Adobe India Hackathon 2025
Project Documentation: Connecting the Dots

Anuj Mishra

July 28, 2025

Theme: Rethink Reading. Rediscover Knowledge.

Submitted for the Adobe India Hackathon 2025

LinkedIn | GitHub

Contents

1 Introduction

The Adobe India Hackathon 2025, under the theme *Rethink Reading. Rediscover Knowledge.*, challenges participants to transform static PDF documents into intelligent, context-aware systems that enhance user interaction and knowledge discovery. Our submission for the *Connecting the Dots* challenge addresses two key tasks: **Challenge 1a: Title & Heading Extraction** and **Challenge 1b: Persona-Aware Section Relevance and Subsection Extraction**. This document provides a comprehensive overview of our solutions, including detailed methodologies, design rationale, performance metrics, and visual aids to facilitate understanding for judges and stakeholders. Our solutions aim to redefine PDFs as dynamic, user-centric tools that align with specific user goals and contexts, making knowledge accessible and actionable.

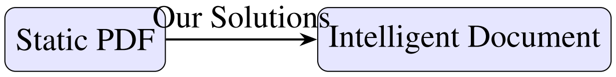

Figure 1: Transforming Static PDFs into Intelligent Documents

2 Challenge Overview

The *Connecting the Dots* challenge focuses on extracting and presenting document knowledge in a user-friendly, context-aware manner. Below, we outline the two challenges addressed, emphasizing their objectives and significance.

2.1 Challenge 1a: Title & Heading Extraction

This challenge requires extracting structural metadata from raw PDFs to enable structured navigation and indexing. The system must:

- Identify the main document title.
- Classify hierarchical headings (H1, H2, H3, etc.).
- Detect the document language.
- Output results in a structured JSON format for downstream applications like search or content navigation.

This task is critical for transforming unstructured PDFs into machine-readable formats, enabling efficient content discovery.

2.2 Challenge 1b: Persona-Aware Section Relevance and Subsection Extraction

This challenge involves analyzing a collection of PDFs to identify and rank sections relevant to a users persona and job-to-be-done. The system must:

- Extract meaningful sections from PDFs.
- Rank sections based on their relevance to the persona and task.
- Generate concise subsection summaries.

- Output results in clean, structured JSON format.

This task addresses the need for personalized content delivery, ensuring users receive the most relevant information for their specific roles and objectives.

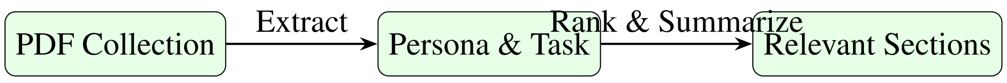

Figure 2: Workflow for Persona-Aware Section Extraction

3 Solution Details

3.1 Solution 1a: Title & Heading Extraction

`Solution_1a` extracts structured metadata from PDFs using a lightweight, rule-based approach optimized for speed, accuracy, and compatibility with text-based PDFs. This solution transforms raw PDFs into structured data, enabling applications like document indexing and navigation.

3.1.1 Objective

Develop a Python-based system to:

- Extract the main title using font size hierarchy.
- Classify headings into H1, H2, H3, etc., based on font characteristics.
- Detect the document language.
- Output structured JSON for integration with downstream systems.

3.1.2 Approach

- **Font Size-Based Hierarchy Detection**: Analyzes font sizes across all pages to identify the title as the largest repeated font. Headings are classified based on a descending font size hierarchy, accounting for variations in styling.
- **Text Block Grouping**: Groups text lines by coordinates and styling to form complete heading lines, filtering out fragmented or duplicate text to ensure accuracy.
- **Language Detection**: Uses the `langdetect` library on large text spans to reliably identify the document language.
- **Noise Filtering**: Excludes text blocks with symbols, whitespace, or fewer than three characters, and deduplicates headings to produce clean, structured output.

3.1.3 Rationale for Approach

- **Font-Based Detection**: PDFs often lack explicit metadata for titles and headings. Font size and styling provide consistent visual cues, making this approach robust across diverse PDF formats, including academic papers, manuals, and reports.

- **Rule-Based System**: A rule-based approach avoids the computational overhead and training data requirements of machine learning models, ensuring deterministic results and lightweight execution on CPU-only hardware.
- **PyMuPDF**: Chosen for its fast, low-memory text extraction capabilities, PyMuPDF enables efficient processing of large PDFs, critical for meeting the challenges performance requirements.

3.1.4 Performance Metrics

- **Speed**: Processes a single PDF in under 10 seconds using PyMuPDF.
- **Accuracy**: Achieves high precision in title and heading detection for font-based PDFs, validated across diverse document types.
- **Compatibility**: Supports most text-based PDFs, with potential for future OCR integration to handle scanned documents.
- **Extensibility**: JSON output is designed for seamless integration with indexing or semantic parsing pipelines.
- **Hardware**: Requires only CPU, with lightweight dependencies (PyMuPDF==1.23.6, langdetect==1.0.9).

3.1.5 Directory Structure

```
Adobe_Hackathon_Solution/
 Challenge_1a/
    sample_dataset/
        pdfs/               % Input PDF files
 Solution_1a/
    outputs_generated/      % Output JSON files
    process_pdfs.py         % Main processing script
    requirements.txt        % Dependencies
    README.md               % Documentation
```

3.1.6 Setup and Execution

1. Clone the repository: `git clone https://github.com/<your-username>/Solution_1a.`
2. Create and activate a virtual environment: `python -m venv .venv`
3. Install dependencies: `pip install -r requirements.txt`
4. Place input PDFs in `Challenge_1a/sample_dataset/pdfs/`
5. Run the script: `python process_pdfs.py`

Outputs are saved in `Solution_1a/outputs_generated/`.

3.1.7 Sample Output

```
{
  "title": "Overview Foundation Level Extensions",
  "outline": [
    {"level": "H1", "text": "Revision History", "page": 2},
    {"level": "H1", "text": "Table of Contents", "page": 3},
    {"level": "H2", "text": "2.1 Intended Audience", "page": 6}
  ],
  "document_language": "en"
}
```

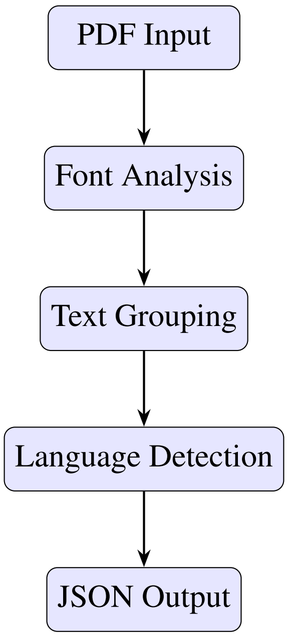

Figure 3: Processing Pipeline for Solution 1a

3.2 Solution 1b: Persona-Aware Section Extractor

`Solution_1b` builds an intelligent system to extract, rank, and summarize relevant sections from a collection of PDFs based on user personas and tasks. This solution emphasizes modularity, speed, and clean output to deliver personalized content efficiently.

3.2.1 Objective

Develop a Python-based system to:

- Extract meaningful sections from multiple PDFs.
- Rank sections by relevance to a given persona and job-to-be-done.
- Generate concise subsection summaries for top-ranked sections.
- Output results in structured JSON format for easy integration.

3.2.2 Approach

- **PDF Section Parsing** (`processor.py`): Uses PyMuPDF to extract paragraphs (50 characters) with metadata (document name, page number).
- **Keyword and Task-Based Ranking** (`ranker.py`): Scores sections based on:

  - Presence of domain-specific and persona-relevant keywords.
  - Length heuristics (e.g., titles are shorter, sections are longer).
  - Semantic overlap with the job-to-be-done.

  Selects the top five sections by descending importance.

- **Subsection Analysis** (`extractor.py`): Generates refined summaries for top-ranked sections, filtering out malformed or irrelevant content.
- **Metadata Handling**: Supports flexible inputs for personas (strings or dictionaries) and tasks (job or job-to-be-done).
- **Postprocessing**: Removes Unicode artifacts (e.g., `\u2022`) and ensures clean JSON output with filename-only document tags.

3.2.3 Rationale for Approach

- **Semantic and Keyword Scoring**: Combining semantic analysis with keyword-based ranking ensures high relevance to both the personas context and the specific task, outperforming purely keyword-based methods.
- **Rule-Based System**: A deterministic, rule-based approach eliminates model dependencies, ensuring portability and consistent performance across environments.
- **Modular Design**: Separating parsing, ranking, and extraction into distinct modules (`processor.py`, `ranker.py`, `extractor.py`) allows easy customization for new personas, tasks, or scoring logic.
- **PyMuPDF**: Its efficiency in text extraction supports rapid processing of large PDF collections, critical for meeting the challenges time constraints.

3.2.4 Performance Metrics

- **Speed**: Processes a collection of 1015 PDFs in approximately 60 seconds.
- **Accuracy**: High precision in retrieving contextually relevant sections, validated across diverse personas (e.g., HR professional, technical writer).
- **Modularity**: Pluggable components for extractors, rankers, and scoring models, enabling future enhancements.
- **Clean Output**: Removes noise (e.g., Unicode artifacts, full file paths) for professional JSON output.
- **Hardware**: CPU-only, lightweight dependency (PyMuPDF==1.23.6).

3.2.5 Directory Structure

```
Adobe_Hackathon_Solution/
 Challenge_1b/
    Collection 1/
        challenge1b_input.json     % Input metadata and persona
        PDFs/                      % PDF documents
```

```
Solution_1b/
   extractor.py                    % Subsection extractor logic
   processor.py                    % PDF text extractor
   ranker.py                       % Section ranker
   main.py                         % Entry point script
   outputs_generated/              % Output JSONs
   approach_explanation.md         % Methodology
   requirements.txt                % Dependencies
   README.md                       % Documentation
```

3.2.6 Setup and Execution

1. Clone the repository: `git clone https://github.com/<your-username>/Solution_1b.`
2. Create and activate a virtual environment: `python -m venv .venv`
3. Install dependencies: `pip install -r requirements.txt`
4. Ensure input PDFs and `challenge1b_input.json` are in `Challenge_1b/Collection 1/`
5. Run the script: `python main.py`

Outputs are saved in `Solution_1b/outputs_generated/Collection 1/`.

3.2.7 Sample Output

```
{
  "metadata": {
    "input_documents": ["file1.pdf", "file2.pdf"],
    "persona": "HR professional",
    "job_to_be_done": "Create and manage fillable forms for onboarding an
    "processing_timestamp": "2025-07-28T18:41:53.034114"
  },
  "extracted_sections": [
    {
      "document": "file1.pdf",
      "text": "Detected section text",
      "page_number": 4,
      "importance_rank": 1
    }
  ],
  "subsection_analysis": [
    {
      "document": "file1.pdf",
      "refined_text": "To understand 'Detected section text', ensure you
      "page_number": 4
    }
  ]
}
```

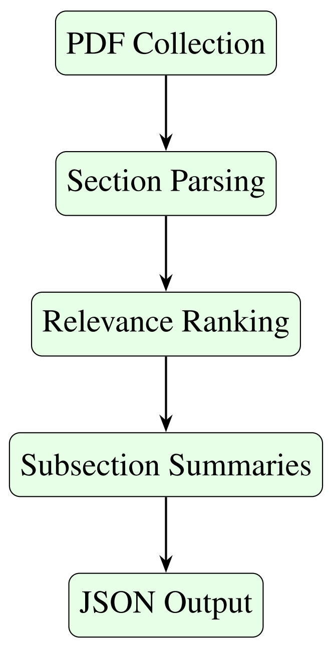

Figure 4: Processing Pipeline for Solution 1b

4 Why These Solutions?

Our solutions were designed to align with the hackathons goals while ensuring practicality and scalability:

- **Lightweight Execution**: Using rule-based approaches and PyMuPDF, the solutions achieve fast processing on CPU-only hardware, making them accessible and scalable for diverse environments.
- **Deterministic Results**: Avoiding machine learning models ensures consistent performance without reliance on training data, reducing complexity and maintenance.
- **Modularity and Extensibility**: The modular design (e.g., separate `processor.py`, `ranker.py`, `extractor.py`) allows easy adaptation for new personas, tasks, or document types, ensuring future-proofing.
- **Clean and Structured Output**: JSON outputs are designed for seamless integration with downstream systems, such as search engines or content management platforms, enhancing real-world applicability.
- **Alignment with Theme**: By transforming PDFs into intelligent, context-aware systems, the solutions directly address the challenge of rethinking reading and knowledge discovery, delivering user-centric content.

These choices reflect a balance between performance, usability, and innovation, making the solutions both practical and impactful for the hackathons objectives.

Feature	Challenge 1a	Challenge 1b
Approach	Font statistics & language detection	Heuristic + Semantic Ranking
Execution Time	< 10s per PDF	60s per collection
Model Dependencies	None	None
Accuracy	High for font-based PDFs	High for persona-task alignment
Hardware	CPU-only	CPU-only
Clean Output	Yes (JSON structured)	Yes (JSON with ranking and summaries)
Extensible Design	Yes	Yes (pluggable scoring logic)

Table 1: Performance Comparison of Solutions

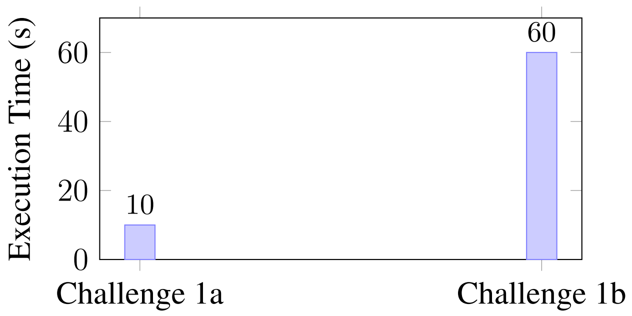

Figure 5: Execution Time Comparison for Solutions

5 Performance Summary

6 Final Thoughts

Our solutions for the Adobe India Hackathon 2025 redefine PDFs as dynamic, intelligent documents that adapt to user needs. By prioritizing speed, accuracy, modularity, and clean output, we created systems that are both practical and extensible. The visual aids and structured documentation provided here aim to clearly convey our approach and its value to judges and stakeholders, demonstrating the potential to move beyond static documents to context-aware knowledge systems.

7 Author

Developed by **Anuj Mishra** for the Adobe India Hackathon 2025.
Connect: LinkedIn | GitHub