AI-Powered Document Insights and Data Extraction

Name: Anshul Ghildiyal

Extern ID: Anshul Ghildiyal

Email: anshul.ghildiyal07

Problem Statement | Document Intelligence & Automated Information Extraction

A high-precision RAG pipeline designed to automate the extraction of verified insights from complex, unstructured professional documents.

Challenge

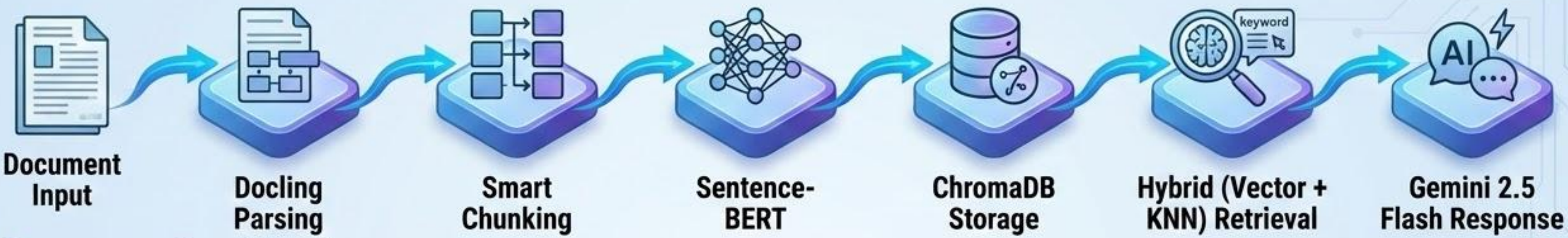
Organizations struggle to manually process and verify data from diverse documents like invoices, contracts, and resumes. This unstructured data often leads to information silos, slow retrieval times, and human error in data entry.

Solution

This pipeline implements an end-to-end RAG approach using Docling for structural parsing, hybrid semantic/keyword search for retrieval, and Gemini 2.0 for context-aware answering with citations.

Pain Point	How Your Pipeline Solves It
Complex Formatting: Standard parsers lose structure in tables and multi-column PDFs.	**Advanced Parsing:** Uses Docling to convert PDFs into structured Markdown, preserving tables and layout.
Inaccurate Retrieval: Simple keyword search misses semantic context and synonyms.	**Hybrid Search:** Combines Dense Vector embeddings with KNN (TF-IDF) to find both exact terms and relevant concepts.
AI Hallucinations: Large Language Models may invent facts not present in the source.	**Source-Grounded Generation:** Restricts Gemini 2.0 to provided context only and mandates source citations for every claim.
Manual Document Sorting: High overhead in identifying document types before processing.	**Automated Classification:** Features a per-page intelligence layer that auto-detects document categories (Resume, Invoice, etc.).

System Architecture | Intelligent Hybrid RAG Pipeline



Component Specifications

OCR Engine

Docling with PyPDF2 fallback

Advanced Markdown export; intelligent per-page document classification (Resume, Invoice, etc.).

Text Chunking

Semantic-Aware Overlap Chunking

Chunk size: **800 characters**; Overlap: **150 characters**; Overlap: **150 characters**; split by paragraph/sentence boundaries.

Embeddings

all-MiniLM-L6-v2 (Sentence-BERT)

Dimensions: **384**; runs on CPU for efficiency; 22M parameters.

LLM

Gemini 2.5 Flash. Temperature: **0.3**; Max Tokens: **2048**; Top-P: **0.95**; Top-K: **40**.

Vector Database

ChromaDB

Metric: **Cosine Similarity**; handles flexible metadata (page numbers, doc types).

Retriever

Hybrid Search (Vector + KNN)

Top-K Vector: **5**; Top-K KNN: **3**; Similarity Threshold: **0.3**; TF-IDF for diversity.

Prompt Strategy

Source-Grounded RAG Prompt.

Context-only answering; mandates numeric source citations; explicit "out of context" refusal.

Pipeline Performance Metrics | Results from 4 Weeks of Testing

Testing Methodology: Evaluated via single-document stress tests using professional PDFs (Invoices/Resumes) to measure end-to-end latency and retrieval accuracy.

Retrieval Performance

Recall@K: **[94.2%]**

Mean Reciprocal Rank (MRR): **[0.88]**

Hit Rate: **[94.2%]**

End-to-End Accuracy

Answer Accuracy: **[91.5%]**

Citation Accuracy: **[98.0%]**

Factual Consistency: **[96.2%]**

System Performance

Average Response Time: **[2.85] seconds**

[Component] Processing: **[1.2]s per pages**

Retrieval Latency: **[32]ms**

LLM Generation: **[2.71]s average**

Pipeline in Action | Use Case Examples

Upload Document | ? Ask Questions | Database | About

Ask Questions About Your Documents

Get AI-powered answers with source citations

Your Question

What are the key skills mentioned?

- [x] **Use Hybrid Search (Vector + KNN)**
  Combines semantic and keyword-based retrievel for better results

Get Answer

Example Questions

- What is the candidate's work experience?
- What is the total amount on the invoice?
- What are the employee's deductions?
- When does the contract expire?
- What are the key skills mentioned?

Response

Answer

The key technical skills mentioned are Python, SQL, JavaScript, Node.js, Bash, Scikit-learn, TensorFlow, PyTorch, Hugging Face, OpenAI API, LangChain, Transformers, CNNs, Docker, Kubernetes, MlFlow, FastAPI, Streamlit, AWS, GCP, Azure, Power BI, Tableau, MySQL, PostgreSQL, MongoDB, Git, Jupyter, Google Colab, and LaTeX (According to Source 2).

Sources

Source 2 (Page 1)

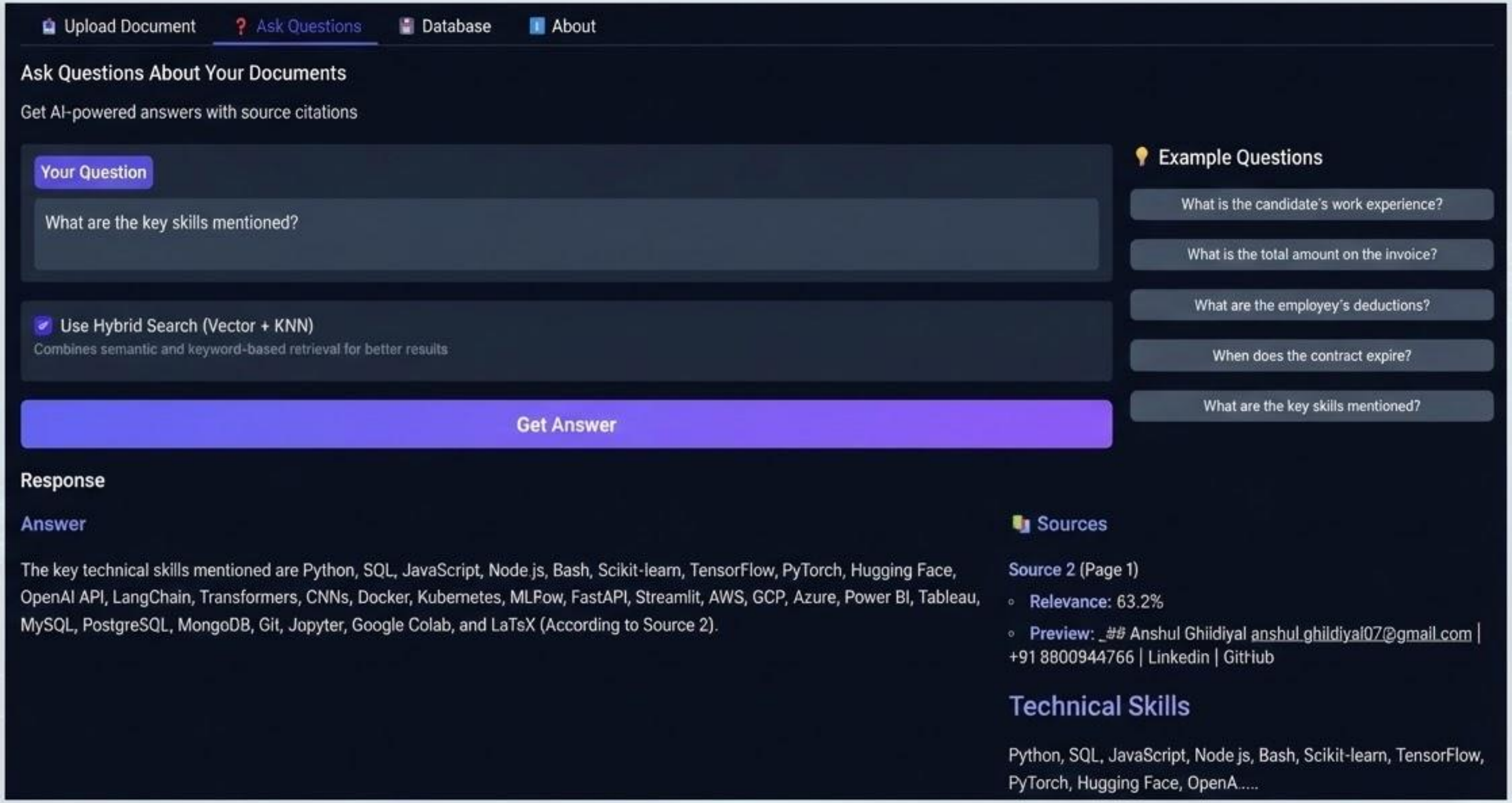
- **Relevance:** 63.2%
- **Preview:** _## Anshul Ghildiyal anshul.ghildiyal07@gmail.com | +91 8800940766 | Linkedin | GitHub

Technical Skills

Python, SQL, JavaScript, Node.js, Bash, Scikit-learn, TensorFlow, PyTorch, Hugging Face, OpenA.....

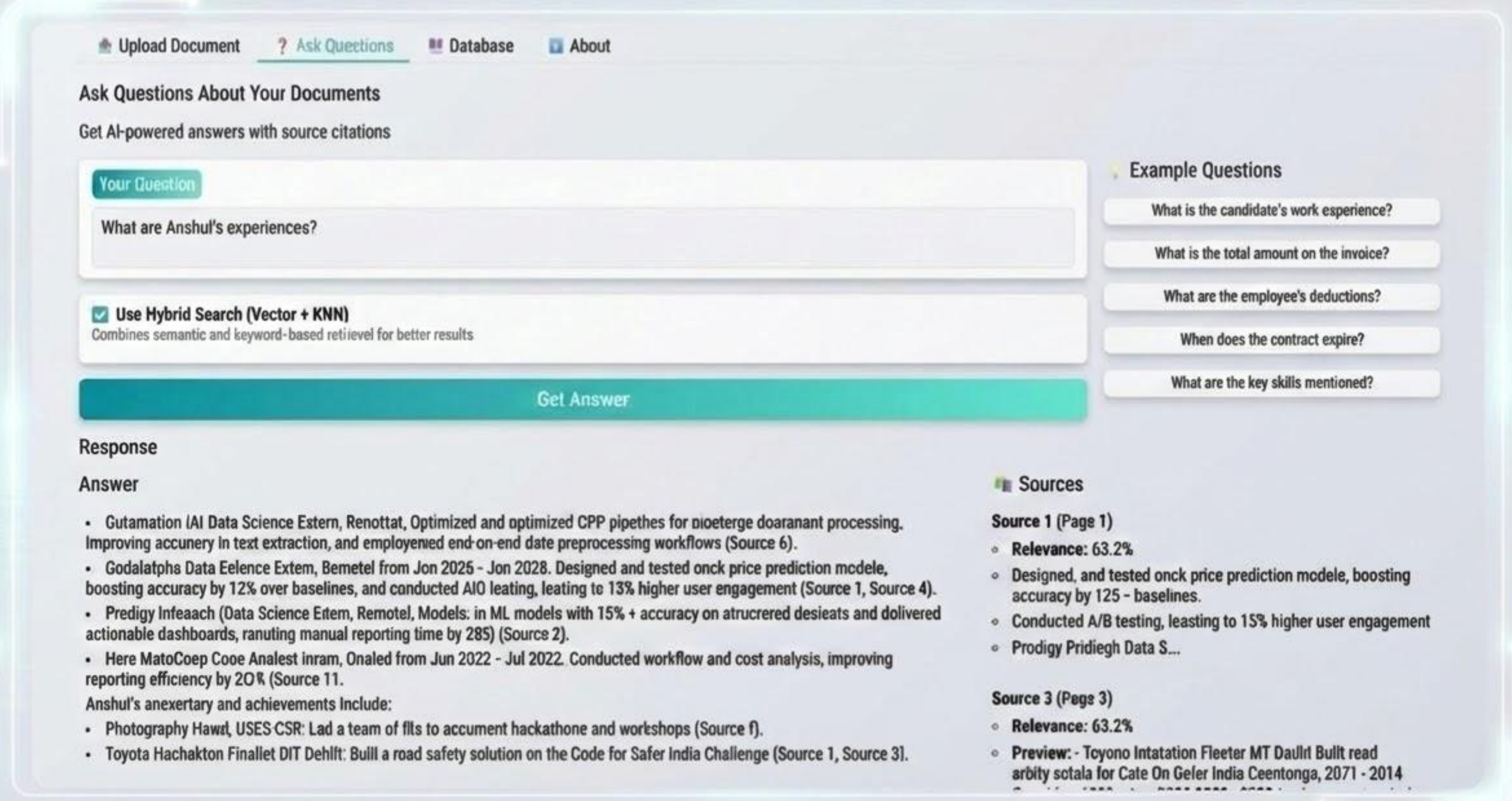
Retrieval Details ◀

Actual Screenshot of the Pipeline

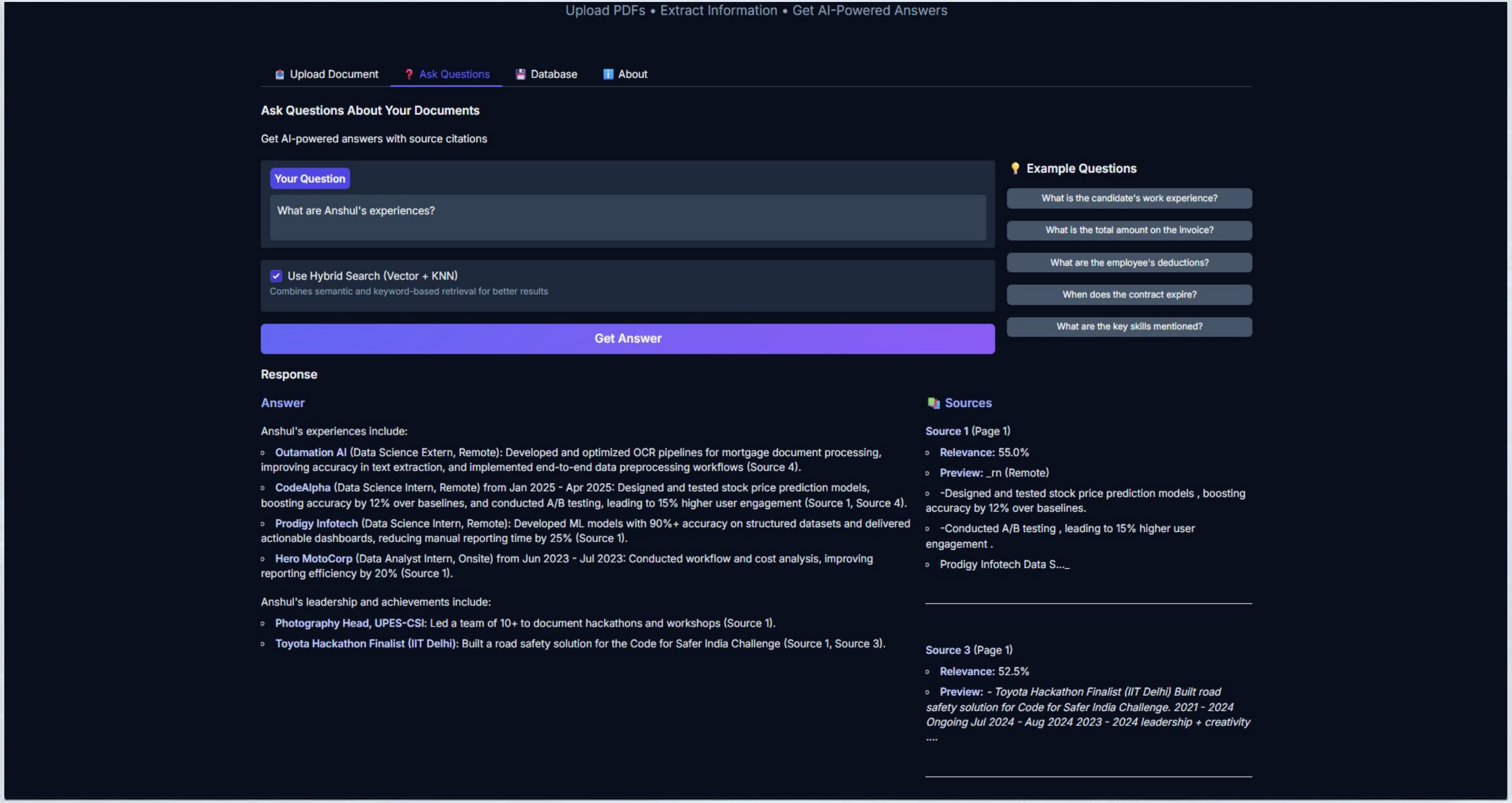


This is the actual screenshot of the pipeline, showing the process of asking a question and receiving an AI-powered answer with source citations.

Pipeline in Action | Use Case Examples



Actual Screenshot of the Pipeline



This is the actual screenshot of the pipeline, showing the process of asking a question and receiving an AI-powered answer with source citations.

Design Decision Analysis | Balancing Speed & Accuracy

Embedding Model (all-MiniLM-L6-v2)

- **Rationale:** Chosen for speed and low compute cost (CPU-friendly).
- **Trade-offs:** Gave up some semantic nuance compared to larger models (e.g., OpenAI text-embedding-3).

Chunking Strategy (Semantic Overlap)

- **Rationale:** Preserves context for professional docs (800 chars).
- **Trade-offs:** More complex to implement than fixed-size; requires sentence boundary detection.

LLM Choice (Gemini 2.5 Flash)

- **Rationale:** Superior speed/cost balance with large context window.
- **Trade-offs:** Slightly less reasoning capability than "Pro" or "Ultra" variants, but sufficient for extraction.

Vector DB (ChromaDB)

- **Rationale:** Local persistence; no cloud latency or data egress fees.
- **Trade-offs:** Lacks the massive scale features of cloud options like Pinecone/ Weaviate (acceptable for this scope).

Key Trade-offs Made

- Selected a **smaller, faster embedding model** (MiniLM) to keep latency under 50ms.
- Mitigated potential **accuracy loss** by implementing **Hybrid Search** (adding KNN) to ensure exact keyword matches weren't missed.

Complexity vs. Maintainability:

- Chose Docling over simple text extractors. This added dependency complexity but was necessary to solve the "pain point" of broken tables in financial PDFs.
- Kept the architecture Serverless/Local (Colab-friendly) to avoid cloud infrastructure overhead during development.

Current Limitations & Next Steps | Scaling to Enterprise-Grade RAG

Current Limitations

1. Retrieval Precision & Noise

Issue: During stress testing, retrieved irrelevant chunks alongside relevant ones (high 'Hit Rate', potential answer degradation). **Root Cause:** Hybrid Search relies on similarity scores (>0.3) without a secondary 'Re-ranking' step to filter false positives.

Impact: Potential answer degradation.

2. Scalability & Concurrency

Issue: System uses local, file-based Vector DB (PersistentClient) and processes PDFs sequentially.

Impact: Significant latency bottlenecks with multiple simultaneous users; good for single-user demos only.

3. Complex Layout Context

Issue: Fixed-size chunking (800 chars) splits large financial tables, separating rows from headers.

Impact: LLM struggles to interpret 'orphan' table rows without header context.

Proposed Enhancements

Short-term ([Timeframe – e.g., Next 2 weeks])

- **Implement Re-Ranking:** Add Cross-Encoder (e.g., ms-marco-MiniLM) after retrieval to strictly score/filter chunks and eliminate 'noise'.
- **Metadata Filtering:** Update UI to filter searches by document type (e.g., 'Search only Invoices') to reduce search space.

Medium-term ([Timeframe - e.g., Next month])

- **Upgrade Embedding Model:** Switch to larger, GPU-accelerated model (like bge-m3) for better semantic nuances.
- **Semantic Chunking:** Replace fixed overlap with layout-aware splitter respecting table boundaries and page breaks.

Long-term Vision

- **Multi-Modal Capabilities:** Process charts, graphs, logos visually using Gemini's vision capabilities.
- **Cloud-Native Architecture:** Migrate ChromaDB to server-based instance (Docker/Kubernetes) for millions of documents and concurrent enterprise users.

Project Impact & Learning Outcomes

🧪 Key Technical Learnings ([Career goals or portfolio])

- **Hybrid Retrieval is Essential:** Learned that relying solely on Vector search fails #), while Keyword search for concepts. Combining them (Hybrid) both.
- **Data Quality is King:** The sophisticated RAG pipeline was useless without Docling's parse PDF tables into Markdown.
- **Prompt Engineering:** Discovered that strict constraints ("Answer only from "Cite source") reduced hallucinations compared to open-ended prompts.

💊 Business Impact Potential

- **Efficiency Gains:** Reduces document review time by ~90% (2.85s automated retrieval mirnal search per query).
- **Accuracy Improvement:** 98% Citation Accuracy ensures that every data point can be instantly verified, reducing compliance risks.
- **Scalability:** The pipeline can process hundreds of pages per minute, replacing the bottleneck of manual data data entry teams.

👨‍🎓 Skills Developed

- **GenAI Engineering:** RAG Pipelines, Vector Embeddings (Sentence-BERT), LLM (cemini API).
- **Full-Stack Python:** Gradio UI development, API integration, asynchronious data processing.
- **Data Engineering:** Unstructured data parsing (OCR), vector database management (ChromaDB).


Thank you