**金融大数据实验四报告**

——161278037 肖扬

**一. 实验目的：**

使用多种机器学习算法对文本进行情感判别，包括KNN、决策树、朴素贝叶斯、支持向量机等，学习如何进行模型训练，如何进行分类预测。要求使用至少两种分类方法。

**二. 实现总体思路：**

先将所有文本转化为多维空间中的点，再用不同的方法进行分类预测。

故一共四个步骤：

1. 由训练集生成特征向量，并附带情感标签

2. 由测试新闻数据生成待预测向量

3. KNN预测

4. NaiveBayes预测

具体的：

合并文件.py是一个py脚本，实现功能为将train\_data中的小文件合并，输出文件格式为文本+情感标签（以数字1,2,3代替3种情感）

DataCreate.java实现步骤1，DataNewCreate.java实现步骤2

KNN.java实现knn分类预测，NaiveBayes.java实现朴素贝叶斯分类预测。

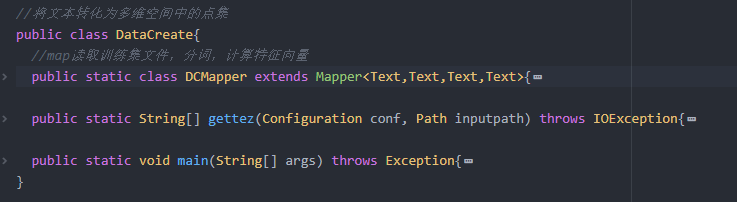
**三. 代码具体实现**

1.python脚本

直接读取文件夹下所有文件内容即可，然后根据文件夹名称附加标签，为和hadoop保持一致，用“\t”作为分隔符。

2.DataCreate.java

代码结构：

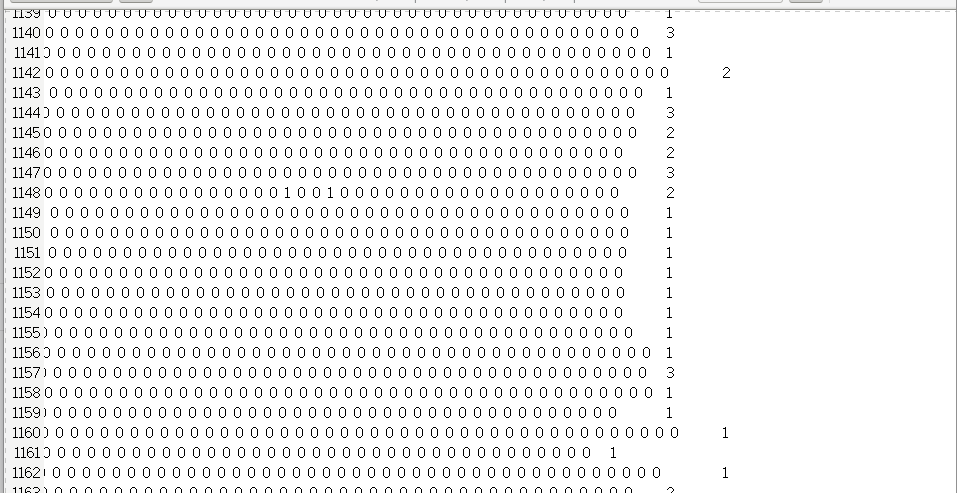


读取文件为python脚本生成的combine.txt

Map阶段对每行数据进行分词，将chi\_word.txt中的词作为特征词，先生成一个相应维度的0特征向量，如果分词结果中有特征词，则相应的特征值加1，输出<特征向量，情感标签>

gettez函数读取chi\_word.txt中的词作为特征词，再在main中设置为全局变量分发给节点。

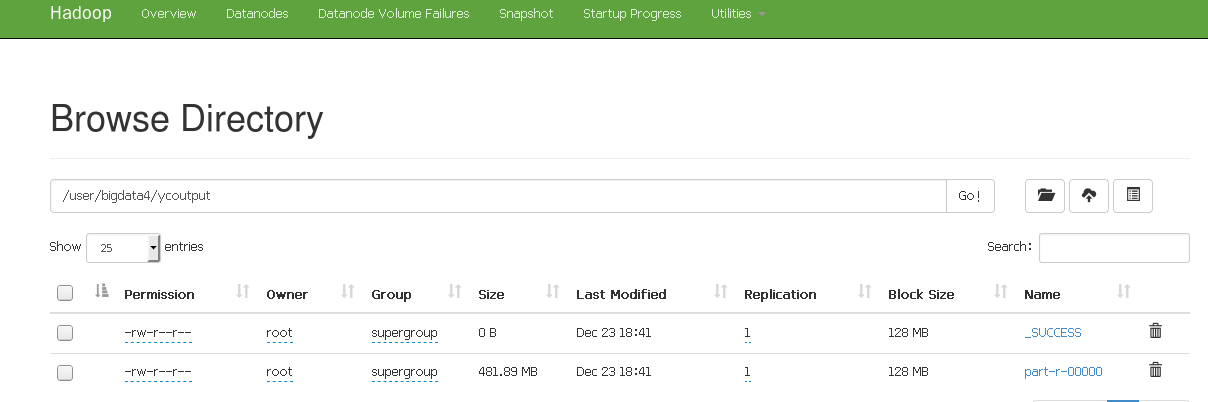
结果：



（训练集）

3. DataNewCreate.java

与生成训练点集的代码类似，不同之处在于map输出结果为<股票代码+新闻标题，特征向量>。具体实现由于文本结构的不同而有少量差异。结果如下：



（生成待预测点集，大小为481M）

取其中少量数据作为后续分类预测算法的测试数据test.txt，内容如下：



4. KNN.java

代码结构如下：



其中getdistance计算点之间的距离，为适应map中的输入，该函数输入了String格式的点和int[]格式的点。

gettrain得到训练集中的数据

map阶段实现功能为统计最靠近待预测点的K的点的情感标签。通过对训练集和最小的K个距离进行遍历，复杂度为O(mK)，m为训练集大小。输出为<新闻，List(情感标签)>

Reduce阶段实现统计每条新闻对应的情感标签中出现频度最高的一个，并转化为相应的情感，输出<新闻,情感>

输出结果见output文件夹

5. NaiveBayes.java

标准的朴素贝叶斯算法核心在于计算Fyi和FxYij，前者为第i个分类出现的频率（度），后者为向量X=（x1,…,xj,…,xn）中xj在类别Yi中出现的频率（度）。由此可计算.

最终的预测分类为

代码结构如下：



由两个mapreduce构成。

(1)train

train实现根据训练集计算所有Fyi和FxYij的功能。总体结构上类似于wordcount程序

Map根据读入的数据可以得到一个X和Yi，由此输出<Yi,1>和<Yi#xname#xvalue，1>。

Reduce对map中的值进行求和。

（2）test

实现分类预测，无需reduce

首先通过gettrainF函数得到train中的结果，根据有无#划分为两个数据集Fy和FXY，设置为全局文件。并在Mapper的setup方法中将数据处理为适当的类型便于计算。

Map读入带预测集中的一行数据，得到一个待分类X。输出预测分类。

算法为：

*对每个Yi，找到Fyi*

*对每个xj，找到FxYij，若没有对应的Yi#xname#xvalue则FxYij=0*

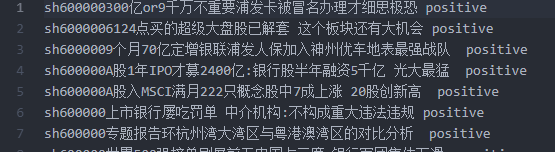
*计算FxYij的积*

*计算指标Fyi\*FXYi*

*选出最大的指标对应的i，得到情感，输出<新闻,情感>*

结果见output文件夹

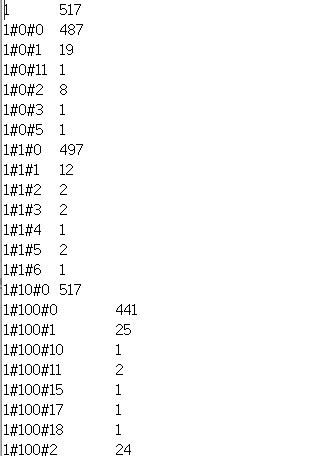
**四. 运行结果。**



最终结果详见output文件夹

训练集和待预测集已在前文给出。

另外一个重要的结果是naivebayes的中间结果：



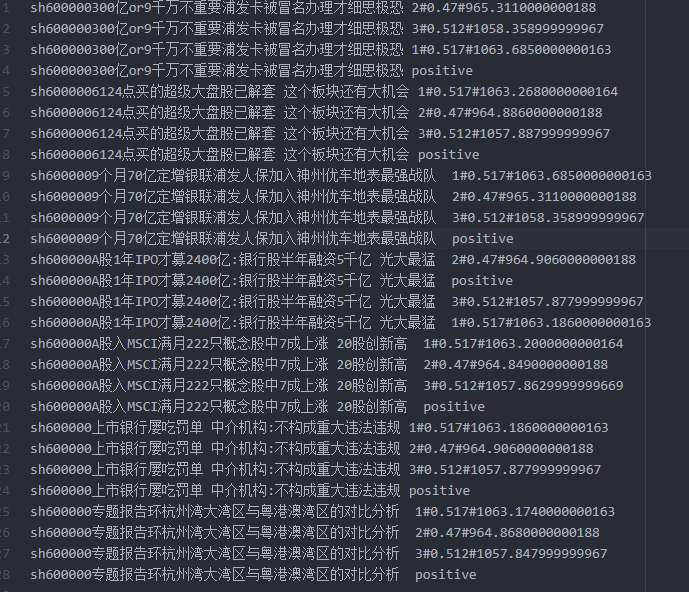
**五. 结果分析及改进。**

以上方法实现效果不够理想。

1. Knn中，由于点的维度太高（2000多维），加之本身算法复杂度高，运行慢且结果不好。需要做降维处理（暂未实现）和算法优化。暂无很好的改进方法。

2. NaiveBayes中，对于本实验中的点集，可以假设所有频率FxYij都不超过0.5，那么计算出来的积将小于，或者是通过频度计算，此时积将很容易大于已经，都超过了double类型能精确表示的范围。因此，在机器精度的前提下，所有的FXYi都相等，所做计算无意义，结果显然也无意义。针对高维数据，简单的改进为计算FXYi改为求和的方式，即：

Sourcecode中的NaiveBayes.java实现的是该计算方式。此时对test.txt的代码测试如下：



以第一则新闻为例，代码测试中输出了4条信息，

第一行为<情感标签2#Fy2/1000#FXY2>，意义为新闻在第2类情感上的指标为0.47\*965.311,而该条新闻的特征向量每个的值都是0，简单查看并计算naivebayes生成的中间文件中的结果知该代码执行是正确的。对前3行进行计算知第1类情感确实是最优的，即情感为positive，对应于第4行。

以上测试说明，NaiveBayes.java执行了正确的朴素贝叶斯分类。

因此，可以合理推断，**数据集是导致分类预测结果不够理想的主要原因**。

由以上分析知，对于本实验还可以**对文本的预处理进行优化**。具体方向为：

（1）生成点集的降维处理。高维数据存在“维数灾难”这样一个问题，一般来说，在机器学习算法中都需要进行降维处理。

（2）特征词及特征向量的优化。可以看出，特征向量过于稀疏，一方面，这是因为新闻标题的简短和特征词文件中存在一些不足以作为特征的词；另一方面，简单的对新闻标题中出现的特征词进行计数也可能存在问题。

（3）放弃（2）中的想法，将文本通过Word2Vec转化为词向量。

**六. 实验总结**

1. 分类算法一般都有较大的计算量。分类算法中knn虽然实现简单，但是复杂度很高，不适合单机运行。实际上，即使可以并行运算，knn算法依然会有计算效率的瓶颈。

2. 文本分析中，机器学习算法虽然重要，但最重要的还是文本特征的构建。不同于普通的分类预测问题，文本的特征是不明确的，需要进行复杂的预处理才能用于分析。如果只进行简单的处理，其预测结果往往不够理想。这一问题是不可能通过机器学习算法预测准确度的提高来解决的。因此，文本分析第一步也是最重要的一步是文本特征的构建。

3. 高维数据不可取。