Big Data Analytics Techniques and Applications

Homework 3

309652022 黃伯丞

A. 資料預處理

這次作業的第一題要找出文字在文章中的出現頻率，文章中有一些是標點符號，為了方便統計我先利用 python 計算句點數以得到文章句數，並且把 .?,—:“ ”; 這些符號刪除，接著將所有字母都，存成新文字檔，程式碼為 hw3_1(pre).py。

```
# Preprocessing data
f=open('Youvegottofindwhatyoulove.txt')
text=f.read()
# number of sentences
sen_num=text.count('.')+text.count('?')
# Delete some char
for a in '.?,—:  “”  ;':
  text=text.replace(a,'')
text=text.replace('\n\n',' ')
# Alphabet with lower form
text=text.lower()
f=open('Youvegottofindwhatyoulove.txt','w')
f.write(text)
```

第二題要統計不同支付方式及乘客人數的平均 total_amount，我利用 pandas 把需要的行 ('passenger_count', 'payment_type', ' total_amount ') 取出後，將 total_amount 大於 0 的資料留下以避免產生過大偏差，然後另存成 csv 檔。程式碼為 hw3_2(pre).py。

```
import pandas as pd
cols=['passenger_count','payment_type','total_amount']

dtypes = {'Embarked': 'category'}
df=pd.read_csv('https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-10.csv',dtype=dtypes,usecols=cols)

df=df.loc[df.total_amount>0]

df.to_csv('yellow_tripdata_2018-10.csv')
```

B. 環境設定

我在虛擬機平台 VMware 上建造兩台主機，分別為 master 跟 node。然後在 master 先安裝 spark (2.4.8 版) 再依照指示於兩台主機建立 hadoop 帳號以及完成 hadoop (2.10.1 版) 的安裝。

C. 結果

I. Q1

把處理過的檔案利用 hdfs dfs –put 指令放到 hdfs 上的 test 資料夾，接著利用程式執行。首先讀取檔案到 rdd 中，然後根據空白來分割文字，接著統計次數，因為是

分散式處理所以需要執行 reduceByKey 把重複的單字合併計算，最後藉由 hw3_1(pre).py 可以知道句數為 155，用來算出頻率並印出前 30 名。

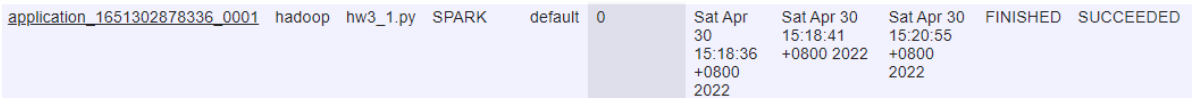
```
sen_num=155
rdd=spark.sparkContext.textFile('hdfs://master:9000/test/Youvegottofindwhatyoulove.txt')
# cut words
rdd2=rdd.flatMap(lambda x: x.split(' '))
# count words
rdd3=rdd2.map(lambda word: (word,1))
# addtion with same words
from operator import add
rdd4=rdd3.reduceByKey(add)
# sorting with value
rdd5=rdd4.sortBy(lambda x: x[1])
# frequency per sentence
rdd6=rdd5.map(lambda x: (x[0], float(x[1]/sen_num)))

# Top 30 most frequent words
list2=rdd6.collect()[-31:-1]
list2=list2[::-1]
print(list2)
```

在 yarn- cluster 模式下執行程式，指令如下所示。

```
./bin/spark-submit --conf spark.yarn.appMasterEnv.PYSPARK_PYTHON=/usr/bin/python3 --master yarn --deploy-mode cluster ~/hw3_1.py
```

完成後在 application 頁面 (master port 8088) 顯示成功訊息。



產生的 output 可以到 logs 的頁面得到 stdout 內容，整理成表格如下。

單字	1~15 名 頻率	單字	16~30 名 頻率
i	0.5548387096774193	me	0.11612903225806452
to	0.45806451612903226	have	0.10967741935483871
and	0.432258064516129	so	同上
it	0.3419354838709677	for	同上
was	0.3096774193548387	all	0.1032258064516129
a	0.2967741935483871	life	同上
of	0.2709677419354839	your	同上
that	0.24516129032258063	as	0.0967741935483871
in	0.21935483870967742	what	同上
you	0.2	on	同上
my	0.1935483870967742	but	0.09032258064516129
is	0.18064516129032257	college	同上
had	0.14193548387096774	be	同上
out	0.12903225806451613	from	0.08387096774193549
with	0.12258064516129032	when	0.07741935483870968

從表格可以發現出現頻率最多的是代名詞、介系詞以及 be 動詞，除此之外值得留意的是 life 跟 college 應該與文章內容有相當密切的關連。

II. Q2

第二題跟第一題一樣先把處理過的檔案利用 hdfs dfs –put 指令放到 hdfs 上的 test 資料夾，接著利用程式執行。只要存入 dataframe 再利用迴圈設定條件就能跑出結果。

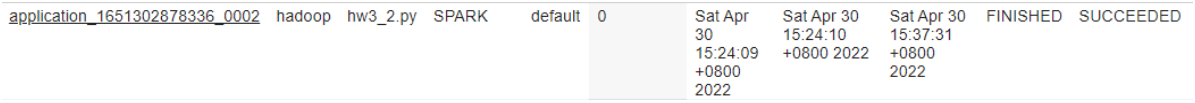
```
# Preprocessing data
df=spark.read.csv('hdfs://master:9000/test/yellow_tripdata_2018-10.csv',header=True,inferSchema=True)

# Determine the mean of total amount over different payment/passengers
from pyspark.sql.functions import mean
for i in range(1,3):
  for j in range(1,5):
    print(df.filter(df.payment_type==i).filter(df.passenger_count==j).select(mean('total_amount')).collect()[0])
```

在 yarn- cluster 模式下執行程式，指令如下所示。

```
./bin/spark-submit --conf spark.yarn.appMasterEnv.PYSPARK_PYTHON=/usr/bin/python3 --master yarn --deploy-mode cluster ~/hw3_2.py
```

完成後在 application 頁面 (master port 8088) 顯示成功訊息。



將結果統計於表格，會發現以不同人數的情況下，Cash 方式的金額差距比 Credit 方式大，但 Credit 方式的平均總額比較大，推測是因為只有信用卡方式才有小費的資訊。

人數\付款方式	Credit	Cash
1	18.098457273556804	13.69313563744156
2	18.823799196674123	14.739681435355779
3	18.398771972976867	14.81734538053979
4	18.68047782466334	15.517858225793551

III. Q3

第三題我是在虛擬機上實作，上網找了不同方法想讓 hadoop UI 的 History 可以運作但沒有效果，所以我直接從原來的 log 上觀察，在第二題中需要收集八次資料，所以整個程式共 map 8 次 (根據 spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 7 to 192.168.116.131:46604)。

從 NameNode metrics 來看，總共有 14 個 block 以及 41 個 file。

```
"name" : "Hadoop:service=NameNode,name=FSNamesystemState",
"modelerType" : "org.apache.hadoop.hdfs.server.namenode.FSNamesystem",
"CapacityTotal" : 19942490112,
"CapacityUsed" : 152215552,
"CapacityRemaining" : 11295596544,
"TotalLoad" : 1,
"SnapshotStats" : "{\"SnapshottableDirectories\":0,\"Snapshots\":0}",
"NumEncryptionZones" : 0,
"FsLockQueueLength" : 0,
"BlocksTotal" : 14,
"MaxObjects" : 0,
"FilesTotal" : 41,
"PendingReplicationBlocks" : 0,
"UnderReplicatedBlocks" : 0,
"ScheduledReplicationBlocks" : 0,
"PendingDeletionBlocks" : 0,
"BlockDeletionStartTime" : 1651302793628,
"FSState" : "Operational",
"NumLiveDataNodes" : 1,
"NumDeadDataNodes" : 0,
"NumDecomLiveDataNodes" : 0,
"NumDecomDeadDataNodes" : 0,
"VolumeFailuresTotal" : 0,
"EstimatedCapacityLostTotal" : 0,
"NumDecommissioningDataNodes" : 0,
"NumStaleDataNodes" : 0,
"NumStaleStorages" : 0,
```


從 DataNode metrics 來看，因為 Remaining / Capacity 超過五成，所以運作上相對安全。


```
"name" : "Hadoop:service=DataNode,name=FSDatasetState",
"modelerType" : "FSDatasetState",
"tag.Context" : "FSDatasetState",
"tag.StorageInfo" : "FSDataset{dirpath='[/usr/local/hadoop/tmp/dfs/data/current]'}",
"tag.Hostname" : "node",
"Capacity" : 19942490112,
"DfsUsed" : 152215552,
"Remaining" : 11295596544,
"NumFailedVolumes" : 0,
"LastVolumeFailureDate" : 0,
"EstimatedCapacityLostTotal" : 0,
"CacheUsed" : 0,
"CacheCapacity" : 0,
"NumBlocksCached" : 0,
"NumBlocksFailedToCache" : 0,
"NumBlocksFailedToUnCache" : 12
```


從 Yarn-cluster 的數據來看，unhealthyNodes, lostNodes, appsFailed 都是 0，代表運作正常。

```
▼<clusterMetrics>
    <appsSubmitted>2</appsSubmitted>
    <appsCompleted>2</appsCompleted>
    <appsPending>0</appsPending>
    <appsRunning>0</appsRunning>
    <appsFailed>0</appsFailed>
    <appsKilled>0</appsKilled>
    <reservedMB>0</reservedMB>
    <availableMB>8192</availableMB>
    <allocatedMB>0</allocatedMB>
    <reservedVirtualCores>0</reservedVirtualCores>
    <availableVirtualCores>8</availableVirtualCores>
    <allocatedVirtualCores>0</allocatedVirtualCores>
    <containersAllocated>0</containersAllocated>
    <containersReserved>0</containersReserved>
    <containersPending>0</containersPending>
    <totalMB>8192</totalMB>
    <totalVirtualCores>8</totalVirtualCores>
    <totalNodes>1</totalNodes>
    <lostNodes>0</lostNodes>
    <unhealthyNodes>0</unhealthyNodes>
    <decommissioningNodes>0</decommissioningNodes>
    <decommissionedNodes>0</decommissionedNodes>
    <rebootedNodes>0</rebootedNodes>
    <activeNodes>1</activeNodes>
    <shutdownNodes>0</shutdownNodes>
```


D. 問題討論

第一題的字數統計除了我在預處理上所調整過的之外，還有單複數、動詞分詞跟縮寫單字是我沒有動的，單複數、動詞分詞可以把同樣的字串聯在一起，縮寫單字則是把它寫成完整的樣子，在意義上有助於統計得更精確。一開始我是使用 spark-shell 來進行操作，因為第二題規定要使用 spark on yarn-cluster mode，所以需要改成先寫成 .jar/.py 檔再利用 spark-submit 傳送到主機運行。在這個部分因為它有許多參數可以調整所以需要花時間去了解。目前我還沒有機會能夠去利用 GCP, AWS 等平台，希望以後能接觸。