Extraer textos e imágenes de documentos PDF y otras imágenes con Python, PyMuPDF y OCRMyPDF

Publicado el El Informatico - 15 de abril de 2022 -

Obtención de texto a partir de un documento PDF

Con este artículo, aprenderás a usar el módulo PyMuPDF y OCRMyPDF para obtener texto, imágenes y fuentes de archivos PDF y escanear archivos PDF con un módulo OCR para poder obtener texto a partir de las imágenes del documento.

PyMuPDF

PyMuPDF es un módulo para Python que te permite leer, modificar y crear documentos PDF de manera sencilla. Está basado en el módulo 'fitz', el cuál ya viene incorporado en el mísmo modulo. También incorpora un módulo OCR basado en TESSERACT con el que es posible extraer texto a partir de una imágen, si bien recomiendo usar el módulo OCRMyPDF para eso.

Instalación

PyMuPDF

Si eres usuario de fitz, deberás desinstalar dicho módulo primero.

```
> pip remove fitz
```

PyMuPDF ya incorpora el módulo fitz, y no es posible usar ambos al mísmo tiempo. No tienes que preocuparte, ya que puedes usar el que viene incorporado con PyMuPDF sin problemas. No tendrás ni que cambiar nada en el código donde lo uses.

Para instalar PyMuPDF:

```
> pip install pymupdf
```

OCRMyPDF

Primero hay que instalar los binarios de ghostscript, tesseract y pngquant. En Linux, instalalo desde tu gestor de paquetes. Por ejemplo, en Debian/Ubuntu:

```
# apt install ghostscript tesseract-ocr pngquant
```

En Windows, instalar OCRMyPDF requiere usar PowerShell. Lo más sencillo es usar el gestor de paquetes Chocolatey. Para ello, inicia una nueva instancia de PowerShell con permisos de administración, y ejecuta los siguientes comandos:

```
PS > Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

Reinicia PowerShell (como administrador) y ejecuta los siguientes comandos para instalar los módulos requeridos:

```
PS > choco install --pre tesseract
PS > choco install ghostscript pngquant unpaper
```

Una vez tengas tesseract y ghostscript (y pngquant), instala ocrmypdf desde PIP:

```
> pip install ocrmypdf
```

Variables de entorno

Hay que establecer la variable de entorno TESSDATA_PREFIX para que apunte al directorio donde tenemos todos los modelos para el OCR.

En Linux puede ser uno de tres directorios:

```
/usr/share/tesseract-ocr/tessdata  /usr/share/tessdata  /usr/share/tesseract-
ocr/4.00/tessdata
```

Localiza cuál de los tres es el correcto, por ejemplo:

```
# find / -type d -name "tessdata"
```

y añadelo a la variable de entorno TESSDATA_PREFIX. Deberás añadirlo al perfíl de tu shell. Por ejemplo, si usas BASH:

```
$ vim ~/.profile
```

Añade la siguiente línea al final asignando a la variable la ruta correcta:

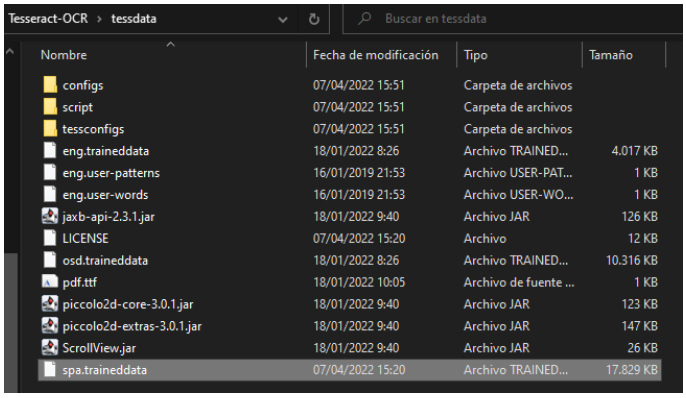
```
export TESSDATA_PREFIX = /usr/share/tesseract-ocr/tessdata
```

En Windows, puedes hacerlo desde PowerShell usando SetEnvironmentVariable (como administrador):

```
PS > [Environment]::SetEnvironmentVariable('TESSDATA_PREFIX','C:\Archivos de programa\Tesseract-OCR\tessdata', 'Machine')
```

Añadir el lenguaje Español a TESSDATA

Descarga el archivo spa.trainedata del repositorio de tesseract-ocr y mueve el archivo a la carpeta *tessdata* que has localizado en el punto anterior.



Si necesitas otros idiomas, puedes descargarlos desde el repositorio que he enlazado antes. Tesseract ya incorpora por defecto el idioma inglés (eng.trainedata), así que no tienes que descargarlo.

Usando PyMuPDF en Python

Abrir un archivo PDF

Para abrir un archivo PDF para su lectura, importamos fitz (que ya viene con PyMuPDF) y usamos **fitz.open**:

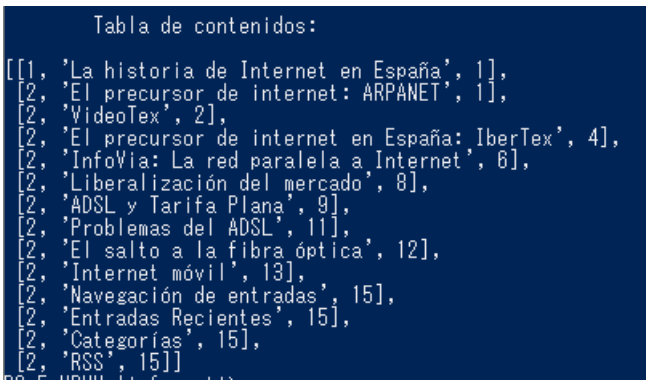
```
import fitz
from pprint import pprint

if __name__ == "__main__":
    # Abrir el documento
    # (https://elinformati.co/pdf/internet_historia_elfinformatico.pdf)
    doc = fitz.open("internet_historia_elfinformatico.pdf")

    # Obtener algunas propiedades del documento
    print(f"""
        Paginas: {str(doc.page_count)}
        Metadatos: {doc.metadata}\n
        Tabla de contenidos:
    """)

    # Imprimir la tabla de contenidos (TOC)
    pprint(doc.get_toc())

    # Cerrar el documento
    # doc.close()
```

La función open de la clase fitz abre el documento PDF y devuelve un objeto Document con las propiedades y métodos relativos al documento.

Tabla de contenidos

La tabla de contenidos se obtiene mediante la función get_toc() del objeto Document. Contiene una lista de listas con la información de cada sección del documento a modo de indice. Cada lista dentro de la lista del TOC contiene tres valores, en el siguiente formato:

[Valor jerárquico (int), Nombre (str), Número de página (int)]



TOC del documento

Abrir una única página

Para abrir una única página, usamos la función load_page(int) de la clase Document. El parámetro numérico representa el número de página (indice 0):

```
    pagina = doc.load_page(0)
    print(pagina.get_textpage().extractText())
```

```
La historia de Internet en España
Publicado el El Informatico - 16 de junio de 2021 -
Kit de conexión a InfoVia de 1996. Incluye un MODEM de 14.400bps (baudios por segundo), y varios
disquetes de 3,5"  con todo el software de conexión.
¿Alguna vez te has preguntado cómo era Internet en los años 80 y 90? ¿Necesitas
algo de información para tu trabajo de clase? ¿O simplemente sientes nostalgia por
la tecnología del pasado? Da igual lo que busques, en éste artículo aprenderás como
fue la transición a Internet en España.
El precursor de internet: ARPANET
El precursor de internet: ARPANET
ARPANET fue una red de sistemas informáticos descentralizada fundada en 1969 en
EEUU. Su origen es militar, ya que se propuso como un medio de comunicación para
las tropas americanas en caso de guerra, tras la crisis de los misiles nucleares de
Cuba en 1962. La idea era la de una red de comunicaciones que, en el caso de que un
nodo fuese destruido, la red pudiera seguir en funcionamiento.

PS E:¥PY¥elinformati>
```

Texto de la primera página del documento

Iterar a traves de las páginas del documento

Podemos usar el iterador Document.pages(start,number,step) para iterar por todas las páginas del documento. Por ejemplo, si quisieramos extraer el texto de cada página:

```
    for pagina in doc.pages(step=1):
        internet = 0
        texto = pagina.get_textpage().extractText()
        palabras = texto.lower().strip().split(" ")
        for palabra in palabras:
            if palabra == "internet":
                internet += 1
        print(f"Veces que se repite la palabra 'internet' en la página
{str(pagina.number)}: {str(internet)}")
```

```
Veces que se repite la palabra 'internet' en la página 0: 3
Veces que se repite la palabra 'internet' en la página 1: 1
Veces que se repite la palabra 'internet' en la página 2: 0
Veces que se repite la palabra 'internet' en la página 3: 2
Veces que se repite la palabra 'internet' en la página 4: 0
Veces que se repite la palabra 'internet' en la página 5: 0
Veces que se repite la palabra 'internet' en la página 6: 0
Veces que se repite la palabra 'internet' en la página 7: 4
Veces que se repite la palabra 'internet' en la página 8: 3
Veces que se repite la palabra 'internet' en la página 9: 0
Veces que se repite la palabra 'internet' en la página 10: 0
Veces que se repite la palabra 'internet' en la página 11: 0
Veces que se repite la palabra 'internet' en la página 12: 2
Veces que se repite la palabra 'internet' en la página 13: 1
Veces que se repite la palabra 'internet' en la página 14: 2
Veces que se repite la palabra 'internet' en la página 15: 0
PS E:¥PY¥elinformati>
```

Extraer texto

Podemos extraer directamente todo el texto de una página con la función extractText() de la clase TextPage. Esta función devolverá un string (str) con el texto plano de la página. Así que, primero, obtenemos el objeto TextPage de la página, para despues obtener su texto con extractText():

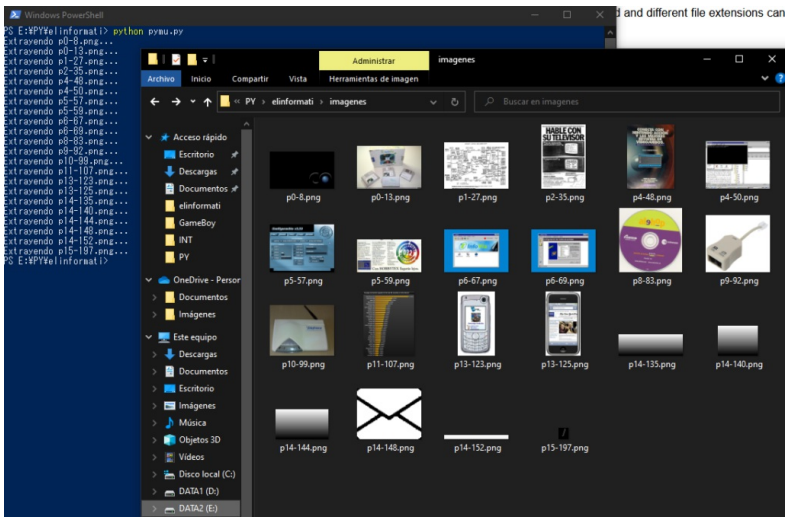
```
texto = pagina.get_textpage().extractText()
print(texto)
```

Extraer imágenes

Puedes extraer las imágenes de una página usando la función get_page_images(int) de la clase Document. Después puedes convertirlas a un objeto Pixmap para procesarlas después.

Por ejemplo, si quisieramos extraer todas las imágenes del documento:

```
    for pagina in doc.pages(step=1):
        for imagen in doc.get_page_images(pagina.number):
            xref = imagen[0] # Obtiene el valor xref de la imágen
            pix = fitz.Pixmap(doc, xref) # Obtiene el Pixmap de la imágen
            print("Extrayendo p%s-%s.png..."%(pagina.number, xref))
            if pix.n < 5: # La imágen está en formato RGB, la guardamos
                pix.save("./imagenes/p%s-%s.png"%(pagina.number, xref))
            else: # La imágen está en formato CMYK, lo convertimos a RGB
primero
                pixrgb = fitz.Pixmap(fitz.csRGB, pix)
                pixrgb.save("./imagenes/p%s-%s.png"%(pagina.number,
xref))
```

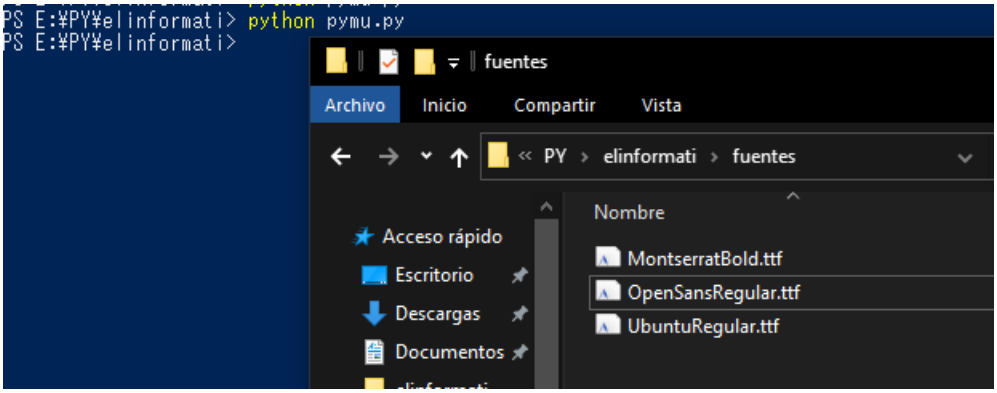


Imágenes extraidas del documento PDF

Extraer fuentes

Podemos usar el método get_fonts de la clase Page para obtener las fuentes de una página, y el método extract_font de Document para extraerla:

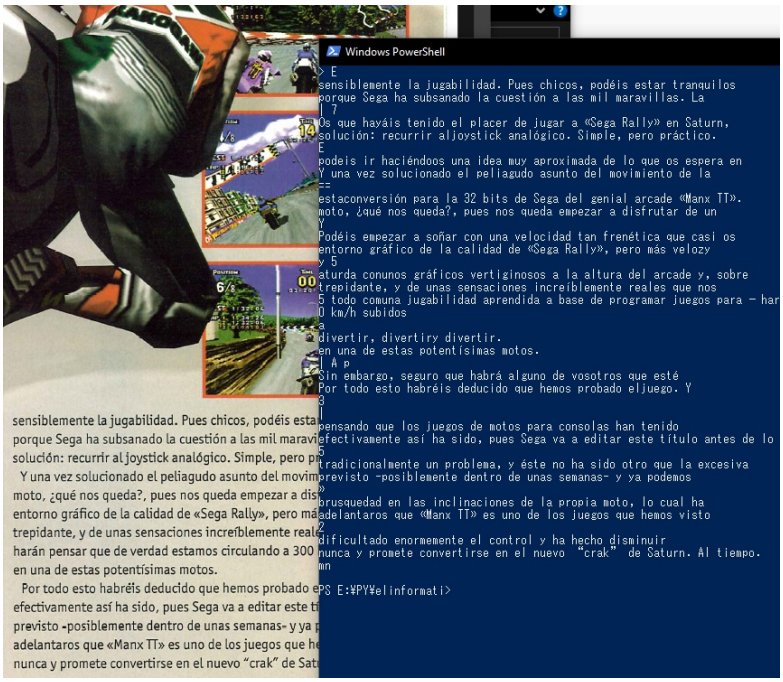
```
pagina = doc.load_page(0)
fonts = pagina.get_fonts(full=True)
for font in fonts:
    xref = font[0]
    nombre, ext, _, contenido = doc.extract_font(xref)
    if contenido is not None:
        archivo = open("./fuentes/" + nombre + "." + ext, "wb")
        archivo.write(contenido)
        archivo.close()
```



Escanear documentos con OCR

PyMuPDF tiene la función de escanear documentos usando OCR para poder extraer texto de imágenes. Es muy sencillo de usar, sólo tenemos que usar el método get_textpage_ocr de la página especificando el lenguaje en el que está escrito y el DPI de la página, y de ahí obtener el texto extraido.

```
# https://archive.org/details/HobbyConsolas066 (Descargado como PDF sin
texto)
doc = fitz.open("Hobby_Consolas_066.pdf")
pagina = doc.load_page(31)
texto_ocr = pagina.get_textpage_ocr(language="spa",dpi=300,full=True)
texto = texto_ocr.extractText()
print(texto)
doc.close()
```

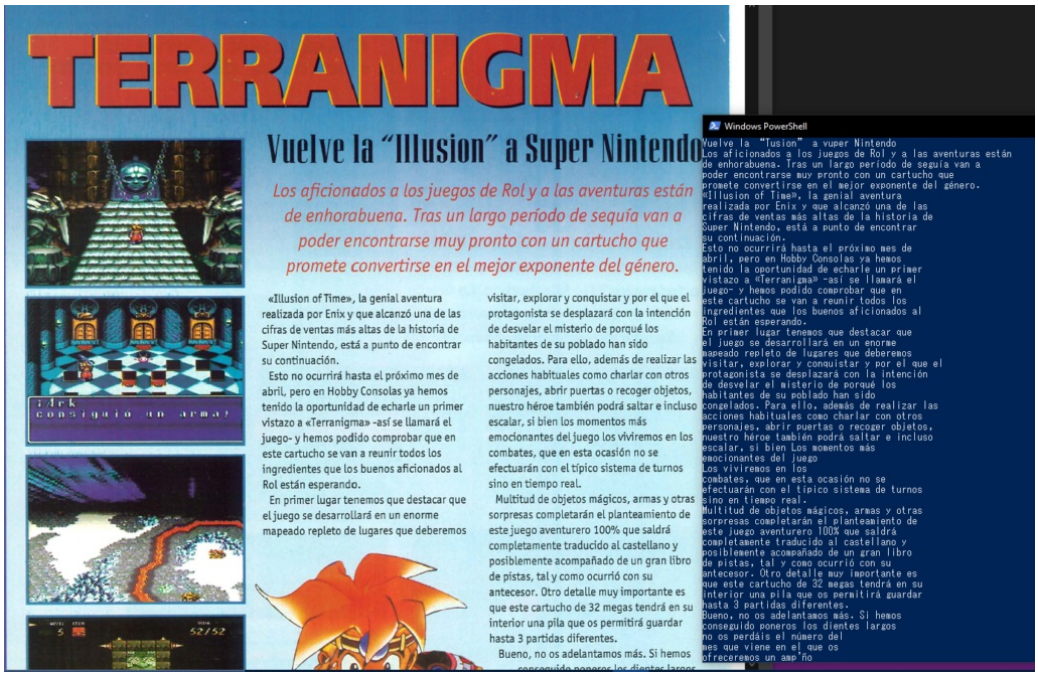


Texto extraido

Escanear con OCRMyPDF

OCRMyPDF es otro módulo que nos permite usar OCR en nuestros documentos. Al igual que MuPDF (OCR usado por PyMuPDF), está basado en TESSERACT para la extracción de texto.

```
import fitz
from pprint import pprint
import ocrmypdf # pip install ocrmypdf
import io

def escanear_pagina_ocr(page: int, language: str) -> str:
    src = page.parent  # Obtener el documento
    doc = fitz.open()  # Archivo temporal
    doc.insert_pdf(src, from_page=page.number, to_page=page.number) #
Copia el documento al temporal
    pdfbytes = doc.tobytes() # Convertir a bytes
    inbytes = io.BytesIO(pdfbytes)  # Convertir a BytesIO
    outbytes = io.BytesIO()  # Salida de datos
    ocrmypdf.ocr(
        inbytes,  # Entrada (documento temporal)
        outbytes,  # Salida
        language=language,  # Idioma del documento
        output_type="pdf",  # Salida del formato (PDF)
    )
    ocr_pdf = fitz.open("pdf", outbytes.getvalue())  # Generar un objeto
de fitz con la salida
    text = ocr_pdf[0].get_text()  # Extraer el texto
    return text  # Devolverlo

if __name__ == "__main__":
    # Abrir el documento
    doc = fitz.open("Hobby_Consolas_066.pdf")
    # Carga la página 34 (indice 0)
    pagina = doc.load_page(33)
    # Obtener el texto de la página con OCR
    texto = escanear_pagina_ocr(pagina,"spa")
    # Mostrarlo en pantalla
    print(texto)
    # Cierra el documento
    doc.close()
```



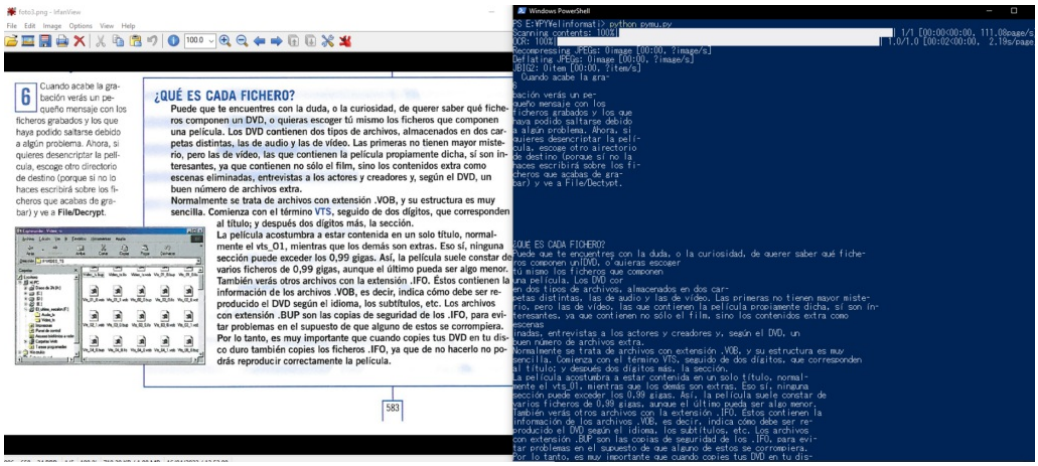
El texto reconocido no es perfecto y dependerá sobre todo de la calidad de las

imágenes sobre las que se aplica, la tipografía usada, el contraste entre el fondo y el texto, y la orientación del texto. Es imprescindible que el texto se muestre en línea y que no tenga ningún tipo de distorsión (por ejemplo, si escaneas un libro, que las páginas no estén dobladas y el texto no se muestre torcido). Probablemente tengas que realizar algunas correcciones al texto.

Escanear imágenes

De la mísma forma que hemos escaneado PDFs, también podemos obtener texto de imágenes. Podemos usar Pillow para cargar la imágen, y de paso aprovechar para realizar algunos ajustes de contraste que podrían ayudar a mejorar la calidad del escaneo OCR. Después le pasamos los datos a OCRMyPDF, y extraemos el texto con fitz igual que hemos hecho antes.

```
import fitz
import ocrmypdf
import io
import math
from PIL import Image, ImageOps, ImageEnhance

def ocr_imagen(img: str, idioma: str = "eng+spa", contraste: float = 0.0)
-> str:
    # Abrimos la imágen usando PIL
    with Image.open(img) as im:
        # OCRMyPDF no trabaja con imágenes con transparencia, así que
        # comprobamos si hay transparencia y, si la hay, la eliminamos.
        if im.mode in ('RGBA', 'LA') or (im.mode == 'P' and
'transparency' in im.info):
            sinalpha = im.convert('RGB')
        else:
            sinalpha = im
        # Obtenemos los DPI del documento. Si no los especifica, le
asignamos
        # 100 por defecto.
        imgdpi = 0
        if 'dpi' in im.info:
            imgdpi = im.info['dpi'][0]
        if imgdpi < 1:
            imgdpi = 100
        # Para intentar mejorar la calidad del escaneo, podemos convertir
la
        # imágen a escala de gríses y ajustar el brillo usando PIL
        byn = ImageOps.grayscale(sinalpha)
        contrastador = ImageEnhance.Contrast(byn)
        contrastada = contrastador.enhance(contraste)
        # Creamos dos streams de bytes para entrada y salida de datos
        inbytes = io.BytesIO()
        outbytes = io.BytesIO()
        # Guardamos los datos de la imágen en el stream de entrada
        contrastada.save(inbytes, format=im.format)
        # No debemos olvidarnos de rebobinar el stream
        inbytes.seek(0)
        # Llamamos a ocrmypdf para que escanee la imágen y nos pase los
datos como PDF
        ocrmypdf.ocr(
            inbytes,
            outbytes,
            image_dpi=int(math.floor(imgdpi)),
            language=idioma,
            output_type="pdf"
        )
        # Abrimos el PDF generado con fitz
        ocr_pdf = fitz.open("pdf", outbytes.getvalue())
        # Obtenemos el texto escaneado, ya como texto plano
        texto = ocr_pdf[0].get_text()
        # Cerramos el documento para liberar memoria
        ocr_pdf.close()
        # Devolvemos el texto
        return texto

if __name__ == "__main__":
    # Probamos la función
    print(ocr_imagen("foto3.png", idioma="spa", contraste=4.0))
```

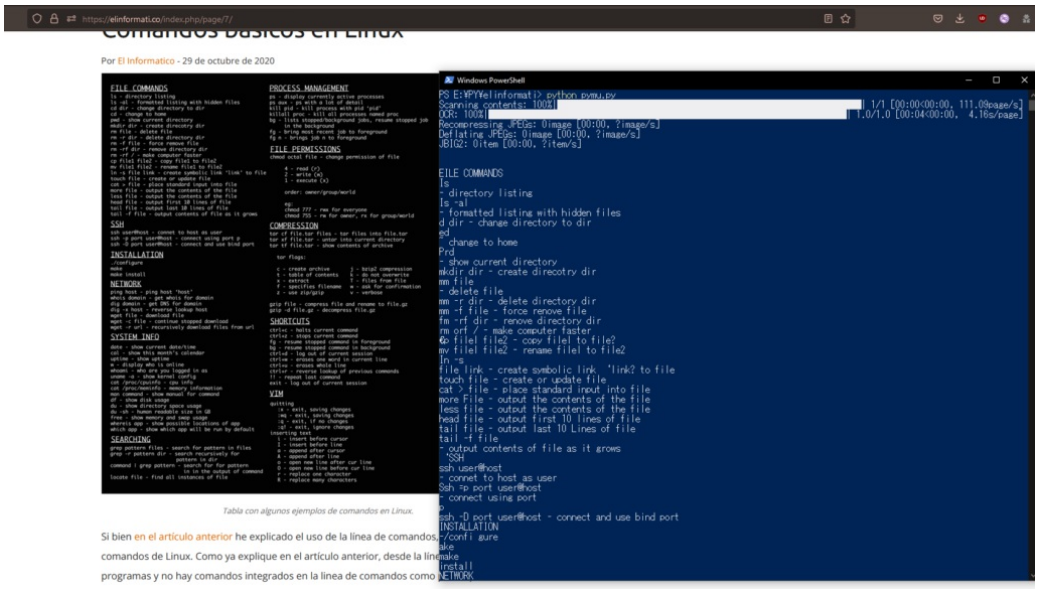


OCR sobre una imágen

Por último, ya puestos, podemos descargar una imágen de internet y escanearla para extraer el texto añadiendo requests a la combinación.

```
import fitz
from pprint import pprint
import ocrmypdf
import io
import requests
import math
from PIL import Image, ImageOps, ImageEnhance

def ocr_imagen_de_internet(url: str, idioma: str = "spa", contraste:
float = 0.0) -> str:
    # Cabeceras HTTP
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:98.0) Gecko/20100101 Firefox/98.0'}
    # Obtenemos la imágen usando requests como un flujo de datos.
    with requests.get(url, headers=headers, stream=True) as r:
        imgbytes = io.BytesIO()
        r.raise_for_status()
        for chunk in r.iter_content(chunk_size=8192):
            imgbytes.write(chunk)
        imgbytes.seek(0)
        # Cargamos la imágen con PIL desde el flujo de datos
        with Image.open(imgbytes) as im:
            # OCRMyPDF no trabaja con imágenes con transparencia, así que
            # comprobamos si hay transparencia y, si la hay, la
eliminamos.
            if im.mode in ('RGBA', 'LA') or (im.mode == 'P' and
'transparency' in im.info):
                sinalpha = im.convert('RGB')
            else:
                sinalpha = im
            # Obtenemos los DPI del documento. Si no los especifica, le
asignamos
            # 100 por defecto.
            imgdpi = 0
            if 'dpi' in im.info:
                imgdpi = im.info['dpi'][0]
            if imgdpi < 1:
                imgdpi = 100
            # Para intentar mejorar la calidad del escaneo, podemos
convertir la
            # imágen a escala de gríses y ajustar el brillo usando PIL
```

```
            byn = ImageOps.grayscale(sinalpha)
            contrastador = ImageEnhance.Contrast(byn)
            contrastada = contrastador.enhance(contraste)
            # Creamos dos streams de bytes para entrada y salida de datos
            inbytes = io.BytesIO()
            outbytes = io.BytesIO()
            # Guardamos los datos de la imágen en el stream de entrada
            contrastada.save(inbytes, format=im.format)
            # No debemos olvidarnos de rebobinar el stream
            inbytes.seek(0)
            # Llamamos a ocrmypdf para que escanee la imágen y nos pase
los datos como PDF
            ocrmypdf.ocr(
                inbytes,
                outbytes,
                image_dpi=int(math.floor(imgdpi)),
                language=idioma,
                output_type="pdf"
            )
            # Abrimos el PDF generado con fitz
            ocr_pdf = fitz.open("pdf", outbytes.getvalue())
            # Obtenemos el texto escaneado, ya como texto plano
            texto = ocr_pdf[0].get_text()
            # Cerramos el documento para liberar memoria
            ocr_pdf.close()
            # Devolvemos el texto
            return texto

if __name__ == "__main__":
    # Probamos la función
    print(ocr_imagen_de_internet("https://elinformati.co/wp-
content/uploads/2020/12/PG1ulC4-1024x893.png", idioma="eng",
contraste=4.0))
```



OCR sobre una imágen de la web

ANTERIOR

QB64 – Interprete de BASIC moderno para sistemas de 32 y 64 bits

SIGUIENTE

Crear documentos PDF con Python y PyMuPDF

Entradas Recientes

Categorías

RSS

Subscribirse al feed RSS

PDFs
Manuales
Política de privacidad
Política de Cookies
Acerca de mi
Acerca de ElInformati.co