ARCHITECTURE · LLM SYSTEMS

The RAG pipeline, from indexing to answer

Retrieval-augmented generation hands a language model relevant passages at answer time. Its quality is decided first at retrieval: what is not found cannot be cited.

Intermediate · 10 min · RAG · embeddings · vector search · BM25 · Example — not reviewed by the author

Sheet
by Simon-Pierre Boucher

sheet.spboucher.ai/s/rag-pipeline
November 17, 2026

A language model only knows what its parameters hold: its memory is frozen at the end of training, and it cannot say where it came from. Retrieval-Augmented Generation (RAG) gives it a **searchable external memory**: before answering, the system fetches relevant passages and hands them to the model [1].

Parametric memory

The model's weights. Vast, but frozen, without provenance, and costly to update.

Non-parametric memory

An index of documents. It can be corrected or extended without retraining the model [1].

The link between them

A retriever that, for each question, selects the few passages to put in front of the model.

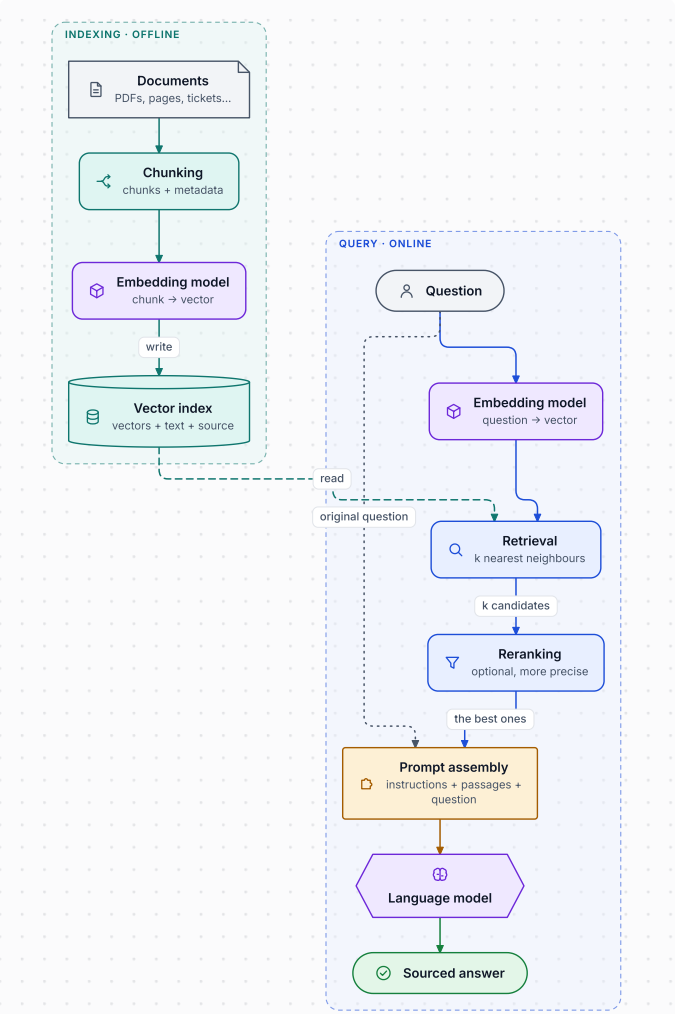
ASSUMPTION

Original RAG and everyday RAG

The 2020 paper trains a dense retriever and a sequence-to-sequence generator **jointly**, over a Wikipedia index [1]. Common practice, described here, is simpler: retriever and language model are used as they are, and the passages are inserted into the prompt [5]. The principle is the same.

01 Two phases, two moments

Indexing (offline) and query (online)



1. **Chunk** — Each document is split into chunks short enough to carry one idea, long enough to remain understandable out of context.
2. **Embed and store** — An embedding model maps each chunk to a vector; vector, text and provenance are stored in the index.
3. **Embed the question** — When a question arrives, the same model projects it into the same vector space.
4. **Retrieve** — The index returns the k chunks whose vectors are closest to the question's.
5. **Rerank** — A costlier model re-reads each question–chunk pair and keeps only the most relevant.
6. **Generate** — The prompt gathers the instructions, the selected passages and the question; the model writes the answer relying on those passages.

Indexing happens once per document; the query path runs again for every question. Both phases must use the **same** embedding model.

02 Finding the right passages

ƒ DENSE RETRIEVAL

$$\text{sim}(q, p) = E_Q(q)^\top E_P(p)$$

Question and passage are projected into one space; their closeness is measured by a dot product. Passage vectors are computed ahead of time, so a query reduces to a nearest-neighbour search [2].

q	Question
p	Passage (document chunk)
E_Q	Question encoder, returning a d-dimensional vector
E_P	Passage encoder, of the same dimension

Sources: 2

ƒ LEXICAL RETRIEVAL: BM25

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D)\,(k_1 + 1)}{f(q_i, D) + k_1 \left(1 - b + b\,\frac{|D|}{\text{avgdl}}\right)}$$

BM25 counts the question's words found in the document, giving more weight to rare words and correcting for length [3]. It remains hard to beat on exact terms: proper names, reference numbers, codes.

q_i	i-th term of the query Q	
$f(q_i, D)$	Number of occurrences of q_i in document D	
$\text{IDF}(q_i)$	Inverse document frequency: high for a term that is rare in the corpus	
$\lvert D \rvert$	Length of the document	words
avgdl	Average document length in the corpus	words
k_1, b	Free parameters: term-frequency saturation and length normalisation	

Sources: 3

ƒ RECALL AT K

$$\text{Recall@}k = \frac{|\mathcal{R} \cap \mathcal{T}_k|}{|\mathcal{R}|}$$

A measure of retrieval alone: of the passages that had to be found, what share appears in the top k? It is computed without calling the language model.

$\mathcal{R}$	Set of passages relevant to the question (hand-annotated)
$\mathcal{T}_k$	Set of the k passages returned by the retriever

Question–chunk similarity

O ILLUSTRATIVE DATA

	Returns	Shipping	Warranty	Battery	Careers
Return period?	0.86	0.41	0.33	0.12	0.05
Shipping fees?	0.38	0.84	0.15	0.10	0.06
Battery warranty?	0.29	0.13	0.81	0.77	0.04

0.04 — 0.86

Illustrative values, chosen by hand: they show what is expected of a good embedding model, not the output of a real one.

	Lexical (BM25)	Dense (embeddings)	Hybrid COMMON
MATCHES	the same words	the same meaning	both
STRENGTH	exact terms, identifiers, acronyms	paraphrases, synonyms	robustness
WEAKNESS	different vocabulary in question and document	rare terms or terms unseen in training	two indexes to maintain, score fusion to tune
MODEL REQUIRED	none	embedding model [4]	embedding model

VERDICT

The two families fail on different cases, which is why they are often combined, with a reranker doing the final sort [5].

WARNING

Evaluate the two halves separately

A bad answer has two possible causes: the right passage was not supplied, or it was and the model did not use it. Without separate measures for retrieval (recall at k) and generation (faithfulness to the passages), fixes are guesswork. **A passage that is not retrieved cannot be recovered by the model.**

03 Generating from the passages

ƒ RAG-SEQUENCE MODEL

$$p_{\text{RAG-Sequence}}(y \mid x) \approx \sum_{z \in \text{top-}k(p_\eta(\cdot|x))} p_\eta(z \mid x) \prod_{i=1}^{N} p_\theta(y_i \mid x, z, y_{1:i-1})$$

The retrieved passage z is treated as a latent variable: the probability of the answer is an average over the top k passages, weighted by their retrieval score [1]. Inserting the passages into a language model's prompt is a simplified version, in which the model reads all passages at once.

x	Input (the question)
y	Generated sequence, with tokens $y_1, \ldots, y_N$
z	Retrieved passage
$p_\eta(z \mid x)$	Retriever, with parameters η
p_θ	Generator, with parameters θ

Sources: 1

Instructions

Answer from the passages provided; say you do not know if the information is not there.

Numbered passages

Each passage carries an identifier and its provenance, so the answer can point back to it.

Question

The original question, as asked.

Cited answer

Each statement points to the passage that supports it, which makes the answer checkable.

04 Where the pipeline breaks

Typical failure modes, by stage. A summary of engineering practice [5].

STAGE	SYMPTOM	LIKELY CAUSE	WHAT TO TRY
Corpus	The model says “I don't know” or makes things up	The information is simply not in the index	Extend the corpus; make the model decline rather than invent
Chunking	Partial or out-of-context answer	The chunk cuts through a table, a list or an argument	Split along structure; overlap; attach the section title
Retrieval	The right passage exists but is not returned	Vocabulary gap, k too small, ambiguous question	Hybrid search; raise k; rewrite the question
Reranking	The right passage is retrieved, then dropped	Threshold too strict or context budget too short	Keep more passages; measure before and after reranking
Generation	The answer contradicts or ignores the passages	Conflicting or too many passages, or vague instructions	Fewer, better-sorted passages; require citations; check faithfulness
Index	Right yesterday, wrong today	Stale index or competing versions of one document	Incremental re-indexing; dates and versions as metadata

05 Check your understanding

QUESTION

A RAG assistant answers a question badly. The logs show that the passage holding the answer was indeed in the prompt. Where should you look first?

A. On the generation side: instructions, number and order of passages, conflicting passages.

B. At the embedding model, which should be replaced.

C. At chunking, by making the chunks smaller.

SHOW THE ANSWER

A — Correct. Correct: retrieval did its job; it is the use of the passage that failed.

B — Incorrect. The passage was found: the embedding is not at fault for this question.

C — Incorrect. Possible if the chunk was truncated, but not the first lead when the full answer is in the prompt.

Does RAG eliminate hallucinations?

No. It reduces them when the right passages are supplied and the model sticks to them; the original paper observes more factual and more specific generations than with a purely parametric model [1]. But the model can still misread a passage, wrongly combine two, or fill gaps from memory.

What chunk size should I use?

There is no universal value: it is a trade-off to measure on your own questions. Too small and the chunk loses its context; too large and its vector blends several topics, diluting the useful passage. Following the document's structure (sections, paragraphs) is a better starting point than a fixed number of characters.

Common mistake: changing the embedding model without re-indexing

Vectors from two different models do not live in the same space: comparing a question encoded by one with chunks encoded by the other is meaningless. Any change of embedding model requires recomputing the whole index.

Does a very large context window make RAG unnecessary?

For a small, stable corpus, putting everything in the prompt may be enough. Beyond that, retrieval remains useful: it bounds the cost per request, keeps a changing corpus up to date, enforces per-document access rights and shows where each piece of information comes from.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. arXiv:2005.11401. arxiv.org/abs/2005.11401

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[3] Stephen Robertson, Hugo Zaragoza (2009). *The Probabilistic Relevance Framework: BM25 and Beyond*. Foundations and Trends in Information Retrieval, 4(1–2), 1–174. doi:10.1561/1500000019

[4] Nils Reimers, Iryna Gurevych (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 3980–3990. doi:10.18653/v1/D19-1410

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv:2312.10997. arxiv.org/abs/2312.10997