基于朴素贝叶斯方法的新闻分类

1300012758　张闻涛

【数据集构造】

1　学习 python 及 scrapy 框架

1.1　因为听闻 python 写网络应用方面较为便捷，而且自己也希望多掌握几门语言。故借此机会学习 python，跟随网上的教程及官方文档，耗时 3 天左右。

2　创建 scrapy 工程 wikicrawl

2.1　设定 start_url 为五个标签的主页面：Crime and law, Culture and entertainment, Economy and business, Disasters and accidents, Education。我从 wiki news 的 archived 不能修改的新闻中爬取各类新闻。

2.2　负责处理类别页面的 parse 函数中，使用 xpath 语句提取出我们需要的新闻链接，并请求访问（再使用 parseSub 分析新闻页面）。再使用 xpath 语句提取跳至下一页的链接，并请求访问（再使用 parse 递归分析），直到没有下一页为止。故可以把一个类别中的新闻按字母表顺序全部遍历一遍。

2.3　负责处理新闻页面的 parseSub 函数中，使用 xpath 语句提出新闻的 TITLE，BODY 和 CATEGORY，再次检查这篇文章是不是我们要的分类中的一个，是否会分到两个类中，判断之后将其写入到文件。

2.4　至此，我们将新闻都爬取到了本地。

3　构造测试集和训练集

3.1　将测试集和训练集按 1：4 划分。总共有 8000 多篇新闻，

'Crime and law': 3006, 'Culture and entertainment': 1392, 'Economy and business': 1569, 'Education': 172, 'Disasters and accidents': 1988

取近似值，我取了 1600 篇新闻作为测试集，放在 Bayes/test 文件夹下。其中各类篇数：

'Crime and law': 570, 'Culture and entertainment': 276,

'Economy and business': 320, 'Education': 34, 'Disasters and accidents': 400。各个类型都大致按 1：4 划分。

3.2　剔除测试集之后，剩余新闻作为训练集，6400 篇，放在 Bayes/train 文件夹下。

【设计贝叶斯分类器】

1　大体思路

1.1　使用词袋模型，忽略每篇新闻中各个词之间的顺序关系。

1.2　统计以下几个量：

1.2.1　每个类别中 各个单词的 词频/这个类别总词数，作为 P(Xi|C)。

1.2.2　每个类别的词数/所有类别的总词数，作为 P(C)。

1.3　测试一个新闻属于哪一类时，对于每一类，只需把这个新闻中的各个词的 P(Xi|C)连乘起来，再乘上 P(C)，计算得出一个概率，然后再在各个类中挑选此概率最大的，将这篇新闻分到这个类。

2　实现细节、trick

2.1　过滤停用词

2.1.1　过滤那些，常见但没有意义的词：as, the, for 等等，我在网上下载了一张停用词表，并用程序读入，进行过滤，统计和分类时无视他们。

2.2　平滑，处理没有出现的单词

2.2.1　在对测试数据进行分类时，我想，若这个新闻中的单词没有在训练集中出

现过怎么办？答案是：可以直接无视，因为每个分类中都不管这个词，不影响概率计算。

2.2.2 对于测试新闻，若这个分类中没有这个词，而其他分类中有这个词，那这个分类的概率怎么计算？如果直接乘 P(Xi|C)，则变成了 0，完全否决了该分类的可能性，不合理。我们不妨认为训练集中每个词都在这个分类里出现过一次，对于那些本来就出现的，就再给他+1。如此计算 P(Xi|C)，就不会乘 0 了。

2.3 用加法运算代替乘法

2.3.1 最初计算概率的时候我用了乘法计算，结果发现分类正确率爆低，每次都不到 50%。通过调试我发现。。计算后验概率用了乘法之后，几乎全都乘成了 0，因为我们认为各个单词独立，而且用词频除以总词数作为条件概率，测试文章的单词又很多，很容易浮点数精度不够最后乘成 0。

2.3.2 故我们采用常用的加法替代乘法的方法，将 P(Xi|C)的连乘用连加来代替：即将 P(Xi|C)取对数，然后加起来即可，对于 P(C)也这么做。当然，由于概率都是小于 1 的东西，我们最后加法的结果都是负数。

2.3.3 采用了加法代替乘法后，分类器总体的正确率马上上升了许多，到达了 7/8。

【实验结果及分析】

```
Total Precision:  correct / total = 1407 / 1600
[Crime and law]
Precision : 528 / 603 = 87.562%
Recall : 528 / 570 = 92.632%
F = 90.026%
[Culture and entertainment]
Precision : 230 / 264 = 87.121%
Recall : 230 / 276 = 83.333%
F = 85.185%
[Economy and business]
Precision : 273 / 319 = 85.580%
Recall : 273 / 320 = 85.312%
F = 85.446%
[Education]
Precision : 10 / 10 = 100.000%
Recall : 10 / 34 = 29.412%
F = 45.455%
[Disasters and accidents]
Precision : 366 / 404 = 90.594%
Recall : 366 / 400 = 91.500%
F = 91.045%
```

整体结果还算令人满意。Education 类的召回率不如人意(训练集中这一类新闻确实很少)，商业经济和文化娱乐两个类别往往有重合，故不易区分，准确率下降也可以理解（在人工看 wiki news 的时候我也发现 一篇新闻同时又这两个分类的情况很多）

换一组分类集 'Crime and law', 'Culture and entertainment', 'Disasters and accidents', 'Science and technology', 'Health'。训练集 6727，测试集 1600

```
Total Precision:  correct / total = 1370 / 1600
[Crime and law]
Precision : 496 / 598 = 82.943%
Recall : 496 / 540 = 91.852%
F = 87.170%
[Culture and entertainment]
Precision : 251 / 283 = 88.693%
Recall : 251 / 300 = 83.667%
F = 86.106%
[Science and technology]
Precision : 241 / 269 = 89.591%
Recall : 241 / 320 = 75.312%
F = 81.834%
[Health]
Precision : 90 / 112 = 80.357%
Recall : 90 / 120 = 75.000%
F = 77.586%
[Disasters and accidents]
Precision : 292 / 338 = 86.391%
Recall : 292 / 320 = 91.250%
F = 88.754%
```

【心得体会】

学习 scrapy 框架爬取新闻的过程最为艰辛，因为那时候我对“框架”的概念还摸不着头脑，不懂有了框架之后，我还要写些什么来完成一个完整的工程。在反复研读 scrapy 的 tutorial 和文档，一步一步照着 tutorial 的过程建立起一个自己的爬虫的时候，我感觉到了编程语言的魅力。

在几天的语言学习“撞完东墙撞西墙”之后，在设计朴素贝叶斯分类器时我感觉明显轻松了很多。这个阶段碰到的问题主要是对贝叶斯分类的理解。在贝叶斯模型中，我开始纠结于 P(Xi|C)与 P(C)的选取。我认为有两种可行的取法，第一种就是我现在采用的，P(Xi|C)=TF/总词数，P(C)=该分类词数/总词数；第二种是 P(Xi|C)=DF/总文档数量，P(C)=该分类文档数/总文档数。经过试验，最终采取了第一种方式。

这次作业对我来说有很大的意义，不仅加深了我对贝叶斯概率模型的理解，也让我的编程技能得到了锻炼。