InvestiGO
Revista Científica Multidisciplinaria
ISSN 2953-6367
http://revistainvestigo.com

Julio - diciembre 2026
Vol. 7 No. 20, PP. 299-313
https://doi.org/10.56519/jw1x1e61

ANÁLISIS COMPARATIVO DE LIBRERÍAS PYTHON PARA EXTRACCIÓN DE TEXTO DESDE DOCUMENTOS PDF DIGITALES

COMPARATIVE ANALYSIS OF PYTHON LIBRARIES FOR TEXT EXTRACTION FROM DIGITAL PDF DOCUMENTS

Angel P. Flores-Orozco[1], Omar G. Bravo-Quezada[2], Diana E. Gómez-García[3]

{aflores@espoch.edu.ec[1], obravo@ups.edu.ec[2], elizabeth.gomez@espoch.edu.ec[3]}

Fecha de recepción: 14/05/2026 / Fecha de aceptación: 22/06/2026 / Fecha de publicación: 08/07/2026

RESUMEN: La extracción automática de texto desde documentos PDF constituye una tarea fundamental para aplicaciones de minería de texto, recuperación de información, gestión documental y construcción de corpus lingüísticos. El objetivo de esta investigación fue comparar el desempeño de cuatro librerías Python para extracción de texto desde PDF: PyMuPDF, pdfminer.six, pdfplumber y pypdf. El experimento se desarrolló con cuatro documentos PDF digitales de 10, 25, 50 y 100 páginas con estructura de complejidad media: tablas, encabezados repetitivos, gráficos con texto, dos idiomas; estos fueron procesados bajo las mismas condiciones computacionales y mediante una rutina homogénea de extracción, limpieza básica y segmentación por punto. Las métricas evaluadas fueron tiempo de extracción, total de frases, porcentaje de frases válidas, porcentaje de duplicados, escalabilidad temporal y páginas con error. Los resultados mostraron que PyMuPDF alcanzó el mejor rendimiento temporal, con un promedio de 0.24 segundos por documento, mientras que pdfminer.six obtuvo el mayor porcentaje de frases válidas, con 59.18 %. pdfplumber presentó el menor porcentaje promedio de duplicados, con 38.12 %. El análisis estadístico evidenció que las diferencias de tiempo fueron significativas al considerar la transformación logarítmica y el bloqueo por documento, mientras que las diferencias de calidad textual fueron más moderadas. Se concluye que la selección de la librería debe depender del objetivo de uso: velocidad, mayor recuperación textual o menor redundancia.

Palabras clave: extracción de texto, PDF, Python, procesamiento documental, corpus lingüístico, minería de texto

ABSTRACT: Automatic text extraction from PDF documents is a fundamental task for text mining, information retrieval, document management, and linguistic corpus building

[1]Grupo de Investigación INFOSO, Escuela Superior Politécnica de Chimborazo sede Morona, Ecuador, https://orcid.org/0000-0003-1484-2949; +593 99 987 9382.

[2]Grupo de Investigación GIHP4C, Universidad Politécnica Salesiana sede Cuenca, Ecuador, https://orcid.org/0000-0002-5543-5672; + +593 96 322 3221.

[3]Carrera Tecnologías de la Información, Escuela Superior Politécnica de Chimborazo sede Morona, Ecuador, https://orcid.org/0009-0004-1182-6980; +593 98 451 0615.

applications. The objective of this research was to compare the performance of four Python libraries for text extraction from PDFs: PyMuPDF, pdfminer.six, pdfplumber, and pypdf. The experiment was conducted with four digital PDF documents of 10, 25, 50, and 100 pages with complex structures: tables, repetitive headings, and graphics with text and two languages. These were processed under the same computational conditions and using a homogeneous routine of extraction, basic cleaning, and point segmentation. The metrics evaluated were extraction time, total number of phrases, percentage of valid phrases, percentage of duplicates, time scalability, and pages with errors. The results showed that PyMuPDF achieved the best performance in terms of time, with an average of 0.24 seconds per document, while pdfminer.six obtained the highest percentage of valid phrases, at 59.18%. pdfplumber showed the lowest average percentage of duplicates, at 38.12%. Statistical analysis revealed that the time differences were significant when considering logarithmic transformation and document locking, while the differences in text quality were more moderate. It is concluded that the selection of the library should depend on the intended use: speed, greater text retrieval, or less redundancy.

Keywords: text extraction, PDF, Python, document processing, linguistic corpus, text mining

INTRODUCCIÓN

Los documentos en formato Portable Document Format (PDF) dominan en la actualidad el panorama digital como estándar para la distribución, preservación y consulta de documentos digitales en contextos académicos, administrativos, técnicos y científicos. Su adopción universal obedece a su capacidad para preservar la integridad y mantener el diseño original de los documentos, lo que favorece su uso en bibliotecas digitales, repositorios institucionales, publicaciones científicas y archivos documentales (1).

El PDF se ha consolidado como el formato de facto en contextos donde la fidelidad visual es crítica, especialmente en documentos legales, financieros, científicos y administrativos, donde la apariencia exacta del contenido es tan importante como el contenido mismo, a pesar de esta omnipresencia, el desafío de extraer información textual de archivos PDF de manera precisa y automatizada continúa siendo una tarea compleja y multifacética. Esta dificultad surge de múltiples factores interrelacionados: la arquitectura interna del formato PDF, que puede variar significativamente entre documentos, las inconsistencias en la codificación de caracteres, las variaciones estructurales en la disposición de contenido, y la diversidad en cómo se presenta visualmente la información en la página (2).

Los lenguajes de marcado estructurado tales como XML u HTML presentan una ventaja fundamental: su arquitectura se basa en una declaración explícita de metadatos a través de etiquetas, que facilita la interpretación semántica del contenido. El formato PDF, por el contrario, fue concebido con una finalidad distinta: la preservación de la fidelidad visual y la disposición espacial del documento original. Este enfoque de diseño implica una consecuencia operativa significativa: aunque la presentación tipográfica resulte legible y coherente para el usuario final, la representación interna del documento se encuentra fragmentada en elementos geométricos

bloques de texto, coordenadas, objetos gráficos que no conservan una estructura semántica explícita. Esta discrepancia entre la manifestación visual y la codificación interna genera desafíos sustanciales durante los procedimientos de extracción automatizada, particularmente cuando los documentos contienen componentes como matrices de datos, encabezados seriados u organizaciones complejas del contenido en múltiples subsecciones (1). Consecuentemente, la recuperación programática de información textual desde archivos PDF demanda el empleo de algoritmos sofisticados y protocolos de procesamiento específicamente diseñados para normalizar y consolidar el contenido extraído con un nivel aceptable de fidelidad.

La capacidad de extraer y procesar automáticamente contenido desde documentos PDF se ha vuelto cada vez más importante en aplicaciones relacionadas con análisis de datos e inteligencia artificial. En la actualidad, este tipo de procesamiento se utiliza en escenarios muy diversos, como la digitalización de archivos históricos, la preservación de documentos patrimoniales, el análisis de publicaciones científicas y la automatización documental en instituciones públicas y privadas. También tiene un papel relevante en sistemas de recuperación de información y en procesos de transformación digital, donde grandes volúmenes de documentos necesitan ser procesados sin intervención manual constante (3). Disponer de herramientas capaces de recuperar texto de manera rápida y consistente representa una necesidad práctica más que únicamente técnica.

La importancia de una extracción textual precisa ha aumentado aún más con el crecimiento de los sistemas de Recuperación Aumentada por Generación (RAG) y de los modelos de lenguaje de gran escala. Estos sistemas dependen directamente de la calidad del texto obtenido durante las primeras etapas del procesamiento documental. Si la extracción introduce errores, duplicados o fragmentación excesiva, esos problemas se propagan hacia las etapas posteriores de indexación, recuperación y generación de respuestas. Estudios recientes señalan que una selección inadecuada de herramientas de extracción puede provocar pérdida de información relevante y afectar el rendimiento general de todo el sistema (4). En documentos complejos, no basta únicamente con extraer grandes cantidades de texto; también es necesario conservar una estructura suficientemente coherente que facilite su posterior análisis y utilización por otros sistemas automáticos.

En el ecosistema Python, que se ha convertido en el estándar para investigadores y desarrolladores en ciencia de datos y procesamiento de lenguaje natural, existen múltiples librerías disponibles para realizar extracción de texto desde PDF. Cada una de estas herramientas implementa diferentes enfoques, con fortalezas y limitaciones particulares que las hacen más o menos adecuadas según el contexto específico de aplicación. Este trabajo presenta un estudio comparativo sistemático de cuatro librerías ampliamente utilizadas en la comunidad: pdfminer.six, pdfplumber, PyMuPDF (también conocido como fitz) y pypdf. Estas herramientas representan diferentes filosofías de diseño arquitectónico: desde enfoques basados en el análisis estructural profundo del formato PDF, que sacrifican velocidad por precisión, hasta optimizaciones centradas en velocidad de procesamiento mediante compilación en lenguajes de bajo nivel, pasando por soluciones que buscan balances intermedios entre ambos extremos (5-8).

Aunque existen múltiples librerías para extracción de texto desde PDF, todavía son limitados los estudios que realizan comparaciones experimentales bajo condiciones homogéneas y utilizando métricas cuantitativas reproducibles. En muchos casos, las investigaciones previas se han centrado en tipos específicos de documentos o en el análisis aislado de determinadas herramientas, dificultando la identificación de ventajas y limitaciones reales entre librerías (9-10). Factores como el tamaño del documento, la presencia de tablas o la estructura visual suelen influir directamente en los resultados de extracción, por lo que comparar herramientas en diferentes escenarios documentales resulta necesario para obtener conclusiones más consistentes (11).

El objetivo principal del estudio fue generar evidencia cuantitativa que permita orientar la selección de herramientas de extracción textual según diferentes necesidades de procesamiento documental. Más allá de identificar cuál librería es más rápida o cuál recupera mayor cantidad de texto, el interés estuvo en analizar el equilibrio entre rendimiento computacional, calidad de segmentación y redundancia del contenido extraído (12). Los resultados obtenidos pueden servir como referencia para investigadores, desarrolladores y profesionales que trabajan en minería de texto, construcción de corpus, sistemas de recuperación de información o aplicaciones basadas en inteligencia artificial, especialmente en contextos donde se requiere procesar grandes volúmenes de documentos PDF de forma automatizada.

MATERIALES Y MÉTODOS

La investigación se inclinó por un enfoque cuantitativo, con diseño experimental comparativo. Se evaluaron cuatro librerías especializadas en extracción de texto desde documentos PDF digitales: PyMuPDF, pdfminer.six, pdfplumber y pypdf. La selección de estas librerías Python se justificó por su uso frecuente en proyectos de procesamiento documental, su disponibilidad por ser open source y su facilidad para escenarios de fácil reproducción en Python (5-8).

El corpus de prueba estuvo conformado por cuatro documentos PDF digitales de diferente extensión: 10, 25, 50 y 100 páginas. Estos documentos presentaron una estructura heterogénea de complejidad media, se caracterizaron por: contenido bilingüe Shuar-Español, múltiples columnas, encabezados repetitivos, elementos gráficos, incluyen tablas complejas, alineaciones irregulares, saltos de línea no estructurados y combinaciones de texto e imágenes; condiciones que incrementan la dificultad de la extracción automática.

La estrategia metodológica excluyó OCR. Puesto que se trataba de PDFs con contenido de texto ya digitalizado, recurrir a técnicas de reconocimiento óptico habría introducido complejidad innecesaria. El objetivo era centrar la atención específicamente en cómo cada librería se desempeña frente a cadenas de texto ya codificadas dentro del documento PDF, sin la interferencia de procedimientos de extracción de imágenes. De esta manera, fue posible evitar que variables externas como la calidad de imagen, la resolución de escaneo, ruido de fondo o errores de reconocimiento de caracteres; influyeran en los hallazgos.

Los experimentos se llevaron a cabo utilizando el entorno en la nube Google Colab, con Python 3 como lenguaje de programación. La infraestructura computacional disponible consistía en procesadores CPU con una capacidad de 12 gigabytes de memoria RAM. El pipeline de procesamiento incluyó: lectura y apertura del PDF, recuperación del contenido textual segmentado por página, normalización del texto (eliminación de saltos de línea innecesarios y consolidación de espacios repetidos), división del contenido en unidades más pequeñas utilizando el punto como elemento de separación, cuantificación de términos y símbolos, validación de coherencia textual para cada segmento, e identificación de repeticiones exactas. Operacionalmente, definimos como "frase" cualquier secuencia de caracteres delimitada por un punto. Si bien esta definición no captura la totalidad de fenómenos lingüísticos posibles en español, tuvo la ventaja de proporcionar criterios de segmentación uniformes e idénticos para todas las herramientas bajo examen.

El proceso de limpieza aplicado al texto extraído consistió únicamente en la eliminación de saltos de línea, normalización de espacios múltiples y eliminación de espacios innecesarios al inicio y final de cada fragmento textual. Posteriormente, se calcularon métricas como número de palabras, número de caracteres, porcentaje de frases válidas, duplicados y tiempo de extracción. Para la identificación de frases válidas se establecieron criterios mínimos automáticos: al menos tres palabras, un mínimo de diez caracteres y presencia de caracteres alfabéticos.

El análisis estadístico se realizó con el propósito de determinar si las diferencias observadas entre las librerías eran significativas. Se aplicó inicialmente un ANOVA de un factor para comparar las medias de las métricas principales entre librerías. Debido al tamaño reducido de la muestra experimental, se incorporó adicionalmente la prueba no paramétrica de Kruskal–Wallis como método complementario de validación. Finalmente, considerando que los mismos documentos fueron procesados por todas las librerías, se aplicó un ANOVA bloqueado por documento y una prueba post hoc de Tukey HSD para identificar diferencias específicas entre pares de librerías en las métricas donde se observaron diferencias significativas.

Tabla 1. Diseño experimental aplicado en la comparación de librerías.

Elemento	Descripción
Librerías Python del experimento	PyMuPDF, pdfminer.six, pdfplumber y pypdf
Documentos de prueba para procesamiento	Cuatro documentos de 10, 25, 50 y 100 páginas
Tipo de PDF	PDF digital, sin OCR
Estructura del documento	Tablas, encabezados repetitivos, 2 idiomas, saltos de línea variables
Unidad de análisis	Frase segmentada por punto
Total de ejecuciones	16 ejecuciones experimentales
Entorno de ejecución	Google Colab con Python 3

El procedimiento experimental se estandarizó con un algoritmo general para las cuatro librerías evaluadas. Cada libro PDF fue procesado utilizando una secuencia común de extracción, limpieza, segmentación y cálculo de métricas, este procedimiento hace fácil la comparación de los resultados bajo condiciones homogéneas y reproducibles.

Algoritmo 1. Procedimiento general de extracción textual desde PDF

```
Entrada:
    D = conjunto de documentos PDF
    L = conjunto de librerías de extracción
Salida:
    Archivos CSV de frases y métricas
Inicio
    Para cada librería l en L hacer
        Para cada documento d en D hacer
            iniciar temporizador
            extraer texto del documento usando l
            limpiar y normalizar texto
            segmentar texto en frases
            calcular métricas:
                - total de frases
                - frases válidas
                - frases duplicadas
                - total de palabras
                - tiempo de extracción
            guardar frases extraídas
            guardar métricas del procesamiento
        Fin Para
    Fin Para
Fin
```

RESULTADOS

Entre Los resultados consolidados permitieron comparar el comportamiento de las cuatro librerías en términos de velocidad, volumen textual recuperado y porcentaje de frases. Ninguna de las ejecuciones registró páginas con error, por lo que todas las herramientas lograron procesar la totalidad de los documentos evaluados.

Tabla 2. Promedios por librería en las métricas principales.

Librería	Tiempo promedio (s)	Total de frases promedio	Frases válidas (%)
PyMuPDF	0.2360	3267.500	58.105
pypdf	3.106	3267.500	58.105
pdfminer.six	7.479	3297.500	59.175
pdfplumber	9.615	3301.000	57.230

Fuente: Elaboración propia a partir de los resultados experimentales.

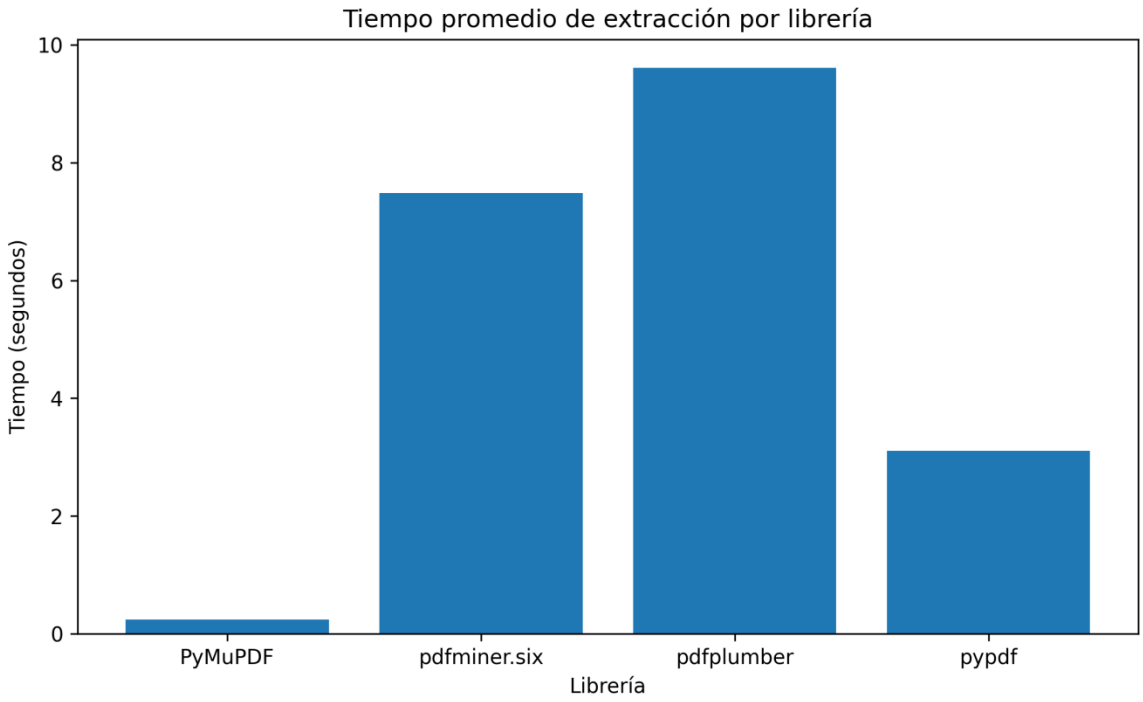

Figura 1. Tiempo promedio de extracción por librería.

Fuente: Elaboración propia.

La Figura 1 muestra diferencias importantes en los tiempos de extracción entre las librerías. En la ejecución de las pruebas experimentales, PyMuPDF fue consistente y alcanzó los menores tiempos de procesamiento, con un promedio de 0.24 segundos por documento. pypdf alcanzó tiempos mayores, aunque todavía considerablemente inferiores a los registrados por pdfminer.six y pdfplumber. En los libros de mayor tamaño, especialmente el archivo de 100 páginas, la diferencia entre librerías se hizo más evidente. Este comportamiento muestra que PyMuPDF tiene una arquitectura más optimizada para tareas de lectura y extracción rápida de texto, estos resultados hacen que PyMuPdf sea el más adecuado cunado el rendimiento y la velocidad de extracción sean la prioridad.

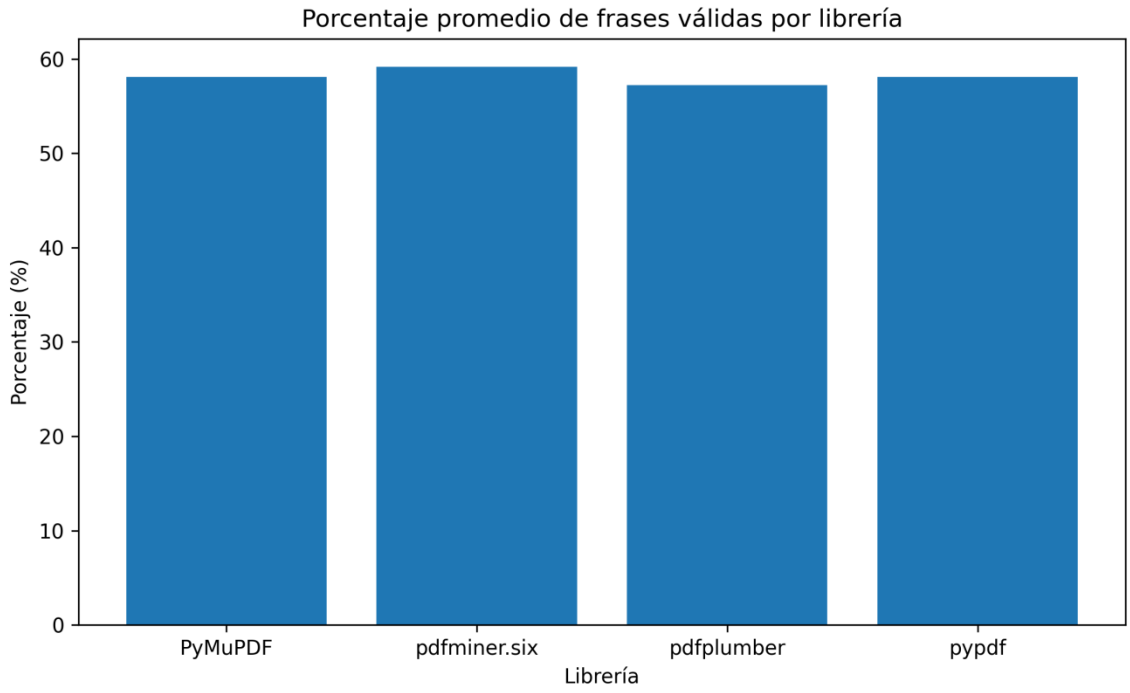

Figura 2. Porcentaje promedio de frases válidas por librería.

Fuente: Elaboración propia.

El porcentaje promedio de frases válidas es representado en la figura 2, pdfminer.six registró el mayor valor de 59.18 %, seguido por PyMuPDF y pypdf de 58.10 %. pdfplumber registró 57.23 %. Las diferencias fueron pequeñas, esto sugiere que, las cuatro librerías lograron mantener niveles de calidad básica bastante cercanos bajo los criterios automáticos de validación definidos en el experimento. Esto sugiere que las diferencias entre herramientas se relacionan más con la forma en que procesan y segmentan el contenido del PDF que con una pérdida significativa de información textual.

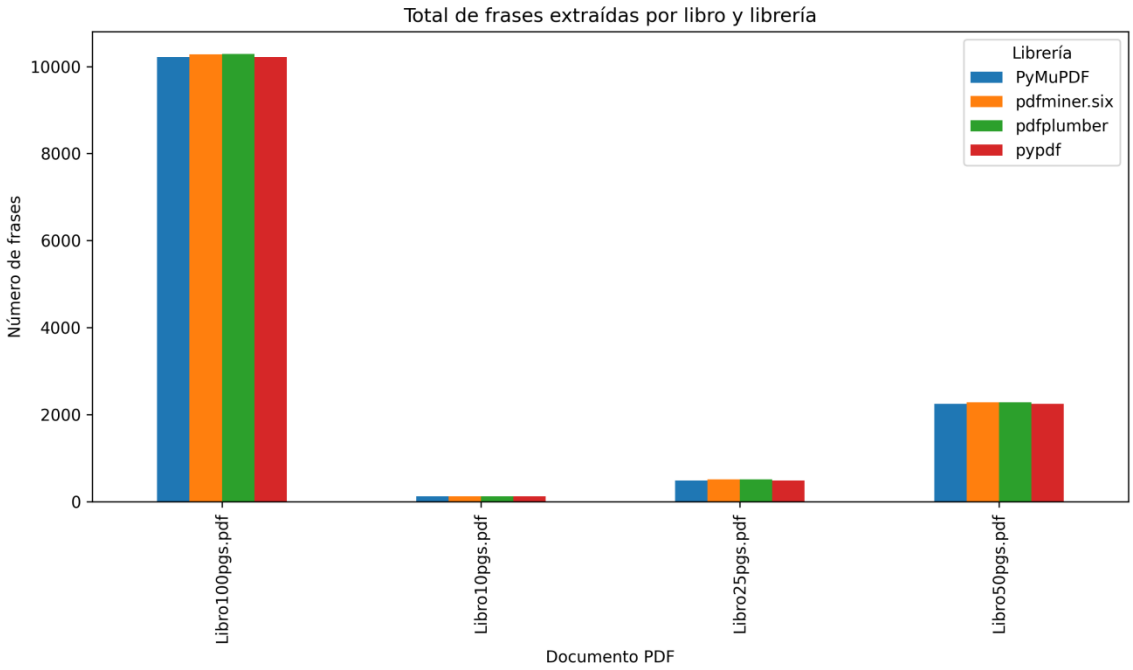

Figura 3. Total, de frases extraídas por libro y librería.

El volumen de frases extraídas se incrementó con el tamaño del documento, así como lo presenta la figura 3; pdfminer.six y pdfplumber tendieron a recuperar un número ligeramente mayor de frases en comparación con PyMuPDF y pypdf, ese aumento en el volumen textual no necesariamente implicó una mejora en el porcentaje de frases válidas extraidas.

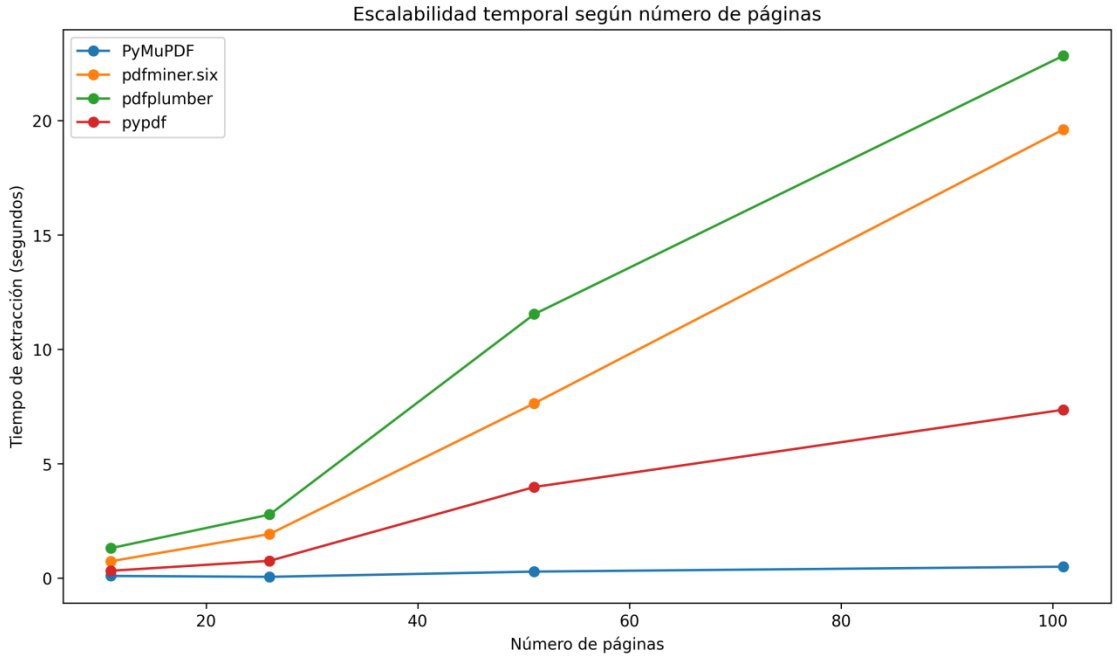

Figura 4. Escalabilidad temporal según número de páginas.

La escalabilidad temporal de las librerías según el número de páginas, es representado en la figura 4. PyMuPDF mantuvo una curva de crecimiento muy baja incluso en el documento de mayor tamaño. En cambio, pdfminer.six y pdfplumber elevaron de forma más clara sus tiempos de procesamiento conforme crecía el tamaño del documento. Este comportamiento indica que el costo computacional de estas librerías es más sensible al tamaño del documento, especialmente en archivos con estructuras complejas y gran cantidad de contenido textual.

Análisis Estadístico

Se aplicaron pruebas estadísticas paramétricas y no paramétricas a las métricas del experimento, para verificar si las diferencias observadas entre las librerías eran realmente significativas y no producto de variaciones aleatorias. Se utilizó ANOVA de un factor para comparar las medias obtenidas por cada librería, así mismo la prueba de Kruskal–Wallis se la utilizó como alternativa no paramétrica por el tamaño reducido de la muestra experimental.

Como los mismos documentos PDF para el experimento fueron procesados por todas las herramientas evaluadas, se empleó un ANOVA bloqueado por documento para verificar el efecto que genera el tamaño y la estructura de cada archivo sobre los resultados obtenidos. Este enfoque permitió una comparación más consistente entre librerías y reducir la influencia de las diferencias propias de cada documento procesado.

Tabla 3. ANOVA de un factor y prueba de Kruskal-Wallis por métrica.

Métrica	ANOVA F	p ANOVA	Kruskal H	p Kruskal-Wallis
Tiempo de extracción	1.562	0.2497	8.890	0.0308
Total de frases	0.0001	1.000	0.3288	0.9545
Porcentaje de frases válidas	0.0049	0.9995	0.3550	0.9494
Porcentaje de duplicados	0.0090	0.9988	1.620	0.6549

Nota: valores p menores a 0.05 se consideran estadísticamente significativos.

El ANOVA de un factor no reveló significancia estadística en las métricas comparadas. Contrariamente, la prueba no paramétrica de Kruskal–Wallis detectó diferencias significativas respecto al tiempo de extracción (H = 8.8897; p = 0.0308), hallazgo que se alinea con los patrones observados en la Figura 1, particularmente en documentos extensos. No obstante, la interpretación requiere precisión: el reducido corpus experimental (limitado número de archivos PDF) restringe la capacidad de generalización estadística, siendo esto una limitación inherente al diseño muestral empleado.

Tabla 4. ANOVA bloqueado por documento.

Métrica	F librería	p librería	F documento	p documento
Tiempo de extracción	3.423	0.0660	5.764	0.0176
Total de frases	5.179	0.0237	343810.279	0.0000
Porcentaje de frases válidas	0.7778	0.5353	635.690	0.0000
Porcentaje de duplicados	8.226	0.0060	3654.029	0.0000
Logaritmo del tiempo	78.045	0.0000	51.612	0.0000

Nota: el bloqueo por documento controla la variabilidad asociada al tamaño del PDF.

El modelo ANOVA con bloqueo por documento reveló que la estructura y magnitud de los archivos PDF ejercieron una influencia estadísticamente significativa sobre la totalidad de métricas evaluadas, confirmando así que el tamaño documental constituyó una variable de control relevante. En cuanto a los efectos atribuibles a cada librería, se identificaron diferencias significativas tanto en el volumen total de oraciones ($p = 0.0237$) como en la tasa de duplicación textual ($p = 0.0060$). La métrica temporal sin transformación aproximó, pero no alcanzó el nivel de significación convencional ($p = 0.0660$).

Al aplicar logaritmos a los tiempos de extracción, el efecto de la herramienta mostró relevancia estadística ($F = 78.0451$; $p < 0.001$), evidenciando que la diferencia del rendimiento entre librerías permanece en un nivel descriptivo.

Tabla 5. Prueba post hoc de Tukey HSD para logaritmo del tiempo.

Grupo 1	Grupo 2	Diferencia media	p ajustado	Límite inferior	Límite superior	Diferencia significativa
PyMuPDF	pdfminer.six	3.116	0.0251	0.3672	5.865	True
PyMuPDF	pdfplumber	3.492	0.0123	0.7429	6.241	True
PyMuPDF	pypdf	2.268	0.1200	-0.4811	5.017	False
pdfminer.six	pdfplumber	0.3757	0.9764	-2.373	3.125	False
pdfminer.six	pypdf	-0.8483	0.7969	-3.597	1.901	False
pdfplumber	pypdf	-1.224	0.5674	-3.973	1.525	False

Nota: la transformación logarítmica fue usada para estabilizar la variabilidad del tiempo de extracción.

La comparación pareada mediante la prueba de Tukey HSD aplicada a la métrica transformada logarítmicamente puso en evidencia diferencias estadísticamente robustas: PyMuPDF se diferencia de pdfminer.six ($p = 0.0251$) y de pdfplumber ($p = 0.0123$). En cambio, la comparación entre PyMuPDF y pypdf no alcanzó criterios de significación formal, aunque los valores descriptivos favorecen sistemáticamente a PyMuPDF. Con respecto a la comparación entre pdfminer.six y pdfplumber, no se detectaron divergencias que alcanzaran el umbral de significación estadística. Estos hallazgos respaldan de manera consistente de que la librería PyMuPDF es la herramienta con desempeño temporal superior, característica que se mantiene independientemente de la extensión de los archivos procesados en el conjunto evaluado.

DISCUSIÓN

Cuando se analizan los hallazgos de este trabajo, resulta evidente que no se puede elegir una librería de extracción de PDF simplemente mirando cuánto texto recupera. En este experimento, las herramientas que extajeron más frases no eran las más veloces ni las que producían menos texto duplicado. Esto sugiere algo importante: el rendimiento de cada librería está estrechamente vinculado a para qué específicamente la vamos a usar. Si necesitamos extraer masivamente, crear un corpus, preservar la estructura o preparar datos para procesamiento posterior de lenguaje natural, nuestras prioridades serán completamente distintas.

Con respecto a PyMuPDF, los resultados muestran claramente que es mucho más rápida que sus competidoras. Lo interesante es que esta superioridad no solo se vio en los números totales, sino que el análisis estadístico (especialmente al transformar logarítmicamente los tiempos) lo confirmó de manera sólida. ¿A qué se debe tanta velocidad? PyMuPDF funciona como un intermediario (wrapper) que accede a MuPDF, una librería escrita en C que está específicamente optimizada para leer y procesar PDFs de forma eficiente (5). Esto significa que gran parte del trabajo pesado no se realiza en Python, sino en código compilado mucho más rápido. Por eso, cuando tienes miles de documentos para procesar, PyMuPDF te ahorra horas de computación. Para aplicaciones industriales donde el tiempo importa, esta característica es definitivamente un punto a favor.

Ahora bien, pdfminer.six nos presenta otra historia. Esta librería logró el porcentaje más alto de frases válidas, aunque la diferencia con las otras no sea dramática. Lo que vimos es que trabaja de forma más minuciosa y cautelosa. Esto tiene sentido si consideramos que está programada casi completamente en Python, lo que exige más trabajo de la CPU pero permite un análisis más profundo. Mientras PyMuPDF prioriza la velocidad, pdfminer.six se toma su tiempo para entender mejor la estructura interna del PDF, analizar posiciones de caracteres, tipografía y otras características internas que ayudan a recuperar texto con mayor precisión. Para trabajos académicos o cuando hay que construir un corpus lingüístico donde la exactitud inicial del texto es más importante que la velocidad, pdfminer.six es una opción que vale la pena considerar.

En cuanto a pdfplumber, el dato más notable fue su menor tasa de duplicación. Esto probablemente está relacionado con su enfoque: fue diseñada específicamente para entender la disposición visual y estructural de los documentos, manejando tablas, líneas y bloques de texto de forma inteligente (7). La librería se construye encima de pdfminer.six y agrega capas adicionales de procesamiento para interpretar mejor cómo está distribuido visualmente el contenido. El problema es que toda esa sofisticación tiene un costo: pdfplumber fue la más lenta en el experimento. Entonces, si se trabaja con documentos complicados, llenos de tablas o formatos visuales complejos, pdfplumber probablemente haga un trabajo mejor. Pero si lo que se necesita es la extracción rápidamente de cientos de documentos simples, pdfplumber no es la mejor opción.

Otro resultado que llamó la atención fue cuán importante resultó ser la característica del documento en sí. El análisis estadístico de varianza (ANOVA con bloqueo) mostró que el tamaño y la forma del PDF cambian significativamente cómo se comportan todas las librerías. Esto nos dice algo crucial: no podemos confiar en una comparación hecha con solo uno o dos documentos. Los cuatro archivos que usamos en el experimento tenían características que los hacen representativos de realidades complejas: tablas intrincadas, contenido que mezclaba shuar y español, encabezados que se repetían constantemente, y estructuras tabuladas difíciles de procesar. Cada una de estas características afecta de manera diferente a cada herramienta. Por eso nuestro diseño incluyó documentos de distintos tamaños.

Desde la metodología, se considera a este experimento como un trabajo que hace una contribución útil, en lugar de simplemente describir los resultados, se usó métricas consistentes, se procesó documentos de tamaños variados y se aplicaron pruebas estadísticas rigurosas. Eso

permitió pasar de un análisis superficial a una evaluación con fundamento empírico real. Aunque el conjunto de documentos es todavía relativamente pequeño, los datos obtenidos abren una puerta a futuras investigaciones en digitalización de documentos, extracción de información y construcción de corpus para lenguas como el shuar.

Limitaciones

El estudio se limitó únicamente a cuatro documentos PDF digitales, lo cual no es una muestra significativa, pese a que los documentos fueron seleccionados con diferente extensión, la muestra aún no cubre toda la diversidad posible de formatos, estilos de maquetación, tablas, columnas, notas al pie o elementos gráficos.

No se evaluaron documentos escaneados ni se aplicó OCR. Esta decisión fue coherente con el objetivo del experimento, pero impide generalizar los resultados a documentos basados en imagen.

La segmentación de frases se realizó utilizando el punto como delimitador. Esta regla favorece la homogeneidad experimental, pero puede separar de manera incorrecta oraciones que posean abreviaturas, numeraciones o expresiones con puntuación interna.

La validez de frases se calculó mediante criterios heurísticos de longitud y número de palabras, razón por la cual no representa una validación lingüística profunda ni una evaluación semántica del contenido.

Trabajos Futuros

Como primera línea de trabajo futuro, se propone el incremento del tamaño documental y la diversidad del corpus experimental incorporando documentos con mayor número de páginas, con estructuras multicolumna, tablas complejas, diagramas y formatos académicos heterogéneos. Esto permitiría evaluar con mayor profundidad la escalabilidad de las librerías y analizar el comportamiento frente a documentos con estructuras complejas y cargas textuales superiores.

Como segunda línea de investigación, se plantea integrar técnicas de reconocimiento óptico de caracteres (OCR) para analizar el desempeño de las librerías sobre documentos PDF escaneados o basados en imagen. La incorporación de motores OCR permitiría comparar no solo la capacidad de extracción textual, sino también la precisión del reconocimiento de caracteres y el impacto del ruido visual en la calidad del texto obtenido.

CONCLUSIONES

Este trabajo de investigación demuestra que existen diferencias importantes entre las librerías Python utilizadas para la extracción automática de texto desde documentos PDF digitales. PyMuPDF presentó el mejor desempeño con respecto al tiempo en todas las ejecuciones experimentales, alcanzando tiempos más cortos frente a pdfminer.six, pdfplumber y pypdf. Este

comportamiento se confirma mediante el análisis estadístico realizado, particularmente con el ANOVA bloqueado y la prueba post hoc de Tukey HSD aplicada sobre el logaritmo del tiempo de extracción. Los hallazgos evidencian que PyMuPDF es una alternativa adecuada para escenarios donde se requiere extracción masiva de textos, construcción rápida de corpus o sistemas que demandan bajo consumo de recursos computacionales.

En referencia a la calidad de extracción textual, pdfminer.six obtuvo el mayor porcentaje promedio de frases válidas, mientras que pdfplumber registró el menor porcentaje de duplicados. Estos resultados confirman que la selección de una librería no debe estar basado solo en la velocidad de procesamiento, sino también en el equilibrio entre recuperación textual, segmentación consistente y reducción de redundancia. El análisis estadístico permitió identificar que el tamaño y la estructura del documento influyen significativamente en el comportamiento de las librerías, lo que confirma que es necesario evaluar múltiples tipos de PDF antes de seleccionar una herramienta para proyectos de procesamiento documental.

La metodología implementada demostró ser útil para comparar herramientas de extracción de texto bajo condiciones homogéneas y reproducibles. La combinación de métricas cuantitativas, análisis estadístico y evaluación experimental sobre documentos de diferente tamaño permitió generar evidencia objetiva sobre el desempeño de cada librería. Los resultados obtenidos constituyen una base relevante para futuras investigaciones orientadas a construcción de corpus bilingües, minería de texto, recuperación de información y sistemas basados en inteligencia artificial.

CONTRIBUCIONES DE AUTOR

Se diseñó y desarrolló la totalidad del experimento comparativo, incluyendo la implementación de los algoritmos de extracción textual mediante las librerías PyMuPDF, pdfminer.six, pdfplumber y pypdf, así como la definición de las métricas de evaluación, el procesamiento de los documentos PDF y el análisis estadístico de los resultados. Con el propósito de garantizar la reproducibilidad y transparencia de la investigación, todos los recursos utilizados en el estudio, incluidos los notebooks desarrollados, los algoritmos de extracción y los archivos de resultados, permanecen disponibles en un repositorio público de GitHub para su validación, reutilización y comparación en futuras investigaciones relacionadas con procesamiento documental y construcción de corpus textuales (*https://github.com/AngelFlores3230/pdf_extract.git*).

REFERENCIAS BIBLIOGRÁFICAS

1. Adobe Systems Incorporated. PDF Reference: Adobe Portable Document Format Version 1.7. 6th ed. San Jose: Adobe Systems; 2006.
2. International Organization for Standardization. ISO 32000-2:2020 Document management - Portable document format - Part 2: PDF 2.0. Geneva: ISO; 2020.
3. Jurafsky D, Martin JH. Speech and Language Processing. 3rd draft ed. Stanford: Stanford University; 2024.

4. Manning CD, Raghavan P, Schutze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press; 2008.
5. PyMuPDF Documentation. PyMuPDF: Python bindings for MuPDF. Available from: https://pymupdf.readthedocs.io/
6. pdfminer.six Documentation. pdfminer.six: Python PDF parser and analyzer. Available from: https://pdfminersix.readthedocs.io/
7. Singer JS. pdfplumber Documentation. Available from: https://github.com/jsvine/pdfplumber
8. pypdf Documentation. pypdf: A pure-python PDF library. Available from: https://pypdf.readthedocs.io/
9. Adhikari N, Agarwal S. A comparative study of PDF parsing tools across diverse document categories. arXiv. 2024.
10. Ferguson N, Pennington J, Beghian N, Mohan A, Kiela D, Agrawal S, et al. ExtractBench: A benchmark and evaluation methodology for complex structured extraction. arXiv. 2026.
11. Lee T, Kim G, Ahn H, Jeong J, Jeong M, Song J. Integrating OCR and LLMs for enhanced document digitization in ERP systems. IEEE; 2024.
12. Lakatos R, Urban EK, Szabo Z, Pozsga J, Csernai E, Hajdu A. Designing prompts and creating cleaned scientific text for retrieval augmented generation. IEEE; 2024.
13. Naik YGR, B S, Amith GK. A review on text extraction techniques for degraded historical document images. IEEE; 2024.
14. Chelliah BJ, Hariharan M, Prakash A, Manoharan B, Senthilselvi A. Harnessing T5 large language model for enhanced PDF text comprehension and Q&A generation. IEEE; 2024.
15. Yang H, Wei Z, Cheng Z, Zou M, Chen F, Luo W, et al. Research on performance comparison of different Python PDF parsing libraries in analyzing protection setting lists. IEEE; 2025.
16. Sharmila SP, Tiwari A. PDFInspect: A unified feature extraction framework for malicious document detection. IEEE; 2026.
17. Böschen I. Evaluation of the extraction of methodological study characteristics with JATSdecoder. Scientific Reports. 2023.
18. Sasirekha D, Chandra E. Enhanced techniques for PDF image segmentation and text extraction. IEEE; 2012.
19. Zaryab MA, Ng CR. Optical character recognition for medical records digitization with deep learning. IEEE; 2023.
20. Bai L, Mulvenna M, Wang Z, Bond R. Clinical entity extraction: comparison between MetaMap, cTAKES, CLAMP and Amazon Comprehend Medical. IEEE; 2021.
21. Litvak IL, Kostin A, Lashkin F, Maksiyan T, Lagutin S. Comparison of unsupervised metrics for evaluating judicial decision extraction. arXiv. 2025.
22. Ramadhan G, Mulyana DI, Adrianto S. Optimization of Tesseract OCR for automatic text extraction on Indonesian ID cards. Indonesian Journal of Systems Engineering and Computer Science. 2025.
23. Ma Z, Huang F, Zhao L, Guo F, Zhai G, Min X. DocIQ: A benchmark dataset and feature fusion network for document image quality assessment. arXiv. 2025.
24. Wilhelmi L, Bruns C, Schumann M. Enhancing the extraction of GHG emission-reduction targets from sustainability reports using vision language models. MAKE. 2026.