Newspaper Opinion/Editorial Extraction & Summarization Pipeline

Candidate: Ishita Shegaonkar
Date: 25-09-2025

1. Objective

The assignment required the development of a **fully automated pipeline** to:

1. Identify Opinion/Editorial pages in multiple newspapers (PDF format).
2. Extract only those pages.
3. Merge extracted pages into a **single consolidated PDF**.
4. Generate summaries of the extracted pages using an **LLM**.

The pipeline needed to work with minimal human intervention, be terminal-friendly, and optionally work on Linux.

2. Approach Step 1: PDF Collection

- All newspaper PDFs are placed in the input_pdfs/ folder.
- The pipeline processes **all PDFs dynamically**, no manual selection required.

Step 2: Page Extraction

- Each page is analyzed using **PyPDF2** to extract text.
- **OCR fallback** is implemented with pytesseract for scanned pages without extractable text.
- **Keyword matching** identifies relevant pages containing “opinion” or “editorial”.
- Relevant pages are added to a **merged PDF** using PdfWriter.

Step 3: Summarization

- Text from extracted pages is passed to an **LLM** (Groq) for summarization.
- Summaries are saved in both .txt and .md formats alongside the merged PDF.
- Batch processing and retry mechanisms handle API rate limits.

Step 4: Output

- Consolidated PDF: final/opinion_editorials.pdf
- Summaries: final/summaries_<timestamp>.txt and .md

3. Tools and Technologies

- **Python 3.10+**: Main programming language.

- **PyPDF2**: PDF reading and merging.
- **pdf2image + pytesseract**: OCR for scanned pages.
- **Groq LLM**: Automated text summarization.
- **Poppler utilities**: Required by pdf2image for PDF-to-image conversion.
- **Cross-platform**: Scripts tested on both Windows and Linux.

---

4. Challenges and Solutions

- **Scanned PDFs with no text** → Solved with OCR fallback using Tesseract.
- **API rate limits** → Handled with retry logic and batching for summarization.
- **Cross-platform compatibility** → Environment variables and path handling implemented for Linux/Windows.
- **Keyword variations** → Case-insensitive matching to ensure all editorial pages are detected.

---

5. Key Highlights

- Fully **automated pipeline**, no manual page selection needed.
- Efficient **logging** for traceability of processed pages.
- Summaries enhance readability and provide quick insights.
- Output folder structure ensures **organized, professional deliverables**.

---

6. Submission Contents

1. **GitHub Repo**: All scripts, README, and setup instructions.
2. **Merged PDF**: final/opinion_editorials pdf.
3. **Summary File**: final/summaries_<timestamp>.txt or .md.
4. **README**: Explains pipeline setup, usage, and cross-platform instructions.

---

7. Conclusion

This solution demonstrates the ability to:

- Break down complex tasks into manageable steps.
- Leverage Python libraries and LLMs for real-world data extraction and summarization.
- Deliver professional, automated outputs suitable for review.