71086032-曾诗仪-第十三周作业

协程有时被称为“微线程”，因为二者有类似的使用场景。和线程相比，协程占用资源更少、切换迅速，且实现简单（如协程是协作式调用，一般不用考虑资源的抢占，所以大部分情况下不需要通过同步原语来避免冲突）。通常协程被用在高并发的IO场景中。

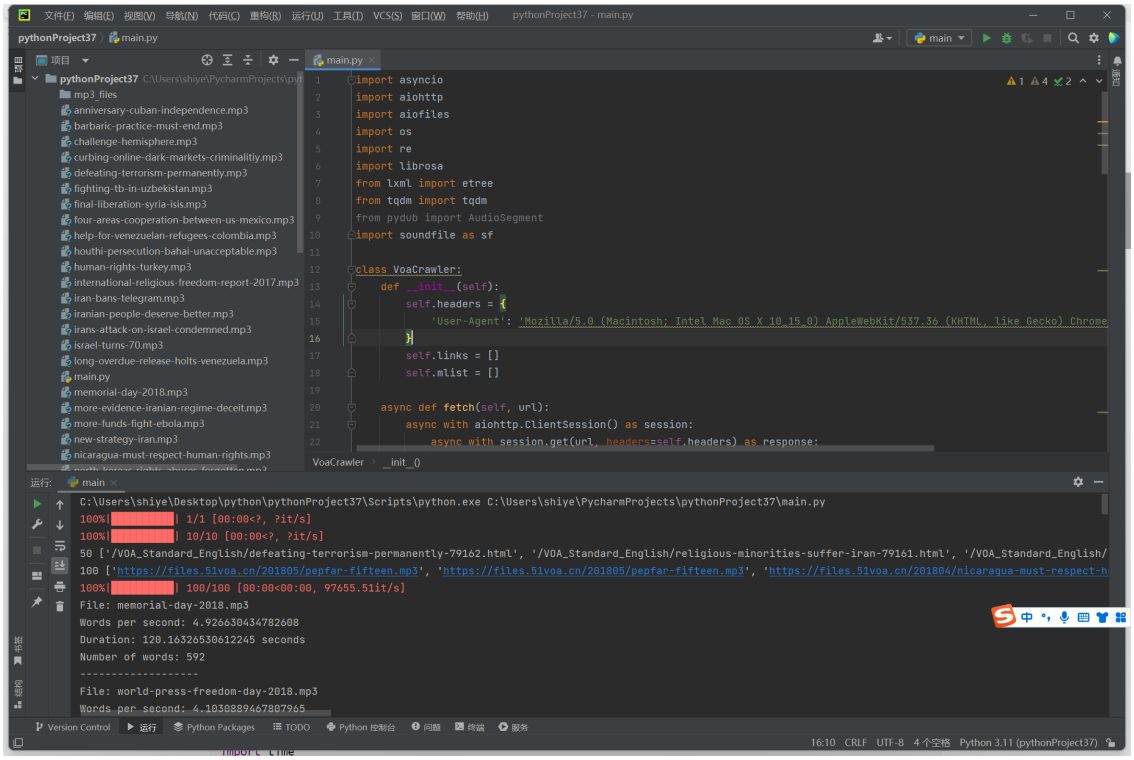
本周要求：

1. 利用协程，实现一个协程爬虫类VoaCrawler，对第十二周作业的数据采集部分进行重新实现，即用协程实现Mp3音频文件的爬取（可以使用gevent，aiohttop等第三方库），用aiofiles实现音频文件的存储。

```
import asyncio
import aiohttp
import aiofiles
import os
import re
import librosa
from lxml import etree
from tqdm import tqdm
from pydub import AudioSegment
import soundfile as sf

class VoaCrawler:
    def __init__(self):
        self.headers = {
            'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36'
        }
        self.links = []
        self.mlist = []

    async def fetch(self, url):
        async with aiohttp.ClientSession() as session:
            async with session.get(url, headers=self.headers) as response:
                return await response.text()

    async def get_links(self, page_num):
        url = f'https://www.51voa.com/VOA_Standard_{page_num}.html'
        html = await self.fetch(url)
        tree = etree.HTML(html)
        links = tree.xpath('//div[@id="righter"]//ul/li/a/@href')
        self.links.extend(links)

    async def get_mp3_links(self, link):
        url = f'https://www.51voa.com{link}'
        html = await self.fetch(url)
        mp3_links = re.findall(r'https://.+?\.mp3', html)
        self.mlist.extend(mp3_links)

    async def download_mp3(self, murl):
        async with aiohttp.ClientSession() as session:
```

```
            async with session.get(murl, headers=self.headers) as response:
                mp3_stream = await response.read()
                fname = os.path.basename(murl)
                async with aiofiles.open(fname, 'wb') as f:
                    await f.write(mp3_stream)

                await self.calculate_speech_rate(fname)

    async def calculate_speech_rate(self, filename):
        try:
            y, sr = librosa.load(filename, sr=None)
            number_of_words = len(librosa.onset.onset_detect(y=y, sr=sr,
units="time", hop_length=128, backtrack=False))
            duration = len(y) / sr
            words_per_second = number_of_words / duration
            print(f'File: {filename}')
            print(f'Words per second: {words_per_second}')
            print(f'Duration: {duration} seconds')
            print(f'Number of words: {number_of_words}')
            print('--------------------')
        except (librosa.LibrosaError, sf.SoundFileError) as e:
            print(f'Error calculating speech rate for {filename}: {str(e)}')

    async def run(self):
        tasks = []
        for i in tqdm(range(3, 4)):
            task = asyncio.create_task(self.get_links(i))
            tasks.append(task)
        await asyncio.gather(*tasks)

        print(len(self.links), self.links[:10])

        tasks = []
        for i in tqdm(range(0, len(self.links), 5)):
            link_batch = self.links[i:i + 5]
            tasks.extend([self.get_mp3_links(link) for link in link_batch])
        await asyncio.gather(*tasks)

        print(len(self.mlist), self.mlist[:10])

        tasks = []
        for murl in tqdm(self.mlist):
            task = asyncio.create_task(self.download_mp3(murl))
            tasks.append(task)
        await asyncio.gather(*tasks)

if __name__ == "__main__":
    crawler = VoaCrawler()
    loop = asyncio.get_event_loop()
    loop.run_until_complete(crawler.run())
```

输出结果：



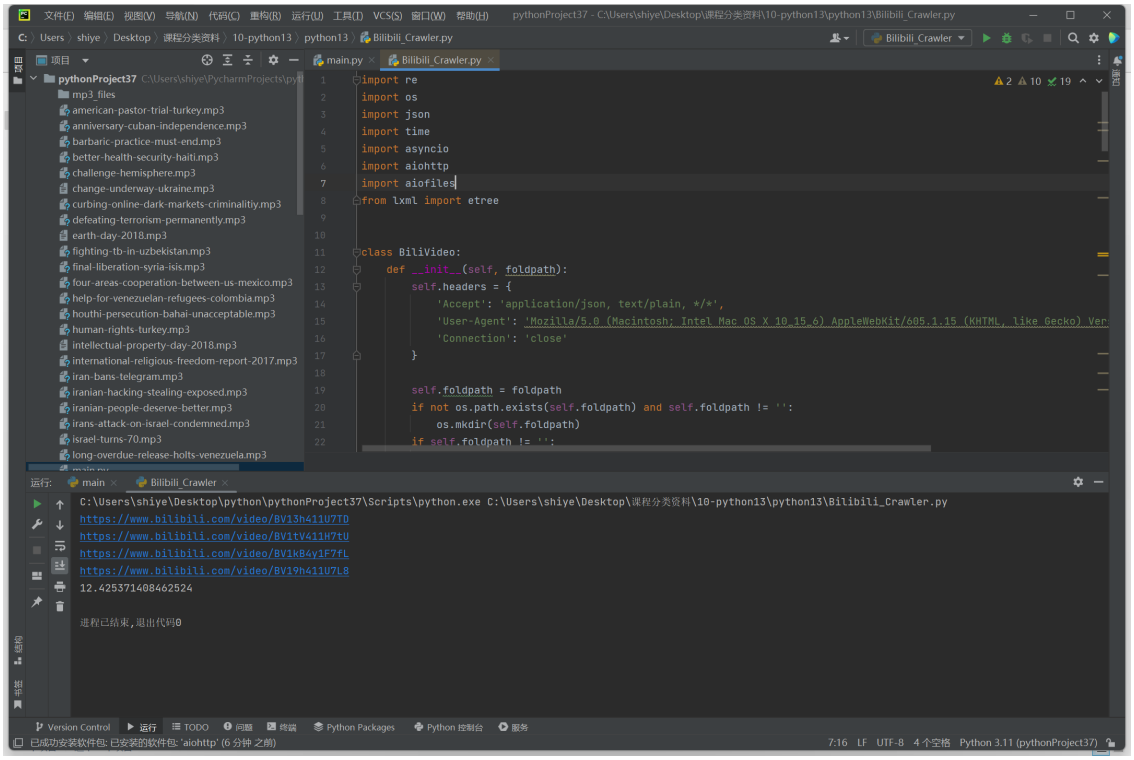
[输出结果.pdf]

2. 如果有余力，也可以继续改进Bilibili_Crawler.py中的例子，使用aiofiles等进行文件存储，并测试大规模视频同时获取是否可行。

```
import re
import os
import json
import time
import asyncio
import aiohttp
import aiofiles
from lxml import etree


class BiliVideo:
    def __init__(self, foldpath):
        self.headers = {
            'Accept': 'application/json, text/plain, */*',
            'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.2 Safari/605.1.15',
            'Connection': 'close'
        }

        self.foldpath = foldpath
        if not os.path.exists(self.foldpath) and self.foldpath != '':
            os.mkdir(self.foldpath)
        if self.foldpath != '':
            self.foldpath = self.foldpath + '/'

    # 得到页面内容
    async def get_response(self, url, headers, type='text'):
        async with aiohttp.ClientSession(
                connector=aiohttp.TCPConnector(limit=64, ssl=False,
keepalive_timeout=5)) as client:
```

```
            async with await client.get(url, headers=headers) as rsp:
                if type == 'text':
                    content = await rsp.text()
                elif type == 'content':
                    content = await rsp.read()
                else:
                    print('error input')
        return content

    # 解析页面
    async def parse_page_video(self, content):
        pagetree = etree.HTML(content)

        title = pagetree.xpath('//*[@id="viewbox_report"]/h1/@title')[0]

        pattern = r'\<script\>window\.__playinfo__=(.*?)\</script\>'
        temp = re.findall(pattern, content)[0]
        video_content = json.loads(temp)
        if video_content['data'] is not None:
            if 'dash' in video_content['data'].keys():
                for item in video_content['data']['dash']['video']:
                    if 'baseUrl' in item.keys():
                        video_url = item['baseUrl']
                        continue
                for item in video_content['data']['dash']['audio']:
                    if 'baseUrl' in item.keys():
                        audio_url = item['baseUrl']
                        continue
                return {
                    'title': title,
                    'video_url': video_url,
                    'audio_url': audio_url
                }
            else:
                print(video_content)
                return 0

    # 下载视频
    async def download_video(self, videoinfo, foldpath, videoheaders):
        title = videoinfo['title']
        vfilename = foldpath + '/' + title + '.m4a'
        afilename = foldpath + '/' + title + '.mp3'
        async with aiohttp.ClientSession(
                connector=aiohttp.TCPConnector(limit=64, ssl=False,
keepalive_timeout=5)) as client:
            async with client.get(videoinfo['video_url'],
headers=videoheaders) as v_rsp:
                v_content = await v_rsp.read()
                async with aiofiles.open(vfilename, 'wb') as v_file:
                    await v_file.write(v_content)

            async with client.get(videoinfo['audio_url'],
headers=videoheaders) as a_rsp:
                a_content = await a_rsp.read()
                async with aiofiles.open(afilename, 'wb') as a_file:
```

```
                    await a_file.write(a_content)

    # 对单个bv的下载
    async def _download_video(self, bv):
        url = 'https://www.bilibili.com/video/' + bv
        bvfold = self.foldpath + bv
        if not os.path.exists(bvfold):
            os.mkdir(bvfold)
        content = await self.get_response(url, self.headers)
        print(url)
        data = await self.parse_page_video(content)
        videoheaders = {
            'Origin': 'https://www.bilibili.com',
            'Referer': url,
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
            'Connection': 'close'
        }
        await self.download_video(data, bvfold, videoheaders)
        await asyncio.sleep(10)

    # 异步
    async def aio_main(self, bvlist):
        tasks = []
        for bv in bvlist:
            tasks.append(self._download_video(bv))
        await asyncio.gather(*tasks)

    def run(self, bvlist):
        asyncio.run(self.aio_main(bvlist))


if __name__ == '__main__':
    starttime = time.time()
    bvlist = [
        'BV1kB4y1F7fL',
        'BV13h411U7TD',
        'BV1tV411H7tU',
        'BV19h411U7L8'
    ]
    foldpath = "./bvdemo"
    BiliVideo(foldpath).run(bvlist)
    duration = time.time() - starttime
    print(duration)
```


文件(F) 编辑(E) 视图(V) 导航(N) 代码(C) 重构(R) 运行(U) 工具(T) VCS(S) 窗口(W) 帮助(H)
pythonProject37 - C:\Users\shiye\Desktop\课程分类资料\10-python13\python13\Bilibili_Crawler.py
C: Users shiye Desktop 课程分类资料 10-python13 python13 Bilibili_Crawler.py
Bilibili_Crawler
项目
pythonProject37 C:\Users\shiye\PycharmProjects\pyt
mp3_files
american-pastor-trial-turkey.mp3
anniversary-cuban-independence.mp3
barbaric-practice-must-end.mp3
better-health-security-haiti.mp3
challenge-hemisphere.mp3
change-underway-ukraine.mp3
curbing-online-dark-markets-criminalitiy.mp3
defeating-terrorism-permanently.mp3
earth-day-2018.mp3
fighting-tb-in-uzbekistan.mp3
final-liberation-syria-isis.mp3
four-areas-cooperation-between-us-mexico.mp3
help-for-venezuelan-refugees-colombia.mp3
houthi-persecution-bahai-unacceptable.mp3
human-rights-turkey.mp3
intellectual-property-day-2018.mp3
international-religious-freedom-report-2017.mp3
iran-bans-telegram.mp3
iranian-hacking-stealing-exposed.mp3
iranian-people-deserve-better.mp3
irans-attack-on-israel-condemned.mp3
israel-turns-70.mp3
long-overdue-release-holts-venezuela.mp3
main.py
Bilibili_Crawler.py
import re
import os
import json
import time
import asyncio
import aiohttp
import aiofiles
from lxml import etree
class BiliVideo:
def __init__(self, foldpath):
self.headers = {
'Accept': 'application/json, text/plain, */*',
'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Ver
'Connection': 'close'
}
self.foldpath = foldpath
if not os.path.exists(self.foldpath) and self.foldpath != '':
os.mkdir(self.foldpath)
if self.foldpath != '':
运行:
main
Bilibili_Crawler
C:\Users\shiye\Desktop\python\pythonProject37\Scripts\python.exe C:\Users\shiye\Desktop\课程分类资料\10-python13\python13\Bilibili_Crawler.py
https://www.bilibili.com/video/BV13h411U7TD
https://www.bilibili.com/video/BV1tV411H7tU
https://www.bilibili.com/video/BV1kB4y1F7fL
https://www.bilibili.com/video/BV19h411U7L8
12.425371408462524
进程已结束,退出代码0
Version Control
运行
TODO
问题
终端
Python Packages
Python 控制台
服务
已成功安装软件包: 已安装的软件包: 'aiohttp' (6 分钟 之前)
7:16 LF UTF-8 4 个空格 Python 3.11 (pythonProject37)