CA675: Cloud Technologies

Assignment 1

Name: Satya Prakash

Student Number: 19210200

Q] Analyse Data by:

a) Acquiring from Stack Exchange

b) Load them with PIG

c) Query them with Hive

d) Calculate TF-IDF (term frequency-inverse document frequency) with MapReduce.

Ans -> We have four subproblems viz. acquiring data from stack exchange, loading with PIG, querying with Hive and then calculating TF-IDF (term frequency-inverse document frequency) with MapReduce. The approach is as follows:

1. ***Acquiring data from stack exchange***

To acquire the data from stack-exchange, we use the following queries:

**SELECT TOP 50000 \* from posts ORDER BY viewCount DESC;**

**SELECT \* from posts ORDER BY viewCount DESC OFFSET 50000 ROWS FETCH NEXT 50000 ROWS ONLY;**

**SELECT \* from posts ORDER BY viewCount DESC OFFSET 100000 ROWS FETCH NEXT 50000 ROWS ONLY;**

**SELECT \* from posts ORDER BY viewCount DESC OFFSET 150000 ROWS FETCH NEXT 50000 ROWS ONLY;**

1. ***Loading them with PIG***

The steps are as follows:

1. Upload all the four files
2. Concatenate these files to a new CSV file using cat command as:

cat 1stSet.csv >> myfile.csv

cat 2ndSet.csv >> myfile.csv

cat 3rdSet.csv >> myfile.csv

cat 4thSet.csv >> myfile.csv

1. Now we replace ‘\n’ with ‘\\n’ using following command and store into a new csv file myfileOut.csv:

sed ‘:a;N;$!ba;s/\n//g’ myfile.csv > myfileOut.csv

1. Now we put it into hdfs using:

hadoop fs -put myfileOut.csv /StoreInPig

1. We move into PIGLATIN using ‘pig’ command and begin with ETL:

i. Define CSVLoader org.apache.pig.piggybank.storage.CSVLoader();

ii. Load using following command: inputData = LOAD '/StoreInPig' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES\_MULTILINE', 'NOCHANGE', 'SKIP\_INPUT\_HEADER') as (id :int,posttypeid :int,acceptedanswerid :int,parentid :int,creationdate :chararray,deletiondate :chararray,score :int,viewcount :int,body :chararray,owneruserid :int,ownerdisplayname :chararray,lasteditoruserid :int,lasteditordisplayname :chararray,lasteditdate :chararray,lastactivitydate :chararray,title :chararray,tags :chararray,answercount :int,commentcount :int,favoritecount :int,closeddate :chararray,communityowneddate : chararray);

iii. Now GENERATE using: temporal = FOREACH inputData GENERATE id,score,viewcount,REPLACE(REPLACE(body,'\n',''),',','') AS body,owneruserid,ownerdisplayname,title,tags;

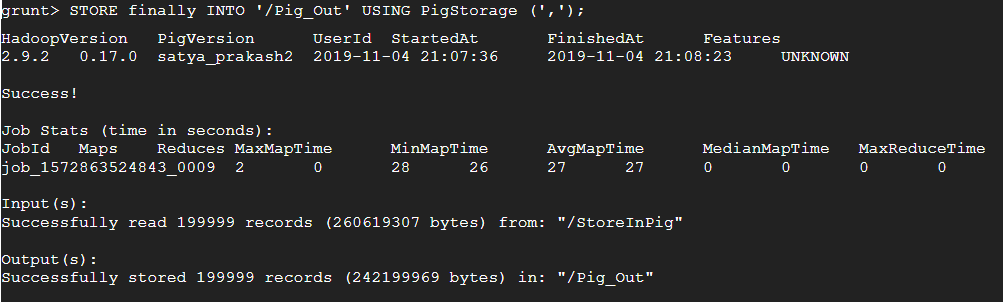
iv. Clean using: finally = FOREACH temporal GENERATE FLATTEN((id,score,viewcount,body,owneruserid,ownerdisplayname,title,tags));

v. To check we first limit ‘finally’ upto 5 and then dump it:

limit\_finally = LIMIT finally 5;

dump limit\_finally;

vi. Now we store ‘finally’ into ‘Pig\_Out’



vi. Pig divides the result in two files in hdfs. We delete the \_SUCCESS so that it does not block the load function in HIVE: hdfs fs -rm /Pig\_Out/\_SUCCESS;

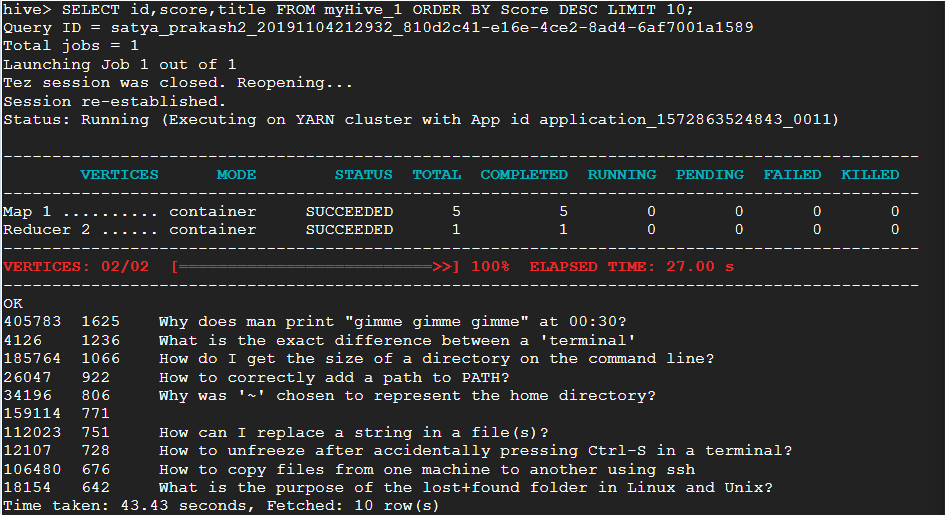
***III. Querying with HIVE***

i. CREATE TABLE myHive\_1(id int, score BIGINT, viewcount BIGINT, body string, owneruserid string, ownerdisplayname string, title string, tags string)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';

ii. LOAD DATA INPATH '/Pig\_Out' INTO TABLE myHive\_1;

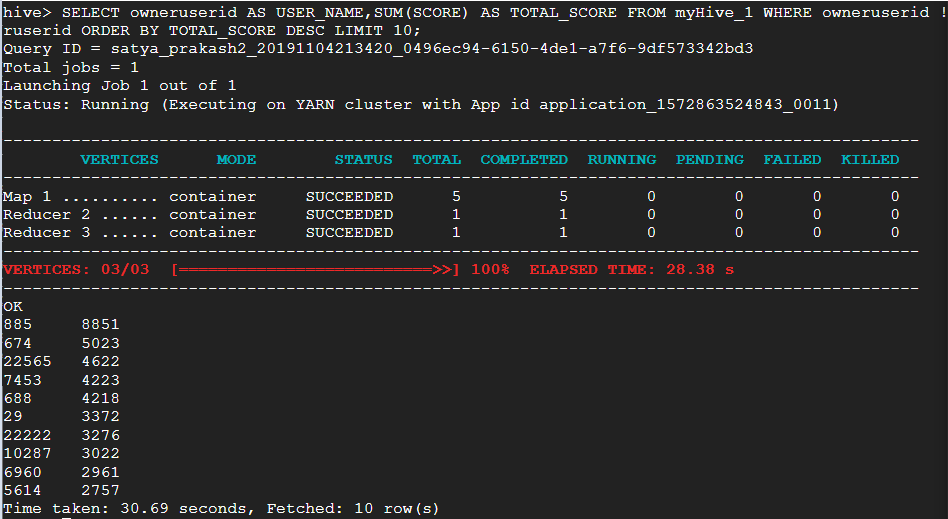
**a] Top 10 posts by Score:**

iii. SELECT id,score,title, FROM myHive\_1 ORDER BY Score DESC LIMIT 10; ***(It will return top 10 posts by score)***



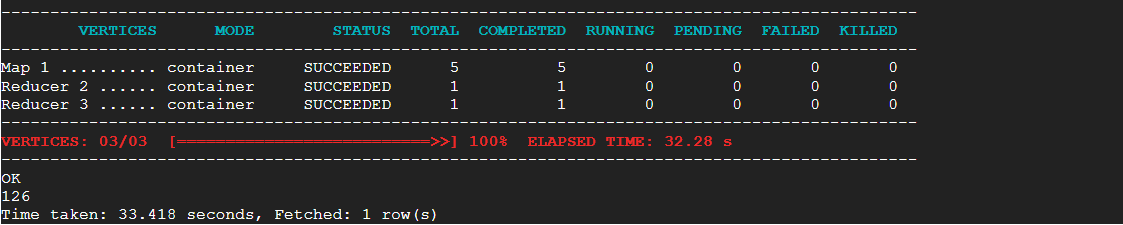
**b] Top 10 users by post score:**

iv. SELECT owneruserid AS USER\_NAME,SUM(SCORE) AS TOTAL\_SCORE FROM myHive\_1 WHERE owneruserid != '' GROUP BY owneruserid ORDER BY TOTAL\_SCORE DESC LIMIT 10; ***(It will return top 10 users by post score)***



**c] The number of distinct users who used the word “Hadoop” in one of their post:**

v. SELECT COUNT(DISTINCT(owneruserid)) FROM myHive\_1 WHERE lower(body) like '%hadoop%' or lower(tags) like '%hadoop%' or lower(title) like '%hadoop%'; ***(It will return the count of distinct users who used the word Hadoop in one of their post)***



vi. insert overwrite local directory '/home/satya.prakash2/myTFIDF' row format delimited fields terminated by ',' select owneruserid, body from myHive\_1 where owneruserid in (select USER\_NAME from MyDATATABLE);

***IV. Calculate TF-IDF (term frequency-inverse document frequency) with MapReduce and submit the top 10 terms from each of the top 10 users from Query III.***

In general TF-IDF is calculated using the following formula:

TF-IDF = TF (*t, d*) \* IDF (*t, D*)

Here, *t* is the *term* that occurs in *document d* and *D* is the total number of documents such that *d ϵ D* (*d* belongs to *D*).

*We start with HIVE Commands as follows:*

i. Table creation for different parameters used in CSV output of Pig: create table MyDATATABLE as SELECT owneruserid AS USER\_NAME,SUM(SCORE) AS TOTAL\_SCORE FROM myHive1 WHERE owneruserid != '' GROUP BY owneruserid ORDER BY TOTAL\_SCORE DESC LIMIT 10;

ii. SELECT USER\_NAME,TOTAL\_SCORE from MyDATATABLE;

iii. SELECT owneruserid, body from myHive1 where owneruserid in (select USER\_NAME from MyDATATABLE);

iv. Override HIVE Table with myTFIDF: insert overwrite local directory '/home/satya.prakash2/myTFIDF' row format delimited fields terminated by ',' select owneruserid, body from myHive1 where owneruserid in (select USER\_NAME from MyDATATABLE);

v. hadoop fs -put /myTFIDF /inputData\_MapReduce

vi. inputData\_MapReduce will obtain the output from myTFIDF with: hadoop fs -ls /inputData\_MapReduce

*Now we implement Mapper and Reducer [1] (the source files have been zipped) as follows:*

**Phase 1:**

i. hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file /home/satya.prakash2/MapperPhaseOne.py /home/satya.prakash2/ReducerPhaseOne.py -mapper "python MapperPhaseOne.py" -reducer "python ReducerPhaseOne.py" -input /inputData\_MapReduce -output /output1

**Phase 2:**

ii. hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file /home/satya.prakash2/MapperPhaseTwo.py /home/satya.prakash2/ReducerPhaseTwo.py -mapper "python MapperPhaseTwo.py" -reducer "python ReducerPhaseTwo.py" -input /output1/part-00000 /output1/part-00001 /output1/part-00002 /output1/part-00003 /output1/part-00004 -output /output2

**Phase 3:**

iii. hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file /home/satya.prakash2/MapperPhaseThree.py /home/satya.prakash2/ReducerPhaseThree.py -mapper "python MapperPhaseThree.py" -reducer "python ReducerPhaseThree.py" -input /output2/part-00000 /output2/part-00001 /output2/part-00002 /output2/part-00003 /output2/part-00004 -output /output3

**Merging:**

iv. hadoop fs -getmerge /output3/part-00000 /output3/part-00001 /output3/part-00002 /output3/part-00003 /output3/part-00004 /home/satya.prakash2/final\_TFIDFData.csv

*In the above four steps, the output of one command to other is provided by MapReduce. The final result from MapReduce is saved in final\_TFIDFData.csv. Now we use ‘sed’ command to replace ‘/s’ with ‘,’ again.*

v. sed -e 's/\s/,/g' final\_TFIDFData.csv > final\_TFIDFData1.csv

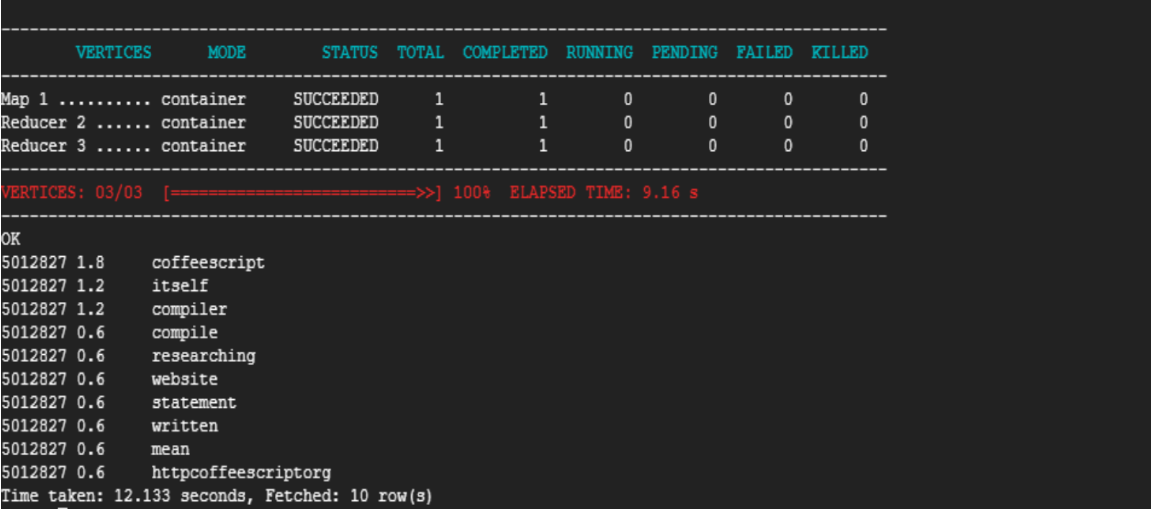
*In order to get the output of 10 words per user, we use HIVE commands as follows:*

i. create external table if not exists Table\_TFIDF\_Data(Term string,Id string,tfidf float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

ii. LOAD DATA LOCAL INPATH 'final\_TFIDFData1.csv' OVERWRITE INTO TABLE Table\_TFIDF\_Data;

iii. CREATE TABLE output\_Tfidf AS (select term,regexp\_replace(id,'[^0-9]','') as id,tfidf from Table\_TFIDF\_Data);

iv. SELECT owneruserid,tfidf,term, from Table\_TFIDF\_Data WHERE owneruserid = 5012827 SORT BY tfidf DESC LIMIT 10;



**Reference(s):**

[1] <https://github.com/SatishUC15/TFIDF-HadoopMapReduce>

**Change made to the available source code in above link:**

The original source file, [1], has a text file for stopwords in MapperPhaseOne.py. In my MapperPhaseOne.py, I have assigned a list of stopwords to the variable ‘stopwords’