邮件分类系统实验报告

161250193张傲

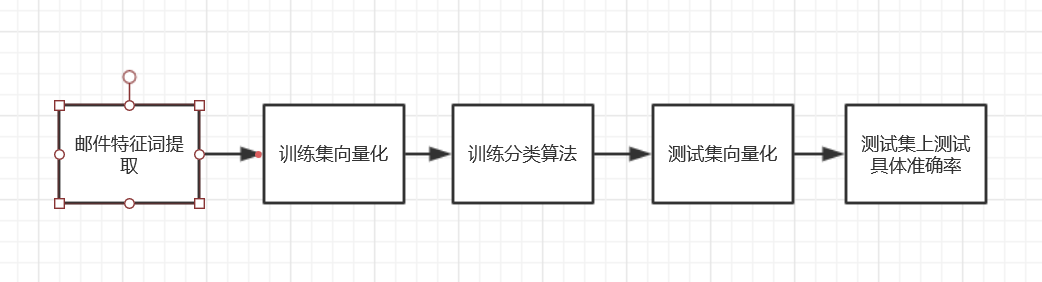
141220106王兴照

161080023 李智昊

1. 综述

项目的任务是使用总共19997份20个分类的邮件数据集作为训练集，进行特征提取

和分类算法的训练，然后在测试集上测试邮件分类的具体准确率。具体流程如下图:

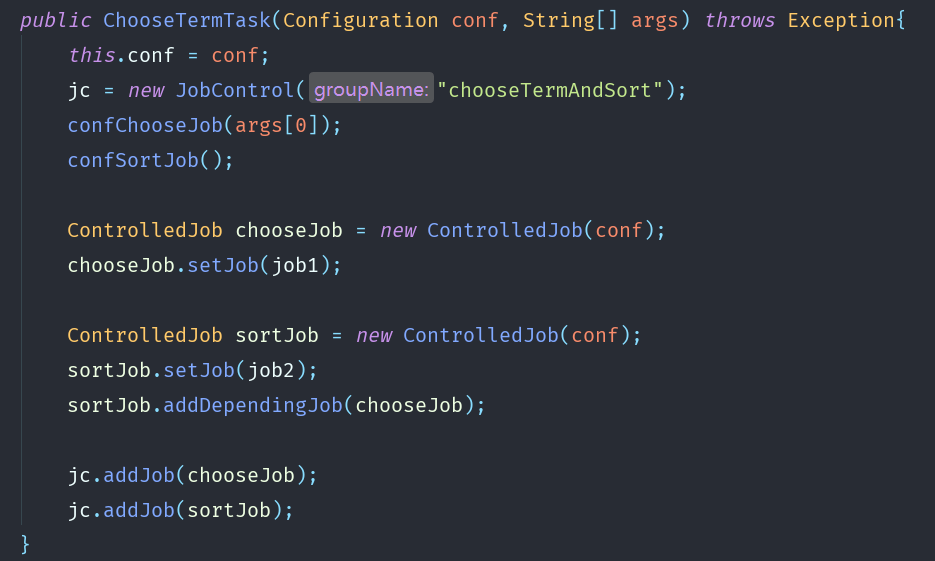


首先使用卡方独立性检验的方法在每个类别内选取最具有代表性的m个关键词。第二步将训练集采取one-hot编码向量化为对应词向量。第三步利用文档的特征向量，进行朴素贝叶斯算法的训练，即计算每一个种类下，每个特征不同取值的不同可能性。第四步，测试集向量化后，应用朴素贝叶斯算法进行准确率测试。

1. 代码组织方式
2. 不同任务的组织方式：

对于不同的独立任务，我们将其封装成不同的task，在task的构造方法中进行MapReduce Job的设置，使用execute方法执行。如此，可以进行方便的小组分工，和Main方法内的不同任务的不同序列组合，以及task内部的组件替换。

例如在选取特征这一task:

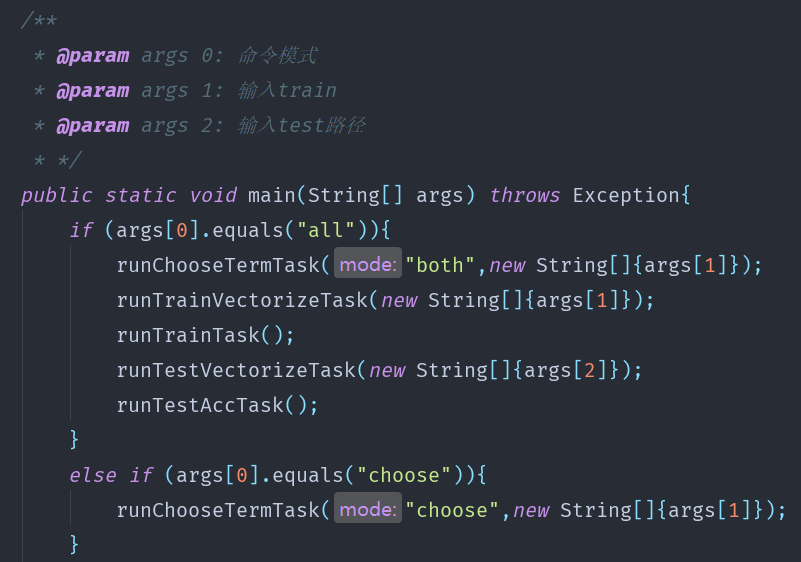


构造方法设定task内部执行的job，并用JobControll进行组织。



具体的job设置。

然后可以方便的根据命令行参数进行不同执行序列的指定:



1. 全局设定:

全部放在MyConf类内部，以方便调参。中间文件的输出路径放在FilePaths类中，以方便更改路径名称。

1. 分词器等辅助工具：

以单件模式工具类的形式存在。

1. 特征选择
2. 特征选择: 基于卡方独立性检验

特征选择分为两个Job，第一步要计算每一个类别和不同词语的卡方独立性检验对应的统计量，即两者之间相关性。第二步，对于n个不同分类，我们在分类内部按照相关性从高到低的顺序，进行特征词排序，选取m个相关性最高的特征词作为该分类的特征词。那么最后将会选出总共mn个特征词最为文档分类特征向量计算的依据。

1. 卡方独立性检验思想介绍：

假设term t与分类c独立，即两者不相关。然后计算其偏差程度，如果偏差越大，那么落入该统计量对应卡方分布拒绝域可能性越大，即两者相关性越差。所以我们可以通过计算该偏差统计量来代表t与c的相关程度。

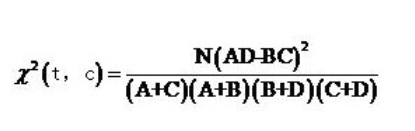
具体原理可以参照<https://blog.csdn.net/xiaocong1990/article/details/70804589>

算法并行化设计部分(原创):



(图片来自上述博客)

统计量公式:



注: 下面term为单词 cls为所属分类

<1>Mapper任务: 对于每一篇doc，将***互不同的***term以(<term,cls>, 1)形式键值对发射。

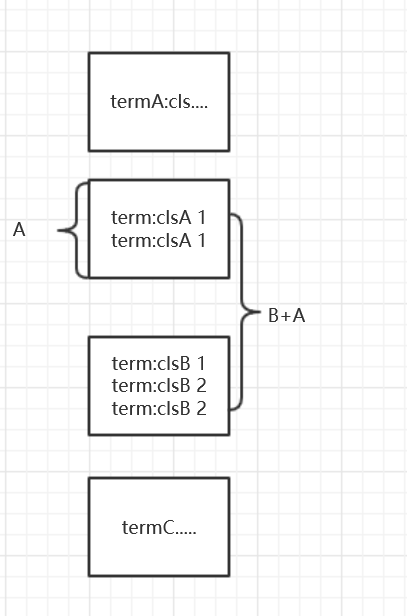
<2>Partitioner：将<term,cls>形式key按照term进行partition，保证每一个term对应的不同cls的键值对能分到同一个Reducer

<3>Reducer：

(1)将(<term,cls>,1)形式键值对的value累加可以得到表格中对应的A值。

(2)然后将每一个term不同cls的A值累加可以得到A+B值，如此可以计算出B。

具体计算如下:



(3)然后A+C对应的是属于某一个cls的所有文档总数。我们可以十分简单的通过文件系统操作来得到这一值。这一部分实现，是在Job开始前对每一个种类的文档数目进行统计，如此在使用时比较方便。

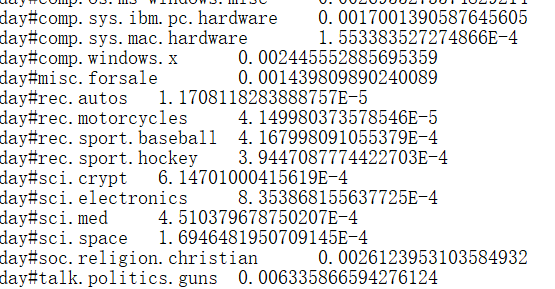
(4)D = N-A-B-C，其中N也可以通过文件系统操作实现。

最后以(<term,cls>, chi2Mark)的形式写结果。

说明: (1)关于Mapper中为什么使用<term,cls>而不是<cls,term>是因为如果使用cls作为排序和partition首要依据，则会因为难以计算A+B的值。由此， B的值无法计算。

(2)注意(A+B)(A+C)(B+D)(C+D)和(AD-BC)^2可能会超过int值，然后会导致结果为NaN，所以建议使用long,或者在后面进行一步过滤。

<4>效果:



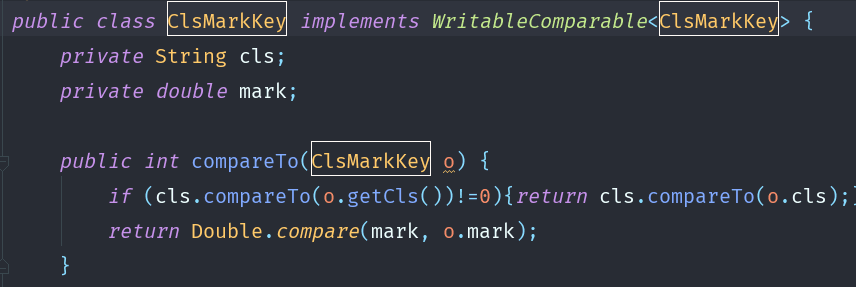
1. 排序

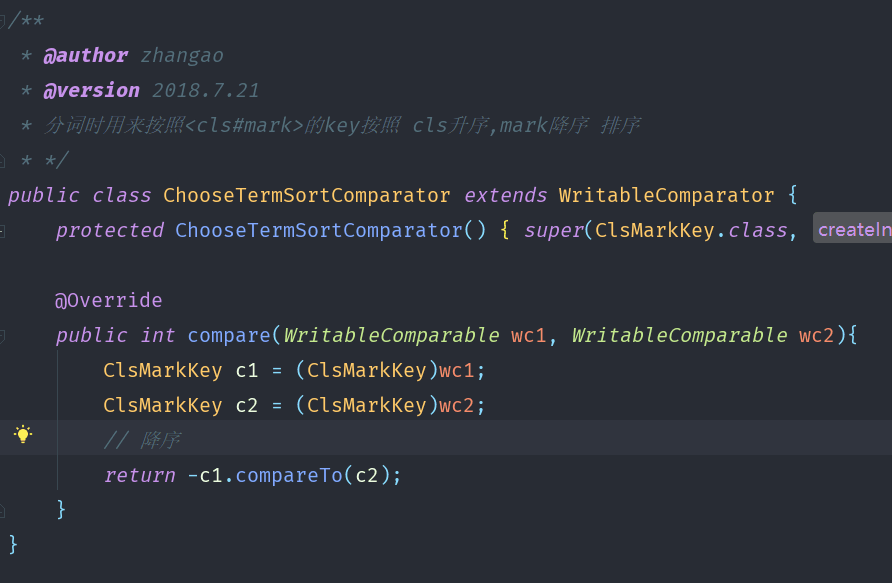
用JobControll组织，保证具有依赖关系的Jobs运行正确

<1>输入文件: 上一步的输出，即每一个term和cls的相关性

<2>Mapper: 对于每一个读入的(<term,cls>, mark)进行重新组织，组织为(<cls, mark>, <term, mark>)的形式进行输出。

说明: (1) key为<cls, mark>，是为了利用map之后，reduce前的sort进行排序。排序按照cls降序，mark降序进行。排序之后使得每一个cls内部的term是按照，mark从高到低排列。具体实现是通过自定义复合键+自定义SortComparator实现:





(2)val为<term,mark>，我们知道key中已经存储了一份mark，那么为什么这里还要再存一边呢？这是因为，下面自定义了GroupComparator，会按照term进行reduce的分组，导致部分mark值会丢失。重复存储在value中可以确保mark的可访问性。

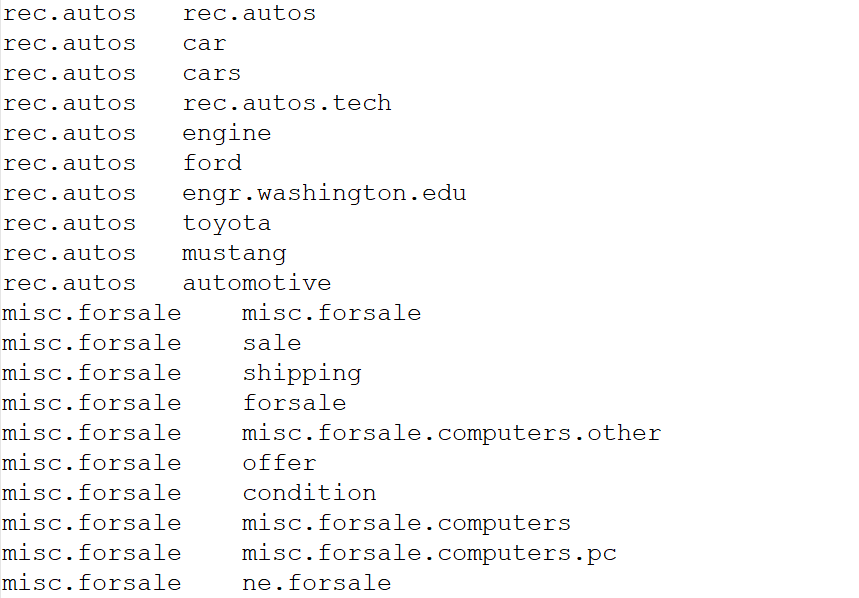
<3>Partitioner：<cls, mark>按照cls进行partition

<4>GroupComparator:<cls,mark>按照term进行group

具体使用参考了<https://www.cnblogs.com/datacloud/p/3584640.html>

<5>Reducer:将同一cls内排名最高的m个单词以(cls,<term,mark>)形式输出

<6>效果: 为方便使用没有输出mark



1. 训练集向量化

因为时间紧迫，我们计划使用较为熟悉的离散型朴素贝叶斯算法，所以向量化就使用的相对应的one-hot编码。也就是文档内有该特征词，则该维度的向量位置1.否则置0.

<1>输入：Train数据集

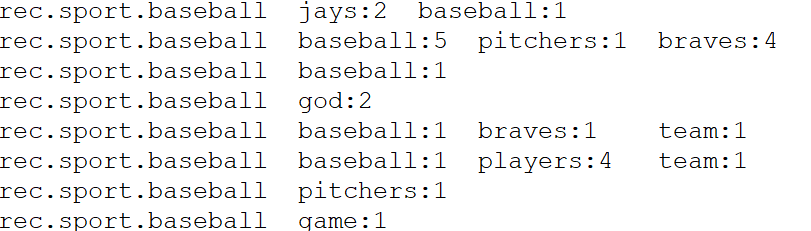
<2>DistributedCache：特征词数据(上一步输出)

<3>Mapper:按照(<fileName, cls>,<term, 1>)输出

<4>Partitoner：无

<5>Reducer：输出(cls, term1:n1 term2:n2…)

<6>效果：



1. 朴素贝叶斯训练

朴素贝叶斯原理，不多加赘述，可以参考南京大学大数据课程教材。

目的:统计每一个cls内Prob(term|cls)=freq(term,cls)/freq(cls)

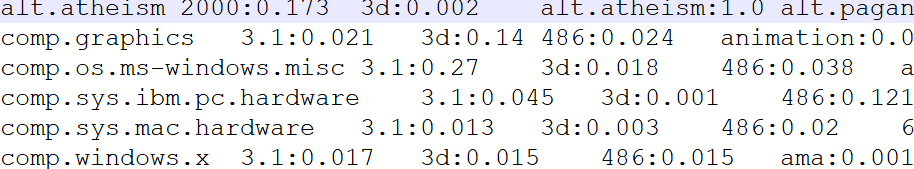
<1>Mapper:输出(<cls,term>,1)形式键值对

<2>Partioner: 按照term进行分类

<3>Reducer:利用Job开始前的文件系统操作可以比较容易的得到freq(cls)，然后累加value得freq(term,cls)可以计算得Prob(term|cls),以(cls, term1:prob1 term2:prob2…)形式输出。

说明:书上的部分要稍微麻烦一些，因为还要在MapReduce任务统计freq(cls)。这里我们由于使用文件系统统计了freq(cls)，所以显得简便了一些。

<4>效果:



1. Test向量化

与Train的向量化过程相同，不多加赘述。

1. 应用分类算法并统计正确率

我们开始时使用了朴素贝叶斯，后来因为朴素贝叶斯效果太差，直接采取了不同类别的特征词匹配计数法，即对 **待分类** 文档的词向量，进行每个cls下的特征词匹配，选取特征词匹配数量最多的一个cls作为预测结果。

1. 朴素贝叶斯的测试

<1>输入:向量化的Test

<2>DistributedCache:朴素贝叶斯的训练后参数

<3>Mapper:将每一个待分类文档词向量按照p=prob(cls)\*prob(term1=1|cls)prob(term2=0|cls)…的公式进行计算。选取可能性最大的一个cls作为预测分类，如果预测分类和实际分类相同，则输出(xx,true)。xx无关紧要，只要保证分到同一个Reducer即可。

说明:prob(cls)=freq(cls)/N,N为总文档数量，由于N的一致性,计算时使用freq(cls)即可

prob(term=0|cls)=1-prob(term=1|cls)

<4>Reducer:累加总共的统计次数和正确次数，输出(准确率, 正确次数/总次数)

<5>效果: 在debug阶段选取train中数据进行小样本测试时，效果极好，能达到80%+，而且出现过90%+。 然而在测试集上实测时，准确度不高，没有超过过50%。

我们发现了一些问题比如,train当中会提取出一些诸如类别名一类词作为关键词，但是在test中没有。于是我们尝试筛选掉这些多余关键词，然而效果依然不好。此外我们尝试增加选取的特征词数量。最后效果都不令人满意。

1. 特征词匹配计数法

输入:向量化的Test

<1>DistributedCache: 每一个类别以及对应的关键词

<2>Mapper:将每一个待分类文档词向量，进行不同类别上的特征词匹配,选取匹配数量最多的一个cls作为预测的分类。

Reducer：与朴素贝叶斯的预测结果相同。

<3>效果:每个类别选取10个特征词时,准确率可以达到70%+，我们又采取了一些关键词词频的加成，最后效果达到了77%左右。之后尝试增加特征数量，无明显的突破。

C:\Users\zhangao\Documents\Tencent Files\3104376275\Image\C2C\L85UM`]BGL2EA_OG$~JW(U4.png

1. 反思

首先特征选择部分比较合理。

我们觉得问题主要出在编码加分类器。

首先one-hot编码显得比较薄弱，无法体现词频的重要性。如果继续尝试，我们会选择TF\*IDF来计算词向量，然后分类算法采取SVM。或者one-hot+随机森林。