B站up主视频中的音频爬取和音频数据处理

娱乐时间：2022-07-14 到 2022-07-17

项目需求

爬取B站音乐区up主（风语·呢喃）的主页里所有的视频的音频信息（265首音乐，按照up主的最新发布顺序爬取），并上传到自己的私人云盘里用来享受生活。
（仅供学习交流，不做任何商业用途）
（本人萌新一枚，欢迎各路大佬指教）

步骤：

0. 全文梗概

整个爬虫项目我分为前半部分和后半部分，

前半部分是：先利用webdriver从up主主页中爬取完整网页，再利用python的re模块和正则表达式获取其中的所有的视频的标题名称、封面图片地址、简介和BV号，把BV号拼成对应视频的网址，保存到tsv文件中

后半部分是：从tsv文件中读取所有视频的标题、网址、图片地址和简介，然后利用pydub进行MP3文件头修改，再高品质地保存音乐。

下面从初始阶段开始记录复盘

1. 初步了解

到达up主主页中的“全部视频”页面：
https://space.bilibili.com/6272252/video 谷歌浏览器按F12在检测工具中的network项中找 earch?mid=6272252&ps=30&tid=0&pn=1&keyword=&order=pubdate&jsonp=jsonp项，可以在它的preview中看到当前服务器反馈的数据，展开data->list->vlist可以看到我们需要的视频信息

2. 获取目标视频的所有信息

2.1 要哪些目标信息呐

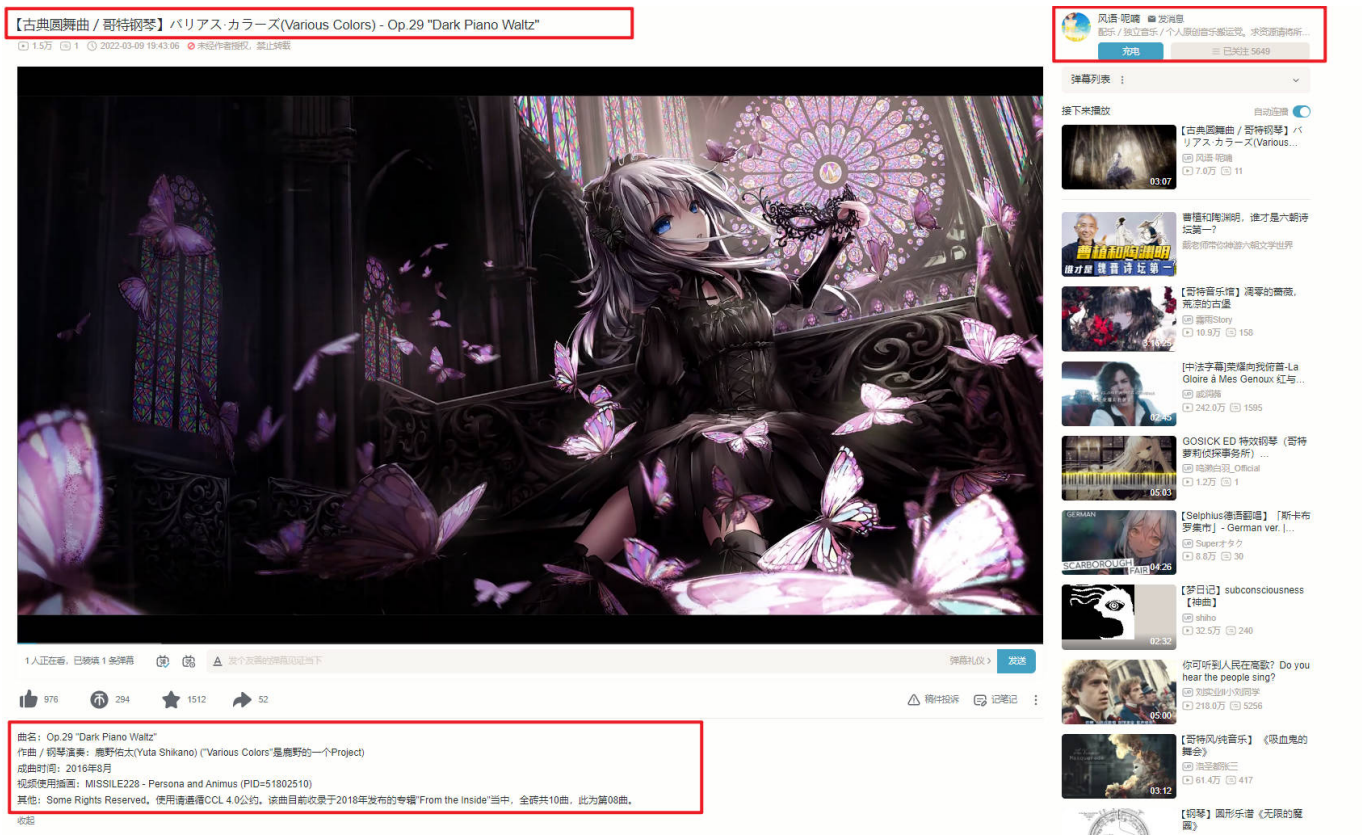
因为up主对每个音乐视频的标题都写了详细的标签，

```
格式形如：
【(?P<style>音乐风格)】(?P<title>作者 - 歌曲名字)


eg:
【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors) - Op.29 "Dark Piano Waltz"
```

其他内容和备注如音乐创作时间、封面图片来源等都做了简要介绍，

```
eg:
曲名：Op.29 "Dark Piano Waltz"
作曲 / 钢琴演奏：鹿野佑太(Yuta Shikano) ("Various Colors"是鹿野的一个Project)
成曲时间：2016年8月
视频使用插画：MISSILE228 - Persona and Animus (PID=51802510)
其他：Some Rights Reserved。使用请遵循CCL 4.0公约。该曲目前收录于2018年发布的专辑"From the Inside"当中，全砖共10曲，此为第08曲。
```



最初我只需要获取每个音乐视频的标题、风格、简介（因为有些音乐视频是合集，有些up主没有完全表明作者和出处，为了方便批(wo)量(de)处(lan)理(duo)），我只需要提取视频BV号（"bvid"）、视频标题("title")和简介("description")即可。

（后面发现这只up猪的视频封面配图选取很nice，所以就加上封面图片的获取，并且希望把图片作为爬取的音乐的封面，所以进行MP3音乐文件的二进制的文件头ID3的数据写入，详见之后的内容）

2.2 requests 和 webdriver 获取网页源码数据

先用request请求服务器返回html页面数据做初步观察

```
import requests

pn = 0  # 批量化处理时的遍历变量（用于翻页）

# 设置请求头等参数，防止被反爬
```

```
header = {
    'Accept': '*/*',
    'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
    'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.60
Safari/537.36"
}

# 网页请求的传入参数
param = {
    #
mid=6272252&ps=30&tid=0&pn=1&keyword=&order=pubdate&jsonp=jso
np
    "mid": 6272252,
    "ps": 30,
    "tid": 0,
    "pn": str(pn),
    "keyword": 0,
    "order": "pubdate",
    "jsonp": "jsonp"
}
# 获取包含对应数据的源代码的url
url = "https://api.bilibili.com/x/space/arc/search"

url_str = url + "?mid=6272252&ps=30&tid=0&pn=" + str(pn) +
"&keyword=&order=pubdate&jsonp=jsonp"

resp = requests.get(url=url, params=param, headers=header,
verify=False)  # verify=False 去掉安全验证，B站这里可要可不要

# 输出中文乱码的解决方法
```

```
# print(resp.encoding)
# print(requests.utils.get_encodings_from_content(resp.text)
[0])  # 利用此句 可以 requests 去猜测获取的文本编码格式
resp.encoding = "utf-8"

# 小睡一秒，防止被当成爬虫而被服务器反爬
time.sleep(1)
# 模拟向服务器发送请求会话完成任务后记得关闭会话
resp.close()

print("第" + str(pn) +"页数据：")
print("resp.text\n")
print(resp.text)
# print(type(resp.text))
# print(type(resp))
```

目前似乎很顺，然后用for循环遍历试图获取所有信息，出了问题：up一共有265个视频（十页，pn= 0 ~ 9 ），可是每次循环只能获得52个视频信息数据，然后就不再获得数据。

观察通过requests请求返回的这52个数据和用浏览器F12看到的数据进行对比，发现二者数据顺序和内容差别很大。初步推测requests是只获得了B站服务器预先加载的52个视频信息数据，怀疑是否被服务器认出来爬虫程序，然后只给一部分小数据。

经过资料查找，发现了 webdriver 库似乎能更好的模拟用户的访问行为。于逝，

```
from selenium import webdriver

# webdriver是一个.exe文件，需要配置
# selenuim需要pip安装
```

```
# pn, header, url, param, url_str 内容同上

chromedriver = "E:\ANACONDA\chromedriver"  # .exe文件存放在和（base虚拟环境中python.exe ？所在位置的同一个位置
bro = webdriver.Chrome(chromedriver)  # 初始化浏览器窗口对象
bro.get(url_str)  # 进入网页

time.sleep(3) # 模拟网页启动速度较慢
# 模拟鼠标滚轮滚动网页（在这里没啥用）
bro.execute_script('window.scrollTo(0,
document.body.scrollHeight)')

print("第" + str(pn) +"页数据：")
print("bro.page_source")
print(bro.page_source)
# print(type(bro.page_source))
# print(type(bro))
```

利用 webdriver 获得的网页源码和通过浏览器 F12 看到的内容一致，ok

2.3 re模块与正则表达式提取超长string中的目标字段

bro.page_source 是网页源码的str类型对象，富含标签和一堆js源码（js源码堆在一行可以利用：在线代码格式化 进行代码格式化展示，便于查看代码）

data->list->vlist可以看到我们需要的视频信息，所以先定位到目标内容的区域

```
bro_str = re.search(r'"vlist":\[(.*?)]}', bro.page_source,
re.S).group().strip('"vlist":\["')
print(bro_str)
```

接下来挨个提取信息

```
# 视频标题名称
obj_title = re.compile(r'"title":"(?P<title>.*?)","', re.S)
#  () 用来分组  其中 ?P<title> 表示该分组名字是 titletitle =
obj_title.finditer(bro_str)


# 网址+该视频的b站番号
obj_web = re.compile(r'"bvid":"(?P<bvid>.*?)","', re.S)
web = obj_web.finditer(bro_str)


# up对音乐的风格分类  (第九页一个小问题，有一个视频不是按照这个规则归类的，但是提取的时候似乎没有什么顺序混乱现象)
obj_styles = re.compile(r'"title":"【(?P<style>.*?)】', re.S)
styles = obj_styles.finditer(bro_str)


## 由于乐曲的作者、成曲时间等部分信息不是在每一个视频里都有，所以直接获取description
obj_desc = re.compile(r'"description":"(?
P<description>.*?)","')
description = obj_desc.finditer(bro_str)
```

```
## 每页数据打包到lst列表中，读取一页视频信息数据就存储一页到tsv文件中（python中的列表对象有存储大小上限吧）


lst = []
for tit, w, st, des in zip(title, web, styles, description):
# python多个迭代对象变量循环，要把被迭代对象用zip封装，否则报错
ValueError: not enough values to unpack (expected 2, got 0)
    # print(tit.group("title"))   # 似乎这个迭代对象在一次循环内被操作后就会被清空
    # print("https://www.bilibili.com/video/" + w.group("bvid"))
    # print(st.group("style"))    # print(des.group("description"))
    tmp = []
    tmp.append(tit.group("title"))
    tmp.append("https://www.bilibili.com/video/" + w.group("bvid"))
    tmp.append(st.group("style"))
    tmp.append(des.group("description"))
    lst.append(tmp)

    for ite in tmp:
        f.write(ite + '\t')
    f.write("\n")
```

2.4 利用webdriver和re模块提取目标信息

完整代码

```
# 前半部分代码整理

import requests
import re
import pandas as pd
import numpy as np
import time
import random

from selenium import webdriver


# 这里选择保存为tsv文件的原因是：信息中包含英文逗号，导致干扰csv的单元格划分，
# 最初我编码格式选择GBK，因为我们使用excel打开tsv文件，而excel默认是utf-8，会导致中文乱码
# 但是使用GBK在识别第三页第5、6个视频的信息时报错UnicodeEncodeError: 'gbk' codec can't encode character '\xf6' in position 57: illegal multibyte sequence
# sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding="gb18030")
# 所以还是选回了utf-8

with open("music_collection_utf_8.tsv", "a", encoding="utf-8") as f:
    f.write("'视频标题名称'\\t'网址'\\t'up对音乐的风格分类'\\t'视频简介'" + "\n")

    cnt = 0
```

```
        for pn in range(1, 10, 1):  # 视频共9页
            # 设置请求头等参数，防止被反爬
            header = {
                'Accept': '*/*',
                'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
                'User-Agent': "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.60 Safari/537.36",
            }

            # 获取源代码
            url =
"https://api.bilibili.com/x/space/arc/search"
            param = {
                #
mid=6272252&ps=30&tid=0&pn=1&keyword=&order=pubdate&jsonp=jso
np
                "mid": 6272252,
                "ps": 30,
                "tid": 0,
                "pn": str(pn),
                "keyword": 0,
                "order": "pubdate",
                "jsonp": "jsonp"
            }

            url_str = url + "?mid=6272252&ps=30&tid=0&pn=" +
str(pn) + "&keyword=&order=pubdate&jsonp=jsonp"

            chromedriver = "E:\ANACONDA\chromedriver"
```

```
            bro = webdriver.Chrome(chromedriver)  # 初始化浏览器窗口对象
            bro.get(url_str)  # 进入网页

            time.sleep(3)  # 模拟鼠标滚轮滚动网页（在这里没啥用）
            bro.execute_script('window.scrollTo(0, document.body.scrollHeight)')

                print("\n\n第" + str(pn) +"页网页源码数据：\n")
                print("bro.page_source")
                print(bro.page_source)


                print("\n正在提取第" + str(pn) +"页视频信息数据：\n")


                bro_str = re.search(r'"vlist":\[(.*?)]}', bro.page_source, re.S).group().strip('"vlist":\["')
                print(bro_str)

                # 视频标题名称
                obj_title = re.compile(r'"title":"(?P<title>.*?)","', re.S)   #  () 用来分组  其中 ?P<title> 表示该分组名字是 titletitle = obj_title.finditer(bro_str)

                # for tit in title:   # title 是可迭代对象 <class 'callable_iterator'>#     print(tit.group("title"))  # group() 用来提出分组截获的字符串

                # 网址+该视频的b站番号
```

```
                obj_web = re.compile(r'"bvid":"(?
P<bvid>.*?)","', re.S)
                web = obj_web.finditer(bro_str)

                # for w in web:    # web 是可迭代对象 <class
'callable_iterator'>#     print(w.group("bvid"))  # group()
用来提出分组截获的字符串



                # up对音乐的风格分类  (第九页一个小问题，有一个视频
不是按照这个规则归类的，但是好像最后正确归类了？？？)
                obj_styles = re.compile(r'"title":"【(?
P<style>.*?)】', re.S)
                styles = obj_styles.finditer(bro_str)

                # for st in styles:
                #     print(st.group("style"))

                ## 由于乐曲的作者、成曲时间等部分信息不是在每一个视频
里都有，所以直接统计description
                obj_desc = re.compile(r'"description":"(?
P<description>.*?)","')
                description = obj_desc.finditer(bro_str)

                # for des in description:
                #     print(des.group("description"))



                lst = []
                for tit, w, st, des in zip(title, web,
styles, description):   # python多个迭代对象变量循环，要把被迭代对
```

```
象用zip封装，否则报错ValueError: not enough values to unpack
(expected 2, got 0)
                    # print(tit.group("title"))    # 似乎这个迭
代对象在一次循环内被操作了就会清空
                    # print("https://www.bilibili.com/video/"
+ w.group("bvid"))
                    # print(st.group("style"))    #
print(des.group("description"))
                    tmp = []
                    tmp.append(tit.group("title"))

tmp.append("https://www.bilibili.com/video/" +
w.group("bvid"))
                    tmp.append(st.group("style"))
                    tmp.append(des.group("description"))
                    lst.append(tmp)

                    for ite in tmp:
                        f.write(ite + '\t')
                    f.write("\n")

print("over!!!")
```

3. 利用爬取的视频链接获取视频音频

3.1 预备知识+测试

热知识：（目前：2022年7月）B站视频播放的时候，是分别发送视频 url 和音频 url （只下载视频的时候发现没有声音。。。。。。）

从up主的主页的全部视频页面中我能获取其全部BV号，即可利用BV号访问相应的所有视频，然后再在视频的播放页面获取对应的视频url和音频url，这两个url发送给你的二进制数据流经过本地解析并整合后才能让你看到视频，听到声音！！！

```
# video和audio下载测试

import re
import time
import requests

url = "https://www.bilibili.com/video/BV1Ts411p7Gb"

header = {
    # 'Accept': '*/*',
    # 'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',    'User-
Agent': "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.60
Safari/537.36",
        "referer": url
        # 这里强调一下要说明referer，不然好像拿不到数据！！！（具体原
因有待研究）
}


response = requests.get(url, headers=header)
html = response.text
time.sleep(2)
```

```
print(html)
response.close()

obj_info = re.compile(r"<script>window\.__playinfo__={(?
P<info>.*?)</script>", re.S)
info = obj_info.findall(html)[0]
print(info)

# 从html（js）页面中获取音频和视频的url
video_url = re.findall(r'"video":.*?,"baseUrl":"(.*?)","',
info)[0]
audio_url = re.findall(r'"audio":.*?,"baseUrl":"(.*?)",',
info)[0]
print("video_url:")
print(video_url)
print("audio_url:")
print(audio_url)

# 下载
video = requests.get(video_url, headers=header)
audio = requests.get(audio_url, headers=header)

with open("test_video" + ".mp4", "wb") as f:
    f.write(video.content)
with open("test_audio" + ".mp3", mode="wb") as f:
    f.write(audio.content)

time.sleep(3)
video.close()
audio.close()
print("over!!!")
```

我只想要好听的音乐，所以就只爬取音乐即可

3.2 数据文件读写+audio批量下载

完整代码

```
import pandas as pd
import re
import time
import random

import requests
import os


# 读取爬虫得到的数据
# data = pd.read_csv("music_collection_utf_8.tsv", sep="\t")
# print(data)
# 返回结果是[264 rows x 1 columns],

# 分类
title = []
web = []
style = []
description = []

flag = 0
```

```
with open("music_collection_utf_8.tsv", 'r', encoding="utf-8") as f:
    for line in f:
        line = line.strip("\n").split("\t")   # 先去掉 "\n" ，再利用 "\t" 切分

        # print(line)
        if flag == 0:  # 不读取第一行表头
            flag = 1
        else:
            print(line[1])
            title.append(line[0])
            web.append(line[1])
            style.append(line[2])
            description.append(line[3])

print(web)


for tit, we in zip(title, web):

    url = we

    header = {
        # 'Accept': '*/*',
        # 'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
        'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.60 Safari/537.36",
        "referer": url
    }
```

```
    # 请求html页面内容（包含音频和视频的实际url）
    response = requests.get(url, headers=header)
    html = response.text
    time.sleep(2)
    # print(html)
    response.close()

    # 利用正则表达式定位对应包含目标url的标签的内容（缩小范围）
    obj_info = re.compile(r"<script>window\.__playinfo__={(?
P<info>.*?)</script>", re.S)
    info = obj_info.findall(html)[0]
    # print(info)

    # 精确地从html（js）页面中获取音频和视频的url
    # video_url = re.findall(r'"video":.*?,"baseUrl":"
(.*?)","', info)[0]    audio_url =
re.findall(r'"audio":.*?,"baseUrl":"(.*?)",', info)[0]
    # print("video_url:")
    # print(video_url)    print("audio_url:")
    print(audio_url)

    # 请求资源下载
    # video = requests.get(video_url, headers=header)
    audio = requests.get(audio_url, headers=header)

    # with open("test_video" + ".mp4", "wb") as f:
    #     f.write(video.content)    with open(tit + ".mp3",
mode="wb") as f:
        f.write(audio.content)
```

```
    time.sleep(3)
    # 睡3秒防止被当成爬虫，然后记得关闭和服务器的会话
    # video.close()
    audio.close()
    print(tit + ".mp3  download  over!!!\n")
```

4. 为音乐文件添加标签

我的新需求：我觉得单纯的爬取音乐文件可能会比较单调，作为网易云9级萌新用户，我希望在播放音乐的时候可以看到旋转的美丽的音乐封面图片，而这只up猪对每个音乐视频的封面选图雀食不戳，并且我还想办法把视频简介中关于音乐和封面的介绍附在MP3文件里面，于逝

4.1 引入ID3v2和FFmpeg

ID3v2 中文文档 (版本 2.3.0)_DREAMER -程序员ITS201_id3v2 - 程序员ITS201

使用Python读取Mp3的标签信息_wx61eaae213a986的技术博客_51CTO博客

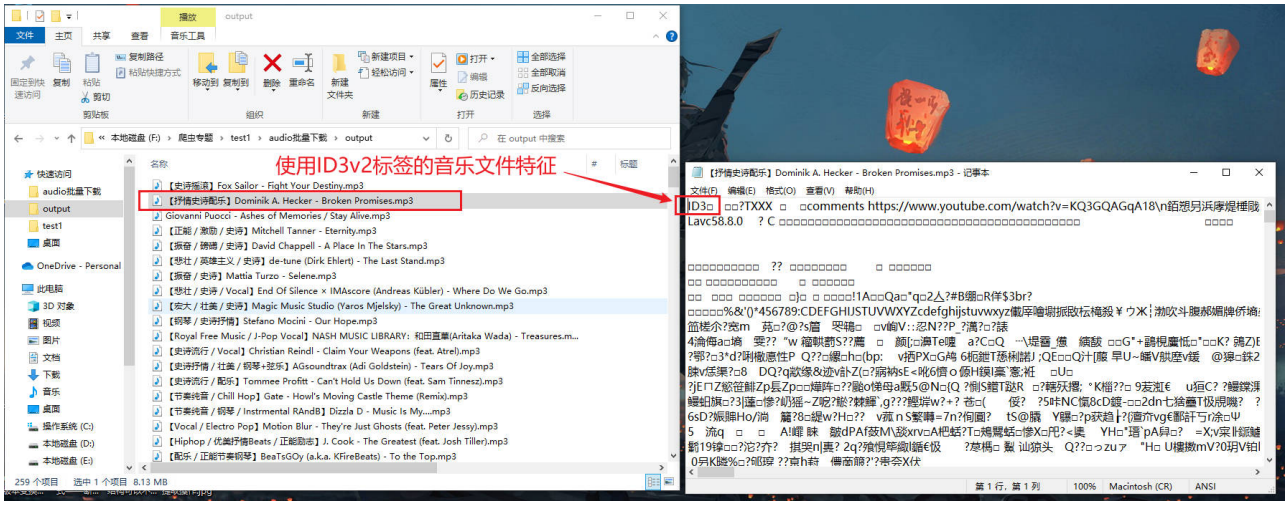
个人简要理解：MP3格式文件由文件头、音频数据、文件尾部等组成。其中，文件头中可以包含一些标签，来显示该音乐的创作者、收录专辑等信息，这些标签的集合的一个代表是ID3v2，应该是现在比较流行的音乐文件头规范的事实标准，我们可以利用其直接在MP3文件头里添加备注、图片信息（好像目前只能MP3添加图片封面）等。

FFmpeg再学习 -- Windows下安装说明_聚优致成的博客-CSDN博客

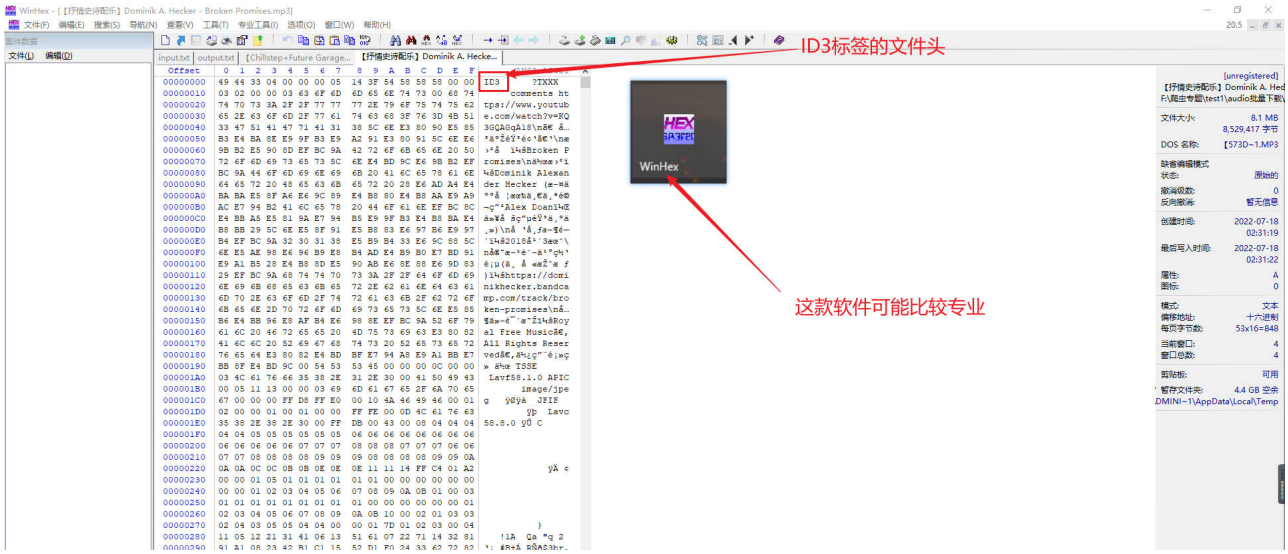
4.2 关于音乐文件标签的探索（引入pydub）

使用ID3v2作为文件头标签的音乐文件会在整个文件的头部写入二进制的‘ID3‘ 这三个字符，如何查看呢？举两个例子

1. 记事本打开



2. WinHex（付费，只有45天免费试用）



如何写入标签呢

我最初看到有利用eyeD3库来处理的帖子，但是我没怎么研究用它提供的插件函数。

我爬取的MP3音乐文件似乎也不是用ID3做标签的。

后来通过ffmpeg找到了pydub库，这个库确实方便。

最好用的python音频库之一：pydub的中文文档（含API） _一朵蘑菇的疯言疯语-程序员宅基地 - 程序员宅基地

```
# 测试：批量加入视频封面图片

import re
import time
import requests
from pydub import AudioSegment
import chardet

# 分类
title = []
web = []
style = []
picture = []
description = []


flag = 0
with open("music_collection_utf_8_pic.tsv", 'r',
encoding="utf-8") as f:
    for line in f:
        line = line.strip("\n").split("\t")   # 先去掉 "\n" ，
再利用 "\t" 切分

        # print(line)
        if flag == 0:  # 不读取第一行表头
            flag = 1
        else:
            print(line[1])
```

```
            title.append(line[0])
            web.append(line[1])
            style.append(line[2])
            picture.append(line[3])
            description.append(line[4])


print(web)


# 下载音乐
pic_path = "F:\\爬虫专题\\test1\\audio批量下载\\pic\\"


for tit, we, pi in zip(title, web, picture):

    url = we

    header = {
        # 'Accept': '*/*',
        # 'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
        'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.60 Safari/537.36",

        "referer": url
    }

    # 请求html页面内容（包含音频和视频的实际url）
    response = requests.get(url, headers=header)
    html = response.text
```

```
    time.sleep(2)
    # print(html)
    response.close()

    # 利用正则表达式定位对应包含目标url的标签的内容（缩小范围）
    obj_info = re.compile(r"<script>window\.__playinfo__={(?
P<info>.*?)</script>", re.S)
    info = obj_info.findall(html)[0]
    # print(info)

    # 精确地从html（js）页面中获取音频和视频的url
    # video_url = re.findall(r'"video":.*?,"baseUrl":"
(.*?)","', info)[0]
    audio_url = re.findall(r'"audio":.*?,"baseUrl":"(.*?)",',
info)[0]
    # print("video_url:")
    # print(video_url)
    print("audio_url:")
    print(audio_url)

    # 请求资源下载
    # video = requests.get(video_url, headers=header)
    audio = requests.get(audio_url, headers=header)

    # with open("test_video" + ".mp4", "wb") as f:
    #     f.write(video.content)
    with open(tit + ".mp3", mode="wb") as f:
        f.write(audio.content)

    time.sleep(3)
    # 睡3秒防止被当成爬虫，然后记得关闭和服务器的会话
```

```
    # video.close()
    audio.close()
    print(tit + ".mp3  download  over!!!\n")


        # 获取图片

        response_pic = requests.get(pi, headers=header)

        obj_bvid = re.compile(r'/video/(?P<bvid>.*?)$', re.S)
        bvid = re.search(obj_bvid, we).group("bvid")

        print("bvid : " + bvid)


        with open(pic_path + bvid + ".jpg", "wb") as f:
            f.write(response_pic.content)

        print("\n" + tit + " -------> " + bvid + ".jpg
download  over!!!\n\n")


        time.sleep(2)

        # 插入图片到MP3文件头

        sound = AudioSegment.from_file("music1.mp3",
                                       frame_rate=44100, channels=2,
                                       sample_width=2)
                                       # , frame_rate=44100,
channels=2, sample_width=2
```

```
        file_handle = sound.export("\\output\\" + tit +
".mp3",
                                   format="mp3",
                                   cover="\\pic\\" + bvid +
".jpg"
                                   )

        print("\n   封面添加完成    !!!\n")
```

5. 细节处理、收尾

5.1 小bug + 处理方案

1. 测试过程中发现有小部分读取文件的编码问题（有5首歌）。

初步猜测可能是由于字符集编码方式的不同（视频标题同时包含中文、英文和日文等），无法用<视频标题的字符串>创建文件。

```
eg:
<视频标题的字符串> = 【节奏纯音/钢琴Beats】ハイウィル(high-will) -
あの夏、あの雨 (Piano Demo 2)

with open("./input/" + "<视频标题的字符串>" + ".mp3", "wb") as
f:
        ......

报错：FileNotFoundError: [Errno 2] No such file or
directory:'./input/【节奏纯音/钢琴Beats】ハイウィル(high-will) -
あの夏、あの雨 (Piano Demo 2).mp3'
```

（目前直接跳过）待研究不同的字符集编码及其兼容性后再做处理

特殊字符"&"被转换成unicode :"\uXX"

处理方法：直接在主页爬取的时候就把"&"替换为"And"

2. 转义字符对表示地址的字符串的影响

有些音乐视频名字的字符串中包含‘“’符号，在从网页爬取的时候，会被自动加上'\'进行转义，我把这个字符串作为我保存的MP3音乐文件的文件名，但是在之后读取音乐文件路径的时候，python会在之前自动加上的'\'前加上'\'进行转义。

```
## python 第一次自动加上 ' \ '  用来转义带有 ' " ' 的符号
## 并且还会为包含多行的字符串的每行行尾加上 ' \n '

# 视频标题名称
obj_title = re.compile(r'"title":"(?P<title>.*?)","', re.S)
#  () 用来分组  其中 ?P<title> 表示该分组名字是 title
title = obj_title.finditer(bro_str)

...

## python 第二次自动加上 ' \ '  用来转义带有 ' \ ' 的符号
## eg: ' \" ' -> ' \\" '   ;    ' \n ' -> ' \\n '

# 从爬取到up主主页信息的tsv文件中读取数据进行音频等内容的爬取
with
open("music_collection_utf_8_Pic_EscapeCharacter_&_And.tsv",
'r', encoding="utf-8") as f:
    for line in f:
        line = line.strip("\n").split("\t")  # 先去掉 "\n" ，再
```

```
利用 "\t" 切分
        flag += 1
        # print(line)
        if flag <= 2:  # 不读取第一行表头
            continue
        else:
            print(line)
            print(line[1])
            title.append(line[0])  ## ' \" ' -> ' \\" '
            web.append(line[1])
            style.append(line[2])
            picture.append(line[3])
            description.append(line[4])



print(web)
```

先尝试使用 repr() 函数来测试输出效果，因为 repr() 会将得到的字符串通常可以用来重新获得该对象，将对象转化为供解释器直接读取的形式。

```
# 在用户看来
# 未使用repr()的测试:
tit = '【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors) - Op.149 \"Dark Gothic Piano Waltz\"'
print(tit)
print('./input/' + tit + '.mp3')
print(str('./input/' + tit.strip("\\") + '.mp3').replace('"', '\"'))
print(str('./input/' + tit.strip("\\") + '.mp3').replace('"', '\\"'))
```

```
# 未使用repr()的输出:
# 【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors) - Op.149
"Dark Gothic Piano Waltz"
# ./input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 "Dark Gothic Piano Waltz".mp3
# ./input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 "Dark Gothic Piano Waltz".mp3
# ./input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 \"Dark Gothic Piano Waltz\".mp3



# 在python看来
# 使用repr()的测试:
file = str(r'./input/' + tit.strip("\\") +
r'.mp3').replace('"', '\\"')
print("\n\n" + file + "\n")

audio_path1 = repr('./input/' + tit + '.mp3')
print(audio_path1)

audio_path2 = repr(str('./input/' + tit.strip("\\") +
'.mp3').replace('"', '\"'))
print(audio_path2)

audio_path3 = repr(str('./input/' + tit.strip("\\") +
'.mp3').replace('"', '\\"'))
print(audio_path3)

audio_path4 = repr(str(r'./input/' + tit.strip("\\") +
r'.mp3').replace('"', '\\"'))
```

```
print(audio_path4)
print(type(audio_path4))

# 使用repr()的输出:

# ./input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 \"Dark Gothic Piano Waltz\".mp3

# './input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 "Dark Gothic Piano Waltz".mp3'
# './input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 "Dark Gothic Piano Waltz".mp3'
# './input/【古典圆舞曲 / 哥特钢琴】バリアス・カラーズ(Various Colors)
- Op.149 \\"Dark Gothic Piano Waltz\\".mp3'
# <class 'str'>
```

可以看到，在字符串输出的结果中，给我们展示的和给系统展示的是不一样的，在无名字符串前直接加r可以取消转义(r"无名字符串")，但是我目前不太清楚有名字符串对象如何取消转义（望各位大佬求解）

open() 函数中第一个参数是路径，open() 函数会先解析路径字符串，如果没有申明取消转义的话，它会自动对字符串中所有的 '' 前再加上 '' ，所以在此情况下，在python看来，不可能有解析成单个 '' 输出的情况

于是，我退而求其次，在爬取up主页的时候就把所有的 ' " ' 字符替换成 ' ; ' ，这样就能正常读取了

3. 简介信息加入MP3文件头中

由于ID3v2似乎没有专门的tag标签来对应存放我能提供的多行简介信息，所以我直接存放在现成的tag标签的"comments"（评论）中，即：

tags={"comments": desc},

```
file_handle = sound.export(file_path + 'output/' + tit +
'.mp3',
                           format="mp3",
                           bitrate="320k",  # 相当于极高音质
320kbit/s
                           tags={"comments": desc},
                           cover="./pic/" + bvid + ".jpg"
                           )

print("\n   封面添加完成    !!!\n")
```

4. MP3文件修改标签后保存的音质问题

之前没有在export中加入bitrate="320k"，导致使用默认的比特率保存（似乎只有192kbit/s），音质大打折扣。

6. 后半部分代码处理

（再次重复本文主线）整个爬虫项目我分为前半部分和后半部分，

前半部分是：先利用webdriver从up主主页中爬取完整网页，再利用python的re模块和正则表达式获取其中的所有的视频的标题名称、封面图片地址、简介和BV号，把BV号拼成对应视频的网址，保存到tsv文件中

后半部分是：从tsv文件中读取所有视频的标题、网址、图片地址和简介，然后利用pydub进行MP3文件头修改，再高品质地保存音乐。

后半部分完整代码：

```
import re
import time
import requests
from pydub import AudioSegment
import chardet

# 分类
title = []
web = []
style = []
picture = []
description = []

flag = 0
# 这里的文件名迭代是根据每次处理的报错修改的
# 最初文件名:"music_collection_utf_8_pic.tsv"
# python自动转义问题导致两次自动
加"\":"music_collection_utf_8_Pic_EscapeCharacter.tsv"
# 特殊字符"&"被转换成unicode :"\uXX"
with open("music_collection_utf_8_Pic_EscapeCharacter_&_And - 副本 - 副本 - 副本.tsv", 'r', encoding="utf-8") as f:
    for line in f:
        line = line.strip("\n").split("\t")  # 先去掉 "\n" ，再利用 "\t" 切分
        flag += 1
```

```
        # print(line)
        if flag <= 2:  # 不读取第一行表头
            continue
        else:
            print(line)
            print(line[1])
            title.append(line[0])
            web.append(line[1])
            style.append(line[2])
            picture.append(line[3])
            description.append(line[4])

print(web)


pic_path = r"F:/爬虫专题/test1/audio批量下载/pic/"

file_path = r"F:/爬虫专题/test1/audio批量下载/"

for tit, we, pi, desc in zip(title, web, picture,
description):
    url = we
    header = {
        # 'Accept': '*/*',
        # 'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
        'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/100.0.4896.60 Safari/537.36",
        "referer": url
    }
```

```
    # 请求html页面内容（包含音频和视频的实际url）
    response = requests.get(url, headers=header)
    html = response.text
    time.sleep(2)
    # print(html)
    response.close()


        # 利用正则表达式定位对应包含目标url的标签的内容（缩小范围）
        obj_info = re.compile(r"<script>window\.__playinfo__=
{(?P<info>.*?)</script>", re.S)
        info = obj_info.findall(html)[0]
        # print(info)

        # 精确地从html（js）页面中获取音频和视频的url
        # video_url = re.findall(r'"video":.*?,"baseUrl":"
(.*?)","', info)[0]
        audio_url = re.findall(r'"audio":.*?,"baseUrl":"
(.*?)",', info)[0]
        # print("video_url:")
        # print(video_url)
        print("audio_url:")
        print(audio_url)

        # 请求资源下载
        # video = requests.get(video_url, headers=header)
        audio = requests.get(audio_url, headers=header)

        with open('./input/' + tit + '.mp3', mode='wb') as f:
            f.write(audio.content)
```

```
        time.sleep(2)
        # 睡3秒防止被当成爬虫，然后记得关闭和服务器的会话
        # video.close()
        audio.close()
        print("./input/" + tit + ".mp3  download  over!!!\n")


        # 获取图片

        response_pic = requests.get(pi, headers=header)

        obj_bvid = re.compile(r'/video/(?P<bvid>.*?)$', re.S)
        bvid = re.search(obj_bvid, we).group("bvid")

        print("bvid : " + bvid)

        with open(pic_path + bvid + ".jpg", "wb") as f:
            f.write(response_pic.content)

        print("\n" + pic_path + " -------> " + bvid + ".jpg
download  over!!!\n\n")


        sound = AudioSegment.from_file(file_path + 'input/' +
tit + '.mp3',
                                       frame_rate=44100,
                                       channels=2,
                                       sample_width=2)
            # , frame_rate=44100, channels=2, sample_width=2

        print("./output/" + tit + ".mp3")
```

```
        print("./pic/" + bvid + ".jpg")
        file_handle = sound.export(file_path + 'output/' +
tit + '.mp3',
                                   format="mp3",
                                   bitrate="320k",
                                   # 相当于极高音质 320kbit/s
                                   tags={"comments": desc},
                                   cover="./pic/" + bvid +
".jpg"
                                   )

        print("\n  封面添加完成   !!!\n")
```