CLUSTER SETUP, AND DATA PREPROCESSING AND EXTRACTION ENGINE

PROBLEM#1

MARCH 7, 2021

SRIKRISHNAN SENGOTTAI KASI
Sr284605@dal.ca

TASK 1:

1) Logged into my GCP Console.
2) Searched for VMInstance in the search bar provided inside the console.
3) On clicking "Create", a new page showing the required configurations for the instance is displayed.
4) Entered the instance name, Specified the boot image as Ubuntu 20.04 LTS and mentioned the region as US pacific.
5) A new E2 instance is created with a RAM of 2 GB. Allowed the firewall check to access it in browser.
6) The created instance is now listed in the VMInstance page.
7) Now login to the console of the instance from the VMInstance listing page.
8) Update the OS by entering sudo apt update. As the instance is new.
9) Make sure the VMInstance has open-jdk, scala and python installed for compiling and running respective files in the VM. You can verify by getting version. If it shows command not found, then install using sudo apt install <package name> -y.
10) On successful console login you can download the apache-spark using the below command,
    a. wget <URL to latest Spark TAR file>
11) Now extract the TAR file and move into that folder. Make this folder as a path for installation of SPARK in the VMInstance. For permanent path set we can edit the Ubuntu profile file to have the path inside it. Configure the SPARK packages bin folder in the path.
12) Move into the extracted spark folder and execute start-master.sh to start the SPARK in the system.
13) Successfully the SPARK instance is started in the VMInstance.

TASK 2:

Repository Link: https://git.cs.dal.ca/kasi/csci-5408-w2021-b00881083-srikrishnan/-/tree/master/Assignment_3.

Tweet Extraction Program algorithm: [Without any third-party library]

call_twitter_search_api():

```
1) BEGIN.
2) Initialize queryParamenters. Add the fields required in response and keywords that
need to be searched. Max limit for each query result is set to 100.
3) Store the Twitter Search URL and Authentication header in a variable.
4) Use Request.get(URL, QueryParameters, Headers) and store the HTTPResponse in a
variable.
5) Now the result json is retrieved from the response.
6) if response status is 200,
    i) call write_to_file(response). (The resultant has only 100 tweets)
    v) If next_token exists in meta of the response,
        1) Check if overall tweet count is less than 3500.
            i) then call_twitter_search_api()
7) Else, Print the tweet fetch is terminated with the error response from twitter.
8) END
```

write_to_file(response)

```
1)Begin.
2)Creates a random file inside output directory using current timestamp.
3)Append the response.text contents in the file.
4)Close the file.
```

```
5)Increment tweet_count, file_count, last_file_count to keep track of the system.
6)END
```

call_twitter_search_api() and write_to_file (response) will be the member functions of a TwitterExtractor class. So, a class object is created and a call to call_twitter_search_api() is made.

Implementation explanation using Tweepy

Alternatively, we can use **Tweepy** module and create a **listener** of our own. We can connect our account to the tweepy using the Auth module with our client_key, secret and bearer token. Then, we can set the filter words in the Tweepy object. By implementing a tweepyListener we can control the events once the listener is initialized. One is **on_status** method. That is overwritten with file logic and append the tweet in the file. Other is **on_error** where we can terminate the connction in case of twitter throwing any error. This in backend uses twitter stream API.

Algorithm for Twitter Filtration Engine

twitter_filter_engine():

```
1) BEGIN
2) Fetch the file names from the output directory.
3) Open connection to the Mongo Database.
4) Store the reference to Database "myMongoTweet".
5) Create a collection called "Tweet".
6) For tweet_file in tweet_op_files:
    a. tweet_json_data  <- get_list_of_tweets_from_file(tweet_file)
    b. cleaned_json_tweets <-  clean_array_of_tweets(tweet_json_data)
    c. insert cleaned_json_tweets into "Tweet" Collection.
7) END
```

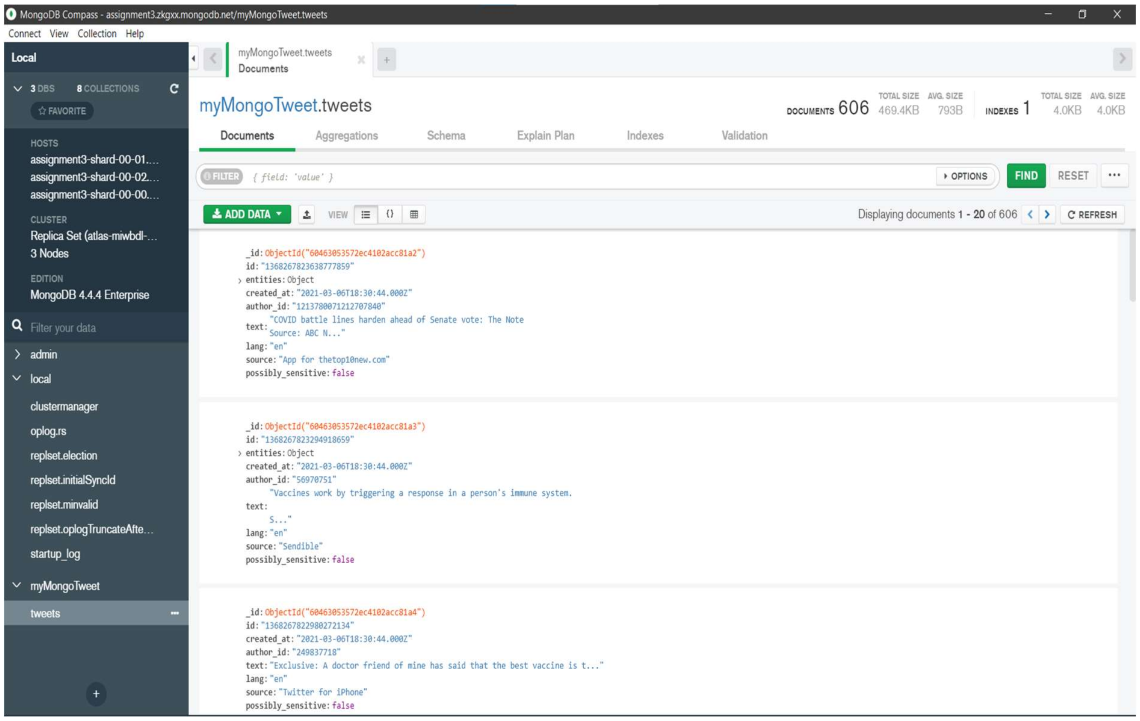
get_list_of_tweets_from_file(tweet_file**)**

```
1) BEGIN
2) Set path as "tweet_file".
3) Open File and read its content as string to "tweet_file_content".
4) Assign the json content from tweet_file_content to "tweet_loaded_json_from_file".
5) Return value in "data" key from the json "tweet_loaded_json_from_file", as it contains
   only the tweet data.
6) END
```

clean_array_of_tweets(tweet_json_data)

```
1) BEGIN
2) Create a python List object to store clean tweet contents.
3) Compile the required regex to remove URL, emoticons, special characters from the "text"
   key in "tweet_json_data" json object.
4) For each tweet in tweet_json_data:
    a. Get the text from the json.
    b. Implement the regex over the text.
    c. Store the cleaned text back into the original object.
    d. If there is no key named "referenced_tweets" in the object
        i. Append it to the result list called "tweet_cleaned_list".
5) Return the "tweet_cleaned_list"
6) END
```

MongoDB Compass Screenshot of Filtered Tweet Data [Retweets are removed]

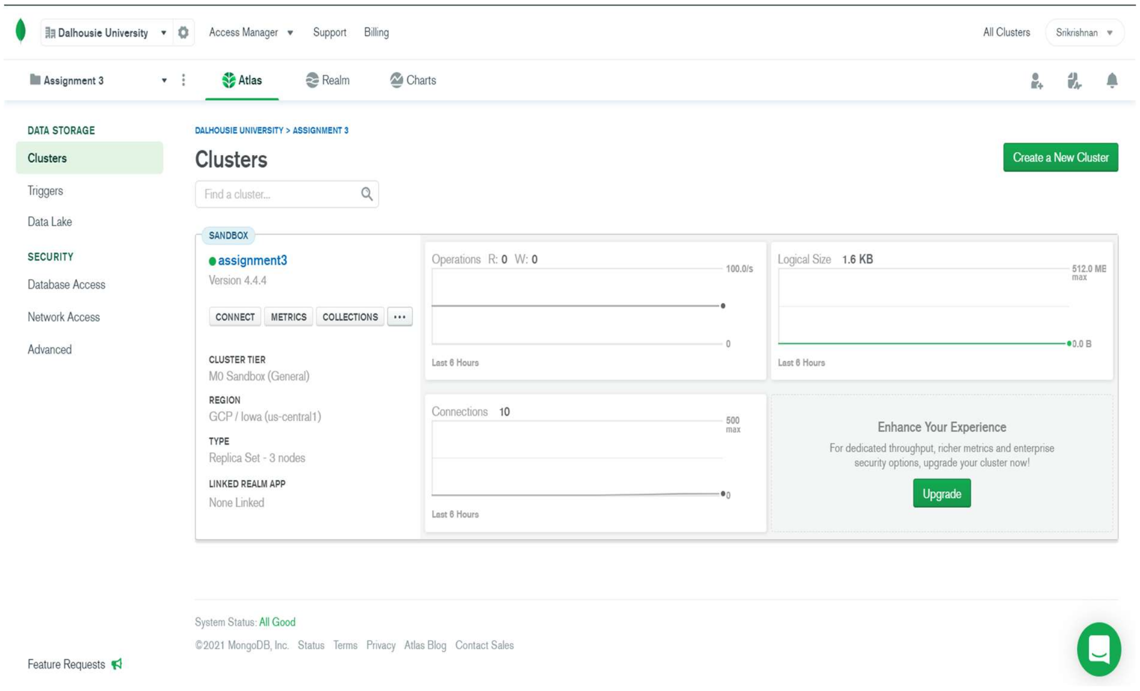

MongoDB Compass - assignment3.zkgxx.mongodb.net/myMongoTweet.tweets
Connect View Collection Help
Local
3 DBS 8 COLLECTIONS
FAVORITE
HOSTS
assignment3-shard-00-01....
assignment3-shard-00-02....
assignment3-shard-00-00....
CLUSTER
Replica Set (atlas-miwbdl-...
3 Nodes
EDITION
MongoDB 4.4.4 Enterprise
Filter your data
admin
local
clustermanager
oplog.rs
replset.election
replset.initialSyncId
replset.minvalid
replset.oplogTruncateAfte...
startup_log
myMongoTweet
tweets
myMongoTweet.tweets
Documents
DOCUMENTS 606 TOTAL SIZE 469.4KB AVG. SIZE 793B INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' } OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 20 of 606
REFRESH

```
_id: ObjectId("60463053572ec4102acc81a2")
id: "1368267823638777859"
> entities: Object
created_at: "2021-03-06T18:30:44.000Z"
author_id: "1213780071212707840"
text: "COVID battle lines harden ahead of Senate vote: The Note
      Source: ABC N..."
lang: "en"
source: "App for thetop10new.com"
possibly_sensitive: false

_id: ObjectId("60463053572ec4102acc81a3")
id: "1368267823294918659"
> entities: Object
created_at: "2021-03-06T18:30:44.000Z"
author_id: "56970751"
      "Vaccines work by triggering a response in a person's immune system.
text:
      S..."
lang: "en"
source: "Sendible"
possibly_sensitive: false

_id: ObjectId("60463053572ec4102acc81a4")
id: "1368267822980272134"
created_at: "2021-03-06T18:30:44.000Z"
author_id: "249837718"
text: "Exclusive: A doctor friend of mine has said that the best vaccine is t..."
lang: "en"
source: "Twitter for iPhone"
possibly_sensitive: false
```

MongoDB Atlas Cluster with 3 Nodes.



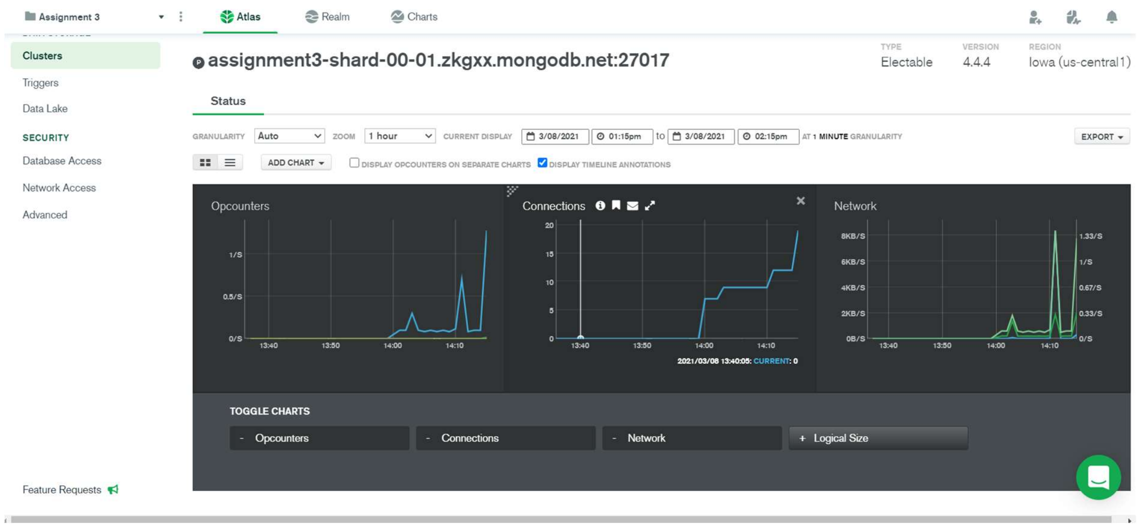
CLUSTER TIER
M0 Sandbox (General)

REGION
GCP / Iowa (us-central1)

TYPE
Replica Set - 3 nodes

LINKED REALM APP
None Linked

Cluster Usage Analysis



assignment3-shard-00-01.zkgxx.mongodb.net:27017

TYPE Electable
VERSION 4.4.4
REGION Iowa (us-central1)

GRANULARITY Auto ZOOM 1 hour CURRENT DISPLAY 3/08/2021 01:15pm to 3/08/2021 02:15pm AT 1 MINUTE GRANULARITY

☐ DISPLAY OPCOUNTERS ON SEPARATE CHARTS ☑ DISPLAY TIMELINE ANNOTATIONS

2021/03/08 13:40:05: CURRENT: 0

References

1) https://dal.brightspace.com/d2l/le/content/143359/viewContent/2263960/View

2) https://docs.tweepy.org/en/latest/streaming_how_to.html

3) https://stackoverflow.com/questions/14630288/unicodeencodeerror-charmap-codec-cant-encode-character-maps-to-undefined

4) https://spark.apache.org/docs/latest/api/python/

5) https://en.wikipedia.org/wiki/Unicode_block

6) https://gist.github.com/Alex-Just/e86110836f3f93fe7932290526529cd1#gistcomment-3208085.

7) https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query.