**Assignment 1: Data Analysis**

**Student Details**

|  |  |
| --- | --- |
| Student ID: | 20211188 |
| Student name: | Kunal Vaidya |
| Student email | kunal.vaidya2@mail.dcu.ie |
| Chosen major: | Data Analytics |
| Module Code: | CA675 Cloud Technologies |
| Date of Submission | 12/11/2020 |

**Data Acquisition**

The data is acquired from a platform called Stack Exchange. The data consists of the posts from Stack Overflow(<https://data.stackexchange.com/stackoverflow/query/new>). The following queries are executed on the platform to fetch the top 200,000 posts based on their ViewCount. The platform restricts to have 50,000 records in a query, so four queries were executed, and the output was saved into 4 CSV files, namely posts\_1.csv, posts\_2.csv, posts\_3.csv, and posts\_4.csv. The queries used for getting the data are as follows:

SELECT TOP 50000 \* FROM posts where ViewCount >80000 ORDER BY ViewCount DESC;

SELECT TOP 50000 \* FROM posts where ViewCount <= 112523 ORDER BY ViewCount DESC;

SELECT TOP 50000 \* FROM posts where ViewCount <= 66244 ORDER BY ViewCount DESC;

SELECT TOP 50000 \* FROM posts where ViewCount <= 47290 ORDER BY ViewCount DESC;

**Extract, Transform and Load (ETL) Processes**

Google Cloud Platform (GCP) was used to perform the operations. Hadoop cluster is created using DataProc, which is a managed Hadoop cluster solution by Google.

The 4 CSV files were uploaded to a temporary GCP bucket and then were loaded into the Name Node machine(cluster-f078-m) of the cluster. The files were copied to the HDFS into the '/data' directory.

## Pig

Pig was used to load and transform the data. All the commands are saved in a pig script called pig **loadData.pig**. Here are the commands used:

1. CSVExcelStorage class is used to load the data into a variable from the HDFS. The data contains 22 columns. Arguments have been specified which enable to process multi-line data and skip the header row.

posts = load 'hdfs://cluster-f078-m/data/' using org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES\_MULTILINE','NOCHANGE','SKIP\_INPUT\_HEADER') as(id:int,posttypeid:int,acceptedanswerid:int,parentid:int,creationdate:DATETIME,deletiondate:DATETIME,score:int,viewcount:int,body:chararray,owneruserid:int,ownerdisplayname:chararray,lasteditoruserid:int,lasteditordisplayname:chararray,lasteditdate:DATETIME,lastactivitydate:DATETIME,title:chararray,tags:chararray,answercount:int,commentcount:int,favoritecount:int,closeddate:DATETIME,communityowneddate:DATETIME,contentlicense:chararray);

1. The body column contains multi-line data which makes it difficult to load into a hive table. Removing the '\n', '\t' and '\r' characters from the body column.

posts = foreach posts generate id,posttypeid,acceptedanswerid,parentid,creationdate,deletiondate,score,viewcount,REPLACE(body,'\\n','') as body,owneruserid,ownerdisplayname,lasteditoruserid,lasteditordisplayname,lasteditdate,lastactivitydate,title,tags,answercount,commentcount,favoritecount,closeddate,communityowneddate,contentlicense;

posts = foreach posts generate id,posttypeid,acceptedanswerid,parentid,creationdate,deletiondate,score,viewcount,REPLACE(body,'\\t','') as body,owneruserid,ownerdisplayname,lasteditoruserid,lasteditordisplayname,lasteditdate,lastactivitydate,title,tags,answercount,commentcount,favoritecount,closeddate,communityowneddate,contentlicense;

posts = foreach posts generate id,posttypeid,acceptedanswerid,parentid,creationdate,deletiondate,score,viewcount,REPLACE(body,'\\r','') as body,owneruserid,ownerdisplayname,lasteditoruserid,lasteditordisplayname,lasteditdate,lastactivitydate,title,tags,answercount,commentcount,favoritecount,closeddate,communityowneddate,contentlicense;

1. The columns body, title and tags contains ',' which breaks the data and the delimiter used is also ','. Removing the ',' from the columns and removing the columns which are not required to perform the intended queries.

formatted\_csv = FOREACH posts GENERATE id AS id, score AS score, REPLACE(body,',\*','') AS body, owneruserid AS owneruserid, REPLACE(title,',\*','') AS title, REPLACE(tags,',\*','') AS tags;

1. Removing all the records having null entries for OwnerUserID and Score column

valid\_csv = FILTER formatted\_csv BY (owneruserid IS NOT NULL) AND (score IS NOT NULL);

1. Loading the processed data into a hive table(Defined in the next section) using HCatalog class

store valid\_csv into 'userdb.posts' using org.apache.hive.hcatalog.pig.HCatStorer();

## Hive

Hive was used to store the data and perform queries on it. Command to create a table to store the data:

create table posts(id int, score int, body String, owneruserid Int, title String, tags String)

row format delimited

FIELDS TERMINATED BY ',';

All the three queries asked was answered through Hive Query Language (HQL). The hive queries for the Task 3 is written in a file called **hiveQueries.sql**

1. The top 10 posts by score

SELECT id, score, owneruserid,title from posts

ORDER BY score DESC

LIMIT 10;

1. The top 10 users by post score

SELECT owneruserid, SUM(score) AS Total\_Score from posts

GROUP BY owneruserid

ORDER BY Total\_Score DESC

LIMIT 10;

### The number of distinct users, who used the word "Hadoop" in one of their posts

SELECT COUNT(DISTINCT owneruserid) AS unique\_user\_Count from posts

WHERE (body like '%hadoop%' OR title like '%hadoop%' or tags like '%hadoop%');

**TF-IDF (Term Frequency-Inverted Document Frequency)**

The calculation of TF IDF per User of the top ten users involved the fetching of top ten users and storing in a separate table called TopUsers.

CREATE TABLE TopUsers

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',' AS

SELECT owneruserid, SUM(score) AS TotalScore

FROM posts

GROUP BY owneruserid

ORDER BY TotalScore DESC LIMIT 10;

The OwnerUserId from this TopUsers table is then used to query the top User's all of the posts' Body, Title and Tags and stored in a table called TopUserPosts.

CREATE TABLE TopUserPosts AS

SELECT owneruserid, body, title, tags

FROM posts

WHERE owneruserid in (SELECT owneruserid FROM TopUsers)

GROUP BY owneruserid, body, title, tags;

This TopUserPosts is then stored onto an HDFS directory.

INSERT OVERWRITE DIRECTORY '/data/hiveResults'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT owneruserid, body, title

FROM TopUserPosts

GROUP BY owneruserid, body, title;

The result partitions from this HDFS directory is merged using the command:

hdfs dfs -getmerge /data/hiveResults hiveResults.csv

and stored at an NFS directory. Hive was used to fetch the TopUserPosts and the queries are enclosed within **getTopUserPosts.sql** file.

The results from the above query, hiveResults.csv consists of all the User's records in the same file which was split into their respective user files to be used as input to TF IDF program. This splitting was done in python script - **splitTopUserPosts.py**

The results of the splitTopUserPosts.py script is fed file by file as input to TF IDF mappers using the **mapreduce.sh** script. The collection of TF IDF mappers reducer programs are stored in **MapReduce** folder. The commands ran to compute the TF IDF is a script called **mapreduce.sh**

bash mapreduce.sh {Name of the txt file}

The mapreduce.sh script used the following commands for executing mappers and reducers:

hadoop jar hadoop-streaming-2.7.3.jar -files mapper1.py,reducer1.py -mapper 'python mapper1.py' -reducer 'python reducer1.py' -input /data/userData/$1 -output /data/output1

hadoop jar hadoop-streaming-2.7.3.jar -files mapper2.py,reducer2.py -mapper 'python mapper2.py' -reducer 'python reducer2.py' -input /data/output1/ -output /data/output2

hadoop jar hadoop-streaming-2.7.3.jar -files mapper3.py,reducer3.py -mapper 'python mapper3.py' -reducer 'python reducer3.py' -input /data/output2/ -output /data/output3

hadoop jar hadoop-streaming-2.7.3.jar -files mapper4.py -numReduceTasks 0 -input /data/output3/ -output /data/output4 -mapper 'python mapper4.py'

The results from the above MapReduce gave results to ten files for each user. It contained the top terms and their weights. To fetch the top ten terms of each user, **sortResults.py** was run. All the final results for the TF-IDF score of the top 10 users are saved in the **TFIDF\_Results.txt** file.

**References**

**TFIDF -** <https://github.com/devangpatel01/TF-IDF-implementation-using-map-reduce-Hadoop-python->

**ETL-** <https://github.com/arunabellgutteramesh/PigHiveOnStackExchangeData>

\*Please check the 'Outputs' folder for the screenshots