71086032-曾诗仪-第十二周作业

利用Python的多线程实现音频文件的采集，并计算语速。

0. import

```
import requests
from tqdm import tqdm
from lxml import etree
import threading
import re
import os
import librosa
import matplotlib.pyplot as plt
import sys
```

1. 通过类继承，实现一个线程类，参考voa.py中的示例，从https://www.51voa.com/VOA_Standard_3.html (其中 "3"可被替换为其他数字，对应翻页操作)中获取新的链接地址列表。

```
class LinkThread(threading.Thread):
    def __init__(self, page_num):
        super(LinkThread, self).__init__()
        self.page_num = page_num
        self.links = []

    def run(self):
        url = f'https://www.51voa.com/VOA_Standard_{self.page_num}.html'
        response = requests.get(url)
        html = etree.HTML(response.text)
        for j in range(0, 50):
            self.links += html.xpath(f'//*
[@id="righter"]/div[3]/ul/li[{j}]/a/@href')

class MP3Thread(threading.Thread):
    def __init__(self, links):
        super(MP3Thread, self).__init__()
        self.links = links
        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.mlist = []

    def run(self):
        for link in self.links:
            url = 'https://www.51voa.com' + link
            response = requests.get(url, headers=self.headers)
            self.mlist += list(set(re.findall(r'https://.+?\.mp3',
response.text)))
```

```
class DownloadThread(threading.Thread):
    def __init__(self, mlist):
        super(DownloadThread, self).__init__()
        self.mlist = mlist
        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'
        }

    def run(self):
        for murl in self.mlist:
            mp3_stream = requests.get(murl, headers=self.headers).content
            fname = murl[murl.rfind('/') + 1:]
            with open(fname, 'wb') as f:
                f.write(mp3_stream)

# 主程序
links = []
threads = []
for i in tqdm(range(3, 4)):
    thread = LinkThread(i)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    links += thread.links

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

mlist = []
threads = []
for i in tqdm(range(0, len(links), 5)):
    thread = MP3Thread(links[i:i+5])
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    mlist += thread.mlist

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

threads = []
for i in tqdm(range(0, len(mlist), 5)):
    thread = DownloadThread(mlist[i:i+5])
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
```

2. 通过类继承，实现一个线程类，参考voa.py中的示例，从1中获取的链接（如https://www.51voa.com/VOA_Standard_English/u-s-supports-diversity-of-energy-sources-in-europe-79541.html）获取mp3文件链接。

```
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'
}

class LinkThread(threading.Thread):
    def __init__(self, page_num):
        super(LinkThread, self).__init__()
        self.page_num = page_num
        self.links = []

    def run(self):
        url = f'https://www.51voa.com/VOA_Standard_{self.page_num}.html'
        response = requests.get(url)
        html = etree.HTML(response.text)
        self.links = html.xpath('//div[@id="righter"]//ul/li/a/@href')

class MP3Thread(threading.Thread):
    def __init__(self, links):
        super(MP3Thread, self).__init__()
        self.links = links
        self.mlist = []

    def run(self):
        for link in self.links:
            url = 'https://www.51voa.com' + link
            response = requests.get(url, headers=headers)
            self.mlist += list(set(re.findall(r'https://.+?\.mp3',
response.text)))

# 获取链接地址列表
links = []
threads = []
for i in tqdm(range(3, 4)):
    thread = LinkThread(i)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    links += thread.links

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

# 获取MP3文件链接
mlist = []
threads = []
for i in tqdm(range(0, len(links), 5)):
    thread = MP3Thread(links[i:i+5])
    thread.start()
    threads.append(thread)
```

```
for thread in threads:
    thread.join()
    mlist += thread.mlist

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

# 下载MP3文件
for murl in tqdm(mlist):
    mp3_stream = requests.get(murl, headers=headers).content
    fname = os.path.basename(murl)
    with open(fname, 'wb') as f:
        f.write(mp3_stream)
```

3. 通过类继承，实现一个线程类，参考voa.py中的示例，利用2中的mp3文件链接（如https://files.51voa.cn/201806/fighting-tb-in-uzbekistan.mp3），将文件保存到本地。

```
class DownloadThread(threading.Thread):
    def __init__(self, murl):
        super().__init__()
        self.murl = murl

    def run(self):
        mp3_stream = requests.get(self.murl, headers=headers).content
        fname = os.path.basename(self.murl)
        with open(fname, 'wb') as f:
            f.write(mp3_stream)

# 获取链接地址列表
links = []
threads = []
for i in tqdm(range(3, 4)):
    thread = LinkThread(i)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    links += thread.links

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

# 获取MP3文件链接
mlist = []
threads = []
for i in tqdm(range(0, len(links), 5)):
    thread = MP3Thread(links[i:i+5])
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    mlist += thread.mlist

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

# 下载MP3文件
download_threads = []
for murl in tqdm(mlist):
    thread = DownloadThread(murl)
    thread.start()
    download_threads.append(thread)

for thread in download_threads:
    thread.join()
```

4. 通过类继承，实现一个线程类，参考sr.py中的示例，对存储的音频文件计算语速。

```
import requests
from tqdm import tqdm
from lxml import etree
import threading
import re
import os
import librosa
import matplotlib.pyplot as plt
import sys

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'
}


class LinkThread(threading.Thread):
    def __init__(self, page_num):
        super().__init__()
        self.page_num = page_num
        self.links = []

    def run(self):
        url = f'https://www.51voa.com/VOA_Standard_{self.page_num}.html'
        response = requests.get(url)
        html = etree.HTML(response.text)
        self.links = html.xpath('//div[@id="righter"]//ul/li/a/@href')


class MP3Thread(threading.Thread):
    def __init__(self, links):
        super().__init__()
        self.links = links
        self.mlist = []

    def run(self):
        for link in self.links:
            url = 'https://www.51voa.com' + link
            response = requests.get(url, headers=headers)
            self.mlist += list(set(re.findall(r'https://.+?\.mp3',
response.text)))


class DownloadThread(threading.Thread):
    def __init__(self, murl):
        super().__init__()
        self.murl = murl

    def run(self):
        mp3_stream = requests.get(self.murl, headers=headers).content
        fname = os.path.basename(self.murl)
        with open(fname, 'wb') as f:
            f.write(mp3_stream)

        # Call speechrate function
        self.calculate_speech_rate(fname)

    def calculate_speech_rate(self, filename):
        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('-------------------')


# 获取链接地址列表
links = []
threads = []
for i in tqdm(range(3, 4)):
    thread = LinkThread(i)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
    links += thread.links

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

# 获取MP3文件链接
mlist = []
threads = []
for i in tqdm(range(0, len(links), 5)):
    thread = MP3Thread(links[i:i + 5])
    thread.start()
    threads.append(thread)
```

```
for thread in threads:
    thread.join()
    mlist += thread.mlist

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

# 下载MP3文件
download_threads = []
for murl in tqdm(mlist):
    thread = DownloadThread(murl)
    thread.start()
    download_threads.append(thread)

for thread in download_threads:
    thread.join()
```

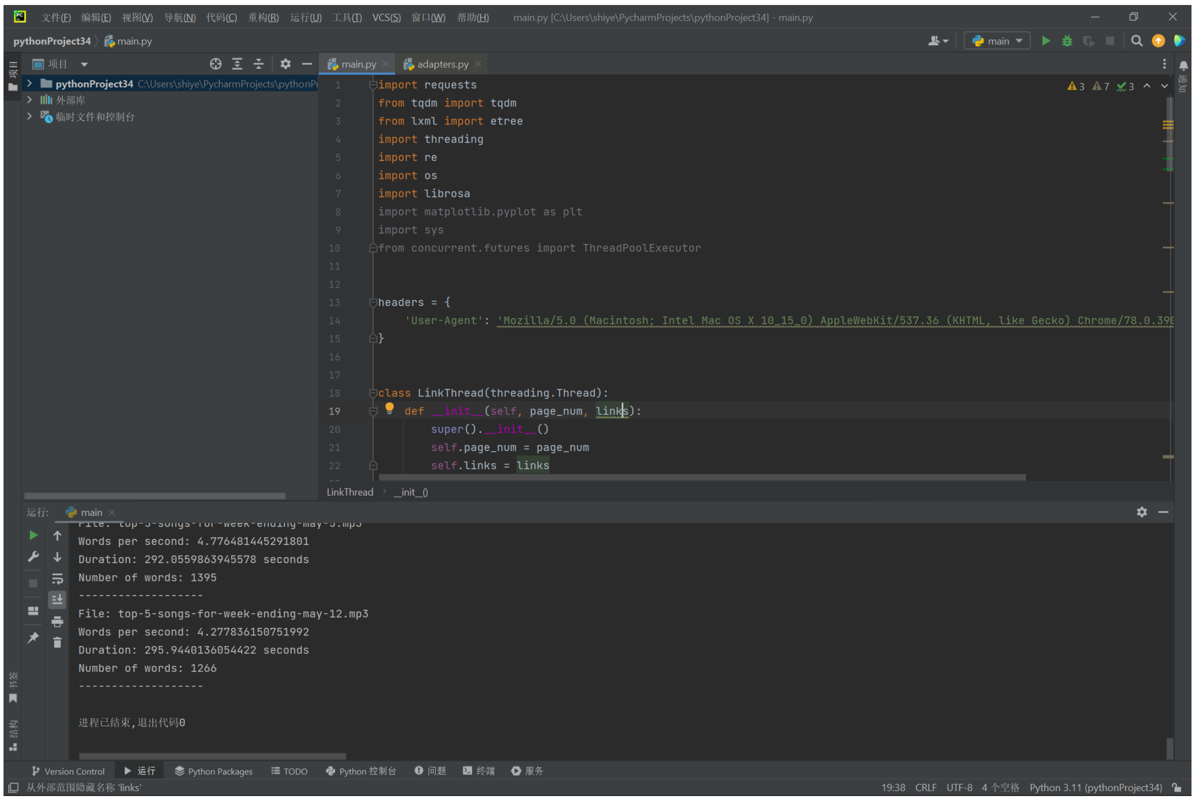
输出结果： 4. 输出结果.pdf

5. 设计一种同步策略（比如用线程池，或锁，或队列等），实现1，2，3，4中几种不同功能线程的配合，实现多线程的mp3文件下载功能，并进行语速的计算和输出。

```
import requests
from tqdm import tqdm
from lxml import etree
import threading
import re
import os
import librosa
import matplotlib.pyplot as plt
import sys
from concurrent.futures import ThreadPoolExecutor

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'
}


class LinkThread(threading.Thread):
    def __init__(self, page_num, links):
        super().__init__()
        self.page_num = page_num
        self.links = links

    def run(self):
        url = f'https://www.51voa.com/VOA_Standard_{self.page_num}.html'
        response = requests.get(url)
        html = etree.HTML(response.text)
        self.links.extend(html.xpath('//div[@id="righter"]//ul/li/a/@href'))


class MP3Thread(threading.Thread):
    def __init__(self, links, mlist):
        super().__init__()
        self.links = links
        self.mlist = mlist

    def run(self):
        for link in self.links:
            url = 'https://www.51voa.com' + link
            response = requests.get(url, headers=headers)
            self.mlist.extend(list(set(re.findall(r'https://.+?\.mp3',
response.text))))


class DownloadThread(threading.Thread):
    def __init__(self, murl):
        super().__init__()
        self.murl = murl

    def run(self):
        mp3_stream = requests.get(self.murl, headers=headers).content
        fname = os.path.basename(self.murl)
        with open(fname, 'wb') as f:
            f.write(mp3_stream)

        # Call speechrate function
        self.calculate_speech_rate(fname)

    def calculate_speech_rate(self, filename):
        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('-------------------')


# 获取链接地址列表
links = []
threads = []
for i in tqdm(range(3, 4)):
    thread = LinkThread(i, links)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()

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

# 获取MP3文件链接
mlist = []
```

```
threads = []
for i in tqdm(range(0, len(links), 5)):
    thread = MP3Thread(links[i:i + 5], mlist)
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()

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

# 下载MP3文件
download_threads = []
for murl in tqdm(mlist):
    thread = DownloadThread(murl)
    thread.start()
    download_threads.append(thread)

for thread in download_threads:
    thread.join()
```

输出结果： 5.输出.pdf



```
import requests
from tqdm import tqdm
from lxml import etree
import threading
import re
import os
import librosa
import matplotlib.pyplot as plt
import sys
from concurrent.futures import ThreadPoolExecutor


headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.39
}


class LinkThread(threading.Thread):
    def __init__(self, page_num, links):
        super().__init__()
        self.page_num = page_num
        self.links = links
```

```
Words per second: 4.776481445291801
Duration: 292.0559863945578 seconds
Number of words: 1395
-------------------
File: top-5-songs-for-week-ending-may-12.mp3
Words per second: 4.277836150751992
Duration: 295.9440136054422 seconds
Number of words: 1266
-------------------

进程已结束,退出代码0
```

从外部范围隐藏名称 'links'

6. （附加）考虑到Python的多进程更适合计算密集型的任务，可否在4中不使用线程，而使用多进程来计算存储文件的语速？即主进程的多线程负责音频文件的下载与存储，另几个子进程则负责语速计算。比较一下与5中线程设计的速度差异。
7. （附加）根据以下要求修改以上代码：音频采集如果时间过长，可能会因为外部因素（网络等）出现意外的中断，可否设计一种"断点"的记录机制，使得即使采集过程被打断，也能够从上次的断点继续开始，避免重复采集。