doc offline.
2. **Query Time**: Calcolo solo vettore query.
3. **Similarity**: Dot-product istantaneo su milioni di vettori.



Cross-Encoder (Re-ranking)

Il cecchino.

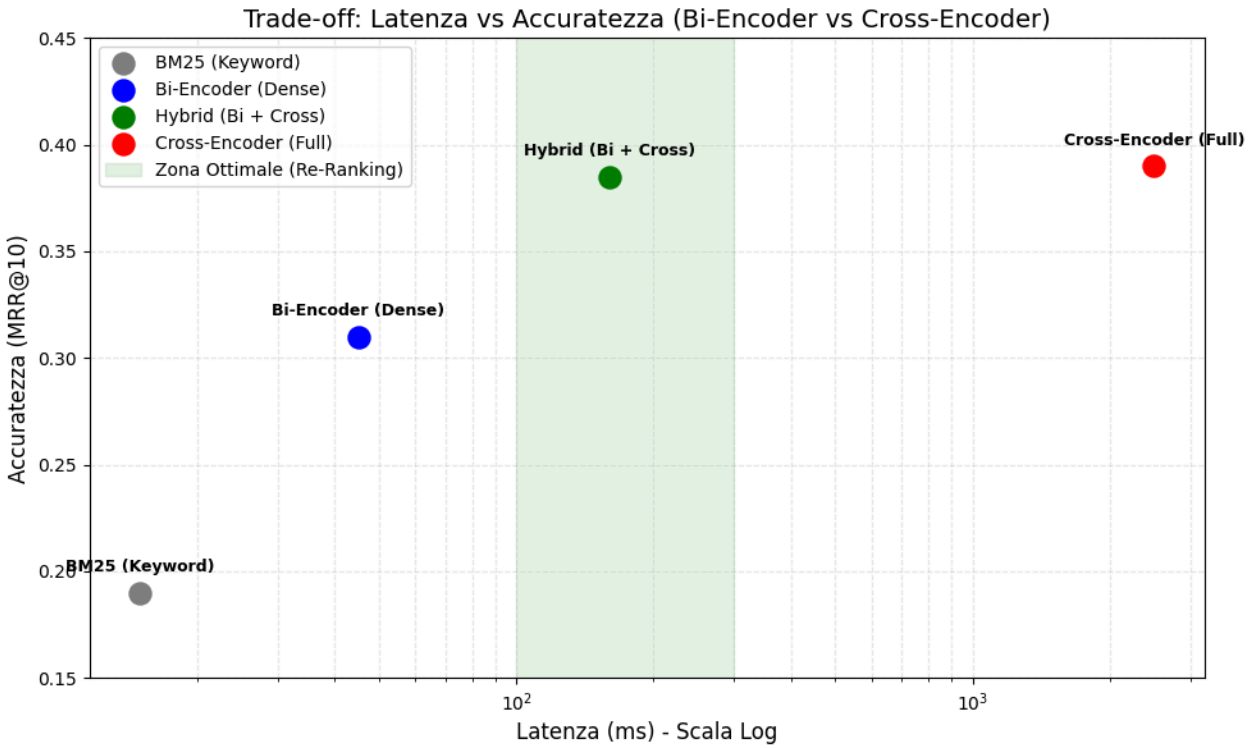
- Non crea vettori separati.
- Prende `[Query] + [Documento]` e li processa insieme strato per strato (Cross-Attention).
- **Vantaggio**: Capisce sfumature impossibili per il Bi-Encoder.
- **Svantaggio**: Lento. Non pre-calcolabile.



Performance vs Latency

Il compromesso ideale: **Pipeline a due stadi**.

1. **Retrieval**: Bi-Encoder recupera top-100 candidati (ms).
2. **Reranking**: Cross-Encoder riordina i top-100 (sec).



Come si addestrano gli Embeddings?

Loss Functions

Come calcoliamo l'errore durante il training?

1. **Cosine Similarity Loss**:
    - Semplice. Minimizza la distanza coseno per coppie positive (target=1) e massimizza per negative (target=0).

Come si addestrano gli Embeddings?

Loss Functions

Come calcoliamo l'errore durante il training?

1. **Cosine Similarity Loss**
2. **Multiple Negatives Ranking Loss (InfoNCE)**:
    - Più potente. Per ogni coppia positiva (a, b), considera *tutte* le altre frasi nel batch come negative.
    - Massimizza la probabilità di scegliere b dato a tra N opzioni.

Scegliere l'Embedder: MTEB Leaderboard

Non tutti i modelli sono uguali.

- **Metriche da guardare**:
  - **Retrieval**: Quanto è bravo a trovare documenti?
  - **Clustering**: Quanto raggruppa bene concetti simili?

Rank (Bor…	Model	Zero-shot	Memory Us…	Number of P…	Embedding D…	Max Tokens	Mean (T…	Mean (TaskT…
1	KaLM-Embedding-Gemma3-12B-2511	73%	44884	11.8	3840	32768	**72.32**	**62.51**
2	llama-embed-nemotron-8b	99%	28629	7.5	4096	32768	69.46	61.09
3	Qwen3-Embedding-8B	99%	14433	7.6	4096	32768	70.58	61.69
4	gemini-embedding-001	99%			3072	2048	68.37	59.59
5	Qwen3-Embedding-4B	99%	7671	4.0	2560	32768	69.45	60.86
6	Octen-Embedding-8B	99%	14433	7.6	4096	32768	67.84	60.28
7	Seed1.6-embedding-1215	89%			2048	32768	70.26	61.34
8	Qwen3-Embedding-0.6B	99%	1136	0.596	1024	32768	64.34	56.01
9	gte-Qwen2-7B-instruct	⚠ NA	29040	7.6	3584	32768	62.51	55.93
10	Linq-Embed-Mistral	99%	13563	7.1	4096	32768	61.47	54.14
11	multilingual-e5-large-instruct	99%	1068	0.560	1024	514	63.22	55.08

Scegliere l'Embedder: MTEB Leaderboard

Non tutti i modelli sono uguali.

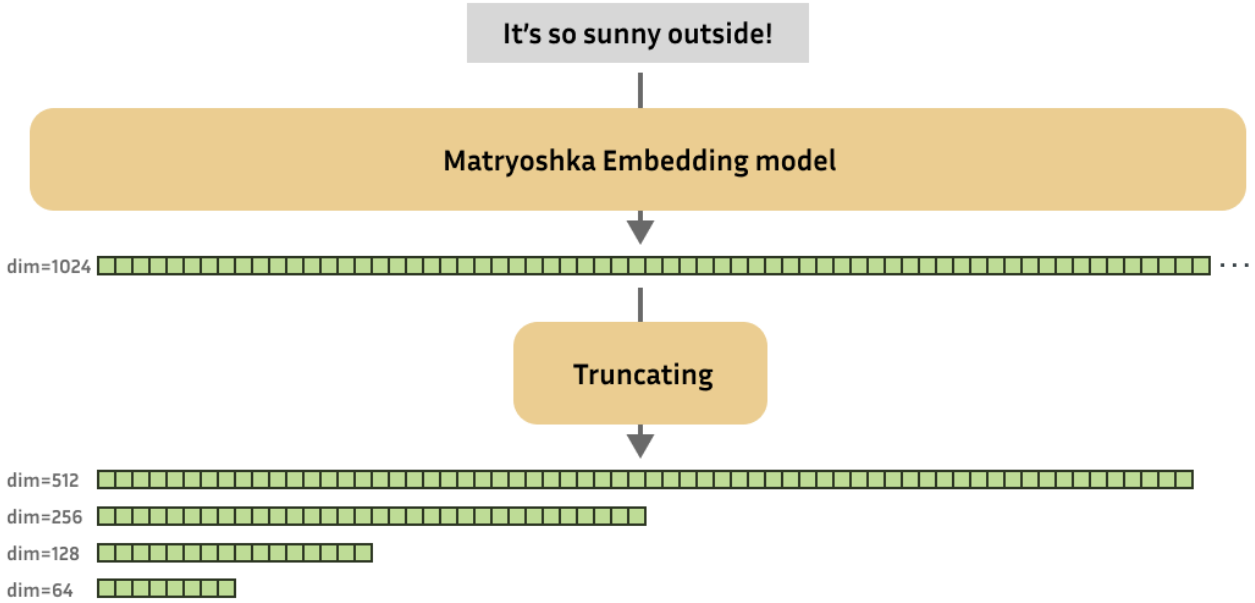
- **Trade-off**:
  - Modelli grandi → Lenti ma precisi.
  - Modelli piccoli → Veloci.

Rank (Bor…	Model	Zero-shot	Memory Us…	Number of P…	Embedding D…	Max Tokens	Mean (T…	Mean (TaskT…
1	KaLM-Embedding-Gemma3-12B-2511	73%	44884	11.8	3840	32768	**72.32**	**62.51**
2	llama-embed-nemotron-8b	99%	28629	7.5	4096	32768	69.46	61.09
3	Qwen3-Embedding-8B	99%	14433	7.6	4096	32768	70.58	61.69
4	gemini-embedding-001	99%			3072	2048	68.37	59.59
5	Qwen3-Embedding-4B	99%	7671	4.0	2560	32768	69.45	60.86
6	Octen-Embedding-8B	99%	14433	7.6	4096	32768	67.84	60.28
7	Seed1.6-embedding-1215	89%			2048	32768	70.26	61.34
8	Qwen3-Embedding-0.6B	99%	1136	0.596	1024	32768	64.34	56.01
9	gte-Qwen2-7B-instruct	⚠ NA	29040	7.6	3584	32768	62.51	55.93
10	Linq-Embed-Mistral	99%	13563	7.1	4096	32768	61.47	54.14
11	multilingual-e5-large-instruct	99%	1068	0.560	1024	514	63.22	55.08

Matryoshka Representation Learning (MRL)

Possiamo troncare i vettori senza ri-addestrare? Sì.

- I modelli MRL sono addestrati per mettere le informazioni più importanti *all'inizio* del vettore.



Vector Databases: Panoramica

Dove salviamo questi milioni di vettori?

Tassonomia Vector DB

Non esiste "un solo tipo" di Vector DB.

1. **SQL Extensions (Es. pgvector)**:
    - Aggiungono colonne vettoriali a DB esistenti (Postgres). Ottimo per non aggiungere infrastruttura.

Tassonomia Vector DB

Non esiste "un solo tipo" di Vector DB.

1. **SQL Extensions (Es. pgvector)**
2. **Vector Libraries (Es. FAISS, ScaNN)**:
    - Leggere, girano in-memory nel tuo codice Python.
    - Nessuna gestione persistenza/replica (No CRUD). Tu devi salvare l'indice su disco.

Tassonomia Vector DB

Non esiste "un solo tipo" di Vector DB.

1. **SQL Extensions (Es. pgvector)**
2. **Vector Libraries (Es. FAISS, ScaNN)**
3. **Vector-Native DB (Es. Pinecone, Weaviate, Qdrant)**:
    - Database completi con CRUD, Replica, Sharding, Cloud Management.

Decision Framework: Criteri di Scelta del DB

Non esiste il "miglior" DB, ma quello giusto per il progetto (Open Source vs Proprietario).

1. **Maturità**: Il sistema è stabile e supportato?
2. **Component Integration**: Si integra col mio stack (es. LangChain, LlamaIndex)?
3. **Compliance**: Supporta RBAC (Role-Based Access Control) per la privacy dei dati?

Deep Dive: Performance (Insertion vs Query)

Due metriche spesso in conflitto:

1. **Insertion Speed** (Write):
    - Cruciale per Real-time ingestion.
    - Tecniche: **Batch Processing**, Parallelization, Sharding.

Deep Dive: Performance (Insertion vs Query)

Due metriche spesso in conflitto:

1. **Insertion Speed** (Write):
    - Cruciale per Real-time ingestion.
2. **Query Speed** (Read):
    - Cruciale per la User Experience (bassa latenza).
    - Tecniche: **Caching**, Index Structures efficienti.

> Se il sistema è "Write-Heavy" (dati cambiano sempre), evitate DB lenti in indicizzazione.

Indexing Algorithms

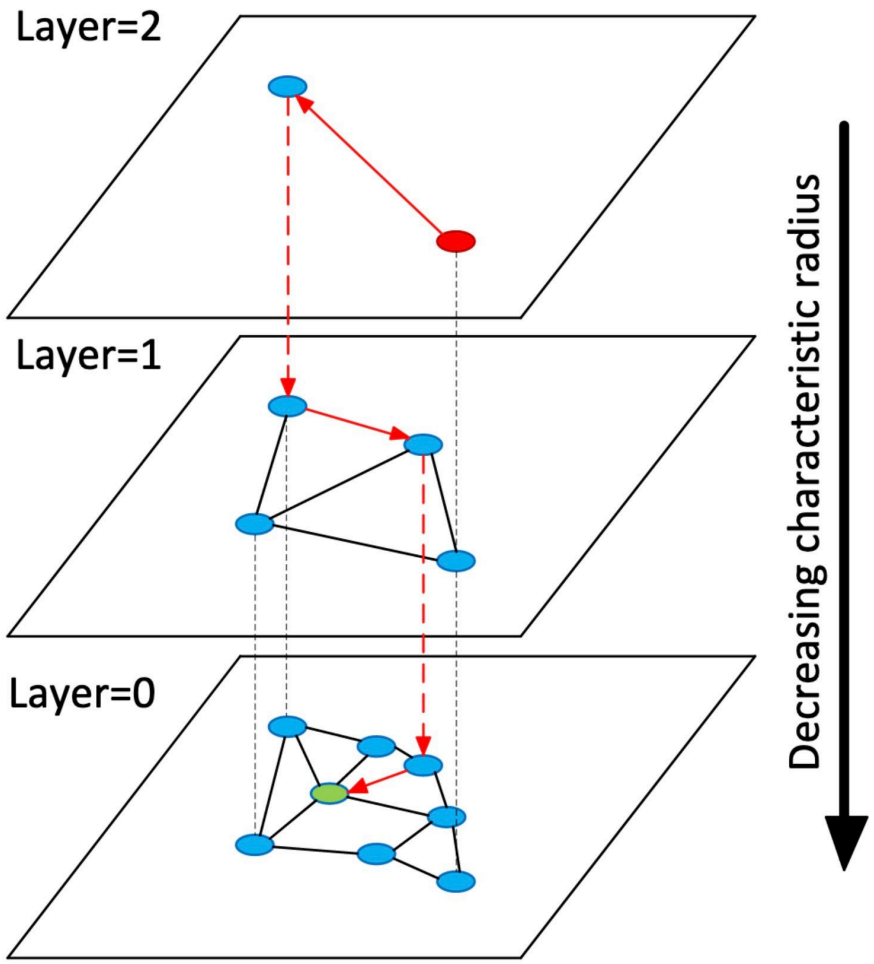
Flat vs ANN

- **Flat Index (Exact Search)**:
  - Confronta la query con **tutti** i vettori.
  - Precisione: 100%. Lentezza: $O(N)$.
  - Usabile solo per dataset piccoli (<100k).
- **ANN (Approximate Nearest Neighbor)**:
  - Accetta di sbagliare l'1% delle volte per essere 100x più veloce.
  - Algoritmi: IVF, HNSW, DiskANN.

Deep Dive: HNSW (Hierarchical Navigable Small World)

Immagina un grafo a più livelli.

- **Livello Alto (Autostrade)**: Pochi nodi, collegamenti lunghi. Ti sposti rapidamente nella "zona giusta" dello spazio vettoriale.
- **Livelli Bassi (Strade Locali)**: Molti nodi, collegamenti corti. Raffini la ricerca per trovare il punto esatto.



Modulo 3

Valutazione e Confronto Strategico

Perché Valutare un RAG è difficile?

Non esiste una "risposta giusta" unica come nella classificazione (Sì/No).

Il sistema ha due punti di fallimento indipendenti:

1. **Retrieval Failure**: Il DB non mi ha dato i documenti giusti.
2. **Generation Failure**: Il DB ha dato i documenti giusti, ma l'LLM ha sbagliato a leggere/rispondere.

Metriche di Retrieval

Recall@K

"Tra i primi K risultati (es. 5), c'è il documento che contiene la risposta?"

- Se la risposta è nel doc #1 -> Ottimo.
- Se la risposta è nel doc #5 -> Bene.
- Se la risposta è nel doc #6 (e io ne passo solo 5 all'LLM) -> **Fallimento**. Il sistema RAG non potrà mai rispondere.

Metriche di Retrieval

MRR (Mean Reciprocal Rank)

Premia i sistemi che mettono il documento giusto **in alto**.

- Doc in posizione 1 $\rightarrow$ Score 1.0
- Doc in posizione 2 $\rightarrow$ Score 0.5
- Doc in posizione 3 $\rightarrow$ Score 0.33

$$MRR = \frac{1}{Q} \sum \frac{1}{\text{rank}_i}$$

Metriche di Retrieval

Precision & MAP

- **Precision**: Di 5 documenti recuperati, quanti sono "utili"? (Spesso in RAG ce ne basta 1, ma averne 4 di rumore può confondere l'LLM).
- **MAP (Mean Average Precision)**: Una media robusta della precisione su diverse query.

Metriche di Generation: LLM-as-a-Judge

Usiamo GPT-4 per dare un voto a GPT-3.5 (o Llama).

Framework standard: **RAGAS**.

Scompone la valutazione in tre assi:

1. Faithfulness
2. Answer Relevance
3. Context Precision

RAGAS Deep Dive

1. Faithfulness (Groundedness)

Obiettivo Check: Hallucination.

- L'LLM Giudice estrae le "affermazioni" (claims) dalla risposta generata.
- Verifica: "Ogni affermazione è supportata dal contesto recuperato?"
- Se la risposta contiene info vere ma non presenti nel contesto $\rightarrow$ Bassa Faithfulness (perché in RAG vogliamo che usi solo il contesto).

RAGAS Deep Dive

2. Answer Relevance

Obiettivo Check: Utilità.

- L'LLM Giudice genera delle "domande artificiali" basandosi sulla risposta data.
- Verifica: "La domanda originale è simile a queste domande artificiali?"
- Penalizza risposte che divagano o non rispondono al punto, anche se vere.

RAGAS Deep Dive

3. Context Precision

Obiettivo Check: Qualità del Retrieval (visto lato LLM).

- Verifica se i "pezzi utili" del contesto sono in alto o sepolti in fondo al noise.
- Fondamentale per il "Lost in the Middle" phenomenon (gli LLM tendono a ignorare info a metà del contesto).

RAG vs Fine-tuning: Il Grande Dibattito

Spesso visti come alternativi, in realtà sono complementari.

RAG: I Punti di Forza

- **Dinamicità**: Aggiorno un file nel DB e il modello lo "sa" istantaneamente.
- **Trasparenza**: Posso dire "L'ho letto nel documento X, pag 4".
- **Economia**: Niente GPU training farm, solo inferenza.
- **Privacy**: Posso avere diversi indici per diversi utenti (ACL).

Fine-tuning: I Punti di Forza

- **Forma e Stile**: Insegnare al modello a parlare come un medico, un legale, o a rispondere in JSON.
- **Linguaggio Settoriale**: Se uso termini iperspecifici (gergo aziendale) che l'LLM base non conosce, il Fine-tuning aiuta a "imparare la lingua".
- **Latenza**: Elimina il tempo di "Retrieval". Risposta diretta.

Matrice Decisionale

Scenario	Soluzione Consigliata
Necessità di conoscenza aggiornata (News, Stock)	**RAG**
Necessità di formato specifico (Code, Medical Report)	**Fine-tuning**
Dominio di conoscenza totalmente oscuro all'LLM	**RAG + Fine-tuning**
Riduzione Allucinazioni Fattuali	**RAG**

L'Approccio Ibrido: RAG + Fine-tuning

La frontiera avanzata.

1. **Fine-tune Embedder**: Addestro il modello di embedding sui MIEI dati per migliorare il retrieval.
2. **Fine-tune LLM**: Addestro l'LLM a essere un bravo "lettore di contesto" (ignorare il rumore, sintetizzare meglio).
3. **RAG Runtime**: Uso questi modelli custom nel flow RAG classico.

Grazie per l'attenzione!

Q&A

Riferimenti

[^1]: Kalai, A. T., Nachum, O., Vempala, S. S., & Zhang, E. (2025). *Why language models hallucinate*. arXiv preprint arXiv:2509.04664.

[^2]: Raieli, S., & Iuculano, G. (2025). *Building AI Agents with LLMs, RAG, and Knowledge Graphs: A practical guide to autonomous and modern AI agents*. Packt Publishing Ltd.