Documentation Intelligence : Extracting Content from PDFs Using Python Libraries

Task Overview :

The task focuses on extracting content from PDF files using Python libraries. We used three different libraries **PyPDF2**, **pdfplumber**, and **PyMuPDF** each serving a different purpose to handle various types of PDFs and the content within them.

Libraries Used :

1. PyPDF2

- **Purpose**: PyPDF2 is a lightweight library that helps in extracting basic text from PDFs. It's useful for PDFs with straightforward content and no complex formatting.

2. pdfplumber

- **Purpose**: pdfplumber is a more advanced library that's excellent for handling PDFs with complex layouts, such as those containing tables. It preserves the structure of the document better than PyPDF2, allowing you to extract text and even tables.

3. PyMuPDF (fitz)

- **Purpose**: PyMuPDF is a powerful library that can extract text, images, and metadata from PDFs. It's particularly useful for complex documents and allows you to access a wide range of content, including images and file properties.

Text Extraction in Engineering+Working+Drawing+Basics :

To extract text from PDF files, search for specific keywords, and save the results in an Excel file. It uses two Python libraries: **pdfplumber** to extract text from the PDFs, and **pandas** to organize and save the data in Excel.

How it works :

1. **Extract Text**: The code reads the content of the PDF and extracts all the text from its pages.
2. **Search Keywords**: It searches for specific keywords (like "Codes and Standards" and "Material or Parts List") in the text. When it finds a keyword, it grabs the surrounding text to give more context.
3. **Save to Excel**: The found results, along with the surrounding text, are saved into an Excel file. The file also includes the name of the original PDF file for reference.
4. **Download**: Once the results are saved, the file can be downloaded to your computer.

The code also includes a feature to upload PDF files through Google Colab. After uploading, it processes the file, looks for the keywords, and saves the results in an Excel file. If no matches are found, it will notify you.

Steps:

1. **Upload PDF**: The user uploads a PDF file through Google Colab.
2. **Extract Text**: The code reads the content of the PDF and extracts all the text from its pages.
3. **Search for Keywords**: The code searches for specific keywords (e.g., "Codes and Standards" or "Material or Parts List") within the extracted text. When a keyword is found, it captures the surrounding text for context.
4. **Save to Excel**: The extracted results, including the found keywords and their surrounding text, are saved in an Excel file. The Excel file also includes the name of the original PDF for reference.
5. **Download**: Once the results are saved in the Excel file, the user can download the file to their computer.

Text Extraction In Table Format Autodesk Inventor Practice Part Drawings :

To extracts specific content, such as "FILE NAME" and "DESCRIPTION," from a PDF file and saves it in an Excel file. It uses two Python libraries: **PyPDF2** to extract text from the PDFs, and **pandas** to organize and save the data into Excel.

How it works:

1. **Extract Text**: The code reads the content of the PDF and extracts the text from the pages you specify.
2. **Find File Names and Descriptions**: It searches for patterns that match "FILE NAME" (ending in .ipt) and the corresponding "DESCRIPTION" using regular expressions.
3. **Save to Excel**: The extracted "FILE NAME" and "DESCRIPTION" pairs are saved into an Excel file. The Excel file also includes the name of the original PDF for reference.
4. **Download**: After saving the data to Excel, the file is made available for download.

The code also allows you to upload a PDF file through Google Colab. After uploading, it processes the PDF, extracts the relevant data, and saves it in an Excel file. If no matches are found, it will notify you.

Steps:

1. **Upload PDF**: The user uploads a PDF file through Google Colab.
2. **Extract Text**: The code reads the content of the specified pages of the PDF and extracts the text.

3. **Find Specific Content**: The code searches for specific patterns like "FILE NAME" (ending with .ipt) and the corresponding "DESCRIPTION" using regular expressions.
4. **Save to Excel**: The extracted "FILE NAME" and "DESCRIPTION" pairs are saved in an Excel file.
5. **Download**: The user can download the Excel file containing the extracted data.

Extracting the width form the specific page number in Autodesk Inventor Practice Part Drawings :

To extracts specific width values from a particular page of a PDF file and saves the results in an Excel file. It uses the **PyMuPDF (fitz)** library to read the PDF and **pandas** to organize and save the data into Excel.

How it works:

1. **Extract Text**: The code reads the content of the specified page of the PDF and extracts the text.
2. **Find Width Values**: It searches for occurrences of the word "width" and tries to capture the numeric value that follows it. If no width value is found, it extracts the largest numeric value from the page.
3. **Save to Excel**: The extracted width (or the largest number) is saved into an Excel file along with the page number.
4. **Download**: After saving the data to Excel, the file is made available for download.

   The code also allows you to upload a PDF file through Google Colab. After uploading, it processes the file, extracts the relevant data, and saves it in an Excel file. If no width values are found, it will notify you.

Steps:

1. **Upload PDF**: The user uploads a PDF file.
2. **Specify Page Number**: You set the page number from which to extract the width data.
3. **Extract Data**: The code processes the page, extracts the width or a related value, and saves it to Excel.
4. **Download**: The user can download the Excel file containing the extracted data

Extracting Widths from Autodesk Inventor Practice Part Drawings for All Figures in the PDF

This code extracts specific width values from each page of a PDF file and saves the results in an Excel file. It uses the **PyMuPDF** library to read the PDF and **pandas** to organize and save the data into Excel.

How it works:

1. **Extract Text**: The code reads the content of each page of the PDF and extracts the text.
2. **Find Width Values**: It searches for occurrences of the word "width" and tries to capture the numeric value that follows it. If no width value is found, it extracts the largest numeric value from the page.
3. **Save to Excel**: The extracted width (or the largest number) is saved into an Excel file along with the corresponding page number.
4. **Download**: After saving the data to Excel, the file is made available for download.

The code also allows you to upload a PDF file through Google Colab. After uploading, it processes the file, extracts the relevant data, and saves it in an Excel file. If no width values are found, it will notify you.

Steps:

1. **Upload PDF**: The user uploads the PDF file.
2. **Extract Width Values**: The code processes each page, extracting any width values associated with the word "width."
3. **Save to Excel**: The width data is saved to an Excel file.
4. **Download**: The user can download the Excel file containing the extracted data.

Why These Libraries?

- **PyPDF2** is lightweight and simple, making it a good choice for basic PDFs that don't require handling of complex layouts or embedded media.
- **pdfplumber** shines when dealing with PDFs that include tables or unusual layouts, as it retains more of the document's structure.
- **PyMuPDF** is the most versatile of the three. It can handle text extraction, images, tables, and metadata, making it a great choice for more complex documents.

Conclusion

This task demonstrated how different Python libraries can be used together to extract a variety of content from PDF files. By using the right library for the job, we were able to efficiently extract the data we needed from a wide range of PDF formats.

Each library serves its purpose:

- **PyPDF2** for simple text extraction,
- **pdfplumber** for structured data like tables,
- **PyMuPDF** for everything else, including images and metadata.