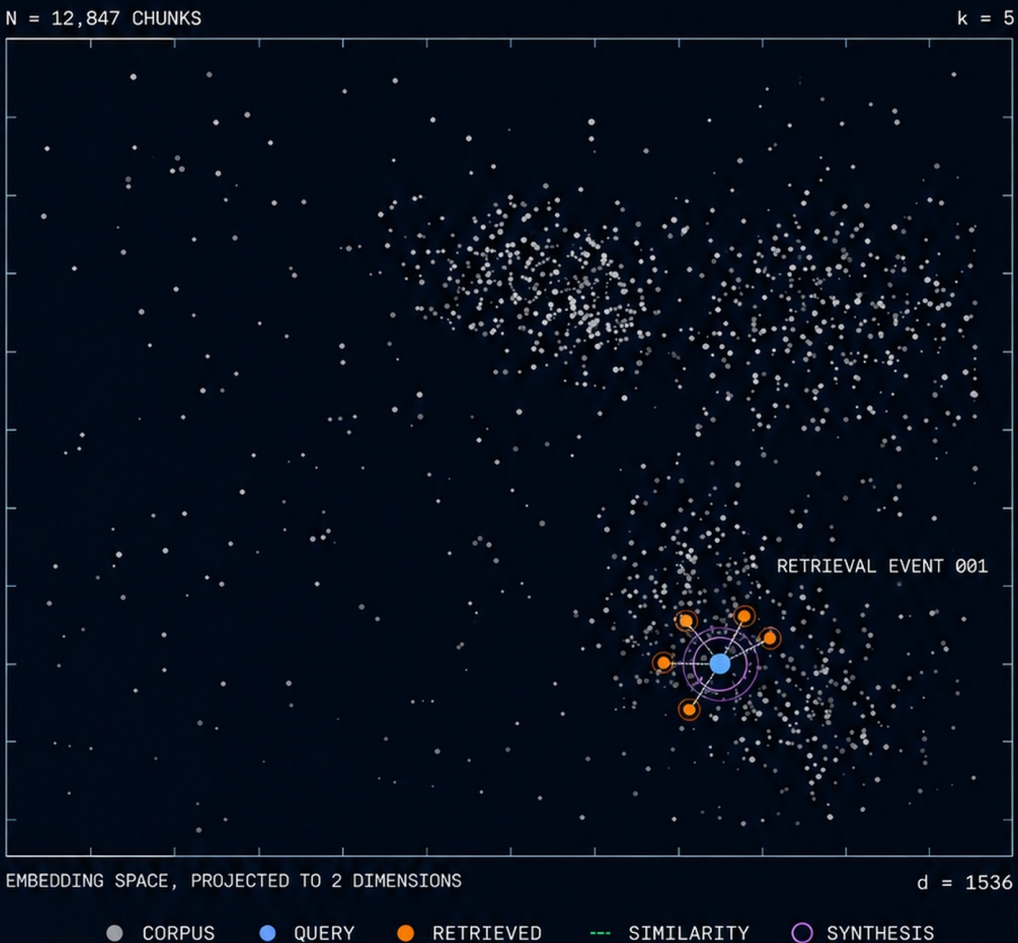
THE FROM SCRATCH SERIES

BUILDING AI THAT KNOWS YOUR DATA

RAG FROM SCRATCH · VOLUME I · FOUNDATIONS



FREE SAMPLE · CHAPTER ONE · THE GHOST IN THE FILING CABINET

JC MARIN

AN ORGANIZATION KNOWS MORE THAN ANYONE IN IT

01

The Ghost in the Filing Cabinet

PART I · THE PROBLEM · CODING REQUIRED: NONE

§ TUESDAY, MARCH, FORTY THOUSAND DOLLARS

On a Tuesday morning in March, Dana Okafor, chief executive of Northwind Industries, asked a question that would eventually cost her company forty thousand dollars.

“Didn’t we handle a dispute almost exactly like this in 2019?”

Nobody knew. Rachel, the company’s most experienced operations manager, thought she remembered something. Marcus in IT was sure the files existed *somewhere*. Northwind had been scanning and saving everything for thirty years, with the enthusiasm of a family that never throws anything away and the organizational system of one too. There were 1.4 million documents on the company’s servers: contracts, memos, emails, invoices, handbooks, engineering drawings, support tickets. Every answer the company had ever produced was in there.

Rachel spent three days searching. She tried the shared drive's search bar. She tried keywords: the customer's industry, the supplier's name, the type of claim. She opened folder after folder with names like `2019_ARCHIVE_FINAL_v2`, which as every office worker knows is never final and never version 2. She found the file on day three, filed under the *customer's parent company's* name, which nobody had thought to search for because nobody knew the customer had a parent company.

Three days of a senior manager's time. And Dana's question had been a *softball*. She knew the file existed, roughly when it happened, and what it was about. Most questions are harder: "Have we ever agreed to a liability cap below one million?" To answer that one, someone would have to read every contract the company ever signed.

Nobody ever answers that question. It's cheaper to guess, and guessing is instant, free, and only occasionally catastrophic.

That is the ghost in the filing cabinet: an organization that *knows* things no individual person can *find*. This series is about building the machine that finds them. By the end of this trilogy, you will have built a system where Dana types her question and gets an answer, with citations to the exact documents, in about two seconds.

But we're not going to start with that machine. We're going to start with why it's so hard to build.

· · ·

§ THE PROBLEM

Every organization suffers from the same strange disease: **it knows more than anyone in it.**

The knowledge is real. It's written down. It survived. But it's spread across a million files, in a hundred formats, organized by whoever was closest to the filing cabinet that day. The organization's memory is enormous, and almost completely unusable. It's a genius with amnesia.

Here's the uncomfortable math. Suppose a Northwind employee needs a piece of information that exists in the archive:

- If they know **exactly which document** holds it, they find it in minutes.
- If they know **roughly where to look**, it takes hours.
- If they only know **that it probably exists**, it takes days. Or, more honestly, it takes never.

Nearly all valuable questions live in that third category. "Have we seen this clause before?" "What did we tell the last customer who asked about this?" "Who negotiated our best supplier deal in this region?" The answers exist. The cost of finding them is just higher than the value of knowing, so institutional knowledge quietly dies in storage, like a houseplant in a conference room.

This isn't a Northwind problem. It's a hospital problem, a law-firm problem, a software-company problem, a government problem. Any organization older than a few years has a ghost.

> **A definition, gently.** Throughout this book we'll call a collection of documents a *corpus*. That's Latin for "body," and it just means "all the documents we care about." Northwind's corpus is its 1.4 million files. Yours might be a folder of PDFs. Both count, and only one of them will make you cry.

§ THE NAIVE SOLUTION

Every beginner, and every enterprise IT department, tries the same first fix: **keyword search**. Index every word in every document. When someone searches "liability cap," return every document containing the words *liability* and *cap*. This is how the shared-drive search bar works. It's how search worked for fifty years, and fifty years of anything earns some respect.

And to be fair: keyword search is fast, cheap, and sometimes exactly right. When Rachel searches for a customer's exact name, keywords shine.

So Marcus sets up a proper keyword index over Northwind's corpus. Search time drops from days to seconds. Marcus updates his resume. Problem solved?

§ WHY IT FAILS

Two weeks later, Rachel is back in Marcus's office with a list. Rachel keeps lists. Each item reveals a crack in the foundation.

Crack 1: People don't use the same words. Rachel searches "liability cap." The contract says "limitation of liability." Zero matches on documents that are *precisely* what she wanted. Lawyers say *indemnification*; everyone else says *who pays if this goes wrong*. Keyword search doesn't know these are the same idea, because keyword search doesn't know anything about ideas. It matches *letters*, not *meaning*.

Crack 2: The same words mean different things. Rachel searches "termination." She gets employment terminations, contract terminations, lease terminations, and one memo about terminating a software license,

written with surprising emotion. The word matched perfectly. The meaning didn't.

Crack 3: Questions aren't keywords. Dana doesn't think in search terms. She thinks in questions: "Have we ever agreed to a liability cap below one million?" A keyword engine can't be asked anything. It can only be handed words and return documents containing them. The gap between *a question* and *a bag of words* is where most of the value leaks out.

Crack 4: Finding is not answering. Even when search works, it returns *documents*. Forty of them, each sixty pages. Rachel's real job was never "find documents." It was "answer Dana's question." The last mile, the reading and understanding and synthesizing, is still entirely human, still entirely slow.

Write those four cracks down somewhere. The entire rest of this book is a response to them. Cracks 1 through 3 are about **matching meaning instead of letters**. That's *retrieval*, and it will occupy us through Volume I and much of Volume II. Crack 4 is about **turning found text into answers**. That's *generation*, and it's where language models enter the story.

· · ·

§ THE BETTER SOLUTION

In late 2022, something shifted. Large language models (we'll spend all of Chapter 2 building an honest intuition for what they are) became good enough to do two things that sound like magic:

1. **Understand meaning.** Given "liability cap" and "limitation of liability," a language model *knows* these are the same idea. Not because someone programmed a synonym list, but for reasons we'll unpack carefully in Chapter 10.

2. **Answer in plain language.** Given a question and some relevant text, a model can read the text and compose an answer, with the pages cited.

The naive reaction is: "Great, the AI knows everything, just ask it!" That reaction is wrong in an important way, and Chapter 3 is entirely about *why* it's wrong. The short version: the model has never seen Northwind's documents, and when it doesn't know something, it doesn't say "I don't know." It improvises, confidently, like a student who didn't do the reading but has excellent posture.

The *engineering* reaction is different, and it's the idea this whole trilogy is built on:

> **Don't teach the model your documents. Teach your system to find the right documents and hand them to the model at the moment of the question.**

That architecture has a name: **Retrieval-Augmented Generation**, or RAG for short. *Retrieval*: find the handful of passages that matter. *Augmented*: attach them to the question. *Generation*: let the language model compose an answer from what it was handed.

---

§ MENTAL MODEL

Picture the world's best new employee. Brilliant, fast, beautifully spoken. Also: they started this morning. They know *language* and *the world in general*, and absolutely nothing about your company, including where the bathroom is.

You wouldn't fix that by making them memorize 1.4 million documents. You'd fix it the way organizations have always fixed it: when a question comes in, a **librarian** pulls the three relevant folders and puts them on the new employee's desk. The employee reads three folders, not a million, and writes the answer.

RAG is exactly this, mechanized:

- The **librarian** is the *retriever*. Its whole job is knowing where things live. (Volumes I and II are mostly about building a superhuman librarian.)
- The **folders** are *retrieved passages*: small, relevant excerpts, not whole archives.
- The **new employee** is the *language model*. Eloquent, general, and totally dependent on what lands on its desk.

One sentence to carry with you for the next thousand pages: **a RAG system is only as good as what the librarian puts on the desk.** When RAG fails, and in Chapter 28 we'll dissect the ways it fails, the eloquent employee is almost never the problem. The desk is.

· · ·

§ INTERNAL MECHANICS

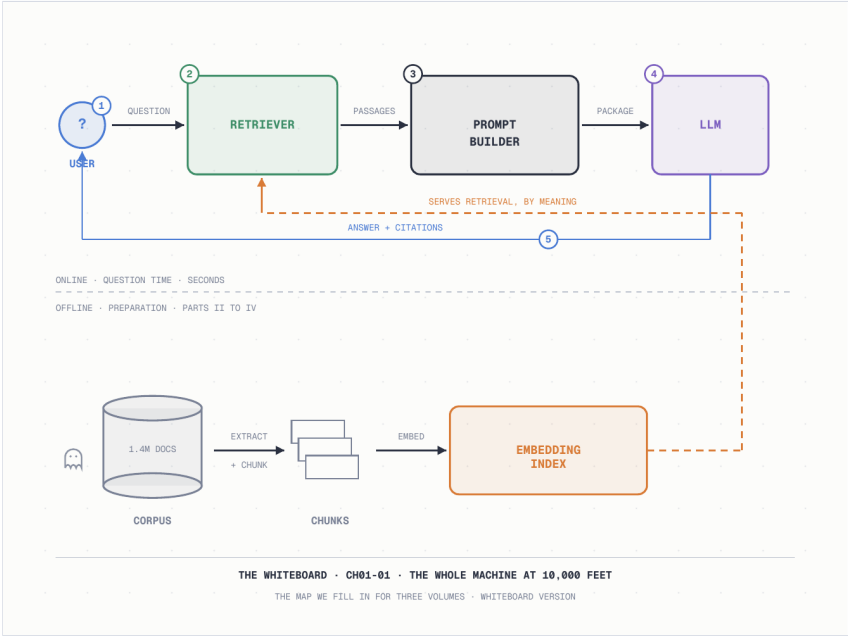
We'll build every box of this pipeline ourselves, but here's the whole machine at 10,000 feet. This is the map we'll spend three volumes filling in. When a question arrives, it flows left to right:

1. **The question** ("Have we ever agreed to a liability cap below $1M?") arrives from a user.
2. The **retriever** searches the corpus, by meaning rather than just keywords, and pulls back the most relevant handful of passages. Not documents; *passages*. (Why passages and not whole documents turns out to matter enormously. That's Chapter 8.)
3. A **prompt builder** assembles a package: the question, the passages, and instructions like *answer only from these excerpts, and cite them*.
4. The **language model** reads the package and writes the answer.

5. The answer returns **with citations**, because the system knows exactly which passages it handed over.

Before the first question can ever be asked, there's an offline preparation phase: reading the corpus, cutting it into passages, and building the "meaning index" that makes step 2 possible. That preparation is Volume I's whole story: extraction (Part II), chunking (Part III), embeddings and indexing (Part IV).

§ VISUAL DIAGRAM



Study the two halves. The top lane runs in seconds, every time someone asks a question. The bottom lane runs ahead of time, and it is where Volume I lives: extracting (Part II), chunking (Part III), and embedding into an index (Part IV). The orange dashed line is the whole trick: the index built offline is what lets the retriever search by meaning when the clock is

running. This exact map reappears at the start of every Part with the components you've built filled in. The production rendering debuts in Chapter 4.

· · ·

§ CODE WALKTHROUGH

None yet, and that's a promise kept, not a corner cut. Part I is 0% code by design. You cannot debug a system you can't picture, so Part I's job is the picture. Your only "implementation" this chapter is in the Build Log below, and it requires nothing but a mouse and the ability to right-click.

§ PRODUCTION VERSION

There's no production code yet either, but there *is* a production mindset to install on day one. Real systems answering real questions for real organizations differ from demos in three ways you should start noticing now:

- **Scale:** the gap between "works on 10 PDFs" and "works on 1.4 million documents" is not a bigger server. It's different architecture. We'll feel this gap personally around Chapter 13.
- **Trust:** Dana will not act on an answer she can't verify. Citations aren't a feature; they're the product. (Chapter 20.)
- **Permissions:** some Northwind documents are for executives only. A search system that shows everyone everything is not a productivity tool. It's a lawsuit generator with a nice interface. (Chapter 37.)

§ COMMON MISTAKES

Mistake 1: "We'll just use ChatGPT." The model has never read your documents, and it improvises when it doesn't know. Asking a public chatbot about Northwind's contracts produces confident, cited-sounding, entirely fabricated answers. Chapter 3 shows this failure in slow motion, and it's a little chilling.

Mistake 2: "We'll fine-tune the model on our documents." Plausible, popular, and almost always wrong as a first move: expensive, stale the moment a document changes, and unable to cite sources. We'll give fine-tuning a fair, full hearing in Volume III, where it will make its case and lose most of it.

Mistake 3: "Better keyword search will get us there." Keyword search is a *component* of great RAG (Chapter 33 brings it back with honor), but it cannot cross cracks 1 through 4 alone. Letters aren't meaning.

Mistake 4: Starting with a framework. Install-a-library tutorials get you a demo in an afternoon and a system you can't debug by Friday. In this book you build every component by hand first; frameworks come at the end (Chapter 41), when you can read them with open eyes and, frankly, a little judgment.

· · ·

§ CHAPTER SUMMARY

- Organizations know more than anyone in them. The knowledge exists but can't be found. That's the ghost.
- The cost of *finding* usually exceeds the value of *knowing*, so institutional memory quietly dies in storage.
- A collection of documents we care about is called a *corpus*.

- Keyword search matches letters, not meaning. Vocabulary mismatch and ambiguity break it.
- Questions are not keywords, and finding documents is not answering questions.
- Language models understand meaning and write fluent answers, but know nothing about *your* corpus, and improvise when ignorant.
- RAG: retrieve the relevant passages, attach them to the question, generate an answer from them.
- Mental model: a brilliant new employee (the LLM) plus a superhuman librarian (the retriever).
- A RAG system is only as good as what the librarian puts on the desk.
- The pipeline: question → retriever → prompt builder → model → cited answer, built on an offline preparation phase.

§ VOCABULARY

corpus · *retrieval* · *generation* · *RAG* · *retriever* · *passage* · *citation* · *pipeline*

§ EXERCISES

Simple. Think of the last time you needed information you *knew* existed at your job or school but couldn't find. Which of the four cracks stopped you?

Intermediate. Write down five real questions someone at your organization might ask its archive. For each, mark: could keyword search answer it? Could a person with unlimited time? Keep this list. In Chapter 23 you'll ask your own system all five, and settling old scores is excellent motivation.

Advanced. Interview one person who's been at any organization ten-plus years. Ask: "What does the organization know that nobody can find

anymore?" Write half a page on what the answer implies about where the *retriever* in their heads lives today.

---

§ QUIZ

1. Why does keyword search fail on "liability cap" vs. "limitation of liability"?
2. In the mental model, what maps to the retriever: the employee or the librarian?
3. What are the three words behind the acronym RAG, and what does each contribute?
4. True or false: RAG works by training the model on your documents. (Explain.)

---

§ BUILD LOG · ENTRY 001

Every chapter ends here, at the running project. It begins today, humbly.

1. Create a folder called `northwind` anywhere on your computer.
2. Inside it, create a folder called `corpus`.
3. Download the Northwind Industries starter pack (companion resources). For now it contains a single file: `employee_handbook.pdf`, 47 pages.
4. Place it in `corpus`.

```
northwind/
└── corpus/
    └── employee_handbook.pdf
```

That's the entire system: one folder, one PDF, zero code. It cannot answer questions. It cannot even read its own document. To a computer, that PDF is currently a sealed envelope. Opening the envelope is Chapter 5.

By the end of this trilogy (Chapter 39, to be exact) this folder will have become a clustered platform serving an entire company. It will never be restarted from scratch. Everything we build, we build onto *this*. Try to feel a little sentimental about it. You two are going to go through a lot together.

Next: before we can put anything on the new employee's desk, we need to understand the employee. Chapter 2: how a language model actually thinks.

END OF THE FREE SAMPLE

THE STORY CONTINUES

CHAPTER TWO

THE AUTOCOMPLETE THAT ATE THE WORLD

MARCUS DEMOS THE LETTER MACHINE.
DANA ASKS HOW IT KNOWS.
THE HONEST ANSWER CHANGES EVERYTHING.

GET THE FULL VOLUME AT

LEANPUB.COM/LEARN2RAG

BUILDING AI THAT KNOWS YOUR DATA

RAG FROM SCRATCH · VOLUME I · FOUNDATIONS

JC MARIN

AN ORGANIZATION KNOWS MORE THAN ANYONE IN IT