# **PDFExtract — Guía Definitiva**

Extractor de texto para archivos **.pdf** con **fallback OCR** (Tesseract) y utilidades de **imágenes** y **layout**.  
 Este documento consolida **README** + **USAGE** + **Referencia técnica** (CLI y funciones) con ejemplos listos para pegar.

## **Índice**

1. Instalación
2. Estructura del proyecto
3. Uso rápido
4. Opciones CLI
5. Ejemplos prácticos
6. Flujo de decisión del OCR
7. Referencia de funciones (API interna)
8. Salidas generadas
9. Rendimiento y calidad
10. Solución de problemas
11. Licencia y créditos

## **Instalación**

**Requisitos**

* Windows 10/11
* Python 3.9+ (recomendado 3.10–3.12)
* (Opcional) **Tesseract OCR** instalado en el sistema (UB Mannheim) para OCR

**Entorno virtual y dependencias**

E:\Phyton\PDFExtract\venv\Scripts\Activate.ps1

pip install -r E:\Phyton\PDFExtract\requirements.txt

requirements.txt recomendado:

pymupdf==1.24.9

pytesseract==0.3.13

Pillow==10.4.0

**Configurar Tesseract (opcional pero recomendado)**

En el **Activate.ps1** del venv añade:

$env:TESSERACT\_CMD = "C:\\Program Files\\Tesseract-OCR\\tesseract.exe"

$env:PATH = "C:\\Program Files\\Tesseract-OCR;$env:PATH"

Verificación rápida:

& "C:\\Program Files\\Tesseract-OCR\\tesseract.exe" --version

## **Estructura del proyecto**

E:\Phyton\PDFExtract\

├─ venv\

├─ requirements.txt

├─ pdf\_extract.py

├─ README.md # opcional

└─ USAGE.md # opcional

## **Uso rápido**

### **Un único PDF**

python pdf\_extract.py "E:\docs\archivo.pdf" -v

### **Carpeta completa**

python pdf\_extract.py "E:\docs\lote" -v

### **Cambiar carpeta de salida de textos**

python pdf\_extract.py "E:\docs\lote" -o "E:\salidas\_txt" -v

## **Opciones CLI**

usage: pdf\_extract.py [-h] [-o OUTDIR] [--force-ocr] [--no-ocr]

[--ocr-lang OCR\_LANG] [--dpi DPI] [--min-chars MIN\_CHARS]

[-v] [--tess-cmd TESS\_CMD] [--tess-config TESS\_CONFIG]

[--tess-autotune]

[--save-pages {none,ocr,all}] [--img-dir IMG\_DIR]

[--extract-embedded] [--place-images]

input

### **Generales**

* input (str) — Ruta a un **archivo .pdf** o a una **carpeta** con PDFs.
* -o, --outdir (str, default: ./output\_txt) — Carpeta de salida para .txt.
* -v, --verbose — Modo verboso (logs).

### **OCR / Texto**

* --force-ocr — Fuerza OCR en **todas** las páginas (ignora texto embebido).
* --no-ocr — Deshabilita OCR (solo extracción directa: más rápido).
* --ocr-lang (str, default: spa) — Idiomas Tesseract, p.ej. spa, eng, spa+eng.
* --dpi (int, default: 300) — DPI para **OCR y capturas** (una sola palanca).
* --min-chars (int, default: 20) — Umbral de caracteres para decidir si una página “necesita OCR”.

### **Tesseract**

* --tess-cmd (str) — Ruta explícita a tesseract.exe (prioriza sobre PATH/entorno).
* --tess-config (str, default: "--oem 1 --psm 6") — Flags extra para Tesseract.
* --tess-autotune — Ajusta **automáticamente** --psm por página a 4/6/11 según el layout detectado.

### **Imágenes y Layout**

* --save-pages {none,ocr,all} (default: none) — Guardar capturas de página:  
  + none: no guarda
  + ocr: solo si **se aplicó OCR** en esa página
  + all: todas las páginas
* --img-dir (str, default: ./output\_images/<pdf\_stem>/) — Carpeta base para imágenes/HTML.
* --extract-embedded — Extrae **imágenes incrustadas** (bitstream original cuando sea posible).
* --place-images — Genera **HTML por página** colocando las imágenes recortadas en su **posición exacta**.

## **Ejemplos prácticos**

### **1) Tu caso: páginas escaneadas con layout complejo**

python pdf\_extract.py "E:\docs\escaneados" --tess-config "--oem 1 --psm 4" -v

### **2) Autotune por página (evita forzar un único PSM)**

python pdf\_extract.py "E:\docs\mixto" --tess-autotune -v

### **3) Idiomas combinados (ES + EN) y DPI alto**

python pdf\_extract.py "E:\docs\tech" --ocr-lang "spa+eng" --dpi 400 --tess-autotune -v

### **4) Forzar OCR (todas las páginas)**

python pdf\_extract.py "E:\docs\solo\_imagenes" --force-ocr --tess-config "--oem 1 --psm 4" -v

### **5) Deshabilitar OCR (PDFs buscables)**

python pdf\_extract.py "E:\docs\buscables" --no-ocr -v

### **6) Capturas de todas las páginas (400 DPI)**

python pdf\_extract.py "E:\docs\lote" --save-pages all --dpi 400 -v

### **7) Solo capturas cuando hubo OCR (para auditar resultados)**

python pdf\_extract.py "E:\docs\escaneados" --save-pages ocr --dpi 300 -v

### **8) Extraer imágenes incrustadas**

python pdf\_extract.py "E:\docs\revista" --extract-embedded -v

### **9) Layout HTML con imágenes en su sitio**

python pdf\_extract.py "E:\docs\manual" --place-images --dpi 300 -v

### **10) Elegir carpeta de salida de imágenes/HTML**

python pdf\_extract.py "E:\docs\lote" --img-dir "E:\salidas\_layout" --place-images --extract-embedded --dpi 300 -v

## **Flujo de decisión del OCR**

1. Para **cada página**:  
   * Si --force-ocr: se aplica OCR siempre.
   * En caso contrario, se extrae texto con PyMuPDF.  
      Si los caracteres < --min-chars, se considera “escaneada” y se aplica OCR.
2. Configuración de Tesseract:  
   * --tess-config define los flags base (--oem, --psm, etc.).
   * Si --tess-autotune, se **sustituye** --psm por 4/6/11 según el layout detectado.
3. El texto resultante se acumula y se guarda en <pdf\_stem>.txt (UTF-8).

## **Referencia de funciones (API interna)**

Estas funciones están en pdf\_extract.py. No necesitas llamarlas directamente para uso normal, pero se documentan para mantenimiento/extensión.

### **setup\_logging(verbose: bool) -> None**

* **Descripción**: Configura el nivel de logging (DEBUG si verbose, si no INFO).

### **configure\_tesseract(explicit\_cmd: Optional[str]) -> Optional[str]**

* **Descripción**: Selecciona el binario de Tesseract en este orden: explicit\_cmd → $TESSERACT\_CMD → PATH → rutas típicas de Windows.
* **Parámetros**:  
  + explicit\_cmd — Ruta explícita a tesseract.exe (puede ser None).
* **Return**: Ruta seleccionada o None si no se encuentra.
* **Notas**: Ajusta pytesseract.pytesseract.tesseract\_cmd y hace log INFO con la ruta usada.

### **find\_pdfs(input\_path: pathlib.Path) -> List[pathlib.Path]**

* **Descripción**: Devuelve la lista de PDFs a procesar a partir de **archivo** o **carpeta**.

### **ensure\_outdir(outdir: Optional[pathlib.Path]) -> pathlib.Path**

* **Descripción**: Crea (si no existe) y devuelve la carpeta de salida para .txt.

### **page\_needs\_ocr(text: str, threshold\_chars: int = 20) -> bool**

* **Descripción**: Heurística básica; si la extracción devuelve menos de threshold\_chars, considera la página como “escaneada” y sugiere OCR.

### **make\_images\_outdir(base\_outdir: pathlib.Path, pdf\_stem: str) -> pathlib.Path**

* **Descripción**: Crea (si no existe) la carpeta base de imágenes para un PDF (por nombre de archivo).

### **ensure\_dir(p: pathlib.Path) -> pathlib.Path**

* **Descripción**: Crea (si no existe) p y lo devuelve.

### **save\_page\_capture(page, outdir: pathlib.Path, page\_num: int, dpi: int = 300) -> pathlib.Path**

* **Descripción**: Guarda una **captura PNG** de toda la página con el DPI indicado.
* **Salida**: page\_XXXX.png en el directorio indicado.

### **extract\_embedded\_images\_from\_page(doc, page, outdir: pathlib.Path, page\_num: int) -> int**

* **Descripción**: Extrae **imágenes incrustadas** (bitstream) usando page.get\_images(full=True) y doc.extract\_image(xref).
* **Salida**: Imágenes en embedded/ con nombre pXXXX\_imgN.ext.
* **Return**: Número de imágenes extraídas.

### **get\_image\_blocks(page) -> list**

* **Descripción**: Analiza el page.get\_text("dict") y devuelve una lista de **bloques de imagen** con su **bbox**.

### **render\_full\_page\_png(page, outdir: pathlib.Path, page\_num: int, dpi: int) -> pathlib.Path**

* **Descripción**: Renderiza y guarda la página completa como PNG.

### **crop\_image\_region(page, bbox, dpi: int) -> fitz.Pixmap**

* **Descripción**: Devuelve un **Pixmap** con el recorte exacto del bbox a dpi dado.

### **export\_page\_layout\_with\_images(doc, page, outdir: pathlib.Path, page\_num: int, dpi: int) -> None**

* **Descripción**: Para cada página, genera:  
  + page\_XXXX.png (fondo),
  + crops/img\_XXX.png (recortes de cada bloque de imagen),
  + page\_XXXX.html (imagenes posicionadas con CSS absoluto sobre el fondo),
  + page\_XXXX\_manifest.json (metadatos: bbox en puntos y píxeles).
* **Uso**: Activado desde CLI con --place-images.

### **\_count\_text\_blocks(page) -> int**

* **Descripción**: Cuenta bloques de **texto** para heurística de layout.

### **pick\_psm\_for\_page(page) -> int**

* **Descripción**: Selecciona --psm según layout detectado:  
   >=6 bloques → 4; <=1 bloque → 11; si no, 6.

### **ocr\_page(page, dpi: int = 300, lang: str = "spa", tess\_config: str = "--oem 1 --psm 6", override\_psm: Optional[int] = None) -> str**

* **Descripción**: Renderiza la página a imagen y ejecuta OCR con Tesseract. Si override\_psm no es None, **sustituye** cualquier --psm presente en tess\_config por el indicado.
* **Return**: Texto OCR.

### **extract\_text\_from\_pdf(...) -> pathlib.Path**

**Firma completa**:

extract\_text\_from\_pdf(

pdf\_path: pathlib.Path,

outdir: pathlib.Path,

force\_ocr: bool = False,

disable\_ocr: bool = False,

ocr\_lang: str = "spa",

dpi: int = 300,

min\_chars\_threshold: int = 20,

tess\_config: str = "--oem 1 --psm 6",

tess\_autotune: bool = False,

save\_pages: str = "none", # none|ocr|all

images\_outdir: Optional[pathlib.Path] = None,

extract\_embedded: bool = False,

place\_images: bool = False

) -> pathlib.Path

* **Descripción**: Procesa un **PDF completo** página a página, aplicando extracción directa y OCR bajo las reglas configuradas.
* **Parámetros clave**:  
  + force\_ocr / disable\_ocr — Fuerza OCR o lo desactiva.
  + ocr\_lang — Idiomas OCR (spa, eng, spa+eng).
  + dpi — Afecta a OCR y a capturas/recortes.
  + min\_chars\_threshold — Umbral de “página con poco texto”.
  + tess\_config — Flags a Tesseract.
  + tess\_autotune — Selecciona --psm por página (4/6/11).
  + save\_pages — none|ocr|all para capturas.
  + images\_outdir — Carpeta base de imágenes/HTML.
  + extract\_embedded — Extraer imágenes incrustadas.
  + place\_images — Exportar HTML con imágenes ubicadas.
* **Return**: Ruta del .txt generado.

### **main() -> None**

* **Descripción**: Orquesta el CLI. No necesita llamadas directas.

## **Salidas generadas**

### **Textos**

* output\_txt/<pdf\_stem>.txt — Texto completo en UTF-8 (\n).

### **Imágenes / Layout (si se activan opciones)**

* **Capturas**: output\_images/<pdf\_stem>/page\_XXXX.png
* **Imágenes incrustadas**: output\_images/<pdf\_stem>/embedded/pXXXX\_imgN.ext
* **Layout por página**:  
  + output\_images/<pdf\_stem>/page\_XXXX/page\_XXXX.png
  + output\_images/<pdf\_stem>/page\_XXXX/crops/img\_XXX.png
  + output\_images/<pdf\_stem>/page\_XXXX/page\_XXXX.html
  + output\_images/<pdf\_stem>/page\_XXXX/page\_XXXX\_manifest.json

## **Rendimiento y calidad**

* **DPI**: 300 es un buen equilibrio; 400 mejora precisión (más coste).
* **PSM**: 4 (multi-columna), 6 (bloque), 11 (texto disperso).
* **Autotune**: útil con lotes heterogéneos; evita forzar un PSM único.
* **OCR idiomático**: usa --ocr-lang "spa+eng" si mezclas idiomas.

## **Solución de problemas**

* **TesseractNotFoundError** — Ajusta --tess-cmd o añade Tesseract al PATH / variable TESSERACT\_CMD.
* **OCR pobre** — Sube --dpi, cambia --psm (4/6/11), combina idiomas.
* **OCR innecesario** — Baja --min-chars o usa --no-ocr si el PDF es buscable.
* **Imágenes no aparecen en HTML** — Abre page\_XXXX.html junto a su carpeta crops y la page\_XXXX.png correspondiente.

## **Licencia y créditos**

* Licencia sugerida: **MIT** (ajústala a tus necesidades).
* Créditos: **PyMuPDF** para render/extracción; **Tesseract** + **pytesseract** para OCR; **Pillow** para imágenes.