POA Document Intelligence & RAG Pipeline — Project Summary

What I Built

I built a working proof-of-concept pipeline that takes Power of Attorney documents from Pennsylvania and Illinois and lets you ask plain-English questions about them. It pulls answers directly from the documents, cites its sources, and tells you when it doesn't have enough information rather than guessing.

The test set includes clean PDFs, digitally filled forms with checkboxes and simulated signatures, and a document with crossed-out sections and margin annotations. These aren't real scanned paper forms — they're generated PDFs designed to approximate real-world conditions. The one actual image-based test (a degraded JPEG) failed due to low resolution, which is a useful data point for planning how we'd handle real scanned documents in production.

The whole thing runs on Azure AI Foundry with Python and covers the full workflow: extracting text and structure from documents, indexing them for hybrid search, and generating grounded answers with GPT-4o.

Why It Matters

Legal documents like Powers of Attorney are dense, state-specific, and time-consuming to compare manually. In practice, they show up in all kinds of conditions — some are clean digital files, others are filled-in forms, and plenty arrive as rough scans. A system that can read these documents and answer specific questions about them saves real time, especially when you're comparing provisions across states or trying to find a specific detail buried in a multi-page form.

How It Works

The pipeline has four stages:

1. Document Preparation I generated seven test POA documents that cover a range of formats and complexity:

Document	State	Type	What It Tests
PA Durable Power of Attorney	PA	Financial	Clean PDF, multi-page with statutory notices
PA Healthcare Power of Attorney	PA	Healthcare	Clean PDF, end-of-life provisions
IL Statutory Short Form POA (Property)	IL	Financial	Clean PDF, 14 power categories
IL Statutory Short Form POA (Healthcare)	IL	Healthcare	Clean PDF, agent provisions
PA SERS Power of Attorney (Filled)	PA	Retirement	Digitally filled checkboxes, dual agents, simulated signatures and notary block
IL Healthcare POA (Filled Form)	IL	Healthcare	Digitally filled checkboxes for end-of-life preferences, organ donation, treatment options
PA Advance Directive (Messy)	PA	Healthcare	Crossed-out sections, simulated handwritten initials, margin notes, rubber stamp overlay

All documents are generated (not real filings) and use fictional principals: Margaret Whitfield (PA), Helen Kowalski (PA), Robert Nowak (IL), and Dorothy Fitzgerald (PA). The forms are modeled after real statutory templates from each state.

2. Text Extraction (Azure AI Document Intelligence) Each document gets sent to Azure's prebuilt Layout model, which extracts text, tables, and structure as Markdown. For the filled forms, it picked up checkbox states (checked vs unchecked) and preserved form field relationships. For the messy document, it captured the text despite the visual clutter.

Worth noting: since these are all natively digital PDFs (not scanned paper), the extraction had an easier job than it would with real-world scans. The Layout model is doing structured text extraction here, not heavy-duty OCR on degraded images.

3. Indexing (Azure AI Search + Azure OpenAI Embeddings) The extracted text gets chunked into overlapping segments (1,000 characters, 200-character overlap), then each chunk gets a vector embedding from text-embedding-ada-002. Everything gets uploaded to an Azure AI Search index that supports three search modes simultaneously:

- **Keyword search** — traditional text matching
- **Vector search** — semantic similarity via embeddings (HNSW algorithm)
- **Semantic ranking** — a deep neural re-ranker that scores results by meaning

Using all three together (hybrid search) gives noticeably better retrieval than any one method alone.

4. RAG Queries (GPT-4o) When you ask a question, the system embeds your question, runs a hybrid search to find the most relevant chunks, then sends those chunks to GPT-4o as context. The system prompt tells GPT-4o to only answer from the provided context, cite sources, and flag when it doesn't have enough information.

What I Tested and What Happened

Questions That Worked Well

Basic retrieval: "Who is the principal in the Pennsylvania POA?" → Correctly identified Margaret A. Whitfield, cited both PA source documents.

Cross-state comparison: "What are the end-of-life care provisions in each state's healthcare POA?" → Pulled from both IL and PA healthcare documents, compared provisions side-by-side, noted that IL explicitly mentions dialysis while PA includes a pain medication preference clause.

Checkbox form data: "Can Helen Kowalski's agents act independently or must they act together?" → Correctly identified from the PA SERS form that the "act individually" checkbox was selected in Section B.

"What healthcare preferences did Robert Nowak specify in his Illinois POA?" → Extracted the checked powers from the IL healthcare form, listed end-of-life preferences (no life-sustaining treatment, wants pain medication even if it hastens death), and cited the source document.

Where It Was Honest About Limitations

"How does Illinois handle successor agents differently from Pennsylvania?" → The search only retrieved Illinois chunks for this query, so GPT-4o correctly said it couldn't make a comparison without Pennsylvania context. It didn't make anything up.

What Didn't Work

Low-quality image scan: I generated a degraded JPEG (~67 DPI effective resolution) to simulate a poor-quality scan. Document Intelligence returned zero extracted characters — the resolution was too low for OCR. This tells us that any production system would need a minimum DPI check (probably 150+) before sending images to extraction.

Honest Assessment of the Test Set

The test documents are *approximations* of real-world conditions, not the real thing:

- The **checkboxes** are digitally generated, not hand-marked on paper. Real filled-in forms would have pen marks, partial checks, or stray marks that are harder to interpret.
- The **signatures** are programmatically drawn curves, not actual handwriting. Document Intelligence extracted the surrounding text but wasn't truly tested on handwriting recognition.
- The **messy document** has simulated annotations (crossed-out text, margin notes, stamps), but it's still a natively digital PDF. A genuinely messy document would be a scan of a physical paper with folds, uneven lighting, and scanner artifacts.
- The **failed image test** is actually the most informative result — it shows where the pipeline breaks and what would need to be addressed for production use.

To fully validate this approach, the next step would be to test with actual scanned POA documents from a real workflow.

Azure Services Used

Service	Role	Cost
Microsoft Foundry (Azure AI Foundry)	Central AI project hub	Included
Azure OpenAI — GPT-4o	Answer generation	Pay per token
Azure OpenAI — text-embedding-ada-002	Vector embeddings	Pay per token
Azure AI Search (Basic tier)	Hybrid search index with semantic ranking	~~$75/month~~ ($2.50/day)
Azure AI Document Intelligence (F0)	Document text extraction	Free tier

The Search service is the main ongoing cost. Everything else is pay-per-use and minimal for a test workload.

Tech Stack

- **Language:** Python 3.10+
- **Key Libraries:** azure-ai-documentintelligence, azure-search-documents, openai, python-dotenv, reportlab, Pillow
- **Infrastructure:** All Azure resources in East US 2, single resource group (`poa-rag-project`)

Scripts

Script	What It Does
`generate_poa_documents.py`	Creates 4 clean POA PDFs
`generate_realistic_poa_documents.py`	Creates 3 filled forms with checkboxes, signatures, and annotations
`step1_extract.py`	Sends documents to Document Intelligence, saves extracted Markdown
`step2_index.py`	Chunks text, generates embeddings, creates and populates search index
`step3_query.py`	Interactive question-answering interface (the RAG part)
`step4_extract_fields.py`	Structured field extraction to JSON/CSV for database use

Where This Could Go Next

1. **Test with real documents** — Run actual scanned POA forms through the pipeline to see how it handles genuine paper-quality issues
2. **Image pre-processing** — Add a DPI check and auto-upscaling step for low-quality scans before sending to Document Intelligence
3. **Custom extraction model** — For high-volume standardized forms, train a custom Document Intelligence model to extract fields directly (faster and cheaper than using GPT-4o for extraction)
4. **Database integration** — Use the structured field extraction (step 4) to populate a database, enabling both SQL queries and natural language queries
5. **Document classification** — Add an automatic classification step to identify document type before extraction (financial POA vs healthcare POA vs advance directive)
6. **Error handling and logging** — The current scripts are proof-of-concept quality. Production use would need retry logic, error handling, audit logging, and authentication via managed identity instead of API keys