Extracting Tables From PDFs

Contents

« Previous Next »

Introduction

When an organization publishes data online, it usually releases it as a series of PDFs. Unfortunately, the PDF file format was not designed to hold structured data, which makes extracting tables from PDFs difficult. The good news, though, is that there are several tools available online to make this task easier. The following tutorial describes how to use PyPDF2 and the PDFTables API for Python to extract tabular data from a PDF and download it as a CSV (or xlsx or xml) file.

The files containing all of the code that I use in this tutorial can be found here.

Using PyPDF2 to Extract the Pages With Tables

Suppose that you have a PDF with tables on several pages, but not on every page. Or, suppose that you have multiple PDFs with tables, and you want to write each of those tables to the same CSV. (If none of these scenarios apply to you, and you just want to learn how to extract a table in a PDF to a CSV, feel free to skip to the next section.) Because PDFTables puts everything (not just tables) in your PDF document into the output CSV, to avoid having a lot of junk data in your CSV you'll want to create a separate PDF with just the tables that you want to extract. PyPDF2 is very useful for this.

Installing and Importing PyPDF2

1. Install PyPDF2 with pip, then
2. import it into your Python code.
3. Create an instance of the PdfFileReader Class, which stores information about your PDF (number of pages, text on pages, etc).

Do these steps like so:

```
import PyPDF2

PDFfilename = "example.pdf" #filename of your PDF/directory where your PDF is stored

pfr = PyPDF2.PdfFileReader(open(PDFfilename, "rb")) #PdfFileReader object
```

Next, you'll learn about the methods you can call on this object to extract the pages you need and add them to a new PDF.

Example 1: A PDF With Tables on Several Pages, But Not on Every Page

Suppose that the tables you need are on pages 4 and 8 of your PDF. To extract these pages, you need to

1. Call the .getPage() method, with the page number + 1 as the parameter (pages start at 0), on your PdfFileReader object
2. create a PdfFileWriter object, which will eventually write to your new PDF.
3. add your pages to it.

   Do these steps like so:

   ```
   pg4 = pfr.getPage(3) #extract pg 4
   pg8 = pfr.getPage(7) #extract pg 8

   writer = PyPDF2.PdfFileWriter() #create PdfFileWriter object
   #add pages
   writer.addPage(pg4)
   writer.addPage(pg8)
   ```

4. The last step is to write these pages to your new PDF, like so:
   ```
   NewPDFfilename = "allTables.pdf" #filename of your PDF/directory where you want your new PDF to be
   with open(NewPDFfilename, "wb") as outputStream: #create new PDF
   writer.write(outputStream) #write pages to new PDF
   ```

That's it! Your new PDF should now be in the directory that you specified.

Example 2: Multiple PDFs With Tables

Suppose that each PDF has a table on its second page. In this case, you'll want to

1. iterate through all of your PDFs. If they are all stored in the same folder, you can iterate through them using the os library's .listdir() method, which takes a path as the parameter and lists all files in that path.
2. For each PDF, pull out page 2 and add it to your PdfFileWriter object.
3. Write all of your pages to a single PDF.

Do these steps like so:

```
import os
import PyPDF2

path = "C:/Users/Someone/Documents/allPDFs" #path to folder
#page number to extract, added 1 to reflect counting starting at 0
page = 3
writer = PyPDF2.PdfFileWriter() #create PdfFileWriter object

for pdf in os.listdir(path):
        PDFfilename = path + "/" + pdf
        pfr = PyPDF2.PdfFileReader(open(PDFfilename, "rb")) #PdfFileReader object
        pg2 = pfr.getPage(page) #extract pg 2
        writer.addPage(pg2) #add pg 2

NewPDFfilename = "allTables.pdf" #filename of your PDF/directory where you want your new PDF to be
with open(NewPDFfilename, "wb") as outputStream: #create new PDF
        writer.write(outputStream) #write pages to new PDF
```

Now that you have a PDF with all of the tables you need, you can use PDFTables to write your table data to a CSV.

Writing the Table Data to a CSV With PDFTables

To post a request to the PDFTables website to do the table extraction for you, you must have an API key. You can get one by creating an account on the site for free, and then visiting the API page again. After that, you can

1. send the content of your PDF file to the site, and
2. write the response to a CSV.

The API page has a good example of how to do this, but I will explain it in more detail here:

```
import requests

def pdfToTable(PDFfilename, apiKey, fileExt, downloadDir):

PDFfilename = 'example.pdf'

fileData = (PDFfilename, open(PDFfilename, 'rb')) #"rb" stands for "read bytes"
```

```
files = {'f': fileData}

apiKey = "my_api_key"

fileExt = "csv" #format/file extension of final document

postUrl = "https://pdftables.com/api?key={0}&format={1}".format(apiKey, fileExt)
#the .format puts value of apiKey where {0} is, etc

response = requests.post(postUrl, files=files)

response.raise_for_status() # ensure we notice bad responses

downloadDir = "example.csv" #directory where you want your file downloaded to

with open(downloadDir, "wb") as f:
    f.write(response.content) #write data to csv
```

Here is a wrapper function that includes all of those steps. Feel free to copy it into your code and use it:

```
def pdfToTable(PDFfilename, apiKey, fileExt, downloadDir):
        fileData = (PDFfilename, open(PDFfilename, 'rb'))
        files = {'f': fileData}
        postUrl = "https://pdftables.com/api?key={0}&format={1}".format(apiKey, fileExt)
        response = requests.post(postUrl, files=files)
        response.raise_for_status()
        with open(downloadDir, "wb") as f:
        f.write(response.content)
```

That’s it! You should now have a CSV with your tabular data in the directory that you specified. Happy scraping!

« Previous Next »

Related Tutorials

Using Python to Convert PDFs to Text Files

Scraping PDFs/Text Files in Python Using Regular Expressions

Writing Data to a CSV With Python