Project Report: Lab Report Extraction System

Abstract

This project implements a Lab Report Extraction System that converts unstructured lab reports (PDFs or images) into structured JSON. The system uses a hybrid approach combining OCR, rule-based extraction, and machine learning (spaCy NER) to identify patient details and test results. The pipeline supports human-in-the-loop review and continuous learning: user-reviewed extractions are appended to a training dataset to retrain and improve the NER model over time. A minimal demo interface and API are provided via Gradio in a Jupyter notebook, and a standalone Flask API was included in the project for deployment.

Key features:

- Robust OCR and text extraction for PDFs and images using pdfplumber and pytesseract.
- Rule-based regex extraction for common report fields (patient name, tests, values, units) with confidence scoring.
- NLP-based entity recognition with spaCy to generalize across formats.
- Human-in-the-loop review and continuous learning mechanism (saves reviewed extractions to `reviewed_extractions.jsonl`).
- Demo UI and API implemented in the notebook using Gradio; production-style REST API included as `lab_report_api.py`.

Methodology

Problem statement

- Medical lab reports often come in different formats and must be converted to structured data for analytics, EMR ingestion, and downstream workflows. The challenge is format variability, noisy OCR outputs, and domain-specific entity types (test names, units).

Approach Overview

1. Data acquisition and preprocessing
    - The system reads report file URLs from `Medical report.xlsx` and downloads them into a local directory.
    - For PDFs, `pdfplumber` extracts text page-by-page. For images, `pytesseract` performs OCR after opening the file with Pillow.
    - Basic cleaning removes excessive whitespace and normalizes common delimiters to improve regex and NER performance.
2. Rule-based extraction
    - We apply regular expressions to capture common patterns (e.g., `Patient Name:`, `<TestName>: <value> <unit>`).
    - Rule-based extraction is deterministic and provides high-confidence matches when the text follows expected patterns.
    - Confidence scores are attached (e.g., 1.0 for exact regex matches, 0.5 for missing fields).
3. Machine learning-based extraction
    - A spaCy pipeline (`en_core_web_sm`) performs named entity recognition for PERSON, ORG, DATE, and other entities.
    - The NER output complements regex extraction; entities detected by spaCy are merged with rule-based outputs to fill missing or ambiguous fields.
    - The notebook includes a mechanism to save reviewed extractions into `reviewed_extractions.jsonl` for future retraining.
4. Human-in-the-loop and continuous learning
    - After automated extraction, results are displayed for user review. The user can confirm or correct the values in the notebook or the Gradio UI.
    - Confirmed/corrected records are appended to a newline-delimited JSON file used to build training sets for fine-tuning spaCy.
5. API and Demo
    - A demo Gradio interface is embedded in the notebook for interactive testing and an API-like POST endpoint for automation.
    - A simple Flask app is included for running as a standalone REST API in production.

Evaluation and Limitations

- Rule-based extraction works well for consistent report formats but fails when layouts change or OCR introduces noise.
- SpaCy's base model offers general NER capability but requires domain-specific training to reliably extract lab-test entities.
- OCR quality is crucial; for low-quality scans advanced pre-processing (deskewing, denoising) or better OCR engines are recommended.

Future Work

- Train a custom spaCy NER model on `reviewed_extractions.jsonl` converted to spaCy format.
- Add layout-aware extraction (PDF table parsing, spatial analysis) using tools like Camelot, tabula-py, or LayoutLM-based models.
- Build a lightweight web dashboard for batch processing and manual correction with user authentication.