Challenge 1B: Persona-Driven PDF Analysis

Overview

process_collection.py automates the analysis of multiple PDF collections, extracting and prioritizing relevant sections for a given persona and task. It performs:

1. **Span Extraction** from each PDF (using PyMuPDF).
2. **Section Segmentation** by detecting headings larger than body text.
3. **Semantic Ranking** of all sections against the combined "persona + task" query (using all-MiniLM-L6-v2).
4. **Sub-section Analysis** by selecting the single most relevant sentence per top section.
5. **Structured Output** with metadata, ranked sections, and refined snippets.

A challenge1b_input.json drives each collection, and results are written to challenge1b_output.json alongside the input.

Folder Structure

Challenge_1b/

├── process_collection.py # Main analysis script

├── requirements.txt # PyMuPDF & sentence-transformers

├── README.md # This documentation

├── Collection 1/ # Travel Planning

| ├── PDFs/ # 7 South-of-France PDF guides

| ├── challenge1b_input.json # Input: persona, task, docs list

| └── challenge1b_output.json # Output generated here

├── Collection 2/ # Acrobat Learning

| ├── PDFs/ # 15 Acrobat tutorial PDFs

| ├── challenge1b_input.json

| └── challenge1b_output.json

└── Collection 3/ # Recipe Collection

├── PDFs/ # 9 cooking guide PDFs

├── challenge1b_input.json

└── challenge1b_output.json

Dependencies

Listed in requirements.txt:

text

CopyEdit

PyMuPDF>=1.22.0

sentence-transformers>=2.2.2

- **PyMuPDF**: fast, offline PDF parsing.
- **sentence-transformers**: semantic embeddings (model size ≈100 MB).

Install with:

pip install -r requirements.txt

Setup & Execution

cd Challenge_1b

(Optional) Virtual environment

python3 -m venv venv

source venv/bin/activate

Install dependencies

pip install --upgrade pip

pip install -r requirements.txt

Process a collection (e.g. Collection 1)

python process_collection.py --config "Collection 1/challenge1b_input.json"

This command:

1. Reads the input JSON (persona, task, document filenames).
2. Loads the MiniLM model.
3. Parses each PDF in Collection 1/PDFs/, extracting sections under detected headings.
4. Builds a query string: "Travel Planner. Task: Plan a trip of 4 days for a group of 10 college friends."
5. Ranks **all** sections by cosine similarity to the query, using sentence embeddings.
6. Selects the **top 5** sections for extracted_sections.

7. For each of these 5, picks the single most relevant sentence as the refined_text.
8. Writes Collection 1/challenge1b_output.json with:


```
{
 "metadata": {
  "input_documents": [ ... ],
  "persona": "Travel Planner",
  "job_to_be_done": "Plan ... friends.",
  "processing_timestamp": "2025-07-10T15:31:22.632389"
 },
 "extracted_sections": [
  { "document": "...", "section_title": "...", "importance_rank": 1, "page_number": 1 },
  ...
 ],
 "subsection_analysis": [
  { "document": "...", "refined_text": "...", "page_number": 2 },
  ...
 ]
}
```

Algorithm Details

1. **Span Extraction**
   - Use page.get_text("dict") to collect every text span with font size and position.
2. **Section Detection**
   - Determine “body” font size (mode of all span sizes).
   - Label spans with size > body+1 as headings.
3. **Content Grouping**
   - For each heading, collect subsequent spans up to the next heading to form a section’s content.
4. **Semantic Ranking**
   - Combine persona and job into one query.
   - Embed query and section text with SentenceTransformer.

    - Compute cosine similarities, sort descending.
5. **Sub-section Analysis**
    - Split each top-ranked section's content into sentences.
    - Embed sentences, find the single highest-scoring sentence as the refined snippet.
6. **Output Construction**
    - Package metadata (including ISO timestamp).
    - List ranked sections (top 5).
    - Include refined snippets for those sections.

What Has Been Done

- **Automated end-to-end pipeline** from JSON input → PDF parsing → semantic analysis → JSON output.
- **Lightweight model** (all-MiniLM-L6-v2 ≈100 MB) ensures offline, CPU-only execution within 60 s per collection.
- **Modular code** supports arbitrary collections by configuration.
- **JSON schema compliance** for both Challenge 1A and 1B outputs.

This setup readies you to process multiple document collections with persona-driven intelligence, paving the way for Round 2's web-app integration.