Introduction

This report explains the design, purpose, and operation of a Python-based tool for extracting meaningful information—like titles and outlines—from PDF documents. The tool is intended for batch processing of many PDFs at once, making it useful for tasks such as digital archiving, document indexing, or generating structured metadata for large collections of files.

The system is designed with performance, reliability, and readability in mind. It uses multithreading to scale across multiple PDFs, applies intelligent heuristics to extract titles and outlines, and stores the results in clean JSON files for downstream use.

Libraries Used

To achieve this functionality, the tool relies on a few powerful Python libraries:

- **PyMuPDF (imported as `fitz`)**: This is the core engine for working with PDFs. It allows access to text blocks, spans, font sizes, and coordinates. Without this, precise control over text extraction wouldn't be possible.

- **pathlib & os**: These are used for managing file paths and directories. They help in locating PDFs, creating output folders, and working across different operating systems.

- **json**: Once data is extracted (title and outline), it needs to be saved in a structured format. JSON is used here due to its simplicity and compatibility.

- **re (regular expressions)**: Text extracted from PDFs often contains irregularities like excessive whitespace, dashes, or newlines. Regular expressions are used to clean and normalize this text.

- **concurrent.futures (ThreadPoolExecutor)**: Some operations (like processing pages or multiple PDFs) are parallelized using threads. This makes the tool much faster for bulk processing.

What the Tool Does

The main goal of the script is to:

1. Extract the **document title** from the first page of each PDF.

2. Build a list of **outline entries** (like headings) across the first 50 pages.
3. Save this information in a JSON file named after the original PDF.

This is done in a clean, accurate, and performance-optimized way.

How It Works (Step-by-Step)

Step 1: Initial Setup

The tool defines a class called `PDFProcessor`, which is responsible for the entire process. When initialized, it prepares memory caches to store previously computed results, avoiding duplicate effort.

Step 2: Cleaning the Text

Before any data is used, it's passed through a `clean_text()` method. This:

- Removes dashes (---).
- Merges hyphenated line breaks (e.g., "exam-\nple" becomes "example").
- Strips excessive whitespace and newlines.
- Ensures all text is in a readable and normalized format.

Step 3: Extracting the Title

The `extract_title()` method processes the **first page** of each PDF. Here's how it identifies the title:

- It scans every text span on the page.
- For each span, it looks at:
    - The **font size** (bigger text is likely more important).
    - The **vertical position** (text closer to the top is more likely to be a title).

- A **weight** is calculated using both size and position.
- The span with the highest weight is selected as the title, as long as it isn't too short or decorative (like a line of dashes or just numbers).

Step 4: Extracting the Outline

This happens across up to 50 pages. Each page is sent to a thread for processing in parallel. The `extract_page_outline()` method does the following:

- It reads two kinds of data:
    - **Formatted spans** (with font size and position).
    - **Raw word layout**, to reconstruct full lines.
- It groups text lines by their Y-position.
- It cleans the text and checks if a line could be a heading.
- A line is considered a heading if:
    - It's not a URL or all-uppercase junk.
    - It has a font size above a threshold (12+).
- Font size determines heading level:
    - H1: Very large (20+)
    - H2: Large (16–19)
    - H3: Medium (14–15)
    - H4: Lower, but still distinguishable (12–13)

Each outline entry is stored with:

- The heading level (H1–H4)
- The text content
- The page number (1-based)

It also skips duplicates and any heading that's identical to the document title.

Step 5: Writing Output

Once the title and outline are extracted:

A dictionary is created with the following structure:

json
CopyEdit

```
{
  "title": "Sample Title",
  "outline": [
    {
      "level": "H1",
      "text": "Introduction",
      "page": 1
    },
    ...
  ]
}
```

- This data is saved to a JSON file named like the original PDF but with a `.json` extension.
- All output files are stored in `/app/output`.

Step 6: Bulk Processing

The method `process_pdfs()` is the orchestrator. It:

- Looks for all `.pdf` files in `/app/input`.
- Runs `process_pdf()` on each one in parallel.
- Prints logs as each file is processed.
- Collects and prints any errors at the end.

Performance Optimizations

The tool uses several techniques to make it fast and efficient:

- **Multithreading**: Speeds up page processing and file handling.
- **Caching**: Stores previously seen titles/outlines to avoid recomputation.
- **Selective Scanning**: Only processes the first 50 pages to avoid unnecessary work on very large documents.

Strengths of the Tool

- Fully automated and scalable for batches of PDFs.
- Very few false positives in title and outline detection due to smart filters.
- Clean and structured output in standard JSON.
- Resilient to errors—continues processing other files if one fails.
- Built-in text cleaning for noisy or inconsistent PDF content.

Potential Improvements

While effective, the tool can be further enhanced:

- Add **OCR support** for scanned or image-based PDFs.
- Improve **hierarchy detection**, grouping subheadings under headings.
- Add a **graphical interface (GUI)** or command-line arguments for user control.
- Export results in multiple formats like XML or CSV.

Conclusion

This PDF processor is a highly efficient and structured tool for extracting essential metadata—like titles and outlines—from PDFs. With careful use of PyMuPDF and multithreading, it can handle a large number of documents quickly and reliably. The modular design makes it easy to maintain and extend, while the thoughtful logic ensures accuracy.