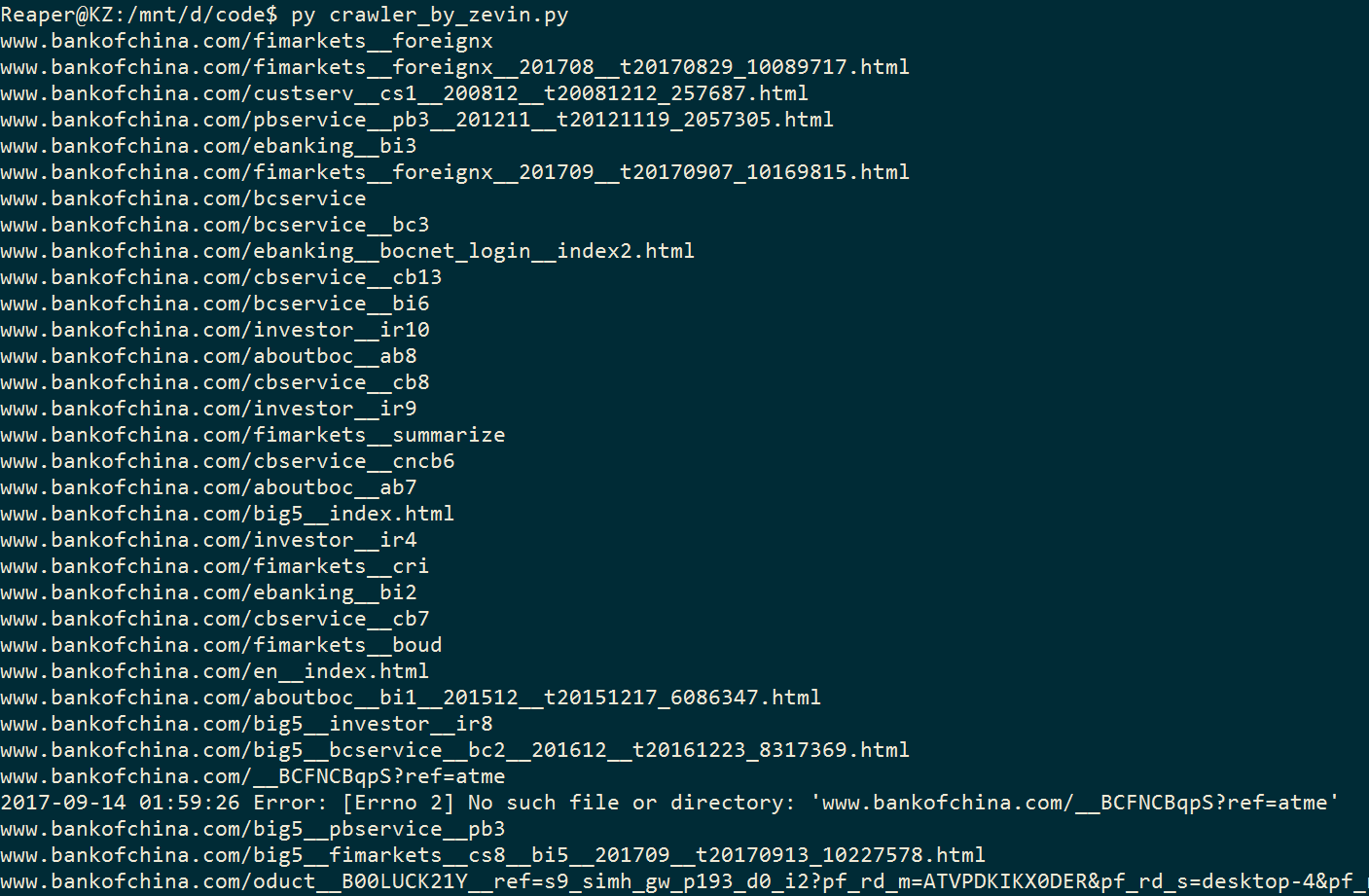
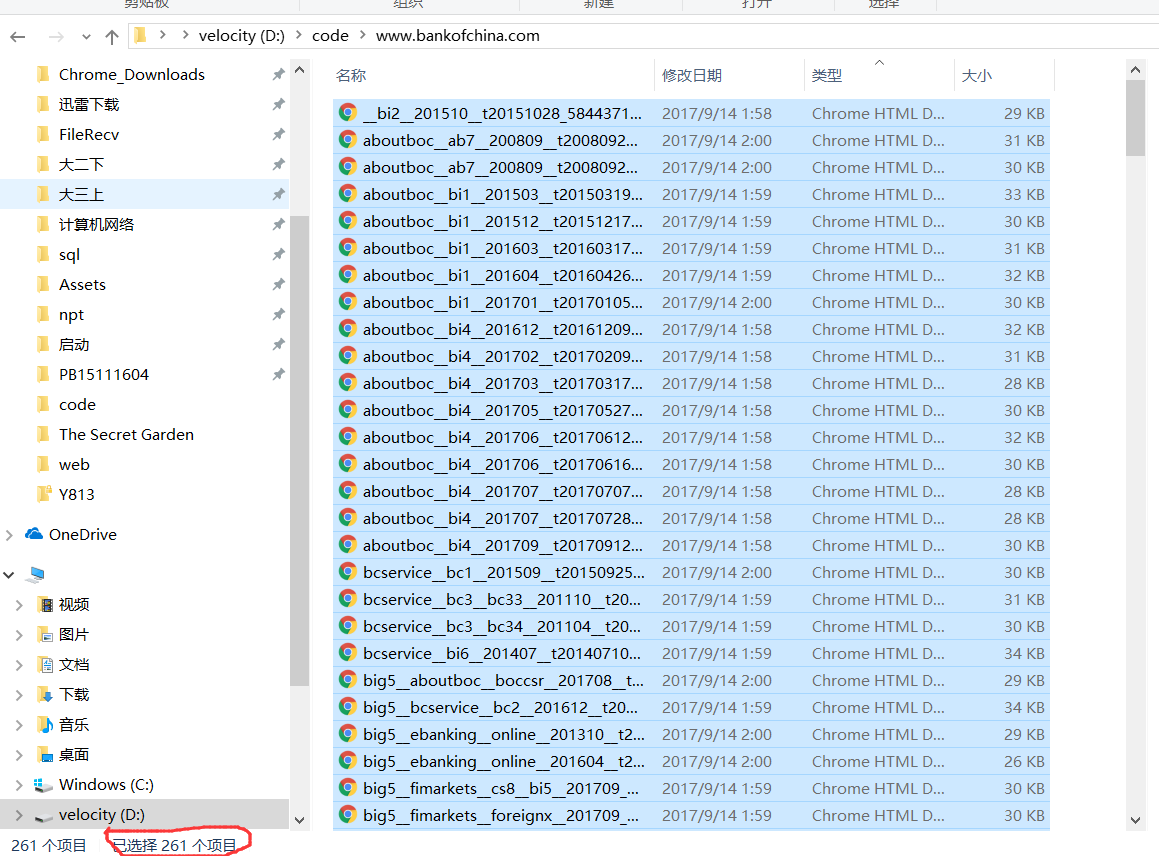
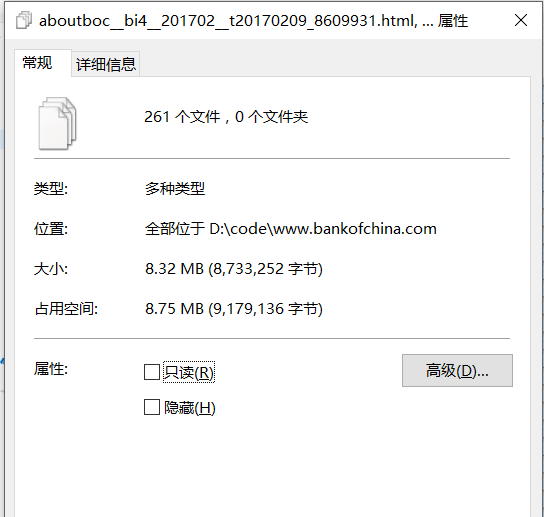
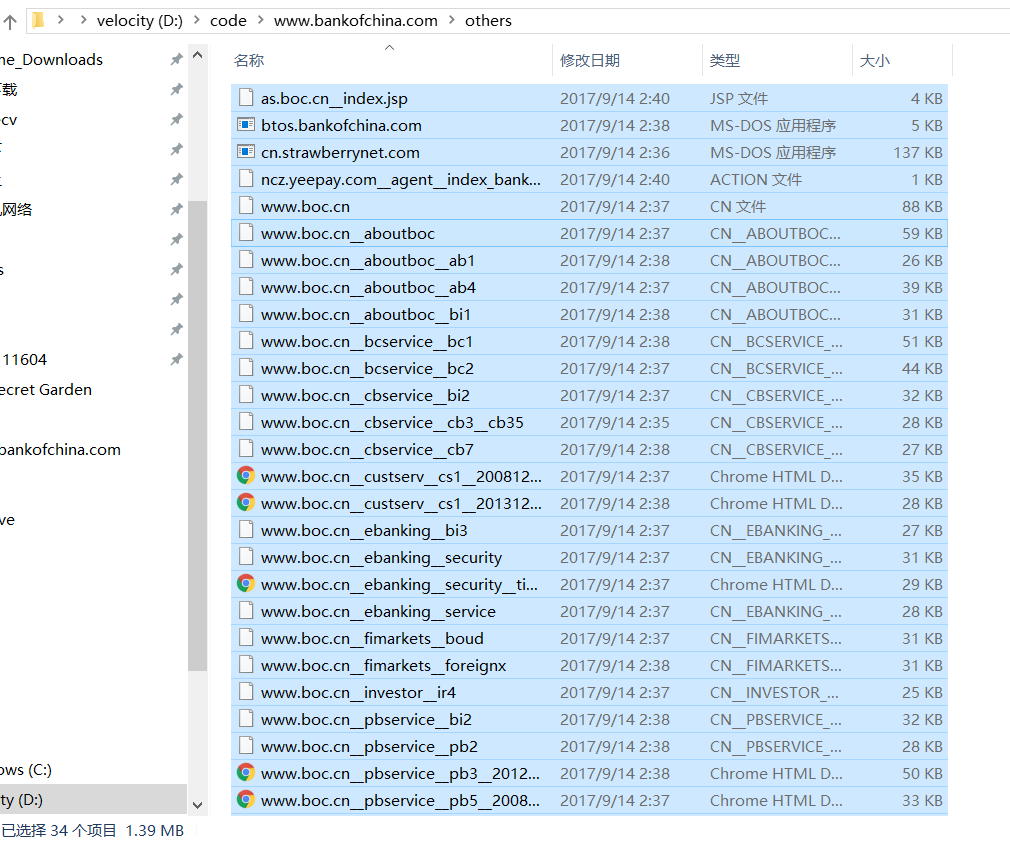
|  |
| --- |
|  |
| HW1.2-爬虫实践 |
|  |
| PB15111604金泽文 |

[**Web Information Processing and Applications**](http://staff.ustc.edu.cn/~jpq/courses/webinfo.html)

HW1.2-爬虫实践

PB15111604金泽文

1. 概述
   1. 目标站点：[中国银行外汇市场](http://www.bankofchina.com/fimarkets/foreignx/)
   2. 是否自己完成：是
   3. 语言：python3
   4. 层数：3层（可以很多层，为实验仅设置3层）
2. 设计
   1. 获得html
      1. 通过urllib的request模块直接获取html
      2. 存储html文件。为了存储，需要将url改为合法的文件名，中国银行主域名[www.bankofchina.com/](http://www.bankofchina.com/) 后的‘/’一律改为‘\_\_’（两个下划线）；不是其子域名的单独处理，妨碍子目录路中的others/ 中。为此我设计了rename\_save()函数实现。
   2. 解析html得到a href
      1. 通过html.parser的HTMLParser模块，以及自己增加的解析规则ZevinParser，解析html流，分离出超链接，并存入集合中。（为了避免重复，所以使用集合。）
      2. 针对html中不同的url结构，比如‘#’，‘../../’等，依次根据目前的url进行修改，修改为完整的url。为了这一部分rename，我设计了ulrparse(),rename\_open(),rename()函数
   3. 依次递归，得到3层html。
3. 爬取过程
   1. 创建[www.bankofchina.com/](http://www.bankofchina.com/) 目录（已创建）
   2. 运行脚本crawler\_by\_zevin.py
4. 爬取结果
   1. 一共爬取261个项目，其中1个pdf文件。
   2. 一共8+MB内容。
   3. 预览结果：
   4. 其他域名：
   5. 最后，可以看出目录结构清晰。内容完整。不过没有考虑的是对于更多层次的数据爬取的效率，以及对内存的需求。
   6. 老师说会的人只要1个小时就能搞定了，而第一次写爬虫的我写了4个小时。不过没有看任何爬虫教程，还是很有成就感的（虽然确实很简单）！

附录 源代码：

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

from urllib import request

from html.parser import HTMLParser

import time

unfinished = set()

present\_url = 'www.bankofchina.com/fimarkets/foreignx/'

domain = 'www.bankofchina.com'

fget = 0

def urlparse(string\_to\_rename):

index = string\_to\_rename.find('../')

if index == -1:

return string\_to\_rename

else:

lindex = string\_to\_rename[:index-1].rfind('/')

return urlparse(string\_to\_rename[:lindex] + string\_to\_rename[index + 2:])

def rename\_open(present\_url, string\_to\_rename):

if string\_to\_rename == '#' or string\_to\_rename == './':

return present\_url

elif string\_to\_rename.startswith('/'):

return domain + string\_to\_rename

elif string\_to\_rename.startswith('./'):

if present\_url.endswith('/'):

return urlparse(urlparse(present\_url + string\_to\_rename[2:]))

else:

index = present\_url.rfind('/')

return urlparse(present\_url[:index] + string\_to\_rename[1:])

elif string\_to\_rename.startswith('../'):

if present\_url.endswith('/'):

index = present\_url[:-1].rfind('/')

return urlparse(urlparse(present\_url[:index] + string\_to\_rename[2:]))

else:

index = present\_url.rfind('/')

return urlparse(present\_url[:index] + string\_to\_rename[2:])

elif string\_to\_rename.startswith('http://'):

return rename\_open(present\_url, string\_to\_rename[7:])

else:

return string\_to\_rename

def rename\_save(string\_to\_rename):

if not 'www.bankofchina.com' in string\_to\_rename:

s = string\_to\_rename.replace('/','\_\_')

else:

index = 'www.bankofchina.com/fimarkets/foreignx/'.find('/')

s = string\_to\_rename[index+1:].replace('/','\_\_')

if s.endswith('\_\_'):

s = s[:-2]

if '?' in s:

index = s.find('?')

s = s[:index]

return s

def rename(zp):

tmp = set()

for l in zp.links:

tmp.add(rename\_open(present\_url, l))

zp.links = tmp

class ZevinParser(HTMLParser):

def \_\_init\_\_(self):

HTMLParser.\_\_init\_\_(self)

self.links = set()

def handle\_starttag(self, tag, attrs):

if tag == 'a' and len(attrs) != 0:

for key, value in attrs:

if(key == 'href'):

self.links.add(value)

def get\_html():

global fget

fget = request.urlopen('http://' + present\_url)

fget = fget.read()

tmp = rename\_save(present\_url)

if 'www.' in tmp or '.com' in tmp or '.cn' in tmp:

location = 'www.bankofchina.com/others/' + tmp

fsave = open(location, 'wb')

else:

location = 'www.bankofchina.com/' + tmp

fsave = open(location, 'wb')

print(location)

fsave.write(fget)

fsave.close()

fget = fget.decode()

# crawl:

get\_html()

zp = ZevinParser()

zp.feed(fget)

zp.close()

zp.links.add(present\_url)

rename(zp)

unfinished = zp.links

for url in zp.links:

present\_url = url

try:

get\_html()

except Exception as e:

if not isinstance(e, UnicodeDecodeError):

print (time.strftime('%Y-%m-%d %X', time.localtime(time.time())), 'Error:', str(e))

break

else:

break

tmpParser = ZevinParser()

tmpParser.feed(fget)

tmpParser.close()

rename(tmpParser)

unfinished = unfinished | tmpParser.links

unfinished = unfinished - zp.links

for url in unfinished:

present\_url = url

try:

get\_html()

except Exception as e:

if not isinstance(e, UnicodeDecodeError):

print (time.strftime('%Y-%m-%d %X', time.localtime(time.time())), 'Error:', str(e))

else:

break

# print('len:', len(zp.links))

# for i in zp.links:

# print('\t', i)

# def output():

# print('len:', len(zp.links))

# for i in zp.links:

# print('\t', i)

# print('----', rename\_open(present\_url, i))

# print('----', rename\_save(rename\_open(present\_url, i)))

# output()