import osfrom PyPDF2 import PdfReaderimport pdfplumberfrom sentence\_transformers import SentenceTransformerimport pinecone# Initialize the embedding modelembedding\_model = SentenceTransformer('all-MiniLM-L6-v2')# Initialize the Pinecone vector databasepinecone.init(api\_key="<sk-proj-tskVYFwmETi2Y5LBUzYOEUG3aNRMDk7mPbFGBkhDzshAAakA9od-1II5A7mohQV4S8Lfxe3hjeT3BlbkFJ4Iynv04e4ddXEwZ0kQL7qlXF-Qnh-9gvats22RlSPU0zLmjB5wPIW\_J2b9F7eElpMB0ebVAesA>", environment="us-west1-gcp")index = pinecone.Index("rag-pipeline-index")# Function to extract text from PDF using pdfplumberdef extract\_text\_from\_pdf(pdf\_path): with pdfplumber.open(pdf\_path) as pdf: text\_data = [] for page in pdf.pages: text\_data.append(page.extract\_text()) return text\_data# Function to chunk text into smaller segmentsdef chunk\_text(text, max\_chunk\_size=300): words = text.split() chunks = [] current\_chunk = [] for word in words: current\_chunk.append(word) if len(current\_chunk) >= max\_chunk\_size: chunks.append(" ".join(current\_chunk)) current\_chunk = [] if current\_chunk: chunks.append(" ".join(current\_chunk)) return chunks# Function to embed and store chunks in Pineconedef store\_chunks\_in\_pinecone(chunks, metadata): for chunk in chunks: embedding = embedding\_model.encode(chunk).tolist() index.upsert([(metadata['id'], embedding, metadata)])# Main pipeline functiondef process\_pdf(pdf\_path): # Extract text from the PDF text\_data = extract\_text\_from\_pdf(pdf\_path) for page\_num, page\_text in enumerate(text\_data): # Chunk text into smaller pieces chunks = chunk\_text(page\_text) # Metadata for the page metadata = { "id": f"{os.path.basename(pdf\_path)}\_page\_{page\_num}", "file\_name": os.path.basename(pdf\_path), "page\_number": page\_num } # Store chunks in Pinecone store\_chunks\_in\_pinecone(chunks, metadata)# Example usageif \_\_name\_\_ == "\_\_main\_\_": pdf\_file\_path = "example.pdf" # Path to your PDF file process\_pdf(pdf\_file\_path) # Example query processing query = "What is the unemployment rate for those with a bachelor's degree?" query\_embedding = embedding\_model.encode(query).tolist() # Perform similarity search in Pinecone results = index.query(query\_embedding, top\_k=5, include\_metadata=True) for match in results["matches"]: print(f"Page {match['metadata']['page\_number']}: {match['metadata']}")