**人工智能实验文本数据的分类与分析实验报告**

目录

[1](#_Toc30801) **[总述](#_Toc30801)** [2](#_Toc30801)

[1.1 实验目的 2](#_Toc20509)

[1.2 实验内容 2](#_Toc9851)

[1.3 实验环境 2](#_Toc26701)

[2](#_Toc31112) **[实验原理](#_Toc31112)** [2](#_Toc31112)

[2.1 文档建模 2](#_Toc4470)

[2.2 朴素贝叶斯分类器 3](#_Toc14269)

[2.2.1 基本原理 3](#_Toc18405)

[2.2.2 训练方法 4](#_Toc26346)

[2.2.3 似然函数公式的改进 4](#_Toc21007)

[2.3 测试指标 5](#_Toc21486)

[3](#_Toc28827) **[实验过程](#_Toc28827)** [5](#_Toc28827)

[3.1 构建语料库 5](#_Toc12075)

[3.1.1 爬虫 6](#_Toc867)

[3.1.2 THUCNews语料 7](#_Toc30504)

[3.1.3处理搜狗语料 7](#_Toc8433)

[3.1.4 语料库预处理与合并 7](#_Toc3883)

[3.2 实验程序设计 8](#_Toc4909)

[3.3 文档建模与分类器训练 8](#_Toc9929)

[3.4 文档分类与结果评估 9](#_Toc10750)

[1）10种分类单独的准确率，召回率与F1-measure： 9](#_Toc20342)

[2）10种分类准确率展示 9](#_Toc22074)

[3）10种分类的开方矩阵 10](#_Toc3862)

**[参考资料](#_Toc9462)** [10](#_Toc9462)

**[附录](#_Toc27177)** [10](#_Toc27177)

[实验代码 10](#_Toc28039)

[1.爬虫代码 10](#_Toc29755)

[1）爬取新京报 10](#_Toc20651)

[2）爬取环球网 13](#_Toc16613)

[3）用类似爬虫的方法处理搜狗语料库 16](#_Toc6661)

[2.处理语料库代码 17](#_Toc8686)

[3.训练数据代码 18](#_Toc17319)

[4.测试数据代码 20](#_Toc31725)

# 1 总述

**1.1 实验目的**

（1）掌握数据预处理的方法，对训练集数据进行预处理；

（2）掌握文本建模的方法，对语料库的文档进行建模；

（3）掌握分类算法的原理，基于有监督的机器学习方法，训练文本分类器；

（4）利用学习的文本分类器，对未知文本进行分类判别；

（5）掌握评价分类器性能的评估方法。

**1.2 实验内容**

利用分类算法实现对文本的数据挖掘：

（1）构建语料库，利用爬虫收集Web文档，随机分为训练集和测试集；

（2）语料库数据预处理，使用词袋模型表达文档，建立数据字典；

（3）使用朴素贝叶斯算法在训练集上训练文本分类器；

（4）对测试集的文本进行分类；

（5）对测试集的分类结果利用正确率和召回率进行分析评价。

实验要求文本类别数不小于10类，训练集和测试集文档数总数不小于5000篇，每类平均5000篇。

**1.3 实验环境**

python3.7

分词工具包：jieba

爬虫工具包：scrapy

# 2 实验原理

**2.1 文档建模**

2.1.1 建立数据字典

我们使用词袋模型来表示文本特征，即只考虑一篇文档中单词出现的频率(次数)，用每个单词出现的频数作为文档的特征。

由于中文文档的词汇不是分离的，所以我们需要用分词工具先将文档分成单个的词汇，同时统计每个词出现的频数。在一篇文档中，并不是所有词汇都与文章类型相关，需要设定停用词来去除与分类的无关的词，此外，名词对文档分类较为有效，可以只关注文档中的名词。

对类文档可以分别做上述处理，选出频数高于的词汇。将各类处理得到的词汇汇总，就可以得到数据字典。代表一个词汇，之后对文章的分类，就使用来表示文档，是出现的频数。

2.2 朴素贝叶斯分类器

2.2.1 基本原理

朴素贝叶斯分类器是一个概率分类器。假设现有的类别。给定一篇文档，朴素贝叶斯分类器会把该文档分到文档最有可能属于的类。该问题的数学公式表示如下

 (1)

其中是在所有的类别中，使得条件概率取最大值的类别。使用贝叶斯公式，可将公式(1)转换成如下形式

 (2)

即对类别C中的每个类型，计算 的值，然后选取最大值对应的那个类型 ，该就是最优解，因此，可以忽略掉分母，将公式(3)变成如下形式

 (3)

这个公式由两部分组成，前面那部分称为似然函数，后面那部分称为先验概率。

2.1.1中使用词袋模型来表示文档，文档的每个特征表示为，公式(3)转化成如下形式

 (4)

如果假设在文档中，各个特征之间是相互独立的，那么有

，公式四转化成如下形式

 (5)

由于每个概率值很小，若干个很小的概率值直接相乘，计算过程可能会出现下溢。引入对数函数，在对数空间中进行计算。然后使用词袋模型的每个单词出现频数作为特征，得到如下公式

 (6)

2.2.2 训练方法

训练朴素贝叶斯的过程其实就是计算先验概率和似然函数的过程。

(1)先验概率的计算

表示在所有的文档中，类别为的文档出现的概率。假设训练数据中一共有篇文档，类别的文档共有篇，先验概率的计算公式如下。

 (7)

(2)似然函数的计算

对于文档中的每个单词，找到训练数据集中所有类别为的文档，统计单词在该类别中出现的次数，再统计训练数据集中类别为c的文档的单词总数。似然函数计算公式如下：

 (8)

其中V，就是数据词典。

2.2.3 似然函数公式的改进

若单词在中从未出现过，则为0，这表示文档被分类到类别的概率为0，但这样显然不够合理。为了提高分类器的泛化能力，使用add-one smoothing方法将似然函数公式变成如下形式：

 (9)

其中|V|是数据字典中所有单词的个数。

**2.3 测试指标**

2.3.1 准确率、召回率、F1-meature

假设是分类涉及的若干类，当评估某一类的预测结果时，有以下几种情况。假设有预测样例x，A类是研究对象，B类是除A类之外的某一类。

|  |  |  |
| --- | --- | --- |
| 研究A类预测 | 阳性（Positive） | 阴性（negative） |
| 真（true） | 真阳性（TP）  x实际属于A类，预测为A类 | 真阴性（TN）  x实际不属于A类，预测不属于B类 |
| 假（false） | 假阳性（FP）  x实际属于B类，预测为A类 | 假阴性（FN）  x实际属于A类，预测属于B类 |

用、、、分别表示测试集中的真阳性、真阴性、假阳性、假阴性的样例数量，准确率P为



召回率R为



F1-meature为



2.3.2 矩阵

矩阵是一个10×10的二维矩阵，每一行对应一类数据的分类预测结果，处于对角线上的数据即为该类的召回率R，除了对角线上的数据都是预测错误的数据；每一列的数据为测试集全部文档预测为某一类的概率，用位于对角线上的数据除以所在列所有数据之和即为该类的准确率P。

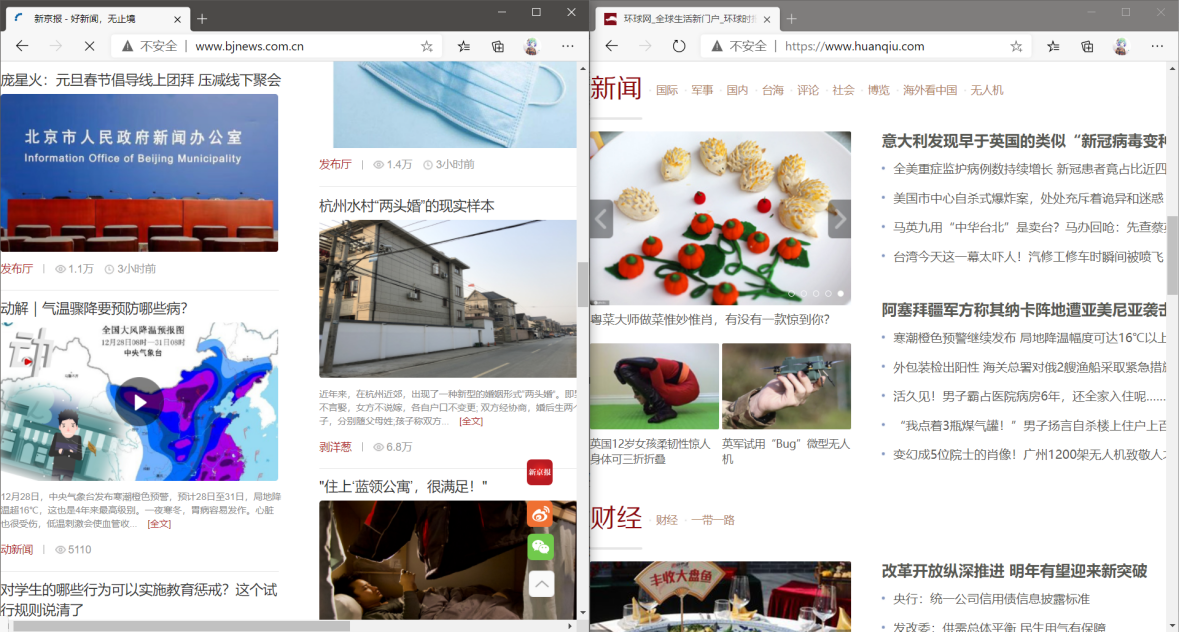
# 3 实验过程

**3.1 构建语料库**

由于实验要求的数据集数量级为每种分类10000篇，其中训练集与测试集各5000篇，数据量较大。故采取爬虫与THUCNews语料2种来源构建适用于本实验的语料库。

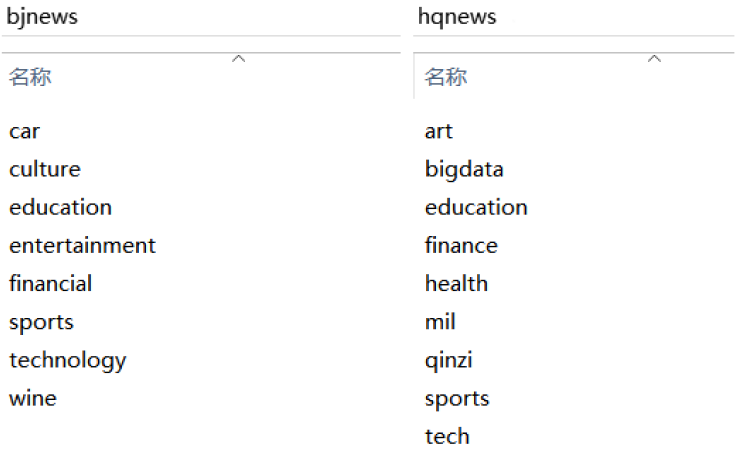
3.1.1 爬虫

采用python下的scrapy框架配合selenium库，爬取环球网与新京报的新闻数据。



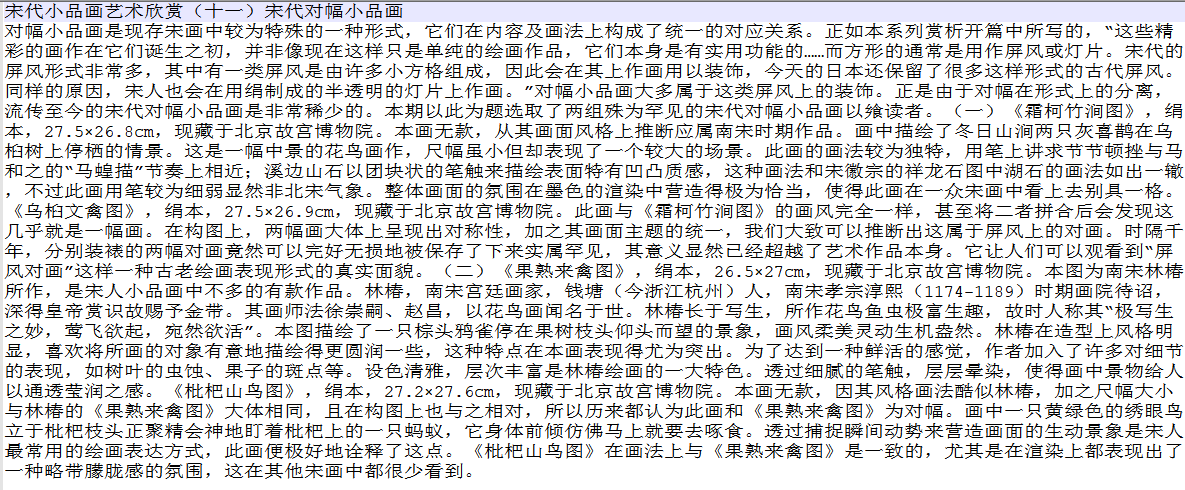
图？ 左-新京报，右-环球网

爬取结果按分类依次存入文件，效果如下：



图？ 爬虫数据目录结构

截取部分文档内容如下：



图？ 爬虫所得文件‘art0001.txt’

爬虫代码见附录[爬虫代码](#爬虫代码)。

3.1.2 THUCNews语料

使用 <http://thuctc.thunlp.org/> 中的已分类新闻数据。

3.1.3处理搜狗语料

用类似爬虫的方法处理搜狗语料库。

处理代码见附录[处理搜狗语料](#搜狗)

3.1.4 语料库预处理与合并

初步查看，爬取的新闻质量参差不齐，部分夹杂广告，部分文本量过少，都不利于正确分类。

1. 去除文件大小小于0.5KB的数据。

2) 合并相同类别的数据集。

3) 剔除数据量三小于10000份的类别。

4) 接着选出合适的10类，即房产，股票，教育，科技，汽车，社会，时尚，体育，游戏，娱乐。

5) 文本命名规范化，将奇数编号文档作为测试集，偶数编号作为训练集。

最终效果如下：



图？ 处理完成的语料库

**3.2 实验程序设计**

实验设计的程序包括三个部分，分别如下。

|  |  |
| --- | --- |
| 模块名 | 功能 |
| train.py | 构建数据词典，统计词频信息，保存为分类器模型表示文件 |
| test.py | 对测试集文档分类，使用分类器模型表示文件进行预测，统计预测正确率和召回率等 |
| utils.py | 共用的工具函数 |

程序中涉及的其他文件说明如下：

|  |  |  |
| --- | --- | --- |
| 文件 | 作用类别 | 说明 |
| stop\_words.txt | 训练阶段输入 | 停止词表 |
| 训练集 | 训练阶段输入 | 所有训练数据放置在一个文件夹中，每类文档分别放在一个子文件中，子文件夹以类别名称命名 |
| word\_bank.txt | 分类器模型表示 | 数据词典，每项包括词汇和频数 |
| [类别].txt | 分类器模型表示 | 每类文档中包含数据词典词汇的情况，每项包括词汇和频数 |
| ariicle\_numbers.txt | 分类器模型表示 | 每类文档中有效文件数量统计，每项包括类别和文档数 |
| word\_numbers.txt | 分类器模型表示 | 每类文档中数据词典内词汇数量统计，每项包括类别和词汇数 |
| 测试集 | 测试阶段输入 | 所有测试数据放置在一个文件夹中，每类文档分别放在一个子文件中，子文件夹以类别名称命名 |

**3.3 文档建模与分类器训练**

以停止词表和训练集为输入，



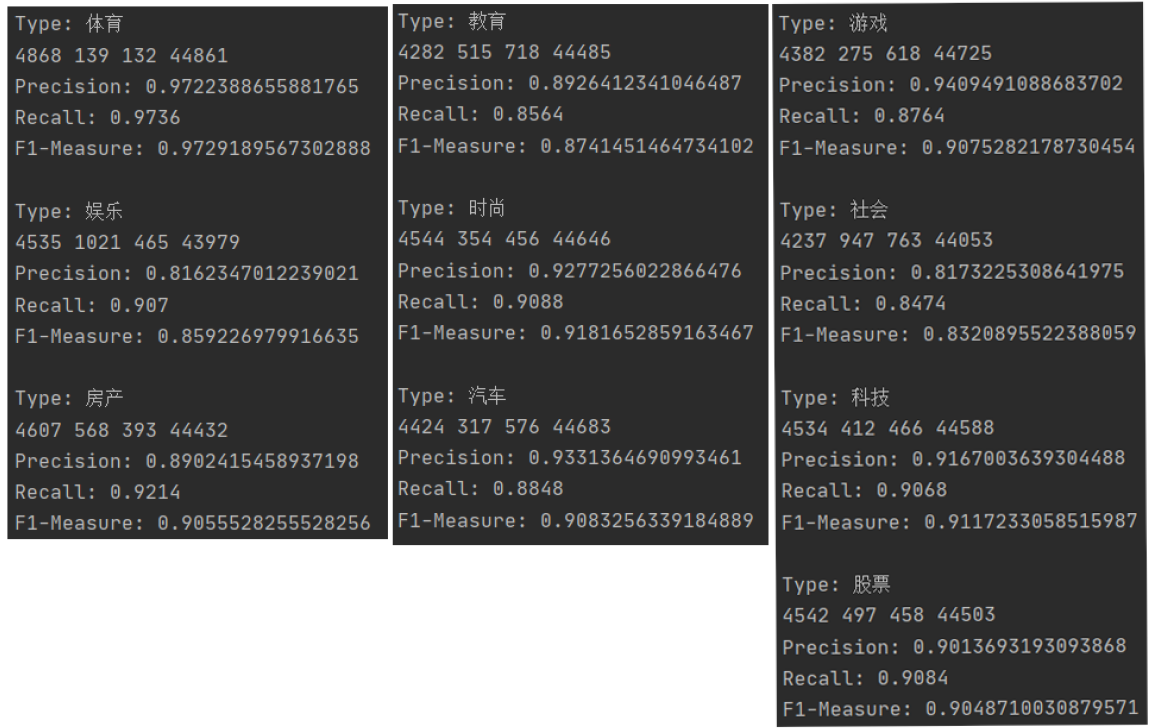
输出的分类器模型表示文件如下：



**3.4 文档分类与结果评估**

以3.3中得到的分类器模型表示文件和测试集为输入，得到的测试结果如下

1）10种分类单独的准确率，召回率与F1-measure：



2）10种分类准确率展示



3）10种分类的开方矩阵



# 参考资料

[1]http://thuctc.thunlp.org/

[2]http://www.sogou.com/labs/resource/cs.php

[3]https://blog.csdn.net/u013063099/article/details/80964865

[4]https://www.cnblogs.com/hapjin/p/8119797.html

# 附录

**实验代码**

**1.****爬虫代码**

**1）爬取新京报**

1. **import** scrapy
2. **import** time
3. **from** xuetang2.items **import** Xuetang2Item
4. **from** scrapy **import** Request

7. **class** CourseSpider(scrapy.Spider):
8. name = 'course'
9. start\_urls = ['https://www.bjnews.com.cn/technology']
11. base\_url\_bjnews = 'https://www.bjnews.com.cn/'
12. all\_categorys\_bjnews = ['financial', 'entertainment', 'culture', 'sports',
13. 'car', 'wine', 'education', 'technology']
14. category = 4
16. all\_links = []
17. link\_index = 0
19. state = 0  # 0表示在爬取link，1表示在爬取内容
20. page = 1
21. count = 0
23. **def** \_\_init\_\_(self):
24. **pass**
26. **def** parse(self, response):
27. item = Xuetang2Item()
28. **print**('\*' \* 50)
30. **if** self.state == 0:  # 如果当前在爬link
31. **for** each **in** response.xpath('//\*[@id="waterfall-container"]/div[\*]'):
32. **if** len(each.xpath('div/div/div/div/a/div/span/text()').extract()) == 0:  # 去除“专题”
33. tmp = each.xpath('div/div/div//@href').extract()
34. **if** len(tmp) > 0:
35. tmp = tmp[0]
36. **print**(tmp)
37. self.all\_links.append(tmp)
38. self.page += 1
39. **if** self.page <= 100:
40. **yield** Request(self.base\_url\_bjnews + self.all\_categorys\_bjnews[self.category] + '/{}.html'.format(self.page))
41. **else**:
42. self.state = 1
43. self.all\_links = set(self.all\_links)  # 去重
44. self.all\_links = list(self.all\_links)
45. **print**(len(self.all\_links))
46. **yield** Request(self.all\_links[0])
48. **else**:  # 如果当前在爬内容
49. title = response.xpath('/html/body/div[3]/div/h1/text()').extract()
50. **print**(title)
51. txt = ''.join(response.xpath('//\*[@id="contentStr"]//text()').extract()).strip()
52. **print**(txt)
53. **if** len(title) > 0 **and** len(txt) > 20:
54. item['title'] = title[0]
55. item['txt'] = txt
56. **yield** item
57. **print**(self.link\_index)
58. self.link\_index += 1
59. **if** self.link\_index < len(self.all\_links):  # 有下一个内容
60. time.sleep(0.1)
61. **yield** Request(self.all\_links[self.link\_index])
62. **else**:  # 无下一个内容，转到爬link
63. time.sleep(2)
64. self.all\_links = []
65. self.state = 0
66. self.link\_index = 0
67. self.count = 0
68. self.page = 1
69. self.category += 1
70. **if** self.category < len(self.all\_categorys\_bjnews):
71. **yield** Request(self.base\_url\_bjnews + self.all\_categorys\_bjnews[self.category])

**储存为文件：**

1. **import** csv
2. **import** os
4. **class** Xuetang2Pipeline(object):
5. **def** open\_spider(self, spider):
6. **pass**
8. **def** process\_item(self, item, spider):
9. spider.count += 1
10. f\_count = str(spider.count).zfill(4)
11. f\_path = "C:\\Users\\87290\\Desktop\\PROGRAM\\Python\\RGZN-2\\bjnews\\" \
12. + spider.all\_categorys\_bjnews[spider.category]
13. f\_name = f\_path + "\\" + spider.all\_categorys\_bjnews[spider.category] + "-" + f\_count + ".txt"
14. **if** **not** os.path.exists(f\_path):
15. os.makedirs(f\_path)
16. with open(f\_name, 'w', encoding='utf8') as f:
17. f.write(item['title'])
18. f.write('\n')
19. f.write(item['txt'])
20. f.close()
22. **return** item
24. **def** close\_spider(self, spider):
25. # spider.driver.quit()
26. # print('Chrome closed')
27. **print**('spider closed')

**2）爬取环球网**

1. **import** scrapy
2. **from** selenium **import** webdriver
3. **from** selenium.webdriver.chrome.options **import** Options
4. **import** time
5. **from** xuetang2.items **import** Xuetang2Item
6. **from** scrapy **import** Request

9. **class** CourseSpider(scrapy.Spider):
10. name = 'course'
11. start\_urls = ['https://tech.huanqiu.com/']
13. all\_urls = ['https://tech.huanqiu.com',
14. 'https://auto.huanqiu.com',
15. 'https://health.huanqiu.com',
16. 'https://mil.huanqiu.com',
17. 'https://qinzi.huanqiu.com',
18. 'https://sports.huanqiu.com',
19. 'https://bigdata.huanqiu.com',
20. 'https://finance.huanqiu.com',
21. 'https://finance.huanqiu.com',
22. 'https://art.huanqiu.com',
23. 'https://lx.huanqiu.com']
24. all\_categorys\_hqnews = ['tech', 'auto', 'health', 'mil', 'qinzi', 'sports', 'bigdata', 'finance', 'art', 'education']
25. category = 0
27. all\_links = []
28. link\_index = 0
30. state = 3  # 0翻页，1爬内容，3，4异常处理(3，4可分配到不同进程处理，加快爬取速度。只需修改category)
31. count = 0
33. **def** \_\_init\_\_(self):
34. chrome\_options = Options()
35. # chrome\_options.add\_argument('--headless')  # 使用无头谷歌浏览器模式
36. chrome\_options.add\_argument('--disable-gpu')
37. chrome\_options.add\_argument('--no-sandbox')
38. # 指定谷歌浏览器路径
39. self.driver = webdriver.Chrome(chrome\_options=chrome\_options,
40. executable\_path='C:/Users/87290/Desktop/PROGRAM/Python/chromedriver.exe')
41. self.driver.implicitly\_wait(10)
42. **print**('webdriver start init success!')
44. **if** self.state == 0:
45. with open(self.all\_categorys\_hqnews[self.category] + "-mulu.txt", 'w', encoding='utf8') as f:
46. **pass**
47. f.close()
49. **if** self.state == 3:
50. fp = open(self.all\_categorys\_hqnews[self.category] + "-mulu.txt", 'r', encoding='utf8')
51. **while** True:
52. line = fp.readline()
53. **if** **not** line:
54. **break**
55. self.all\_links.append(line)
56. fp.close()
57. self.all\_links = set(self.all\_links)  # 去重
58. self.all\_links = list(self.all\_links)
60. **pass**
62. **def** parse(self, response):
63. item = Xuetang2Item()
64. **print**('\*' \* 50)
66. **if** self.state == 0:  # 获取网址list
67. **for** each **in** response.xpath('//\*[@id="recommend"]/li[\*]'):
68. **if** len(each.xpath('a/@href').extract()) > 0:
69. tmp = self.all\_urls[self.category] + each.xpath('a/@href').extract()[0]
70. self.all\_links.append(tmp)
71. with open(self.all\_categorys\_hqnews[self.category] + "-mulu.txt", 'a', encoding='utf8') as f:
72. f.write(tmp + '\n')
73. f.close()
74. **print**(tmp)
76. self.state = 1
77. **print**(len(self.all\_links))
78. **yield** Request(self.all\_links[0])
79. **pass**
81. **elif** self.state == 1:  # 切换类别
82. time.sleep(2)
83. self.all\_links = []
84. self.state = 0
85. self.link\_index = 0
86. self.count = 0
87. self.category += 1
88. **if** self.category < len(self.all\_categorys\_hqnews):
89. **yield** Request(self.all\_urls[self.category])
91. **elif** self.state == 3:  # 转入4前的准备状态
92. self.state += 1
93. **yield** Request(self.all\_links[0])
95. **else**:  # 爬内容
96. title = response.xpath('//div[@class="t-container-title"]/h3/text()').extract()
97. **print**(title)
98. txt = ''.join(response.xpath('//div[@class="l-con clear"]/article/section//text()').extract()).strip()
99. **print**(txt)
100. **if** len(title) > 0 **and** len(txt) > 20:
101. item['title'] = title[0]
102. item['txt'] = txt
103. **yield** item
104. **print**(self.link\_index)
105. self.link\_index += 1
106. **if** self.link\_index < len(self.all\_links):  # 有下一个内容
107. time.sleep(0.1)
108. **yield** Request(self.all\_links[self.link\_index])
109. **pass**

**储存为文件：**

1. **import** scrapy
2. **from** selenium **import** webdriver
3. **from** selenium.webdriver.chrome.options **import** Options
4. **import** time
5. **import** random
7. **class** Xuetang2Middleware(object):
8. **def** \_\_init\_\_(self):
9. **pass**
11. **def** process\_request(self, request, spider):
12. wd = spider.driver
13. **print**("In middleware")
14. **if** spider.state == 0:
15. wd.get(request.url)
16. time.sleep(1.5)
17. **print**("scrolling", end='')
18. **for** i **in** range(500):
19. **print**(".", end='')
20. wd.execute\_script('window.scrollBy(0,100000)')
21. time.sleep(1.5)
22. **print**("Return html")
23. html = spider.driver.page\_source
24. # print(html)
25. **return** scrapy.http.HtmlResponse(url=spider.driver.current\_url, body=html.encode('utf-8'), encoding='utf-8',
26. request=request)
27. **pass**
28. **def** \_\_del\_\_(self):
29. **pass**

**3）用类似爬虫的方法处理搜狗语料库**

1. **import** scrapy
2. **from** xuetang2.items **import** Xuetang2Item
3. **import** time

6. **class** CourseSpider(scrapy.Spider):
7. name = 'course'
8. start\_urls = ['file:///C:/Users/87290/Desktop/PROGRAM/Python/RGZN-2/news\_sohusite\_xml\_UTF-8.html']
10. category\_dict = {}
11. tmp\_list = []
13. count = 0
15. **def** \_\_init\_\_(self):
16. **pass**
18. **def** parse(self, response):
19. **print**('\*' \* 50)
20. count = 0
21. lenth = len(response.xpath('/html/body/doc[\*]'))
22. item = Xuetang2Item()
23. **print**('\*' \* 50)
25. **for** each **in** response.xpath('/html/body/doc[\*]'):
26. count += 1
27. **print**(count, '/', lenth)
28. self.tmp\_list.append(each.xpath('url/text()').extract()[0].split('/')[2])
29. type = each.xpath('url/text()').extract()[0].split('/')[2].split('.')[-3]
30. self.category\_dict[type] = self.category\_dict.get(type, 0)
31. title = each.xpath('contenttitle/text()').extract()
32. txt = each.xpath('content/text()').extract()
33. **if** len(title) > 0 **and** len(txt) > 0:
34. **print**(type)
35. **print**(title[0])
36. **print**(txt[0])
37. **print**('-' \* 50)
38. item['type'] = type
39. item['title'] = title[0]
40. item['txt'] = txt[0]
41. **yield** item
42. # time.sleep(0.1)
44. with open('category\_dict.txt', 'a', encoding='utf8') as f:
45. **for** key, value **in** self.category\_dict.items():
46. f.write(key)
47. f.write(' : ' + str(value) + '\n')
48. f.close()
50. self.tmp\_list = set(self.tmp\_list)
51. with open('category\_set.txt', 'a', encoding='utf8') as f:
52. **for** every **in** self.tmp\_list:
53. f.write(every)
54. f.write('\n')
55. f.close()

**2.处理语料库代码**

1. **import** os, shutil
2. **import** time

5. base\_path = 'D:\A temp\RGZN-2\THUCNews-backup'
6. move\_path = 'D:\A temp\RGZN-2\THUC-split\\'
8. jump = True
9. file\_cnt = 0
10. flag = 0  # 0重命名，1移动文件
12. **for** \_, \_\_, \_\_\_ **in** os.walk(base\_path):
13. **if** jump:
14. jump = False
15. **continue**
16. **else**:
17. file\_cnt = 0
18. **print**("\*" \* 50)
19. **print**('\_', \_)
20. **print**('\_\_\_', len(\_\_\_))
21. **for** each\_file **in** \_\_\_:
22. file\_cnt += 1
23. **print**('moving', file\_cnt, '/', len(\_\_\_))
24. **if** flag == 0:
25. os.rename(\_ + '\\' + each\_file, \_ + '\\' + \_.split('\\')[4] + str(file\_cnt).zfill(6) + '.txt')
26. **else**:
27. **if** int(each\_file[len(each\_file) - 10: len(each\_file) - 4]) % 2 == 0:
28. **if** **not** os.path.exists(move\_path + \_.split('\\')[-1] + '\\train\\'):
29. os.makedirs(move\_path + \_.split('\\')[-1] + '\\train\\')
30. shutil.move(\_ + '\\' + each\_file, move\_path + \_.split('\\')[-1] + '\\train\\' + each\_file)
31. **else**:
32. **if** **not** os.path.exists(move\_path + \_.split('\\')[-1] + '\\test'):
33. os.makedirs(move\_path + \_.split('\\')[-1] + '\\test')
34. shutil.move(\_ + '\\' + each\_file, move\_path + \_.split('\\')[-1] + '\\test\\' + each\_file)
35. **pass**
36. **pass**
37. **pass**

**3.训练数据代码**

1. **import** os
2. **from** utils **import** utils

5. threshold = 100
7. # D:\\Projects\\Pycharm\\JB\\复旦测试集
8. train\_set\_path = "D:\\A temp\\RGZN-1\\ML文本分类\\训练集"
9. times\_record\_name = "times.txt"
11. A = []
13. #  得到训练集文件的路径组织
14. **for** \_, \_\_, \_\_\_ **in** os.walk(train\_set\_path):
15. #print(\_)
16. #print(\_\_)
17. #print(\_\_\_)
18. A.append([\_, \_\_, \_\_\_])
20. #  初始化记录训练结果的数据结构
21. word\_bank = dict()  # 词库
22. word\_total = 0  # 词库单词数目
23. article\_total = 0  # 训练使用的文章数目
25. #  训练（即开始分词并统计）
26. **for** i **in** range(0, len(A[0][1])):    # 子文件夹组成的列表
27. sub\_word\_bank = dict()
28. sub\_article\_total = 0
30. article\_type = A[0][1][i]
31. **print**("Type: "+A[0][1][i])
32. **for** txt **in** A[i+1][2]:
33. txt\_path = train\_set\_path + "\\" + article\_type + "\\" + txt
34. **print**(txt\_path)
36. content = utils.open\_file(txt\_path)
37. **if** content == None:
38. **continue**
39. **else**:
40. # 统计有效文章数
41. article\_total += 1
42. sub\_article\_total += 1
44. sub\_word\_bank = utils.update\_dict(content, sub\_word\_bank)
46. sub\_word\_bank = sorted(sub\_word\_bank.items(), key=**lambda** d: d[1], reverse=True)
48. **if** len(sub\_word\_bank) > 501:
49. sub\_word\_bank = sub\_word\_bank[1:501]  # 降维到500
51. **print**(sub\_word\_bank)
53. dict\_name = article\_type + ".txt"
54. sub\_word\_total, sub\_word\_bank= utils.save\_sub\_word\_bank(sub\_word\_bank, threshold, dict\_name)
56. word\_total += sub\_word\_total
58. utils.append\_dict(article\_type, sub\_word\_total, "word\_numbers.txt")
59. utils.append\_dict(article\_type, sub\_article\_total, "article\_numbers.txt")
61. utils.update\_word\_bank(word\_bank, sub\_word\_bank)
62. **print**("\*"\*50)
64. utils.save\_dict(word\_bank, "word\_bank.txt")  # 写入词库,每一项包括词汇和频数
65. utils.append\_dict("total", word\_total, "word\_numbers.txt")
66. utils.append\_dict("total", article\_total, "article\_numbers.txt")

**4.测试数据代码**

1. **import** jieba.analyse
2. **import** os
3. **from** utils **import** utils
4. **import** math

7. train\_set\_path = "D:\\A temp\\RGZN-1\\ML文本分类\\测试集"
9. #  得到训练集文件的路径组织
10. A = []
11. **for** \_, \_\_, \_\_\_ **in** os.walk(train\_set\_path):
12. #print(\_)
13. #print(\_\_)
14. #print(\_\_\_)
15. A.append([\_, \_\_, \_\_\_])
17. #  获取文章类型列表
18. article\_types = A[0][1]
19. **print**("Involved types:")
20. **print**(article\_types)
22. #  读入训练好的模型（即统计数据）
23. word\_numbers = utils.load\_dict("word\_numbers.txt")  # 读入训练集词库中词汇数目的统计信息
24. article\_numbers = utils.load\_dict("article\_numbers.txt")  # 读入训练集文章数目统计信息
25. word\_bank = utils.load\_dict("word\_bank.txt")  # 读入词库,每一项包括词汇和频数
26. sub\_word\_bank = [utils.load\_dict(article\_types[i]+".txt") **for** i **in** range(len(article\_types))]  # 读入子词库,每一项包括词汇和在特定类别出现的频数
27. jieba.load\_userdict("word\_bank.txt")  # 将词库载入jieba
29. #  计算先验
30. priors = dict()
31. **for** type **in** article\_numbers.keys():
32. **if** type != "total":
33. priors[type] = int(article\_numbers[type])/int(article\_numbers["total"])
35. #  初始化统计指标
36. features = [[0, 0, 0, 0] **for** i **in** range(len(article\_types))]  # 每行的四个数据分别代表TP、FP、FN、TN

39. #  定义分类器
40. **def** judge(dict, article\_types, word\_numbers, priors, word\_bank, sub\_word\_bank):
41. scores = [0 **for** i **in** range(len(article\_types))]
43. V = len(word\_bank)
44. # print("V",V)
46. #  先验得分
47. **for** i **in** range(len(article\_types)):
48. scores[i] += math.log(priors[article\_types[i]])
50. **for** word **in** dict.keys():
51. # print(word)
52. #  先检查这个单词是否在词库里，不在则不纳入考虑
53. **if** word **not** **in** word\_bank.keys():
54. **continue**
56. times = dict[word]
57. **for** i **in** range(len(article\_types)):
58. #  获取子词库的总词数
59. total\_times = word\_numbers[article\_types[i]]
60. #  获取这个单词在特定子词库出现的频数
61. **if** word **in** sub\_word\_bank[i]:
62. sub\_times = sub\_word\_bank[i][word]
63. **else**:
64. sub\_times = 0
65. #  add-one smoothing方式计算得分
66. scores[i] += times \* math.log((sub\_times + 1)/(total\_times+V))
68. max\_score = scores[0]
69. max\_i = 0
70. **for** i **in** range(1, len(article\_types)):
71. **if** scores[i] > max\_score:
72. max\_score = scores[i]
73. max\_i = i
75. # print(article\_types[max\_i])
76. **return** max\_i
78. #  定义更新统计数据的函数
79. #  每行的四个数据分别代表TP、FP、FN、TN
80. **def** update\_features(prediction, real, features):
81. **if** prediction == real:
82. **for** i **in** range(len(article\_types)):
83. **if** i == prediction:
84. features[i][0] += 1
85. **else**:
86. features[i][3] += 1
87. **else**:
88. **for** i **in** range(len(article\_types)):
89. **if** i == prediction:
90. features[i][1] += 1
91. **elif** i == real:
92. features[i][2] += 1
93. **else**:
94. features[i][3] += 1
96. result = [[0 **for** j **in** range(len(article\_types))] **for** i **in** range(len(article\_types))]
98. #  测试
99. **for** i **in** range(0,len(article\_types)):
101. article\_type = article\_types[i]
102. **print**(article\_type)
104. **for** txt **in** A[i+1][2]:
106. D = dict()  # 用来记录单篇文章的提取词信息
107. txt\_path = train\_set\_path + "\\" + article\_type + "\\" + txt
108. **print**(txt\_path)
110. content = utils.open\_file(txt\_path)
111. **if** content == None:
112. **continue**
113. D = utils.update\_dict(content, D)
115. prediction = judge(D, article\_types, word\_numbers, priors, word\_bank, sub\_word\_bank)
117. result[i][prediction] += 1
119. **print**("prediction: "+article\_types[prediction]+", answer: "+article\_types[i])
120. update\_features(prediction, i, features)

123. #  打印测试结果
124. total\_TP = 0
125. total\_FP = 0
126. total\_FN = 0
127. total\_TN = 0
129. **for** i **in** range(len(article\_types)):
130. **print**("Type: "+article\_types[i])
132. TP = features[i][0]
133. FP = features[i][1]
134. FN = features[i][2]
135. TN = features[i][3]
137. **print**(TP, FP, FN, TN)
139. P = TP/(TP + FP) **if** (TP + FP) > 0 **else** 0
140. R = TP/(TP + FN) **if** (TP + FN) > 0 **else** 0
141. FM = 2\*P\*R/(P+R) **if** (P+R) > 0 **else** 0
143. **print**("Precision: " + str(P))
144. **print**("Recall: " + str(R))
145. **print**("F1-Measure: " + str(FM))
146. **print**()
148. total\_TP += TP
149. total\_FP += FP
150. total\_FN += FN
151. total\_TN += TN
153. total\_P = total\_TP/(total\_TP + total\_FP) **if** (total\_TP + total\_FP) > 0 **else** 0
154. total\_R = total\_TP/(total\_TP + total\_FN) **if** (total\_TP + total\_FN) > 0 **else** 0
155. total\_FM = 2\*total\_P\*total\_R/(total\_P + total\_R) **if** (total\_P + total\_R) > 0 **else** 0
157. **print**("Totol:")
158. **print**("Precision: " + str(total\_P))
159. **print**("Recall: " + str(total\_R))
160. **print**("F1-Measure: " + str(total\_FM))

163. # 卡方矩阵输出
164. **print**('卡方矩阵')
165. **for** i **in** range(-1,10):
166. **for** j **in** range(-1,10):
167. **if** i==-1 **and** j==-1:
168. **print**('%10s' % '',end = '')
169. **elif** i==-1and j!=-1:
170. **print**('%9s' % article\_types[j],end = '')
171. **elif** i!=-1 **and** j==-1:
172. **print**('%10s' % article\_types[i],end = '')
173. **else**:
174. **print**('{:9.2f}%'.format(float(result[i][j]/50)),end = '')
175. **print**()
176. **print**()
178. sum = [0 **for** i **in** range(10)]
179. **for** j **in** range(10):
180. **for** i **in** range(10):
181. sum[j]+=result[i][j]
183. # 准确率输出
184. **print**('准确率')
185. **for** i **in** range(10):
186. **print**('%10s' % article\_types[i],end = ':')
187. **print**('{:9.2f}%'.format(float((result[i][i]\*100)/sum[i])),end = '')
188. **print**()