Title: Processing Document Collections with LLMs: A Practical Workflowpr

Abstract: Every organization has stacks of similar documents - customer complaints, resumes, error logs - that need the same questions answered about each one. This talk walks through a systematic workflow for processing these document collections with LLMs, covering the full pipeline from messy input to polished results. I'll share real examples and the tools I built to automate the repetitive parts across different projects, including wrangling LLM outputs and creating modular display components.

Title: Processing Document Collections with LLMs: A Practical Workflow

I had a deadline. I was building a tool to analyze public comments for a proposed regulation to see if they supported or opposed it, and I wanted my pipeline ready before the comment period closed so we could update the site and make it final as soon as the last comments were out.

I'd built document processing tools before. I had my usual blocks – PyPDF2 for PDFs, python-docx for Word docs. These blocks had worked great on previous projects.

But this time, some of the attachments were weird. Dark, scanned PDFs. Random image files. My usual blocks were failing.

So I added new blocks. Local OCR for images. But that was spitting out gibberish sometimes. So I built another block – a gibberish detector. Now I'm debugging my gibberish detector, watching the deadline approach, when I finally said fuck it.

Let me just try Gemini on everything that fails my initial pdf/word blocks. It’s a multimodal model, let’s see how it does.

Seven cents. That's what it cost to process the weird documents. It didn’t always work, but it worked enough of the time for this project. Seven cents and I could delete two annoying blocks.

When those 35,000 comments dropped, I was ready. Got them analyzed and on the website within days.

Here's what I’ve learned during the last year, as I’ve been working on document processing pipelines, or tools for taking your stack of messy documents and asking the same questions of each of them: Every text processing pipeline follows the same basic pattern – get text, process it, do something with results. But the specific blocks you need? Those depend entirely on your problem.

By the end of this, I want you to come away with:

1. Why you might want this workflow
2. Some blocks you're probably going to want, regardless of your project

3. The right questions to ask early – about cost, model restrictions, accuracy needs – so you build the right blocks for YOUR problem

Processing Document Collections with LLMs: A Practical Workflow

Slides

Slide 1: Title

Building Blocks for Text Processing Pipelines

[Your name]

Slide 2: The Seven-Cent Solution

[Image: Receipt showing $0.07]

Slide 3: The Deadline

Comment Period Closing Soon

[Calendar icon with countdown]

Slide 4: My Usual Blocks

[Visual: LEGO blocks labeled "PyPDF2" and "python-docx"]

Slide 5: The Weird Files

[Image: Dark scanned PDF] [Image: Meme attachment] [Image: Blurry photo]

Slide 6: Building More Blocks

[Visual: Stack of blocks labeled "OCR", "Gibberish Detector", "Error Handler"]

Slide 7: Debugging the Gibberish

Þ¿½¾ñ╠╩◄♦

Is this... text?

Slide 8: Seven Cents

[Image: Gemini logo with $0.07]

Delete 2 blocks

Slide 9: Success!

35,000 comments processed

[Screenshot: Schedule F website]

Slide 10: Flashback

Every project was the same pattern...

[Visual: Multiple project screenshots fading together]

Slide 11: The Pattern

[Visual: Three LEGO blocks in a row]

GET → PROCESS → DO

Slide 12: Where This Pattern Lives

- 📋 Medical Records → Fraud?
- 📄 Resumes → Interview?
- 💬 Customer Complaints → Issues?
- 🚨 Error Logs → Root Cause?
- 💼 Job Postings → Skills?

Slide 13: My First Pipeline

[Visual: Simple 3-block LEGO structure]

CSV → GPT-4 → GitHub Pages

Slide 14: It Worked!

[Screenshot: Simple job posting analysis site]

"I've solved document processing!"

Slide 15: Then Came Schedule F

Same pattern?

- ~~Hundreds~~ → 35,000 documents
- ~~CSV files~~ → CSV + PDFs + Images + ???
- ~~One-time~~ → Daily updates
- ~~Just me~~ → Frontend developer needs schemas

Slide 16: Block Explosion

[Visual: Original 3 blocks with dozens of new blocks stacking chaotically]

- CAPTCHA Handler
- Resume Logic
- Deduplicator
- Schema Validator
- Attachment Processor
- State Manager
- Retry Logic

Slide 17: The Abstraction Trap

[Visual: Tangled web of blocks labeled "LangChain", "LiteLLM", "Controlflow"]

Built for flexibility I never needed

Slide 18: The Right Questions

🔐 **Can I use cloud APIs?**

📊 **Good enough or perfect?**

🔄 **One-time or ongoing?**

📈 **Real scale?**

🎯 **Do I need flexibility?**

Slide 19: Your Document Stack

[Visual: Stack of documents with question marks]

What questions do YOU ask repeatedly?

Slide 20: Start Simple

[Visual: 3 basic LEGO blocks]

Get → Process → Do

What's your seven-cent solution?

[GitHub link]

These slides:

- Heavy on visuals (LEGO blocks reinforce your theme)
- Minimal text
- Tell your story chronologically after the hook
- Build complexity visually (3 blocks → explosion of blocks)
- End where you started but with new knowledge

The LEGO metaphor works throughout - simple structures becoming complex, removing unnecessary blocks, building the right thing.

CAPTION: get, process, do block

CAPTION: seven cents

Caption: gemini block


Gemini

Caption:gibberish detector block

CAPTION: resume, validate, manage, retry, deduplicate blocks


RESUME
VALIDATE SCHEMA
MANAGE STATE
RETRY
DEDUPLICATE