

CA675 Assignment 1 - Data Analysis using Cloud technologies

Sarthak Batra, Email - sarthak.batra2@mail.dcu.ie, Student ID - 21261776, GitHub Repository - <https://github.com/sarthakbatra99/CA675Assignment1.git>

Abstract—This report is for the first assignment for CA675, 2021 taught by professor Manoj Kesavulu. In this assignment, I work on performing data analysis on Stack Exchange data. The goal is to get data from Stack Exchange, load it into the chosen technology, query the data, and calculate the TF-IDF scores. So, there are four major steps into the assignment and this report is organized into various sections explaining how the tasks have been achieved and the various steps are undertaken. All the 200,000 records, screenshots, and code snippets are stored in a GitHub repository.

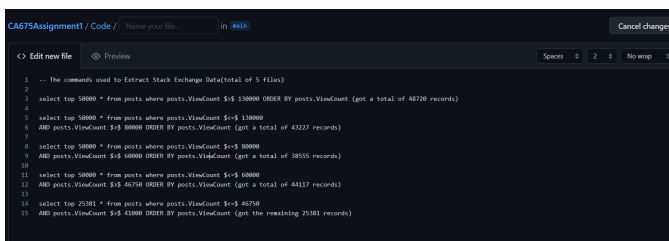
I. DATA ACQUISITION

For this assignment, the first step is data collection. The task is to acquire the top 200,000 posts by View Count from Stack Exchange. This is done by using the Data Explorer feature of Stack Exchange. It takes SQL queries from the user according to a certain set of parameters and gives the results and the number of rows. These can be downloaded as a CSV file.

As it is noted in the assignment that the maximum number of records that can be downloaded at a time is 50,000, there have to be at least 4-5 different CSV files in order to get all the 200,000 records. Therefore, to acquire the data from Stack Exchange, I had to run five queries to retrieve the top 200,000 posts according to View Count. First, I tried the following query with Viewcount 1000000.

This query gave me a total of 1506 records, which were too low. This meant that the Viewcount needed to be reduced significantly to get a greater number of posts. As I reduced the ViewCount to 100000, I got 50000 records. I think that this is because some records were being omitted due to there being more than 50000 records.

Finally, after trial and error and modifying the ViewCount, I reached the optimal value of ViewCount to get the desired data. The queries that I used to extract the records were as follows -



```

1 ... The commands used to Extract Stack Exchange Data (total of 5 files)
2
3 select top 50000 * from posts where posts.ViewCount >= 100000 ORDER BY posts.ViewCount (got a total of 46738 records)
4
5 select top 50000 * from posts where posts.ViewCount <= 100000
6 AND posts.ViewCount >= 50000 ORDER BY posts.ViewCount (got a total of 43227 records)
7
8 select top 50000 * from posts where posts.ViewCount <= 50000
9 AND posts.ViewCount >= 10000 ORDER BY posts.ViewCount (got a total of 38555 records)
10
11 select top 50000 * from posts where posts.ViewCount <= 50000
12 AND posts.ViewCount >= 46738 ORDER BY posts.ViewCount (got a total of 44117 records)
13
14 select top 25381 * from posts where posts.ViewCount <= 46738
15 AND posts.ViewCount >= 41800 ORDER BY posts.ViewCount (got the remaining 25381 records)
  
```

Fig. 1: SQL Queries used in Data Explorer

Finally, I had a total of 5 different CSV files which had all the records containing top 200,000 posts from Stack Exchange.

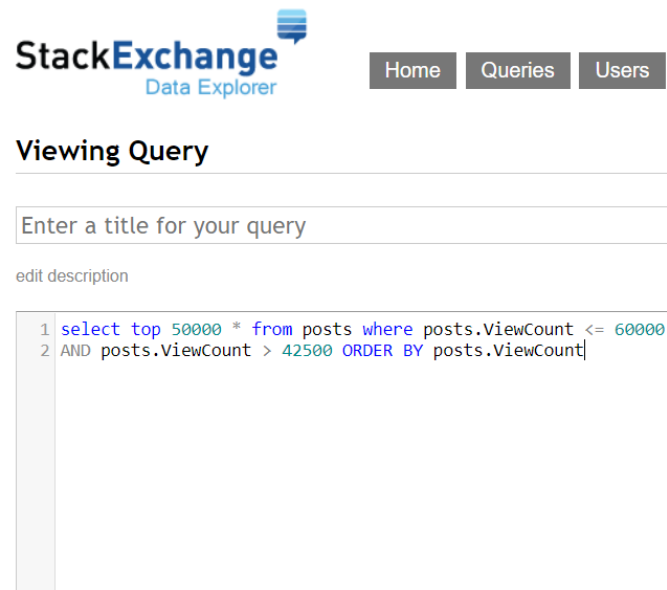


Fig. 2: SQL Query Snippet in Data Explorer

II. LOADING AND CLEANING DATA

The next step involved loading the data that is acquired to clean it and run queries. The first step involved loading the data into the Hadoop file system. A directory named Records was made and all the five CSV files were moved to it from the local storage.

The next step involved loading the data from the Hadoop file system into Pig, cleaning it for further use. For this, I chose to use Pig since pig can enable me to write a script to load and clean the data which can be run thereafter. This cannot be done in Hive, where commands need to be written one by one in Hive interface like SQL, therefore making debugging harder in this case. So, I chose to use Pig.

In my pig script, I had commands for loading the data and cleaning the data. For loading the data into Pig, I used the LOAD command, which helped in loading the CSV files from the HDFS into Pig. Then, I decided to clean the data. For this, I decided to filter out some columns and only chose the columns that I thought were most relevant, i.e. - Id, Score, OwnerUserId, ViewCount, and body. I removed the NULL

values, new lines from the body and removed HTML tags, Finally, the pig script contained the command for storing the cleaned data back to the HDFS directory for further use.

```
hadoop@sarthak-VirtualBox:~$ hadoop fs -put /home/hadoop/records/QueryResultsONE.csv /Records/
2021-10-26 01:22:56,930 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
hadoop@sarthak-VirtualBox:~$ hadoop fs -put /home/hadoop/records/QueryResultsTWO.csv /Records/
2021-10-26 01:23:14,081 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
hadoop@sarthak-VirtualBox:~$ hadoop fs -put /home/hadoop/records/QueryResultsTHREE.csv /Records/
2021-10-26 01:23:28,623 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
hadoop@sarthak-VirtualBox:~$ hadoop fs -put /home/hadoop/records/QueryResultsFOUR.csv /Records/
2021-10-26 01:23:46,182 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
hadoop@sarthak-VirtualBox:~$ hadoop fs -put /home/hadoop/records/QueryResultsFIVE.csv /Records/
2021-10-26 01:24:00,410 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
```

Fig. 3: HDFS Data load commands

```
Job Stats (time in seconds):
JobId  Maps  Reduces MaxMapTime  MinMapTime  AvgMapTime  MedianM
apTime MaxReduceTime MinReduceTime  AvgReduceTime  MedianReduceTime  A
llas  Feature Outputs
job_1635280290026_0007  3    1    51    22    40    47    41    4
1    41    41    filtered_stackdata,final_stackdata,stackdata  DISTINCT
T    hdfs:///fin,

Input(s):
Successfully read 200003 records (251456399 bytes) from: "hdfs:///Records/*.csv"

Output(s):
Successfully stored 200001 records (182230092 bytes) in: "hdfs:///fin"

Counters:
Total records written : 200001
Total bytes written : 182230092
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
```

Fig. 4: Pig Script Successfully executed

```
hadoop@sarthak-VirtualBox:~$ hadoop fs -ls /
2021-10-27 15:35:18,173 WARN util.NativeCodeLoader: Unable to load native-hadoop
p library for your platform... using builtin-java classes where applicable
Found 7 items
drwxr-xr-x - hadoop supergroup 0 2021-10-26 01:24 /Records
-rw-r--r-- 1 hadoop supergroup 1110 2021-10-26 22:15 /cleandata.pig
drwxr-xr-x - hadoop supergroup 0 2021-10-26 19:28 /cleanedrecords
-rw-r--r-- 1 hadoop supergroup 1172 2021-10-26 18:27 /datacleaning.pig
drwxr-xr-x - hadoop supergroup 0 2021-10-26 22:17 /fin
drwxrwxr-x - hadoop supergroup 0 2021-10-26 22:17 /tmp
drwxr-xr-x - hadoop supergroup 0 2021-10-25 19:34 /user
```

Fig. 5: Cleaned data in hdfs in the directory fin

III. QUERYING THE DATA

A. Choice of platform and implementation

For querying the data, I chose to use Hive. This is because it has an SQL-like interface and is easy to use. Also, the queries run pretty fast. I loaded the data that has been cleaned by Pig. The first step involved creating a table in accordance with the data to be loaded in Hive. Then, I loaded the data that I had cleaned earlier using pig into the table. This data is loaded using the fin directory in the Hadoop file system. This data was available for querying and finding the top 10 posts by score, top 10 users by post scores, and the number of distinct users who used the word "cloud" in one of their posts.

```
hive> CREATE TABLE stack_data(id int, score int, OwnerUserId int, ViewCount int
, body string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
> LINES TERMINATED BY '\n';
OK
Time taken: 0.697 seconds
hive> show tables;
OK
stack_data
Time taken: 0.068 seconds, Fetched: 1 row(s)
hive>
```

Fig. 6: Creating the stack_data table in Hive

```
hive> LOAD DATA INPATH 'hdfs:///fin/part-r-00000' INTO TABLE stack_data;
Loading data to table default.stack_data
OK
Time taken: 0.478 seconds
hive>
```

Fig. 7: Loading the data in Hive table

B. Hive Querying

There were a total of three queries that had to be run as part of the assignment. After the creation of the table, I started executing the queries. I verified the total number of posts in the table using the count * command in hive. This verified the total number of commands in my hive table to be 200,000

The first query involved finding the top 10 posts by score.

```
2021-10-28 12:07:04,581 Stage-1 map = 0%, reduce = 0%
2021-10-28 12:07:09,769 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.54 sec
2021-10-28 12:07:14,956 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.4 sec
MapReduce total cumulative CPU time: 5 seconds 400 msec
Ended Job = job_1635417201668_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.4 sec HDFS Read: 179496480 HDFS Write: 340 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 400 msec
OK
id score 25933
11227809
927358 23348
2083505 18514
292357 12834
231767 13551
477816 10921
348170 10879
5767325 9931
5591213 8792
1642028 9560
Time taken: 17.618 seconds, Fetched: 10 row(s)
hive>
```

Fig. 8: Query 1 Output

The second query involved finding the top 10 users by post scores.

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.65 sec HDFS Read: 179498793 HDFS Write: 2658832 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 5.05 sec HDFS Read: 2603690 HDFS Write: 325 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 700 msec
OK
owneruserid total_score
87224 23872
9951 26799
6068 25944
89904 24624
51816 23719
49153 20283
95202 19479
178736 19429
63851 19345
Time taken: 42.777 seconds, Fetched: 10 row(s)
hive>
```

Fig. 9: Query 2 Output

The final query involved searching for the number of distinct users who used the word cloud in one of their posts.

```
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.78 sec HDFS Read: 179496114 HDFS Write: 103 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 780 msec
OK
c0
102
Time taken: 20.591 seconds, Fetched: 1 row(s)
hive>
```

Fig. 10: Query 3 Output

```

-- Query 1 ( Top 10 posts by scores)
SELECT id,score from stack_data ORDER BY score DESC LIMIT 10;

-- Query 2 ( Top 10 users by post scores)
SELECT OwnerId, sum(score) AS total_score FROM stack_data GROUP BY OwnerId ORDER BY total_score DESC LIMIT 10;

-- Query 3 ( Number of distinct users with the word cloud in one of their posts
SELECT COUNT(DISTINCT OwnerId) from stack_data WHERE OwnerId IS NOT NULL AND body LIKE '% cloud %' OR body LIKE '% Cloud %';

```

Fig. 11: Hive Queries

IV. ISSUES FACED

While working on this assignment, I faced some issues -

- 1) - While loading the data in Pig, I was getting a syntax error, which was a mismatch error, caused by an extra ',' in the line. I searched for it online on forums to understand the error and was able to resolve it.
- 2) - An issue that I faced was the fact that I was not able to run the pig script completely. The script took too long to execute and would get stuck at 60-65% eventually. I searched for a solution and had to increase the number of processors to 2 and the base memory to 5120 MB in my Ubuntu Virtual Machine.
- 3) - After resolving the memory issue, I tried to run my Pig script again. One issue that I was facing was that after 95 %, there was an issue where the connection with the Hadoop server stopped and it kept retrying it to no avail. I decided to debug it by running each command individually in the grunt shell. I found the issue was with the last command, which stored the data back into the HDFS file system. It took quite a while to figure out because of very little documentation online. I eventually figured out that I had to start the job history daemon in Hadoop before running the script. I ran the following command, which solved my issue -
`mr-jobhistory-daemon.sh start historyserver`
- 4) - I wasn't able to do the fourth task which involved calculating the TF-IDF scores. I wrote the scripts for MapReduce in python but was not able to make them run properly in time. Therefore, I was unable to complete this task.

V. RESULTS

Finally, after completing the assignment, I was able to find the top 10 posts by score, top 10 users by post scores and the number of distinct users who used the word "cloud" in one of their posts. I found this assignment to be really informative and helped me learn about using hive and pig. I was not familiar with these technologies before and working on the assignment gave me a practical knowledge about how to load data into the HDFS file system, cleaning data using Pig and querying the data using Hive.

While working on my assignment, I referred to online sources to help me in installation of Hadoop, Hive and Pig. I also referred to sources to help me understand some commands, official documentations and forums to help solve errors that I faced. The following references section contains the links that I used to assist me while doing my project.

REFERENCES

- [1] How to install Apache Hive on ubuntu step-by-step guide. Knowledge Base by phoenixNAP. (2021, August 23). Retrieved October 27, 2021, from <https://phoenixnap.com/kb/install-hive-on-ubuntu>.
- [2] How to install hadoop on ubuntu 18.04 or 20.04. Knowledge Base by phoenixNAP. (2021, July 1). Retrieved October 27, 2021, from <https://phoenixnap.com/kb/install-hadoop-ubuntu>.
- [3] Apache pig installation: Setting up apache pig on linux. Edureka. (2019, May 22). Retrieved October 27, 2021, from <https://www.edureka.co/blog/apache-pig-installation>.
- [4] sparkcodegeeks1. (2020, December 14). Hadoop FS: HDFS DFS commands with Examples - Sparkbyexamples. Spark by Examples. Retrieved October 27, 2021, from <https://sparkbyexamples.com/apache-hadoop/hadoop-hdfs-dfs-commands-and-starting-hdfs-dfs-services/>.
- [5] Learner, P. by A. A., Anand, P. by, Anand, Learner, A., amp; Anand, V. all posts by. (2017, July 5). Pig programming: Create your first apache pig script. BIG IS NEXT- ANAND. Retrieved October 27, 2021, from <https://bigishere.wordpress.com/2016/07/14/pig-programming-create-your-first-apache-pig-script/>.
- [6] Assignment 1 files/code/pig code · master · Daniel Higgins / CA675 cloud technologies. GitLab. (n.d.). Retrieved October 27, 2021, from <https://gitlab.com/computing.dcu.ie/higgid23/ca675-cloud-technologies/blob/master/Assignment%201%20Files/Code/Pig%20Code>.
- [7] Prix, PrixPrix 1911 silver badge55 bronze badges, Gaurav PhapaleGaurav Phapale 94911 gold badge77 silver badges2121 bronze badges, Lester MartinLester Martin 19155 bronze badges, Indrajeet GourIndrajeet Gour 2, karthikkarthik 55177 silver badges2121 bronze badges, NagaNaga 1, NarasimhaNarasimha 1133 bronze badges, amp; pradeeppradeep 3. (1962, December 1). How to import/load .CSV file in pig? Stack Overflow. Retrieved October 27, 2021, from <https://stackoverflow.com/questions/25598645/how-to-import-load-csv-file-in-pig>.
- [8] Crowder, C. (2021, August 16). How to fix ubuntu freezing in Virtualbox. Make Tech Easier. Retrieved October 27, 2021, from <https://www.maketecheasier.com/fix-ubuntu-freezing-virtualbox/>.
- [9] Language manual. Language Manual - Apache Hive - Apache Software Foundation. (n.d.). Retrieved October 27, 2021, from <https://cwiki.apache.org/confluence/display/Hive/LanguageManual>.
- [10] Pig Latin Basics. (n.d.). Retrieved October 27, 2021, from <https://pig.apache.org/docs/r0.17.0/basic.html>.