

人工智能实验文本数据的分类与分析实验报告

目录

1 总述	2
1.1 实验目的	2
1.2 实验内容	2
1.3 实验环境	2
2 实验原理	2
2.1 文档建模	2
2.2 朴素贝叶斯分类器	3
2.2.1 基本原理	3
2.2.2 训练方法	4
2.2.3 似然函数公式的改进	4
2.3 测试指标	5
3 实验过程	6
3.1 构建语料库	6
3.1.1 爬虫	6
3.1.2 THUCNews 语料	7
3.1.3 处理搜狗语料	7
3.1.4 语料库预处理与合并	7
3.2 实验程序设计	8
3.3 文档建模与分类器训练	8
3.4 文档分类与结果评估	9
1) 10 种分类单独的准确率, 召回率与 F1-measure:	9
2) 10 种分类准确率展示	9
3) 10 种分类的开方矩阵	10
参考资料	10
附录	10
实验代码	10
1.爬虫代码	10
1) 爬取新京报	10
2) 爬取环球网	13
3) 用类似爬虫的方法处理搜狗语料库	16
2.处理语料库代码	17
3.训练数据代码	18
4.测试数据代码	20

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 c_i 类文档可以分别做上述处理，选出频数高于 t_i 的词汇。将各类处理得到的词汇汇总，就可以得到数据字典 $\mathbf{V} = \{w_1, w_2, \dots, w_n\}$ 。 w_i 代表一个词汇，之后对文章的分类，就使用 $d = \{f_1, f_2, \dots, f_n\}$ 来表示文档， f_i 是 w_i 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 $C = \{c_1, c_2, \dots, c_m\}$ 。给定一篇文档 d ，朴素贝叶斯分类器会把该文档分到文档 d 最有可能属于的类 \hat{c} 。该问题的数学公式表示如下

$$\hat{c} = \arg \max_{c \in C} P(c | d) \quad (1)$$

其中 \hat{c} 是在所有的类别 $C = \{c_1, c_2, \dots, c_m\}$ 中，使得条件概率 $P(c | d)$ 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

$$\hat{c} = \arg \max_{c \in C} P(c | d) = \arg \max_{c \in C} \frac{P(d | c)P(c)}{P(d)} \quad (2)$$

即对类别 C 中的每个类型，计算 $\frac{P(d | c)P(c)}{P(d)}$ 的值，然后选取最大值对应的那个类型 c_i ，该 c_i 就是最优解 \hat{c} ，因此，可以忽略掉分母 $P(d)$ ，将公式(3)变成如下形式

$$\hat{c} = \arg \max_{c \in C} P(d | c)P(c) \quad (3)$$

这个公式由两部分组成，前面那部分 $P(c | d)$ 称为似然函数，后面那部分 $P(c)$ 称为先验概率。

2.1.1中使用词袋模型来表示文档 d ，文档 d 的每个特征表示为 $d = \{f_1, f_2, \dots, f_n\}$ ，

公式(3)转化成如下形式

$$\hat{c} = \arg \max_{c \in C} \underbrace{P(f_1, f_2, \dots, f_n | c)}_{\text{似然函数}} \underbrace{P(c)}_{\text{先验概率}} \quad (4)$$

如果假设在文档 d 中，各个特征之间是相互独立的，那么有

$P(f_1, f_2, \dots, f_n | c) = P(f_1 | c)P(f_2 | c) \dots P(f_n | c)$, 公式四转化成如下形式

$$\hat{c} = \arg \max_{c \in C} P(c) \prod_{f \in d} P(f | c) \quad (5)$$

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 w_i 出现频数作为特征, 得到如下公式

$$\hat{c} = \arg \max_{c \in C} [\log P(c) + \sum_{w_i \in V} \log P(w_i | c)] \quad (6)$$

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 $P(c)$ 的计算

$P(c)$ 表示在所有的文档中, 类别为 c 的文档出现的概率。假设训练数据中一共有 N_{doc} 篇文档, 类别 c 的文档共有 N_c 篇, 先验概率的计算公式如下。

$$\hat{P}(c) = \frac{N_c}{N_{doc}} \quad (7)$$

(2)似然函数 $P(w_i | c)$ 的计算

对于文档 d 中的每个单词 w_i , 找到训练数据集中所有类别为 c 的文档, 统计单词 w_i 在该类别中出现的次数 $count(w_i, c)$, 再统计训练数据集中类别为 c 的文档的单词总数 $\sum_{w \in V} count(w, c)$ 。似然函数计算公式如下:

$$\hat{P}(w_i | c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)} \quad (8)$$

其中 V , 就是数据词典。

2.2.3 似然函数公式的改进

若单词 w_i 在 c_i 中从未出现过, 则 $count(w_i, c)$ 为 0, 这表示文档 d 被分类到类别 c 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:

$$\hat{P}(w_i | c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\sum_{w \in V} count(w, c) + |V|} \quad (9)$$

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-measure

假设 $C = \{c_1, \dots, c_A, \dots, c_B, \dots, c_n\}$ 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 n_{TP} 、 n_{TN} 、 n_{FP} 、 n_{FN} 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

$$P = \frac{n_{TP}}{n_{TP} + n_{TN}}$$

召回率R为

$$R = \frac{n_{TP}}{n_{TP} + n_{FN}}$$

F1-measure为

$$F = \frac{2PR}{P + R}$$

2.3.2 χ^2 矩阵

χ^2 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。

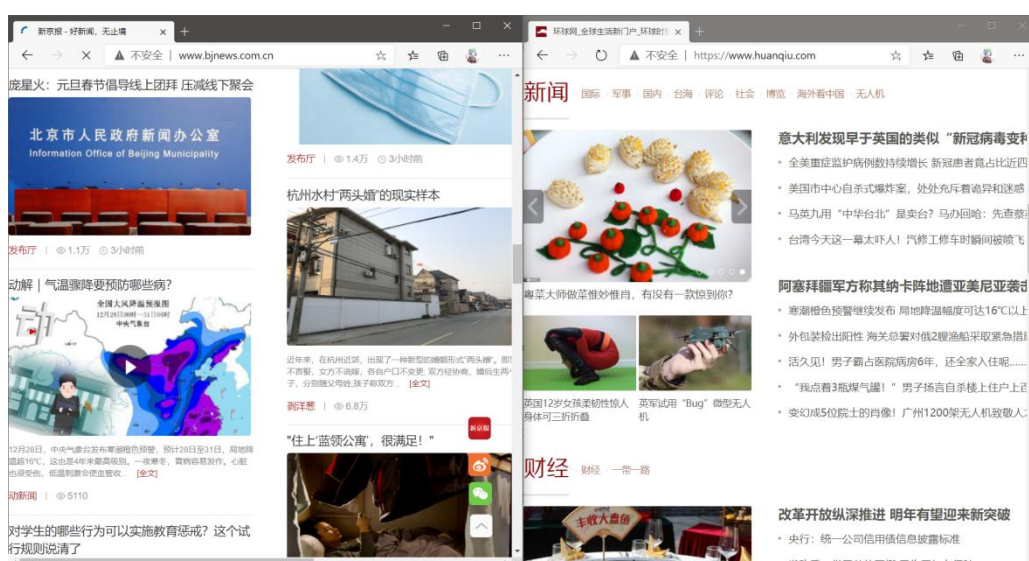
3 实验过程

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：

bjnews	hnews
名称	名称
car	art
culture	bigdata
education	education
entertainment	finance
financial	health
sports	mil
technology	qinzi
wine	sports
	tech

图？ 爬虫数据目录结构

截取部分文档内容如下：

宋代小品画艺术欣赏（十一）宋代对幅小品画

对幅小品画是现存宋画中较为特殊的一种形式，它们在内容及画法上构成了统一的对应关系。正如本系列赏析开篇中所写的，“这些精彩的画作在它们诞生之初，并非像现在这样只是单纯的绘画作品，它们本身是有实用功能的……而方形的通常是用作屏风或灯片。宋代的屏风形式非常多，其中有一类屏风是由许多小方格组成，因此会在其上作画用以装饰，今天的日本还保留了很多这样形式的古代屏风。同样的原因，宋人也会在用绢制成的半透明的灯片上作画。”对幅小品画大多属于这类屏风上的装饰。正是由于对幅在形式上的分离，流传至今的宋代对幅小品画是非常稀少的。本期以此为题选取了两组殊为罕见的宋代对幅小品画以飨读者。（一）《霜柯竹涧图》，绢本，27.5×26.8cm，现藏于北京故宫博物院。本画无款，从其画面风格上推断应属南宋时期作品。画中描绘了冬日山涧两只灰喜鹊在乌桕树上停栖的情景。这是一幅中景的花鸟画作，尺幅虽小但却表现了一个较大的场景。此画的画法较为独特，用笔上讲求节节顿挫与马和之的“马蝗描”节奏上相近；溪边山石以团块状的笔触来描绘表面特有凹凸质感，这种画法和宋徽宗的祥龙石图中湖石的画法如出一辙，不过此画用笔较为细弱显然非北宋气象。整体画面的氛围在墨色的渲染中营造得极为恰当，使得此画在众多宋画中看上去别具一格。《乌柏文禽图》，绢本，27.5×26.9cm，现藏于北京故宫博物院。此画与《霜柯竹涧图》的画风完全一样，甚至将二者拼合后会发现这几乎就是一幅画。在构图上，两幅画大体上呈现出对称性，加之其画面主题的统一，我们大致可以推断出这属于屏风上的对画。时隔千年，分别装裱的两幅对画竟然可以完好无损地被保存了下来实属罕见，其意义显然已经超越了艺术作品本身。它让人们可以观看到“屏风对画”这样一种古老绘画表现形式的真实面貌。（二）《果熟来禽图》，绢本，26.5×27cm，现藏于北京故宫博物院。本图为南宋林椿所作，是宋人小品画中不多的有款作品。林椿，南宋宫廷画家，钱塘（今浙江杭州）人，南宋孝宗淳熙（1174-1189）时期画院待诏，深得皇帝赏识故赐予金带。其画师法徐崇嗣、赵昌，以花鸟画闻名于世。林椿长于写生，所作花鸟鱼虫极富生趣，故时人称其“极写生之妙，鸢飞欲起，宛然欲活”。本图描绘了一只棕头鸦雀停在果树枝头仰头而望的景象，画风柔美灵动生机盎然。林椿在造型上风格明显，喜欢将所画的对象有意地描绘得更圆润一些，这种特点在本画表现得尤为突出。为了达到一种鲜活的感觉，作者加入了许多对细节的表现，如树叶的虫蚀、果子的斑点等。设色清雅，层次丰富是林椿绘画的一大特色。透过细腻的笔触，层层晕染，使得画中景物给人以通透莹润之感。《枇杷山鸟图》，绢本，27.2×27.6cm，现藏于北京故宫博物院。本画无款，因其风格画法酷似林椿，加之尺幅大小与林椿的《果熟来禽图》大体相同，且在构图上也与之相对，所以历来都认为此画和《果熟来禽图》为对幅。画中一只黄绿色的绣眼鸟立于枇杷枝头正聚精会神地盯着枇杷上的一只蚂蚁，它身体前倾仿佛马上就要去啄食。透过捕捉瞬间动势来营造画面的生动景象是宋人最常用的绘画表达方式，此画便极好地诠释了这点。《枇杷山鸟图》在画法上与《果熟来禽图》是一致的，尤其是在渲染上都表现出了—种略带朦胧感的氛围，这在其他宋画中都很少看到。

图？ 爬虫所得文件 ‘art0001.txt’

爬虫代码见附录[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](#)

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：

测试集				训练集			
文件	主页	共享	查看	文件	主页	共享	查看
名称	修改日期	类型		名称	修改日期	类型	大小
房产	2020/12/8 22:41	文件夹		房产	2020/12/9 9:05	文件夹	
股票	2020/12/8 22:48	文件夹		股票	2020/12/8 22:48	文件夹	
教育	2020/12/8 22:41	文件夹		教育	2020/12/8 22:41	文件夹	
科技	2020/12/8 23:38	文件夹		科技	2020/12/8 23:38	文件夹	
汽车	2020/12/25 15:07	文件夹		汽车	2020/12/25 15:10	文件夹	
社会	2020/12/8 22:43	文件夹		社会	2020/12/8 22:43	文件夹	
时尚	2020/12/8 22:41	文件夹		时尚	2020/12/8 22:41	文件夹	
体育	2020/12/8 23:36	文件夹		体育	2020/12/8 23:36	文件夹	
游戏	2020/12/8 22:43	文件夹		游戏	2020/12/8 22:43	文件夹	
娱乐	2020/12/8 22:40	文件夹		娱乐	2020/12/8 22:40	文件夹	
10个项目				10个项目			

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
article_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，














 stop_words.txt

2020/12/3 21:03

TXT 文件

1 KB

输出的分类器模型表示文件如下：

 article_numbers.txt	2020/12/25 15:38	TXT 文件	1 KB
 word_bank.txt	2020/12/25 15:38	TXT 文件	35 KB
 word_numbers.txt	2020/12/25 15:38	TXT 文件	1 KB
 股票.txt	2020/12/25 15:38	TXT 文件	7 KB
 科技.txt	2020/12/25 15:37	TXT 文件	7 KB
 社会.txt	2020/12/25 15:35	TXT 文件	7 KB
 游戏.txt	2020/12/25 15:32	TXT 文件	7 KB
 汽车.txt	2020/12/25 15:30	TXT 文件	7 KB
 时尚.txt	2020/12/25 15:27	TXT 文件	6 KB
 教育.txt	2020/12/25 15:26	TXT 文件	7 KB
 房产.txt	2020/12/25 15:23	TXT 文件	7 KB
 娱乐.txt	2020/12/25 15:20	TXT 文件	7 KB
 体育.txt	2020/12/25 15:17	TXT 文件	7 KB

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:

Type: 体育 4868 139 132 44861 Precision: 0.9722388655881765 Recall: 0.9736 F1-Measure: 0.9729189567302888	Type: 教育 4282 515 718 44485 Precision: 0.8926412341046487 Recall: 0.8564 F1-Measure: 0.8741451464734102	Type: 游戏 4382 275 618 44725 Precision: 0.9409491088683702 Recall: 0.8764 F1-Measure: 0.9075282178730454
Type: 娱乐 4535 1021 465 43979 Precision: 0.8162347012239021 Recall: 0.907 F1-Measure: 0.859226979916635	Type: 时尚 4544 354 456 44646 Precision: 0.9277256022866476 Recall: 0.9088 F1-Measure: 0.9181652859163467	Type: 社会 4237 947 763 44053 Precision: 0.8173225308641975 Recall: 0.8474 F1-Measure: 0.8320895522388059
Type: 房产 4607 568 393 44432 Precision: 0.8902415458937198 Recall: 0.9214 F1-Measure: 0.9055528255528256	Type: 汽车 4424 317 576 44683 Precision: 0.9331364690993461 Recall: 0.8848 F1-Measure: 0.9083256339184889	Type: 科技 4534 412 466 44588 Precision: 0.9167003639304488 Recall: 0.9068 F1-Measure: 0.9117233058515987
		Type: 股票 4542 497 458 44503 Precision: 0.9013693193093868 Recall: 0.9084 F1-Measure: 0.9048710030879571

2) 10种分类准确率展示

体育:	97.22%
娱乐:	81.62%
房产:	89.02%
教育:	89.26%
时尚:	92.77%
汽车:	93.31%
游戏:	94.09%
社会:	81.73%
科技:	91.67%
股票:	90.14%

3) 10种分类的开方矩阵

	体育	娱乐	房产	教育	时尚	汽车	游戏	社会	科技	股票
体育	97.36%	0.96%	0.14%	0.30%	0.08%	0.04%	0.16%	0.88%	0.04%	0.04%
娱乐	0.20%	90.70%	0.26%	1.20%	2.20%	0.02%	1.42%	3.32%	0.52%	0.16%
房产	0.04%	0.78%	92.14%	0.82%	0.06%	0.44%	0.18%	1.92%	0.24%	3.38%
教育	0.30%	2.64%	1.06%	85.64%	1.32%	0.14%	0.80%	6.40%	0.94%	0.76%
时尚	0.26%	4.34%	0.52%	0.46%	90.88%	0.70%	0.84%	1.04%	0.86%	0.10%
汽车	0.06%	0.52%	1.08%	1.18%	0.16%	88.48%	0.40%	2.38%	1.12%	4.62%
游戏	0.84%	6.64%	0.34%	0.28%	0.24%	0.54%	87.64%	0.44%	2.74%	0.30%
社会	0.40%	3.38%	2.50%	4.42%	1.54%	0.40%	0.58%	84.74%	1.70%	0.34%
科技	0.36%	1.10%	0.74%	1.48%	1.46%	0.82%	1.00%	2.12%	90.68%	0.24%
股票	0.32%	0.06%	4.72%	0.16%	0.02%	3.24%	0.12%	0.44%	0.08%	90.84%

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`


```

47.
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
3.
4.     class Xuetang2Pipeline(object):
5.         def open_spider(self, spider):
6.             pass
7.
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()
21.
22.     return item
23.
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
7.
8.
9.     class CourseSpider(scrapy.Spider):
10.         name = 'course'
11.         start_urls = ['https://tech.huanqiu.com/']
12.
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
26.
27.         all_links = []
28.         link_index = 0
29.
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
32.
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!')
43.
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()
48.
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)
59.
60.         pass
61.
62.     def parse(self, response):
63.         item = Xuetang2Item()
64.         print('*' * 50)
65.
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)
75.
76.         self.state = 1
77.         print(len(self.all_links))
78.         yield Request(self.all_links[0])

```



```

79.         pass
80.
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])
90.
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])
94.
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
6.
7.  class Xuetang2Middleware(object):
8.      def __init__(self):
9.          pass
10.
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.scrollTo(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
4.
5.
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']
9.
10.        category_dict = {}
11.        tmp_list = []
12.
13.        count = 0
14.
15.        def __init__(self):
16.            pass
17.
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)
24.
25.            for each in response.xpath('/html/body/doc[*]'):

```

```

26.         count += 1
27.         print(count, '/', lenh)
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)
43.
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()
49.
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
3.
4.
5.     base_path = 'D:\A temp\RGZN-2\THUCNews-backup'
6.     move_path = 'D:\A temp\RGZN-2\THUC-split\'
7.
8.     jump = True
9.     file_cnt = 0
10.    flag = 0 # 0 重命名, 1 移动文件
11.
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
3.
4.
5.     threshold = 100
6.
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"
10.
11.     A = []
12.
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([_, __, ___])
19.
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
24.
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
29.
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)
35.
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
43.
44.                 sub_word_bank = utils.update_dict(content, sub_word_bank)
45.
46.                 sub_word_bank = sorted(sub_word_bank.items(), key=lambda d: d[1], reverse=True)
47.
48.                 if len(sub_word_bank) > 501:
49.                     sub_word_bank = sub_word_bank[1:501] # 降维到 500
50.
51.                 print(sub_word_bank)
52.
53.                 dict_name = article_type + ".txt"
54.                 sub_word_total, sub_word_bank = utils.save_sub_word_bank(sub_word_bank, threshold, dict_name)
55.
56.                 word_total += sub_word_total
57.
58.                 utils.append_dict(article_type, sub_word_total, "word_numbers.txt")
59.                 utils.append_dict(article_type, sub_article_total, "article_numbers.txt")
60.
61.                 utils.update_word_bank(word_bank, sub_word_bank)

```

```

62.     print("*"*50)
63.
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
5.
6.
7.     train_set_path = "D:\\A temp\\RGZN-1\\ML 文本分类\\测试集"
8.
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([_, __, ___])
16.
17.    # 获取文章类型列表
18.    article_types = A[0][1]
19.    print("Involved types:")
20.    print(article_types)
21.
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
28.
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"])
34.
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

```



```

37.
38.
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))]
42.
43.     V = len(word_bank)
44.     # print("V",V)
45.
46.     # 先验得分
47.     for i in range(len(article_types)):
48.         scores[i] += math.log(priors[article_types[i]])
49.
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
55.
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))
67.
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
74.
75.         # print(article_types[max_i])
76.         return max_i
77.
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
95.
96.     result = [[0 for j in range(len(article_types))] for i in range(len(article_types))]
97.
98.     # 测试
99.     for i in range(0, len(article_types)):
100.
101.         article_type = article_types[i]
102.         print(article_type)
103.
104.         for txt in A[i+1][2]:
105.
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)
109.
110.             content = utils.open_file(txt_path)
111.             if content == None:
112.                 continue
113.             D = utils.update_dict(content, D)
114.
115.             prediction = judge(D, article_types, word_numbers, priors, word_bank, sub_word_bank)
116.
117.             result[i][prediction] += 1
118.
119.             print("prediction: "+article_types[prediction]+", answer: "+article_types[i])
120.             update_features(prediction, i, features)
121.
122.
123.     # 打印测试结果
124.     total_TP = 0

```

```

125.     total_FP = 0
126.     total_FN = 0
127.     total_TN = 0
128.
129.     for i in range(len(article_types)):
130.         print("Type: "+article_types[i])
131.
132.         TP = features[i][0]
133.         FP = features[i][1]
134.         FN = features[i][2]
135.         TN = features[i][3]
136.
137.         print(TP, FP, FN, TN)
138.
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
142.
143.         print("Precision: " + str(P))
144.         print("Recall: " + str(R))
145.         print("F1-Measure: " + str(FM))
146.         print()
147.
148.         total_TP += TP
149.         total_FP += FP
150.         total_FN += FN
151.         total_TN += TN
152.
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
156.
157.     print("Total:")
158.     print("Precision: " + str(total_P))
159.     print("Recall: " + str(total_R))
160.     print("F1-Measure: " + str(total_FM))
161.
162.
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()
177.
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]
182.
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()
```