# Informe técnico: Extracción de caracteres de comprobantes

**Autor:** Frans Edward Paxi Juchani **Fecha:** 2025-10-21 **Github:** [FransPaxiJ](https://github.com/FransPaxiJ/machine-learning/tree/master/unidad_2/examen_final)

## 1. Resumen

Este informe describe un sistema desarrollado para extraer información clave (cliente, DNI, RUC, razón social, tipo de comprobante, fecha de emisión e importe total) a partir de comprobantes (PDF/imagen) usando PaddleOCR para el reconocimiento de texto y web scraping para validar datos (DNI en eldni.com y RUC en la SUNAT). El proyecto prioriza precisión en la extracción mediante reglas basadas en expresiones regulares, estrategias de búsqueda contextuales y validación externa.

## 2. Objetivos

* Extraer automáticamente los campos relevantes de un comprobante fiscal.
* Priorizar candidatos (p. ej. varios DNIs detectados) y validar con fuentes externas.
* Procesar el documento realizando OCR una sola vez y reutilizar el resultado para todas las extracciones.
* Entregar un resumen final estructurado (importe, cliente, empresa, RUC, tipo, fecha).

## 3. Tecnologías y librerías principales

* **PaddleOCR**: motor OCR principal. Permite enviar PDFs directamente y ofrece buena precisión en documentos impresos.
* **Selenium + webdriver\_manager**: para realizar consultas web (eldni.com y e-consultaruc.sunat.gob.pe) y validar DNI/RUC.
* **Python (dataclasses, re, datetime, concurrent.futures)**: estructura y orquestación del procesamiento.

## 4. Arquitectura del sistema (visión general)

1. **OCR (única pasada)**: OCRProcessor ejecuta PaddleOCR sobre el archivo (PDF/imagen) y devuelve arrays por página con rec\_texts, rec\_scores y page\_index.
2. **Extracción de candidatos**: extractores independientes recorren el resultado OCR para generar candidatos para cada entidad:
   * DNIExtractor → candidatos DNI (prioriza misma línea que la palabra “DNI”).
   * RUCExtractor → candidatos RUC (admite variantes como “R.U.C.”).
   * TipoComprobanteExtractor → detecta tipo (factura, boleta, nota, ticket…).
   * FechaExtractor → extrae fechas y las normaliza a YYYY-MM-DD.
   * NumberAnalyzer + DecimalExtractor → extraen números decimales (importe total) con varios patrones de formato.
3. **Validación externa (paralela)**: validadores usan web scraping para confirmar:
   * DNIValidator consulta eldni.com para obtener nombre completo.
   * RUCValidator consulta la web de SUNAT para obtener razón social. Ambos se ejecutan en paralelo (ThreadPoolExecutor) para mejorar rendimiento.
4. **Respuesta**: se arma ComprobanteResponse con los campos extraídos/validados.

## 5. Descripción de componentes clave

### 5.1 Clases de datos

Se usan dataclass para modelar resultados: ComprobanteResponse, NumeroDetectado, DNIInfo, PersonaInfo, RUCInfo, EmpresaInfo, TipoComprobanteInfo, FechaEmisionInfo.

### 5.2 OCRConfig / OCRProcessor

OCRConfig centraliza opciones (idioma, unwarping, límites). OCRProcessor inicializa PaddleOCR con esas opciones y ejecuta predict(input=...).

### 5.3 Extracción y análisis numérico

DecimalExtractor implementa varios patrones para capturar formatos comunes (1,234.56 / 1.234,56 / 150.00 / enteros con separador). NumberAnalyzer recorre los textos OCR y genera NumeroDetectado con valor numérico y confianza.

### 5.4 Extracción de DNI y RUC

* DNIExtractor utiliza 3 estrategias: buscar DNI en la misma línea que la etiqueta, buscar en líneas siguientes tras encontrar la etiqueta DNI, y (comentado) buscar todos los números de 8 dígitos. Incorpora ventanas de búsqueda y prioriza detecciones con mejor confianza.
* RUCExtractor similar, detecta RUC y R.U.C. y busca números de 11 dígitos en la misma línea o en líneas siguientes.

### 5.5 Tipo de comprobante y fecha

* TipoComprobanteExtractor usa un diccionario de patrones regex robustos (mayúsculas, acentos) para clasificar tipos: factura electrónica, boleta, nota de crédito/débito, recibo por honorarios, ticket.
* FechaExtractor contiene varios patrones (ISO, DD/MM/YYYY, DD-MM-YYYY, DD.MM.YYYY), estrategias de búsqueda por keyword (“FECHA”, “F. EMISIÓN”), y normaliza a YYYY-MM-DD.

### 5.6 WebScraperService y validadores

* WebScraperService (eldni.com) y WebScraperServiceRUC (SUNAT) usan Selenium con opciones de Chrome headless. WebScraperServiceRUC incluye ajustes anti-detección (user-agent, excludeSwitches) y reintentos.
* DNIValidator y RUCValidator prueban los candidatos en orden y retornan la primera coincidencia válida.

### 5.7 Orquestador optimizado

ComprobanteOrchestrator.process\_comprobante(file\_path) ejecuta OCR una sola vez, extrae candidatos y valida DNI/RUC en paralelo. Finalmente arma y devuelve ComprobanteResponse.

## 6. Flujo de procesamiento (paso a paso)

1. Cargar archivo (PDF/imagen).
2. Ejecutar PaddleOCR: obtener ocr\_result (por página: textos, scores, índice de página).
3. Extraer candidatos (DNI, RUC, fechas, tipo, números) a partir del ocr\_result.
4. Ejecutar validación externa en paralelo para DNI y RUC.
5. Seleccionar el importe total como el número float más alto detectado.
6. Construir ComprobanteResponse y devolver.

## 7. Ejemplo de salida (estructura)

* Importe Total: 1234.56
* Cliente (nombre): Juan Pérez
* DNI: 12345678
* Empresa (razón social): CLINICA SAN PABLO S.A.
* RUC: 20123456789
* Tipo: FACTURA ELECTRÓNICA
* Fecha: 2025-07-16

(Esta estructura corresponde al ComprobanteResponse definido en el código).

## 10. Validaciones, métricas y criterios

* **Confianza OCR**: cada texto detectado tiene una puntuación (rec\_scores). Las estrategias priorizan candidatos con mayor confianza.
* **Reglas heurísticas**: prioridad a detecciones en la misma línea que la etiqueta (p. ej. “DNI 78887021”), luego buscar en líneas siguientes hasta una ventana.
* **Validación externa**: se considera válido un DNI/RUC solo si la consulta web devuelve datos (nombre o razón social).

## 11. Limitaciones y riesgos

* **Web scraping**: páginas externas pueden cambiar su estructura, bloquear solicitudes o limitar accesos. Recomendable reemplazar por APIs oficiales si existen. Además, puede haber limitaciones legales y de privacidad al consultar DNI en masa.
* **Errores OCR**: documentos con baja calidad, ruido, tipografías raras o rotación pueden producir falsos positivos/negativos.
* **Fechas ambiguas**: formatos regionales y errores tipográficos pueden generar parseos incorrectos; el extractor intenta mitigar probando múltiples formatos.

## 12. Recomendaciones y mejoras futuras

* Reemplazar scraping por **APIs oficiales** o servicios autorizados para validar identidad y RUC.
* Agregar **caché** para resultados de validación (evitar consultas repetidas al mismo DNI/RUC).
* Introducir **umbral mínimo de confianza** para considerar candidatos sin validación externa.
* Añadir pruebas unitarias y tests de integración que simulen ocr\_result para cada extractor.
* Registrar métricas (TP, FP) usando un dataset anotado para evaluar precisión por campo.
* Considerar un paso de post-corrección con modelos de lenguaje pequeño para normalizar nombres y corregir OCR (de ser necesario).

## 13. Guía rápida de ejecución

1. Instalar dependencias: pip install paddleocr selenium webdriver-manager + driver Chrome.
2. Ejecutar el script principal (ejemplo):

python tu\_script.py

1. Verificar salida por consola: ComprobanteResponse impreso con los campos.

## 14. Conclusión

El sistema combina reglas determinísticas (regex, ventanas de búsqueda) con validación externa para ofrecer una extracción robusta de comprobantes. La decisión de ejecutar OCR una sola vez y reusar el resultado reduce significativamente el tiempo de proceso. Para producción, es importante mitigar riesgos del scraping y añadir métricas y pruebas que garanticen calidad.