Financial Document Information Extractor - Usage Guide

Overview

This tool extracts specific information attributes from PDF documents (especially annuity policies) for Financial Services Compliance review. It combines OCR for image-based content and LLM processing for intelligent extraction.

Features

- **OCR Processing**: Uses EasyOCR with PyTorch for high-quality text extraction from images
- **LLM Integration**: Leverages OpenAI's GPT-4 for intelligent information extraction
- **Chunking Strategy**: Processes documents page by page with context preservation
- **Flexible Configuration**: Enable/disable OCR or LLM modules independently
- **Exception Logging**: Detailed logging of processing issues and data quality concerns
- **JSON Output**: Structured output with file names, page numbers, and field locations

Installation

1. Install Python Dependencies

```bash
pip install -r requirements.txt
```

2. Set up OpenAI API Key

```bash
# Option 1: Environment variable
export OPENAI_API_KEY="your-api-key-here"

# Option 2: Pass as argument (see usage examples)
```

3. Prepare Your Attributes File

Create a text file listing the information attributes you want to extract, one per line. See `sample_attributes.txt` for examples.

Usage Examples

Basic Usage

```bash
python financial_doc_extractor.py \
    --input-dir /path/to/pdf/files \
    --output-dir /path/to/output \
    --attributes-file attributes.txt
```

OCR Only (No LLM)

```bash
python financial_doc_extractor.py \
    --input-dir /path/to/pdf/files \
    --output-dir /path/to/output \
    --attributes-file attributes.txt \
    --disable-llm
```

LLM Only (No OCR)

```bash
python financial_doc_extractor.py \
    --input-dir /path/to/pdf/files \
    --output-dir /path/to/output \
    --attributes-file attributes.txt \
    --disable-ocr \
    --openai-api-key "your-api-key"
```

Full Processing with Custom API Key

```bash
python financial_doc_extractor.py \
    --input-dir /path/to/pdf/files \
    --output-dir /path/to/output \
    --attributes-file attributes.txt \
    --openai-api-key "your-api-key"
```

Command Line Arguments

Argument	Required	Description
`--input-dir`	Yes	Directory containing PDF files to process
`--output-dir`	Yes	Directory where results will be saved
`--attributes-file`	Yes	Text file with target information attributes
`--openai-api-key`	No*	OpenAI API key (*required if LLM enabled)
`--enable-ocr`	No	Enable OCR processing (default: True)
`--disable-ocr`	No	Disable OCR processing
`--enable-llm`	No	Enable LLM processing (default: True)
`--disable-llm`	No	Disable LLM processing

Output Files

The tool generates several output files with timestamps:

1. Extraction Results (`extraction_results_YYYYMMDD_HHMMSS.json`)

```json
```

```
[
  {
    "file_name": "policy_123.pdf",
    "page_number": 1,
    "field_label": "Policy Number",
    "field_value": "POL-2024-001234",
    "location_description": "Top right corner of page, in header section",
    "confidence": "HIGH",
    "exception_notes": "",
    "processing_notes": ""
  }
]
```


2. Exception Log (`exception_log_YYYYMMDD_HHMMSS.json`)


```
json

[
  {
    "file_name": "policy_456.pdf",
    "page_number": 2,
    "field_label": "Policyholder Age",
    "issue_type": "UNCLEAR_VALUE",
    "description": "Value found but unclear or ambiguous",
    "exception_notes": "Handwritten text partially illegible",
    "timestamp": "2024-01-15T14:30:00"
  }
]
```


3. Processing Log (`processing_log_YYYYMMDD_HHMMSS.log`)

Detailed log of all processing activities, errors, and statistics.

Key Features Explained

Accuracy-First Approach

- Extracts content **exactly as reported** in documents
- No guessing or inference of values
- Clear marking of unclear or missing information
- Detailed exception reporting for compliance review

OCR Integration

- Automatically detects image-only pages
- High-quality text extraction using EasyOCR
- Confidence filtering to avoid low-quality OCR results
- Seamless integration with text-based extraction

LLM Processing

- Uses GPT-4 for intelligent information extraction
- Context-aware processing with XML-tagged prompts
- Structured JSON output with location information
- Confidence scoring for extracted values

Exception Handling

- Comprehensive error logging
- Data quality issue reporting
- Processing statistics and summaries
- Graceful handling of document processing failures

Best Practices

Document Quality

- Ensure PDFs are as high quality as possible
- For handwritten documents, scan at 300+ DPI
- Remove any security restrictions from PDFs

Attributes File

- Use clear, specific attribute names
- One attribute per line
- Match the exact terminology used in your documents
- Keep attributes focused and unambiguous

Processing Strategy

- Process documents in batches for better resource management
- Review exception logs carefully for compliance issues
- Validate extracted data against original documents
- Use both OCR and LLM for maximum coverage

Performance Optimization

- For large document sets, consider processing in smaller batches
- Monitor API usage and costs with OpenAI
- Use SSD storage for better I/O performance
- Ensure adequate RAM for OCR processing

Troubleshooting

Common Issues

OCR Initialization Fails

- Ensure PyTorch is properly installed
- Check CUDA compatibility if using GPU
- Try CPU-only mode if GPU issues persist

OpenAI API Errors

- Verify API key is valid and has sufficient credits
- Check rate limits and adjust processing speed
- Ensure proper internet connectivity

Memory Issues

- Reduce batch size for large documents
- Close unnecessary applications
- Consider processing fewer documents simultaneously

Poor Extraction Quality

- Review and refine attributes file
- Check document image quality
- Adjust OCR confidence thresholds
- Review LLM prompts for specific document types

Support and Customization

This tool is designed to be customizable for specific compliance requirements. Key areas for customization:

- **Prompt Engineering**: Modify LLM prompts for specific document types
- **OCR Settings**: Adjust confidence thresholds and preprocessing
- **Output Format**: Customize JSON structure for your systems
- **Error Handling**: Add specific exception types for your use cases

For production deployment, consider:

- Adding database integration for results storage
- Implementing batch processing queues
- Adding user interface for non-technical users
- Setting up monitoring and alerting systems