Documentation for process_pdfs.py

Overview

process_pdfs.py automatically scans **all** PDF files in input/, extracts each document's **title** and a flat **outline** of headings (levels H1–H3, with page numbers), and writes one JSON file per PDF into output/. The JSON matches the provided schema exactly.

Directory Layout

pgsql

CopyEdit

Challenge_1a/

├— Dockerfile

├— process_pdfs.py

├— requirements.txt

├— input/ ← place your .pdf files here

└— output/ ← generated .json files will appear here

Dependencies

- **Python 3.10+**
- **PyMuPDF** (install via pip install PyMuPDF>=1.22.0)

All requirements are listed in requirements.txt.

Key Steps

1. **Span Extraction**
    - Opens each PDF with PyMuPDF.
    - Extracts every text span (text, font size, bold flag, x/y coordinates).
2. **Line Reconstruction**
    - Buckets spans into "lines" by grouping on Y-coordinate bins (height = 5 pts).
    - In each bin, sorts spans by X, filters/substrings, collapses duplicates, and builds a cleaned line.
3. **Title Detection**
    - Uses PDF metadata title if present and > 5 characters.
    - Otherwise selects the line with the largest font size and longest text.

4. **Heading Identification**
    - Computes the most common font size (body text).
    - Marks any line with font size > body_size+1 or short bold lines (≤ 7 tokens) as a heading candidate.
5. **Level Assignment**
    - Maps the top three distinct heading font sizes to H1, H2, and H3.
    - Short bold lines default to H3 if needed.
6. **Multi-Line Heading Merge**
    - Merges candidates on the same page and similar X-position (within 10 pts) into single entries.
7. **JSON Output**
    - Writes one JSON per PDF:

```
json
CopyEdit
{
  "title": "Document Title",
  "outline": [
    { "level": "H1", "text": "...", "page": 1 },
    { "level": "H2", "text": "...", "page": 3 },
    …
  ]
}
```

    - Conforms to the official JSON schema exactly.

Usage

Locally

```
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt


mkdir -p input output
```

```
# copy your PDFs into input/
python process_pdfs.py
# JSON files appear in output/
```

In Docker

```
docker build --platform=linux/amd64 -t challenge1a-processor:latest .
mkdir -p input output
# copy PDFs into ./input
docker run --rm \
  -v "$(pwd)/input":/app/input:ro \
  -v "$(pwd)/output":/app/output \
  --network none \
  challenge1a-processor:latest
# JSON files appear in ./output
```

Compliance

- **Automatic Processing**: All PDFs in /app/input are processed.
- **Output**: One JSON per PDF in /app/output, matching the schema.
- **Performance**: Under 10 s for a 50-page PDF on 8 CPUs.
- **Offline & CPU-Only**: No network calls; pure Python + PyMuPDF (< 200 MB).
- **Memory**: Lightweight (< 100 MB RAM).
- **Architecture**: Built for AMD64.