CSC555 project phase 2

Xiaotong He

# Part 1: Data Transformation (15 pts)

Task: Transform part.tbl table into a \*-separated (‘\*’) file: Use Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions).

Get data part.tbl:

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl>

***Method 1: Using Hive***

Before using hive, need to connect to hdfs first:

start-dfs.sh; start-yarn.sh; mr-jobhistory-daemon.sh start historyserver

Create table part in hive:

create table part (

p\_partkey int,

p\_name varchar(22),

p\_mfgr varchar(6),

p\_category varchar(7),

p\_brand1 varchar(9),

p\_color varchar(11),

p\_type varchar(25),

p\_size int,

p\_container varchar(10)

)

> row format delimited fields terminated by '|' stored as textfile;

;

A close up of text on a black background

Description automatically generated

Load data from local to table part:

hive> load data local inpath '/home/ec2-user/part.tbl'

> overwrite into table part;

A picture containing food, drawing

Description automatically generated

Check if data is successfully loaded:

Hive> select \* from part;

A close up of a screen

Description automatically generated

Create a transformed table part\_new: (the first and the last column swap)

create table part\_new (

p\_container varchar(10),

p\_name varchar(22),

p\_mfgr varchar(6),

p\_category varchar(7),

p\_brand1 varchar(9),

p\_color varchar(11),

p\_type varchar(25),

p\_size int,

p\_partkey int

)

row format delimited fields terminated by '\*' stored as textfile;

Transformation:

hive> INSERT OVERWRITE TABLE part\_new

SELECT p\_container,p\_name, p\_mfgr, p\_category, p\_brand1, p\_color,p\_type, p\_size, p\_partkey FROM part;

hadoop fs -cat /user/hive/warehouse/part\_new/000000\_0

A picture containing photo

Description automatically generated

***Method 2: Using MapReduce with HadoopStreaming***

Put data to hdfs:

hadoop fs -mkdir -p /data/part

hadoop fs -put part.tbl /data/part

hadoop jar hadoop-streaming-2.6.4.jar -input /data/part/part.tbl -mapper part\_mapper.py -file part\_mapper.py -output /data/part\_output

A screenshot of a cell phone

Description automatically generated

hadoop fs -cat /data/part\_output/part-00000

A screenshot of a cell phone

Description automatically generated

***Method 3: Using Pig***

Create table in pig: (assuming data is in the home directory on HDFS)

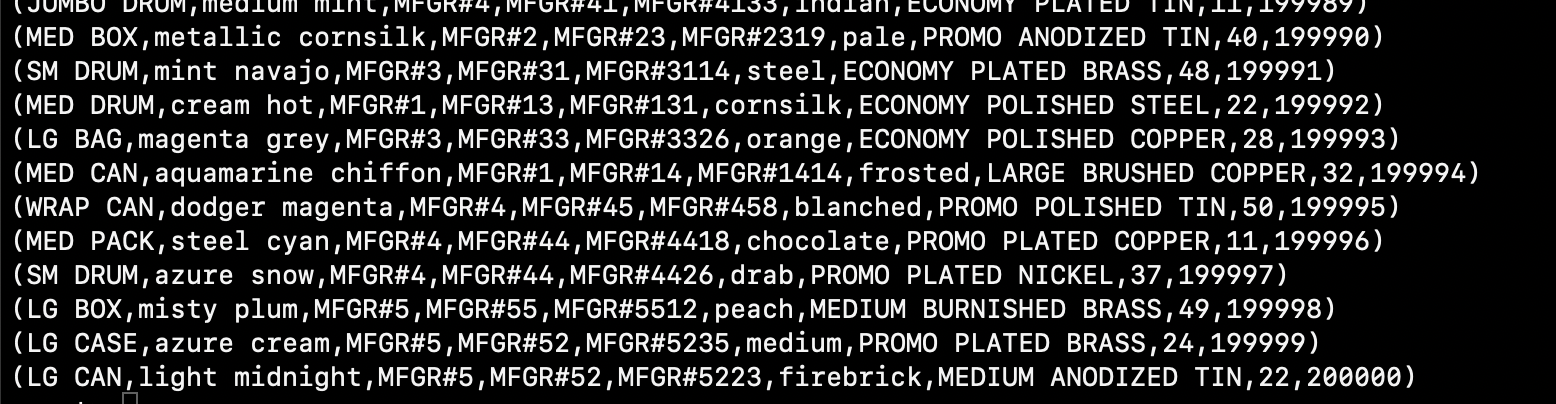
grunt> part\_data = LOAD '/user/ec2-user/part.tbl' USING PigStorage('|') AS (p\_partkey:int,p\_name:chararray, p\_mfgr:chararray, p\_category:chararray,p\_brand1:chararray,p\_color:chararray,p\_type:chararray,p\_size:int,p\_container:chararray);

A picture containing screenshot

Description automatically generated

OneCol = FOREACH part\_data GENERATE p\_container,p\_name, p\_mfgr, p\_category,p\_brand1,p\_color,p\_type,p\_size, p\_partkey;

DUMP OneCol



STORE OneCol INTO ‘out’ USING PigStorage(‘\*’);

A screenshot of a cell phone

Description automatically generated

A close up of a screen

Description automatically generated

A picture containing photo

Description automatically generated

# Part 2: Querying (25 pts)

***Method 1: Hive***

Get data:

Wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl>

Wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl>

Make table customer and lineorder and load data into table

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Execute the query, and it took 37.771s

A screenshot of a cell phone

Description automatically generated

***Method 2: MapReduce with HadoopStreaming***

Mapper1.py

#!/usr/bin/python

import sys

for line in sys.stdin:

line = line.strip()

line = line.split('|')

if line[1].startswith('C'):

c\_nation = line[4]

c\_custkey = line[0]

c\_region = line[5]

if len(c\_nation)>0 and len(c\_custkey)>0 and len(c\_region)>0:

if c\_region == 'AMERICA':

print int(c\_custkey),'\t',c\_nation,'\t','C'

else:

lo\_extendedprice=line[9]

lo\_custkey = line[2]

lo\_discount = line[11]

if len(lo\_extendedprice)>0 and len(lo\_custkey)>0 and len(lo\_discount)>0:

if lo\_discount == '5':

print int(lo\_custkey),'\t',eval(lo\_extendedprice),'\t','L’

Reducer1.py

#!/usr/bin/python

import sys

prevkey=None

c\_count=0

lo\_count=0

sum=0

for line in sys.stdin:

line = line.strip().split('\t')

key = line[0]

table = line[-1]

if prevkey == None:

if table == 'C':

c\_count +=1

nation = line[1]

elif table == 'L':

lo\_count +=1

sum += eval(line[1])

prevkey = key

elif key == prevkey:

if table == 'C':

c\_count +=1

nation = line[1]

elif table == 'L':

lo\_count +=1

sum += eval(line[1])

elif key != prevkey:

if c\_count \* lo\_count >0:

print prevkey,'\t',nation,'\t',sum

if table =='C':

c\_count=1

lo\_count=0

else:

c\_count=0

lo\_count=1

sum=0

prevkey=key

if c\_count \* lo\_count >0:

print key,'\t',nation,'\t',sum

Mapper2.py

#!/usr/bin/python

import sys

for line in sys.stdin:

line = line.strip().split('\t')

key = line[1]

price = line[2]

print key,'\t',price

Reducer2.py

#!/usr/bin/python

import sys

prevkey=None

lo\_price=0

sum=0

for line in sys.stdin:

line = line.strip().split('\t')

key = line[0]

price = eval(line[1])

total = eval(line[2])

if prevkey == None:

prevkey = key

lo\_price +=price

sum += total

elif key == prevkey:

lo\_price +=price

sum += total

elif key!= prevkey:

avg = lo\_price / sum

print prevkey,'\t',avg

lo\_price =0

sum =0

prevkey = key

avg = lo\_price / sum

print key,'\t',avg

test case:

A picture containing drawing

Description automatically generated

hadoop jar hadoop-streaming-2.6.4.jar -input /data/customer -output /data/customer/output -mapper MR/mapper1.py -file MR/mapper1.py -reducer MR/reducer1.py -file MR/reducer1.py -mapper MR/mapper2.py -file MR/mapper2.py -reducer MR/reducer2.py -file MR/reducer2.py

A screenshot of a cell phone

Description automatically generated

hadoop fs -cat /data/customer/output/part-00000

A picture containing black, drawing, meter

Description automatically generated

***Method 3: Using Pig***

Create table and load data into table (from HDFS):

hadoop fs -put customer.tbl

hadoop fs -put lineorder.tbl

customer\_data= LOAD '/user/ec2-user/customer.tbl' USING PigStorage('|') AS (c\_custkey:int, c\_name:chararray,c\_address:chararray,c\_city:chararray,c\_nation:chararray,c\_region:chararray,c\_phone:chararray,c\_mktsegment:chararray);

A close up of text on a black background

Description automatically generated

lineorder\_data = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|') AS (lo\_orderkey:int, lo\_linenumber:int, lo\_custkey:int, lo\_partkey:int, lo\_suppkey:int, lo\_orderdate:int, lo\_orderpriority:chararray, lo\_shippriority:chararray, lo\_quantity:int, lo\_extendedprice:int, lo\_ordertotalprice:int, lo\_discount:int, lo\_revenue:int, lo\_supplycost:int, lo\_tax:int, lo\_commitdate:int, lo\_shipmode:chararray);

A close up of text on a black background

Description automatically generated

a\_join = JOIN customer\_data BY c\_custkey, lineorder\_data BY lo\_custkey;

table = FILTER a\_join BY (c\_region == 'AMERICA') AND (lo\_discount == 5);

dis = GROUP table BY c\_nation;

result = FOREACH dis GENERATE group, AVG(table.lo\_extendedprice);

dump result;

A screenshot of a cell phone

Description automatically generated

STORE result INTO ‘out\_join’ USING PigStorage(‘\t’);

A picture containing food

Description automatically generated

# Part 3: Clustering (30 pts)

3a)

Generate some random data with 3 columns: user ID, movie ID, user rating

(user ID range from 1-6064, movie ID range from 1-3600, user rating range from 1-5)

Ger\_random.py

#!/usr/bin/python

import random

file = open('/home/ec2-user/random.txt','w')

for i in range(25000):

col1 = str(random.randint(1,100))+' '

col2 =str(random.randint(1,100))+' '

col3 = str(random.randint(1,100))+' '

col4 = str(random.randint(1,100))+ '\n'

line= col1+col2+col3+col4

file.write(line)

file.close()

more random.txt

A close up of a logo

Description automatically generated

Put random txt to HDFS

hadoop fs -mkdir -p testdata

hadoop fs -put random.txt testdata

time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job

A picture containing computer

Description automatically generated

mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --output clusteranalyze.txt

A screenshot of a cell phone

Description automatically generated

More clusteranalyze.txt

A picture containing computer

Description automatically generated

3b)

Manually create 5 centers randomly.

A picture containing drawing

Description automatically generated

Mapper.py

#!/usr/bin/python

import sys

import math

centers=[]

file = open('/home/ec2-user/center.txt','r')

content=file.readlines()

for line in content:

line = line.strip()

line = line.split('\t')

for i in range(len(line)):

line[i] = int(line[i])

centers.append(line)

for line in sys.stdin:

line = line.strip().split('\t')

result=[]

for i in range(len(line)):

line[i]=int(line[i])

for center in centers:

dis = math.sqrt((line[0]-center[0])\*\*2+(line[1]-center[1])\*\*2+(line[2]-center[2])\*\*2+(line[3]-center[3])\*\*2)

result.append(dis)

mindis=min(result)

for i in range(len(result)):

if mindis == result[i]:

print '\t'.join([str(i),str(line[0]),str(line[1]),str(line[2]),str(line[3])])

reducer.py

#!/usr/bin/python

import sys

curtkey =None

prevkey =None

clusters=[]

count=0

for line in sys.stdin:

line =line.strip().split('\t')

key = line[0]

c1=int(line[1])

c2=int(line[2])

c3=int(line[3])

c4=int(line[4])

new = [c1,c2,c3,c4]

if curtkey ==None:

curtkey = key

clusters.append(new)

prevkey = curtkey

elif key == prevkey:

clusters.append(new)

elif key!=prevkey:

new\_cens=[]

for i in range(4):

sum=0

new\_cen=[0,0,0,0]

for each in clusters:

sum+=each[i]

new\_cen[i]=sum/len(clusters)

new\_cens.append(new\_cen[i])

print prevkey,'\t',new\_cens[0],'\t',new\_cens[1],'\t',new\_cens[2],'\t',new\_cens[3],'\n'

clusters=[]

prevkey = key

new\_cens=[]

for i in range(4):

new\_cen=[0,0,0,0]

sum=0

for each in clusters:

sum+=each[i]

new\_cen[i]=sum/len(clusters)

new\_cens.append(new\_cen[i])

print key,'\t',new\_cens[0],'\t',new\_cens[1],'\t',new\_cens[2],'\t',new\_cens[3],'\n'

cat random.txt | python mapper\_km.py| sort | python reducer\_km.py

A close up of a screen

Description automatically generated

hadoop jar hadoop-streaming-2.6.4.jar -input /data/kmeans -output /data/kmeans/output -mapper mapper\_km.py -file mapper\_km.py -reducer reducer\_km.py -file reducer\_km.py

A screenshot of a cell phone

Description automatically generated

Stored the ouput and run the second time:

hadoop jar hadoop-streaming-2.6.4.jar -input /data/kmeans -output /data/kmeans/output2 -mapper mapper\_km.py -file mapper\_km.py -reducer reducer\_km.py -file reducer\_km.py

A screenshot of a cell phone

Description automatically generated

A close up of a screen

Description automatically generated