Automated Label-Value Extraction from Multilingual Documents: A Deep Learning Approach Using Optical Character Recognition

elm19-others

August 10, 2025

Abstract

This report presents the first phase of developing an intelligent system for automated extraction of label-value pairs from multilingual documents including forms, invoices, and various administrative documents. Our approach leverages advanced Optical Character Recognition (OCR) technologies to process documents in French, Arabic, and English languages. The system aims to extract textual content with spatial positioning information and subsequently store structured data in a database. This phase focuses on comparative analysis of OCR engines, specifically Tesseract and PaddleOCR, evaluating their performance across different languages and document complexities.

1 Introduction

The digitization of paper-based documents and the extraction of structured information from unstructured formats represents a critical challenge in modern information management systems. Traditional manual data entry processes are time-consuming, error-prone, and economically inefficient. This project addresses the need for an automated solution capable of processing multilingual documents containing various forms, invoices, and administrative papers.

Our objective is to develop an AI-powered system that can:

- Extract textual content from images and PDF documents
- Handle multilingual content (French, Arabic, English)
- Identify spatial relationships between labels and their corresponding values
- Store structured data efficiently in a database

This report documents Phase 1 of the project, focusing on text extraction methodologies and OCR engine evaluation.

2 Methodology

2.1 OCR Engine Selection and Evaluation

Given the multilingual requirements of our project, we evaluated two prominent OCR solutions: Tesseract OCR and PaddleOCR. Both engines provide comprehensive language support and are well-documented with active community support.

2.1.1 Tesseract OCR

Tesseract, developed by Google, is an open-source OCR engine that supports over 100 languages. It utilizes Long Short-Term Memory (LSTM) neural networks for character recognition.

Listing 1: Tesseract Implementation Example

```python
import pytesseract
from PIL import Image
import cv2

def extract_text_tesseract(image_path, language='fra+ara+eng'):
    """
    Extract text using Tesseract OCR
    """
    # Load and preprocess image
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply OCR
    extracted_text = pytesseract.image_to_string(
        gray,
        lang=language,
        config='--psm␣6'
    )

    return extracted_text

# Example usage
text = extract_text_tesseract('sample_document.jpg')
print(text)
```

2.1.2 PaddleOCR

PaddleOCR is a practical ultra-lightweight OCR system developed by PaddlePaddle. It supports detection and recognition of natural scenes and document texts in multiple languages with high accuracy.

Listing 2: PaddleOCR Implementation Example

```python
from paddleocr import PaddleOCR
import cv2
import numpy as np

def extract_text_paddle(image_path, language='multilingual'):
    """
    Extract text using PaddleOCR with bounding box information
    """
    # Initialize OCR
    ocr = PaddleOCR(use_angle_cls=True, lang=language)

    # Perform OCR
    result = ocr.ocr(image_path, cls=True)

    extracted_data = []
    for line in result:
        for word_info in line:
            bbox = word_info[0]  # Bounding box coordinates
            text = word_info[1][0]  # Extracted text
            confidence = word_info[1][1]  # Confidence score

            extracted_data.append({
                'text': text,
```

```
                'bbox': bbox,
                'confidence': confidence
            })

    return extracted_data

# Example usage
data = extract_text_paddle('sample_document.jpg')
for item in data:
    print(f"Text: {item['text']}, Confidence: {item['confidence']:.2f}")
```

2.2 Comparative Analysis Results

An analysis of online reviews and technical specifications for both OCR engines reveals some notable distinctions:

Table 1: OCR Engine Comparison

Criteria	Tesseract	PaddleOCR
Printed Text Accuracy	High	Very High
Handwritten Text	Limited	Better
Complex Layouts	Moderate	Excellent
Multilingual Support	Good	Excellent
Bounding Box Info	Limited	Native
Processing Speed	Fast	Moderate

3 Implementation and Results

3.1 PaddleOCR: The Chosen Solution

Based on our comparative analysis, PaddleOCR emerged as the superior choice for our requirements. The decision was primarily driven by:

1. **Superior handling of complex layouts**: Documents such as invoices and forms often contain tables, multiple columns, and varied text orientations.
2. **Better handwritten text recognition**: Many forms contain handwritten elements that Tesseract struggles to process accurately.
3. **Native bounding box extraction**: Critical for our subsequent label-value pairing algorithms.
4. **Robust multilingual capabilities**: Seamless processing of mixed-language documents.

3.2 Bounding Box Extraction and Spatial Analysis

PaddleOCR's architecture follows a two-stage approach:

1. **Text Detection**: Identifies text regions in the image using a detection model
2. **Text Recognition**: Recognizes characters within detected regions using a recognition model

The bounding box coordinates provided by PaddleOCR are essential for our project as they enable spatial relationship analysis between labels and values without requiring additional computer vision models.

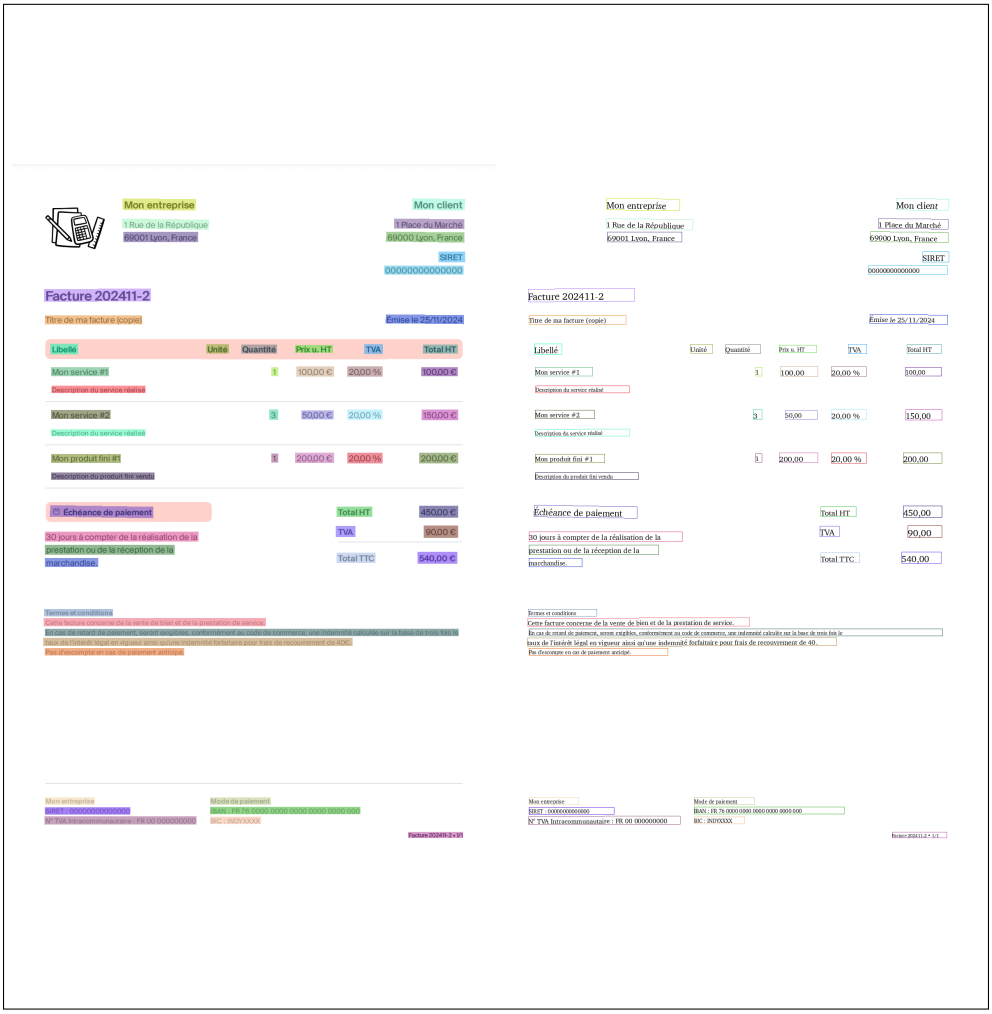

Figure 1: Sample Document with PaddleOCR Bounding Box Detection(more clear version in our code

3.3 Language-Specific Processing

3.3.1 French and English Processing

For Latin-script languages (French and English), PaddleOCR performs exceptionally well with minimal preprocessing required.

3.3.2 Arabic Text Processing

Arabic text processing presented unique challenges due to its right-to-left (RTL) reading direction. While PaddleOCR correctly identifies Arabic characters, the default output maintains left-to-right ordering. We implemented a post-processing function to handle this:

Listing 3: Arabic Text Direction Correction

```
import arabic_reshaper
from bidi.algorithm import get_display

def process_arabic_text(text):
```

```
    """Process Arabic text for proper display (right-to-left)"""
    try:
      parts = text.split()

      fixed_parts = []
      for part in parts:
          # Reverse Arabic-like parts only (heuristic: check if contains Arabic
              letters)
          if any('\u0600' <= c <= '\u06FF' for c in part):
              part = part[::-1]  # Reverse characters in the part
          fixed_parts.append(part)

      fixed_text = "␣".join(fixed_parts)
      # Reshape Arabic text for proper display
      reshaped_text = arabic_reshaper.reshape(fixed_text)
      # Apply bidirectional algorithm
      display_text = get_display(reshaped_text)
      return display_text
    except:
        # If reshaping fails, return original text
        print("failed")
        return text
```

4 Current Challenges and Limitations

4.1 Multi-language Document Processing

Currently, there is no single OCR configuration that optimally handles all three target languages simultaneously. For documents containing mixed languages or when the primary language is unknown, we propose implementing a confidence-based selection mechanism:

1. Execute OCR with each language configuration (French, Arabic, English)
2. Compare confidence scores for extracted text
3. Select results from the configuration with highest overall confidence
4. Merge results where appropriate

4.2 Domain-Specific Text Recognition

The nature of our target documents may include:

- Domain-specific terminology
- Handwritten annotations
- Low-quality scanned images
- Irregular formatting

These factors may necessitate fine-tuning PaddleOCR models on domain-specific datasets. However, this approach presents challenges:

1. **Data Collection**: Acquiring sufficient annotated training data for specialized domains
2. **Model Complexity**: PaddleOCR's architecture requires careful consideration for fine-tuning
3. **Arabic Language Optimization**: Limited resources for Arabic OCR fine-tuning compared to Latin scripts

5 Future Work

Phase 2 of the project will focus on:

1. **Label-Value Pair Extraction**: Developing algorithms to identify relationships between text elements using spatial positioning
2. **Multi-language Confidence Integration**: Implementing the proposed confidence-based language selection system
3. **Database Integration**: Designing and implementing structured data storage solutions
4. **Model Fine-tuning**: If required, fine-tuning OCR models on domain-specific datasets
5. **Performance Optimization**: Enhancing processing speed and accuracy for production deployment

6 Conclusion

Phase 1 of our project successfully established a robust foundation for multilingual document text extraction. PaddleOCR emerged as the optimal solution, providing superior accuracy for complex document layouts while offering essential spatial information through bounding box coordinates. The system currently handles French, Arabic, and English texts with appropriate post-processing for RTL languages.

The identification of current limitations, particularly in multi-language document processing and domain-specific optimization, provides clear direction for subsequent development phases. The spatial information captured during text extraction positions the project well for the next phase of automated label-value pair identification.

Acknowledgments

We acknowledge the open-source communities behind Tesseract OCR and PaddleOCR for their invaluable contributions to optical character recognition technology.