Detecting Sentiment Polarity说明文档

1. 任务定义

设计一个情感分类器，对未知情感极性的句子进行正确分类。通过所给的情感词典数据（其中已做好情感极性、强弱标注），还有已经分好极性的句子，进行训练得到一个基于朴素Bayes的情感分类器。使用该情感分类器对未知极性的句子进行分类，与正确的分类结果进行比较，用Precision、Recall方法进行评估。

1. 输入输出

输入为subjclueslen1-HLTEMNLP05.tff（情感词典），rt-polarity.pos（正极性句子），rt-polarity.neg（负极性句子）。取句子的80%作为训练集，其余的作为测试集。

输出Precision、Recall。（随机选取的训练集与测试集，多次测量体现平均性能）

1. 源码运行环境

使用python 3.6进行编写，在Windows 10上运行通过。源代码为DetectingSentimentPolarity.py。

1. 方法描述

* 数据处理

观察所给的正极性句子与负极性句子，其中：每一条语料为几个单词，用空格分隔。我们需要将每个句子的单词提取出来，使用正则表达式\W+捕捉所有非字母数字的模式串，将其作为分割符，对每个句子进行分割，将一个句子分割为几个单词，得到一个元素。每个句子作为一个元素，最终得到清洗后的数据集。

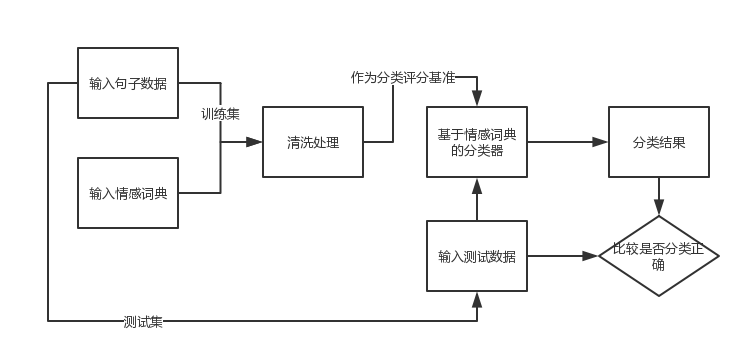
观察所给的情感词典，其中：每一个词有type（该词的情感强弱），len（该次的长度），word（该词），pos（该词的词性），stemm（不知道什么，没用上），polarity(该词的极性)。形如

我们需要将每个词的各项属性提取出来，使用正则表达式=\w+捕捉所有以=开头后面为字母或数字的字符串，然后用空串替换=，得到词的各项属性。对于词的强弱，我们可以将其替换成score属性，即强positive的score为2，弱positive的score为1，强negative的score为-2，若negative的score为-1，中性词的score为0。

* 训练集与测试集

在处理数据时可以得到语料的数量，我们选择其中的20%作为测试集，80%作为训练集。生成一个随机数序列，从语料集中取该序列对应的语料，保证测试集的随机性。

* 基于情感词典的分类器

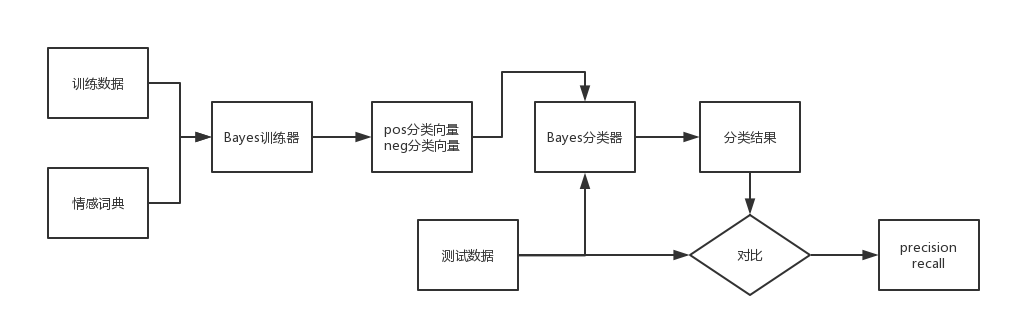
该部分算法流程如下：

该方法无需训练，通过上述数据处理步骤中，将词的强弱转化为得分score，通过对输入句子的每个单词的得分，进行求和，得到该句子的得分。通过对句子的得分进行简单判断，若得分大于0，则认为是positive句子，否则是negative句子。

|  |
| --- |
| **for** word **in** i:  **if** word **in** polar:  score += polar[word]['score']  **if** word **in** notlist:  cnt\_not += 1 **if** cnt\_not % 2 == 1:  score = -score #以上得到单个句子的情感得分 **if** score < 0:#若判定为负面情感则错误  false\_pos += 1 **else**:  ture\_pos += 1 |

在此判断中加入了否定词的判断，即有一个否定词时，句子的得分取相反数；两个否定词时，句子得分不变。即score=(-1)­nscore。但是经过测试后发现，不进行否定词的判断使得分类结果更准确。对这个方面存在疑问，可能是否定词词典太小，或者是训练数据不够。

* 基于朴素Bayes的分类器

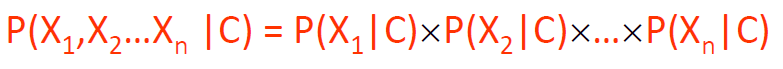
该部分算法流程如下：

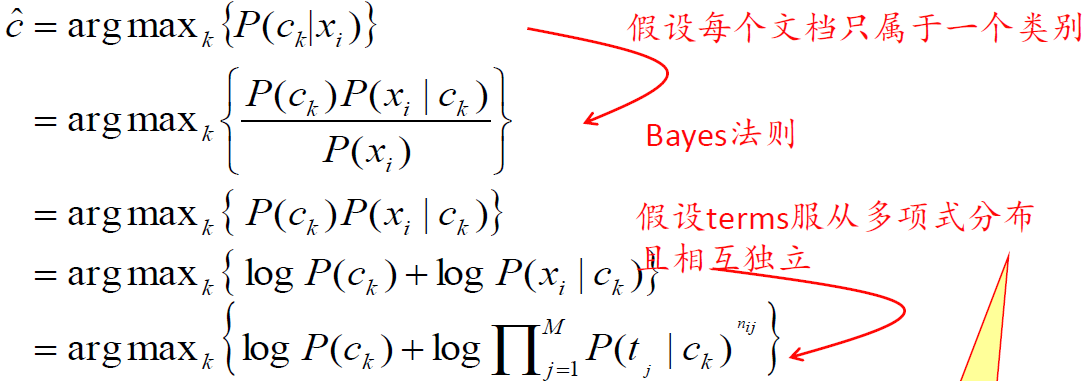
* Bayes训练器分类向量的理论计算

把输入文本看作一个m\*n矩阵，m为文本条数，n为情感词典宽度。则所需分类向量应该为n维向量。这样输入文本矩阵乘分类向量就能得到一个m维的概率向量，确定某条文本是positive还是negative。

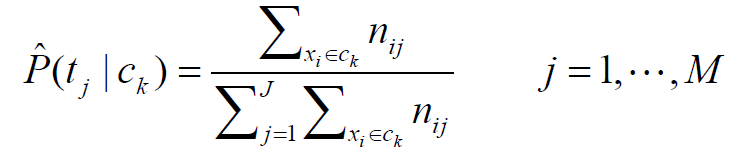
确定了向量的维度后，接下来来研究如何计算出该分类向量。

朴素Bayes有一个条件独立性假设：已知类别C，则变量X1，X2…Xn之间是相互独立的，可以把参数规模从O(X^n \*C)降到O(X \* C)，即有：



给定一条输入文本Xi，Xi由几个单词t1,t2…tn组成，且Xi只为positive或negative则其最可能的分类C计算如下：

由于分类只有positive和negative，故P(c1)=P(c2)=0.5。

接下来是对已知类别ck时，term tj的出现概率进行计算，即：

此方法使用了极大似然估计，故还需进行平滑处理，此处选择进行delta=1的平滑方法，至此，所有计算方法已完成。

* Bayes训练器分类向量的编程实现

先创建两个分类向量，维度为情感词典polar的宽度：

|  |
| --- |
| bayes\_pos = [0 **for** i **in** range(len(polar))]#pos分类向量 bayes\_neg = [0 **for** i **in** range(len(polar))]#neg分类向量 |

统计训练语料中各词出现的频次，在对应向量位置上计数：

|  |
| --- |
| **for** i **in** train\_pos:  **for** j **in** i:  **if** j **in** polar\_list:  bayes\_pos[polar\_list.index(j)] += 1  **for** i **in** train\_neg:  **for** j **in** i:  **if** j **in** polar\_list:  bayes\_neg[polar\_list.index(j)] += 1 |

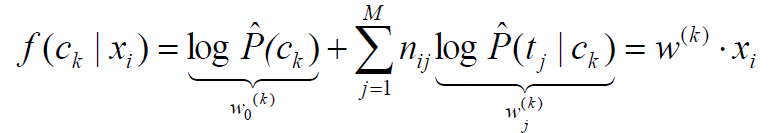
计算

即把向量每个位置上的频次，进行极大似然估计，再平滑：

|  |
| --- |
| **for** i **in** range(len(bayes\_pos)):  bayes\_pos[i] = log((bayes\_pos[i] + delta) / (len(train\_pos) + len(polar) \* delta)) **for** i **in** range(len(bayes\_neg)):  bayes\_neg[i] = log((bayes\_neg[i] + delta) / (len(train\_neg) + len(polar) \* delta)) |

至此得到positive分类向量和negative分类向量。

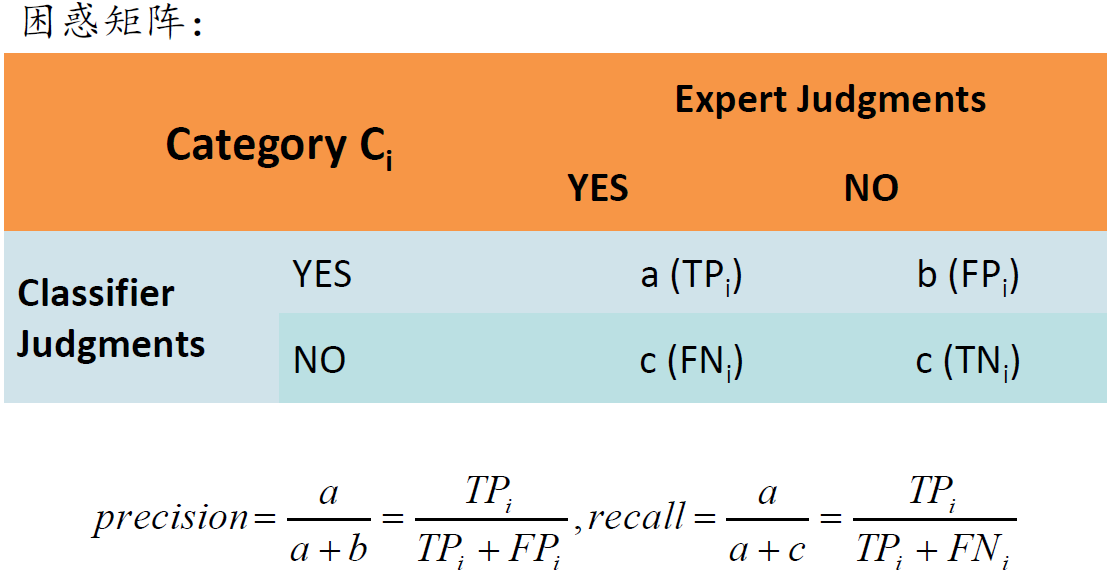
* Bayes分类器

接下来只需要将测试矩阵与分类向量相乘，得到分为positive的概率和分为negative的概率，即可进行分类。

实现如下：

|  |
| --- |
| **for** i **in** test\_pos:  p\_to\_pos = sum(map(**lambda** a\_b:a\_b[0]\*a\_b[1],zip(i,bayes\_pos))) + log(p\_pos)#测试语料与分类向量点积  p\_to\_neg = sum(map(**lambda** a\_b:a\_b[0]\*a\_b[1],zip(i,bayes\_neg))) + log(p\_neg)#测试语料与分类向量点积  **if** p\_to\_pos > p\_to\_neg:  ture\_pos += 1  **else**:  false\_pos += 1 |

* 结果评价



编程实现如下：

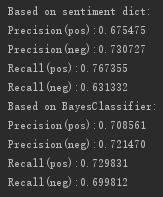
|  |
| --- |
| prec\_pos = ture\_pos / (ture\_pos + false\_neg) prec\_neg = ture\_neg / (ture\_neg + false\_pos) recall\_pos = ture\_pos / len(test\_pos) recall\_neg = ture\_neg / len(test\_neg) |

1. 结果分析（性能评价）

由于训练集与测试集都是在所给的语料库中随机提取，故每次测量的Precision、Recall都会不一样，p代表Precision, r代表Recall，在此测试了5次作平均评价。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dict1 | dict2 | dict3 | dict4 | dict5 | bayes1 | bayes2 | bayes3 | bayes4 | bayes5 |
| p(pos) | 0.671 | 0.673 | 0.686 | 0.668 | 0.676 | 0.704 | 0.707 | 0.714 | 0.695 | 0.706 |
| p(neg) | 0.724 | 0.730 | 0.729 | 0.723 | 0.757 | 0.714 | 0.732 | 0.710 | 0.724 | 0.719 |
| r(pos) | 0.760 | 0.768 | 0.757 | 0.761 | 0.802 | 0.720 | 0.747 | 0.708 | 0.742 | 0.727 |
| r(neg) | 0.628 | 0.627 | 0.653 | 0.621 | 0.617 | 0.698 | 0.691 | 0.717 | 0.674 | 0.696 |

可以看到，基于情感词典的分类器不太稳定，准确性波动比较大。而基于朴素Bayes的分类器稳定在70%，可靠性比较强。但是两者的综合性能差不多。

 程序输出：