UNA SERIE SULL'INTELLIGENZA ARTIFICIALE

# Io, Gemini e altri quattro

## Parte VII

*RAG e la Memoria Contestuale: Dare un Cervello Privato all'IA*

ROCCO MASSIMILIANO MONDO

## ABSTRACT

*Ogni LLM ha una data di emissione che coincide quasi sempre con quella di fine addestrato, questo limite temporale segna l'inizio dell'invecchiamento della stessa LLM, da questa data in poi il modello non ha più conoscenza dei fatti e degli eventi che si susseguono. Non conosce le circolari interne della vostra azienda, le specifiche tecniche del vostro prodotto, le versioni aggiornate delle normative CEI. Un addestramento su misura? Si, sarebbe possibile, ma molto improbabile. Costoso, lento, complesso. Esiste però una soluzione elegante e già matura: il* ***Retrieval Augmented Generation****,* ***RAG****. In questo settimo episodio scopriamo come funziona la tecnica che permette di "iniettare" conoscenza privata e aggiornata nel processo di inferenza, senza toccare il modello base. Capiremo meglio il meccanismo degli* ***embedding*** *e i* ***database vettoriali****, vedremo come costruire un sistema RAG funzionante su documenti tecnici propri, e analizzeremo le implicazioni pratiche in termini di privacy, accuratezza e costo. Il tutto con un caso d'uso concreto:* ***un assistente RAG*** *che risponde a domande sulle normative CEI partendo dai testi ufficiali.*




*Rocco M. Mondo, Luglio 2026*

# PARTE VII

## RAG E LA MEMORIA CONTESTUALE: DARE UN CERVELLO PRIVATO ALL'IA

### *PREMESSA: IL PEZZO MANCANTE DEL PUZZLE*

Nelle prime sei puntate di questa serie abbiamo realizzato un percorso di base completo. Siamo partiti dalla panoramica sulla rivoluzione AI del CES 2026 (***Parte I***), abbiamo ripercorso la storia dell'intelligenza artificiale da Turing ai robot neurali (***Parte II***), ci siamo addentrati nell'anatomia tecnica degli LLM con l'architettura Transformer (***Parte III***), abbiamo imparato a comunicare con questi strumenti attraverso il prompt engineering (***Parte IV***), abbiamo confrontato i cinque grandi modelli (***Parte V***) e abbiamo esplorato il mondo degli AI Agent (***Parte VI***).

Ogni puntata ha aggiunto un tassello in più di conoscenza su questo rivoluzionario mondo della IA. Eppure, al termine di questa prima fase del nostro percorso, rimane aperto un problema pratico fondamentale: come si fa a far conoscere all'IA i nostri documenti, le nostre normative, i nostri archivi? Come si trasforma un modello generico (addestrato su miliardi di testi pubblici) in un assistente che conosce la specifica circolare CEI pubblicata il mese scorso, o le schede tecniche dei prodotti della nostra azienda?



In parole povere, come possiamo rendere edotta una generica LLM open source sui nostri dati personali e soprattutto come possiamo proteggere tali dati da sistemi esterni?

La risposta arriva con il sistema ***RAG*** (***Retrieval Augmented Generation)***. È la tecnologia che, a mio avviso, rende l'IA generativa davvero utilizzabile nel contesto professionale e tecnico. Non richiede competenze da ricercatore, non ha i costi proibitivi del ***fine-tuning***, e funziona già oggi con strumenti open source maturi e stabili.

### *IL PROBLEMA: LA CONOSCENZA CONGELATA*

Ogni LLM ha una "data di emissione" che coincide con il giorno in cui l'addestramento è terminato. Dopo quella data, il modello non sa nulla di ciò che è accaduto nel mondo. Non ha letto la circolare CEI del febbraio scorso, non conosce le specifiche tecniche del vostro ultimo prodotto, non ha accesso ai vostri archivi progettuali. Claude 3.7, GPT-4o, Gemini 2.0: tutti condividono questa limitazione strutturale. Attenzione, il problema va ben oltre la semplice

mancanza di aggiornamenti. Riguarda anche (e soprattutto) la conoscenza privata e specializzata: la documentazione interna della vostra azienda, i calcoli storici, le relazioni tecniche archiviate, le corrispondenze con i clienti. Questo materiale non è mai entrato, per ovvie ragioni di riservatezza, nel corpus di addestramento di nessun LLM pubblico. Quindi come si possono implementare queste informazioni all'interno dell'LLM? La risposta come possiamo intuire da quello che abbiamo detto precedentemente è: non si può fare, non con l'attuale tecnologia.

*Tre categorie di conoscenza che l'LLM non possiede*

- ***Conoscenza temporale***: tutto ciò che è accaduto dopo la data di rilascio del modello: aggiornamenti normativi, nuovi prodotti, cambiamenti regolatori.
- ***Conoscenza privata:*** documentazione interna aziendale, archivi tecnici, specifiche proprietarie, corrispondenze riservate.
- ***Conoscenza iperspecializzata:*** documenti tecnici di nicchia, normative nazionali di settore, standard industriali non rappresentati nei corpus pubblici.



**[Il fine-tuning: perché non è la soluzione giusta per la maggior parte dei casi]**

*La soluzione intuitiva (addestrare il modello sui propri dati privati) esiste e si chiama* ***Fine-Tuning*** ma questa soluzion, *ossia riaddestrare una LLM, ha costi proibitivi. Fare fine-tuning su un modello da 70 miliardi di parametri richiede:*

- ***Hardware****: cluster GPU con VRAM aggregata di centinaia di GB (NVIDIA A100 o H100)*
- ***Tempo****: da alcune ore a diversi giorni di calcolo continuo*
- ***Dati****: migliaia di esempi etichettati nel formato corretto*
- ***Competenze****: esperienza in ML engineering per gestire il processo*

***Il RAG*** *offre risultati comparabili per il recupero di fatti e regole, a una frazione del costo. Per la maggior parte dei casi professionali, è la scelta corretta.*

# PARTE VII

# RAG E LA MEMORIA CONTESTUALE: DARE UN CERVELLO PRIVATO ALL'IA

## *IL RAG: COME FUNZIONA*

Il principio alla base del ***RAG*** è semplice quanto elegante: invece di insegnare all'LLM nuove informazioni durante l'addestramento, forniamo le stesse al momento giusto, inserendole direttamente nel contesto della richiesta. Il modello non cambia, non viene modificato. ***Si arricchisce semplicemente del contesto*** di cui ha bisogno, istante per istante.

L'analogia più efficace è quella scolastica: la differenza tra uno studente che ha memorizzato tutto il manuale durante anni di studio (***addestramento/fine-tuning***) e uno che ha il manuale aperto davanti a sé durante l'esame (***RAG***). Il secondo approccio è più flessibile, più aggiornabile, e (fatto cruciale per il contesto professionale) più verificabile: sappiamo esattamente da dove viene la risposta.



### *Il Flusso RAG in quattro passi*

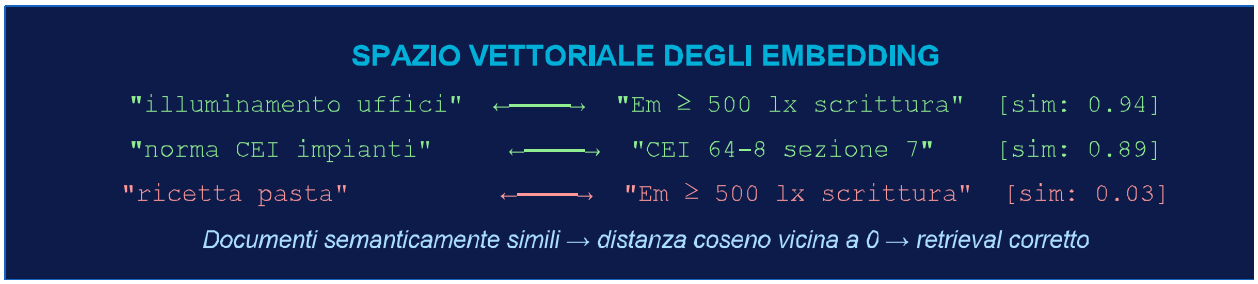
| | |
|---|---|
| **1 · INDICIZZAZIONE** | I documenti PDF/DOCX vengono segmentati in ***chunk*** (500–800 token). Ogni ***chunk*** è convertito in un vettore numerico (***embedding***) e salvato nel database vettoriale con i suoi metadati (fonte, pagina, sezione). |
| **2 · QUERY** | L'utente pone una domanda in linguaggio naturale. Il sistema converte anche la domanda in un ***embedding*** usando lo stesso modello dei documenti. |
| **3 · RETRIEVAL** | Il database vettoriale calcola la distanza coseno tra il vettore della domanda e tutti i ***chunk indicizzati***. Restituisce i 5–10 chunk semanticamente più vicini — anche se usano parole diverse dalla domanda. |
| **4 · AUGMENTED GENERATION** | ***I chunk recuperati*** vengono inseriti nel ***prompt*** insieme alla domanda originale. L'LLM genera la risposta con accesso sia alla sua conoscenza generale sia al contesto specifico — e cita la fonte. |

## *GLI EMBEDDING: LA MATEMATICA DEL SIGNIFICATO*

Prima di proseguire, è necessario comprendere bene cosa sono gli ***embedding***, perché sono il

componente più critico di un sistema RAG e paradossalmente il più ***trascurato. Sbagliare il modello di embedding*** significa costruire un sistema che non trova ciò che l'utente cerca, o che porta in superficie materiale irrilevante.

***Un embedding*** è una rappresentazione matematica del significato di un testo. Ogni parola, frase o paragrafo viene trasformato in un punto in uno spazio a centinaia  o migliaia  di dimensioni (ricordate l'episodio III, dove abbiamo definito un LLM come un database multidimensionale) . La proprietà fondamentale: testi semanticamente simili si trovano vicini tra di loro in questo spazio multidimensionale, anche se utilizzano parole completamente diverse.




Questo significa che una domanda come "Quanto deve essere illuminato un ufficio per scrivere?" troverà correttamente il paragrafo della normativa che recita "per i lavori di scrittura e lettura è richiesto Em ≥ 500 lx", anche se non condividono nessuna parola chiave. È una ricerca per significato, non per parole chiave.

*Confronto modelli di embedding per testi tecnici in italiano*

| Modello | Dimensioni vettore | Linguaggi | Uso consigliato |
|---|---|---|---|
| text-embedding-3-small (OpenAI) | 1.536 | Multilingue | Cloud, rapporto qualità/costo ottimo |
| text-embedding-3-large (OpenAI) | 3.072 | Multilingue | Cloud, massima precisione |
| nomic-embed-text (Nomic AI) | 768 | Multilingue | Locale via Ollama, open source |

| | | | |
|---|---|---|---|
| multilingual-e5-large (Microsoft) | 1.024 | Multilingue | Eccellente per italiano tecnico |
| bge-m3 (BAAI) | 1.024 | 100+ lingue | Standard per documenti normativi |

Per documenti tecnici in italiano (normative CEI, relazioni di progetto, guide illuminotecniche) i modelli che hanno prodotto i risultati migliori nei miei test sono ***multilingual-e5-large e bge-m3***. Entrambi eseguibili localmente senza costi di API, con la libreria ***sentence-transformers*** di Python.

## *IL CHUNKING: L'ARTE DI SEZIONARE I DOCUMENTI*

Prima di poter ***indicizzare un documento***, questo deve essere suddiviso in ***frammenti chiamati chunk***. La dimensione del ***chunk*** è uno dei parametri più importanti dell'intero sistema RAG, e la scelta ottimale dipende fortemente dal tipo di documento.



| Tipo documento | Chunk size consigliato | Overlap | Motivazione |
|---|---|---|---|
| Normative tecniche (CEI, UNI) | 500–800 token | 10–15% | Articoli autocontenuti, tabelle da preservare |
| Manuali tecnici | 400–600 token | 15–20% | Alta densità di riferimenti incrociati |
| Relazioni di progetto | 300–500 token | 10% | Sezioni logicamente separate |
| Corrispondenza tecnica | 200–300 token | 5–10% | Messaggi brevi e autonomi |
| Libri/guide divulgative | 600–1000 token | 20% | Argomenti con ampio contesto narrativo |

**[Nota tecnica: il chunk overlap]**
*L'overlap è la percentuale di testo condivisa tra chunk contigui. Serve a evitare che un concetto spezzato a metà tra due chunk venga perso nel retrieval.*
***Esempio**: con chunk_size=600 e overlap=80, il chunk n.2 inizia 80 token prima della fine del chunk n.1. Il costo è un database più grande; il beneficio è una copertura semantica più continua.*

# PARTE VII

# RAG E LA MEMORIA CONTESTUALE: DARE UN CERVELLO PRIVATO ALL'IA

## *I DATABASE VETTORIALI: DOVE VIVE LA MEMORIA*

Una volta calcolati gli ***embedding***, questi devono essere salvati in un sistema che permetta la ricerca veloce tra milioni di vettori. I database relazionali tradizionali non sono progettati per questo: una query SQL su tabelle di numeri in alta dimensione sarebbe inutilmente lenta. Servono strutture dati specializzate per la ***"Approximate Nearest Neighbor Search" (ANN).***

| Database | Tipo | Punto di forza | Ideale per |
|---|---|---|---|
| ChromaDB | Open source, locale | Setup 5 minuti, zero config | Prototipazione, uso personale |
| FAISS (Meta) | Open source, locale | Velocissimo su grandi collezioni | Milioni di documenti, no infra |
| Qdrant | Open source / Cloud | Filtri avanzati, alta performance | Produzione con dati strutturati |
| Pinecone | Cloud managed | Zero infrastruttura, scalabile | Enterprise cloud |
| pgvector (PostgreSQL) | Estensione SQL | Integra vettori in DB esistente | Chi già usa PostgreSQL |

**[Raccomandazione per chi inizia]**
*Per iniziare con RAG su documenti tecnici propri, ChromaDB è la scelta più semplice:*

- *Si installa con: pip install chromadb*
- *Non richiede servizi esterni né configurazione server*
- *Persiste su disco locale: i vettori sopravvivono al riavvio*
- *Supporta metadati: filtrare per norma, per data, per autore*

*Una volta che il sistema funziona correttamente, potete migrare a FAISS o Qdrant se le performance diventano un requisito critico.*

## *RAG o FINE-TUNING: QUANDO USARE COSA*

La domanda ricorrente che spesso ci si pone, quando si è agli inizi è: quando ha senso usare il RAG e quando invece è necessario il fine-tuning? Non sono approcci alternativi ma

complementari. La scelta dipende dalla natura del problema e, soprattutto, dall'obiettivo.

| Criterio | RAG | Fine-Tuning |
|---|---|---|
| Costo | Basso (storage + embedding) | Alto (GPU, tempo, dati etichettati) |
| Aggiornabilità | Immediata (aggiungi un documento) | Richiede nuovo ciclo di training |
| Tracciabilità | Alta (cita sempre la fonte) | Bassa (conoscenza nei pesi) |
| Stile comunicativo | Non modificabile | Personalizzabile (tono, formato) |
| Conoscenza di dominio | Ottima per fatti e regole | Ottima per ragionamento specializzato |
| Privacy | Totale se eseguito in locale | Totale se il training è locale |
| Ideale per | Accesso a documenti aggiornati | Specializzare il comportamento |

La combinazione più potente è RAG + fine-tuning: un modello specializzato sul dominio (attraverso fine-tuning) che ha anche accesso ai documenti aggiornati (con RAG). Questo è esattamente l'architettura del progetto descritto nella Parte X: un ricercatore per l'ingegneria elettrica addestrato sul dominio della fisica quantistica applicata e aggiornato via RAG sui documenti di ricerca peculiari.



## *PRIVACY E SICUREZZA NEL RAG*

Uno dei vantaggi più rilevanti del RAG per il contesto professionale è la possibilità di eseguire l'intero sistema in locale, senza che nessun dato lasci mai il computer. Nella Parte V abbiamo visto che ***LLaMA 3.3*** via ***Ollama*** permette l'inferenza completamente ***on-premise***. Lo stesso vale ***per l'intera pipeline RAG***.

### *L'architettura RAG completamente locale*

- **Modello di embedding locale:** bge-m3 o multilingual-e5-large via Sentence Transformers — nessuna chiamata API esterna.
- **Database vettoriale locale:** ChromaDB o FAISS, file salvati su disco locale — dati mai trasmessi.

- **LLM locale:** LLaMA 3.3 o Mistral via Ollama, inferenza completamente on-premise.
- **Interfaccia:** Open WebUI o interfaccia Python custom — tutto offline, anche senza connessione internet.

Per chi lavora con documentazione tecnica riservata, brevetti, dati aziendali o informazioni personali di clienti, questa architettura non è un dettaglio implementativo: è un requisito di compliance. Nessun dato tecnico riservato transita mai su server di terze parti.

**[Verifica sempre: la regola d'oro del RAG professionale]**
*Anche con il RAG, l'LLM può generare risposte che vanno oltre i documenti recuperati, mescolando la propria conoscenza generale con il contesto fornito. In ambito normativo, questa è la differenza tra una risposta corretta e una pericolosamente plausibile ma sbagliata.*

*Regole operative:*

1. *Chiedete sempre al sistema di citare la fonte specifica (norma, articolo, pagina).*
2. *Verificate manualmente le citazioni sui documenti originali prima di usarle in un progetto.*
3. *Se il sistema risponde senza citare una fonte, trattate la risposta come non verificata.*
4. *L'LLM accelera la ricerca; la responsabilità tecnica resta sempre al professionista.*



## *APPLICAZIONI PRATICHE PER L'INGEGNERE ELETTRICO*

Il RAG non è uno strumento astratto. Nella mia attività di progettista elettrico, vedo applicazioni immediate e ad alto valore aggiunto. Descriverò le tre che ritengo più strategiche.

### *1. Assistente Normativo CEI/UNI*

Un database vettoriale indicizzato con le normative CEI 64-8, UNI EN 12464-1, CEI EN 60269, CEI EN 61439 e le relative guide applicative diventa un assistente che risponde a domande come: "Quali sono i requisiti di coordinamento delle protezioni per un quadro BT secondo CEI

EN 61439-2?". La risposta include la fonte, è verificabile, e si aggiorna semplicemente aggiungendo il nuovo PDF al database.

*2. Sistema di Ricerca su Archivi Progettuali*

Anni di relazioni tecniche, calcoli illuminotecnici, capitolati di appalto archiviati in PDF diventano interrogabili semanticamente. "Mostrami tutti i progetti con tensione di alimentazione 11 kV e potenza installata superiore a 1 MVA" restituisce i documenti pertinenti anche se non usano esattamente quelle parole chiave.

*3. Verifica di Conformità Automatizzata*

Data la descrizione di un impianto o di un componente, il sistema RAG può confrontarla con i requisiti normativi indicizzati e segnalare le potenziali non conformità. Non sostituisce la revisione del progettista, ma riduce drasticamente il tempo di una prima verifica sistematica.

## *CONCLUSIONE*



Il RAG è, a mio avviso, la tecnologia che rende l'IA generativa davvero utilizzabile nel contesto professionale e tecnico. Trasforma un modello generico in un assistente che conosce i vostri documenti, le vostre normative, la vostra storia progettuale — mantenendo la piena tracciabilità delle fonti e la privacy dei dati.

Non richiede competenze da ricercatore per essere implementato. Con strumenti open source come ***ChromaDB***, ***LangChain*** e un modello di ***embedding locale***, un sistema RAG funzionante su documenti tecnici può essere implementato in tempi veramente brevi. Il risultato è un assistente che risponde con la precisione di chi ha il manuale davanti — e sa sempre citare da quale pagina ha letto.

Prima di chiudere questo episodio, vorrei esprimere un ultimo concetto che penso sia importante sottolineare e che riguarda il fatto che un ***sistema RAG*** può essere implementato con tecniche e modalità applicative molto diverse fra di loro tutto dipenderà dall'obbiettivo che vogliamo raggiungere e dalla conoscenza tecnica che abbiamo acquisito nel campo della ***IA Automation.***

*Rocco M. Mondo, Luglio 2026*

*Rocco M. Mondo, Luglio 2026*