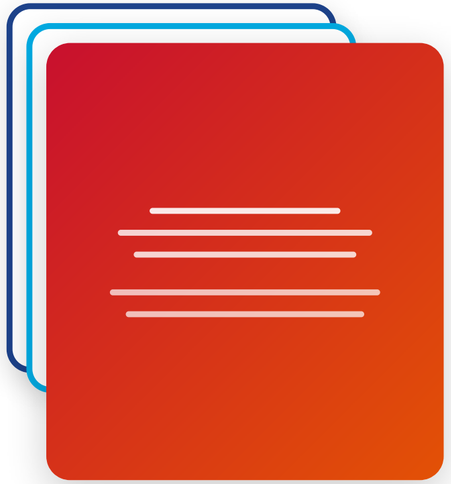
Context Engineering

Building AI Systems with Your Own Data

UBUS 670 | AI for Business Leaders

Day 3 • Week 1 • Spring 2026



Today's Learning Objectives

By the end of today, you will be able to:

1. Explain what context engineering is and why it matters for business AI
2. Recognize structured formats (Markdown, JSON, XML) and when structure improves AI results
3. Describe how RAG, embeddings, and vector databases work conceptually
4. Apply context engineering using Gemini Gems and document uploads

Today's Skill:

Context Engineering

From prompts → systems

Section 1

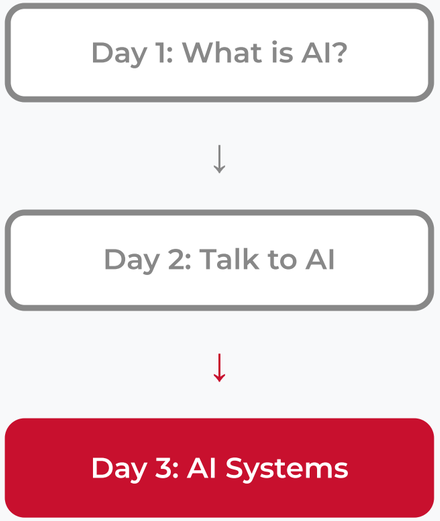
From Prompts to Context

Quick Recap: What We Learned About Prompts

Last time we mastered Prompt Engineering:

- RCTFC framework — Role, Context, Task, Format, Constraints
- Zero-shot, few-shot, chain-of-thought — techniques for different tasks
- Iteration — refining prompts systematically
- Safety — prompt injection defenses

The gap: Good prompts are powerful, but they have limits. What if your AI needs 100 pages of company context?



The Limitation of Prompts Alone

Imagine hiring a brilliant consultant but only giving them a one-sentence brief.

What a prompt CAN do:

- Ask specific questions
- Give clear instructions
- Provide a few examples
- Set output format

What a prompt CAN'T do (easily):

- Include 50+ pages of policies
- Reference entire databases
- Keep track of ongoing projects
- Remember previous interactions

The reality: Most business AI tasks need more information than fits in a single prompt.

What is Context Engineering?

Definition

Context Engineering is the practice of preparing and organizing all the information an AI system needs to perform tasks effectively — not just the immediate question, but the full background, documents, instructions, and constraints.

It's about building an information environment for AI, not just asking questions.

Prompt Engineering

What you say to the AI

Context Engineering

What you give the AI to work with

The Context Stack

Effective AI systems combine multiple layers of context:

1. Prompt
The immediate question or instruction

2. Documents
Reference materials: policies, reports, data

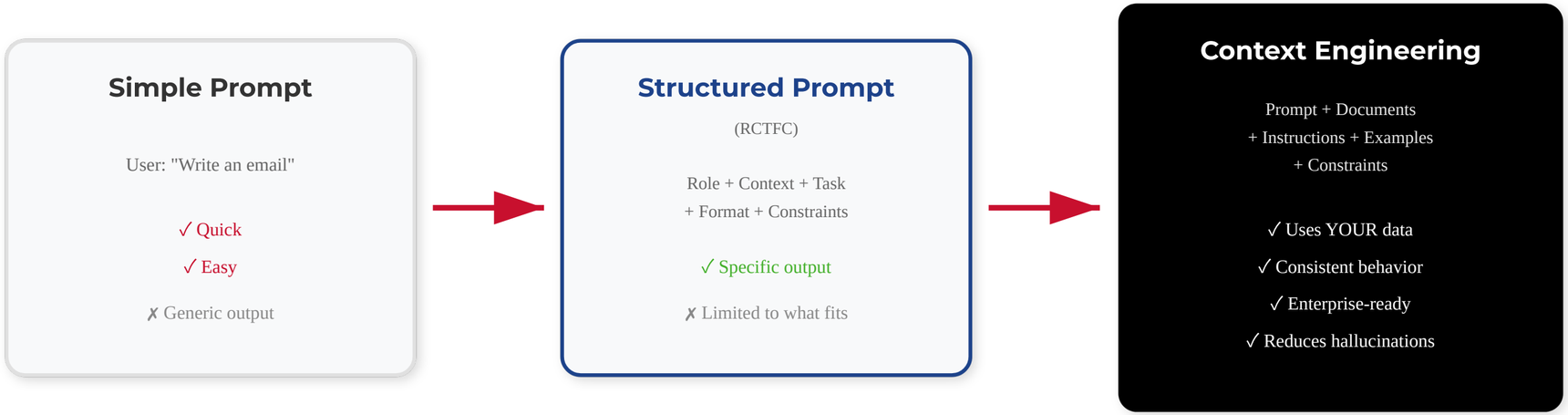
3. Instructions
System-level guidelines on how to behave

4. Examples
Sample inputs and outputs to learn from

5. Constraints
Rules, boundaries, and safety guidelines

Together, these layers create a comprehensive context that enables sophisticated AI applications.

The Evolution of AI Interaction



Real Business Example: Beacon's HR Challenge

The Problem

Beacon's HR team needs to screen 200 seasonal applications. They want AI help, but the AI needs to understand:

- Company culture and values
- Job requirements across 3 roles
- Legal hiring guidelines
- Brand voice for communications
- Previous successful candidate profiles

That's 50+ pages of context. Too much for one prompt.

The Solution: Context Engineering

1. Upload documents: Employee handbook, job descriptions, legal guidelines
2. Set instructions: "Screen for culture fit and required skills"
3. Provide examples: 3 strong applications from last year
4. Add constraints: "Never make final decisions; flag for human review"

When Should You Use Context Engineering?

Not every AI task needs context engineering. Use this decision matrix:

Scenario	Approach	Why
One-off question using public knowledge	**Simple Prompt**	No context needed
Repeated Q&A about internal policies	**Context Engineering**	Reusable knowledge base saves time
50+ pages of required reading	**Context Engineering**	Exceeds prompt capacity
Real-time data (stock prices, live feeds)	**API Integration**	Uploaded docs go stale fast
Specialized tone/jargon (legal, medical)	**Fine-Tuning + RAG**	Both deep knowledge and current docs needed

> Rule of thumb: If you find yourself pasting the same background information into multiple prompts, it's time for context engineering.

Discussion: What Context Would YOU Need?

Imagine it's your first day at a new company.

What documents and information would you need to do your job effectively?

Think about...

Policies, procedures, past projects, team structure, customer data, brand guidelines, etc.

Key insight

AI systems need the same onboarding materials you would. Context engineering is like creating an employee orientation packet for AI.

The gap

Most companies have this information scattered across emails, drives, and people's heads. Context engineering requires organization.

Section 2

Structure, Formats & Chunking

Structure Helps AI Think Better

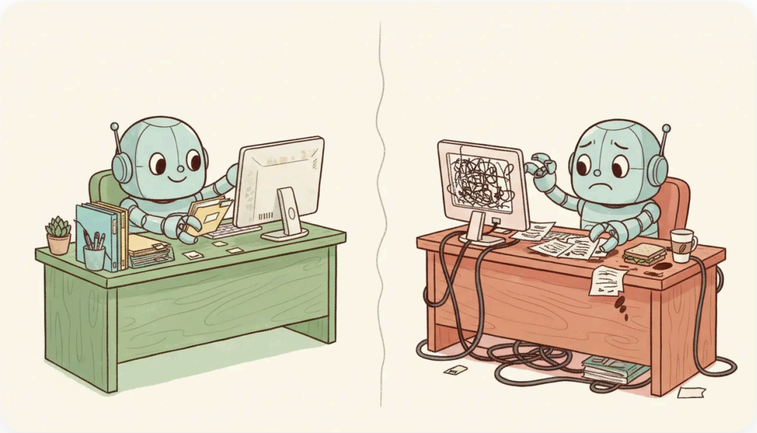
Modern AI can read almost anything — scanned PDFs, handwritten notes, messy spreadsheets. But structured input produces better, more consistent results.

What AI handles well in 2026:

- Scanned PDFs and images (built-in OCR)
- Complex tables and multi-column layouts
- Handwritten text
- Mixed-format documents

But structure still helps because:

- AI finds information faster
- Answers are more precise and consistent
- Less ambiguity = fewer mistakes
- Easier for YOU to verify the output



Analogy: Think of an organized desk vs. a messy desk. You can find things on both — but you'll find them faster and more reliably on the organized one.

Meet the Formats: Markdown

Markdown is a simple way to add structure to plain text. You may already use some of these patterns without knowing it.

Plain Text (Unstructured)

```
Beacon Retail Group Vacation Policy

Full-time employees get 15 days per year.
Part-time employees get 5 days per year.
Request through HR portal.
Manager approval required.
Contact HR@beacon.com for questions.
```

With Markdown (Structured)

```
## Vacation Policy

**Applies to:** Full-time employees
**Annual days:** 15 per year
**Request process:** Submit via HR portal
**Advance notice:** 2 weeks minimum
**Approval:** Direct manager

### Part-Time Employees
**Annual days:** 5 per year
**Same process as above**

**Contact:** HR@beacon.com
```

Why better: Clear headings (##), bold labels, explicit values, self-contained sections

What is Markdown?

Markdown is a lightweight formatting language. # = heading, **bold** = bold, - = bullet point. Many AI tools natively understand it. You'll see it used throughout business and tech.

Meet the Formats: JSON & XML

Two other structured formats you should recognize. You don't need to write these — just know what they look like.

JSON (JavaScript Object Notation)

```
{
  "employee": "Sarah Chen",
  "department": "HR",
  "role": "Manager",
  "stores": [1, 5, 12],
  "full_time": true
}
```

Used by: APIs, web apps, AI tool configurations

XML (Extensible Markup Language)

```
<employee>
  <name>Sarah Chen</name>
  <department>HR</department>
  <role>Manager</role>
  <stores>1, 5, 12</stores>
  <fullTime>true</fullTime>
</employee>
```

Used by: Enterprise systems, government, healthcare

Key point: Both formats use key-value pairs to organize data. When your tech team says "the AI returns JSON" — now you know what that means.

When Does Structure Matter Most?

Not every interaction needs structured input. Here's a practical guide:

Scenario	Structure Needed?	Why
Quick question ("What's the capital of France?")	**No**	AI handles conversational queries well
Analyzing a 30-page report	**Helpful**	Headings help AI navigate sections
Multi-step instructions for an AI workflow	**Yes**	Numbered steps prevent skipping
Data with relationships (employees, stores, roles)	**Yes**	Tables/JSON preserve connections
Uploading a photo of a receipt	**No**	Multimodal AI reads images natively

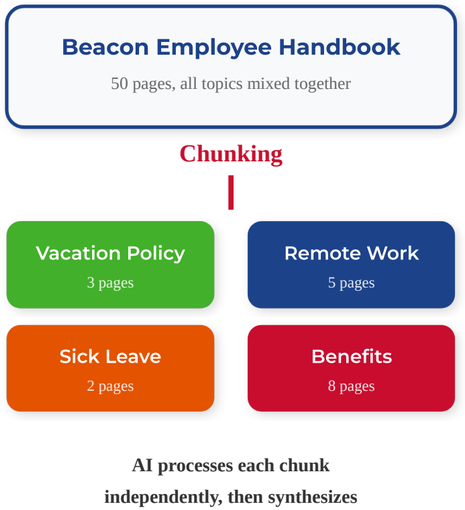
> Rule of thumb: The longer and more complex the input, the more structure helps. For quick questions, just ask naturally.

Chunking: Breaking Documents into Pieces

Even with large context windows (Gemini 2.5 supports up to 1 million tokens via Google AI Studio), smart chunking produces better results.

No-code example: NotebookLM

Google's NotebookLM automatically chunks your uploaded documents and enables question-answering. When you upload a PDF, it's doing chunking behind the scenes — you just ask questions.



AI processes each chunk independently, then synthesizes

How to Chunk Documents: Three Strategies

1. By Semantic Section

Keep related ideas together. Split at natural topic boundaries (chapters, headings).

Example: Employee handbook → "Vacation Policy" chunk, "Sick Leave" chunk, "Remote Work" chunk

Best for most business documents

2. By Fixed Size

Split every 500–1,000 tokens. Simple but can cut ideas mid-sentence.

Tip: Add 100-token overlap between chunks so no idea is lost at the boundary.

Use when docs lack clear structure

3. By Task Relevance

Only include what's needed for the current task. Different questions need different chunks.

Example: HR question? Upload HR docs only. Finance question? Upload financial docs only.

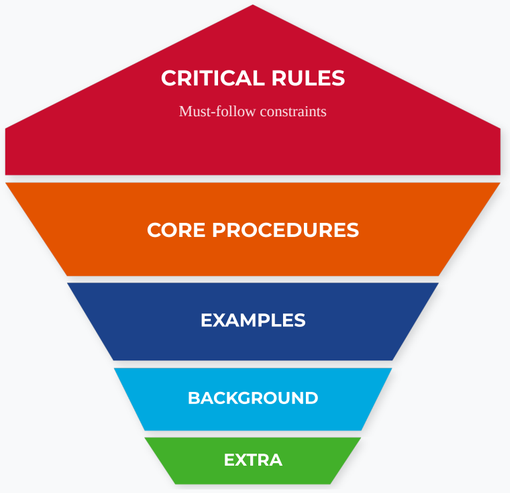
Fastest and cheapest approach

Priority Ordering: Most Important Context First

AI attention fades over long documents. Information at the beginning and end is weighted more heavily than the middle.

Best Practice: Inverted Pyramid

1. Critical rules — must-follow constraints
2. Core procedures — how to do the task
3. Examples — reference cases
4. Background — context and history
5. Appendices — supplementary info



Interactive: The Structure Challenge

Take this unstructured Beacon text and add Markdown formatting. Then compare AI responses with and without structure.

Unstructured (Your Starting Point)

```
Beacon has 25 stores and 1200 employees.
The CEO is Maria Torres. Revenue was
$180M last year. The biggest challenge
is seasonal hiring - they need 300 temp
workers for holiday season. HR uses
paper applications still. Finance wants
better expense tracking. Marketing needs
customer data insights.
```

Your Task: Add Markdown

```
## Company Overview
**Name:** Beacon Retail Group
**CEO:** Maria Torres
**Stores:** 25 | **Employees:** 1,200
**Revenue:** $180M (last fiscal year)

## Key Challenges
1. **Seasonal hiring** - 300 temp workers
2. **Expense tracking** - manual process
3. **Customer insights** - data underused
```

Try it: Paste both versions into Gemini and ask "What are Beacon's top 3 priorities?" — compare the responses. Did structure change the quality?

Section 3

RAG: Retrieval-Augmented Generation

What is RAG?

Definition

RAG (Retrieval-Augmented Generation) is an AI technique where the system first retrieves relevant information from a document collection, then uses that information to generate an informed answer.

Think of it as a two-step process:

Step 1: Retrieval

Search your documents for the most relevant chunks of information related to the query.

Step 2: Generation

Feed those relevant chunks to the LLM along with the query, and generate an answer based on that context.

Key benefit: The AI doesn't need to "memorize" everything. It looks it up when needed, just like you would reference a manual.

How does retrieval actually work?

The system converts your question and document chunks into mathematical representations (called embeddings) that capture meaning, not just keywords. When you ask "What's our return policy?", it finds chunks about returns even if they use words like "refunds" or "exchanges" — because the meaning is similar. This is called semantic search, and it's why RAG is much more powerful than Ctrl+F.

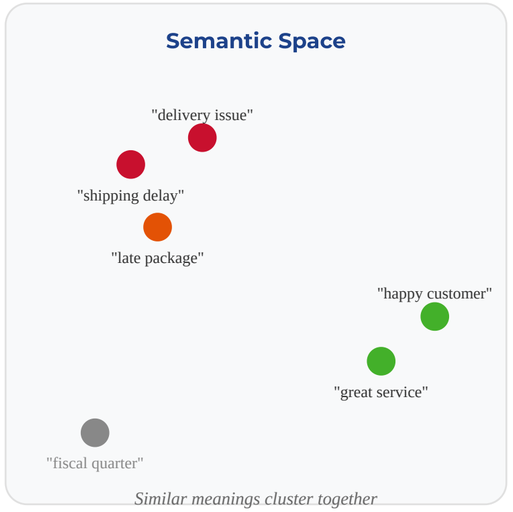
Embeddings: How AI Understands Meaning

In Day 1, we learned that AI reads text as tokens. Now let's go deeper: AI converts those tokens into embeddings — mathematical representations that capture meaning.

Definition

Embedding = a list of numbers (a "vector") that represents the meaning of a word, sentence, or document. Similar meanings produce similar numbers.

Why this matters for business: Embeddings are what power semantic search, recommendation systems, and RAG. When you search for "customer complaints about shipping delays" and the system finds documents about "delivery timeline issues" — that's embeddings at work.



Embeddings & Vector Databases

Two Types of Embeddings

	Dense	Sparse
Captures	Meaning & context	Keywords & exact terms
Good for	"Find similar ideas"	"Find exact matches"
Example	"customer complaints about shipping"	"shipping delay refund"
Used by	Modern AI, RAG	Traditional search

Vector Databases: The AI Filing Cabinet

Embeddings need a special kind of database that can search by meaning, not just keywords. These are called vector databases.

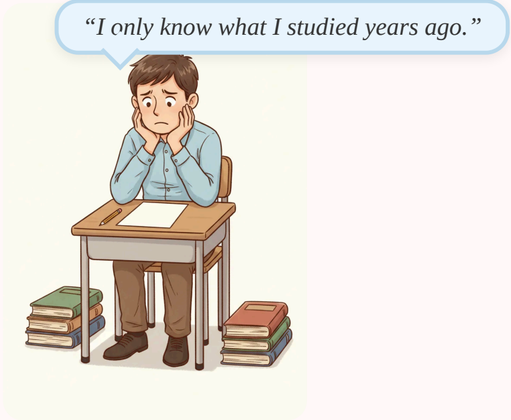
Products to know: Pinecone, Weaviate, Chroma

> Business analogy: A regular filing cabinet finds files by label. A vector database finds files by what they're about — even if the labels are different.

You don't need to build one — but your tech team will. And products like Gemini Gems use them behind the scenes.

Analogy: Open-Book vs. Closed-Book Exam

CLOSED-BOOK EXAM (Traditional AI)



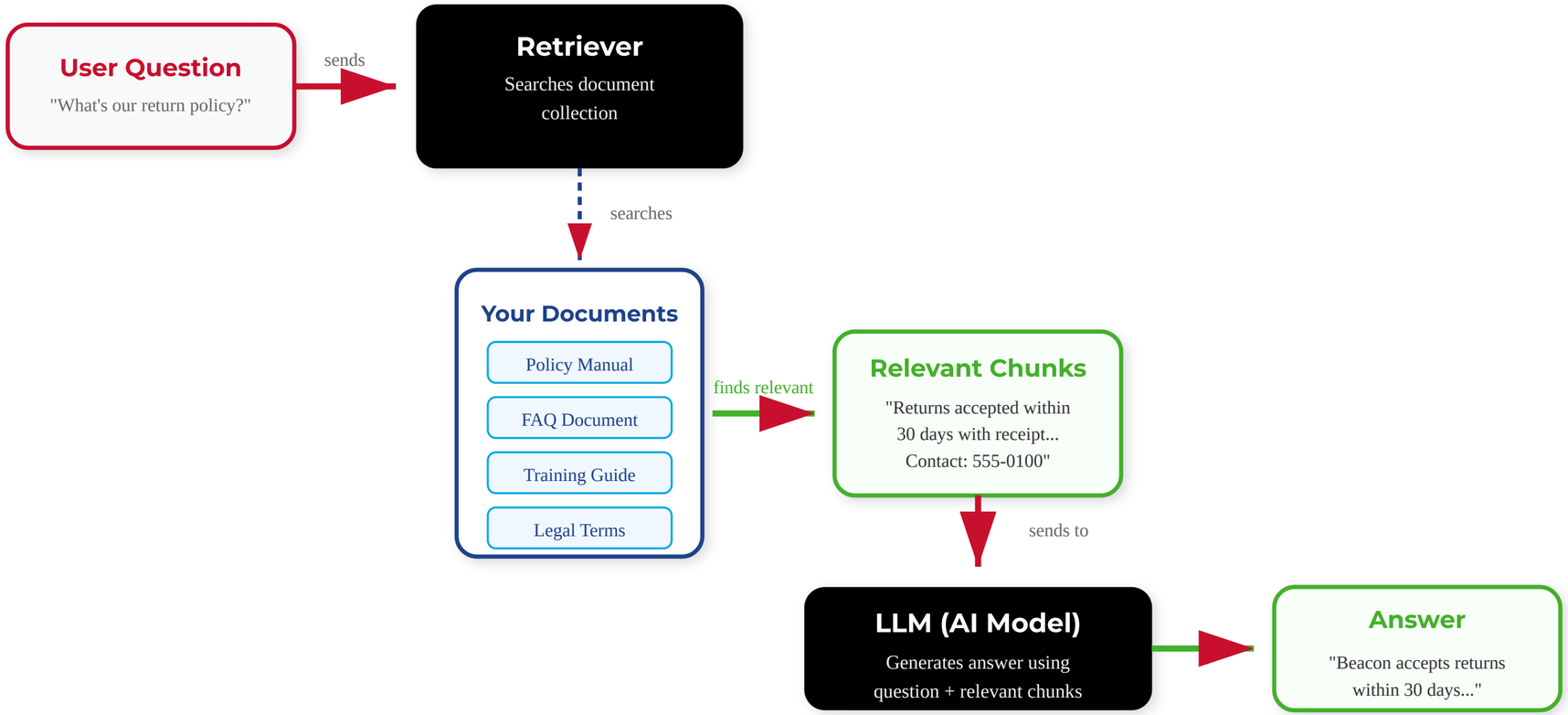
- **Relies on "memorized" training data**
- **No new info access**
- **Hallucinations if unsure**
- **Knowledge cutoff date**

OPEN-BOOK EXAM (RAG System)

- **Real-time look-up**
- **Uses YOUR proprietary data**
- **More accurate, fewer hallucinations**
- **Always current (as your docs)**

The insight: RAG turns AI from a "know-it-all" into a "know-where-to-look-it-up" system. Much more practical for business.

How RAG Works: Simplified Architecture



The retriever finds relevant information, then the LLM generates a natural language answer based on that context.

Why RAG Matters for Business

Reduces Hallucinations

By grounding answers in real documents, RAG dramatically reduces made-up information. The AI cites what it finds.

Keeps AI Current

Update your documents, update the AI's knowledge. No retraining needed. Your Q4 sales data? Available immediately.

Uses YOUR Company Data

RAG works with proprietary information that wasn't in the model's training data. Your policies, your customers, your products.

Transparent & Auditable

Many RAG systems show which documents were used to generate an answer. Critical for compliance and trust.

Business impact: RAG is the difference between a chatbot that "sounds smart" and an AI system that's actually useful for your specific organization.

RAG vs. Fine-Tuning: A Quick Comparison

	RAG	Fine-Tuning
Purpose	Add knowledge	Change behavior
Analogy	Giving someone a reference book	Training someone over months
Cost	**Low (just add documents)**	**High (compute + data + expertise)**
To update	Swap documents anytime	Retrain from scratch
2026 trend	**Growing (most use cases)**	Shrinking (general models are smart enough)

The 2026 consensus: For most business applications, RAG is the right choice. General models like Gemini 2.5 and Claude are now smart enough that fine-tuning is only needed for highly specialized domains.

Story: The "Accidental Fine-Tuner"

Dr. Sarah Chen, an accounting professor, wanted to train auditors to be better listeners.

She created a Custom ChatGPT (similar to a Gemini Gem) and started training it through conversation:

The AI kept doing this:

```
Student: "I'm confused about depreciation."
AI: "Here's how to calculate depreciation: Step 1..."
```

She wanted this instead:

```
Student: "I'm confused about depreciation."
AI: "I hear you. Tell me more about what's confusing — is it
the concept or the calculation?"
```

She repeatedly "chastised" the AI: "Stop solving! Just listen and empathize." Over weeks of interaction, it became a remarkably good listener.

The "Accidental Fine-Tuner" (Part 2)

Sarah's "empathetic auditor trainer" was so effective that she wanted to replicate and deploy it.

She hired a software developer to "extract the fine-tuned weights and embeddings" from her Custom ChatGPT and build a standalone application.

But when the developer looked inside...

there were no modified weights to extract.

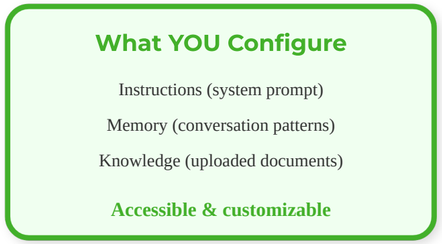
What Really Happened (Teaching Moment)

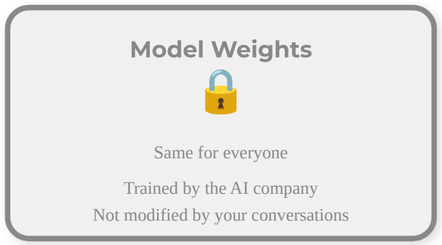
Custom ChatGPTs (and Gemini Gems) do NOT modify model weights. They use:

1. Instructions — a system prompt telling the AI how to behave
2. Memory — stored conversation patterns and preferences
3. RAG — uploaded documents as reference knowledge

Key Takeaway

Behavioral configuration ≠ fine-tuning. And that's actually good news — it means YOU can do this too, without any technical expertise!





Limitations of RAG (It's Not Magic)

Garbage In = Garbage Out

If your documents are poorly written, outdated, or inaccurate, RAG will retrieve and use that bad information.

Retrieval Quality Matters

The system has to find the right chunks. If retrieval fails (wrong keywords, poor indexing), the answer will be wrong.

No Deep Reasoning

RAG is great for factual Q&A, but complex reasoning across multiple documents is still challenging.

Key Insight

RAG amplifies your documentation quality. If your documents are messy, RAG makes that problem more visible. Use RAG as motivation to improve your knowledge management.

Section 4

Context Tools in Practice

How Context Tools Work Today

Modern AI platforms offer built-in context engineering features. Here's the basic workflow:

1

Upload Documents

PDFs, text files, spreadsheets to the AI platform

2

Index Content

System automatically processes and indexes the documents for search

3

Query with Context

Ask questions and the AI retrieves relevant sections before answering

Gemini Gems: Your Personal AI Assistant

Google Gemini offers two ways to use context engineering. In today's lab, you'll try both:

Option 1: Upload to Chat

- Attach files directly in a conversation
- Quick and easy for one-time questions
- Documents are temporary — lost when chat ends

Good for: ad-hoc analysis, quick questions

Option 2: Gemini Gem

- Create a custom AI with persistent Knowledge
- Upload up to 10 files that stay across conversations
- Write custom instructions (system prompt)
- Share via link with colleagues

Good for: reusable assistants, team tools

Gems are free! Available on personal Google accounts. No paid subscription required. This is no-code RAG in action.

Demo: Creating a Gemini Gem

Scenario: Build a "Beacon Knowledge Assistant" Gem for the HR team.

1. Go to gemini.google.com
2. Click "Gem manager" in the left sidebar
3. Click "New Gem"
4. Name it: "Beacon Knowledge Assistant"
5. Write instructions: "You are Beacon Retail Group's knowledge assistant. Answer using only the uploaded documents. Always cite which document your answer comes from."
6. Upload Knowledge: Company Overview, Org Chart, Financials, Strategic Plan, CEO Memo
7. Save and test: "How many stores does Beacon operate?"

Share your Gem!

Click the three-dot menu next to your Gem, then Share > Copy link. Send it to a classmate — they can use your Gem instantly.

Tips for Effective Context Engineering

1. Label Your Documents

When uploading, provide context about the context: "This is Beacon's 2025 financial summary" not just "Summary.pdf"

2. Start Specific

Upload only relevant documents first. Add more if needed. Too much context can dilute retrieval quality.

3. Test Retrieval

Ask questions you know the answers to first. Verify the system is finding the right information.

4. Combine with RCTFC

Context engineering + prompt engineering = powerful. Use RCTFC in your queries even with uploaded docs.

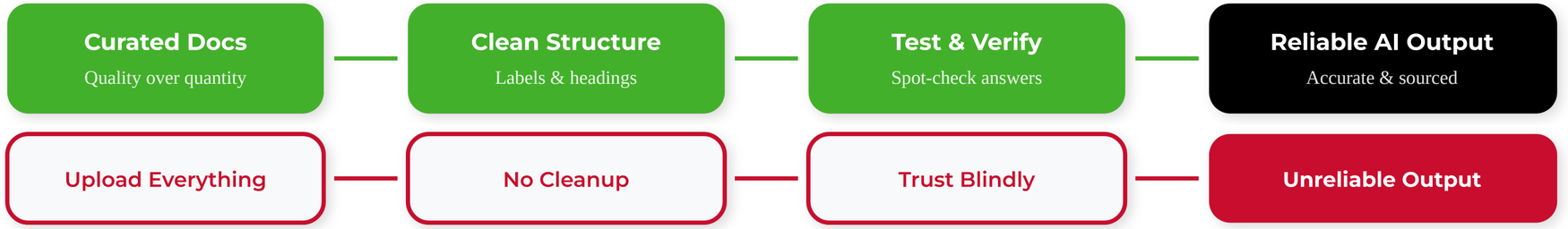
5. Verify Outputs

Always spot-check AI answers against source documents, especially for critical business decisions.

6. Keep Docs Updated

Old documents = old answers. Set a review schedule for your AI context library.

Best Practices & Common Mistakes



Best Practices

- Curate carefully: Quality over quantity
- Use descriptive filenames: "BeaconReturnPolicy2025.pdf" not "doc1.pdf"
- Organize by topic: Group related documents
- Provide metadata: "This policy applies to all stores"
- Version control: Date documents clearly

Common Mistakes

- Upload everything: Overwhelms the system
- No formatting cleanup: Messy docs = messy answers
- Conflicting information: Multiple versions of same policy
- Assuming accuracy: Always verify critical answers
- Ignoring privacy: Don't upload sensitive data to public AI tools

Security Warning

Be cautious about uploading confidential business information to public AI platforms. Many companies use private, enterprise versions with stronger data protections.

Wrap-Up

Bringing It All Together

Key Takeaways

1. Context engineering builds information environments for AI — it's the difference between a chatbot and a business tool.
2. Structure improves results. Markdown, JSON, and clear formatting help AI work faster and more consistently (even though it can read messy docs too).
3. RAG + Embeddings power semantic search — finding information by meaning, not just keywords. Vector databases store these embeddings at scale.
4. Gemini Gems give you no-code RAG — persistent Knowledge, custom instructions, and shareable via link. Free tier.
5. RAG beats fine-tuning for most business cases. Behavioral configuration (instructions + memory) is accessible to everyone; true fine-tuning rarely needed.

What's Next

Day 4 Preview: Multimodal AI

Beyond text: AI that understands images, audio, video, and how to combine them.

- Vision AI for image analysis
- Speech recognition and synthesis
- Video understanding
- Combining modalities for richer insights

Next Step

Head to the lab to create your own Gemini Gem and run a three-way context comparison!

Estimated Lab Time

90-120

minutes

Start Lab →

Discussion: How Could Context Engineering Help YOUR Future Employer?

Think about an internship or job you're interested in.

What company documents could you upload to AI to make your work more efficient?

Marketing

Brand guidelines, past campaigns, customer personas, messaging frameworks

Operations

Process manuals, SOPs, vendor contracts, supply chain documentation

Finance

Financial reports, budgets, policies, compliance documentation

Questions?

Before we move to the lab...