From Hallucination to Grounded Answers

An Introduction to Retrieval-Augmented Generation.

Turn a confident guesser into a research tool you can **trust — and check**.

Nastaran Shahparian

SHARCNET · Compute Ontario · Digital Research Alliance of Canada

THE PROBLEM

A language model answers from memory — fluently, and sometimes completely wrong.

YOU

How many paid vacation days does our **employee handbook** give a new hire in year one?

LANGUAGE MODEL

New hires get 15 paid vacation days in their first year.

✗ **Invented — it never saw your handbook.**

It has never seen your handbook — but it answers anyway, **confidently**.
A confident wrong answer is worse than none.

THE FIX

Don't ask it to remember. Hand it the documents, then ask.

YOU

How many paid vacation days does our **employee handbook** give a new hire in year one?

📄 Employee Handbook · Leave Policy — retrieved

LANGUAGE MODEL

10 days in year one, rising to 20 after five years.

✓ **Grounded — Handbook §3.1**

Same question — but now it answers **only from the retrieved handbook**, and cites it. Its job shifts from *recall* to *reading comprehension*.

THE FAIR QUESTION

“Why not just upload the PDFs to ChatGPT?”

	UPLOAD TO A CHATBOT	RAG (YOUR OWN DOCS)
Scale	a few files	hundreds of papers
Reuse	re-upload every session	build the index once, query forever
Privacy	leaves your control	stays under your control
Sources	sometimes	every claim cited

Be honest: for 2–3 PDFs and one question, just upload them — it's simpler. RAG earns its keep at scale, reuse, and privacy.

THE NAME, DECODED

Retrieval-Augmented Generation.

R

Retrieval

Search your documents for the passages that bear on the question.

A

Augmented

Add those passages to the prompt — you *augment* what the model sees.

G

Generation

The model writes the answer, using only what was retrieved.

Retrieve the right context → **augment** the prompt → **generate** a grounded answer.

WHAT IT'S BEST FOR

Any body of text the model hasn't memorized.

📚 Literature review

survey a set of papers, with citations *(today's demo)*

🔒 Private & internal docs

reports, wikis, SOPs the model never saw

📋 Policies & handbooks

answer from the exact rulebook, not a paraphrase

💻 Code & large repos

find and explain code across a big codebase

🧬 Domain corpora

clinical notes, legal filings, field data — any text

🎓 Course material

a tutor grounded in your syllabus & readings

The common thread: knowledge that's **too new, too private, or too specific** for the model to have learned — but that you can hand it as documents.

IT'S A FRAMEWORK, NOT A PAPER-READER

Same pipeline — point it at any source.

🌐 Web page
docs, a wiki, a URL

🔎 Web search
live results on a topic

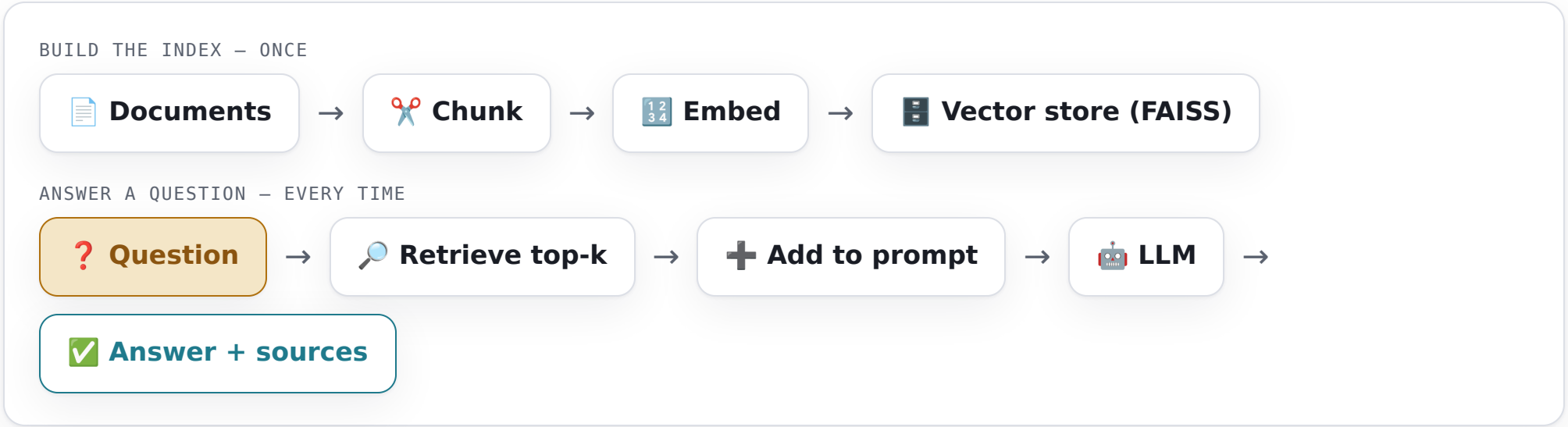
📁 Local folder
your PDFs, notes, drafts

📚 arXiv
by topic, or by ID

Point it at **your** documents — lab notes, theses, grant drafts, internal reports.

HOW IT WORKS

Six steps. Swap the input; the rest is identical.



Two concepts do the heavy lifting — chunking and embeddings. Coming up next.

CONCEPT 1 — CHUNKING

A document is too big to embed whole. Split it into overlapping windows.

Chunk 1 RAG splits each document into chunks. **The chunks overlap so an idea**

Chunk 2 **The chunks overlap so an idea** that spans a boundary is never lost. **Each chunk becomes a vector**

Chunk 3 **Each chunk becomes a vector** and is stored so we can search it later.

The highlighted overlap is deliberate: it keeps a sentence that crosses a boundary from being cut in half.

CONCEPT 1 — THE DETAILS

Two knobs: how big each chunk is, and how much they overlap.

📏 Chunk size — ~1000 chars

≈ 150–200 words. **Too big** → one vector must average many ideas, so retrieval turns blurry. **Too small** → ideas get fragmented and lose their context.

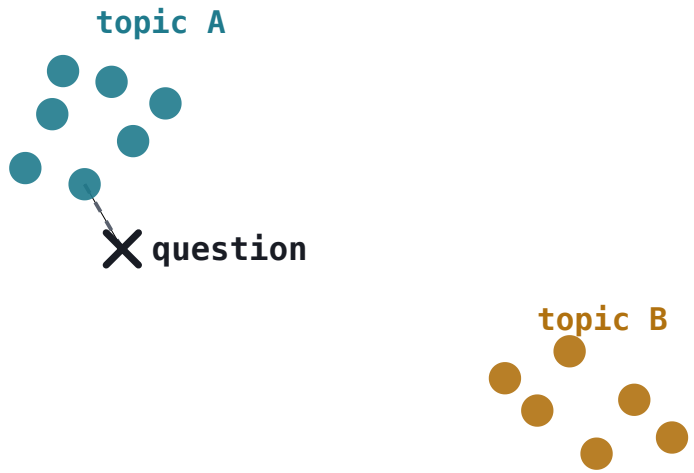
🔗 Overlap — ~200 chars

Each chunk repeats the tail of the one before it, so a sentence that crosses a boundary still lives *whole* in one chunk — retrieval won't cut it in half.

- **It splits on natural boundaries first** — paragraphs → lines → sentences → words — so a chunk rarely ends mid-word or mid-thought.
- **Example:** a 5,000-character section → about **6 chunks** of ~1000 chars, each sharing ~200 with its neighbour.

CONCEPT 2 — EMBEDDINGS

Every chunk becomes a point in space. Similar meaning → nearby points.

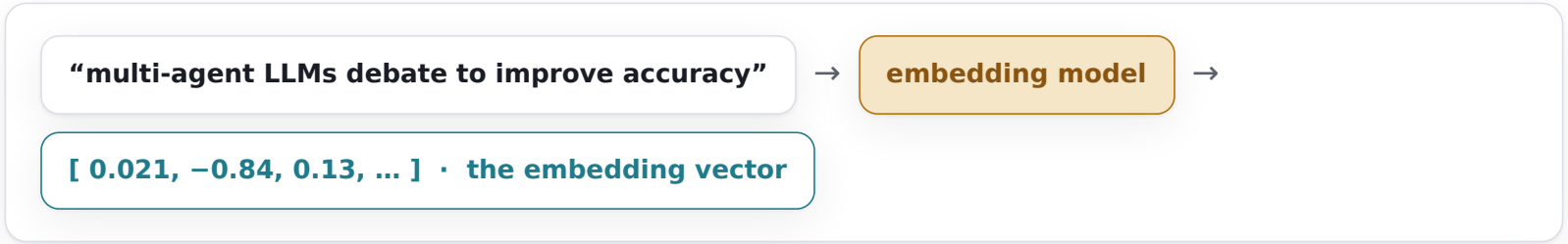


- Text about the **same topic** lands in the same neighborhood.
- A **question** is embedded the same way.
- Retrieval = **“find the nearest points to the question.”**

Model: **qwen3-embed** — a high-dimensional vector, flattened to 2D here.

CONCEPT 2 — UNDER THE HOOD

So what *is* an embedding?



- **Text in → a vector out.** The **embedding model** reads each chunk and returns a fixed-length list of numbers — the **embedding**. That list of numbers *is* what we store and search.
- **The numbers capture meaning, not spelling.** It's trained so text with similar *ideas* gets similar vectors — *“car” ≈ “automobile,”* even with no letters in common.
- **“Nearby” = cosine similarity** — the angle between two vectors. That one number is what ranks every chunk against your question (it's the score you saw on the map).
- **It lives in many dimensions, not 2.** The scatter flattens it to 2-D (PCA) just to draw it; the real search runs in the full space, inside a **vector store** (**FAISS**) — an index holding every chunk's vector — that finds the nearest ones in milliseconds.

CONCEPT 2 — SIMILARITY, IN NUMBERS

How do we measure “nearby”? Cosine similarity.

The cosine is the angle between two vectors — **1.0** = same direction (same meaning), **0** = unrelated:

```
cosine(A, B)  =  (A · B) / (|A| × |B|)     ← for normalised vectors it's just the dot product
A · B
```

```
toy 3-D example (real ones have hundreds to thousands of dimensions):

  "cat"       A = [1.0, 0.2, 0.0]
  "kitten"    B = [0.9, 0.3, 0.1]      cosine(A, B) ≈ 0.99   ← same idea, different word
  "airplane"  C = [0.1, 0.9, 0.4]      cosine(A, C) ≈ 0.28   ← unrelated
```

- Same meaning, different words → **high cosine**; unrelated → low. *That* is what “semantic” search means.
- The **question and the chunks are embedded by the same model**, into the same space — so they're directly comparable.
- Real embeddings are **high-dimensional**, and the range is compressed: a strong match is ~**0.6**, unrelated ~**0.4** — the cosine you saw on the map.

HOW DOES IT FIND THE RIGHT CHUNKS?

Two ways to search — by *words*, or by *meaning*.

You've chunked the docs. Now a question comes in — how do you pull the chunks that answer it?

🔤 Keyword search

Match the **literal words** — the classic method is **BM25**. **No embeddings.** Fast and exact, but it misses synonyms: search *“car”* and it won't find *“automobile.”*

🧠 Semantic search — what we use

Match the **meaning**, using the **embeddings** you just saw — *“car”* finds *“automobile.”* Nearest vector = most relevant chunk.

Everything on the last three slides — **embed → nearest by cosine** — *is* semantic search. Keyword search is the older alternative that skips it. This demo is semantic.

DEMONSTRATION · PART 1 — PROVE THE CONCEPT

First, give the model a document it cannot possibly know.

A synthetic M.Sc. thesis proposal — fictional student, fictional advisor, and an invented benchmark called **OrchBench**. Nothing in any training set.

The planted fact:

“OrchBench parametrically varies two task axes: decomposability and verifiability.”

Then we ask the same question two ways.

SAME MODEL · SAME QUESTION

“What is OrchBench, and what does it vary?”

× WITHOUT RAG — FROM MEMORY

“OrchBench is a benchmark suite for **container-orchestration systems** (Kubernetes, Docker Swarm)… it varies **task duration** and **task resource demand**.”

Fluent. Authoritative. Entirely invented — even a 120B model saw “Orch” and guessed Kubernetes.

✓ WITH RAG — FROM THE DOCUMENT

“OrchBench parametrically varies two task axes: **decomposability** and **verifiability**.”

→ Source: msc_thesis_proposal.md

It didn't refuse — it fabricated. RAG is the difference between a confident guess and a grounded, citable answer.

DEMONSTRATION · PART 2 — THE REAL PAYOFF

Now point it at real papers.

Seven works on multi-agent LLM systems, pulled straight from arXiv by ID — then ask across all of them at once.

AutoGen | MetaGPT | Multi-Agent Debate | Mixture-of-Agents | LLM Multi-Agents: a Survey | Why MAS Fail | Single- vs Multi-agent

✓ “DO MULTIPLE AGENTS BEAT A SINGLE MODEL — AND WHEN?”

“Under **equal compute**, single-agent systems can match or beat multi-agent ones on multi-hop reasoning **(Single- vs Multi-agent…)**; debate improves factuality **(Multi-Agent Debate)**; a mixture-of-agents raises quality at higher cost **(Mixture-of-Agents)**.”

DON'T EYEBALL IT — MEASURE IT

How do you know the pipeline actually works?

① Retrieval — did we *find* it?

- **Recall@k** — was the right source in the top-k *at all*? ← the headline
- **MRR · nDCG** — was it ranked near the top?
- **Diversity** — do the chunks span many papers, or repeat one? (MMR's job)

② Generation — did we *use* it right?

- **Faithfulness** — is the answer grounded in the chunks, not *hallucinated*?
- **Answer relevance** — did it actually answer the question?

Two layers: did we **find** the right chunk, and did the model **use** it faithfully? We score answers with an LLM-as-judge over a fixed set of test questions.

OUR PIPELINE, MEASURED

The scorecard — 11 test questions.

```
Retrieval — did we find it?
  Recall@12 ....... 0.86   right paper in top-12 (6/7)
  MRR ............. 0.57
  nDCG@12 ......... 0.57
  Precision@12 .... 0.14   (low is fine — multi-paper)
  Synthesis cov ... 0.88   of the expected papers
  Diversity ....... 5.5 papers / 12 chunks

Generation — did we use it?
  Faithfulness .... 0.86   grounded / honest
  Answer rel. ..... 0.70   (declines pull this down)

Honesty test
  “Tree-of-Thought?” (not in corpus) → declined ✓

Speed:  retrieval 0.10s   generation 24.4s
```

Grounded — but honest about gaps

Faithfulness 0.86 vs **Answer relevance 0.70** — the gap *is* the story.

On pinpoint-fact questions, retrieval finds the right **paper** but not always the exact **chunk**. So the model says **“I don't know”** rather than invent — faithful, but unanswered.

The fix for those: keyword / hybrid search to grab the buried chunk.

Real output from the eval cell. It finds the right papers, stays grounded — and refuses to make things up.

THE ENGINEERING

Small, reversible dials — each fixed a real failure.

LEVER	WHAT WE DID	WHY
Corpus	curate by ID, not keyword	relevance is everything (Lesson 1)
Retrieval	MMR · k = 12	spread across papers; no single doc dominates the context
Answer length	max_tokens = 2048	room to cover several papers *and* synthesize
Prompt	cite · compare · admit gaps	grounded, comparative, honest
Citations	title · author · year	every claim is checkable

UNDER THE HOOD — A RETRIEVAL CHOICE THAT MATTERS

Relevant *and* different: why we use MMR.

Plain similarity search grabs the chunks most like the question — which often means several chunks from the **same one or two papers**, all saying nearly the same thing. **MMR** asks for chunks that are relevant *and* varied.

PLAIN SIMILARITY

→ 2 papers

"AutoGen says… AutoGen says…" — one paper quoted five times.

MMR — WHAT WE USE

→ 6 papers

Coverage across the reading list — the answer *surveys* instead of parroting.

Same question, two settings — count the distinct papers each pulls. (Illustrative.)

LESSON 1 — CURATION IS 80% OF RAG

Garbage in, garbage out — visualized with real embeddings.

RAG is only as good as what you feed it. Point it at a **broad, open source** — a web search, or a vague keyword query — and it pulls in whatever *loosely* matches. The pipeline never judges relevance for you: it embeds *everything* you hand it, then at question time simply returns the chunks sitting nearest the question. Whatever you loaded is what the answer gets built from — near-misses and all.

Each dot = one real chunk, placed by its embedding vector (PCA to 2D). Hover to see the paper.

LESSON 2 — CITE, THEN CHECK

Trust, but verify.

We asked: *do multiple agents beat a single model?* For the **Multi-Agent Debate** paper, gpt-oss-120b answered: *“debates do not reliably produce more correct answers.”* So we opened the paper.

- ⚠️ **It's backwards.** The paper finds the *opposite* — debate outperforms the single model, in its own words *“significantly enhances … reasoning”* and *“improves the factual validity.”*
- The catch: the limitations it cited are all **real** (models “confidently affirm” wrong answers; fixate on recent turns). It reported the paper's *caveats* as its conclusion — real citations, backwards answer.

Grounded, cited — and still wrong. You catch it only by opening the source — that's what the citation is *for*.

LESSON 3 — THE HONEST GAP

Teach it to say “I don't know.”

? A QUESTION THE PAPERS DON'T COVER

“What causes inflation in an economy?”

→ “The provided sources are about multi-agent LLM systems — they don't cover inflation.”

That's not luck — it declines because **one line of the prompt** tells it to:

IN THE PROMPT

“If the context doesn't fully cover the question, say what's missing — don't guess.”

A truthful **“I don't know”** is the single thing that separates a research tool from a confident guesser.

CHOOSING THE RIGHT TOOL

RAG, fine-tuning, or just a longer prompt?

APPROACH	BEST FOR	NOT FOR
Paste into the prompt hand the model the text directly (like a ChatGPT upload)	a few documents, a one-off question	large or changing corpora — cost balloons, and facts get "lost in the middle"
RAG search your docs, feed only the relevant parts to the model	many / large / private / changing docs; answers that must be cited	changing the model's *style* or teaching a new skill
Fine-tuning train the model itself on examples to change how it behaves	a consistent format, tone, or skill	injecting facts — expensive, forgets, and can't cite

Rule of thumb: RAG adds knowledge; fine-tuning changes behaviour.

Different jobs — and you can combine them.

GETTING THE MOST OUT OF IT

Habits that make RAG work.

- **Pick the right documents.** A focused set beats a big grab-bag — relevance is most of the quality.
- **Ask specific questions.** "What does X say about Y" beats "tell me everything."
- **Always check the sources.** Citations exist to be verified — trust, then verify.
- **Diversify retrieval.** Don't let one document dominate; too few chunks miss, too many add noise.
- **Mind the chunk size.** ~500–1000 characters with overlap is a sane default.
- **Welcome "I don't know."** Don't force an answer the sources don't support.
- **Keep sensitive data local.** Run it where the documents are allowed to live.

Most "RAG is bad" experiences are really a **corpus** or a **question** problem — rarely the model.

THE HONEST LIMITS

What RAG can't do.

- **It isn't exhaustive.** It only sees the top-k chunks — it can miss a paper entirely.
- **Deep synthesis is capped by the model.** A small local model summarizes well; it doesn't reason like a senior reviewer.
- **A single short document doesn't need it.** Just read it — or paste it.

TO GO FURTHER

If you want to push it beyond the demo.

🔤 Hybrid search

meaning-search *plus* literal keyword-search — so exact acronyms, benchmark names, and gene IDs aren't missed

🥇 Reranking

a small second model re-reads the shortlist and re-scores it — fast search finds candidates, a careful reader picks finalists

🧠 Try other models

the endpoint is a drop-in menu — swap the model name to try a larger or reasoning-tuned model, a one-line change

📏 Measure it

don't eyeball quality — score faithfulness & coverage on a fixed question set (e.g. RAGAS)

🏷️ Metadata filters

restrict retrieval by year, author, or venue *before* searching

🔁 Multi-step retrieval

for hard questions: retrieve, reason, then retrieve again (agentic RAG)

⚡ Approximate index

the demo's exact flat search is ideal for thousands of chunks; at **millions**, swap in an approximate index (**HNSW**, **IVF**) — a little accuracy for big speed

🧩 RAG frameworks

you built this in **LangChain**; **LlamaIndex** is the other leading choice (more indexing-focused) — **same workflow, different API**

All optional. The demo pipeline is deliberately simple — these are the dials you reach for when the corpus grows or the questions get harder.

TAKE THESE HOME

Four principles.

01

Ground it

answer from documents, not memory

02

Curate it

relevance beats every clever trick

03

Cite it

make every claim checkable

04

Let it pass

a truthful “I don't know” builds trust

Grounded, cited, and honest. RAG turns a fluent guesser into a research assistant whose every claim traces back to a source you can open and check.

WRAP-UP

Thank you — questions?

🚀 Try it yourself — SHARCNET AIaaS

The models in this demo run on SHARCNET's **AI-Inference-as-a-Service (AIaaS)** pilot. Open to users on request — **send us an email to get access**.

📓 Get the code from:

```
https://staff.sharcnet.ca/nast/langchain_demo_endpoint.ipynb
```

Nastaran Shahparian

SHARCNET · Compute Ontario · Digital Research Alliance of Canada