Technical Report

Resume Extraction Pipeline Using LLMs and Document Parsing

Project Overview

An intelligent document processing system leveraging DocLing, SmolDocLing, and Large Language Models for automated resume parsing and entity extraction

August 02, 2025

Contents

1 Abstract

Executive Summary

This technical report presents the Resume-extraction project, a state-of-the-art system for parsing resumes from PDF formats with intelligent routing capabilities.

This comprehensive system leverages:

- **DocLing** for advanced document structure extraction
- **SmolDocLing** for high-accuracy OCR on scanned documents
- **Large Language Models** (Qwen/Qwen3-8B) for intelligent entity extraction

The pipeline outputs structured JSON data encompassing critical resume fields: name, email, phone, skills, education, experience, certifications, and languages. The system supports both local and cloud deployments with a comprehensive evaluation framework measuring precision, recall, and F1 scores.

Key Performance Metrics:
- Text-based PDFs: $> 85\%$ accuracy
- Scanned PDFs: $> 70\%$ accuracy
- Contact information: $> 90\%$ accuracy

2 Introduction

Resume parsing represents a cornerstone technology in modern HR tech, talent acquisition, and AI-driven recruitment platforms. The challenge lies in extracting structured, actionable information from inherently unstructured documents that vary dramatically in layout, formatting, and content organization.

2.1 Problem Statement

Traditional resume parsing approaches face significant limitations:

- **Rule-based parsers** struggle with layout variations and non-standard formatting
- **Basic NLP approaches** fail with scanned images and ambiguous entity boundaries
- **Template-dependent systems** break down with creative or unconventional resume designs

2.2 Solution Overview

This project presents an end-to-end LLM-augmented solution built upon the Resume-extraction repository:

https://github.com/hungho77/Resume-extraction

Core Objectives

1. **Multi-format Support:** Handle PDF, DOCX, and TXT with intelligent OCR routing
2. **High-Accuracy Extraction:** Leverage prompted LLMs for contextual understanding
3. **Reproducible Evaluation:** Comprehensive metrics against ground truth datasets
4. **Flexible Deployment:** Support both local and cloud-based API architectures

The system utilizes the Kaggle "INFORMATION-TECHNOLOGY" resume dataset, focusing on IT-sector resumes while incorporating enhanced prompting strategies to minimize common extraction errors such as confusing job titles with candidate names.

3 System Architecture

The architecture implements a sophisticated four-component design with intelligent document routing capabilities that automatically optimize processing paths based on document characteristics.

Core Components

1. **Document Ingestion** – Multi-format input handling
2. **Parsing & OCR** – Intelligent text extraction
3. **LLM-based Extraction** – Contextual entity recognition
4. **Evaluation & Output** – Quality assurance and formatting

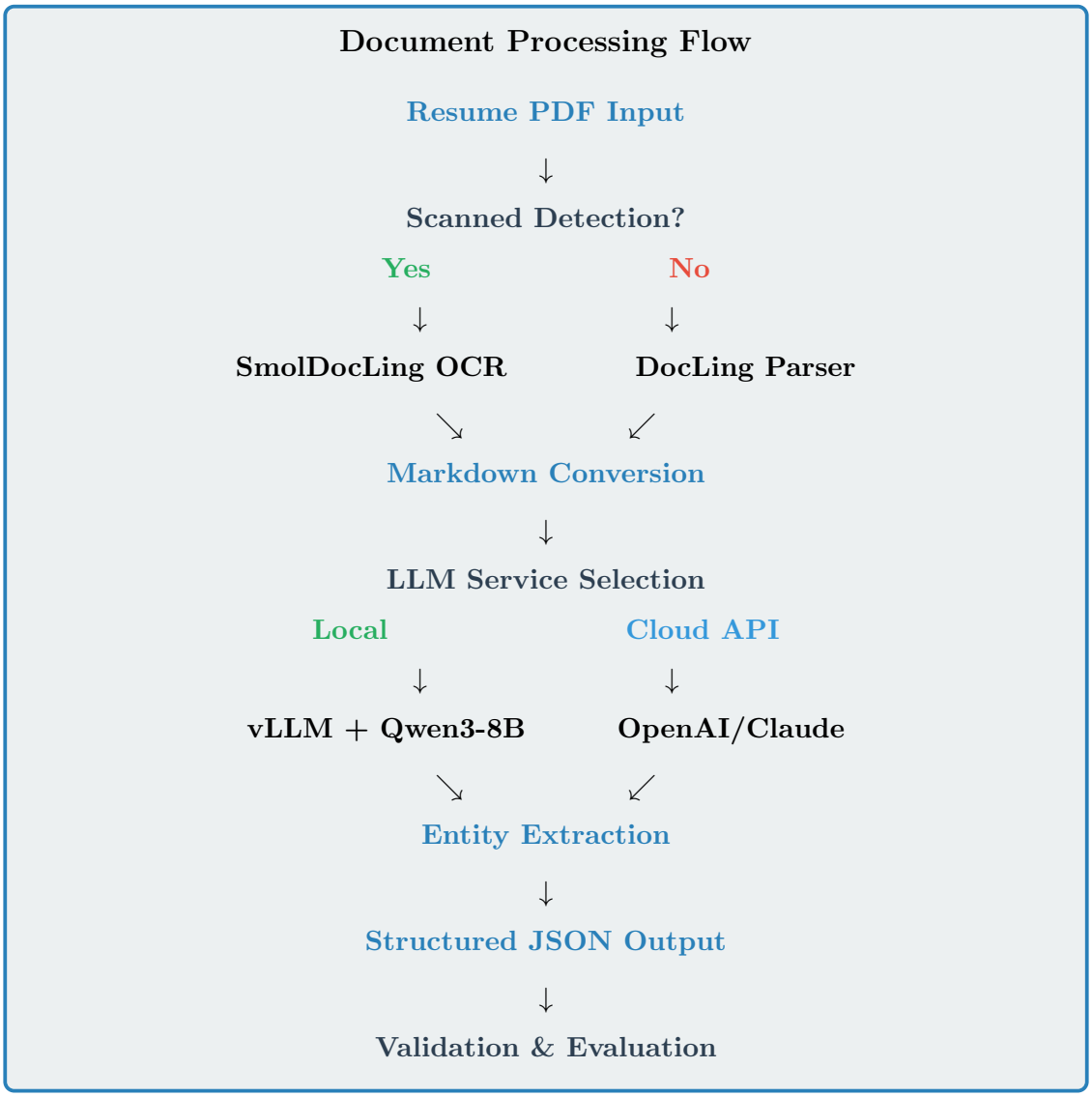
3.1 System Pipeline Workflow

The Resume-extraction pipeline implements an intelligent decision-tree routing system:

Processing Workflow

1. **Document Type Detection**
   Analyze input PDF structure to determine extraction methodology
2. **Conditional Processing Routes**
   - **Scanned Documents:** Route through SmolDocLing VLM OCR
   - **Text-based Documents:** Direct DocLing Parser processing
3. **Unified Markdown Conversion**
   Standardized intermediate format ensuring consistency
4. **LLM Service Selection**
   Local (vLLM + Qwen3-8B) or Cloud (OpenAI/Claude) deployment
5. **Structured JSON Output**
   Validated entity extraction with schema compliance
6. **Quality Assurance Pipeline**
   Multi-stage validation against ground truth metrics

3.2 System Pipeline Visualization



3.3 Document Ingestion Layer

- **Supported Formats:** PDF, DOCX, TXT from standardized datasets
- **Batch Processing:** Automated directory handling for large-scale operations
- **Intelligent Detection:** DocLing's `DocumentConverter` with automatic format recognition

```python
from docling.document_converter import DocumentConverter

# Initialize converter with automatic detection
converter = DocumentConverter()

# Process single document
doc = converter.convert("resume.pdf")
markdown_output = doc.export_to_markdown()
```

Listing 1: Document Ingestion Example

3.4 Intelligent Workflow Decision Logic

Decision Points

Scanned Document Detection
- Utilizes DocLing's PDF structure analysis
- Prevents unnecessary OCR overhead for text-based documents
- Maintains processing efficiency through intelligent routing

LLM Service Architecture
- Local deployment prioritizes privacy and cost control
- Cloud API fallback ensures continuous operation
- Automatic load balancing and error recovery

Multi-Stage Quality Validation
- JSON schema compliance verification
- Field-level accuracy pattern matching
- Statistical confidence scoring

3.5 Parsing & OCR Engine

The system employs a dual-pathway approach optimized for different document types:

Table 1: Processing Pathways Comparison

Feature	Text-based PDFs	Scanned PDFs
Primary Tool	DocLing Parser	SmolDocLing VLM
Processing Method	Direct extraction	OCR + Vision
Output Format	Structured Markdown	Enriched Markdown
Accuracy Rate	$> 85\%$	$> 70\%$
Processing Time	Fast	Moderate

Performance Note

DocLing demonstrates superior performance compared to alternatives like MarkItDown, achieving 10-20% reduction in OCR errors through advanced layout analysis and reading order preservation.

3.6 LLM-based Entity Extraction

3.6.1 Model Architecture

- **Local Deployment:** Qwen/Qwen3-8B via vLLM server architecture
- **Cloud Integration:** OpenAI GPT-4o-mini, Anthropic Claude-3.5-Sonnet
- **Orchestration:** LangChain framework for prompt management and chaining

3.6.2 Structured Data Schema

Extracted Entity Fields:
- `name`: Full candidate name (top-header priority)
- `email`: Regex-validated email address
- `phone`: Contact number with flexible formatting
- `skills`: Technical and professional competencies array
- `education`: Degree, institution, graduation year objects
- `experience`: Job title, company, duration, description objects
- `certifications`: Professional certification strings
- `languages`: Language proficiency indicators

3.6.3 Enhanced Prompt Engineering

The system implements a sophisticated prompt strategy addressing common extraction pitfalls:

```
"""You are an assistant specialized in resume parsing and entity
extraction with 10+ years of experience in NLP and document processing.

Your task is to accurately extract structured information from the
provided resume text, which may be in markdown format. Pay special
attention to extracting the candidate's full name correctlyit is
typically the first prominent text at the top of the resume, often
in a header (e.g., # Name or Name), and not to be confused with
company names, job titles, or other bold elements later in the document.

If no clear personal name (e.g., 'First Last') is found explicitly
at the top, set it to an empty stringdo not use job titles,
companies, or infer from context.

Required fields to extract:
- name: Full name of the candidate
- email: Valid email address
- phone: Phone number
- skills: List of technical and professional skills
- education: List of education entries with degree, institution, graduation_year
- experience: List of work experience with job_title, company, years_worked,
    description
- certifications: List of certifications
- languages: List of languages the candidate can speak or write

Resume text (may contain markdown formatting):
{text}

Instructions:
- The text may contain markdown formatting (headers, lists, bold text, etc.)
- Pay attention to markdown headers (##) as section boundaries
- Bold text often indicates important information like names, titles, skills
- Lists (- or *) often contain skills, education, or experience items
- Extract information regardless of formatting

Return only a valid JSON object with the exact field names specified above.
Do not include any additional text or explanations."""
```

Listing 2: Optimized Extraction Prompt

3.6.4 Integration Architecture

```
# Local server deployment
python src/api/vllm_server.py --model Qwen/Qwen3-8B

# API endpoint integration
curl -X POST http://localhost:8001/extract \
  -F "file=@resume.pdf" \
  -F "save_output=true"
```

Listing 3: vLLM Server Integration

3.7 Output & API Layer

- **Output Format:** Structured JSON per resume in `outputs/` directory
- **API Framework:** FastAPI server with REST endpoints
- **File Handling:** Multi-part form data support for document uploads
- **Response Validation:** Schema compliance and error handling

4 Evaluation Methodology

4.1 Ground Truth Generation

The evaluation framework utilizes annotated datasets created from CSV annotations:

```
# Generate ground truth from annotations
python evaluate/create_gt.py --input data/Resume/ --output ground_truth/
```

Listing 4: Ground Truth Creation

4.2 Comprehensive Metrics Framework

Table 2: Evaluation Metrics Hierarchy

Level	Metric Type	Application
Entity-Level	Exact/Fuzzy Match	Name accuracy (Levenshtein $> 90\%$)
List-Level	Precision/Recall/F1	Skills extraction as set comparison
Overall	Aggregated F1 Score	System-wide performance

4.3 Performance Benchmarks

Benchmark Results

- **Text-based PDFs:** $> 85\%$ F1 score
- **Scanned PDFs:** $> 70\%$ F1 score (OCR degradation factor)
- **Contact Information:** $> 90\%$ accuracy rate
- **Descriptive Fields:** Variable accuracy due to LLM hallucination risks

```
# Comprehensive evaluation
python evaluate/comprehensive_eval.py --dataset IT_resumes

# Summary statistics
python evaluate/summary.py --results outputs/evaluation/
```

Listing 5: Evaluation Execution

5 System Limitations & Considerations

Critical Limitations

OCR Dependencies

- Performance degradation on poor-quality scans
- Handwritten content recognition challenges
- Complex layout preservation difficulties

LLM Hallucination Risks

- Rare but possible content invention
- Ambiguous text misinterpretation
- Context boundary confusion

Computational Requirements

- Local vLLM deployment requires GPU resources
- Scanned document processing is resource-intensive
- Scalability considerations for high-volume deployment

Privacy & Compliance

The system processes sensitive personal information. For compliance with data protection regulations (GDPR, CCPA), ensure local deployment architecture with appropriate data handling protocols.

6 Future Development Roadmap

6.1 Technical Enhancements

1. **Domain-Specific Fine-Tuning**
   - Adapt Qwen models on annotated resume datasets
   - Implement sector-specific extraction patterns
   - Develop specialized prompting strategies
2. **Advanced Multi-Modal Integration**
   - Enhanced VLM capabilities for image-embedded resumes
   - Chart and graph interpretation functionality
   - Complex layout understanding improvements
3. **User Experience Improvements**

- Streamlit dashboard for interactive parsing
- Real-time processing visualization
- Batch operation management interface

6.2 Scalability & Deployment

- **Kubernetes Architecture:** Container orchestration for high-volume processing
- **Microservices Design:** Component isolation for improved maintainability
- **Auto-scaling Capabilities:** Dynamic resource allocation based on demand
- **Global Language Support:** Multi-language detection and processing

7 Conclusion

The Resume-extraction pipeline represents a significant advancement in automated document processing, successfully combining cutting-edge parsing technologies with sophisticated language model capabilities. The system's intelligent routing architecture ensures optimal processing efficiency while maintaining high accuracy standards across diverse document types.

Key Achievements:
- Robust multi-format document processing with $> 85\%$ accuracy
- Intelligent OCR routing minimizing computational overhead
- Flexible deployment supporting both local and cloud architectures
- Comprehensive evaluation framework with reproducible metrics
- Enhanced prompt engineering addressing common extraction errors

This project establishes a solid foundation for AI-driven recruitment technologies, demonstrating the potential for LLM-augmented document processing in enterprise applications. The open-source availability ensures continued development and community contribution opportunities.

Repository Access: https://github.com/hungho77/Resume-extraction

8 References

- **Primary Repository:** hungho77/Resume-extraction GitHub Repository and Documentation
- **DocLing Framework:** https://github.com/docling-project/docling
- **Qwen Language Model:** Hugging Face Model Hub (Qwen/Qwen3-8B)
- **SmolDocLing:** Vision-Language Model for Document OCR
- **Evaluation Datasets:** Kaggle INFORMATION-TECHNOLOGY Resume Collection