Real Estate Document Processor

An intelligent document processing system that automatically splits merged PDF files containing multiple real estate documents and classifies them into specific categories.

Overview

This project addresses the common challenge in real estate workflows where multiple documents (deeds, property cards, tax documents) are merged into single PDF files. The system uses a combination of pattern recognition, OCR technology, and machine learning to automatically:

- **Split** merged PDFs into individual documents
- **Classify** documents into predefined categories
- **Extract** text from various PDF formats including scanned documents
- **Organize** output files with appropriate naming conventions

System Architecture

```
Input: Merged PDF (Multiple Documents)
    ↓
Document Boundary Detection
    ↓
Text Extraction (Multi-tier: PDFPlumber → PyPDF2 → OCR)
    ↓
Hybrid Classification (Keywords + Machine Learning)
    ↓
Output: Categorized Individual PDFs
```

Features

- **Advanced PDF Processing**: Multi-method text extraction with OCR fallback
- **Intelligent Boundary Detection**: Identifies document separators using legal patterns
- **Hybrid Classification**: Combines rule-based keywords with ML models
- **Robust Error Handling**: Graceful failure recovery and comprehensive logging
- **High Accuracy**: Achieved 100% classification accuracy on training data
- **Scalable Architecture**: Modular design for easy maintenance and extension

Technologies Used

- **Python 3.7+**
- **PDFPlumber** - Primary PDF text extraction
- **PyPDF2** - PDF manipulation and fallback extraction
- **Tesseract OCR** - Image-based text recognition
- **Scikit-learn** - Machine learning pipeline (TF-IDF + Multinomial Naive Bayes)
- **NumPy** - Numerical operations
- **Pillow** - Image processing for OCR

Installation

Prerequisites

1. **Python 3.7 or higher**
2. **Tesseract OCR** (for image-based PDFs)

Step 1: Install Tesseract OCR

Ubuntu/Debian:

```bash
sudo apt-get update
sudo apt-get install tesseract-ocr
```

Windows:

- Download from: https://github.com/UB-Mannheim/tesseract/wiki
- Add to system PATH

macOS:

```bash
brew install tesseract
```

Step 2: Install Python Dependencies

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

Or install individually:

bash

```
pip install pdfplumber scikit-learn PyPDF2 numpy pytesseract Pillow
```

Step 3: Clone Repository

bash

```
git clone https://github.com/yourusername/real-estate-document-processor.git
cd real-estate-document-processor
```

Usage

Basic Usage

python

```
from real_estate_processor import ImprovedRealEstateDocumentProcessor

# Initialize processor
processor = ImprovedRealEstateDocumentProcessor()

# Train the model (if you have training data)
processor.train_model("path/to/training_data")

# Process merged PDF
processor.process_merged_pdf(
    pdf_path="path/to/merged_document.pdf",
    output_dir="path/to/output_directory"
)
```

Directory Structure

```
project/
├── training_data/
│   ├── deeds/
│   │   ├── deed1.pdf
│   │   └── deed2.pdf
│   ├── property_cards/
│   │   ├── card1.pdf
│   │   └── card2.pdf
│   └── tax_documents/
│       ├── tax1.pdf
│       └── tax2.pdf
├── merged_documents/
│   └── sample_merged.pdf
└── output/
    ├── deeds_1.pdf
    ├── property-cards_1.pdf
    └── tax-documents_1.pdf
```

Configuration

Update the paths in the script:

```python
# Configure your paths
train_path = "path/to/training_data"
merged_pdf_path = "path/to/merged_document.pdf"
output_dir = "path/to/output_directory"
```

Performance Results

- **Training Accuracy**: 100%
- **Cross-Validation Score**: 1.000
- **Document Categories**: 3 (Deeds, Property Cards, Tax Documents)
- **Test Case**: Successfully processed 70-page PDF into 8 classified documents

Sample Output

```
Cross-validation scores: [1. 1. 1. 1. 1.]
Average CV score: 1.000

Processing Summary:
  tax_documents: 2 documents
  deeds: 4 documents
  property_cards: 2 documents
```

Document Classification

Supported Document Types

1. **Deeds**
   - Keywords: warranty deed, quitclaim deed, grantor, grantee, conveyance
   - Legal patterns: recording stamps, notarization markers
2. **Property Cards**
   - Keywords: parcel id, assessed value, tax assessment, zoning
   - Patterns: property characteristics, lot descriptions
3. **Tax Documents**
   - Keywords: tax bill, amount due, property tax, tax collector
   - Patterns: payment notices, tax liens

Technical Approach

Document Boundary Detection

The system employs multiple strategies to identify document boundaries:

- **Recording Stamp Recognition**: Searches for legal recording stamps using regex patterns
- **Document Separator Detection**: Identifies headers, footers, and transition markers
- **Fallback Strategy**: Treats each page as a separate document when boundaries are unclear

Text Extraction Pipeline

- **Primary Method**: PDFPlumber for high-quality text extraction
- **Secondary Method**: PyPDF2 for compatibility with various PDF formats
- **Tertiary Method**: Tesseract OCR for scanned or image-based documents

Classification Algorithm

- **Keyword-Based Classification**: Uses weighted scoring with comprehensive keyword dictionaries
- **Machine Learning Classification**: TF-IDF vectorization with Multinomial Naive Bayes
- **Confidence-Based Decision**: Combines both approaches with configurable thresholds

Customization

Adding New Document Types

1. **Update Keywords Dictionary**:

```python
self.keywords['new_document_type'] = [
    'keyword1', 'keyword2', 'specific_term'
]
```

2. **Add Training Data**: Create folder in training directory
3. **Retrain Model**: Run training with updated dataset

Adjusting Classification Thresholds

```python
# In classify_document method
keyword_threshold = 0.3  # Adjust as needed
ml_threshold = 0.4       # Adjust as needed
```

File Structure

```
real-estate-document-processor/
├── README.md
├── real_estate_processor.py
├── requirements.txt
├── docs/
│   └── technical_approach.md
├── examples/
│   ├── sample_input.pdf
│   └── sample_output/
├── training_data/
│   ├── deeds/
│   ├── property_cards/
│   └── tax_documents/
└── .gitignore
```

Requirements

```
pdfplumber>=0.9.0
scikit-learn>=1.3.0
PyPDF2>=3.0.1
numpy>=1.24.0
pytesseract>=0.3.10
Pillow>=9.5.0
```

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/improvement`)
3. Commit changes (`git commit -am 'Add new feature'`)
4. Push to branch (`git push origin feature/improvement`)
5. Create Pull Request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Documentation

- Technical Approach Document
- API Documentation

- Training Data Guidelines

Support

For issues and questions:

- Create an issue on GitHub
- Contact: [Your Email Address]

Future Enhancements

- Support for additional document types
- Web-based interface
- Batch processing capabilities
- Integration with cloud storage services
- Advanced OCR preprocessing
- Multi-language support

Acknowledgments

Built for automated real estate document processing workflows.