How I Built a Local RAG System on My Laptop Using Google's Gemma Models

5 min read · 11 hours ago

Civil Learning Following

I've been experimenting a lot with on-device AI — partly because I'm tired of API limits, and partly because I like the idea of my models functioning *offline*.

And one thing that has become clear: if you aim to make your AI both *intelligent* and *accurate*, you need **RAG** — Retrieval-Augmented Generation.

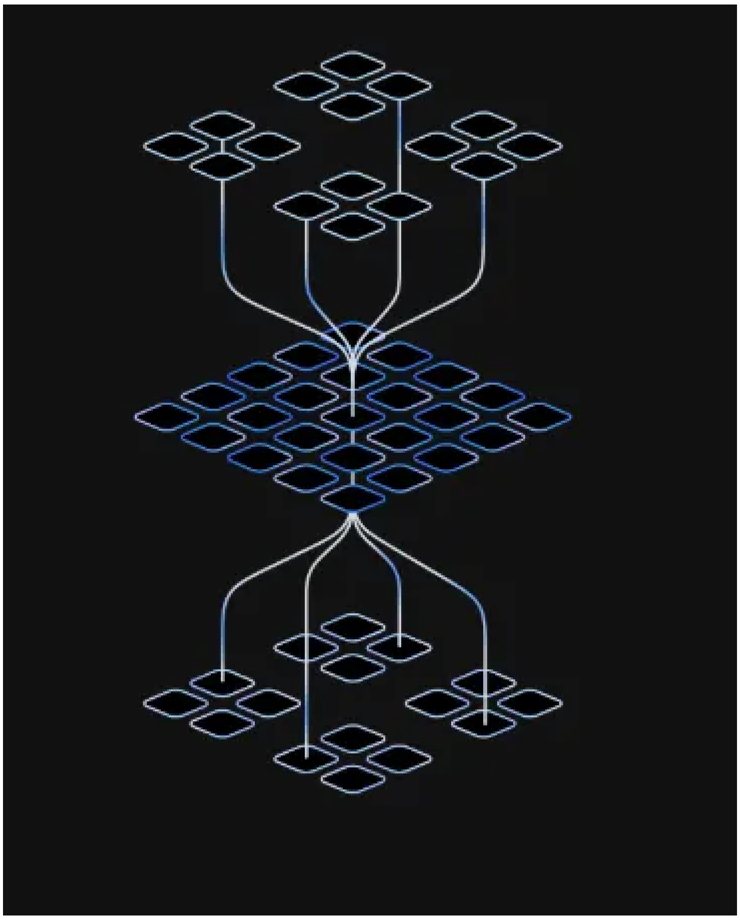
. . .

What's RAG again?

Think of it like this.
A typical LLM (such as a 7B or 70B model) is akin to a knowledgeable person who hasn't read the news in the past two years.

RAG gives that model a library card.

It allows the system *to retrieve* actual information (retrieval) and then *generate* responses based on it (generation). The result: your AI suddenly understands what's in your PDFs, documents, or web pages — without retraining.



. . .

Why I Picked Gemma

Google quietly removed the **Gemma** models some time ago, and I've honestly fallen in love with them for small, local workflows.

They're lightweight. They work on laptops and even Android phones, and they remain surprisingly effective at understanding and generating text.

Specifically, I used:

- **EmbeddingGemma 300M** → for turning text into embeddings
- **Gemma 3 1B** → for generating context-aware responses

It's essentially all you require for a private, end-to-end RAG setup.

. . .

Step 1 — Grab the Text from a PDF

I began with a PDF located in my app's assets folder. To extract the text, I used the reliable **iText Core** library.

Here's the short version:

```
context.assets.open(assetFileName).use { inputStream ->
    val pdfReader = PdfReader(inputStream)
    val pdfDocument = PdfDocument(pdfReader)

    val text = StringBuilder()
    val numberOfPages = pdfDocument.numberOfPages

    // Extract text from all pages (limit to first n pages to avoid overwhelmin
    val pagesToProcess = minOf(numberOfPages, 100)

    for (page in 1..pagesToProcess) {
        val pageText =
            PdfTextExtractor.getTextFromPage(pdfDocument.getPage(page))
               if (pageText.isNotBlank()) {
                   text.append("## Page $page\n")
                   text.append(pageText.trim())
                   text.append("\n\n")
               }
        }

    pdfDocument.close()

    val result = text.toString().trim()
        result.ifBlank {
          "[PDF Document: $assetFileName - No readable text content found]"
        }
    }
```

Basically: open — read — extract — done.
Please keep it to 100 pages or your phone will overheat.

. . .

Step 2 — Chunk the Text

LLMs don't handle long inputs well. You should split your document into smaller parts — ideally about 256 tokens each.

Here's how I split mine using **DJL's tokenizer:**

```
private fun loadTokenizer() {
    try {
        tokenizer =
            HuggingFaceTokenizer.newInstance(Paths.get("/data/local/tmp/tokeniz
        Log.d("GemmaTokenizer", "Tokenizer loaded successfully.")
      } catch (e: Exception) {
        Log.e("GemmaTokenizer", "Failed to load tokenizer", e)
    }
 }
.....
val chunker = ChunkerHelper.RecursiveTextChunker(
    tokenizer = tokenizerAdapter,
    maxChunkTokens = 256, // Your target chunk size in TOKENS
    overlapTokens = 40,  // Your target overlap in TOKENS
    separators = listOf("\n\n", "\n", ". ", " ", "")
)

// Create the chunks
val chunks = chunker.createChunks(fileTextContent ?: "")

.....

fun createChunks(text: String): List<String> {
    return chunkTextRecursively(text.trim(), separators)
}

/**
 * Recursively splits text into chunks of a desired size.
*/
private fun chunkTextRecursively(text: String, separators: List<String>): List<
    val finalChunks = mutableListOf<String>()
    // 1. Base Case: If the text is small enough, return it as a single chunk.
    if (tokenizer.countTokens(text) <= maxChunkTokens) {
        return listOf(text)
    }

// 2. Recursive Step: Try to split by the next available separator.
val currentSeparator = separators.firstOrNull()
    if (currentSeparator == null) {
        // If no more separators, do a hard split. This is the final fallback.
        val hardChunks = mutableListOf<String>()
        for (i in 0 until text.length step maxChunkTokens) {
            hardChunks.add(text.substring(i, minOf(i + maxChunkTokens, text.len
        }
```

```
        return hardChunks
    }
```

Overlap helps maintain a smooth flow of context between chunks — similar to puzzle pieces fitting together.

. . .

Step 3 — Generate and Save Embeddings

Once I had my chunks, I processed them using the **EmbeddingGemma 300M** model.

Each chunk becomes a vector — essentially a list of floating-point numbers that represent its meaning.

```
val embeddingsMap = HashMap<String, FloatArray>()

chunks.forEach { sentence ->
    val embedding = runEmbedding(sentence)
    if (embedding.isNotEmpty()) {
        embeddingsMap[sentence] = embedding
        Log.d(
            "EmbeddingLog",
            "Computed embedding for '$sentence': [${
                embedding.take(10).joinToString(", ")
            }...]"
        )
    }
}

.....
ObjectOutputStream(FileOutputStream(embeddingsFile)).use { stream ->
   stream.writeObject(embeddingsMap)
}
```

This part takes a while.
Perform it once, save the vectors, and reuse them later.

. . .

Step 4 — Turn the User Query into a Vector Too

When someone types a question, I apply the same embedding process to it.

```
private fun runEmbedding(query: String): FloatArray {
    if (tokenizer == null || interpreter == null) {
        Log.e("EmbeddingError", "Tokenizer or Interpreter not initialized.")
        return floatArrayOf()
    }
    val prompt = "task: search result | query: "
    val fullInput = prompt + query
    val encoding = tokenizer!!.encode(fullInput)
    val currentIds = encoding.ids
    val sequenceLength = 256
    val truncatedIds = if (currentIds.size > sequenceLength) {
        currentIds.take(sequenceLength)
    } else {
        currentIds.toList()
    }
    val paddedIds = IntArray(sequenceLength) { 0 }
    for (i in truncatedIds.indices) {
        paddedIds[i] = truncatedIds[i].toInt()
    }
    val inputArray = arrayOf(paddedIds)
    val outputBuffer = TensorBuffer.createFixedSize(
        intArrayOf(1, 768),
        DataType.FLOAT32
    )
    try {
        interpreter?.run(inputArray, outputBuffer.buffer)
        return outputBuffer.floatArray
    } catch (e: Exception) {
        Log.e("EmbeddingError", "Failed to run TFLite interpreter", e)
        return floatArrayOf()
    }
}
```

Now we can compare this query vector with all our chunk vectors.

. . .

Step 5 — Find the Closest Matches (Cosine Similarity)

If you remember high school math: cosine similarity essentially measures the *angle* between two vectors. Smaller angle → more similar.

```
fun cosineSimilarity(vectorA: FloatArray, vectorB: FloatArray): Float {
    if (vectorA.size != vectorB.size) {
        throw IllegalArgumentException("Vectors must be of the same size")
    }

    var dotProduct = 0.0
    var normA = 0.0
    var normB = 0.0
    for (i in vectorA.indices) {
        dotProduct += vectorA[i] * vectorB[i]
        normA += vectorA[i] * vectorA[i]
        normB += vectorB[i] * vectorB[i]
    }

    val magnitudeA = sqrt(normA)
    val magnitudeB = sqrt(normB)
    if (magnitudeA == 0.0 || magnitudeB == 0.0) {
        return 0.0f
    }

    return (dotProduct / (magnitudeA * magnitudeB)).toFloat()
}
```

Then grab the top few matches:

```
val topThree = allMatches.sortedByDescending { it.similarity }.take(3)
val bestMatches = topThree.joinToString("\n\n---\n\n") { it.text }
```

. . .

Step 6 — Ask Gemma for an Answer

Finally, the fun part.
We feed the retrieved chunks into **Gemma 3 1B**, along with the user query.

```
val inputPrompt =
  "You are a helpful assistant that answers the user query: $query, based ONLY

llmInference?.generateResponseAsync(inputPrompt.take(1280)) { partialResult, do
    stringBuilder += partialResult
    onResult(stringBuilder)
}
```

This makes the LLM stick strictly to the retrieved text.
No hallucinations. No wild guesses.

```
private fun loadLLM() {
  val taskOptions = LlmInferenceOptions.builder()
        .setModelPath("/data/local/tmp/Gemma3-1B-IT_seq128_q8_ekv1280.task")
        .setPreferredBackend(Backend.CPU)
        .setMaxTokens(MAX_TOKENS) // 1280
        .build()
    llmInference = LlmInference.createFromOptions(this, taskOptions)
}
.....
val inputPrompt =
    "You are a helpful assistant that responds to user query: ${query}, based O
Log.v("EmbeddingMatches", inputPrompt)
var stringBuilder = ""
    llmInference?.generateResponseAsync(inputPrompt.take(MAX_TOKENS)) { partial
        if (partialResult != ".\\" && partialResult != "\\" && partialResult !=
            stringBuilder += partialResult
        }
        Log.v("finished_res", stringBuilder)
        onResult(stringBuilder)
}
```

. . .

The End Result

It’s honestly pretty satisfying to see it work. You can ask questions like:

> “What’s mentioned in section 3 about model optimization?”

And receive an accurate, context-aware response — directly from your local PDF.

No internet. No API keys. No "GPT-4 is at capacity."

. . .

Takeaway

RAG is the secret ingredient that transforms a static model into a dynamic, context-aware assistant.
And Gemma models enable all of that *without relying on the cloud*.

It's not just about privacy — it's about ownership.
Your model, your data, your answers.

Google AI Ai Agent Ai Tools Llm

Follow

Published in Coding Nexus

8.1K followers · Last published 1 hour ago

Coding Nexus is a community of developers, tech enthusiasts, and aspiring coders. Whether you're exploring the depths of Python, diving into data science, mastering web development, or staying updated on the latest trends in AI, Coding Nexus has something for you.

Following

Written by Civil Learning

3.1K followers · 6 following

We share what you need to know. Shared only for information.

Responses (1)

Bgerby

What are your thoughts?

KT Lau
6 hours ago

are the codes above written in Kotlin or some other programming languages?

Reply

More from Civil Learning and Coding Nexus

In Coding Nexus by Civil Learning

The Guy Who Let ChatGPT Trade for Him—and Somehow It Worked

You know how everyone says, “Don’t let AI touch your Money”? Well, someone on Reddit decided to ignore that.

Oct 8 188 8

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You’ll Install This Year

I didn’t expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 382 15

In Coding Nexus by Algo Insights

How to Build Your Own AI Rig for Running Local LLMs (Gemma, Mistral, Qwen, GPT-OSS and Llama)

About three months ago, I realised I was utterly dependent on companies that didn't care about anything except power, Money, and control.

Oct 8 · 358 · 14

In Coding Nexus by Civil Learning

MarkItDown: Convert Anything into Markdown—the Smart Way to Feed LLMs

You know that feeling when you're trying to feed a PDF or a Word document into an LLM, and it just doesn't understand what's going on...

Oct 15 · 262 · 4

See all from Civil Learning

See all from Coding Nexus

Recommended from Medium

In AI Software Engineer by Joe Njenga

Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild

Cursor has released version 2.0 , bringing the most powerful agentic AI we have seen yet, more autonomous than ever before,here's what's...

3d ago 325 6

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 382 15

In Towards AI by Teja Kusireddy

We Spent $47,000 Running AI Agents in Production. Here's What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16 1.8K 49

In Generative AI by Gao Dalie (高達烈)

DeepSeek-OCR + LLama4 + RAG Just Revolutionized Agent OCR Forever

During the weekend, I scrolled through Twitter to see what was happening in the AI community. Once again, DeepSeek has drawn worldwide...

6d ago 193

In Artificial Intelligence in Plain English by Surendra Pandar

Every Paid AI—Now FREE & UNLIMITED (100% Legal)

The founder raised $5.7M to make this possible

Oct 21 75 4

Agen.cy

20+ Genius Ways Power Users Are Using Claude Code Right Now

Here are 10+ ways power users are using Claude Code🧵

Oct 23 33 1

See more recommendations