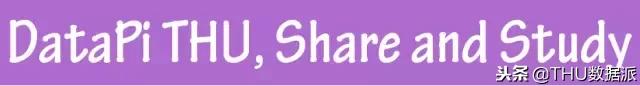
<https://www.toutiao.com/a6628693797901435396/?tt_from=weixin&utm_campaign=client_share&wxshare_count=1&timestamp=1543560212&app=news_article&utm_source=weixin&iid=52365825080&utm_medium=toutiao_android&group_id=6628693797901435396>

**手把手教你如何用Python从PDF文件中导出数据（附链接）**

原创 THU数据派 2018-11-28 22:01:00



作者：Mike Driscoll

翻译：季洋

校对：丁楠雅

本文约**4000字**，建议阅读10分钟。

本文介绍了在提取出想要的数据之后，如何将数据导出成其他格式的方法。

有很多时候你会想用Python从PDF中提取数据，然后将其导出成其他格式。不幸的是，并没有多少Python包可以很好的执行这部分工作。在这篇贴子中，我们将探讨多个不同的Python包，并学习如何从PDF中提取某些图片。尽管在Python中没有一个完整的解决方案，你还是应该能够运用这里的技能开始上手。提取出想要的数据之后，我们还将研究如何将数据导出成其他格式。

让我们从如何提取文本开始学起！

**使用PDFMiner提取文本**

最被大家所熟知的可能是一个叫做PDFMiner的包。PDFMiner包大约从Python 2.4版本就存在了。它的主要目的是从PDF中提取文本。实际上，PDFMiner可以告诉你某文本在分页上具体的位置和字体信息。对于Python 2.4到2.7版本，你可以参考以下网站来了解PDFMiner的更多信息：

GitHub – https://github.com/euske/pdfminerPyPI – https://pypi.python.org/pypi/pdfminer/Webpage – https://euske.github.io/pdfminer/

PDFMiner是不兼容于Python 3的。幸运的是，PDFMiner家族的一个分支PDFMiner.six在Python 3上完全能胜任同样的功能。

你可以在以下网站上找到：

https://github.com/pdfminer/pdfminer.six

关于PDFMiner的安装说明已经比较过时了。其实你可以用pip命令来安装它：

手把手教你如何用Python从PDF文件中导出数据（附链接）

如果你要在Python 3上安装PDFMiner（这也许就是你现在正在做的），你需要这样安装：

手把手教你如何用Python从PDF文件中导出数据（附链接）

PDFMiner的相关文档很少。你将很大可能地需要使用Google和Stack Overflow两个查询工具来弄清楚如何在这篇贴子的涵盖内容之外有效地使用PDFMiner。

**提取所有文本**

有时你会想要提取PDF文件中的所有文本。PDFMiner包提供了一些不同的方法使你能够做到这一点。我们先来探讨一些编程的方法。让我们试着从一个国税局W9表单中读取所有的文本。

你可以从这里得到表单副本：

https://www.irs.gov/pub/irs-pdf/fw9.pdf

保存完这个PDF文件之后，你可以参考以下代码：



当你直接使用PDFMiner包时，往往会有点繁琐。这里，我们从PDFMiner的不同模块中引入多个不同的类。由于这些类都没有文档说明，也没有实现其文档字符串属性，我将不会深入讲解它们做了什么。如果你真的好奇的话，尽管可以深入地研究它们的源代码。无论如何，我认为我们可以大致照以上代码行事。

我们做的第一件事就是创建一个资源管理器的实例。然后通过Python的输入输出（io）模块创建一个似文件对象。如果你使用的是Python 2，你应该使用StringIO模块。接下来的步骤是创建一个转换器。在这个例子里，我们选择使用TextConverter，如果你想要的话，你还可以使用HTMLConverter或XMLConverter。最后，我们创建一个PDF解释器对象，携带着我们的资源管理器和转换器对象，来提取文本。

最后一步是打开PDF文件并且循环遍历每一页。结尾部分，我们抓取所有的文本，关闭不同的信息处理器，同时打印文本到标准输出(stdout)。

**按页提取文本**

通常我们并不需要从一个多页文档中抓取所有的文本。你一般会想要处理文档的某些部分。那么，让我们改写代码以便它提取文本呈分页的格式。这将允许我们在检查文本时，一次一页地进行：



在这个例子中，我们创建了一个生成器函数按页**生成（yield）**了文本。extract\_text函数按页打印出文本。此处我们可以加入一些分析逻辑来得到我们想要的分析结果。或者我们可以仅是将文本（或HTML或XML）存入不同的文件中以便分析。

你可能注意到这些文本没有按你期望的顺序排列。因此你需要思考一些方法来分析出你感兴趣的文本。

PDFMiner的好处就是你可以很方便地按文本、HTML或XML格式来“导出”PDF文件。

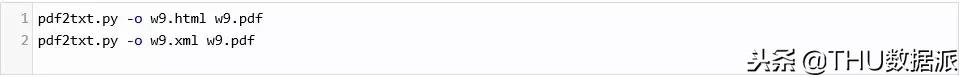
你也可以使用PDFMiner的命令行工具，**pdf2txt.py**和**dumppdf.py**，来为你执行导出工作。如果你不想试图自己弄明白PDFMiner。根据**pdf2txt.py** 的源代码，它可以被用来导出PDF成纯文本、HTML、XML或“标签”格式。

**通过pdf2txt.py导出文本**

**伴随着PDFMiner一起的pdf2txt.py命令行工具会从一个PDF文件中提取文本并且默认将其打印至标准输出（stdout）。它不能识别文字图片，就像PDFMiner不支持光学字符识别（OCR）一样。让我们尝试用最简单的方法来使用它，那就是仅仅传递给它一个PDF文件的路径。我们会使用w9.pdf文件。**打开一个终端并且定位到你存放PDF文件的位置，或修改一下命令指向待处理文件：

手把手教你如何用Python从PDF文件中导出数据（附链接）

如果你执行这条命令，它将打印出所有的文本到标准输出（stdout）。你也可以使pdf2txt.py 将文本写入文件成文本、HTML、XML或“带标签PDF”格式。XML格式将给出关于PDF的大部分信息，因为它包含了每一个字母在文件中的位置以及字体信息。不推荐使用HTML格式，因为pdf2txt生成的标记往往会很丑。以下是教你如何生成不同格式输出的方法：



第一条命令将创建一个HTML文件，而第二条将创建一个XML文件。

最终的结果看上去有点怪，但是它并不太糟糕。XML格式的输出极其冗长，因此我不能将它完整地在这里重现，以下是一小段示例：



**使用Slate提取文本**

Tim McNamara觉得PDFMiner使用起来太过愚蠢和费力，因此他写了一个围绕它的包装器叫做slate，以使它更简单地从PDF中提取文本。不幸的是，它和Python 3不兼容。如果你想试用，你可能需要easy\_install以便于安装distribute包，如下：

手把手教你如何用Python从PDF文件中导出数据（附链接）

我不能使用pip 正确安装这个包。然而一旦安装了它，你将能够使用pip来安装slate：

手把手教你如何用Python从PDF文件中导出数据（附链接）

注意最新的版本是0.5.2，而pip未必能拿到这个版本。如果拿不到，那么你可以从GitHub上直接获取slate安装：

手把手教你如何用Python从PDF文件中导出数据（附链接）

现在我们已经准备好写一些代码来从PDF中提取文本了：



正如你能看到的，让slate分析一个PDF文件，你只需要引进slate然后创建一个它的PDF类的实例。PDF类其实是Python内置类list的一个子类，所以它仅是返回了一列/可遍历的文本页。如果PDF文件设有密码，你可以传入一个密码参数。不管怎样，一旦文件被分析，我们只要打印出每一页的文本即可。

我非常喜欢slate，它用起来更简单。不幸的是，这个包也几乎没有什么相关文档。在浏览过它的源码之后，它看起来只支持纯文本提取。

**导出你的数据**

现在我们得到了一些文本，我们会花费一些时间来学习如何导出数据成各种不同的格式。具体来说，我们将学习如何以如下方法导出文本：

* XML
* JSON
* CSV

让我们开始吧！

**导出成XML**

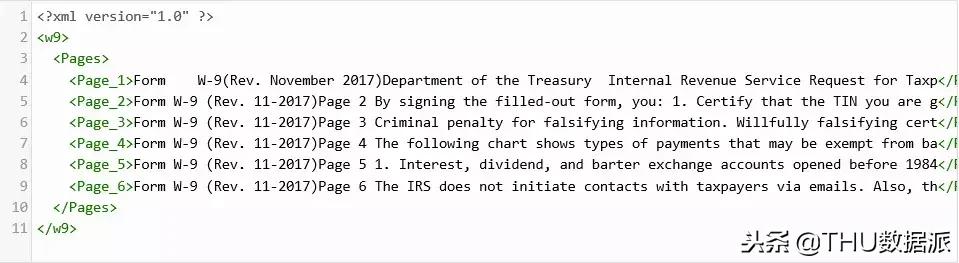
可扩展标记语言（XML）格式是最为人所熟知的输入输出格式之一。它被广泛运用于互联网中的许多不同的事物。正如我们已经在本贴中看到的，PDFMiner也支持XML作为它的输出之一。

话虽这么说，让我们创建我们自己的XML生成工具。如下是一个简单的例子：



这段代码将使用Python内置的XML库，**minidom**和**ElementTree。我们也引入PDFMiner生成器代码以用于每次抓取一页文本。在这个例子中，我们用PDF的文件名创建了我们顶层的元素。然后在它的下层增加了一个页（Pages）**元素。下一步是for循环，在此循环中我们从PDF中提取每一页然后保存想要的信息。此处你可以加入一个特定的分析程序，其中你可以将页分成句子或者单词，从而分析出更有趣的信息。比如，你可能只想得到有某个特定名字或日期/时间戳的句子。你可以运用Python的正则表达式来找出这类东西，或者仅是检查子字符串在句子中的存在。

对于这个例子，我们仅仅是提取了每一页的前100个字符并将其存入一个XML的**子元素（SubElement）**中。接下来的一段代码可以简化成仅是写出XML文件。然而，ElementTree不会做任何事来使得XML易读。它最后看上去有点像压缩的JavaScript似的一块巨型文本。所以我们在写入文件之前使用minidom 通过空格来“美化”XML，而不是将整块文本写入磁盘。最终看上去像这样：



上面是漂亮干净的XML，同时它也是易读的。锦上添花的是，你可以运用你在PyPDF2章节中所学到的知识从PDF中提取元数据（metadata），然后将其也加入到XML中。

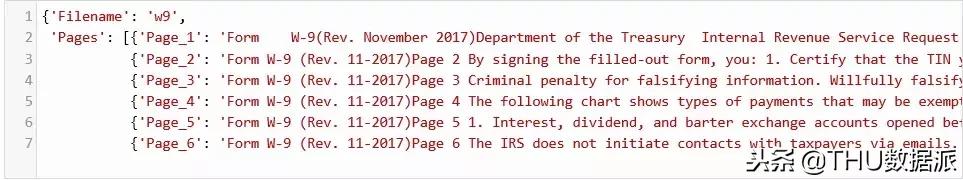
**导出成JSON**

JavaScript对象注释, 或者JSON, 是一种易读易写的轻量级的数据交换格式。Python包含一个json 模块于它的标准库中，从而允许你用编程方式来读写JSON。让我们运用从前一章节学到的内容来创建一个导出器脚本来输出JSON而不是XML：



这里，我们引入所需要的不同的库，包括PDFMiner模块。然后创建一个函数，以PDF文件的输入路径和JSON文件的输出路径为参数。在Python中JSON基本上就是一个字典，所以我们创建一对简单的顶层的键：Filename和Pages。Pages键对应一个空的表单。接着，我们循环遍历PDF的每一页并且提取每一页的前100个字符。然后创建一个字典变量以页号作为键100个字符作为值并将其添加到顶层的页表单中。最后，我们利用json 模块的dump 命令生成文件。

文件的内容最终看上去像这样：



又一次，我们得到了易读的输出。你也可以通过PDF的元数据（metadata）来加强这个例子，如果你乐意的话。请注意输出将会改变，它依赖于你想从每一页或文档中分析出什么样的结果。

现在让我们来快速看一下怎样导出CSV文件。

**导出成CSV**

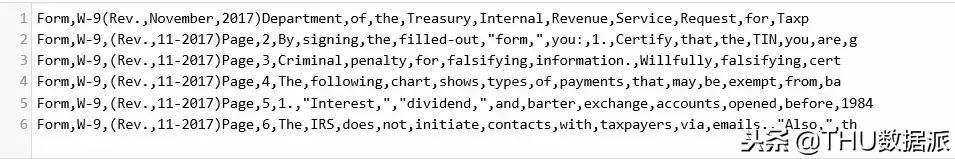
CSV是 \*\*comma separated values\*\* （逗号分隔值）的缩写。它是一种漂亮的标准格式，并且已经存在了很长时间。CSV的优点就是Microsoft Excel和 LibreOffice都能够自动地以漂亮的电子表格的方式将它们打开。你也可以在一个文本编辑器中打开CSV文件，如果你乐意看到它的原始值的话。

Python有一个内置的csv 模块，你可以用它来读写CSV文件。在这里我们将用它从我们由PDF中提取的文本来创建一个CSV。让我们看一下代码：



这个例子中，我们引入了Python的csv库。除此以外，引入的库和前一个例子相同。在函数中，我们利用CSV文件路径创建了一个CSV文件处理器。然后用文件处理器作为唯一的参数初始化了一个CSV写入器对象。接着像之前一样遍历了PDF页。这里唯一的不同就是我们将前100个字符分割成了单个的词。这将允许我们拥有一些真实的数据来加入到CSV中。如果不这样做，那么每一行将只会有一个元素在其中，那就不算一个真正的CSV文件了。最后，我们将一列单词写入CSV文件中。

这就是得到的结果：



我认为这个例子同JSON或XML的例子相比读起来难了点，但是它不算太难。现在让我们继续来看一下怎样才能将图片从PDF中提取出来。

**从PDF中提取图片**

不幸的是，并不存在Python包可以真正地做到从PDF中提取图片。我找到的最接近的东西是有一个叫**minecart**的项目宣称可以做到这一点，但是它只在Python 2.7上有效。我没法使其运行于我的PDF样本。在Ned Batchelder的博客上有一篇文章谈到了一点儿如何从PDF中提取JPG图片。代码如下：



这同样对我使用的PDF文件无效。有一些人在留言中宣称代码对他们的一些PDF文件有效，同时也有一些留言例举了修改后的代码。Stack Overflow网站上有关于这个的各种代码，其中一些这样或那样地使用了PyPDF2。但没有一个对我有效。

我的建议是使用一个类似于**Poppler** 的工具来提取图片。Poppler有一个工具叫做**pdfimages，你可以同Python的subprocess模块一起来使用。以下是你如何在没有Python的情况下使用它：**

手把手教你如何用Python从PDF文件中导出数据（附链接）

请确保images文件夹（或你想新建的任何输出文件夹）已经被创建，因为pdfimages不会为你创建它。

让我们写一个Python脚本来执行同样的命令，请确保输出文件夹已经存在：



在这个例子中，我们引入了subprocess和os模块。如果输出路径不存在，我们会尝试创建它。然后我们运用subprocess的call函数来执行pdfimages命令。使用call函数是因为它将等到 pdfimages命令完全执行完才返回。你可以代之以Popen，但是那将基本上在后台运行命令进程。最后，我们打印出输出路径下的细节，以确定所有的图片都被提取进了其中。

还有一些网络上的其它文章引用了一个叫做**Wand** 的库，你也许可以试一试。它是一个ImageMagick的包装器。还有一个值得关注的是绑定了Poppler的Python叫做**pypoppler，**虽然我没有能够找到任何和这个包相关的提取图片的例子。

**总结**

这篇文章网罗了很多信息。我们学习了一些可以用来从PDF中提取文本的包，如PDFMiner或Slate。我们还学习了如何运用Python的内置库来导出文本到XML、JSON和CSV。最后，我们研究了一下从PDF中导出图片这个棘手的问题。尽管Python目前没有任何出色的库可以完成这个工作，你可以采用其它工具的变通方案，例如Poppler的pdfimage工具模块。

原文标题：

Exporting Data From PDFs With Python

原文链接：

https://dzone.com/articles/exporting-data-from-pdfs-with-python

**译者简介**



季洋，苏州某IT公司技术总监，从业20年，现在主要负责Java项目的方案和管理工作。对大数据、数据挖掘和分析项目跃跃欲试却苦于没有机会和数据。目前正在摸索和学习中，也报了一些线上课程，希望对数据建模的应用场景有进一步的了解。不能成为巨人，只希望可以站在巨人的肩膀上了解数据科学这个有趣的世界。

**— 完 —**

关注清华-青岛数据科学研究院官方微信公众平台“**THU数据派**”及姊妹号“**数据派THU**”获取更多讲座福利及优质内容。