
Advanced PDF Document Outline Extractor
Technical Documentation & User Guide
Project: Adobe India Hackathon Challenge 1a
Author: IshSharma41
Date: July 2025

Executive Summary

The Advanced PDF Document Outline Extractor is a sophisticated Python-based system designed to automatically extract hierarchical outlines and metadata from PDF documents. This solution leverages advanced text analysis, multilingual Natural Language Processing (NLP), and intelligent document structure recognition to generate structured JSON outputs containing document titles and hierarchical outlines.

Project Overview

This innovative system processes PDF documents through an 8-stage pipeline that combines rule-based heuristics with NLP-powered intelligence to deliver accurate heading classification and meaningful title extraction across multiple languages and document formats.

Key Capabilities:

- Intelligent text extraction using PyMuPDF for robust document parsing
- Multilingual language detection supporting 15+ languages including CJK scripts
- Smart heading classification using dynamic threshold analysis
- Hierarchical outline structuring with logical flow validation
- Automated title derivation from document content analysis

Technical Architecture

System Design Philosophy

Our solution employs a **hybrid approach** that combines the reliability of rule-based systems with the intelligence of modern NLP technologies. This design ensures both accuracy and adaptability across diverse document types and languages.

Core Methodology

1. Rule-Based Heuristics

- Dynamic font size analysis and positioning pattern recognition
- Formatting-based classification using typography cues
- Contextual relationship analysis between text elements

2. NLP-Powered Intelligence

- Multilingual text analysis for content quality assessment
- Semantic understanding for meaningful content extraction
- Language-specific processing optimizations

3. Contextual Feature Engineering

- 15+ engineered features including font prominence, text centering, vertical gaps
- Document-specific adaptive thresholds
- Hierarchical relationship validation

4. Adaptive Processing

- Document-specific font size threshold calculation
- Language-aware text processing and truncation
- Script-specific handling for different writing systems

Key Innovations

- **Language-Aware Processing**: Script detection with tailored processing (CJK, Latin, Arabic, Cyrillic)
- **Intelligent Fragment Merging**: Combines broken text spans and reconstructs line-wrapped content
- **Multi-Strategy Title Extraction**: Combines content analysis with metadata examination
- **Hierarchical Validation**: Ensures logical heading flow (H1→H2→H3→H4)

Technology Stack & Dependencies

Core Technologies

- **PDF Processing**: PyMuPDF (1.24.1)
- **Language Detection**: SpaCy + spacy-langdetect
- **Multilingual NLP**: xx_ent_wiki_sm (15MB)

- **English NLP**: en_core_web_sm (12MB)
- **Machine Learning**: scikit-learn
- **Data Handling**: NumPy, Pandas
- **Progress Tracking**: tqdm (4.66.2)

Architecture Benefits

- Offline Operation (no internet required)
- Language-Agnostic (supports 15+ languages)
- Scalable batch processing
- Robust, fail-safe implementation

Processing Pipeline

Stage 1: Initial Sampling
Stage 2: Language Detection
Stage 3: Full Text Extraction
Stage 4: Text Block Analysis
Stage 5: Heading Classification
Stage 6: Title Derivation
Stage 7: Outline Structuring
Stage 8: Output Generation

Each stage contributes to reliable heading detection and title extraction.

Installation & Setup

System Requirements

- **CPU**: x86_64 (Intel/AMD)
- **RAM**: 4GB minimum (8GB recommended)
- **Storage**: 200MB minimum
- **OS**: Linux/macOS/Windows
- **Python**: 3.10+
- **Docker**: Latest

Installation Methods

1. Docker Deployment (Recommended)

```
git clone https://github.com/kushagra8881/adobe_india_hackathon.git

cd adobe_india_hackathon/Challenge_1a

docker build -t pdf-outline-extractor .
```

2. Local Python Installation

```
python -m venv foradobe

source foradobe/bin/activate

pip install -r requirements.txt

python download_models.py
```

Usage Instructions

Docker Execution:

```
mkdir -p inputs outputs

docker run -v $(pwd)/inputs:/app/inputs -v $(pwd)/outputs:/app/outputs pdf-outline-
extractor
```

Local Execution:

```
python main.py
```

The system processes each PDF in the inputs folder and generates JSON files in the outputs folder.

Output Format & Structure

JSON Schema Example:

```
{
  "title": "Document Title Extracted from Content",
  "outline": [
    { "level": "H1", "text": "Chapter 1: Introduction", "page": 1 },
    { "level": "H2", "text": "1.1 Background", "page": 2 }
  ]
}
```

- H1–H4 headings with smart scoring
- Page-level accuracy
- Multilingual support
- Content-based title derivation

Language Support

Supported Languages:

- English (Latin)
- Chinese/Japanese/Korean (CJK)
- Arabic (RTL)
- Russian (Cyrillic)
- Hindi (Devanagari)
- French, German, Spanish, Dutch, etc.

Detection Accuracy: 90–99%
Processing Quality: Good to Excellent
Truncation: Word-based or character-based depending on script

---

Performance Metrics

- **Small PDFs (1–10 pages)**: 2–4 seconds
- **Medium (11–30 pages)**: 4–6 seconds
- **Large (31–50 pages)**: up to 10 seconds

Accuracy Benchmarks:

Metric	Accuracy
Heading Detection	85–95%
Language Detection	95%+
Title Extraction	80–90%
Hierarchy Validation	90%+

Resource Usage:

- Memory: 2–4 GB
- CPU: Single-threaded
- Storage: ~100MB

Configuration & Customization

Examples from configuration files:

extract_blocks.py

FONT_SIZE_TOLERANCE_MERGE = 0.5

PAGE_MARGIN_HEADER_FOOTER_PERCENT = 0.15

classify_headings.py

MIN_CONFIDENCE = {"H1": 15.0, "H2": 10.0}

WEIGHTS = {"font_size_prominence": 4.5, "is_bold": 5.0}

structure_outline.py

MAX_TITLE_WORDS = 7

MAX_TITLE_CHARS_CJK = 20

Troubleshooting Guide

Model Loading Error:

- Run: python download_models.py

Memory Error:

- Use machines with ≥8GB RAM
- Limit batch size

Unicode Errors:

- Check for non-text/image-only PDFs
- Ensure UTF-8 support

Poor Heading Detection:

- Adjust font size thresholds
- Check document formatting consistency

Docker Configuration Details

Features:

- Offline model preloading
- Small image footprint
- Platform-specific builds (linux/amd64)
- Volume mapping for input/output

Advanced Build Examples:

```
docker build --target production -t pdf-extractor:prod .
```

Development & Contributing

Setup:

```
git clone <repo>

cd Challenge_1a

python -m venv dev-env

source dev-env/bin/activate

pip install -r requirements.txt
```

Standards:

- Code Formatting: Black
- Linting: Flake8
- Type Checking: MyPy
- Testing: Pytest

Contribute:

- Fork → Create branch → Code → Run checks → PR

Technical Specifications

Component	Requirement	Recommended
OS	Linux/macOS/Windows	Linux
Python	3.10+	3.11
RAM	4GB	8GB
CPU	x86_64	Multi-core
Storage	200MB	500MB

Core Libraries: PyMuPDF, SpaCy, NumPy, Pandas, Scikit-learn
NLP Models: xx_ent_wiki_sm, en_core_web_sm
Utilities: tqdm, joblib, langdetect

Support & Resources

- Internal Documentation & Code Comments
- SpaCy: https://spacy.io/usage
- PyMuPDF: https://pymupdf.readthedocs.io/
- Docker Docs: https://docs.docker.com/develop/dev-best-practices/

For help:

1. Review this documentation
2. Check GitHub issues
3. Submit new issue with logs and PDFs

Acknowledgments

- **Adobe India** – For hosting this challenging hackathon
- **SpaCy** – For NLP tools
- **PyMuPDF** – For document parsing
- **Open Source Community** – For foundational libraries

Conclusion

The Advanced PDF Document Outline Extractor offers a powerful solution for intelligent document structure analysis. It combines heuristic and NLP-based strategies, supports 15+ languages, and operates fully offline. With scalable Docker support and smart processing pipelines, it stands as a robust tool for real-world document intelligence tasks.

Built with ❤️ for Adobe India Hackathon Challenge 1a
Advancing document intelligence through multilingual AI and smart text analysis

---

Document Version: 1.0
Last Updated: July 27, 2025

Let me know if you want this in .docx or .pdf format directly.