

Project 2: MapReduce 高级编程

实验报告

课 程：金融大数据

实验名称：MapReduce 高级编程

学生姓名：李振安

学 号：151278019

时 间：2017 年 12 月 15 日

一、 程序设计思路

程序实现了挖掘新闻中蕴含的情感信息，使用机器学习算法对新闻情感进行判断和预测。本程序主要在 Hadoop 大数据框架中实现了对新闻文本的向量化处理，以及并行化的 KNN 和 SVM 算法，最终实现了对新闻感情的预测。

程序共有 4 个 job，功能如下：

Job1：分词并统计词频，以倒排索引方式输出

Job2：根据倒排索引计算 tf-idf，将文本向量化后以稀疏格式输出

Job3：并行化 KNN 算法

Job4：并行化 SVM 预测（训练过程使用 libsvm 库函数）

程序设计思路以及运行流程如图所示：

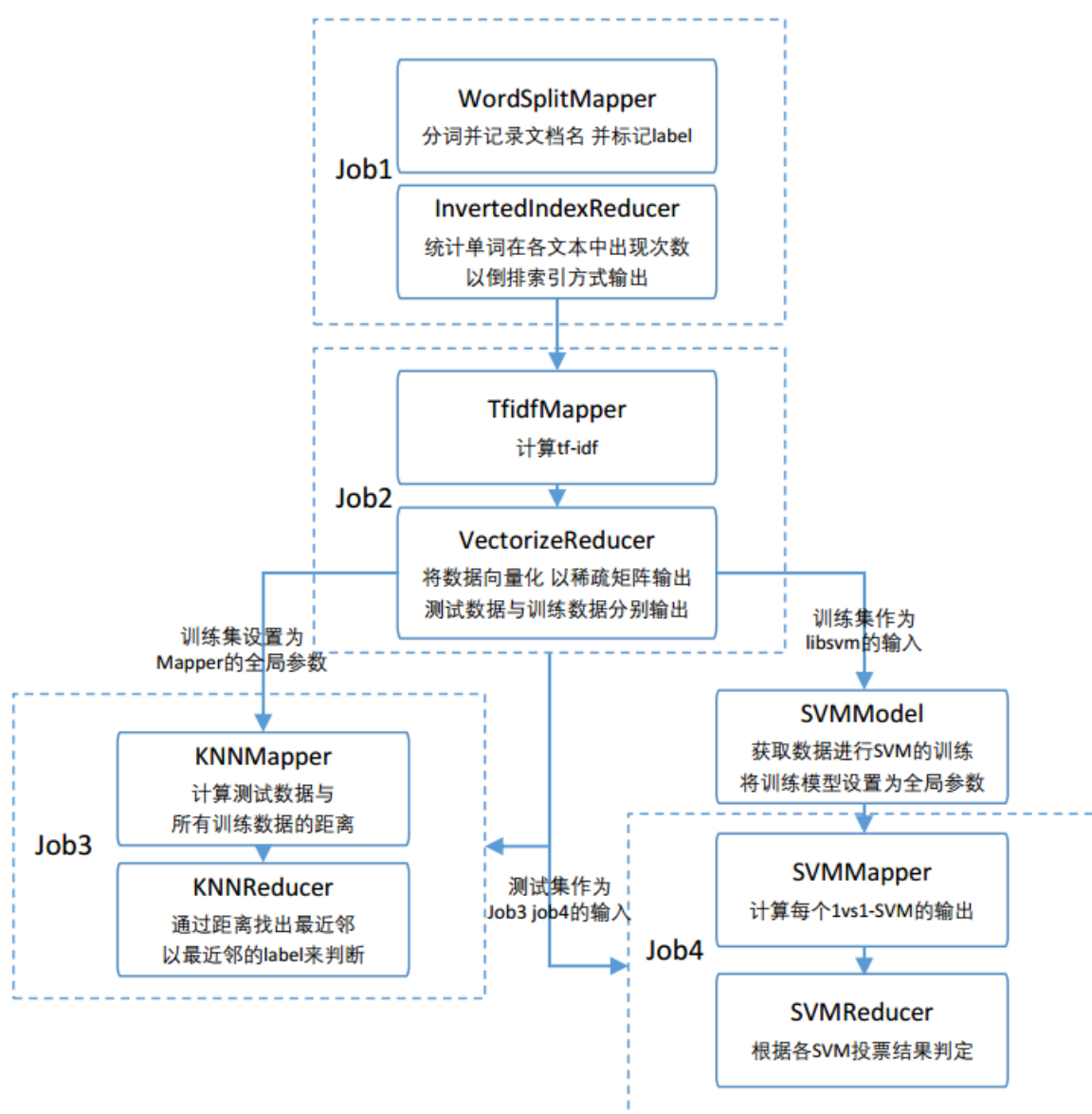


图 1 程序设计思路和运行流程

程序需要输入的参数 <in> <out>

其中 in 为输入文件夹 输入文件夹的格式如下所示

```
--in
  --train
    --positive
    --negative
    --neutral
  --test
    --positive
    --negative
    --neutral
```

即输入文件夹中包含 train 和 test 两个文件夹，表示训练集和测试集，而训练集和测试集中各有 positive、negative、neutral 三个文件夹标志着文件属性。而 out 为输出文件夹路径，KNN 输出文件夹为 out+ “KNN”，SVM 输出文件夹为 out+ “SVM”，如参数<out>若为 myoutput，则 KNN 的输出文件夹为 myoutputKNN，SVM 的输出文件夹为 myoutputSVM。

二、 算法设计与主要类说明

Job1:

WordSplitMapper

1. 分词
2. 获取文档 label、以及是否为训练集的标记, 与 doc 名合并
此时 filename 是一个三元组: <1. 是否为训练集 2. label 3. name>
3. 以 Word 作为 value, 以三元组 和 1 作为 value, 这里 1 表示出现了 1 次, 为了方便 reducer 计算
4. 对于每个单词输出键值对

<key>	<value>
Word	train/test+label+filename:1

eg: 上涨	trainP001:1
股票	testN056:1

InvertedIndexReducer

1. 对每个单词进行倒排索引，并计数
2. 为单词编号, 将单词替换为编号
3. 输出键值对

<key>	<value>
Word_ID	filename1:times1, filename2:times2, ..., filenameN:timesN

eg: 1	trainP001:4, trainN101:2, testN045:10
34	testN056:1, trainN101:2

Job2 :

TfidfMapper

(以 Job1 的输出作为输入)

1. 以每个单词在各文本中出现的次数值作为 tf
2. 以每个单词在所有文本中出现的情况计算 idf
3. 计算 tfidf
4. 根据 tfidf 过滤掉一些词汇
4. 把倒排索引拆分, 以 filename 作为 key, WordID 和 tfidf 值作为 value

<key> <value>
Filename Word_ID : tfidf

eg: trainP001 1 : 2.3564
testN056 34 : 8.6542

VectorizeReducer

1. 按 filename 进行<ID: tfidf>的整合, 成为稀疏形式的向量
2. 从 filename 中获取 label
3. 从 filename 中获取该文本来源于训练集还是测试集, 对于训练集和测试集的数据分别输出
4. 输出键值对

<key> <value>
Filename label, Word_ID1:tfidf1 Word_ID2:tfidf2 ... Word_IDN:tfidfN

eg: trainP001 1 1:2.3564 78:12.8914 859:7.8221
testN056 2 34:8.6542 456:67.2313

Job3

KNNMapper

1. 获取全全局参数: 训练集向量
2. 计算输入向量与所有训练集向量的距离
3. 以测试样本名作为 value, 到某训练集向量的距离作为 value

<key> <value>
testFilename trainFilename : distance

eg: testN056 trainP001 : 4.236
testN056 trainP002 : 6.282

KNNReducer

1. 对于每个训练集向量找到与其距离最近的向量作为最近邻
2. 从最近邻的 filename 中获得其 label
3. 以最近邻的 label 作为预测结果输出

<key> <value>
testFilename label

eg: testN056 negative
testN057 neutral

Pre_Job4 : SVMModel

1. 从 HDFS 中获取训练集数据
2. 进行 SVM 的训练
3. 解析训练模型的结果，并将结果设置为 SVMMapper 的全局参数

Job4 :

SVMMapper

1. 获取支持向量等参数
2. 由于是多个 1vs1-SVM，所以需要分别计算不同 SVM 的输出
3. 将结果输出

`<key>` `<value>`
`testFilename` `SVMtype : SVMoutput`

eg: testN056 1v2 : 1.2398
testN056 2v3 : -0.2354

SVM Reducer

1. 对于各测试样本，统计各 SVM 的判断结果
2. 投票判断，获取最终判断结果
3. 输出结果

`<key>` `<value>`
`testFilename` `label`

eg: testN056 negative
testN057 neutral

三、 程序性能

程序在小规模数据集（30-50 个样本）上运行速度较快，约 1 分钟左右，而且预测结果具有一定精度。

在完整数据集上运行约 10 分钟时间。

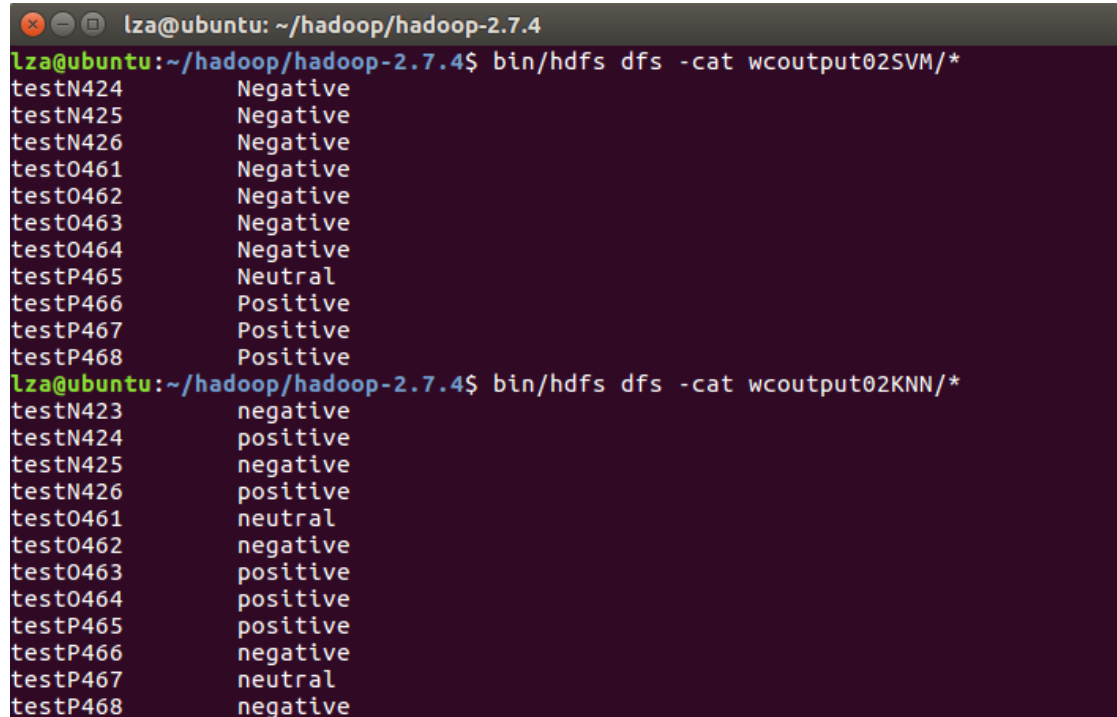
四、 待改进之处

目前对于输入格式是固定的，而且文件夹 label 名称、数量不能发生变化，即目前只能做到处理三分类问题。未来可以改进输入格式，增加输入参数来控制类别数量。

目前，SVM 算法使用的是线性核函数，未来可以额外加入参数，控制核函数种类以及其他 SVM 的超参数。

SVM 训练过程调用了 libsvm 库函数，没有实现并行化。

五、 运行结果截图



```
lza@ubuntu: ~/hadoop/hadoop-2.7.4
lza@ubuntu:~/hadoop/hadoop-2.7.4$ bin/hdfs dfs -cat wcoutput02SVM/*
testN424      Negative
testN425      Negative
testN426      Negative
testO461      Negative
testO462      Negative
testO463      Negative
testO464      Negative
testP465      Neutral
testP466      Positive
testP467      Positive
testP468      Positive
lza@ubuntu:~/hadoop/hadoop-2.7.4$ bin/hdfs dfs -cat wcoutput02KNN/*
testN423      negative
testN424      positive
testN425      negative
testN426      positive
testO461      neutral
testO462      negative
testO463      positive
testO464      positive
testP465      positive
testP466      negative
testP467      neutral
testP468      negative
```

六、 其他说明

已经经过转码的完整数据集我也一并上传到 GitHub 中，并且已经将测试集和训练集分开，可以直接作为我的程序的输入文件夹。