金融大数据实验 2 实验报告

151220006 陈安宇

一、实验需求

对上市公司财经新闻进行情感分析。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、实验设计说明

本次实验我使用的是 KNN 与朴素贝叶斯的方法对模型进行训练。

实验步骤：

1 选取特征词汇

由于词频高的词可能是一些无意义的词，所以我选择用 TF-IDF 值选择特征词。

分别在 negative、positive、neutral 中运行 ChooseWord 函数，分别得到三个文件夹下每个词的 tf-idf 值。

ChooseWord 函数说明：总共分为 3 个 mapreduce 阶段

阶段一：统计每个词在文件中的词频/总词数

Key:词 文件名 value:TF

阶段二：计算每个文件中单词的 tf-idf

Key:词 value:文件名 TF TF-IDF

阶段三 ：加和词的 TF-IDF,输出该词在整个文件夹下的 TF-IDF

Key:词 value:TF-IDF

执行完 ChooseWord 后，在三个文件夹下都会生成一个格式为“词 -> TF-IDF”的文件用 CompareWord 分别对以上生成的三个文件按照 TF-IDF 排序。生成三个按 TF-IDF 值降序排列的格式为“TF-IDF -> 词”的文件。在三个文件中分别选 10-15 词，组成一个

含有 44 个特征词的特征词文件。

说明 1： CompareWord 的思路与实验 1 中词频降序排列类似，要注意的是 TF-IDF 是浮点数，因此在排序时可以先乘 1000，再转化为整数排序。

说明 2： 因为本次的实验训练集中一个文件中词数较少，因此只选择 44 维度作为文本向量化的维度。

用 LoadFeature 函数将特征词加载一个数组 loadfe，便于文本向量化的时候调用。

下面是特征词的截图：

2.文本向量化

TextVector 函数对于**训练集**进行文本向量化。其中计算每个词的 TF-IDF 的思路与选取特征词类似。将一个文件中句子分完词，并计算出每个词的 TF-IDF 后，调用 LoadFeature 函数。在特征词数组 loadfe 的每个维度上判断该文件是否含有该词，若有，则把该词的 TF-IDF 写入向量的对应维度，若没有，则在该维度上写 0。循环判断特征词组的每个维度后，一个文本文件就被转化为了一个向量。分别在三个文件夹下运行 TextVector，每个文件夹下的文件都会转化为向量写入对应的文件中。最后三个文件夹下的文本文件转化为三个写有向量的文件 positiveVec.txt，negativeVec.txt，neutralVec.txt。

说明：为了程序运行的简洁性，我将 negative 设为 label 设为 0，neutral 设为 1，positive 设为 2。因此须在三个文件夹下分别运行 TextVector，且随着运行的文件夹的不同，TextVector 函数每次的 key 值等于对应的 label 值。

最后训练集的文本向量化输出格式为“label -> 对应的向量（每一维度间用逗号隔开）”

用 TextVectorTestData 函数对**测试集**向量化。整体思路与训练集文本向量化一致。本次实验我是用实验 1 中的 download_data 中的一部分作为测试集。其中一个文件转化为一个向量。输出的格式为“文件名 -> 对应的向量（每一维度间用逗号隔开）”

将训练集与测试集都转化为向量后，用 KNN 和朴素贝叶斯进行训练。

下面是 negative 文件夹下文本向量化的截图：

3.分类算法

KNN:

算法思想：采用 KNN 算法对文件进行分类时，首先要计算待分类 文件与训练数据集中所有文件的相似度（夹角余弦值）。然后挑选出相似度值最大的 K 个训练文件，并将待分类的文件分到 K 个训练文件中多数属于的哪个类别中。

实现步骤：

1.将训练集的向量数据读入 `float[] inData` 中，将所属的类读入 `oType` 中，将测试集的数据读入 `testData` 中

2. `Mapper`:遍历所有测试数据，调用 `calDistanceAndInsert` 计算每个训练数据和测试数据的距离

3. `Reducer`：初始化 k 个近邻值，将多个 Mapper 的值合并到一起，更新 k 紧邻值，计算预测值,写入 Output

4. `MergeDriver`:如果 `KnnDriver` 的 `Reducer` 个数有多个，那么需要把结果合并,按照 id 顺序排序。

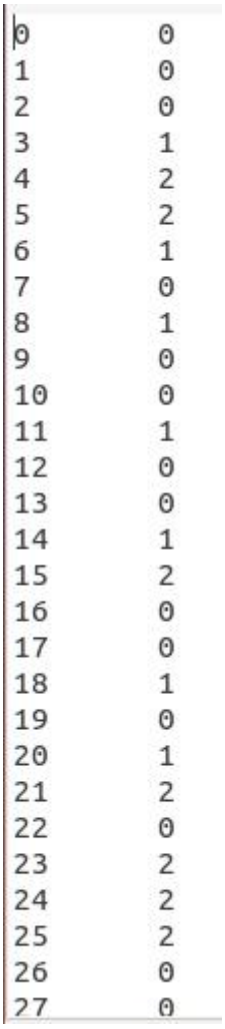
输出格式“向量对应行数 -> label”

程序运行：`./bin/hadoop jar ./share/hadoop/mapreduce/knn.jar knn.KnnDriver -i input15 -t input14/TestVector.txt -o output12 -knnk 5 -reducernum 1 -column 45 -delimiter ,`

其中-knnk 后面输入的是 k 个训练集；-reducernum 后面输入的是 reduce 的个数；-column 后面输入的是向量维度加 1；-delimiter 后面输入的是切分数据的符号

下面是 knn 运行结果的截图：此时选的是 K=3

文件名



其中左边是行数，右边是属于的类名。0 代表 negative,1 代表 neutral,2 代表 positive。

朴素贝叶斯：

算法思想：

贝叶斯分类算法的理论基于贝叶斯公式，P(B|A)=(P(A|B)P(B))/P(A) ，其中 P(A|B)称为条件概率，P(B)先验概率，对应 P(B|A)为后验概率。

朴素贝叶斯分类器基于一个简单的假定，即给定的目标值属性之间是相互独立。贝叶斯公式之所以有用是因为在日常生活中，我们可以很容易得到 P(A|B)，而很难得出 P(B|A)，但我们更关心 P(B|A)，所以就可以根据贝叶斯公式来计算。

应用在文档分类中有：P(c|d)=(P(c)P(d|c))/P(d)，而对于一份文档在分类时则考虑其中的每个单词的概率在该类中出现的概率 P(d | c)=P(tk1 | c)P(tk2 | c)...P(tkn|c)，判断每个文档属

于哪个类别时时总有 P(d)不变，所以等式左边正比于分子，即公式：

$$P(c|d) \propto P(c)\prod_{1\le k\le n} P(tk|c)$$

最后对于待分类文档，求出该文档在各类别中的概率，哪个概率大就将该项分类到对应的类别中，即$c = \operatorname{argmax}_{c\in C} P(c|d) = \operatorname{argmax}_{c\in C} P(c)\prod_{1\le k\le n} P(tk|c)$

实现步骤：分为 2 个 mapreduce 阶段

阶段一：读入训练集数据

计算每个 label 出现的频度，存入 label 数组中。格式“类名 频率”

计算每个 label 中每个属性属性值出现的频度，存入数组 frequence 中。格式“类名 属性名 属性值 频率”

阶段二：读入测试集数据

对于每一个测试集数据：在不同类下该向量每个属性值 x_i 出现在该类的频度与该类在出现的频度相乘，每个类得到一个值。找出所有类的最大值。该测试集属于这个类。

输出格式“文件名 -> label”

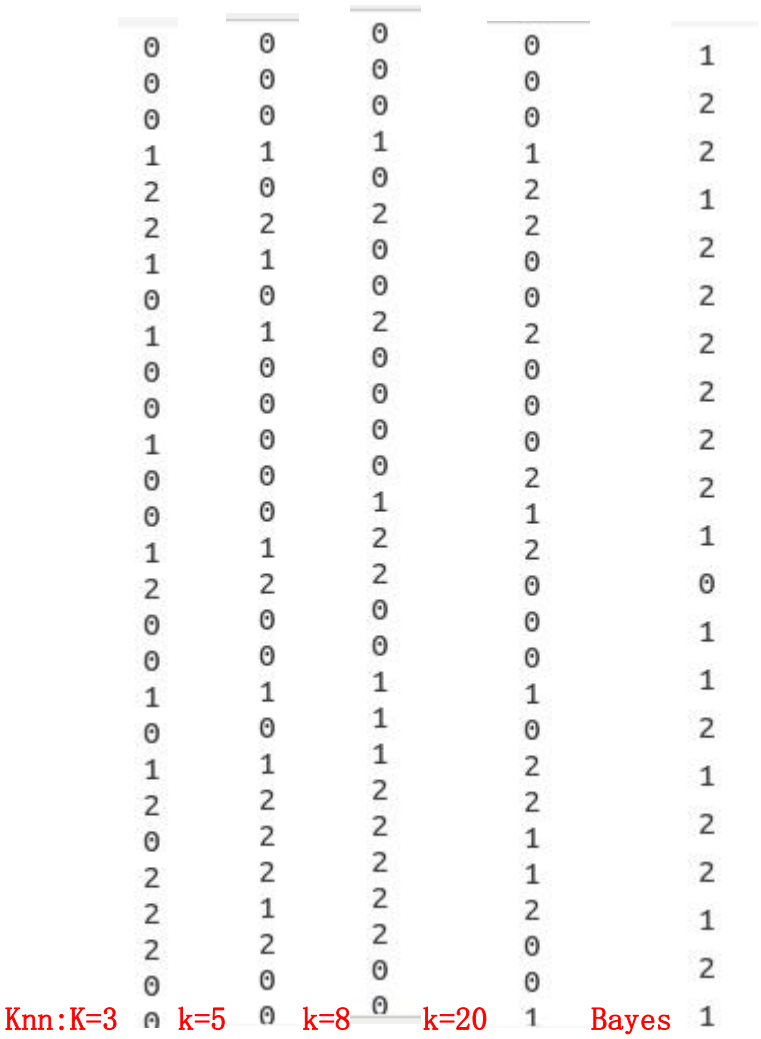
程序运行：./bin/hadoop jar ./share/hadoop/mapreduce/NaiveBayes.jar NaiveBayes input15 output11 input14 output

下面是朴素贝叶斯运行结果的截图：

sh600000浦发银行.txt	1
sh600004白云机场.txt	2
sh600006东风汽车.txt	2
sh600007中国国贸.txt	1
sh600008首创股份.txt	2
sh600009上海机场.txt	2
sh600010包钢股份.txt	2
sh600011华能国际.txt	2
sh600012皖通高速.txt	2
sh600015华夏银行.txt	2
sh600016民生银行.txt	1
sh600017日照港.txt	0
sh600018上港集团.txt	1
sh600019宝钢股份.txt	1
sh600020中原高速.txt	2
sh600021上海电力.txt	1
sh600022山东钢铁.txt	2
sh600023浙能电力.txt	2
sh600026中远海能.txt	1
sh600027华电国际.txt	2
sh600028中国石化.txt	1

0 代表 negative,1 代表 neutral,2 代表 positive。

4.实验结果对比：



5.实验不足之处：

1.可以通过结果看到 KNN 算法选取不同 K 值时对文本的分类结果不同，且 KNN 算法与朴素贝叶斯算法对文本的分类也存在差异。这其中有对 KNN 的 K 值的选择问题。K 值的选取会对 KNN 算法的分类结果产生极大影响。当 K 值较小时，就会用较少的训练实例对输入实例进行预测，这时“学习”的近似误差会 很小，因为只有与输入实例相似的训练实例才会对预测的结果起作用。但是这样会增 加“学习”的估算误差，因为相似的实例点对预测结果影响很大。如果相似的实例点 恰好是噪声，那么预测就会出错。也就是说，较小的 K 值会使整体模型变得复杂，发 生过拟合的概率会增大。当 K 值较大时，就会有较多的训练实例对输入实例进行预测。这样虽然可以减少 学习的估计误差，但是学习的近似误差会很大。这时与输入实例不太相似（距离远） 的训练实例也会影响预测的结果，并且容易发生错误。也就是说，较大的 K 值就意味 着整体模型会变得简单。

2.特征词的选择问题。通过 TF-IDF 可以舍弃一些在整个文件夹中出现次数较多但无情感倾向的词。但一些在某个文档中出现频率高其他文档中不出现的专有词汇，即使不具有区分情感的功能，其 TF-IDF 也会较大。

6.可扩展性与可改进之处：

可以进一步比较 KNN 取不同 K 值时与贝叶斯算法的结果的相似性，从而选取较好的 K 值。