

Sagar Nenwani

Student Id 21261506

sagar.nenwani2@mail.dcu.ie

<https://github.com/SagarNenwani/CA-675-Assignment-1>

CA 675 Assignment - 1

Task 1: Data Acquisition - Fetch 200K records from StackExchange by ViewCount

Stack-exchange returns 50000 rows at a time, so 4 queries were used to fetch 200000 records based on descending order of view count. To limit the first query, the max of view count was checked and for the consecutive queries, the last view count of the result from the previous query was used to limit the view count. For consecutive queries, a check for previously fetched post id not being the same (to avoid repetition of data). An additional check for OwnerUserId not being null was added to all the queries.

It is better to download only the required columns "Id, Score, ViewCount, Body, OwnerUserId, Title, Tags" as the others column would increase the noise and size of the data for evaluation.

```
SELECT MAX(ViewCount) AS MaxViewCount FROM Posts;
result 10062790

select top 50000 Id, OwnerUserId,Score, ViewCount, Body, Title, Tags
from posts
where posts.ViewCount <= 10063000
and OwnerUserId is not null and OwnerUserId >=0
ORDER BY posts.ViewCount desc

result  Last id  326818
        Last view count 124974

select top 50000 Id, OwnerUserId,Score, ViewCount, Body, Title, Tags
from posts
where posts.ViewCount <= 124974 and posts.id != 326818
and OwnerUserId is not null and OwnerUserId >=0
ORDER BY posts.ViewCount desc

result  Last id  30226113 and 1135573 and 15493769
        Last view count 73140
```

These queries are present in "stack_exchange_queries.txt". The results were then downloaded as a CSV format present in "stack_exchange data" folder.

Task 2: Load data into chosen cloud technology (MapReduce/Pig/Hive)

Before loading the data, it needed to be cleaned as the body, title and tags contained commas(,) as value which conflicted with comma used for separation. These commas created additional rows while importing data to pig as “CSVExcelStorage” treated each comma as a separator.

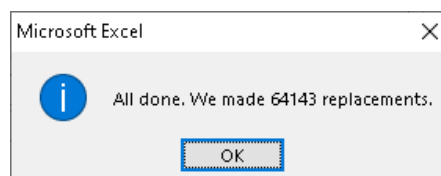
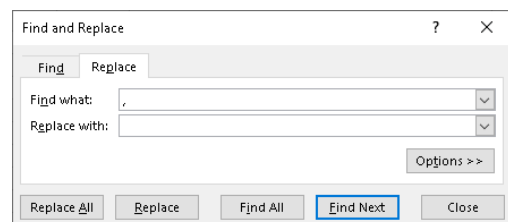
The separators were changed from (“,”) to just comma (,) while the comma (,) in values of the body, title and tags were changed to a blank () using regex in notepad++ and excel.

Also, the headers from files 2,3, and 4 were removed so that it does not add a new entry while merging all rows.

File 2 Before

	A	B	C	D	E	F	G	H	I
1	Id	OwnerUse	Score	ViewCour	Body	Title	Tags		
2	25710875	2411173	81	73139	<p>I have	python ec	<python><r><frequency>		

After removing headers and comma



	A	B	C	D	E	F	G	H	I	J
1	25710875	2411173	81	73139	<p>I have	python ec	<python><r><frequency>			
2	36328907	2879085	55	73138	<p>Let's	Ansible: G	<ansible><jinja2><ansible-playbook>			

Loading the Data

Loading of data was done in pig as it provides an easier option to directly import CSV using CSVExcelStorage and merge the data. For pig to use the CSV from the cluster, the CSV were first loaded into Hadoop using “Hadoop fs -put” command.

```
sagar_nenwani2@cluster-f74b-m:~$ hadoop fs -put QueryResults2.csv /
sagar_nenwani2@cluster-f74b-m:~$ hadoop fs -put QueryResults3.csv /
sagar_nenwani2@cluster-f74b-m:~$ hadoop fs -put QueryResults4.csv /
```

```
-rv-r--r-- 1 sagar_nenwani2 hadoop 48311127 2021-10-28 02:47 /QueryResults1.csv
-rv-r--r-- 1 sagar_nenwani2 hadoop 56021820 2021-10-28 02:50 /QueryResults2.csv
-rv-r--r-- 1 sagar_nenwani2 hadoop 58736246 2021-10-28 02:55 /QueryResults3.csv
-rv-r--r-- 1 sagar_nenwani2 hadoop 62537473 2021-10-28 02:54 /QueryResults4.csv
```

Each of the 4 CSV files were loaded into a pig variable. The first CSV was loaded with parameter ‘SKIP_INPUT_HEADER’ as it had column names/headers to CSVExcelStorage while the remaining were loaded with parameter ‘READ_INPUT_HEADER’. All the CSVs were checked for rowcount before it was processed for merging.

Queries used for loading the data can be found in “pig_load.txt”.

Task 3

Pig Querying

Querying of data was done in both pig and hive. The merged data in pig was queried by creating groups based on id to get the top 10 posts by score. To get the top 10 users by post score, the data was grouped based on OwnerUserId.

Queries can be found in "pig_retrieve.txt".

The merged data from pig was then cleaned to remove symbols, escape sequences and other clutter from the body column and then exported as csv to Hadoop cluster using the "Store" command.

Hive Querying

Hive provides with SQL type syntax to query the database. The results of pig query was also checked with hive queries. A new table "clean_data_posts" was created by using the exported data from pig and again verified for the count of rows imported.

Queries used for Task 3 are present in "hive_retrieve.txt"

Task 4 - Calculate TF-IDF

To find the TF-IDF for top 10 words used by top 10 users, the result from task 3 where we found the list of top 10 distinct users based on their sum of scores is first stored into a new table "TopUsers". We'll need to have all the posts by these 10 users and was stored using nested queries in a new table "TopUserPosts". Similar to SQL, hive provides with macros which are predefined functions and can be user defined. For this, "define-all.hive" with hive ML (hivemall-core-0.4.2-rc.2-with-dependencies) was used.

Both of this files were imported using GCP SSH to HDFS system and then imported in hive using following commands.

```
hive> add jar /home/sagar_nenwani2/hivemall-core-0.4.2-rc.2-with-dependencies.jar;  
Added [/home/sagar_nenwani2/hivemall-core-0.4.2-rc.2-with-dependencies.jar] to class path  
Added resources: [/home/sagar_nenwani2/hivemall-core-0.4.2-rc.2-with-dependencies.jar]  
hive> source /home/sagar_nenwani2/define-all.hive;
```

A new table "TokenizedBodyView" was created using the data of "TopUserPosts" to tokenize each word of the body column and assign values.

The words from "TokenizedBodyView" were used to create a table to map each word with its frequency of use. The distribution of OwnerUserId over this temporary frequency table is stored in "tfidf_body" table.

Queries to this can be found in "hive_tfidf_queries.txt"

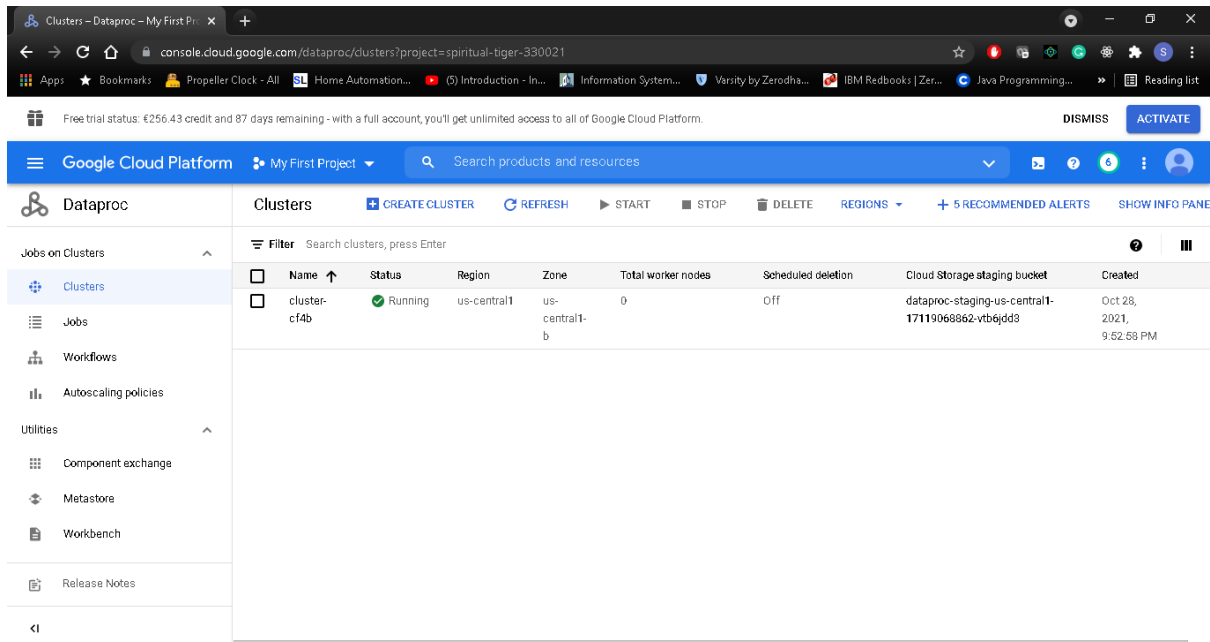
Task 5: Google Cloud Platform

All the above tasks have been performed with GCP DataProc which provided with cluster and VM instances with preinstalled Hadoop, hive and pig.

A new cluster "cluster-f74b" was created within a project with n1 memory strage and 4 core CPU.

A SSH key was generated using puttygen and updated in the VM instance of the cluster. This was done to connect to VM using third party tools like putty and winscp.

Moving of initial data like csv, .jar and .hive files were done using WinScp and Google SSH window.



The screenshot shows the Google Cloud Platform console for the project 'spiritual-tiger-330021'. The left sidebar shows the 'Dataproc' menu with options like Clusters, Jobs, Workflows, Autoscaling policies, Utilities, Component exchange, Metastore, Workbench, and Release Notes. The main panel displays the 'Clusters' page with a table of clusters.

Name	Status	Region	Zone	Total worker nodes	Scheduled deletion	Cloud Storage staging bucket	Created
cluster-cf4b	Running	us-central1	us-central1-b	0	Off	dataproc-staging-us-central1-17119068862-ytb6jdd3	Oct 28, 2021, 9:52:58 PM

The Results of all tasks are stored under “Result” folder

References

<http://xquaredl.blogspot.com/2016/08/load-and-store-data-using-pig.html>

<https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089>

<https://github.com/apache/incubator-hivemall/blob/master/resources/ddl/define-all.hive>

<https://github.com/myui/hivemall/releases/download/v0.4.2-rc.2/hivemall-core-0.4.2-rc.2-with-dependencies.jar>