实验4：MapReduce高级编程 上市公司财经新闻情感分析

实验目的

1. 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。
2. 核心算法在MapReduce上运行，通过实验更深入地理解MapReduce的各种特性。

实验设计说明

提交文件结构

本次实验中，采用了7个MapReduce过程完成了文本情感分类，使用Python进行了训练集的爬取和一些中间文本转换的工作。

```
├─data
│   │  features.txt  \\选择出的1500个特征词汇
│   │  KNNnegativeVecMatrix.txt \\ KNN训练使用的消极、中性、积极词汇
│   │  KNNneutralVecMatrix.txt
│   │  KNNpositiveVecMatrix.txt
│   │  NBayes.conf   \\NBayes训练使用的配置文件和train test向量文件
│   │  NBayes.test
│   │  NBayes.train
│   │  scratchTestData.txt \\爬取的新闻向量化文件
│   │  test.csv  \\爬取的新闻集合
│   │  testDataScratch.zip  \\爬取新闻的原始文件
│   │  tfIdf_test.txt \\爬取新闻的tfidf
│   │  trainData.txt \\训练集向量文件
│   │
│   ├─algorithm_test_vec  \\算法评估部分向量文件
│   │      KNNtest.txt
│   │      KNNtrain.txt
│   │
│   └─tfIdf_train  \\训练集的tfidf
│           negativeTfIdf.txt
│           neutralTfIdf.txt
│           positiveTfIdf.txt
│
├─results
│   │  scratchBayesResult.txt    \\Bayes的分类结果
│   │  scratchBayesResultCpr.csv \\Bayes分类结果与原文本对比的展示
│   │  ScratchKNNresult.txt  \\KNN分类结果
│   │  scratchKNNResultCpr.csv \\KNN的分类结果与原文本对比的展示
│   │
│   └─algo_evaluation  \\算法评估中的部分结果
```

```
|          evalu1
|          evalu2
|          evalu3
|          evalu4
|          evalu5
|
└─src
    ├─java
    |   |  Word2Vec.java  \\文本向量化
    |   |
    |   ├─KNN   \\KNN训练算法
    |   |      Distance.java
    |   |      Instance.java
    |   |      KNNDriver.java
    |   |      ListWritable.java
    |   |
    |   ├─NBayes  \\NBayes训练算法
    |   |      NaiveBayesConf.java
    |   |      NaiveBayesMain.java
    |   |      NaiveBayesTest.java
    |   |      NaiveBayesTrain.java
    |   |      NaiveBayesTrainData.java
    |   |
    |   └─tfIdf  \\tfIdf计算
    |          TfIdf.java
    |          WordCountPerDoc.java
    |          WordsTotalDoc.java
    |
    └─python
           featuresSelected.py  \\特征向量的筛选
           jsonFile.json
           resultConvert.py  结果的格式转化
           sratchData.py \\新闻爬取
           trainIdConvert.py \\训练集的格式转化
```

设计思路

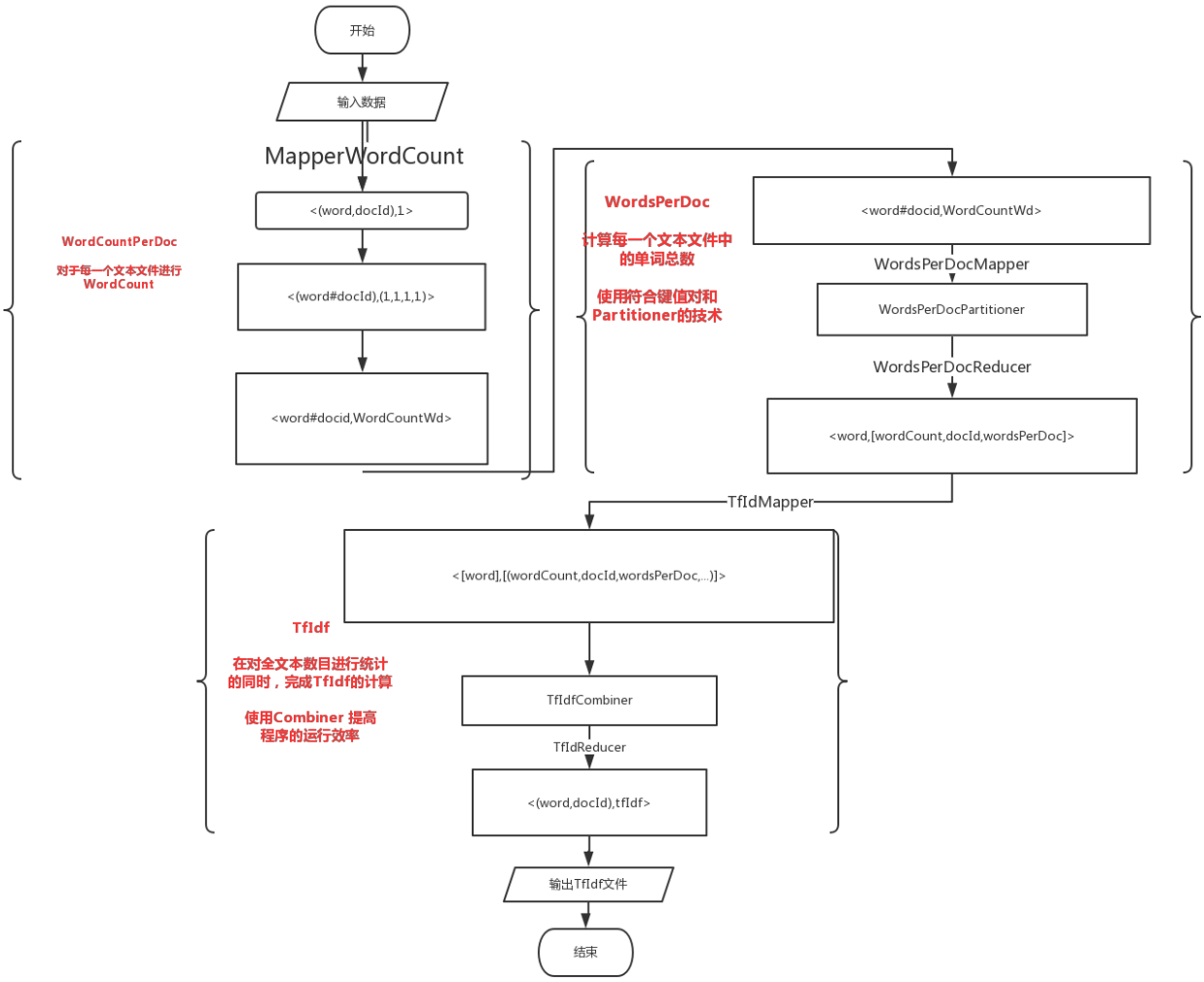
- 用机器学习算法进行文本分类的基本思路就是使用训练集训练出一个模型，使得此模型能够作为一个黑箱，将测试集作为输入，就能得到文本分类的结果。这其中需要有五个步骤：文本预处理，文本向量化，训练，预测，评估。
- 对于文本预处理阶段，使用Tf-Idf来描述词汇所包含的信息量大小；
- 文本向量化阶段选择一些特征，对于文本进行向量化，将每一个文本转化成一个高维矩阵；
- 训练阶段分别采用KNN和NBayes对模型进行训练；
- 预测阶段：由于实验所给训练数据的质量并不好，所以采用了自己爬取的财经新闻测试集进行了分类；
- 评估：将训练集按照3个类别成比率地分出120条作为测试集，作为交叉检验，多测测试，取正确率的平均值。

```
http://naotu.baidu.com/file/8cb2c2f7c0b37d0ff0246c9aed8adf88?token=e9277fa59825c206
```

这是使用思维导图完成的整个项目的原始初始设计，和最终的实现有一定的距离，比如最初选择了1000个特征值，最终选择了1500个等，如果有兴趣可以查看

TfIdf计算

流程图



核心类说明

类名	作用	核心子类	输入	输出
WordCountPerDoc	对于每个训练集的文本进行词频统计。	WordCountPerDocMapper	(line ,offset)	((word,docId),1)
		WordsPerDocReducer	((word,docId),1)	(word#docid,WordCountWd)
WordsPerDoc	计算每个文本中单词总数。	WordsPerDocMapper	(word#docid,WordCountWd)	(word#docid,WordCountWd)
		WordsPerDocPartitioner	确保所有同DocId的被分类再一起	
		WordsPerDocReducer	(word#docid,WordCountWd)	(word,(wordCount,docId,wordsPerDoc）)
Tfidf	在完成全文文本数量的统计的同时，完成Tfidf的计算。	WordsPerDocMapper	(word,[wordCount,docId,wordsPerDoc])	(word,[wordCount,docId,wordsPerDoc])
		WordsPerDocCombiner	对同节点的(k,v)对进行初步的合并	
		WordsPerDocReducer	([word],[(wordCount,docId,wordsPerDoc,...)])	((word,docId),tfidf)

文本向量化

特征的选择-features.py

在每一个情感分类中选在tfIdf排名前1000的词汇，将三个分类的词汇合并之后，从中选出1500个词汇作为最终的特征，这样进行选择的原因是，有研究显示，只要训练样本的数据足够大，算法的有效性会趋同，前1500个词汇的tfidf还相对较高，排名更靠后的词汇就没有太多的有意义词汇了。

以下是筛选出的部分特征词汇


```
{"减磅": 3135.680861737221, "武夷": 2743.720754020069, "电力": 2664.3747344687968, "电梯":
2663.362416870104, "周": 2612.5204818429156, "套现": 2367.4332594400926, "交易日":
2135.7762023073237, "惠": 2038.3814659718655, "中直": 2021.7232810785988, "石油":
2011.8281830479152, "待定": 1975.8898852264467, "逢": 1928.3979474311402, "泉":
1907.4622778218493, "佳": 1899.5359673050157}
```


文本向量化-Word2Vec.java

对于文本进行向量化，将tfidf转换成1500维的稀疏矩阵

- 程序输入 args=[inputPath, featuresPath, cl(num)]
- Word2VecMapper
  - 输入：(offset,((word,docId),tfIdf)
  - 输出：(docId, (word, tfIdf))
  - 功能：转换文本格式，用于给Reducer传递数据
- Word2VecReducer
  - 输入： (docId, (word, tfIdf)...)
  - 输出： (docId (tfIdfVec, cln))
  - 核心功能： 将每个文本转化成向量

```
for(String currWord:featureList){
                if(tfIdfs.containsKey(currWord)){
                    vec=vec+" "+tfIdfs.get(currWord);
                }
                else{
                    vec=vec+" "+"0";
                }
            }
```

KNN

设计思路

> 在模式识别领域中，**最近邻居法**（**KNN**算法，又译**K-近邻算法**）是一种用于分类和回归的非参数统计方法。
> 在这两种情况下，输入包含特征空间（Feature Space）中的**k**个最接近的训练样本。

> 最近邻居法采用向量空间模型来分类，概念为相同类别的案例，彼此的相似度高，而可以借由计算与已知类别案例之相似度，来评估未知类别案例可能的分类。

程序结构

- Distantce.java
  - 计算两个点之间的欧氏距离
- Instance.java
  - 定义了一条数据的数据结构。
  - 属性包括：Id（文本id），attributeValue（tfIdf向量），label（本条文本的属性）。
- KNNDriver.java
  - KNNMapper
    - 输入：(offset,trainLine)
    - 输出：(testId,(label,distance))
    - 功能：首先在缓存汇总取到测试集文件，重载setup将测试集文件初始化，计算一条trainLine到每一个训练集的距离，并将训练集id作为key发送出去。
  - KNNReducer
    - 输入：(testId,(label,distance)...)
    - 输出：(testId,label)
    - 功能：对统一testId下的距离进行排序，取前k个最近的邻居，最后最近的邻居们投票，选择出测试集的最终分类。
- ListWritable.java
  - 定义了一个Writable的类，便于对于邻居们的排序和最终的投票选择。

KNN输入文件

训练集 格式：【docId】【tfidf值1】【tfidf值2】…【tfidf值1500】【 分类标号】 示例： （以二维数据为例）

```
negative0 1.233 2.333 …… 1
positive98 2.8998 3.4444 …… 3
```

训练集 格式：【docId】【tfidf值1】【tfidf值2】…【tfidf值1500】 示例： （以二维数据为例）

```
89 1.233 2.333 ……
63 2.8998 3.4444 ……
```

NBayes

设计思路

理论上，概率模型分类器是一个条件概率模型。

$$p(C|F_1,\ldots,F_n)$$

独立的类别变量C有若干类别，条件依赖于若干特征变量

$F_1, F_2, \ldots, F_n$。但问题在于如果特征数量n较大或者每个特征能取大量值时，基于概率模型列出概率表变得不现实。

所以我们修改这个模型使之变得可行。

贝叶斯定理有以下式子：

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\,p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

用朴素的语言可以表达为：$\mathrm{posterior} = \frac{\mathrm{prior} \times \mathrm{likelihood}}{\mathrm{evidence}}.$

对于朴素贝叶斯概率模型。一个普通的规则就是选出最有可能的那个：最大后验概率（MAP）决策准则。

相应的分类器便是如下定义的公式：

$$\mathrm{classify}(f_1,\ldots,f_n) = \operatorname*{argmax}_c \; p(C=c)\prod_{i=1}^{n} p(F_i=f_i|C=c).$$

程序结构

NaiveBayesConf.java 用于处理配置文件。

NaiveBayesTest.java 用于分类过程的MapReduce。

NaiveBayesTrain.java 用于训练过程的MapReduce。

NaiveBayesTrainData.java 在测试之前读取训练后的数据。 NaiveBayesMain.java 主程序入口，由于完成整个过程的调度。

NBayes配置文件

配置文件NBayes.conf用于描述分类内容 格式：

- 第一行，第一个是分类的个数N，后面跟着N个字符串（空格分隔）每个代表类名
- 第二行，第一个是类中属性的个数M，后面跟着M个<字符串，整数>的分组
- 第二行的每个分组中的字符串是属性名，整数是该属性最大值

举例说明：3个分类，类名为cl1,cl2,cl3；分类有3个属性（即词组），为p1,p2,p3

```
3 cl1 cl2 cl3
p1 10000 p2 10000 p3 10000
```

NBayes.train 用来存放训练集 每一行描述一个训练向量，每行第一个为类名，后面接M个值，空格分隔，代表此向量各属性值
举例：

```
cl1 3 4 6
cl2 1 8 7
```

NBayes.test 用来存放测试集 每一行描述一个训练向量，每行第一个该变量ID，后面接M个值，空格分隔，代表此向量各属性值 举例：

```
1 6 9 3
2 4 8 1
```

实验技术实现

实验环境

- 代码编写环境IntelliJ IDEA+Hadoop2.9.1+Ubuntu16.04
- 代码调试环境：

```
BasicConfigurator.configure();// 采用在IDE中输出所有log，便于完成调试。
```

高级MapReduce程序设计方法的使用

- 复合键值对的使用
  - 在tfIdf运算的第一和第二阶段同时使用word#docId作为键值对，标识了更清晰的信息
- 自定义数据类型
  - 在KNN算法中定义了一个类似于列表的ListWritable，能将KNN中的节点装入List中，使得排序以及投票选择最终分类都更加方便。
- 链式MapReduce任务
  - 在TfIdf的运算中，使用链式MapReduce任务，对三个连续的任务进行了调度，减少了手工调度代码的复杂程度。
- 多数据源的连接
  - 对于训练集，由于在文本向量化的过程中将三个情感类型的训练集分类到了三个地址，需要在训练过程中对于文件进行连接，所以使用将文件复制到DistributedCacheFile中的方法将文件复制，完成了多元数据的连接。
- 全局作业参数的传递
  - 比如KNN中需要传递K的个数作为全局变量Configuration.set()方法来传递全局参数，并在setup()中解析全局参数。
- 数据文件的传递
  - 比如在KNN处理中，我们的测试集较小，所以对训练集进行并行化的处理。测试集不能从输入方法中载入Mapper，所以需要在作业Configuration时将文件存入Distributed Cache，并且在setup()的时候完成对于文件的解析。

实验主要工作总结

- 计算tf-idf：3个MapReduce Job完成
- 文本向量化：1个MapReduce Job 完成
- 训练及预测：
  - KNN: 1个MapReduce Job 完成
  - NBayes：2个MapReduce Job 完成

- 测试集爬取：
  - 获取了20181117到20181217的995条财经新闻数据
- 其他工作（id转换，特征选取，结果格式转换）
  - 使用Python脚本完成

实验结果

KNN

index	atri	context
0	negative	纳指跌幅收窄至0.5%。
1	neutral	美股五大科技股“FAANG”多数转涨，Facebook涨逾1%，亚马逊涨近1%，奈飞、谷歌均翻红。
2	negative	ICE原糖期货下跌3%，至12.41美分/磅，创1个月来新低。
3	negative	据路透：欧盟议会经济委员会投票支持任命Andrea Enria为欧洲央行银行监管机构主管。

NBayes

index	atri	context
0	neutral	纳指跌幅收窄至0.5%。
1	neutral	美股五大科技股“FAANG”多数转涨，Facebook涨逾1%，亚马逊涨近1%，奈飞、谷歌均翻红。
2	negative	ICE原糖期货下跌3%，至12.41美分/磅，创1个月来新低。
3	negative	据路透：欧盟议会经济委员会投票支持任命Andrea Enria为欧洲央行银行监管机构主管。

算法有效性评估

KNN

每次按照三种情绪的比例将老师给的训练集中选出8%的文件作为测试集，其他的作为训练集，测试多次取平均。

如下是分类的结果

> negative81 negative negative82 negative negative83 negative negative84 negative negative85 positive negative86 neutral negative87 negative negative88 positive negative89 positive negative9 negative negative90 negative negative91 negative negative92 negative negative93 negative

评估结果

属性\邻居个数	5	10	15	20	50
negative	60%	50%	30%	20%	15%
neutral	38%	32%	60%	75%	93%
positive	82%	79%	63%	34%	16%
	60%	54%	51%	43%	41%

NBayes

分类结果

> neutral62 neutral neutral63 neutral neutral64 neutral neutral65 neutral neutral66 neutral

评估结果

属性	negative	neutral	positive	总计
正确率	63%	84%	51%	66%

实验总结

可改进之处

- 代码结构整合有待提高
  - 采用了Python和Java的MapRduce两种语言系统，而且Python是在windows系统下完成的，MapReduce是在Ubuntu系统中完成的。在步骤的衔接过程中需要进行系统间手工复制文件等工作。完成整个项目需要多次手工操作。
  - 未来希望能使用MapReduce的链式任务调度将整个项目整合起来。提高代码的耦合程度。
- 设计和最终实现有所差距，中间的设计缺陷使用了一些Python程序进行修补
  - 比如说对于测试集分类结果输出的友好性，是在完成了MapReduce的分类之后，使用Python将最终分类和源文本整合在了一起，以便做出对比。
  - 未来可以对代码进行重构，将项目整合起来，使得项目能够自行完成全流程的工作。
- 分类的准确性大约在50%~60%，未来可以通过更新算法来提高准确性。

优点

- 自行爬取使用了与训练集相关度较高的测试集
  - 提供的测试集是新闻的标题，其中的信息量较少，而且时间上也和训练集的差距较大，训练集中除了有一些上市公司的财经新闻，也包括一些行业和宏观的新闻数据。所以自行收集了一些和训练集形式类似的时效性高的财经新闻。
- 使用了多种MapReduce的高级特性
  - 使用了MapReduce的多种高级程序设计的方法，提高了程序的效率

- 所有主要步骤都使用了MapReduc的并行化算法
    - 效率更高