Python学习之路-NLP（人物提取）

原创

m0_58998775 于 2021-06-11 18:00:18 发布 1377 收藏 1
分类专栏： Python 文章标签： python nlp
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/m0_58998775/article/details/117824196
版权


Python 专栏收录该内容
2 篇文章 0 订阅
订阅专栏

目标：

- 读取四大名著中的一部
- 按章节进行人名抽取，并排序
- 合并所有排序结果至最终结果字典中
- 显示最终结果

遇到问题：

- 【listdir 获取文件乱序】，【解决方法】加个按创建时间排序 st_mtime，但实际上Linux下并没有创建时间，实际上是最后修改时间

```
file_names.sort(key = lambda x:os.stat(dir + '/'+x).st_mtime)
```

- 【解析文件报错】因为是Mac电脑，解析文件的时候，最后总报错，经调试知道，原来是读取了系统生成的.DS_Store文件，【解决方法】加个文件名正则判断，是否以数字开头^[0-9]

```
#判断是否是章节(数字开头)，排除掉mydict.txt 和 .DS_Store 文件
        if re.findall("^[0-9]", file.split('/')[-1]) == []:
            continue
```

完整代码：

```
import os
import jieba.posseg as psg
import jieba
import time
import re
novel = 'hlm'
dir = './Text/'+novel
#jieba.load_userdict(dir+'/mydict.txt')

# 字典合并
def combineListToDic(list):
    if len(list) <= 1:
        return list
    temp = {}
```

```
    temp = {}
    for l in list:
        for k in l:
            temp = sumDic(temp,{k[0]:k[1]})
    return temp

def main():
    # 获取文件名
    file_names = os.listdir(dir)
    # listdir出来的文件是乱序
    # 根据创建时间排序（创建时间与章节同序）
    # file_names.sort(key = compare) 同下
    file_names.sort(key = lambda x:os.stat(dir + '/'+x).st_mtime)
    #print(file_names)
    # 文件名拼接路径
    file_list = [os.path.join('./Text/'+novel+'/', file) for file in file_names]
    #print(file_list)
    text = []
    # name_countList,存储所有章节独立的[人名+计数]
    name_countList = []
    for file in file_list:
        #判断是否是章节(数字开头)，排除掉mydict.txt 和 .DS_Store 文件
        if re.findall("^[0-9]", file.split('/')[-1]) == []:
            continue
        time_start = time.time()  # 开始计时
        f = open(file, encoding='utf-8')
        text = f.readlines()
        #分词+排序
        name_count = getNameCountListFromText(text)
        name_countList.append(name_count)
        time_end = time.time()  # 结束计时
        time_c = time_end - time_start  # 运行所花时间
        print(time_c,file,name_count)
    # 打印全部章节[人名+计数]
    for n in name_countList:
        print(len(n), n)
    # 人名合并
    name_countListDic =  combineListToDic(name_countList)
    # 排序
    name_count_total = sorted(name_countListDic.items(), key=lambda x: x[1], reverse=True)
    print('========人名频次总排名========')
    print(name_count_total)

# 根据文本内容，经过分词，提取人名并按出现次数排序
def getNameCountListFromText(text):
    # 分词
    # for t in text:
    #     res = psg.cut(t)
    #     print([(item.word, item.flag) for item in res])

    # 计数
    dict = {}
    for t in text:
        # 忽略空行
        if t.strip() == '':
            text.remove(t)
            continue
        res = psg.cut(t)
        for item in res:
            if item.flag == 'nr' and item.word in dict:
```

```
                dict[item.word] += 1
            elif item.flag == 'nr' and item.word not in dict:
                dict[item.word] = 1
    #print(dict)

    # 排序
    name_count = sorted(dict.items(), key=lambda x: x[1], reverse=True)
    return name_count

#根据创建时间排序
def compare(x):
    return os.stat(dir + '/'+x).st_ctime

#字典合并，相同key的count相加
def sumDic(dict1,dict2):
    temp = dict()
    # python3,dict_keys类似set； | 并集
    for key in dict1.keys() | dict2.keys():
        # 根据业务需求修改下面方法，
        temp[key] = sum([d.get(key, 0) for d in (dict1, dict2)])
    return temp

if __name__ == '__main__':
    main()
```

过程：

- 每篇文章大概处理400~600ms，但是章节数量多，整体单线程比较慢，下次学习一下多线程【已更新】
- 最后一遍合并排序比较慢，优化不是我的重点，暂时忽略

最终结果：

```
[('宝玉', 3737), ('贾母', 1250), ('凤姐', 1215), ('黛玉', 1074), ('王夫人', 975), ('老太太', 965), ('宝钗', 753
```

遗留问题：

1. 同人问题（林黛玉/林姑娘/林妹妹...）
2. XX曰、XX道、XX。。。
3. 自定义字典还没调试明白

多进程版本【20210616】

多进程，想使用返回值，通过队列（先进先出），其中，q需要传递到方法中

【遗留】多线程的没搞明白...两个问题：

1.进程间传值，不是引用的形式，回来值为空

2.jieba多线程速度并没有变快，反而更慢了，每个线程执行时间跟线程数成正比

```
q = multiprocessing.Queue(maxsize=0)
```

```
q.put（XXX）#存

q.get()#取
```

```
import multiprocessing
import os
import jieba.posseg as psg
import jieba
import time
import re
novel = 'xyj'
dir = './Text/'+novel
#jieba.load_userdict(dir+'/mydict.txt')

# 字典合并
def combineListToDic(list):
    if len(list) <= 1:
        return list
    temp = {}
    for l in list:
        for k in l:
            temp = sumDic(temp,{k[0]:k[1]})
    return temp


def getNameCountListFromFileList(file_list, q, pCount):
    i = 0
    while i < len(file_list):
        if i % 4 != pCount - 1:
            i += 1
            continue
        time_start = time.time()  # 开始计时
        f = open(file_list[i], encoding='utf-8')
        text = f.readlines()
        # 分词+排序
        name_count = getNameCountListFromText(text)
        q.put(name_count)
        time_end = time.time()  # 结束计时
        time_c = time_end - time_start  # 运行所花时间
        print('Thread-', pCount,':',time_c, file_list[i], name_count)
        i += 1
    else:
        print('Thread-', pCount, 'is done')

def main():
    # 获取文件名
    file_names = os.listdir(dir)
    # file_names = []
    # for root, dirs, files in os.walk(dir):
    #     file_names.append(files)
    # listdir出来的文件是乱序
    # 根据创建时间排序（创建时间与章节同序）
    # file_names.sort(key = compare) 同下
    file_names.sort(key = lambda x:os.stat(dir + '/'+x).st_mtime)
    #print(file_names)
    # 文件名拼接路径
```

```
    # 文件名拼接路径
    file_list = [os.path.join('./Text/'+novel+'/', file) for file in file_names]
    # 处理一下file_list中无效的文件
    i = 0
    while i < len(file_list):
        if re.findall("^[0-9]", file_list[i].split('/')[-1]) == []:
            file_list.remove(file_list[i])
            continue
        i += 1

    # name_countList,存储所有章节独立的[人名+计数]
    name_countList = []
    # 多进程改写
    q = multiprocessing.Queue(maxsize=0)
    p1 = multiprocessing.Process(target=getNameCountListFromFileList, args=(file_list, q, 1))
    p2 = multiprocessing.Process(target=getNameCountListFromFileList, args=(file_list, q, 2))
    p3 = multiprocessing.Process(target=getNameCountListFromFileList, args=(file_list, q, 3))
    p4 = multiprocessing.Process(target=getNameCountListFromFileList, args=(file_list, q, 4))
    p1.daemon = True
    p2.daemon = True
    p3.daemon = True
    p4.daemon = True
    p1.start()
    p2.start()
    p3.start()
    p4.start()
    # p1.join()
    # p2.join()
    # p3.join()
    # p4.join()
    for l in file_list:
        name_countList.append(q.get())

    # 打印全部章节[人名+计数]
    for n in name_countList:
        print( n)
    # 人名合并
    name_countListDic =  combineListToDic(name_countList)
    # 排序
    name_count_total = sorted(name_countListDic.items(), key=lambda x: x[1], reverse=True)
    print('========人名频次总排名========')
    print(name_count_total)

    # 将人名写到文件
    path = './Text/'+novel+'/PersonName'
    if os.path.exists(path) == False:#路径是否存在，不存在就创建
        os.makedirs(path)
    f = open(path+'/PersonName.txt', 'w')
    for i in name_count_total:
        f.write(str(i)+'\n')
    f.close()

# 根据文本内容，经过分词，提取人名并按出现次数排序
def getNameCountListFromText(text):
    # 计数
    dict = {}
    for t in text:
        # 忽略空行
        if t.strip() == '':
            text.remove(t)
```

```
            continue
        res = psg.cut(t)
        for item in res:
            if item.flag == 'nr' and item.word in dict:
                dict[item.word] += 1
            elif item.flag == 'nr' and item.word not in dict:
                dict[item.word] = 1
    #print(dict)

    # 排序
    name_count = sorted(dict.items(), key=lambda x: x[1], reverse=True)
    return name_count

#根据创建时间排序
def compare(x):
    return os.stat(dir + '/'+x).st_ctime

#字典合并，相同key的count相加
def sumDic(dict1,dict2):
    temp = dict()
    # python3,dict_keys类似set； | 并集
    for key in dict1.keys() | dict2.keys():
        # 根据业务需求修改下面方法，
        temp[key] = sum([d.get(key, 0) for d in (dict1, dict2)])
    return temp

if __name__ == '__main__':
    main()
```