**Step1: WordCount**

datafile目录中有许多本小说文本，统计出现次数最高的10个单词：

1. 通过多个进程进行统计，限定每个进程最多读取5M的数据；
2. 尝试先将首字母相同的单词写入相同文件，然后对每个文件单独进行统计。（模拟shuffle过程，使得学生更好理解shuffle在map后以及reduce前所起的作用）
3. #!/usr/bin/env python
4. # -\*- coding: utf-8 -\*-
5. '''''
6. hadooptool
7. '''
8. **from** abc **import** ABCMeta, abstractmethod
9. **import** multiprocessing
10. **from** time **import** time
11. **from** filetool **import** getFileByte, listAllFilesUnderDir, makeDir, delAllFilesUnderDir, readLines, readLinesOfFiles,\
12. writeLines
14. TEMP\_DIR = 'temp/'

17. **class** Shuffle(object):
18. \_\_metaclass\_\_ = ABCMeta
19. @abstractmethod
20. **def** partition(self):
21. **pass**
22. @abstractmethod
23. **def** sort(self):
24. **pass**

27. **class** Mapper(object):
28. \_\_metaclass\_\_ = ABCMeta
29. @abstractmethod
30. **def** do(self, lines):
31. **pass**

34. **class** Reducer(object):
35. \_\_metaclass\_\_ = ABCMeta
36. @abstractmethod
37. **def** do(self, lines):
38. **pass**

41. **class** Task(multiprocessing.Process):
42. **def** run(self):
43. **print**('start ' + multiprocessing.current\_process().name)
44. start = time()
45. self.tasker.do( self.lines)
46. end = time()
47. **print**('end %s: %.3f s' % (multiprocessing.current\_process().name, end-start))
48. **def** \_\_init\_\_(self, tasker, lines):
49. multiprocessing.Process.\_\_init\_\_(self)
50. self.tasker = tasker
51. self.lines = lines

54. **class** Job(object):
56. **def** \_\_init\_\_( self, mapper, reducer, mem\_size\_limit, input\_dir, output\_dir ):
57. self.mapper = mapper
58. self.reducer = reducer
59. self.memSizeLimit = mem\_size\_limit
60. self.inputDir = input\_dir
61. self.outputDir = output\_dir
63. **def** runMapper(self):
64. # 创建目录 #
65. makeDir(TEMP\_DIR)
66. # ---END--- #
67. # 清空数据目录 #
68. delAllFilesUnderDir(TEMP\_DIR)
69. # ----END---- #
70. fileList = []
71. fileTotalByte = 0
72. taskList = []
73. **for** file **in** listAllFilesUnderDir(self.inputDir):
74. fileByte = getFileByte(self.inputDir+file)
75. **if** fileTotalByte + fileByte > self.memSizeLimit:
76. taskList.append( Task( self.mapper, readLinesOfFiles(fileList) ) )
77. fileList = []
78. fileTotalByte = 0
79. fileList.append(self.inputDir+file)
80. fileTotalByte = fileTotalByte + fileByte
81. **if** len(fileList) > 0:
82. taskList.append( Task( self.mapper, readLinesOfFiles(fileList) ) )
83. **for** task **in** taskList:
84. task.start()
86. **def** runReducer(self):
87. fileList = []
88. fileTotalByte = 0
89. taskList = []
90. **for** file **in** listAllFilesUnderDir(TEMP\_DIR):
91. fileByte = getFileByte(TEMP\_DIR+file)
92. **if** fileTotalByte + fileByte > self.memSizeLimit:
93. taskList.append( Task( self.reducer, readLinesOfFiles(fileList) ) )
94. fileList = []
95. fileTotalByte = 0
96. fileList.append(TEMP\_DIR+file)
97. fileTotalByte = fileTotalByte + fileByte
98. **if** len(fileList) > 0:
99. taskList.append( Task( self.reducer, readLinesOfFiles(fileList) ) )
100. **for** task **in** taskList:
101. task.start()
103. **def** filtOutput(self, top = 10, oper = **lambda** x : int(x.split('\t')[1][:-1]), rev = True):
104. # 创建目录 #
105. makeDir(self.outputDir)
106. # ---END--- #
107. # 清空数据目录 #
108. delAllFilesUnderDir(self.outputDir)
109. # ----END---- #
110. lines = readLines(TEMP\_DIR + 'data.out')
111. lines.sort(key = oper, reverse = rev)
113. writeLines(self.outputDir + 'data.out', lines)
115. **print**("# \tkey\tvalue\n")
116. **for** index **in** range(top):
117. **print**(str(index+1) + ".\t" +lines[index].strip())
118. #!/usr/bin/env python
119. # -\*- coding: utf-8 -\*-
120. '''''
121. wordmapreduce.py
122. '''
123. **import** string
124. **from** filetool **import** writeOneLine, getWriteOneLineHandle
125. **from** hadooptool **import** Mapper, Reducer, Job, Shuffle, TEMP\_DIR

128. **class** WordShuffle(Shuffle):
129. **def** \_\_init\_\_(self):
130. self.partitionPathHandlerDict = {}
131. **for** ch **in** string.ascii\_lowercase:
132. self.partitionPathHandlerDict[ch] = getWriteOneLineHandle(TEMP\_DIR + 'partition\_' + ch)
134. **def** partition(self, key, value):
135. key2 = key.strip("[\s+\.\!\/\_,$%^\*(+\"\']+|[+——！，。？、~@#￥%……&\*（）:：；;]+").lower()
136. **if** (key2[:1] >= 'a' **and** key2[:1] <= 'z'):
137. writeOneLine(self.partitionPathHandlerDict[key2[:1]], ('%s\t%s' % (key2, value)))
139. **def** sort(self, lines):
140. lines.sort(key=**lambda** x:x.split('\t')[0])
142. **def** \_\_del\_\_(self):
143. **for** key **in** self.partitionPathHandlerDict:
144. self.partitionPathHandlerDict[key].close()
146. **class** WordMapper(Mapper):
147. **def** do(self, lines):
148. **print**('WordMapper')
149. wordShuffle = WordShuffle()
150. **for** line **in** lines:
151. line = line.strip()
152. words = line.split()
153. **for** word **in** words:
154. wordShuffle.partition(word, 1)

157. **class** WordReducer(Reducer):
159. **def** do(self, lines):
160. **print**('WordReducer')
162. file\_handler = getWriteOneLineHandle(TEMP\_DIR + 'data.out')
163. wordShuffle = WordShuffle()
164. wordShuffle.sort(lines)
166. current\_word = None
167. current\_count = 0
168. word = None
170. **for** line **in** lines:
171. line = line.strip()
172. word, count = line.split('\t', 1)
173. **try**:
174. count = int(count)
175. **except** ValueError:
176. **continue**
177. **if** current\_word == word:
178. current\_count += count
179. **else**:
180. **if** current\_word:
181. writeOneLine(file\_handler, ('%s\t%s' % (current\_word, current\_count)))
182. current\_count = count
183. current\_word = word
185. **if** current\_word == word:
186. writeOneLine(file\_handler, ('%s\t%s' % (current\_word, current\_count)))
188. file\_handler.close()
189. #!/usr/bin/env python
190. # -\*- coding: utf-8 -\*-
191. '''''
192. run\_map.py
193. '''
194. **from** hadooptool **import** Job
195. **from** wordmapreduce **import** WordMapper, WordReducer
196. **if** \_\_name\_\_ == "\_\_main\_\_":
197. mapper = WordMapper()
198. reducer = WordReducer()
200. job = Job(mapper, reducer, 1000000, 'input/', 'output/')
201. job.runMapper()
202. #!/usr/bin/env python
203. # -\*- coding: utf-8 -\*-
204. '''''
205. run\_reduce.py
206. '''
207. **from** hadooptool **import** Job
208. **from** wordmapreduce **import** WordMapper, WordReducer
209. **if** \_\_name\_\_ == "\_\_main\_\_":
210. mapper = WordMapper()
211. reducer = WordReducer()
213. job = Job(mapper, reducer, 1000000, 'input/', 'output/')
215. job.runReducer()
216. #!/usr/bin/env python
217. # -\*- coding: utf-8 -\*-
218. '''''
219. run\_output.py
220. '''
221. **from** hadooptool **import** Job
222. **from** wordmapreduce **import** WordMapper, WordReducer
223. **if** \_\_name\_\_ == "\_\_main\_\_":
224. mapper = WordMapper()
225. reducer = WordReducer()
227. job = Job(mapper, reducer, 1000000, 'input/', 'output/')
229. job.filtOutput()

**Step2: 特征统计**

统计train文件中的domain的分布。

* 1. 通过单步调试测试代码；
  2. 通过hadoop stream完成统计。

> head -300000 train | python mapper.py | sort -k1,1 | python reducer.py | sort -k2,2nr | head -n 10

> sudo -u hdfs hadoop jar /opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop-mapreduce/hadoop-streaming.jar \

-files mapper.py,reducer.py \

-mapper mapper.py \

-reducer reducer.py \

-input /exp/kaggle/input/\* -output /exp/kaggle/output

1. #!/usr/bin/env python
2. """
3. mapper.py
4. """
6. **import** sys
7. line1 = sys.stdin.readline()
8. **for** line **in** sys.stdin:
9. line = line.strip()
10. keys = line.split(',')
11. **print** '%s\t%s' % (keys[7], 1)
12. #!/usr/bin/env python
13. """
14. reducer.py
15. """
17. **from** operator **import** itemgetter
18. **import** sys
20. current\_word = None
21. current\_count = 0
22. word = None
24. **for** line **in** sys.stdin:
25. line = line.strip()
26. word, count = line.split('\t', 1)
27. **try**:
28. count = int(count)
29. **except** ValueError:
30. **continue**
31. **if** current\_word == word:
32. current\_count += count
33. **else**:
34. **if** current\_word:
35. **print** '%s\t%s' % (current\_word, current\_count)
36. current\_count = count
37. current\_word = word
39. **if** current\_word == word:
40. **print** '%s\t%s' % (current\_word, current\_count)

**Step3: Reduce过程的负载均衡：**

1. 统计train中总的点击次数和未点击次数，并查看reducer的工作情况。
2. 由于现在的计算机都是多核的，且计算框架也趋向分布式。思考如何增加reducer的数目，提高运算速率。（提示：修改hadoop streaming的参数）

> head -300000 train | python mapper.py | sort -k1,1n | python reducer.py

> sudo -u hdfs hadoop jar /opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop-mapreduce/hadoop-streaming.jar \

-D map.output.key.field.separator=, \

-D mapred.text.key.partitioner.options=-k3,3 \

-D mapred.reduce.tasks=8 \

-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \

-input /exp/kaggle/input/\* \

-output /exp/kaggle/output11 \

-mapper mapper.py \

-reducer reducer.py \

-file mapper.py \

-file reducer.py

mapper.py

1. #!/usr/bin/env python
2. """
3. mapper.py
4. """
6. **import** sys
7. line1 = sys.stdin.readline()
8. **for** line **in** sys.stdin:
9. line = line.strip()
10. keys = line.split(',')
11. **print** '%s\t%s' % (keys[1], 1)

reducer.py

同上

**Step4: 二次排序：**

通过上一步学习到的partition配置，统计每个site\_category下面site\_domain的分布。

> head -300000 train | python mapper.py | sort -k1,1 | python reducer.py | sort -k2,2nr | head -n 10

> sudo -u hdfs hadoop jar /opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop-mapreduce/hadoop-streaming.jar \

-D map.output.key.field.separator=, \

-D mapred.text.key.partitioner.options=-k8,8 \

-D mapred.reduce.tasks=4 \

-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \

-input /exp/kaggle/input2/\* \

-output /exp/kaggle/output12 \

-mapper mapper.py \

-reducer reducer.py \

-file mapper.py \

-file reducer.py

mapper.py

1. #!/usr/bin/env python
2. """
3. mapper.py
4. """
6. **import** sys
7. line1 = sys.stdin.readline()
8. **for** line **in** sys.stdin:
9. line = line.strip()
10. keys = line.split(',')
11. **print** '%s\t%s' % (keys[6]+','+keys[7], 1)

reduce.py

同上

**Step5: 特征提取：**



其中，t0：当天未点击次数；t1：当天点击次数；p1：前一天点击次数；p7：前7天点击次数。