

批量提取pdf文档图片

一、应用场景

我们有时候想要保存pdf文档的图片，但当pdf文档中有多张图片时，频繁的操作会让人头皮发麻而且会花费大量的时间，这里教大家如何使用python，一步到位，快速提取并保存pdf文档中的所有图片。

二、项目代码

```
1 import fitz, time, re, os
2
3 # 从pdf中提取图片
4 def pdf_pic(path, pic_path):
5     t0 = time.clock()
6
7     # 使用正则表达式来查找图片
8     checkX0 = r"/Type(?! = */XObject)"
9     checkIM = r"/Subtype(?! = */Image)"
10
11     # 打开pdf
12     doc = fitz.open(path)
13
14     # 图片计数
15     imgcount = 0
16     lenXREF = doc._getXrefLength()
17
18     # 打印PDF的信息
19     print(" 页数: {}, 对象: {}".format(len(doc), lenXREF - 1))
20
21     # 遍历每一个对象
22     for i in range(1, lenXREF):
23         # 定义对象字符串
24         text = doc._getXrefString(i)
25         isXObject = re.search(checkX0, text)
26
27         # 使用正则表达式查看是否是图片
28         isImage = re.search(checkIM, text)
29
30         # 如果不是对象也不是图片，则continue
```

```

31     if not isXObject or not isImage:
32         continue
33     imgcount += 1
34
35     # 根据索引生成图像
36     pix = fitz.Pixmap(doc, i)
37     # 根据pdf的路径生成图片的名称
38     new_name = "img{}.png".format(imgcount)
39     new_name = new_name.replace(':', '')
40
41     # 如果pix.n<5,可以直接存为PNG, 否则先转换CMYK
42     if pix.n < 5:
43         pix.writePNG(os.path.join(pic_path, new_name))
44     else:
45         pix0 = fitz.Pixmap(fitz.csRGB, pix)
46         pix0.writePNG(os.path.join(pic_path, new_name))
47         pix0 = None
48
49     # 释放资源
50     pix = None
51     t1 = time.clock()
52     print("运行时间:{}s".format(t1 - t0))
53     print("提取了{}张图片".format(imgcount))
54
55 if __name__ == '__main__':
56     path = input('请输入需要批量提取图片的pdf文档路径: ') # pdf文档路径
57     pic_path = input('请输入保存图片的文件夹路径: ') # 保存图片的文件夹路径
58     pdf_pic(path, pic_path)

```

三、项目操作及解析

运行代码前需要确保安装好pymupdf模块。在运行时，只需要输入需要提取图片的pdf文档路径与保存图片的文件夹路径即可。

3.1 pymupdf模块安装

模块安装命令：

1. pip install pymupdf (Windows系统)
2. pip3 install pymupdf (Mac系统)

关于pymupdf模块:


MuPDF的原始渲染库称为Libart。“在Artifex Software收购MuPDF项目之后，开发重点转移到了编写一个新的现代图形库上。Fitz最初是作为R&D项目来替代老化的Ghostscript图形库，但已成为支持MuPDF的渲染引擎。

3.2 图片字符段识别

关键代码:

```
1 # 该代码块不能单独使用
2 checkXO = r"/Type(?! */XObject)"
3 checkIM = r"/Subtype(?! */Image)"
4
5 doc = fitz.open(path) # 返回的是doc对象
6 lenXREF = doc._getXrefLength() # 计算出有多少个对象数
7
8 # 遍历每一个对象
9 for i in range(1, lenXREF):
10     # 定义对象字符串
11     text = doc._getXrefString(i) # 将每一个对象转为字符串
12
13     # 使用正则表达式查看是否是对象
14     isXObject = re.search(checkXO, text)
15     # 使用正则表达式查看是否是图片
16     isImage = re.search(checkIM, text)
17
18     if not isXObject or not isImage: #如果既不是对象也不是图片，则回到循环继续寻找
19         continue
20
21     pix = fitz.Pixmap(doc, i) # 根据索引生成图像
```

代码解析:

1. 第5行代码中，doc变量是一个对象，可以理解为doc是由多个小对象组成的。例如对象doc = () + doc2 + doc3.....
2. 第11行代码，是将doc里面的每一个小对象里的数据，都解析为一个字符串，由于图片的具有特殊的字符片段（与checkXO与checkIM变量值的内容一致），第14行代码与第15行代码，则通过正则表达式进行字符片段匹配，查找出具有图片特殊字符片段的小对象。