

课程设计报告

小组信息

组名: nju_st38

组员:

吴启龙 181830196 181830196@smail.nju.edu.cn (组长)

潘勉之 181240045 181240045@smail.nju.edu.cn

戎奕名 181240047 181240047@smail.nju.edu.cn

研究领域: 人工智能

课程设计题目

新闻自动分类

课题小组分工

潘勉之: 文本特征提取及表示 任务1-2

戎奕名: 文本分类, 样本预测 任务3-4

吴启龙: Spark实现与性能比较 附加题

摘要

我们基于MapReduce实现了对于大数据新闻文本的TF-IDF特征提取及表示, 并基于此完成了文本分类的任务(linear SVM算法)。在此基础上, 我们用PySpark完成了同样的任务, 在PySpark上还尝试了Naïve Bayes和Logistic Regression两种算法, 并进行了PySpark和MapReduce的性能比较。我

们完整的完成了文本分类预测任务并达到了令人满意的准确率和运行速度，也验证了PySpark相对于MapReduce便于使用和速度更快的优势。

研究问题背景

文本分类是自然语言处理的经典任务，对于推荐系统等下游应用至关重要。研究发现大数据隐含着更为准确的事实。而随着数据量的增长，单机很多情况下已经难以完成任务，因此需要我们利用大规模数据并行处理，更好地利用大数据训练出更好的模型。

主要技术难点和拟解决的问题

难点：数据量大，单机运行慢或根本无法运行，因此需要使用Spark并行处理。

拟解决的问题：用Spark实现新闻分类

主要解决方法 and 设计思路

并行读取文本文件并提取TF-IDF特征，基于提取出的特征选择不同分类算法进行分类。由于PySpark提供更高程度的抽象，按照PySpark的规范调用就可以实现并行，而不需要关心底层细节。

注：以下内容PySpark部分，文本特征选择&表示部分见11-14页，SVM多分类器及预测见15-19页。

详细设计说明

feature.py

完成特征提取(feature_original.py不进行词语的过滤，此外与feature.py相同)

首先需要初始化spark会话，指定好主节点和数据的路径。

调用load函数，该函数实现读取文本文件和进行分词，输出中间结果，并返回PySpark DataFrame.最后调用tfidf函数计算并输出提取出的TFIDF特征。

1.load 函数 (feature.py)

(1) 读取文件

利用每个sc.wholeTextFiles一次性读取某个类别的所有文本文件到一个DataFrame中，逐个添加类别标签并使用union方法合并，实现了将所有数据都读取到一个DataFrame

(2) 分词

利用map调用jieba包实现分词，通过DataFrame的RDD自动实现并行化。之后需要将RDD再转化回DataFrame。

(3) 去除停用词

调用StopWordsRemover实现，输入自定义词表。

(4) 构建关键词词典

规则：中文词，在训练集出现次数大于10，长度大于1。

具体实现上，先统计出每个单词的出现次数，之后筛选出词频大于10的词语，最后选出长度大于1的中文词构成词典

(5) 单词过滤

仅保留词典中出现的词。用pyspark.sql.functions.udf包装_filter_words函数实现单词过滤。此处用python词典实现，基于哈希表，查找时间复杂度低。

2.tfidf 函数(feature.py)

(1) Tokenize

调用HashingTF实现，转化词语列表为稀疏矩阵，存储词的出现次数

(2) 归一化

调用Normalizer实现，将提取的TF值除以文档的总次数（可选，构成两种TFIDF的算法，即TF分别表示总出现次数和在全文的出现的比例）

(3) TFIDF计算

调用IDF，输入之前的TF即可实现

输出结果到文件

models.py

完成分类预测。

初始化设置好后，读取之前的TFIDF，调用相应的算法的包

（LinearSVC，LogisticRegression，NaiveBayes，LinearSVC需要结合OneVsRest实现多分类）即可训练分类模型。正确率统计用MulticlassClassificationEvaluator。

输入输出文件数据和详细输入输出数据格式

feature.py输入为给定的数据集，输出train（训练集的TFIDF），test（测试集的TFIDF），train_normed（归一化的训练集TFIDF），test_normed（归一化的测试集的TFIDF），train_dataset（训练集分词后的中间结果），train_filtered（训练集分词并过滤后的中间结果），test_dataset（测试集分词后的中间结果），test_filtered（测试集分词并过滤后的中间结果）。均使用PySpark DataFrame的.write.save输出。

程序运行实验结果说明和分析

特征提取

利用Spark实现的特征提取速度是MapReduce的46倍多，而且相当一部分时间是在读取数据；另外，对单词进行过滤仅增加了120s的运行时间。这两点充分说明了Spark内存计算的优势，达到远高于MapReduce的性能。

特征提取的结果保存在‘编程操作空间’根目录，feature.txt记录过滤单词特征提取需要的时间，为767s，feature_original.txt记录不过滤单词时特征提取需要的时间，为647s。与MapReduce的对边见下表。

	MapReduce (s)	Spark (s)	ratio
tokenize - train	27981	767	
tokenize - test	7137		
IDF	18		
BOW - train	224		
BOW - test	101		
sum	35461	767	46.233377

分类

模型预测结果（准确率和运行时间）分两个文件夹保存在‘编程操作空间’下，original下保存不过滤单词的结果，filter下保存过滤单词的结果。带有‘normed’表示计算TFIDF时进行了归一化。

	original	time (s)	normed	time (s)	filtered	time (s)	filtered+normed	time (s)
Logistic Regression	88.8	1618	85.5	1408	90.6	1373	87.8	767

Linear SVM	93	5702	85.5	1419	92.7	6470	91.5	8626
Naive Bayes	90.7	96	89.2	114	90.8	94	90.1	95

Naive Bayes最快，其次是Logistic Regression， Linear SVM最慢。

Naive Bayes算法记录的运行时间中，初始化和资源分配等占了相当的比例（overhead），算法本身运行时间更短。

Logistic Regression比Naive Bayes慢一个数量级，准确率也不如Naive Bayes。

Linear SVM 的实现具有较好的扩展性，较快的完成了分类任务，准确率也是最高的（平均不到2小时）。

归一化后效果会变差。

筛选词语整会使不进行归一化时Linear SVM准确率下降，Logistic Regression准确率上升，Naive Bayes基本不变；进行归一化时能使准确率有所提升。

Linear SVM虽然准确率最高，但是运行时间和稳定性都不如Naive Bayes，可能与收敛较难有关。更深层的原因可能是矩阵过于稀疏，优化容易出现异常状况，例如0梯度，导致收敛困难，训练不稳定。其他基于贝叶斯的方法可能会取得更好的效果。

总结

PySpark实现文本分类在实际操作中的主要难点在于本地环境不易搭建（调试用）和不适应PySpark的API的使用。当前的程序功能相对单一，但是扩展性较好。用其他特征提取方式只需要调用其他的特征计算的包，而且往

往有相似的API便于改写；调用其他分类算法也是如此。性能方面已经相对令人满意。

不足和改进之处：可以组合更多的特征提取方法和分类算法或进行更好的特征处理提升性能。

集群运行方式

```
spark-submit --py-files jieba.zip feature.py
spark-submit --py-files jieba.zip feature_original.py
spark-submit models.py LR LR.txt train test
spark-submit models.py SVM SVM.txt train test
spark-submit models.py LR LR_normed.txt train_normed test_normed
spark-submit models.py LR SVM_normed.txt train_normed test_normed
```

输出路径不能提前存在（train, test, train_normed, test_normed, train_dataset, train_filtered, test_dataset, test_filtered）


参考文献

[1] LIBSVM -- A Library for Support Vector Machines. (2021). Retrieved 04 July 2021, from <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

[2] Java Machine Learning Library 0.1.7. (2021). Retrieved 05 July 2021, from <http://java-ml.sourceforge.net/api/0.1.7/>

[3] Classification and regression - Spark 3.1.2 Documentation. (2021). Retrieved 7 July 2021, from <https://spark.apache.org/docs/latest/ml-classification-regression.html>

Spark 运行截图



南京理工大学

南京理工大学计算机学院 - 数据科学

大数据处理课程在线实验平台

北京

时间: 16:30:26 | nju_xst38 | 退出

在线编程空间

编程工作空间

在线提交Jar包

并行计算平台

MapReduce

并行计算

Spark

并行计算

文件存储系统

HDFS分布式存储

Alluxio内存存储

查询管理系统

科教实验列表

个人中心管理

spark

2.4.0

Spark Master at spark://192.168.1.1:7077

URL: spark://192.168.1.1:7077

Alive Workers: 17

Cores in use: 272 Total, 0 Used

Memory in use: 1018.2 GB Total, 0.0 B Used

Applications: 0 Running, 27 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (17)

Worker id	Address	State	Cores	Memory
worker-20210706160725-192.168.1.14-40389	192.168.1.14-40389	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161050-192.168.1.10-39405	192.168.1.10-39405	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161408-192.168.1.16-40181	192.168.1.16-40181	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161437-192.168.1.19-33158	192.168.1.19-33158	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161510-192.168.1.17-35541	192.168.1.17-35541	ALIVE	16 (0 Used)	46.0 GB (0.0 B Used)
worker-20210706161648-192.168.1.15-34186	192.168.1.15-34186	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161702-192.168.1.13-39055	192.168.1.13-39055	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161709-192.168.1.11-35302	192.168.1.11-35302	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161748-192.168.1.20-42135	192.168.1.20-42135	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706161749-192.168.1.18-44158	192.168.1.18-44158	ALIVE	16 (0 Used)	46.0 GB (0.0 B Used)
worker-20210706161836-192.168.1.9-34484	192.168.1.9-34484	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162048-192.168.1.3-39732	192.168.1.3-39732	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162109-192.168.1.7-35517	192.168.1.7-35517	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162120-192.168.1.2-39445	192.168.1.2-39445	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162244-192.168.1.1-46148	192.168.1.1-46148	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162250-192.168.1.5-36613	192.168.1.5-36613	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)
worker-20210706162327-192.168.1.6-45170	192.168.1.6-45170	ALIVE	16 (0 Used)	61.7 GB (0.0 B Used)

Running Applications (0)

Not secure | 114.212.190.92:8082/#spark

NJU-PASA 大数据平台运维与实验平台

北京时间: 16:31:34 | nju_st38 | 退出

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

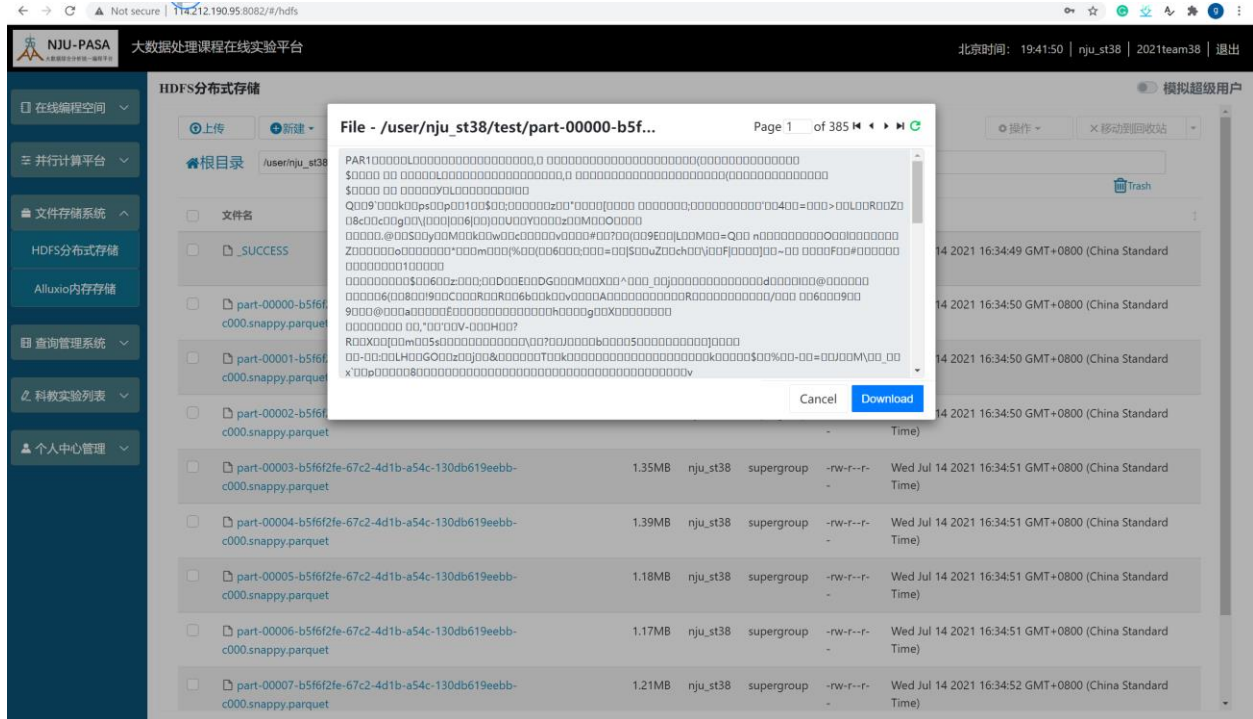
Completed Applications (27)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20210715111559-0026	SVM.txt	20	30.0 GB	2021/07/15 11:15:59	nju_st38	FINISHED	1.1 h
app-20210715105944-0025	SVM.txt	20	30.0 GB	2021/07/15 10:59:44	nju_st38	FINISHED	15 min
app-20210715105815-0024	SVM.txt	20	30.0 GB	2021/07/15 10:58:15	nju_st38	FINISHED	2 s
app-20210715104735-0023	SVM.txt	20	30.0 GB	2021/07/15 10:47:35	nju_st38	FINISHED	57 s
app-20210714163626-0022	NB.txt	20	30.0 GB	2021/07/14 16:36:26	nju_st38	FINISHED	1.3 min
app-20210714163619-0021	NB_normed.txt	20	30.0 GB	2021/07/14 16:36:19	nju_st38	FINISHED	1.3 min
app-20210714161013-0020	NB.txt	20	30.0 GB	2021/07/14 16:10:13	nju_st38	FINISHED	1.3 min
app-20210714160943-0019	NB_normed.txt	20	30.0 GB	2021/07/14 16:09:43	nju_st38	FINISHED	1.5 min
app-20210713160538-0016	SVM.txt	20	30.0 GB	2021/07/13 16:05:38	nju_st38	FINISHED	1.6 h
app-20210713160525-0015	LR.txt	20	30.0 GB	2021/07/13 16:05:25	nju_st38	FINISHED	27 min
app-20210713160604-0018	SVM_normed.txt	20	30.0 GB	2021/07/13 16:06:04	nju_st38	FINISHED	23 min
app-20210713160550-0017	LR_normed.txt	20	30.0 GB	2021/07/13 16:05:50	nju_st38	FINISHED	23 min
app-20210713155223-0014	feature	20	30.0 GB	2021/07/13 15:52:23	nju_st38	FINISHED	10 min
app-20210711190529-0010	SVM_normed.txt	20	30.0 GB	2021/07/11 19:05:29	nju_st38	FINISHED	2.4 h
app-20210711191443-0013	SVM.txt	20	30.0 GB	2021/07/11 19:14:43	nju_st38	FINISHED	1.8 h
app-20210711190645-0012	LR.txt	20	30.0 GB	2021/07/11 19:06:45	nju_st38	FINISHED	23 min
app-20210711190555-0011	LR_normed.txt	20	30.0 GB	2021/07/11 19:05:55	nju_st38	FINISHED	13 min
app-20210711185040-0009	feature	20	30.0 GB	2021/07/11 18:50:40	nju_st38	FINISHED	12 min
app-20210711184410-0008	LR	20	30.0 GB	2021/07/11 18:44:10	nju_st38	FINISHED	1.8 min
app-20210711152155-0005	SVM	20	30.0 GB	2021/07/11 15:21:55	nju_st38	FINISHED	1.8 h
app-20210711152248-0006	LR	20	30.0 GB	2021/07/11 15:22:48	nju_st38	FINISHED	17 min
app-20210711152532-0007	MLP	20	30.0 GB	2021/07/11 15:25:32	nju_st38	FINISHED	4.1 min
app-20210710113128-0004	feature	20	30.0 GB	2021/07/10 11:31:28	nju_st38	FINISHED	10 min
app-20210710111943-0003	feature	20	30.0 GB	2021/07/10 11:19:43	nju_st38	FINISHED	4.2 min

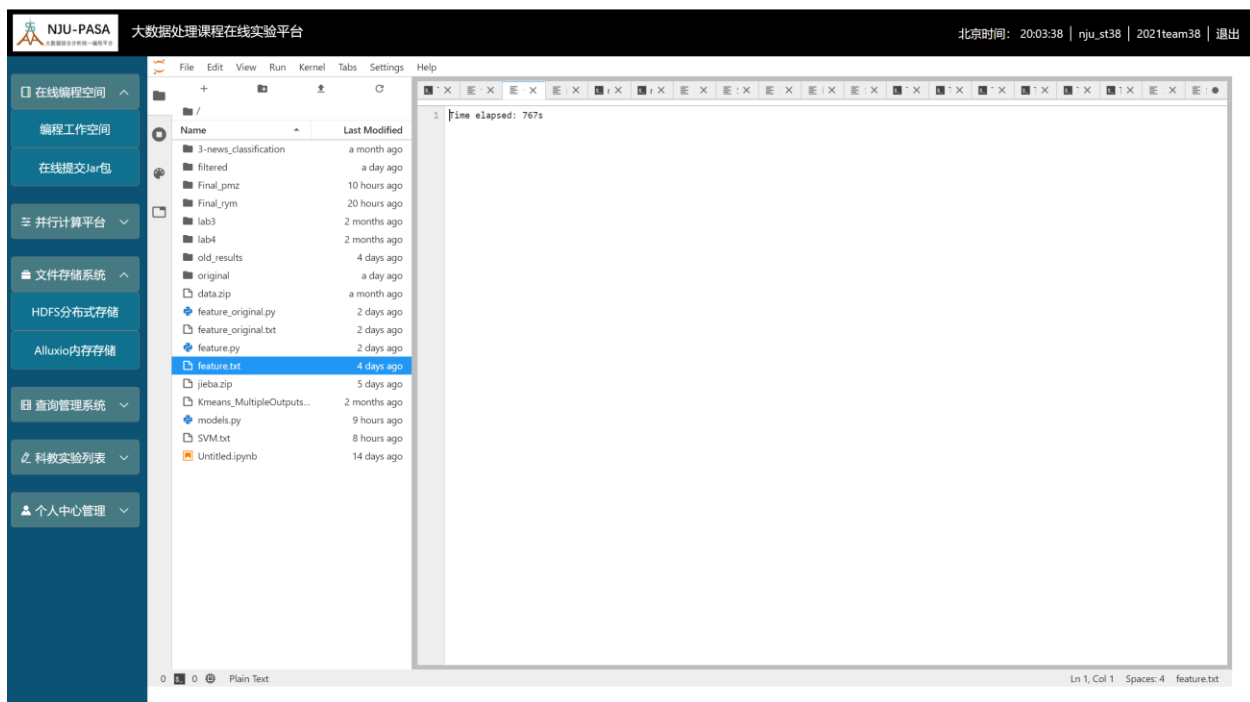
app-20210706162339-0000	LR	20	30.0 GB	2021/07/06 16:23:39	nju_st38	FINISHED	22 min
-------------------------	----	----	---------	---------------------	----------	----------	--------

运行结果

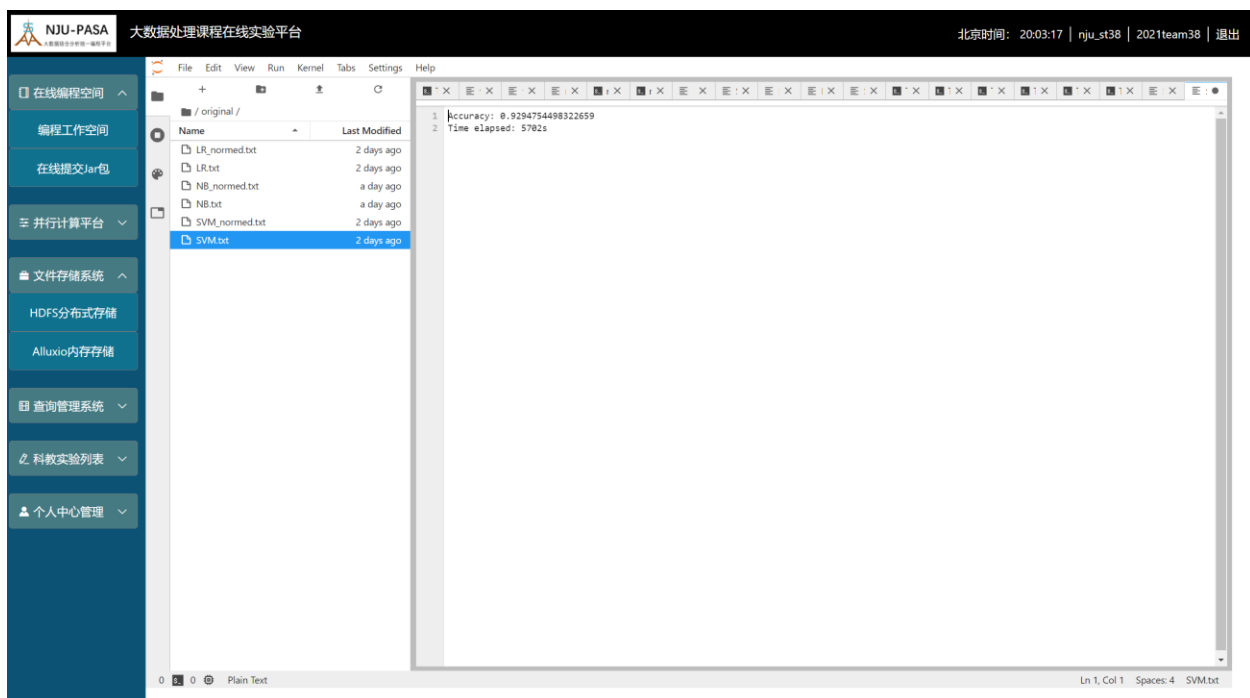
中间结果（train, test, train_normed, test_normed, train_dataset, train_filtered, test_dataset, test_filtered 路径下，由多个部分构成）如下图，保存在 HDFS 上，直接打开为乱码。



特征提取 (以 feature.txt 为例)



预测（以 original/SVM.txt 为例）



文本特征选择 & 表示

关于文本特征的选择与表示，本次实验主要包括三个步骤：分词与特征词筛选、计算特征词的逆文档频率（IDF）以及生成每个文档的TF-IDF表示。

分词与特征词筛选

代码及jar包: `tokenize.java` , `tokenize.jar`

编译指令:

```
$ javac tokenize.java -cp IK-Analyzer-2012FF/dist/IKAnalyzer2012_FF.jar:IK-Analyzer-2012FF/dist/IKAnalyzer2012FF_u1.jar:${hadoop classpath}
$ jar -cvf tokenize.jar tokenize*.class
```

集群运行命令（分别对训练集和测试集分词）：

```
$ hadoop jar /home/nju_st38/Final_pmz/tokenize.jar tokenize
/user/nju_st38/pmzFinal_pkg hdfs://pasa-share/data/2021spring/3-
news_classification/ /user/nju_st38/pmz_trainTokenout
$ hadoop jar /home/nju_st38/Final_pmz/tokenize.jar tokenize
/user/nju_st38/pmzFinal_pkg hdfs://pasa-share/data/2021spring/3-
news_classification/ /user/nju_st38/test_Tokenout
```

运行截图分别为

nju_st38	tokenize	MAPREDUCE	root.2021team38	Thu Jul 15 00:34:05 +0800 2021	Thu Jul 15 08:20:26 +0800 2021	FINISHED	SUCCEEDED
nju_st38	tokenize	MAPREDUCE	root.2021team38	Thu Jul 15 09:40:23 +0800 2021	Thu Jul 15 11:39:30 +0800 2021	FINISHED	SUCCEEDED

注意对训练集和测试即分词均使用的是 `tokenize.jar` 包，但是其对应的源代码略有不同，区别在于 `tokenize.java` 的第183行

```
FileStatus[] status = fs.listStatus(new Path(args[1] + "test")); // or train
```

算法流程：Mapper的输入为一行文本，使用 `IK-Analyzer` 分词包分词，输出键为

`term#SEG#filename`，其中 `#SEG#` 为分隔标识。输出值为 `frequent`，即词语 `term` 在文件 `filename` 的词频。

为确保同一个词语能够发送到同一个Reducer上，实验中定制了Partitioner，临时将词语 `term` 作为新的键。

Reducer接受Mapper的输出，最终Reducer的输出形式为 `[word] total_num`
`file1:num1;file2:num2;...`，其中 `total_num` 为词语 `word` 在所有文档中出现次数总和。

注意在以上过程中，由于词语总数非常多，我们对特征词进行了筛选，主要包括如下方法：

- 1、删除停用词，这里将实验提供的停用词表上传到Distributed Cache上。
- 2、删除总词频 `total_num` 小于某个阈值的词，实验中阈值设置为10。
- 3、删除所有词语中包含的数字以及特殊字符。
- 4、删除长度为 1 的词语。
- 5、仅仅保留中文词

最终输出结果示例如下:

```
[兼容机] 5, 娱乐_136669.txt: 4;游戏_407377.txt: 1
[兼容问题] 5, 体育_101363.txt: 1;体育_101528.txt: 1;彩票_261184.txt: 2;游戏_407352.txt: 1
[兼差] 5, 教育_287016.txt: 1;教育_290031.txt: 1;星座_405614.txt: 1;星座_405628.txt: 1;星座_405680.txt: 1
```

计算特征词的IDF

代码及jar包: `IDF.java` , `IDF.java`

编译指令:

```
$ javac IDF.java -cp $(hadoop classpath)
$ jar -cvf IDF.jar IDF*.class
```

集群运行命令（仅仅利用训练集计算IDF）：

```
$ hadoop jar /home/nju_st38/Final_pmz/IDF.jar IDF hdfs://pasa-
share/data/2021spring/3-news_classification/ /user/nju_st38/pmz_Tokenout/part-r-
00000 /user/nju_st38/small_IDF_cn
```

运行截图为

nju_st38	IDF	MAPREDUCE	root.2021team38	Wed Jul 14 23:33:05 +0800 2021	Wed Jul 14 23:33:23 +0800 2021	FINISHED	SUCCEEDED
----------	-----	-----------	-----------------	--------------------------------	--------------------------------	----------	-----------

算法流程： Mapper的输入为前一阶段生成的特征词文件，计算词语t的IDF的公式为

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}| + 1}$$

其中N为训练集中所有文档总数，通过如下代码实现:

```
int filesnum = 0;
FileSystem fs = FileSystem.get(conf);
FileStatus[] status = fs.listStatus(new Path(args[0] + "train"));
for(FileStatus f: status) {
    if(f.isDir()) {
        FileStatus[] son_status = fs.listStatus(f.getPath());
        filesnum += son_status.length;
        System.out.println(filesnum);
    }
}
conf.setInt("filesnum", filesnum);
```

分母的计算只需要统计一下每一条词的输入记录即可。

Reducer不需要额外做其他事，只要将Mapper的键值原样输出即可，最终输出格式如下:

[一个箭步] 8.193772
[一个萝卜一个坑] 9.769308
[一个词] 6.4734716
[一个都不能少] 7.732426

生成每个文档的TF-IDF表示

代码及jar包: BOW.java, BOW.jar

编译指令:

```
$ javac BOW.java -cp $(hadoop classpath)
$ jar -cvf BOW.jar BOW*.class
```

集群运行命令（两条分别是训练集和测试集）:

```
$ hadoop jar /home/nju_st38/Final_pmz/BOW.jar BOW
/user/nju_st38/small_IDF_cn/part-r-00000 /user/nju_st38/pmz_Tokenout/part-r-
00000 /user/nju_st38/small_BOW_cn
$ hadoop jar /home/nju_st38/Final_pmz/BOW.jar BOW
/user/nju_st38/small_IDF_cn/part-r-00000 /user/nju_st38/test_Tokenout/part-r-
00000 /user/nju_st38/test_BOW_cn
```

运行截图分别为

nju_st38	BOW	MAPREDUCE	root.2021team38	Wed Jul 14 23:38:22 +0800 2021	Wed Jul 14 23:42:06 +0800 2021	FINISHED	SUCCEEDED
nju_st38	BOW	MAPREDUCE	root.2021team38	Wed Jul 14 23:44:48 +0800 2021	Wed Jul 14 23:46:29 +0800 2021	FINISHED	SUCCEEDED

算法流程: 首先将第二阶段生成的IDF文件上传到Distributed Cache中, 以 `HashMap` 形式保存, 同时为每个词语赋予一个编号 (0, 1, 2, ...)。

Mapper的输入为第一阶段生成的文件, 输入的键为 `offset`, 输入值的格式即为 `[word] total_num file1:num1;file2:num2;...`。输出键为 `file`, 输出值为 `word##SEG##num`, 其中 `##SEG##` 为分隔符。

Reducer的输入键为 `file`, 输入值为 `[word1##SEG##num1, word2##SEG##num2, ...]`。词语 `wordi` 的词频即为 `numi` (此处为未归一化的TF, 归一化的为 $\frac{num_i}{\sum_j num_j}$) , 再从Distributed Cache中获得其IDF, 计算得出TF-IDF后将向量对应编号的值设置为TF-IDF。由于向量维度较大, 最终输出文件中未保存值为0的位置, 最终格式为 `file, idx1:TF-IDF1 idx2:TF-IDF2 ...`, 样例如下

体育_0.txt 12775:3.2613 8859:2.9546 27583:4.8360 70817:1.3732 29042:3.1281 912:3.9753 244:3.5766 4628:6.3737 74997:4.5214
14360:5.8030 8690:4.3048 49562:2.4419 8822:3.2740 6017:5.4716 3694:4.7280 74669:5.0029 47742:3.0207 79129:13.5018
32081:3.2888 32460:1.2368 22588:7.0724 5316:3.6184 73529:5.3525 13681:4.4512 60107:4.7400 39726:4.7687 23271:13.1549
63596:6.7328 44502:7.8602 75174:5.9051 27531:7.4339 31309:6.9761 36320:3.2389 723:2.0773 33537:5.5449 33431:2.4066
80564:8.0784 50243:2.3724 75137:2.0690 9771:3.2824 7575:2.2497 1731:6.2226 3030:5.8877 13996:7.3714 79333:3.0335
13581:2.9901 15328:6.8337 44809:4.0162 1993:3.2690 31393:3.3073 9836:2.9989 50606:8.6707 51211:40.3236 21000:4.4096
5664:3.3493 72009:3.6822 41375:5.9795 37448:4.0628 81875:5.2137 76372:2.6573 45513:2.0280 23113:4.2935 80244:19.8583
75749:3.7354 11609:8.7423 8767:4.6745 28374:9.0762 69459:1.2539 18970:6.2984 18364:2.6898 5307:2.1651 35101:6.5107
26669:33.5452 77154:3.9587 15004:7.7999 70738:7.0502 57957:5.3150 37056:1.9382 27817:3.7985 35287:2.0286 30439:2.5121
13992:1.7503 37493:23.0479 80611:8.0646 12736:6.2984 80231:13.8527 61614:3.4740 2728:2.3893 5911:6.2984 2331:3.4275
55518:2.5212 45493:3.1083 72999:4.4545 70040:4.7542 27720:7.9235 21708:6.4491 39753:8.8926 70737:45.9760 64738:9.2585
19320:3.2104 75280:1.9471 39829:2.9316 25312:8.7279 44657:5.2825 42590:3.5454 48658:9.5330 39664:2.0465 5276:3.7930
77534:2.1405 15901:3.4503 19760:7.5180 9421:5.1912 54817:13.2968 78661:18.1188 45475:3.5990 73468:7.5180 2013:11.0234
57885:4.8992 23272:66.7375 10489:6.1951 71158:7.1543 79122:12.5972 13310:2.4081 15033:3.8617 36800:3.2888 35335:3.9090

戎奕名 svm多分类器及预测

svm多分类器的搭建、测试集的预测以及正确率的计算

SVM多分类器训练

源代码

SVMtrainer.java

运行方式

```
javac SVMtrainer.java -cp $(hadoop classpath):/usr/java/javaml-0.1.7/javaml-0.1.7.jar
jar -cvf SVMtrainer.jar SVMtrainer*.class
hadoop jar SVMtrainer.jar SVMtrainer /user/nju_st38/no_divided_test_B0W
/user/nju_st38/rym_library /user/nju_st38/nju_st38/no_divided_B0W
/user/nju_st38/svm_train
```

参数说明： `hadoop jar SVMtrainer.jar SVMtrainer test_file_path dependency_library_path train_file_path output_path`

依赖库

Java Machine Learning Library 0.1.7 LibSVM

设计思路

参考课上所讲的短文本SVM多分类器MapReduce设计，并行地训练14个SVM二分类器，每条测试样本对每个类别都会产生true/false的label。

Map阶段

input key: 默认偏移量(LongWritable), input value: filename idx1:val1 idx2:val2... (Text) output key: svm number(IntWritable), output value: filename idx1:val1 idx2:val2...(Text)

Map阶段，将每条文本向量对14个类别打上positive/negative的标签，发往14个Reducer训练二分类器。

Reduce阶段

output key: NullWritable, output value: filename|svm number|positive/negative|score

Reduce阶段，将svm number一致的训练样本放在一个Reducer上训练二分类器。同时，将测试集放在DistributedCache上分发到每个Reducer进行预测，输出分数。

遇到的问题及解决方法

数据维度过大，集群报错"GC Overhead limit exceeded"

利用conf.set()设置参数，增加节点分配内存。

```
conf.set("mapred.child.java.opts", "-Xmx8192m");
conf.set("mapred.reduce.memory.mb", "8192");
conf.set("mapred.reduce.child.java.opts", "-Xmx8192m");
```

尽量对循环中的变量进行复用。

```
String line = "";
String[] buffer = {" "};
int label = 0;
String vector_str = "";

while(it.hasNext()) {
    ...
}
```

数据维度及数据量过大，无法运行，集群报错"Time out for 600 sec"

尝试使用Hash trick降维，800维时可运行所有训练集并测试全部测试集。

若不降维，迫不得已对原训练集和测试集进行均匀采样，降低数据规模。

二分类器输出score数值相似，难以比较

计算TF-IDF时不除在句子中的总词数，放大score区分度。

测试结果预测

源代码

SVMpredictor.java

运行方式

```
javac SVMpredictor.java -cp $(hadoop classpath)
jar -cvf SVMpredictor.jar SVMpredictor*.class
hadoop jar SVMpredictor.jar SVMpredictor /user/nju_st38/svm_train
/user/nju_st38/svm_predict
```

参数说明： hadoop jar SVMpredictor.jar SVMpredictor input_path output_path

设计思路

Map阶段

input key: 默认偏移量(LongWritable), input value: filename|svm number|positive/negative|score output key: filename(Text), output value: svm number|positive/negative|score

Map阶段将同一个测试样本的所有二分类分数发到一个Reducer。

Reduce阶段

output key: filename(Text), output value: predict label(Text)

Reduce阶段比较一个测试样本二分类分数，取最大的作为测试类别

输出准确率

源代码

SVMpredictor.java

运行方式

```
javac SVMpredictor.java -cp $(hadoop classpath)
jar -cvf SVMpredictor.jar SVMpredictor*.class
hadoop jar SVMpredictor.jar SVMpredictor /user/nju_st38/svm_train
/user/nju_st38/svm_predict
```

参数说明： hadoop jar SVMpredictor.jar SVMpredictor input_path output_path

设计思路

Map阶段

input key: 默认偏移量(LongWritable), input value: predict label(Text) output key: 0(IntWritable) 发到一个Reducer， output value: 0/1(IntWritable)

Map阶段，比较文件真实类别和预测类别，输出是正是误(0/1)。

Reduce阶段

output key: Accuracy(DoubleWritable), output value: NullWritable

Reduce阶段，统计正确个数和总个数，计算准确率。

实验结果

不除句子中总词数，使用Hash降维，使用全体数据集和全体测试集，准确率8.53%

不除句子中总词数，不降维，使用2187训练集，1312测试集，准确率15.77%

不除句子总词数，不降维，使用1050训练集，263测试集，准确率12.9%（未增大集群资源时）

准确率最高情况（15.77%）集群运行截图：

交互式统一大数据编程计算平台

markdown加粗_百度搜索

Markdown文字加粗与斜体

114.212.190.95:8082/#/yarn

Update

NJU-PASA

大数据处理课程在线实验平台

北京时间：12:10:30 | nju_st38 | 退出

在线编程空间

并行计算平台

MapReduce并行计算

Spark并行计算

文件存储系统

查询管理系统

科教实验列表

个人中心管理

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Application application_1626070675586_0183

Logged in as: dr:who

Kill Application

Application Overview

User: nju_st38

Name: SVMTrainer

Application Type: MAPREDUCE

Application Tags:

YarnApplicationState: FINISHED

Queue: root.2021team38

FinalStatus Reported by AM: SUCCEEDED

Started: Mon Jul 12 18:43:31 +0800 2021

Elapsed: 9mins, 41sec

Tracking URL: History

Diagnostics:

Application Metrics

Total Resource Preempted: <memory:0, vCores:0>

Total Number of Non-AM Containers Preempted: 0

Total Number of AM Containers Preempted: 0

Resource Preempted from Current Attempt: <memory:0, vCores:0>

Number of Non-AM Containers Preempted from Current Attempt: 0

Aggregate Resource Allocation: 228446688 MB-seconds, 7508 vcore-seconds

Show 20 entries

Attempt ID

Started

Node

Logs

Logs

Blacklisted Nodes

agpattempt_1626070675586_0183_000001

Mon Jul 12 18:43:31 +0800 2021

http://slave013.8042

Logs

N/A

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

交互式统一大数据编程计算平台

markdown加粗_百度搜索

Markdown文字加粗与斜体

114.212.190.95:8082/#/yarn

Update

NJU-PASA

大数据处理课程在线实验平台

北京时间：12:11:40 | nju_st38 | 退出

在线编程空间

并行计算平台

MapReduce并行计算

Spark并行计算

文件存储系统

查询管理系统

科教实验列表

个人中心管理

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Application application_1626070675586_0185

Logged in as: dr:who

Kill Application

Application Overview

User: nju_st38

Name: SVMpredictor

Application Type: MAPREDUCE

Application Tags:

YarnApplicationState: FINISHED

Queue: root.2021team38

FinalStatus Reported by AM: SUCCEEDED

Started: Mon Jul 12 18:55:18 +0800 2021

Elapsed: 15sec

Tracking URL: History

Diagnostics:

Application Metrics

Total Resource Preempted: <memory:0, vCores:0>

Total Number of Non-AM Containers Preempted: 0

Total Number of AM Containers Preempted: 0

Resource Preempted from Current Attempt: <memory:0, vCores:0>

Number of Non-AM Containers Preempted from Current Attempt: 0

Aggregate Resource Allocation: 3646156 MB-seconds, 112 vcore-seconds

Show 20 entries

Attempt ID

Started

Node

Logs

Logs

Blacklisted Nodes

agpattempt_1626070675586_0185_000001

Mon Jul 12 18:55:18 +0800 2021

http://slave011.8042

Logs

N/A

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

交互式统一大数据编程计算平台 | markdown加粗_百度搜索 | Markdown文字加粗与斜体 | 114.212.190.95:8082/#/yarn

NJU-PASA 大数据处理课程在线实验平台 北京时间: 12:11:58 | nju_st38 | 退出

在线编程空间
并行计算平台
MapReduce 并行计算
Spark 并行计算
文件存储系统
查询管理系统
科教实验列表
个人中心管理

hadoop

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Application application_1626070675586_0186

Logged in as: drwho

Kill Application

Application Overview

User:	nju_st38
Name:	Accuracy
Application Type:	MAPREDUCE
Application Tags:	
YarnApplicationState:	FINISHED
Queue:	root.2021team38
FinalStatus Reported by AM:	SUCCEEDED
Started:	Mon Jul 12 18:57:14 +0800 2021
Elapsed:	17sec
Tracking URL:	History
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	322518 MB-seconds, 31 vcore-seconds

Show 20 entries

Attempt ID	Started	Node	Logs	Blacklisted Nodes
appattempt_1626070675586_0186_000001	Mon Jul 12 18:57:14 +0800 2021	http://slave014.8042	Logs	N/A

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

总结

准确率普遍不高，但高于随机猜测，实现应该没有太大问题，可能有一下几点原因。

- Hash降维损失信息过多，即使使用全部训练集结果也不佳，降维方法不可取。
- 由于集群性能限制，不降维时最佳情况时使用2187训练集预测1312测试集。
- 二分类器准确率实际上很高，可能发生了过拟合，但Java ML libsvm没有相应参数解决过拟合的问题。
- Java ml libsvm输出的分数原理不明，不同二分器之间的分数可能存在偏差。