

爬虫 | Python下载m3u8视频

目录

- 从 m3u8 文件中解析出 ts 信息
- 按时间截取视频
- 抓取 ts 文件
 - 单文件测试
 - 批量下载
- 合并 ts 文件
- 将合并的ts文件转化为视频文件

参考资料：

- m3u8格式介绍
- ts文件格式介绍
- 视频下载
- Python读取m3u8文件
- ts转mp4

```
# 配置环境

import requests,re
import sys,time
import os
import numpy as np
import glob

work_dir = os.getcwd()
print(work_dir)

# 用来保存ts文件
file_dir = os.path.join(work_dir, 'file_tmp')

if not os.path.exists(file_dir):
    os.mkdir(file_dir)
```

先定义保存文件的函数

```
def savefile(file_url,file_name):
    # 配置headers防止被墙，一般问题不大
    headers = {
        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3
    }

    r = requests.get(file_url,headers=headers)

    if r.status_code == 200:
        with open(file_name, 'wb') as f:
            f.write(r.content)
```

从 m3u8 文件中解析出 ts 信息

怎么查找m3u8文件？

假设在chrome上打开视频页

≡ CONTENTS ×

1. 从 m3u8 文件中解析出 t...

2. 按时间截取视频

3. 抓取 ts 文件

3.1. 单文件测试

3.2. 批量下载

4. 合并 ts 文件

5. 将合并的ts文件转化为视...



一般可以看到两个m3u8地址, 其中一个是带hls的, 这个文件可以解析出ts信息

拿个网址来举例, 比如[这个视频](#)

```
# 如果url中没有hls的, 那就是源m3u8文件
# 源m3u8文件会跳转到另一个m3u8文件, 这个地址中就带有hls

# 这个是源m3u8文件, 不带hls
url_m3u8 = 'https://wuji.zhulong-zuida.com/20190706/762_c260ca6c/index.m3u8'

r = requests.get(url_m3u8)
r.encoding='utf-8'

# 查看内容

print(r.text)
```

输出:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=800000,RESOLUTION=1080x608
800k/hls/index.m3u8
```

可以看到最后一行就是跳转后的m3u8地址

```
# 合成带有hls的m3u8地址
if r.text.split('\n')[-1] == '':
    hls_mark = r.text.split('\n')[-2] # 以防\n结尾
else:
    hls_mark = r.text.split('\n')[-1]
url_m3u8_hls = url_m3u8.replace('index.m3u8', hls_mark)
url_m3u8_hls
```

输出:

```
'https://wuji.zhulong-zuida.com/20190706/762_c260ca6c/800k/hls/index.m3u8'
```

```
# 不过有时候可能没法查到跳转后的带hls的连接
# 但是视频加载文件的网址格式为 主url+文件名.ts
# 这个主url是带hls的
# m3u8的index目录 格式为 主url/index.m3u8
url_m3u8_hls = 'https://wuji.zhulong-zuida.com/20190706/762_c260ca6c/800k/hls/index.m3u8'
```

```
# 带有hls的m3u8文件中获得的是ts信息
# 包括ts文件名称, 以及该文件的持续时间
```

```
# 这个文件有用, 先保存一下
file_m3u8 = url_m3u8_hls.split('/')[1]
with open(file_m3u8, 'wb') as f:
    f.write(r.content)
```

```
# iter_lines得到的是bytesstring
text_bytes = list(r.iter_lines())

# 转化成正常string
text_string = [i.decode('utf-8') for i in text_bytes]
```

```
# 筛选以.ts结尾的行
# 有些情况下可能是以其他格式的文件, 比如png, 下载后修改后缀即可
# ts_name = [i for i in text_string if i.endswith('.ts')]
```

CONTENTS

1. 从 m3u8 文件中解析出 t...
2. 按时间截取视频
3. 抓取 ts 文件
 - 3.1. 单文件测试
 - 3.2. 批量下载
4. 合并 ts 文件
5. 将合并的ts文件转化为视...

Copy

Copy

Copy

Copy

Copy

输出：

```
[ '36962c1a1b000000.ts', '36962c1a1b0000001.ts', '36962c1a1b0000002.ts' ]
```

有时候ts文件信息中可能还包含一部分路径信息。
因为路径都是统一的，所以我们只需要文件名就可以了

```
if '/' in ts_name[1]:
    # 部分ts文件名中带有路径信息，只保留文件名即可
    ts_name = [i.split('/')[-1] for i in ts_name]

ts_name[:3]
```

接下来处理时间戳。

```
# 筛选带有时间的行
ts_time = [float(re.findall('[.\d]+',i)[0]) for i in text_string if i.startswith('#EXTINF')]

ts_time[:3]
```

输出：

```
[4.1283, 4.3785, 4.17]
```

```
# 检验解析出来的时间戳和文件名数量是否匹配
len(ts_name) == len(ts_time)
```

输出：

```
True
```

按时间截取视频

```
# 建立时间基准
# 得到累计时间序列
time_cum = np.cumsum(ts_time)

# 那如果我要看51分05秒~55分46秒，应该下载哪些文件呢？

time_start = 1*3600+26*60+5
time_end = 1*3600+46*60+20

# 如果有多段时间截取，可以写个函数将时间序列进行转化
# 输入：[(0.0.0,0.9.30),(0.10.0,0.20.0)]
# 输出 累计时间戳的index [(0,38),(40,80)]

# 对于起始时间
# 筛选累计时间戳<开始时间的最大值，再找对应的index

index_start = sum(time_cum<time_start)+1-1
# +1是为了截取
# -1 是为了矫正index序号
```

这个就是最终的index，我们这里的原则是最后取到的时间区间是完全包含目标区间的

这里的index实际上做了一个位移的，本来index是0开始，所以说是351

e.g. 假设前3个ts的时间为2,3,3

现在我要的是第4秒后的信息，得到的累计时间序列是2,5,8

CONTENTS

1. 从 m3u8 文件中解析出 t...

2. 按时间截取视频

3. 抓取 ts 文件

3.1. 单文件测试

3.2. 批量下载

4. 合并 ts 文件

5. 将合并的ts文件转化为视...

```
# 对于截止时间
index_end = len(time_cum) - sum(time_cum>time_end) +1 - 1
```

同样假设 整个累计时间序列为 2,5,8,10

现在截取到6秒，所以要取到第3个ts文件(Python index为2)

时间序列长度4-大于6的个数2+偏移1位-index矫正1位

这就是最终的index

```
print(index_start,index_end)
```

输出：

```
1290 1594
```

抓取 ts 文件

单文件测试

```
ts_name[0] # 这个是片头
```

输出：

```
'36962c1a1b0000000.ts'
```

```
# 这个网址去除掉最后的文件名就是**主url**了。
file_url = 'https://wuji.zhulong-zuida.com/20190706/762_c260ca6c/800k/hls/36962c1a1b0000000.ts'

# 提取文件名
file_name = os.path.join(file_dir,file_url.split('/')[-1])

# 下载ts文件到本地
savefile(file_url,file_name)
```

批量下载

```
# 先看下我们要抓取的ts文件的index的起始位置
print(index_start,index_end)
```

输出：

```
1290 1594
```

```
# ts文件的主url以/hls/结束
url_m3u8_hls
```

输出：

```
'https://wuji.zhulong-zuida.com/20190706/762_c260ca6c/800k/hls/index.m3u8'
```

```
# 提取主url
url_ts_main = url_m3u8_hls.replace('index.m3u8','')
# 绝大部分hls文件的名称都是index.m3u8,个别的也可能是其他名字

# range 函数取头不取尾，所以+1
for idx in range(index_start,index_end+1):
    # 拼接url
    file_name = ts_name[idx]
```

Copy

≡ CONTENTS

×

1. 从 m3u8 文件中解析出 t...

2. 按时间截取视频

3. 抓取 ts 文件

3.1. 单文件测试

3.2. 批量下载

4. 合并 ts 文件

5. 将合并的ts文件转化为视...

Copy


[首页](#)
[归档](#)
[资源](#)
[新随笔](#)
[联系](#)
[订阅](#)
[管理](#)


```
file_url = url_ts_main+file_name
```

```
# 对于后续可能是其他格式的情况下，保存为以.ts结尾的文件即可
```

```
# 有的服务器可能会改变后缀假装自己不是ts文件
if not file_name.endswith('.ts'):
    tmp_name = file_name.split('.')[::-1]
    tmp_name.append('.ts')
    file_name = '.'.join(tmp_name)

file_path = os.path.join(file_dir,file_name)

# 保存文件
savefile(file_url,file_path)

# 提示进度
sys.stdout.write('\r当前进度 第%d页 剩余%d页'%(idx,index_end-idx))
sys.stdout.flush()
time.sleep(0.1)
```

CONTENTS

1. 从 m3u8 文件中解析出 t...
2. 按时间截取视频
3. 抓取 ts 文件
 - 3.1. 单文件测试
 - 3.2. 批量下载
4. 合并 ts 文件
5. 将合并的ts文件转化为视...

输出：

```
当前进度 第1594页 剩余0页
```

合并 ts 文件

第一种方式可以考虑，使用命令行操作

如果是在windows上操作

- `copy /b 路径*.ts 路径\合并文件.ts`
- `copy /b "1.ts"+"2.ts"+...+"n.ts" /y "combine.ts"`

如果是在mac上操作

- `cat 1.ts 2.ts > combine.ts`

注意：文件的顺序要正确才行

```
# 如果有ts文件的index
# 那么可以用index直接来生成有顺序的list即可
file_list = [os.path.join(file_dir,ts_name[i]) \
              for i in range(index_start,index_end+1)]

# 直接扫描路径下的ts文件也是可以的
# 也可以删掉部分ts文件
# file_list = glob.glob(os.path.join(file_dir,'*.ts'))
# file_list.sort()

# 这里是在mac上操作，所以名称以空格相连
filepath_cat = ' '.join(file_list)

cmd_str = 'cat ' + filepath_cat + '> merge.ts'
# cat 1.ts 2.ts > combine.ts

os.system(cmd_str)
```

[Copy](#)

执行成功的话，会返回 0

第二种合并ts文件的方式，可以将所有的ts文件按顺序写到一个新的文件中。

```
file_out = 'merge_02.ts'

with open(file_out,'wb') as f_out:
    for f_in in file_list:
        f_out.write(open(f_in,'rb').read())
```

最后，我们将合成的ts文件转化成视频文件(比如MP4格式)

这里我们调用 ffmpeg 将 ts 文件转化为视频文件。

转化命令为

```
• ffmpeg -i 文件名称.ts -c copy [视频名称]

• e.g. ffmpeg -i merge.ts -c copy '视频截取片段.mp4'

file_in = os.path.join(work_dir, 'merge.ts')

#如果路径中有空格，所以路径需要用上双引号，否则会找不到该文件

file_out = "merge.mp4"

# 这里是去ffmpeg官网下载编译好的软件包，免安装的
cmd = './ffmpeg -i '+file_in +' -c copy ' + file_out

os.system(cmd) # 运行正常返回0
```

≡ CONTENTS ×

1. 从 m3u8 文件中解析出 t...

2. 按时间截取视频

3. 抓取 ts 文件

3.1. 单文件测试

3.2. 批量下载

4. 合并 ts 文件

5. 将合并的ts文件转化为视...

！

作者：dataxon
出处：https://www.cnblogs.com/dataxon/p/12533110.html
版权：本文采用「署名-非商业性使用-相同方式共享 4.0 国际」知识共享许可协议进行许可。
转载请注明作者及出处，公众号【dataxon】同步发文

分类: 03_编程技能

Sponsor

推荐 0

反对 0

« 上一篇：书单 | 数据分析

» 下一篇：Python基础 | 日期时间操作

posted @ 2020-03-20 17:29 dataxon 阅读(1232) 评论(0) 编辑 收藏

登录后才能发表评论，立即 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

首页

新闻

博问

专区

闪存

班级