Retrieval-Augmented Generation

Embeddings · Vector Search · Chunking · Advanced RAG

Why RAG?

Hallucination
fluent but wrong

Knowledge cutoff
can't know new facts

No domain depth
generic, not expert

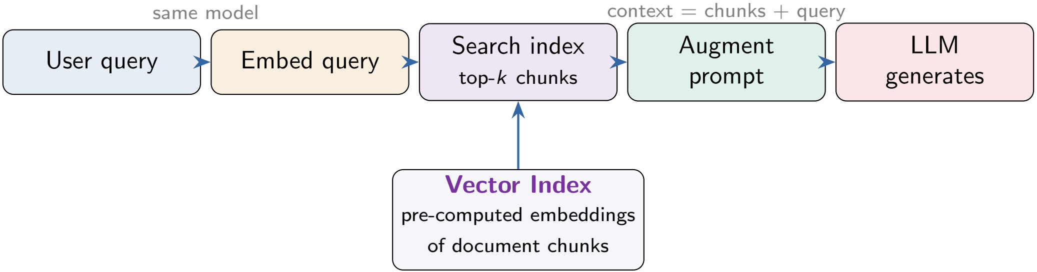
No source citation
"trust me, bro"

↓ **Solution**

RAG: give the LLM access to **external knowledge** at inference time

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

The Big Picture



Sparse vs Dense Retrieval

BM25 (Sparse)

Bag-of-words: exact term matching

$$\text{score} = \sum_i \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1(1 - b + b\frac{|D|}{\text{avgdl}})}$$

No training needed, fast

Fails on synonyms/paraphrases

Dense Retrieval (DPR)

Neural embeddings: semantic matching

$$p_\eta(z|x) \propto \exp\big(\mathbf{q}(x)^\top \mathbf{d}(z)\big)$$

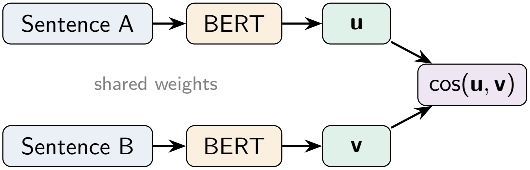
Captures meaning, not just words

Requires training data & compute

Hybrid search = BM25 + Dense $\rightarrow$ best of both worlds (Reciprocal Rank Fusion)

Embedding Models for Retrieval

Sentence-BERT (Reimers & Gurevych, 2019)



Model	**Dim**	**Ctx**
ada-002	1536	8K
embed-3-sm	1536	8K
E5-large	1024	512
BGE-M3	1024	8K
Nomic v1	768	8K
Cohere v3	1024	512

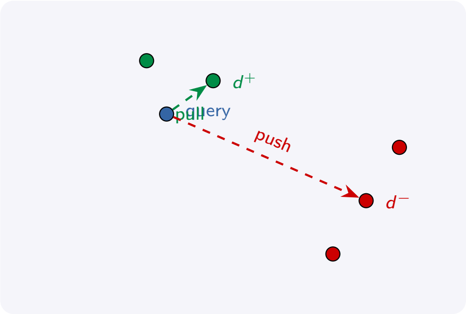
Cosine: $\frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|}$

Dot: $\mathbf{u}\cdot\mathbf{v}$

L2: $\sqrt{\sum(u_i - v_i)^2}$

Contrastive Training for Embeddings

Embedding Space



InfoNCE loss:

$$\mathcal{L} = -\log \frac{\exp\left(\mathrm{sim}(\mathbf{q}, \mathbf{d}^+)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\mathrm{sim}(\mathbf{q}, \mathbf{d}_j)/\tau\right)}$$

τ = temperature, N = batch (in-batch negatives)

Hard negatives are critical:
documents that look relevant but aren't

Used in: Sentence-BERT, DPR, SimCSE, CLIP

Vector Databases & ANN Search

1M vectors $\times$ 768 dims $=$ 3 GB — exact search is $O(n \cdot d)$ per query $\rightarrow$ need **ANN**

HNSW

Hierarchical Navigable Small World graph

$O(\log n)$ search

95–99% recall

high memory

IVF

k-means clusters

search nearest centroids

tunable n_{probe}

85–95% recall

moderate memory

Product Quantization

split vector into sub-vectors

quantize each to 1 byte

3072 B $\rightarrow$ 96 B

80–90% recall

very low memory

Vector Database Landscape

FAISS	Pinecone	Weaviate	Chroma	Qdrant
Meta, library	managed SaaS	hybrid search	lightweight	Rust, fast

Chunking Strategies

Document

paragraphs
sentences
sections
…

chunk →

Fixed-size
every N tokens

Sentence-based
split on "."

Recursive
\n\n → \n
→ "." → " "

Semantic
split when similarity drops

Structure-aware
headers, sections, HTML

Too small (<128 tokens)
missing context, fragmented

Sweet spot (256–1024 tokens)
precise retrieval + enough context

Too large (>2048 tokens)
diluted relevance, poor embeddings

Overlap: 10–20% of chunk size
prevents information loss at boundaries

e.g., 512-token chunks
with 50-token overlap

Phase 1: Indexing (Offline)

Load documents → Clean & preprocess → Chunk 512 tokens → Embed → vectors → Store in vector DB

each chunk stored with metadata: source, page, date, section

Example: 1,000 PDFs → recursive splitting at 512 tokens → ∼50,000 chunks → embed with E5-large (1024-dim) → ∼200 MB FAISS index

Phase 2: Query (Online)

User question → Embed query → Similarity search (top-k) → Re-rank cross-encoder → LLM generates

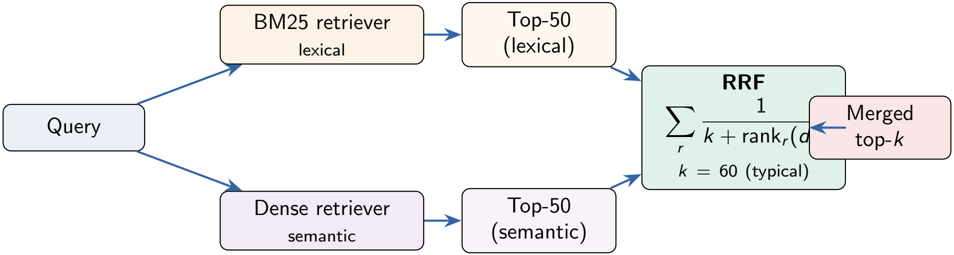
Bi-encoder (retrieval):
encode q and d independently
fast: pre-compute d embeddings
top-50 in $\sim$10 ms

$\xrightarrow{50}$

Cross-encoder (re-ranking):
encode (q, d) jointly through BERT
much more accurate but $O(n)$
re-rank 50 $\rightarrow$ top-5 in $\sim$100 ms

```
Context: {chunks}
Question: {query}
Answer from
context only.
```

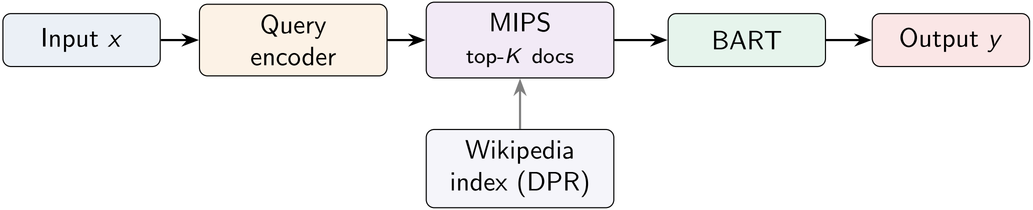
Hybrid Search & Reciprocal Rank Fusion



Hybrid consistently outperforms BM25-only **or** dense-only

BM25 catches exact keywords; dense catches synonyms/paraphrases

The Original RAG Paper (Lewis et al., 2020)



RAG-Sequence:

$$p(y|x) = \sum_z p(z|x) \prod_i p(y_i|x, z, y_{<i})$$

same document for entire sequence

RAG-Token:

$$p(y|x) = \prod_i \sum_z p(z|x)\, p(y_i|x, z, y_{<i})$$

can use different doc per token

Joint end-to-end training of retriever + generator.
SOTA on Natural Questions, TriviaQA, WebQuestions

Query Transformation: HyDE

Problem: query–document mismatch

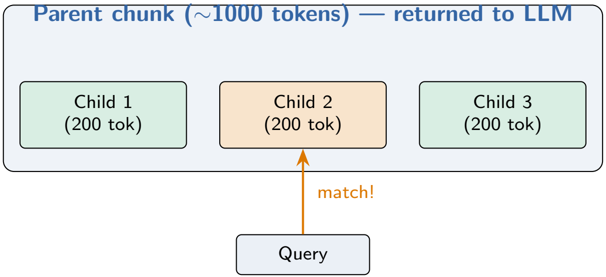
Short queries embed differently from long document passages

User query "What causes aurora?" → LLM generates hypothetical answer → Hypothetical document (may hallucinate) → Embed & retrieve real documents

Key insight: the hypothetical doc is in the same "linguistic space" as real documents
paragraph-length, descriptive → better embedding similarity with real passages

Gao et al. (2022). nDCG@10 = 61.3 on TREC DL-20 (vs. 44.5 for Contriever without labels)

Parent-Child Chunking



Small chunks for retrieval
precise embeddings, focused meaning

Large parent for generation
full context, surrounding paragraphs

Best of both worlds:
retrieval precision + generation context

Also called "small-to-big retrieval" — search over children, return the parent

Self-RAG (Asai et al., 2023)

Key idea: the model *itself* decides when to retrieve and evaluates what it retrieved

Generate text → [Retrieve?] →(Yes) Fetch passages → [IsRelevant?] → Generate candidates → [IsSupported?]

Reflection tokens

[Retrieve]: Yes / No
[IsRel]: relevant?
[IsSup]: supported?
[IsUse]: useful?

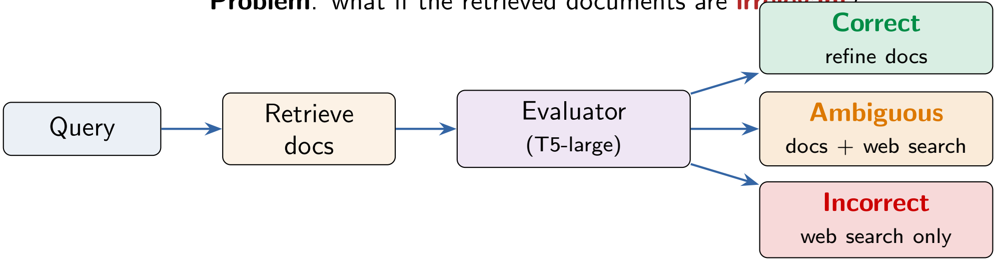
Training:

1. GPT-4 labels data with reflection tokens
2. Fine-tune Llama 2 to predict text + tokens
3. No separate retriever or critic at inference

Result: outperforms vanilla RAG and ChatGPT on factual benchmarks

Corrective RAG (Yan et al., 2024)

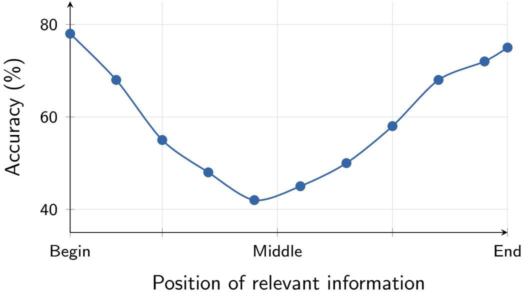
Problem: what if the retrieved documents are **irrelevant**?



Decompose-then-recompose: split docs into "knowledge strips," score each for relevance, filter out irrelevant strips, reassemble

CRAG: 61.8% on PopQA (vs. 54.9% for Self-RAG), 86.2 FactScore on biographies

Lost in the Middle (Liu et al., 2023)



LLMs attend strongly to the **beginning** and **end**

Information in the **middle** is often **missed**

RAG implication:
place most relevant chunks
at the **start** or **end** of the prompt

RAG vs Fine-Tuning

	RAG	**Fine-Tuning**
Knowledge type	facts, up-to-date info	style, format, reasoning
Update knowledge	update the index (easy)	retrain (expensive)
Latency	+100–500 ms overhead	no overhead
Hallucination	reduced (grounded)	can still hallucinate
Source citation	✓ can cite	× no attribution
Setup	vector DB + chunking + embedding	training pipeline + curated data
Best for	factual QA, docs, support	classification, style, extraction

Hybrid (increasingly common): fine-tune for task format + RAG for factual grounding
e.g., fine-tune Llama on medical QA format + RAG over PubMed abstracts

Evaluating RAG: RAGAS Framework

Context Precision
signal-to-noise of retrieved chunks

Context Recall
was all relevant info retrieved?

Retrieval

Faithfulness
are all claims supported by context?

Answer Relevance
does the answer address the question?

Generation

$$\textbf{Faithfulness} = \frac{\#\text{ claims supported by context}}{\#\text{ total claims in answer}}$$ (LLM-as-judge, no ground truth needed)

Common failure modes:

retrieval miss | context overflow | lost in the middle | unfaithful generation

Practical Considerations

Latency Breakdown

- Embed query: 10–50 ms
- Vector search: 5–20 ms
- Re-rank (50): 50–200 ms
- LLM generate: 500–3000 ms

RAG overhead ~100–500 ms

Cost

Embedding 1M chunks (OpenAI):
512M tokens $\times$ \$0.02/1M $\approx$ \$10

Storage (1M $\times$ 1024-dim):
~4 GB raw, ~8 GB with HNSW

Open-source (E5, BGE):
free (your GPU only)

Index freshness
incremental updates, versioning

Chunk quality
garbage in $\rightarrow$ garbage out

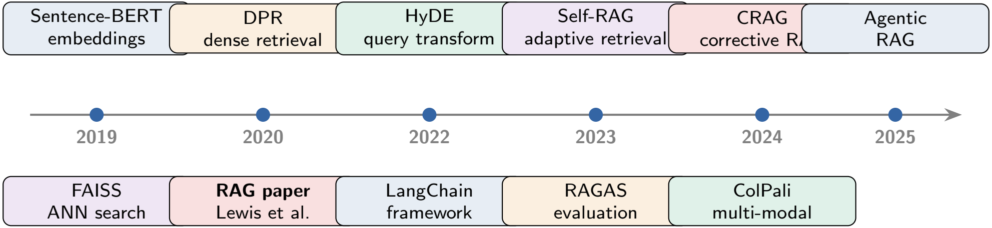
Multi-modal
tables, images, PDFs

Access control
who can see what?

Real-World Applications

- Customer support
- Legal search
- Medical QA
- Code assistants
- Enterprise search

RAG Evolution



Trend: simple retrieve-and-read $\rightarrow$ adaptive, self-correcting, multi-step, agentic

The RAG Playbook

Start Simple
BM25 or single embedding model + FAISS/Chroma

Chunk Well
512 tokens, recursive splitting
10–20% overlap

Add Hybrid
BM25 + dense retrieval with RRF merging

Re-rank
cross-encoder on top-50
much better precision

Evaluate
RAGAS metrics:
faithfulness + relevance

Go Advanced
HyDE, Self-RAG, CRAG
parent-child, agentic

Don't over-engineer from day one — iterate based on evaluation results

Further reading

Foundations

- Lewis et al. (2020), "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"
- Karpukhin et al. (2020), "Dense Passage Retrieval for Open-Domain Question Answering" (DPR)

Advanced RAG

- Asai et al. (2024), "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection"
- Gao et al. (2024), "Retrieval-Augmented Generation for Large Language Models: A

Embedding & Evaluation

- Muennighoff et al. (2023), "MTEB: Massive Text Embedding Benchmark"
- Es et al. (2024), "RAGAS: Automated Evaluation of Retrieval Augmented Generation"

Questions?

Next: Hallucination & Grounding