LLM *with* Retrieval-Augmented Generation

Tirtharaj Dash

Department of CS & IS
BITS Pilani, Goa Campus, India
Homepage

February 2026

Outline

LLMs: what we know so far

An autoregressive LLM factorises the joint distribution over a token sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ as:

$$P(\mathbf{x};\, \theta) \;=\; \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1};\, \theta)$$

- The model parameters θ compress all knowledge seen during training
- At inference: given a prompt q, the model generates

$$y^* = \arg\max_{y}\; P(y \mid q;\, \theta)$$

- Generation is purely *parametric*: no external lookup, no memory

Vaswani et al., *Attention is All You Need*, NeurIPS 2017

Three fundamental limitations

Stale knowledge

θ is fixed after training. Any fact that changes or emerges after the training cutoff is inaccessible to the model.

Hallucination

$P(y \mid q; \theta)$ is a *probability distribution* over tokens, not a lookup table. The model generates fluent text even when it has no reliable basis for the claim.

No attribution

There is no mechanism to trace *which* training document caused a particular output making verification impossible.

A motivating example

Query

"What symbolic representation does Dash et al. use for molecules in their neurosymbolic drug discovery framework?"

LLM-only response (typical):

"Dash et al. likely represent molecules using SMILES strings fed into a graph neural network. . . "

- Sounds reasonable but may be entirely fabricated
- No source, no verifiability
- Worse: the paper may postdate the model's training cutoff

⇒ We need a way to supply the LLM with *relevant documents at query time*.

The RAG idea

Key insight: instead of compressing all knowledge into θ, maintain an *external knowledge base* $\mathcal{K}$ and retrieve from it on demand.

Standard LLM

$$y^* = \arg\max_y P(y \mid q;\, \theta)$$

Knowledge source: θ only

RAG

$$y^* = \arg\max_y P(y \mid q,\, \mathcal{D}_q;\, \theta)$$

Knowledge sources: θ *and* $\mathcal{D}_q$

where $\mathcal{D}_q = \textsc{Retrieve}(q,\, \mathcal{K},\, k)$ is a set of k document chunks retrieved from $\mathcal{K}$.

Lewis et al., *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*, NeurIPS 2020

Two phases of RAG

Phase 1: indexing (offline)

Build a searchable index over the knowledge base $\mathcal{K}$ once.

(i) **Chunking**: split documents into manageable passages $\{d_1, d_2, \ldots, d_N\}$

(ii) **Encoding**: map each chunk to a dense vector $\mathbf{e}_i = f_\phi(d_i) \in \mathbb{R}^m$

(iii) **Storing**: persist $\{\mathbf{e}_i\}$ in a vector database

Phase 2: query (online)

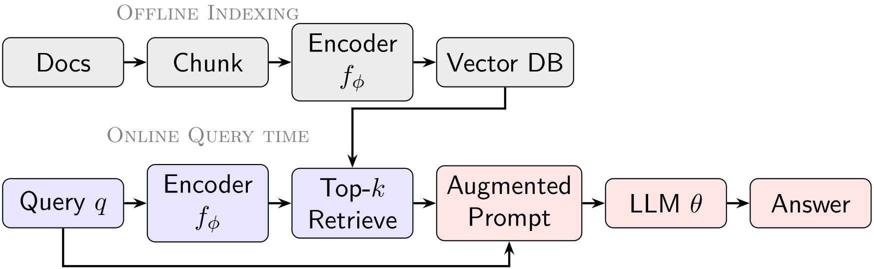
At inference time, for each user query q:

(i) **Retrieve**: find top-k chunks most similar to q

(ii) **Augment**: prepend retrieved chunks to the prompt

(iii) **Generate**: pass augmented prompt to the LLM

RAG pipeline: overview



Augmented prompt:

$$\underbrace{[\text{System instruction}]}_{\text{role/task}} + \underbrace{[\mathcal{D}_q]}_{\text{retrieved chunks}} + \underbrace{[q]}_{\text{user query}}$$

Dense retrieval: semantic embeddings

The encoder f_ϕ maps text to a dense vector in $\mathbb{R}^m$ such that *semantically similar* texts are geometrically close.

Cosine similarity between query q and chunk d_i:

$$\mathsf{sim}(q,\, d_i) = \frac{f_\phi(q)^\mathsf{T} f_\phi(d_i)}{\|f_\phi(q)\|\,\|f_\phi(d_i)\|} \;\in [-1,\, 1]$$

Retrieval: return the k chunks with highest similarity

$$\mathcal{D}_q = \operatorname*{arg\,max}_{S \subseteq \mathcal{K},\; |S|=k} \sum_{d_i \in S} \mathsf{sim}(q,\, d_i)$$

In practice: **FAISS** (Facebook AI Similarity Search) enables approximate nearest-neighbour search at scale.

Johnson et al., IEEE TBDM 2019

Sparse vs. dense vs. hybrid retrieval

Property	**Sparse (BM25)**	**Dense**	**Hybrid**
Representation	TF-IDF weights	$\mathbb{R}^m$ vector	Both
Matching	Exact term	Semantic	Both
Paraphrase recall	Poor	Good	Good
Rare terms	Good	Can miss	Good
Compute	Fast	Moderate	Higher

Reciprocal Rank Fusion (RRF) for hybrid scoring:

$$\text{score}_{\text{RRF}}(d_i) = \frac{1}{r_{\text{sparse}}(d_i) + c} + \frac{1}{r_{\text{dense}}(d_i) + c}$$

where $r(\cdot)$ is rank and c is a smoothing constant (typically 60).

Chunking strategy matters

The granularity of $\{d_1, \ldots, d_N\}$ directly affects retrieval quality.

Fixed-size

Split every w words with overlap δ. Simple but may cut sentences.

Sentence-based

Chunk at sentence boundaries. Semantically cleaner; variable length.

Recursive / hierarchical

Respect document structure (sections, paragraphs). Preserves context best.

Chunk size trade-off:

- *Too small*: retrieved chunk lacks context $\Rightarrow$ poor generation
- *Too large*: floods the LLM context window; retrieval precision drops

Conditional generation

Given retrieved chunks $\mathcal{D}_q = \{d_1^*, \ldots, d_k^*\}$, the LLM generates:

$$y^* = \arg\max_y \; P(y \mid q,\, d_1^*, \ldots, d_k^*;\; \theta)$$

Using the chain rule, this factorises token-by-token as before:

$$P(y \mid q, \mathcal{D}_q;\, \theta) = \prod_{t=1}^{|y|} P(y_t \mid y_{<t},\, q,\, \mathcal{D}_q;\, \theta)$$

Nothing changes in the LLM itself.
The retrieved chunks are simply prepended to the input; θ remains fixed during inference.

$\Rightarrow$ RAG is a *prompting strategy*, not a fine-tuning strategy.

The context window constraint

The LLM can only process a finite number of tokens at once (the *context window*).

Budget decomposition:

$$L_{\text{total}} = L_{\text{system}} + \underbrace{\sum_{i=1}^{k} |d_i^*|}_{\text{retrieved context}} + L_q + L_y \;\leq\; L_{\max}$$

Increasing k gives more grounding but:

- Consumes more of the context window
- Risks burying the relevant chunk among noise (*lost-in-the-middle* problem)
- Increases latency

Practical choice: $k \in \{3, 5\}$ for most research QA tasks.

Liu et al., *Lost in the Middle: How LMs Use Long Contexts*, TACL 2024

Reranking: improving retrieval precision

First-stage retrieval (cosine similarity) is fast but imprecise.

Reranker: a cross-encoder that scores each (q, d_i) pair jointly:

$$s_i = g_\psi(q,\, d_i) \;\in\; \mathbb{R}$$

- Sees *both* query and passage together $\Rightarrow$ much more accurate
- Too slow to run over all N chunks; applied only to top-k' from first stage

Two-stage pipeline:

$$\underbrace{\text{retrieve top-}k' \text{ by cosine}}_{\text{fast, bi-encoder}} \xrightarrow{\text{rerank}} \underbrace{\text{select top-}k}_{\text{accurate, cross-encoder}} \;\to\; \text{LLM}$$

Nogueira & Cho, *Passage Re-ranking with BERT*, arXiv 2019

RAG vs. fine-tuning vs. long context

Property	RAG	Fine-tune	Long context
Updates knowledge	Dynamic	Static	Dynamic
Trains θ	No	Yes	No
Scales to large $\mathcal{K}$	Yes	N/A	No
Attribution	Yes	No	Partial
Cost (inference)	Low	Low	High
Hallucination	Reduced	Moderate	Moderate

RAG is the preferred approach when the knowledge base is *large*, *dynamic*, or requires *traceable sources*.

How do we evaluate RAG?

Two sub-systems to evaluate independently and end-to-end:

Retrieval quality:

- *Recall@k*: fraction of relevant chunks in top-k retrieved
- *Mean Reciprocal Rank (MRR)*:
  $\text{MRR} = \frac{1}{|Q|} \sum_q \frac{1}{\text{rank of first relevant chunk}}$

Generation quality (RAGAS framework):

- *Faithfulness*: is the answer supported by $\mathcal{D}_q$?
- *Answer relevancy*: does the answer address q?
- *Context recall*: do retrieved chunks cover the ground-truth answer?

Es et al., *RAGAS: Automated Evaluation of RAG*, EACL 2024 (arXiv:2309.15217)

RAG inference

Input: query q, knowledge base $\mathcal{K}$, encoder f_ϕ, LLM θ, k

↓

Embed: $\mathbf{e}_q \leftarrow f_\phi(q)$

↓

Retrieve: $\mathcal{D}_q \leftarrow \textsc{TopK}(\mathbf{e}_q, \{\mathbf{e}_i\}, k)$

↓

Augment: $p \leftarrow$ [instruction]$\|\mathcal{D}_q\|q$

↓

Generate: $y^* \leftarrow \arg\max_y P(y \mid p; \theta)$

↓

Output: answer y^* with source attribution from $\mathcal{D}_q$

Demo: NeSy QA (click here)

Knowledge base $\mathcal{K}$: published papers on neurosymbolic AI and drug discovery.

Queries of interest:

- *"What symbolic representation is used for molecules?"*
- *"Which datasets are used for drug-likeness prediction?"*
- *"How is background knowledge encoded in the ILP framework?"*
- *"What is the performance vs. GNN baselines?"*

Comparison: LLM-only answer **vs.** RAG-augmented answer with source chunk displayed.

We will observe: LLM hallucinates method details; RAG cites exact text from the correct paper.