Code Documentation for PDF to Text Extraction Script

Imports

python

Copy code

```
import PyPDF2

import os
```

- PyPDF2: A Python library that allows you to read and manipulate PDF files. In this script, it is used to extract text from the pages of a PDF.
- os: A built-in Python module that provides functions to interact with the operating system, such as handling file paths and directories.

Checking and Creating the 'temp' Directory

python

Copy code

```
if not os.path.isdir("temp"):

    os.mkdir("temp")
```

- This code checks whether a directory named "temp" exists in the current working directory using os.path.isdir().
- If the directory doesn't exist, os.mkdir() creates it. The "temp" directory will be used to store the extracted text files.

Variables for File Paths

python

Copy code

```
txtpath = ""

pdfpath = ""
```

- txtpath: This variable will store the output path where the extracted text from the PDF will be saved as a .txt file.
- pdfpath: This variable will store the path of the input PDF file from which the text will be extracted.

Taking User Input for PDF and Output File Paths

python

Copy code

```
pdfpath = input("Enter the name of your pdf file - please use backslash when typing in directory path: ")
```

```
txtpath = input("Enter the name of your txt file - please use backslash when typing in directory path: ")
```

- The input() function prompts the user to enter the file paths for the PDF and the output .txt file.
- The user needs to enter the full path of the PDF file (e.g., C:\\Users\\User\\Documents\\file.pdf).
- Similarly, the user is asked to provide the path for the output text file (e.g., C:\\Users\\User\\Documents\\output.txt).

Set Default Base Directory for Text Files

python

Copy code

```
BASEDIR = os.path.realpath("temp")

print(BASEDIR)
```

- os.path.realpath("temp") retrieves the absolute path of the "temp" directory, which is where the text files will be saved if no path is provided by the user.
- BASEDIR will store the absolute path of the "temp" directory.
- print(BASEDIR) displays the absolute path of the "temp" directory on the console.

Default Output Path If Not Provided

python

Copy code

```
if len(txtpath) == 0:

    txtpath = os.path.join(BASEDIR, os.path.basename(os.path.normpath(pdfpath)).replace(".pdf", "") + ".txt")
```

- This block of code checks if the user has provided a path for the output text file (txtpath).

- If no path is provided (i.e., txtpath is an empty string), the script constructs a default output path:
  - os.path.basename(os.path.normpath(pdfpath)) extracts the file name (without path) from the pdfpath.
  - .replace(".pdf", "") removes the .pdf extension from the file name.
  - The .txt extension is added, so the output text file will have the same name as the input PDF file but with a .txt extension.
- The default file path is then stored in txtpath.

Opening the PDF File

python

Copy code

with open(pdfpath, 'rb') as pdfobj:

- open(pdfpath, 'rb') opens the PDF file in read-binary mode ('rb').
- The with statement ensures that the file is properly closed once the block is completed.

Creating a PdfReader Object

python

Copy code

pdfread = PyPDF2.PdfReader(pdfobj)

- PyPDF2.PdfReader(pdfobj) creates a PdfReader object that allows reading and extracting content from the opened PDF file.
- The pdfread object is used to interact with the PDF file and extract data from it.

Get the Number of Pages in the PDF

python

Copy code

x = len(pdfread.pages)

- pdfread.pages is a list of all the pages in the PDF file.
- len(pdfread.pages) returns the total number of pages in the PDF, which is stored in the variable x.

Looping Through Each Page to Extract Text

python

Copy code

```
for i in range(x):
    pageObj = pdfread.pages[i]
```

- A for loop is used to iterate through all the pages in the PDF.
- range(x) generates a range from 0 to the total number of pages (i.e., x).
- pdfread.pages[i] gets the i-th page from the PDF, and this page object is stored in pageObj.

Extracting Text from the Page and Writing It to a Text File

python

Copy code

```
with open(txtpath, 'a+') as f:
    text = pageObj.extract_text()
    f.write(text)
    print(text)
```

- The open(txtpath, 'a+') opens the output text file in append mode ('a+'), allowing the script to add content to the file without overwriting it.
- pageObj.extract_text() extracts the text content from the current page of the PDF and stores it in the text variable.
- f.write(text) writes the extracted text into the text file.
- print(text) outputs the extracted text to the console, providing a preview of the content being written to the output file. You can remove this line if you don't want to display the text.

Full Documentation

Purpose:

This script converts a PDF file to a plain text file. It extracts the text from each page of the input PDF and saves it into an output .txt file.

Features:

- Allows users to specify the input PDF file and output text file paths.
- Automatically creates a "temp" directory if it doesn't exist.
- If no output file path is provided, a default output file path is generated in the "temp" directory.
- Extracts text from all pages of the PDF and saves it to the output text file.

How It Works:

1. The user provides the path for the input PDF file and the output text file.
2. The script reads the PDF file using PyPDF2.PdfReader.
3. It loops through all pages of the PDF and extracts the text content.
4. The extracted text is written to the specified .txt file (or the default file in the "temp" directory).
5. If the user did not provide an output path, a default text file is generated using the PDF file's name.

Dependencies:

- PyPDF2: Install with pip install PyPDF2.

Example Usage:

- **Input**: C:\\Users\\User\\Documents\\file.pdf
- **Output**: C:\\Users\\User\\Documents\\output.txt or default in the "temp" directory.

Important Notes:

- Ensure that the PDF is not password-protected, as PyPDF2 cannot extract text from password-protected PDFs.
- Text extraction might not be perfect for complex PDFs (e.g., scanned images, non-standard fonts).