lab1

计55 许翰翔 2015011370

Write-up

How long do you think this project would take you to finish?

5h

How much time did you actually spend on this project?

很惭愧…大概花了20h… 装环境+熟悉环境大约花了一半以上的时间… 实际代码时间在2h内，代码量并不是很大。 剩下的时间基本上花在处理编译运行、从服务器上传/下载、从hdfs上传/下载。

Acknowledge any assistance you received from anyone except assigned course readings and the course staff.

全部独立完成，其中hadoop的代码参考了04 Hadoop实现原理C - 分布式编程环境MapReduce.pdf中的代码。

What test corpus did you load into HDFS? Where is it located (i.e. HDFS path)?

在hdfs的/user/2015011370/lab1/freword.txt中，储存了part1要求过滤掉的”noise”。

What, if any, “noisy words” did you find as a result of your WordCount? By what criteria did you determine that they were “noisy”? Were you surprised by any of these words?

简单的使用了单词的总出现次数>100即判定为”noise”，确实会过滤掉一些不应过滤的单词，但是大体效果还是很好的。以下列举了其中的一部分：

```
oath
octavius
of
off
offence
offer
office
officer
oft
often
old
olivia
on
once
one
only
open
or
order
```

可以发现大多数确实是在任何地方都算得上高频的单词。

Which extensions did you do? Please also detail any interesting results you found.

前4个扩展均做了（最后2个扩展不用做？）。

Part1： 先用WordCount.java得到单词频率，之后用checkword.py，将高频单词处理出来。

Part2： 使用InvertedIndex.java计算倒排列表，详情见之后的部分。

扩展1： 在WordCount.java部分，map的时候使用Java的正则表达式匹配单词：

```
    public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable>{
        private Text outputKey = new Text();
        private IntWritable outputVal = new IntWritable(1);
        private static final String REGEX = "\\w+";

        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {

            String line = value.toString();
            Pattern p = Pattern.compile(REGEX);
            Matcher m = p.matcher(line);
            while (m.find())
            {
                String word = m.group().toLowerCase();
                outputKey.set(word);
                context.write(outputKey, outputVal);
            }
        }
    }
```

扩展3： 在InvertedIndex.java部分，首先用setup过滤掉”noise”，之后在map中将传进来的key（即文件中seek的偏移量，单位是字节）存入write的value中，这样就可以得到在文件中查询到的偏移量。

```
    public static class InvertedIndexMapper extends Mapper<Object, Text, Text,
Text>{

        private static final String stopWordPath = "lab1/freword.txt";

        private Text outputKey = new Text();
        private Text outputVal = new Text();
        private static final String REGEX = "\\w+";
        private Set<String> stopWords = new HashSet<String>();
        private FileSystem fs;

        protected void setup(Context context) throws IOException,
InterruptedException {
            Configuration conf = new Configuration();
            fs = FileSystem.get(conf);
            FSDataInputStream dis = fs.open(new Path(stopWordPath));
            while (true) {
                String line = dis.readLine();
                if (line == null || line.isEmpty()) break;
                StringTokenizer tokenizer = new StringTokenizer(line);
                while (tokenizer.hasMoreTokens())
                    stopWords.add(tokenizer.nextToken());
            }
        }

        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
            FileSplit fileSplit = (FileSplit) context.getInputSplit();
            String fileName = fileSplit.getPath().getName();
            String line = value.toString();
            Pattern p = Pattern.compile(REGEX);
            Matcher m = p.matcher(line);
            while (m.find())
            {
                String word = m.group().toLowerCase();
                if (stopWords.contains(word)) continue;
                outputKey.set(word);
                outputVal.set(fileName + ":" + key.toString());
                context.write(outputKey, outputVal);
            }
        }
    }
```

扩展2与扩展4： 使用了python处理询问，在main.py中，Dict存了”文件名”“偏移量”->单词出现的那一行，InvertedIndex则存了倒排列表中是否存在这个单词。 询问时死循环输入一个单词，返回文档中查询的结果，并用**标注单词：

```
import os
import re

Input = open("InvertedIndex.txt", "r")

dataPath = ".\\shakespeare"
Dict = {}

for root, Dir, files in os.walk(dataPath) :
    for filename in files:
        now = open(dataPath + "\\" + filename, "r")
        Dict[filename] = {}
        cnt = 0
        for line in now:
            Dict[filename][cnt] = line.strip()
            cnt += len(line)

InvertedIndex = {}

for line in Input:
    now = line.strip("\n").split("\t")
    InvertedIndex[now[0]] = now[1].split(", ")

while 1:
    print("Please input your query string.")
    query = input()
    check = 0
    if InvertedIndex.get(query):
        for kv in InvertedIndex[query]:
            check = 1
            kv = kv.split(":")
            line = Dict[kv[0]][int(kv[1])]
            print("in", kv[1], "of", kv[0], ":\n   ", re.sub(query,
"*"+query+"*", line, flags=re.I))
    if check == 0:
            print("No such word or too normal.")
```

以下是几个例子:

```
Please input your query string.
7
in 104030 of kinghenryviii :
    *7*. SUFFOLK, in his robe of estate, his coronet
Please input your query string.
8
in 104226 of kinghenryviii :
    *8*. A canopy borne by four of the Cinque-ports;
Please input your query string.
9
in 104417 of kinghenryviii :
    *9*. The old Duchess of Norfolk, in a coronal of
Please input your query string.
10
in 104524 of kinghenryviii :
    *10*. Certain Ladies or Countesses, with plain
Please input your query string.
magic
in 96917 of hamlet :
    Thy natural *magic* and dire property,
in 87856 of antonyandcleopatra :
    The noble ruin of her *magic*, Antony,
in 61081 of macbeth :
    And that distill'd by *magic* sleights
in 140264 of winterstale :
    There's *magic* in thy majesty, which has
in 143651 of winterstale :
    If this be *magic*, let it be an art
in 5303 of tempest :
    And pluck my *magic* garment from me. So:
in 82884 of tempest :
    [Enter PROSPERO in his *magic* robes, and ARIEL]
in 85344 of tempest :
    By my so potent art. But this rough *magic*
in 12869 of othello :
    If she in chains of *magic* were not bound,
in 95798 of othello :
    OTHELLO     'Tis true: there's *magic* in the web of it:
in 19233 of othello :
    What conjuration and what mighty *magic*,
in 3507 of 1kinghenryvi :
    By *magic* verses have contrived his end?
in 1649 of timonofathens :
    *magic* of bounty! all these spirits thy power
Please input your query string.
light
No such word or too normal.
Please input your query string.
```