

# Processing Document Collections with LLMs

A Practical Workflow

Abigail Haddad

**⚠ THIS IS NOT ABOUT RAG ⚠**

# Every Organization Has Documents

(and the same questions about all of them)



Medical Records



Resumes



Customer  
Complaints



Error Logs



Contracts



Reports

# Agenda

- 📌 The Project
- 📌 The Larger Problem
- 📌 Applying This Pattern
- 📌 What I Learned
- 📌 Evaluation
- 📌 Building A Couple More Blocks

# Part 1: The Project

# Regulations.gov Comments

The screenshot shows a web page from [Regulations.gov](#). At the top, there's a blue header bar with the site logo and the text "Your Voice in Federal Decision Making". On the right side of the header is a "SUPPORT" button. Below the header, there's a navigation link "[View Docket](#)". In the center, there's a section for a "PROPOSED RULE" with a blue icon and the text "PROPOSED RULE". To the right of this section is a "Share" button with a dropdown arrow. The main content area features a bold title: "**Improving Performance, Accountability and Responsiveness in the Civil Service**". Below the title, it says "Posted by the **Office of Personnel Management** on Apr 23, 2025". On the left, there's a dark button with white text that says "[Closed for Comments](#)". On the right, there's a box containing a message: "Comment Period Ended: Jun 7, 2025 at 11:59 PM EDT".

# What Did the Comments Say?

**AGREE**



**DISAGREE**



# My Deadline



# My Usual Blocks



# The Weird Stuff

📄 Dark scanned PDFs

🎭 Random meme attachments

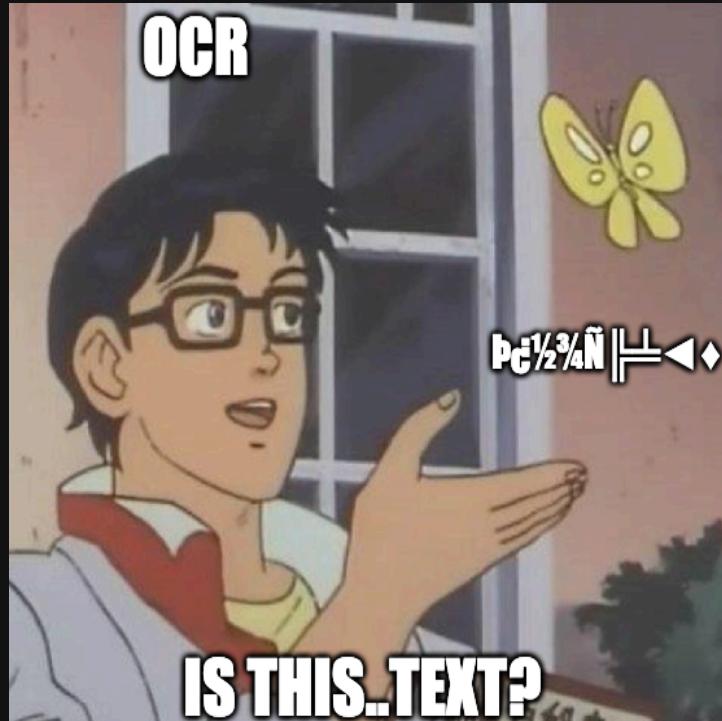
📸 Blurry photos of handwritten notes



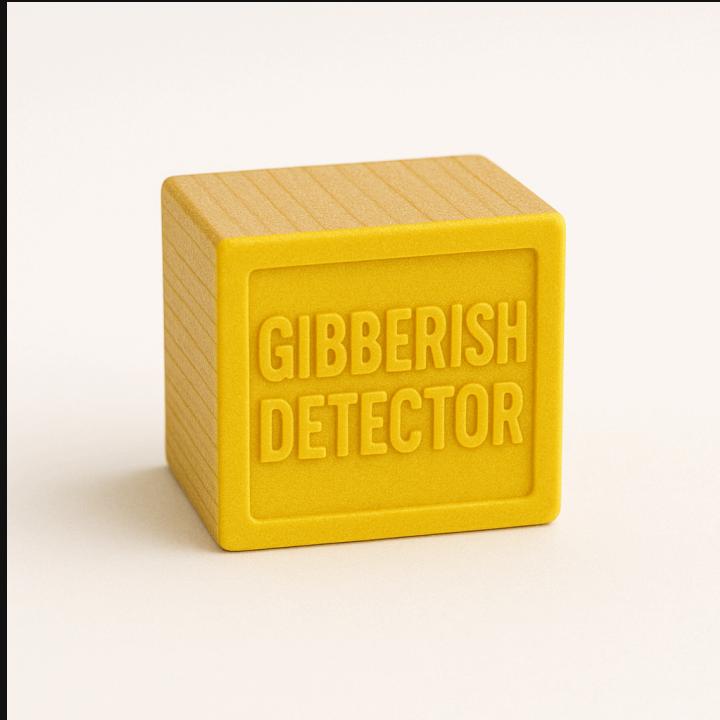
# I Added a New Block



# I Occasionally Got Gibberish



# So I Built a Gibberish Detector



# Now I'm Debugging My Gibberish Detector



# Finally, I Try Gemini To Extract Text...

```
if extraction_fails(doc):
    result = gemini.process(doc)
```

It Works!!



# Gemini Initially Cost Me Seven Cents



# The Results

**33,647**

Against  
(94%)

**1,596**

Support  
(4.5%)

**306**

Neutral  
(0.9%)

Many "support" comments were duplicates/campaigns

# Here It Is

**Public Comments on "Schedule F" Regulation**  
Comments about the proposed Schedule Policy/Career Regulation - Improving Performance, Accountability and Responsiveness in the Civil Service ↗

**Statistics Overview**

TOTAL COMMENTS <b>35,550</b>	FOR <b>1,596</b> For	AGAINST <b>33,647</b> Against	NEUTRAL/UNCLEAR <b>306</b> Neutral/ Unclear
---------------------------------	----------------------------	-------------------------------------	--

**Unique Statistics (Duplicates Removed)**

UNIQUE COMMENTS <b>28,096</b>	FOR <b>251</b> For	AGAINST <b>27,546</b> Against	NEUTRAL/UNCLEAR <b>298</b> Neutral/ Unclear
----------------------------------	--------------------------	-------------------------------------	--

**Filters**

No filters applied  
Use the filter buttons below to refine your data view

Filter Options

Title, Stance, Key Quote, Themes, Rationale, Comment, Category, Duplicates, Source Link, Comment ID, Has Attachments, Posted Date, Received Date, Organization, Attachment Titles

Search

Search comments...

# Part 2: The Larger Problem

# Every Organization Has Documents



Medical Records



Resumes



Customer Complaints



Error Logs



Contracts

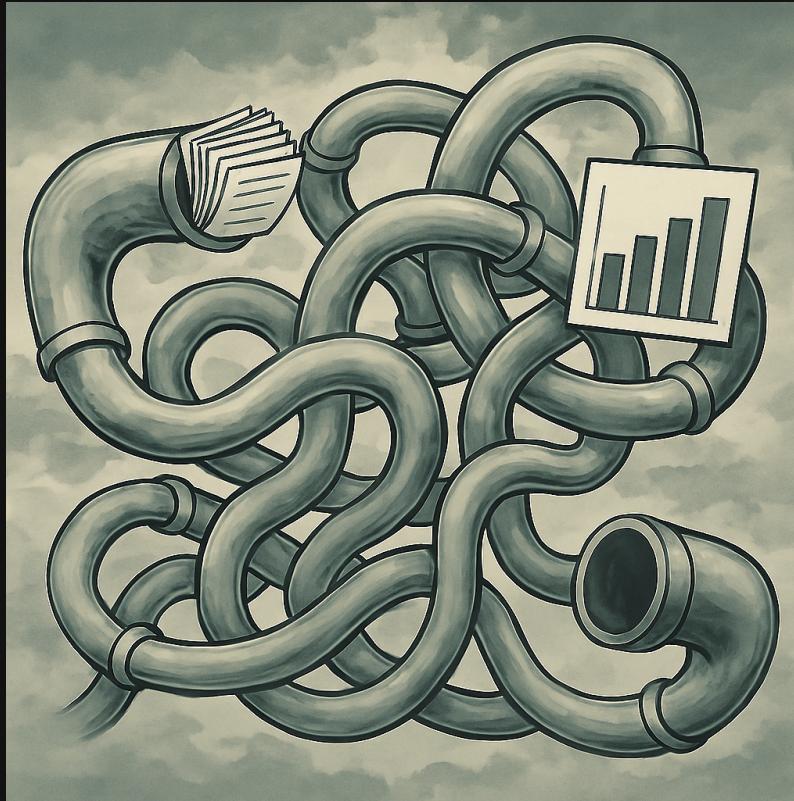


Reports

# Same Question Every Time

- Is this fraudulent? → **yes/no**
- What's the complaint about? → **category**
- Does this candidate qualify? → **yes/no + reason**
- What's the root cause? → **specific issue**

# The Answer? Document Processing Pipeline



# The Three Steps

## GET

Pull text from PDFs, Word docs, APIs, or user input

## PROCESS

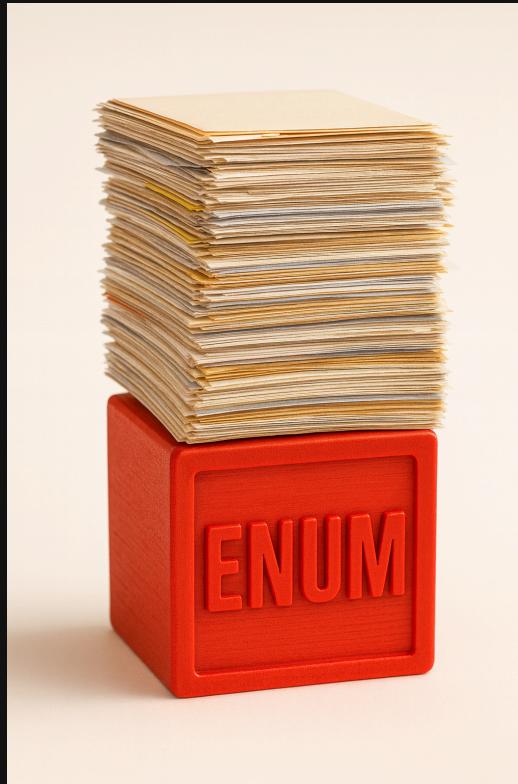
Use LLMs, regex, BERT, or other tools to extract meaning

## DO

Create visualizations, populate templates, or trigger actions



# Getting Back ONLY "Agree" or "Disagree"



Enums → Structured output → Wrangleable results

# LiteLLM Structured Output + Pydantic

```
import os
from litellm import completion
from pydantic import BaseModel

# add to env var
os.environ["OPENAI_API_KEY"] = ""

messages = [{"role": "user", "content": "List 5 important events in the XIX century"}]

class CalendarEvent(BaseModel):
    name: str
    date: str
    participants: list[str]

class EventsList(BaseModel):
    events: list[CalendarEvent]

resp = completion(
    model="gpt-4o-2024-08-06",
    messages=messages,
    response_format=EventsList
)

print("Received={}".format(resp))
```

# Put Anything You Want in The Prompt

```
class CommentAnalyzer:  
    """LiteLLM-based analyzer for public comments using Pydantic models for structured output."""  
  
    def __init__(self, model=None, timeout_seconds=None):  
        self.model = model or config.llm.model  
        self.timeout_seconds = timeout_seconds or config.llm.timeout  
  
        # Ensure API key is available  
        if "OPENAI_API_KEY" not in os.environ:  
            raise ValueError("OPENAI_API_KEY not found in environment variables or .env file")  
  
    def get_system_prompt(self):  
        return """You are analyzing public comments submitted regarding a proposed rule to implement "Schedule F" (or "Schedule Policy/Career Development"). This proposed rule would allow federal agencies to reclassify career civil servants in policy-influencing positions into a new employment category. Your task is to determine the stance of each comment relative to this rule.  
  
1. Stance: Determine if the comment is "For" (supporting the rule), "Against" (opposing the rule), or "Neutral/Unclear" by examining both explicit and implicit language.  
  
Classification guidelines with special attention to boundary cases:  
  
- "For": Comment supports the rule, defends its merits, or argues for implementation. Look for: praise of accountability, presidential authority, or specific policy details.  
- "Against": Comment opposes the rule, including indirect opposition through thematic alignment. Critical indicators include:  
  * Questions about constitutionality or legal concerns, even without explicit opposition  
  * Opposition to the rule's core principles or goals  
  * Requests for changes to the rule's implementation or scope  
  * Calls for further study or review before a final decision  
  
2. Content: Analyze the comment for specific content related to the rule, such as:  
  * Specific provisions or clauses mentioned  
  * References to other laws or regulations  
  * Examples of how the rule would affect different groups of employees  
  * Requests for more information or context  
  
3. Tone: Assess the overall tone of the comment, including:  
  * Whether it is formal or informal  
  * Whether it uses technical language or everyday language  
  * Whether it expresses enthusiasm or concern  
  
4. Context: Consider the context in which the comment was made, such as:  
  * The source of the comment (e.g., social media, email, news article)  
  * The date the comment was posted  
  * Any relevant political or historical events occurring at the time  
  
5. Conclusion: Summarize the main points of the comment and provide a classification based on the analysis above.  
"""
```

# What I Actually Extracted

- **Stance:** agree / disagree / neutral-other
- **Themes:** multi-class classification
- **Summary:** unstructured text
- **Representative quote:** actual text
- **Reasoning:** why this classification

# Part 3: Applying This Pattern



# Your Documents

- ✓ Clean PDFs?
- ✓ Scanned images?
- ✓ Mixed formats?
- ✓ Handwritten?

 Your Goals

- Extract fields?
- Classify docs?
- Summarize?
- Find patterns?

# Your Constraints



## Privacy

Can data leave?



## Budget

API costs OK?



## Time

One-off or ongoing?



## Explain

Show your work?

Should I Throw It Into an LLM?



MAYBE!!!



# Remember My Seven Cents?

Gemini only worked because:

- ✓ Data could leave my environment
- ✓ 35,000 documents, most with no attachments = small enough for the budget

**Different constraints = Different solution**

# Your Toolkit

## GET

- PyPDF2, pdfplumber
- Tesseract OCR
- BeautifulSoup
- APIs

## PROCESS

- OpenAI, Claude, Gemini
- BERT
- Regex

## DO

- Quarto
- Email/Slack alerts
- API endpoints

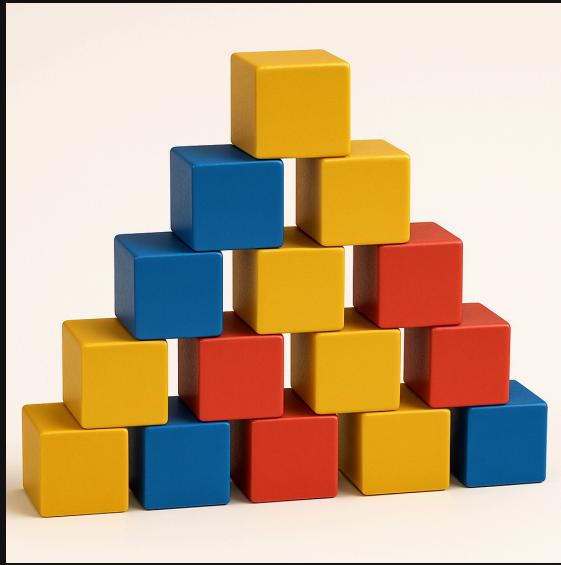
# Part 4: What I Learned

# Start Simple



Build only the blocks you need

# You Can't Build All The Blocks



# Scaling Up



**Handle Failures**



**Parallelize**



**Track State**



**Real Databases**

# Part 5: Evaluation

# Systematic Evaluation



## Build test sets:

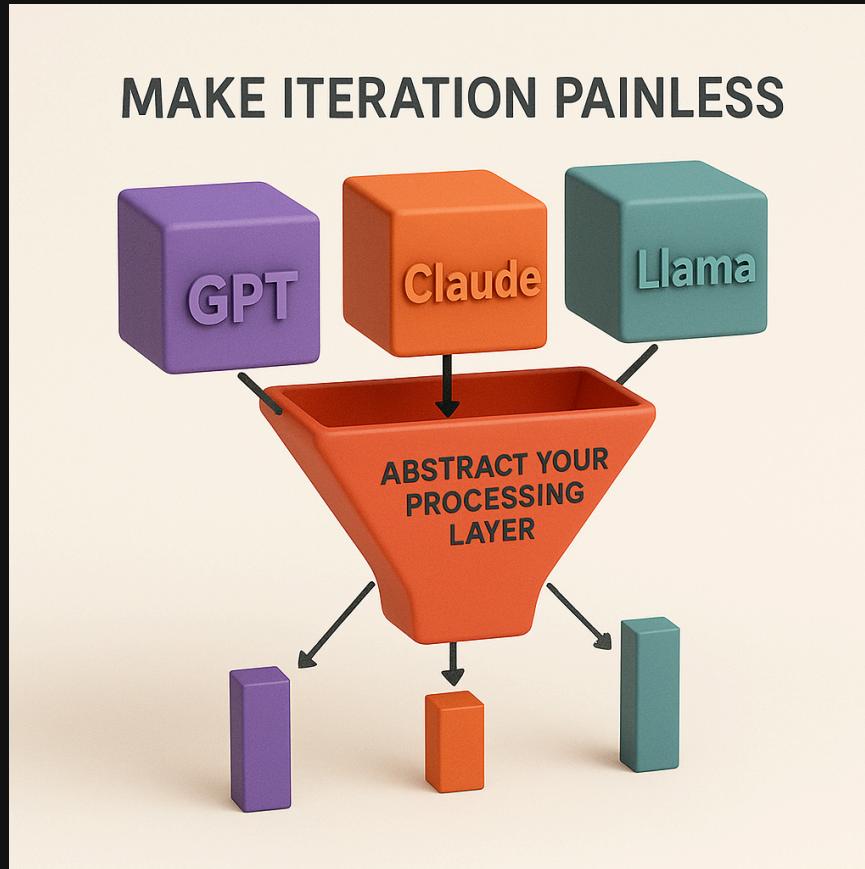
- Real examples from your data
- The weird edge cases
- What failed before



## Test what matters:

- Accuracy on YOUR data
- How it fails (not just how often)
- **What keeps me up at night?**

# Make Iteration Painless



# Part 6: Building A Couple More Blocks

# From One-Off Script to Product



## Built an automated tool that:

- ✓ Takes any public comment dataset
- ✓ Discovers the arguments automatically
- ✓ Writes its own classification prompts
  - ✓ Runs fully autonomous analysis

*Structured output/Enum is still the secret sauce*

# I Built A Couple More Blocks



# New Analysis!

## Request for Information on Health Technology Ecosystem

This regulation seeks public input on the development and integration of health technology ecosystems, focusing on digital health products and their impact on healthcare delivery and patient outcomes.

CMS Docket: CMS-2025-0050

50

Total Comments

35

With Attachments

May 28, 2025 to June 25,  
2025

Date Range

### Themes Distribution

Click on any theme to see positions, then click positions to see relationships

Health Data Interoperability

(2 positions) ▾

# Find Me on the Internet!

🌐 [abigailhaddad.netlify.app](https://abigailhaddad.netlify.app)

**Want to hire me to consult?**

Fill out the form on my website!

Also find me on:

 [github.com/abigailhaddad](https://github.com/abigailhaddad)

 [presentofcoding.substack.com](https://presentofcoding.substack.com)

 [github.com/abigailhaddad/rny\\_2025](https://github.com/abigailhaddad/rny_2025) (this presentation)

 [github.com/abigailhaddad/generic-comment-analyzer](https://github.com/abigailhaddad/generic-comment-analyzer) (code to get started)