****



**Python编程技术**

学院： 信息与电子工程学院

班级： 电子信息专硕2102班

姓名： 董超

学号： 2021420073

# Python编程技术

## **任务描述**

编写网络爬虫程序，爬取山东工商学院2015年1月1日之后的所有新闻，统计每年所有新闻中出现次数最多的10个词语并分别绘制柱状图显示。

## 代码

### Getadata.py

1. **import** requests
2. **from** lxml **import** etree
3. **from** datetime **import** datetime
4. '''
5. 说明:
6. 本部分完成的是数据的获取,通过endtime控制爬取数据的截止日期,将每条新闻分别保存到path目录下
7. '''
8. **if** \_\_name\_\_ == '\_\_main\_\_':
9. url = "https://www.sdtbu.edu.cn/index/ssyw.htm"
10. path = "result/"
11. endtime = "2015-01-01"
12. headers = {
13. 'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36'
14. }
15. TOPURL = "https://www.sdtbu.edu.cn/"
16. statue = {
17. 200: "OK",
18. 404: "Not Found"
19. }
21. i = 1
22. # 爬虫部分
23. **while** True:
24. **try**:
25. # 新闻列表页进行请求
26. response = requests.get(url=url, headers=headers)
27. response.encoding = response.apparent\_encoding
28. **print**(url, f"第{i}页:的爬取状态:", statue[response.status\_code])
30. # 创建解析器
31. html = response.text
32. parse = etree.HTML(html)
34. # 解析页面中的新闻详情页url
35. a\_path = "//div[4]/div[2]/ul/li/a/@href"  # 获取新闻详情页url
36. a\_lst = parse.xpath(a\_path)
37. t\_path = '//div[4]/div[2]/ul/li/a/@title'  # 获取新闻title
38. t\_lst = parse.xpath(t\_path)
39. ti\_path = '//div[4]/div[2]/ul/li/p/text()'  # 获取时间time
40. ti\_lst = parse.xpath(ti\_path)
42. # 对每个详情页面进行请求
43. **for** a, t, ti **in** zip(a\_lst, t\_lst, ti\_lst):
44. newurl = TOPURL + a[2:]
45. newresponse = requests.get(newurl, headers=headers)
46. newresponse.encoding = newresponse.apparent\_encoding
47. # print("\t", newurl, "详情页:", newresponse.status\_code)
48. newhtml = newresponse.text
50. # 创建解析器
51. newparse = etree.HTML(newhtml)
53. # 文字
54. infopath = '//\*[@id="vsb\_content"]//text()'
55. info = newparse.xpath(infopath)
57. text = ""
58. # 去除转义字符
59. **for** in\_ **in** info:
60. **if** in\_ == "\r\n" **or** in\_ == "\r\n  " **or** in\_ == "\ufeff" **or** in\_ == "\xa0\xa0":
61. **pass**
62. **else**:
63. text += in\_
64. t = t.replace(r'"', r'”')
66. y = datetime.strptime(ti, '%Y-%m-%d')  # 新闻发布的时间
67. z = datetime.strptime(endtime, '%Y-%m-%d')  # 截止日期
69. diff = z - y
70. chazhi = diff.days \* 86400 + diff.seconds  # 时间差
72. # 设置时间限制
73. **if** chazhi <= 0:
74. # print("y = ", y, "diff = ", diff)
75. filepath = f"{path}" + ti + " " + t + ".txt"
77. with open(filepath, encoding=newresponse.encoding, mode="w") as fp:
78. fp.write(text)
79. **else**:
80. exit(0)
82. # 获取"下页"的url
83. nextpath = '//table//div/a[@class="Next"]/@href'
84. nexturl = parse.xpath(nextpath)
85. url = TOPURL + "index/ssyw/" + nexturl[0][-7:]
87. **except** Exception as e:
88. **print**(e)
90. i += 1

### haufen.py

1. **import** os
2. '''
3. 说明:
4. 将爬取到的新闻按照年份汇总,保存到yearpath目录,每个文件文件名为年份
5. '''
6. **if** \_\_name\_\_ == '\_\_main\_\_':
7. path = "result/"
8. yearpath = "resultByyear/"
9. # 按年进行新闻合并
10. every\_year\_lst = {}
11. # print(os.listdir("./result"))
12. text = ''
13. **for** in\_ **in** os.listdir(path):
14. fp = open(path + in\_, mode="r", encoding='utf-8')
16. # 比较时间
17. **if** int(in\_[:4]) == 2022:
18. f = fp.read().replace("\n", "")
19. f = f.replace(" ", "")
20. text += f
21. every\_year\_lst[2022] = text
23. **if** int(in\_[:4]) == 2021:
24. f = fp.read().replace("\n", "")
25. f = f.replace(" ", "")
26. text += f
27. every\_year\_lst[2021] = text
29. **if** int(in\_[:4]) == 2020:
30. f = fp.read().replace("\n", "")
31. f = f.replace(" ", "")
32. text += f
33. every\_year\_lst[2020] = text
35. **if** int(in\_[:4]) == 2019:
36. f = fp.read().replace("\n", "")
37. f = f.replace(" ", "")
38. text += f
39. every\_year\_lst[2019] = text
41. **if** int(in\_[:4]) == 2018:
42. f = fp.read().replace("\n", "")
43. f = f.replace(" ", "")
44. text += f
45. every\_year\_lst[2018] = text
47. **if** int(in\_[:4]) == 2017:
48. f = fp.read().replace("\n", "")
49. f = f.replace(" ", "")
50. text += f
51. every\_year\_lst[2017] = text
53. **if** int(in\_[:4]) == 2016:
54. f = fp.read().replace("\n", "")
55. f = f.replace(" ", "")
56. text += f
57. every\_year\_lst[2016] = text
59. **if** int(in\_[:4]) == 2015:
60. f = fp.read().replace("\n", "")
61. f = f.replace(" ", "")
62. text += f
63. every\_year\_lst[2015] = text
65. # 分年份保存
66. **for** x, y **in** every\_year\_lst.items():
67. filepath = yearpath + str(x) + ".txt"
68. with open(filepath, encoding="utf-8", mode="w", ) as fp:
69. **print**(x, "年的保存完成！")
70. fp.write(y)

### fenci.py

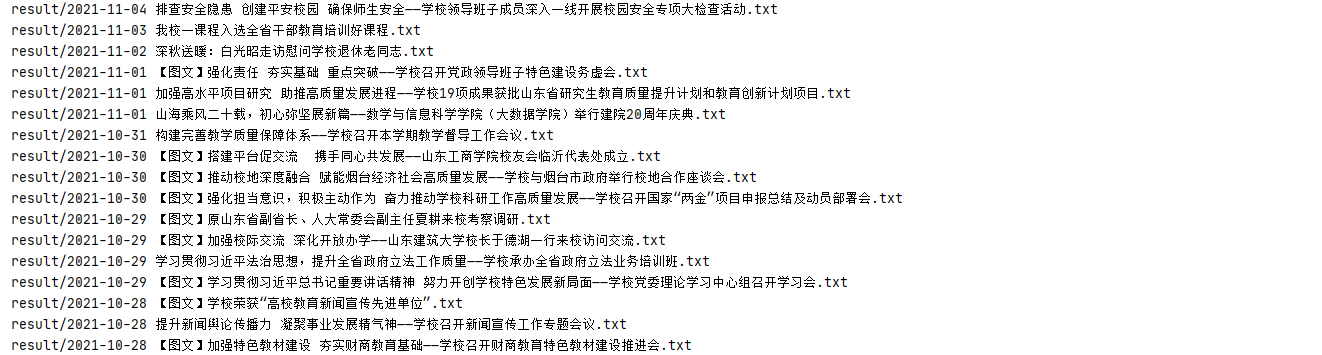
1. **import** os
2. **import** jieba
3. **import** matplotlib.pyplot as plt
5. plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体 SimHei为黑体
6. plt.rcParams['axes.unicode\_minus'] = False  # 用来正常显示负
8. '''''
9. 说明:
10. 使用结巴分词,对每一年的新闻进行分词,并完成词频统计,截取词频前十的词语,绘制柱状图
11. '''
12. **if** \_\_name\_\_ == '\_\_main\_\_':
13. yearpath = "resultByyear/"
14. pic = "pic/"
15. **for** in\_ **in** os.listdir(yearpath):
16. timelst = []
17. wordlst = []
18. # print("年份：",in\_[:4])
19. fp = open(yearpath + in\_, mode="r", encoding='utf-8')
20. text = fp.read()
22. # 结巴分词
23. cut\_text = (jieba.lcut(text))
25. dic = {}
26. **for** word **in** cut\_text:
27. **if** word **not** **in** dic:
28. dic[word] = 1
29. **else**:
30. dic[word] += 1
32. # 统计每个词出现次数，从高到第排序
33. swd = sorted(list(dic.items()), key=**lambda** lst: lst[1], reverse=True)
35. # 排除那些虚词，连词，标点符号等
36. f1 = open('中文虚词列表.txt', encoding="utf-8")
37. stop\_wds = f1.read()
38. f1.close()
40. count = 0
41. **for** kword, times **in** swd:
42. **if** kword **not** **in** stop\_wds **and** count <= 10:  # 当前词未包含在排除的那些词里面，就输出现次数
43. count += 1
44. timelst.append(times)
45. wordlst.append(kword)
46. # print(kword, times)
48. p1 = plt.figure(figsize=(8, 6), dpi=80)  # 确定画布大小
49. plt.title(f'{in\_[:4]}年份词频统计')  # 设置标题
50. plt.bar(range(len(timelst)), timelst, fc='blue')  # 绘制柱状图
52. **for** a, b **in** zip(list(range(len(timelst))), timelst):
53. plt.text(a, b, '%.1f' % b, ha='center', va='bottom', fontsize=10)  # 添加数据标签
54. plt.xticks(range(len(timelst)), wordlst)
56. plt.savefig(f'{pic + in\_[:4]}年份词频统计.png')
57. # plt.show()
59. fp.close()
60. # break

### pyechart.py

1. **from** pyecharts.charts **import** Bar
2. **from** pyecharts **import** options as opts
3. **import** os
4. **import** jieba
5. **from** pyecharts.globals **import** ThemeType
6. **if** \_\_name\_\_ == '\_\_main\_\_':
7. yearpath = "resultByyear/"
8. pic = "pic/"
9. **for** in\_ **in** os.listdir(yearpath):
10. timelst = []
11. wordlst = []
12. # print("年份：",in\_[:4])
13. fp = open(yearpath + in\_, mode="r", encoding='utf-8')
14. text = fp.read()
15. cut\_text = (jieba.lcut(text))
17. dic = {}
18. **for** word **in** cut\_text:
19. **if** word **not** **in** dic:
20. dic[word] = 1
21. **else**:
22. dic[word] += 1
24. swd = sorted(list(dic.items()), key=**lambda** lst: lst[1], reverse=True)  # 统计每个词出现次数，从高到第排序
26. f1 = open('中文虚词列表.txt', encoding="utf-8")  # 排除那些虚词，连词，标点符号等
27. stop\_wds = f1.read()
28. f1.close()
29. count = 0
30. **for** kword, times **in** swd:
31. **if** kword **not** **in** stop\_wds **and** count <= 10:  # 当前词未包含在排除的那些词里面，就输出现次数
32. count += 1
33. timelst.append(times)
34. wordlst.append(kword)
35. # 使用pyechart绘制柱状图
36. bar = (
37. Bar(init\_opts=opts.InitOpts(theme=ThemeType.WALDEN))
38. .add\_xaxis(wordlst)
39. .add\_yaxis(str(in\_[:4]), timelst)
40. .set\_global\_opts(title\_opts=opts.TitleOpts(title=f'{in\_[:4]}年份词频统计'))
41. )
42. bar.render(f'{pic + in\_[:4]}年份词频统计.html')
43. # break

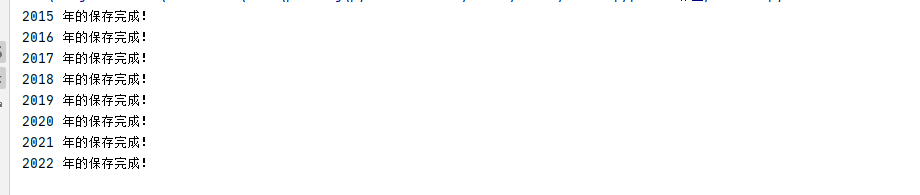
## 运行结果

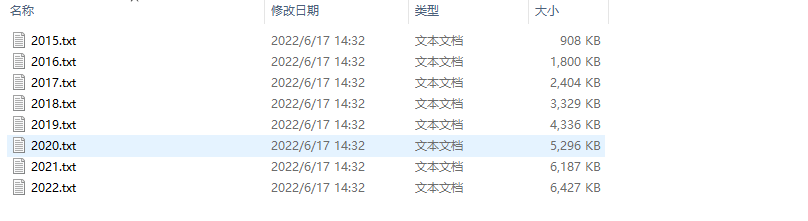
### 爬虫部分运行结果



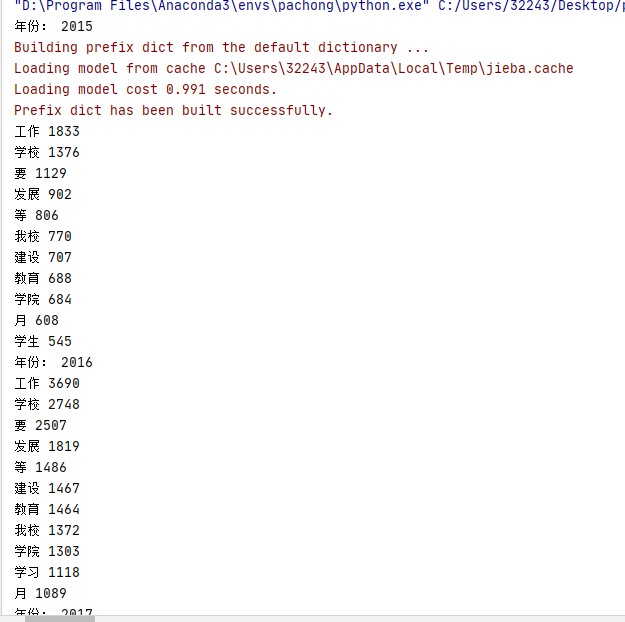


### 将新闻按照年份进行分类保存结果

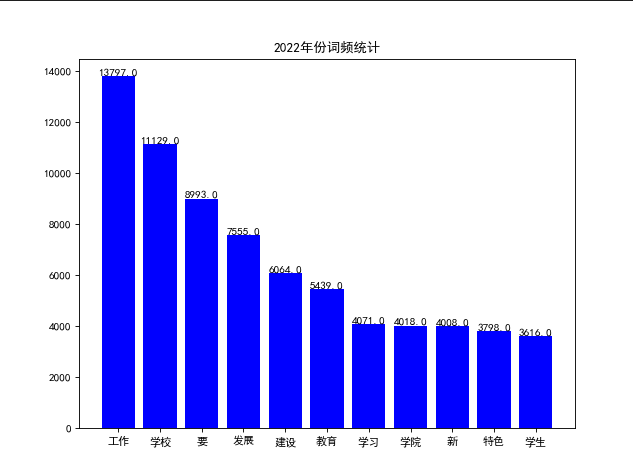




### 分词并统计词频部分



### 利用matplotlib制作柱状图



### 利用pyechart制作柱状图

