

**LO ESENCIAL** · Módulo 7 · Documentos y OCR

El ruteo por niveles y por qué convertir a texto plano destruye información. Medición campo por campo.

## 7.1 · El ruteo por niveles

### Objetivos

- Clasificar un documento en PDF con texto, OCR clásico, parser o VLM
- Justificar el costo de cada nivel

### No todos los documentos se leen igual

Una factura electrónica en PDF, una remisión escaneada torcida, un contrato con sellos y firmas, y una foto de celular de una cotización son cuatro problemas distintos. Tratarlos igual —«pasarlos todo por el modelo grande»— es caro y falla. El **ruteo por niveles** manda cada documento al método más barato que lo resuelve.

| Nivel | Método                                                | Costo     | Cuándo                                                   | Qué falla                                 |
|-------|-------------------------------------------------------|-----------|----------------------------------------------------------|-------------------------------------------|
| 0     | PDF con capa de texto → extracción directa            | ínfimo    | facturas electrónicas, documentos generados por software | nada, si la capa es fiel                  |
| 1     | OCR clásico (Tesseract o similar)                     | bajo, CPU | escaneos limpios, texto corrido                          | tablas, columnas, sellos, manuscrito      |
| 2     | Modelo especializado de <i>parsing</i> (layout + OCR) | medio     | formularios, tablas, facturas escaneadas                 | manuscrito, fotos malas                   |
| 3     | VLM generalista                                       | alto      | fotos de celular, layouts raros, manuscrito legible      | cifras exactas (inventas), páginas densas |

[VOLÁTIL – verificado: 2026-09] en herramientas; el criterio de escalar solo cuando el nivel inferior falla es estable.

### El router

```
# router.py – decide el nivel por evidencia barata, no por el nombre del archivo
import fitz, subprocess, json, sys
def nivel(ruta):
    doc = fitz.open(ruta)
```

```

texto = "".join(p.get_text() for p in doc)
if len(texto.strip()) > 100 * len(doc):          # ~100 chars/página de capa real
    return 0, "capa de texto"
pix = doc[0].get_pixmap(dpi=72)
if pix.width < 800:                             # foto de celular / baja resolución
    return 3, "baja resolución → VLM"
# OCR rápido de prueba en la primera página
ocr = subprocess.run(["tesseract", "-l", "spa", "-", "--psm", "6"], input=pix.tobytes("png"), capture_output=True).stdout.decode()
if len(ocr.split()) > 80 and "|" not in ocr and ocr.count("\n\n") < 5:
    return 1, "escaneo limpio → OCR"
return 2, "layout complejo → parser"
for r in sys.argv[1:]: print(r, *nivel(r))

```

Corra sobre 30 documentos mezclados del cliente. Lo que debe ver: una mayoría en 0 y 1 (baratos), una minoría en 2, y muy pocos en 3. Si la mayoría cae en 3, el problema es cómo el cliente digitaliza, y eso se arregla con un escáner, no con un modelo.

#### PROCEDIMIENTO · Laboratorio 7.1

**Entregable:** `router.py` y `labs/07-01/ruteo.csv` con las 30 decisiones y, para 10 de ellas, la verificación manual de que el nivel fue el correcto. Commit: «Capa 7.1: router».

#### Qué se degrada en cada perfil

- A — Niveles 0–1 sin problema; nivel 2 lento; nivel 3 impracticable en vivo (lote nocturno)
- B — Niveles 0–2 en vivo; nivel 3 con VLM pequeño
- C — Todos en vivo

#### Cuándo NO usar esto El ruteo por niveles

- No mande todo al VLM «porque lee todo»: lee todo y **se equivoca en cifras**. Módulo 7.4 lo mide.
- No use OCR clásico en tablas: devuelve una sopa de números sin columnas.

#### PRÁCTICA · Práctica

1. Un contrato de 20 páginas, escaneado, con sellos en 3. ¿Un nivel para todo el documento o por página? Justifique.
2. ¿Qué cambia en el router si el cliente compra un escáner de 300 dpi?

## Referencias

1. Tesseract OCR — <https://github.com/tesseract-ocr/tesseract>
2. Docling — <https://github.com/docling-project/docling> [VOLÁTIL – verificado: 2026-09]
3. PyMuPDF — <https://pymupdf.readthedocs.io/>

## 7.2 · Por qué texto plano destruye layouts

### Objetivos

- Comparar tres conversiones del mismo formulario
- Detectar tablas, columnas, sellos y firmas perdidos

### Lo que se pierde al «convertir a texto»

Un PDF de una factura tiene columnas: descripción, cantidad, precio, total. Al extraer «el texto» se obtiene una lista de palabras en orden de lectura, y el orden de lectura de una tabla **no es el de las filas**. El resultado es correcto letra por letra e inútil como dato.

Lo mismo con: columnas de un contrato, notas al pie, sellos que se superponen al texto, firmas, casillas marcadas, y cualquier cosa cuya información esté en **dónde** está y no solo en **qué** dice.

#### PROCEDIMIENTO · Laboratorio 7.2 · el mismo formulario, tres conversiones

Tome un formulario escaneado con una tabla de 5 filas (una remisión sirve) y conviértalo de tres formas:

```
cd ~/labs/el-tornillo && mkdir -p labs/07-02 && cd labs/07-02
# 1) texto plano por OCR clásico
tesseract remision.png salida-ocr -l spa --psm 6
# 2) parser con estructura (Markdown con tablas) [VOLÁTIL – verificado: 2026-09]
docling remision.pdf --to md --output salida-parser
# 3) VLM: "Transcriba esta remisión como tabla Markdown" con la imagen adjunta
python3 vlm.py remision.png "Transcriba la tabla de esta remisión en Markdown, sin
inventar celdas." > salida-vm.md
```

Compare las tres contra la remisión real, celda por celda:

|                   | Filas correctas | Cifras correctas | Encabezados | Tiempo |
|-------------------|-----------------|------------------|-------------|--------|
| OCR clásico       |                 |                  |             |        |
| Parser con layout |                 |                  |             |        |
| VLM               |                 |                  |             |        |

Lo que debe ver [VOLÁTIL – verificado: 2026-09], típicamente: el OCR clásico tiene las cifras pero pierde las filas; el parser conserva la tabla; el VLM conserva la tabla **y a veces cambia una cifra**. Esa última fila es la razón de la lección 7.4.

**Entregable:** las tres salidas y la tabla comparativa. Commit: «Capa 7.2: lo que destruye el texto plano».

Qué se degrada en cada perfil

- A — OCR y parser en CPU (segundos por página); VLM solo en lote
- B — Los tres en vivo

Cuándo NO usar esto Por qué texto plano destruye layouts

- No convierta a Markdown un documento cuyo valor está en el layout exacto (un plano, un formulario oficial que hay que reproducir): ahí se guarda la imagen y se extraen **campos**.

PRÁCTICA · Práctica

1. ¿Por qué el orden de lectura de una tabla de dos columnas sale «columna 1 entera, luego columna 2» en muchos OCR?
2. Proponga cómo detectar automáticamente que una conversión destruyó una tabla.

Referencias

1. Docling — informe técnico: arXiv:2408.09869 (2024).
2. Tesseract — modos de segmentación de página (--psm): documentación del proyecto.

7.3 · Extracción con esquema

### Objetivos

- Extraer campos con confianza por campo
- Rechazar y encolar lo que no cumple el esquema

## Campos, no texto

La salida útil de un documento para la PyME no es su transcripción: son sus **campos**. Número, fecha, proveedor, NIT, líneas, totales. La extracción con esquema (módulo 5.2) se aplica aquí sobre la salida del nivel que decidió el router, y añade **confianza por campo** y una regla de **rechazo** clara.

## Confianza por campo

El modelo declara, por campo, cuánto confía (0–1). No es una probabilidad calibrada, pero **ordena**: los campos con 0,4 se equivocan mucho más que los de 0,95. Umbral inicial: 0,8. Se ajusta con la medición de 7.4.

Reglas de coherencia además del esquema: `subtotal + iva = total`; fecha no futura ni anterior a 10 años; nit con dígito de verificación válido (algoritmo de la DIAN); número único en la base.

### PROCEDIMIENTO · Laboratorio 7.3 · extractor de facturas de proveedor

`labs/07-03/extraer_factura.py` = router (7.1) + conversión (7.2) + extracción con esquema (5.2) + coherencia + decisión (aceptar / revisar). Corra sobre 40 facturas.

```
python3 extraer_factura.py facturas/*.pdf --salida resultados.jsonl
python3 - <<'PY'
import json
r=[json.loads(l) for l in open("resultados.jsonl")]
print("aceptadas:", sum(1 for x in r if not x.get("_revisión_humana")), "· a revisión:",
      sum(1 for x in r if x.get("_revisión_humana")))
PY
```

Lo que debe ver: una mayoría aceptada y una minoría a revisión, **con el motivo**. Una tasa de revisión del 10–25 % al principio es normal; baja con 7.4.

**Entregable:** el extractor y `resultados.jsonl`. Commit: «Capa 7.3: extracción».

### Qué se degrada en cada perfil

**A** — Lote nocturno: 40 facturas en decenas de minutos

**B** — En vivo: segundos por factura

### Cuándo NO usar esto Extracción con esquema

- No baje el umbral de confianza para «aceptar más»: acepta más errores. El umbral se mueve con la medición, no con la prisa.
- No extraiga sin coherencia: un JSON válido con total inventado pasa el esquema.

### PRÁCTICA · Práctica

1. Implemente la verificación del dígito de verificación del NIT.
2. Una factura trae dos IVA distintos (19 % y 5 %). ¿Cómo cambia el esquema?

### Referencias

1. DIAN — dígito de verificación del NIT: documentación pública de la DIAN (algoritmo módulo 11).
2. Módulo 5.2 de este curso — el esquema y la validación.

## 7.4 · Set de prueba con documentos reales y medición campo a campo

### Objetivos

- Anotar 40 documentos del cliente
- Reportar precisión por campo, no «funciona»

### «Funciona» no es un número

Antes de entregar un extractor se construye un **set de prueba** con documentos **del cliente** —no de internet— anotados a mano, y se mide **campo por campo**. Es la única forma de decir «lee bien el 96 % de los totales» en vez de «funciona».

### Construir el set

1. **40 documentos representativos**: los 3–4 proveedores principales, distintas calidades de escaneo, al menos 5 «raros» (manuscritos, sellos encima, fotos).

2. **Anotación a mano**, por dos personas para 10 de ellos (para medir el desacuerdo humano: si dos personas no coinciden en un campo, el modelo tampoco va a coincidir).
3. **Anonimizar** lo que no sea del proveedor (nombres de personas), pero **no** alterar cifras ni layouts: el set tiene que ser fiel.
4. Guardar como `set/NNN.pdf + set/esperado.json`, con el hash de cada PDF.

## Medir

```
# medir.py - exactitud por campo, y qué documentos fallan
import json
esp = json.load(open("set/esperado.json")); res = json.load(open("resultados.json"))
campos = ["numero", "fecha", "proveedor", "nit", "subtotal", "iva", "total"]
for c in campos:
    ok = sum(res[d][c] == esp[d][c] for d in esp)
    print(f"{c:10s} {ok/len(esp)*100:5.1f} % fallan: {[d for d in esp if res[d][c] != esp[d][c]][:5]}")
```

Lo que debe ver: un porcentaje por campo y **qué documentos** fallan. Los campos con menos del 95 % son donde se trabaja: casi siempre proveedor (nombres largos) y numero (prefijos raros), casi nunca total.

## Qué se hace con la tabla

| Resultado                                       | Acción                                                 |
|-------------------------------------------------|--------------------------------------------------------|
| un campo < 90 % en <b>todos</b> los proveedores | el prompt o el esquema de ese campo                    |
| un campo < 90 % en <b>un</b> proveedor          | regla específica (o pedirle al proveedor un formato)   |
| fallan los 5 «raros»                            | están en el nivel de router equivocado (7.1)           |
| el desacuerdo humano es alto                    | el campo está mal definido; se redefine con el cliente |

**Entregable:** `set/` (40 documentos, `esperado.json`), `medir.py`, y la tabla por campo en `labs/07-04/medicion.md` con la fecha. Commit: «Capa 7: extractor medido».

Nada en la medición. En A, correr el set completo tarda una hora; se hace de noche.

### Cuándo NO usar esto Set de prueba con documentos reales y medición campo a campo

- No mida con los mismos documentos con los que ajustó el prompt: sepárelos (20 para ajustar, 20 para medir).
- No reporte el promedio de todos los campos: esconde el campo malo.

---

## PRÁCTICA · Práctica

1. total 99 %, proveedor 78 %. ¿Qué le dice al cliente y qué hace primero?
2. Diseñe cómo crece el set cada mes sin que la medición deje de ser comparable.

## Referencias

1. Módulo 12 de este curso — este set se convierte en el eval permanente.
2. Módulo 2.3 — la anonimización del set respeta la política de datos.