Extracting Context from PDFs and Web Content

1. PyMuPDF
   * Purpose: Extract text from PDFS
   * Implementation Code
     + import fitz # PyMuPDF
     + def extract\_text\_pymupdf(pdf\_path):
     + """Extracts text from a structured (non-scanned) PDF."""
     + doc = fitz.open(pdf\_path)
     + text = "\n".join(page.get\_text("text") for page in doc)
     + return text
     + pdf\_text = extract\_text\_pymupdf("example.pdf")
     + print(pdf\_text[:500]) # Preview first 500 characters
   * Reference: <https://pymupdf.readthedocs.io/en/latest/index.html>
2. PDFPulmber
   * Purpose: Extract content from tables and images
   * Implementation Code
     + import pdfplumber
     + def extract\_text\_pdfplumber(pdf\_path):
     + """Extracts text from PDFs, including tables."""
     + text = ""
     + with pdfplumber.open(pdf\_path) as pdf:
     + for page in pdf.pages:
     + text += page.extract\_text() + "\n"
     + return text
     + pdf\_text = extract\_text\_pdfplumber("example.pdf")
     + print(pdf\_text[:500])
   * Reference: <https://github.com/jsvine/pdfplumber>
3. Beautiful Soup
   * Purpose: Extracting text from webpages
   * Code
     + import requests
     + from bs4 import BeautifulSoup
     + def extract\_text\_webpage(url):
     + """Extracts text from an article webpage."""
     + response = requests.get(url)
     + soup = BeautifulSoup(response.text, "html.parser")
     + # Extract text from paragraphs
     + text = "\n".join(p.get\_text() for p in soup.find\_all("p"))
     + return text
     + web\_text = extract\_text\_webpage("https://example.com/article")
     + print(web\_text[:500])

Combined Extraction Pipeline

* Analyzes the type of document provided and uses the appropriate context extraction method.
* Code
  + def extract\_text(source):
  + """Extracts text from PDF or web URL."""
  + if source.endswith(".pdf"):
  + return extract\_text\_pymupdf(source) # Use PyMuPDF for fast extraction
  + elif source.startswith("http"):
  + return extract\_text\_webpage(source) # Use BeautifulSoup for web articles
  + else:
  + raise ValueError("Unsupported file type. Provide a PDF or URL.")
  + # Example usage
  + pdf\_text = extract\_text("example.pdf")
  + web\_text = extract\_text("https://example.com/article")
  + print("Extracted PDF Text:", pdf\_text[:500])
  + print("Extracted Web Text:", web\_text[:500])

Providing the Extracted Context to the Generative AI model

* Code
  + from vllm import LLM
  + llm = LLM(model="mistral-7b")
  + prompt = f"Create a course module using this material:\n\n{pdf\_text[:2000]}" # Trim long text
  + response = llm.generate([prompt])
  + print(response[0].outputs[0].text)