CSCI 5408

Assignment 2

HALIFAXINFO

MEGHNA RAMACHANDRA HOLLA | B00812604

A. Cluster Setup

1. Creating Cloud Account

Steps to create a Cloud Account: (AWS web Service)

- Created an AWS Account on Amazon and launched EC2 Instance. I followed steps mentioned in Data warehousing Lab 3[1] and AWS website [2].
- Used the Public DNS of my cloud instance and the private key generated to connect to putty.
- I have attached a screenshot of the running putty in image1(referenceImages folder)

2. Initializing Apache Spark on AWS

I followed the steps mentioned on Data warehousing Lab 4[3] to setup Apache and to start master-slave node. I have attached a screenshot of the running spark in my cloud instance in image2(referenceImages).

3. Installation of MongoDB

- Installed MongoDB using sudo apt-get install -y mongodb-org and started the mongod service.
- Stored clean data in 'twitterData' db using mongoimport command.
- Images of importing process and collections in image3-1 and image3-2(referenceImages)

B. Data Extraction and Transformation

1. Searching Twitter Data

- Wrote a Shell script 'call_search.sh'(service) that runs a Twitter command(twurl)[4] to gather data having keyword 'Halifax'. I had initially written the code using Twython but moved to twurl I thought we were not supposed to use these libraries. But our lecturer told us that we could use them to get the twitter data but not to clean it.
- The twurl command takes arguments q, result_type and count. The argument q is query keyword(q=Halifax), result_type gives recent, popular or mixed tweets, count is the number of tweets I want to gather. By default, it is 15. I did not put 100(maximum for Standard Users) as 'count' value because I did not want to collect redundant values.
- I ran this service in the background using 'sh call_search.sh start &'. It runs the 'twurl' command every two hours. I gathered raw data in 'halifax_raw_search.json'. To stop the service, I ran 'sh call_search.sh stop'
- I added square brackets in the start and end of 'halifax_raw_search.json' and included commas in between the json objects using a sed command that created 'consolidated_searching.json'.

```
sudo sed -i 's/}}{\"statuses"/}}\,{\"statuses"/g' halifax_raw_search.json>>consolidated_searching.json
```

- The 'consolidated_searching.json' file is now in a proper json format with all twitter values like id, location, user_name, screen_name, metadata and so on with special characters, URL, emojis.
- I removed all special characters, emojis and URLs from 'consolidated_searching.json' using a python script 'clean_search_data.py'[6][7].

- The output of this script is stored in 'cleanSearchFile.csv'. It has only necessary fields like id, user_name, tweet_text, created_at and retweet_count. Metadata is collected as well, but for now, the lecturer asked me to keep it as is, therefore I did not process it in this file.
- I stored clean and transformed 'cleanSearchFile.csv' file in MongoDB as mentioned in StepA-3

2. Streaming Twitter Data

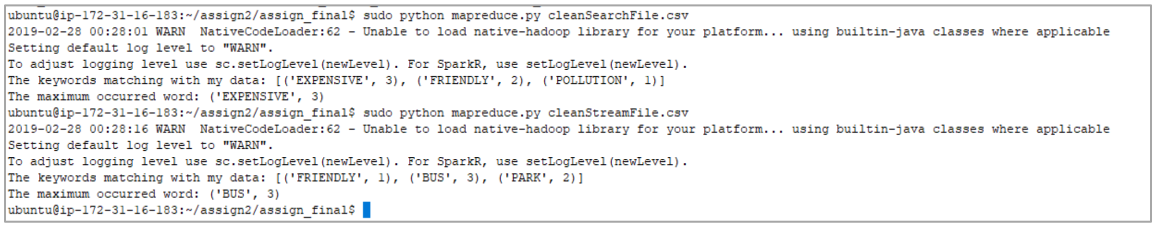
- Wrote a shell script 'call_stream.sh'(service) that uses Tweepy[5] library to extract Twitter streams with keyword 'Halifax'. The shell script calls 'get_stream_tweets.py' to extract tweets.
- I ran this service in the background using 'sh call_stream.sh start &'. It ran twice a day for three days and gathered raw data in 'halifax_raw_stream.json'. To stop the service, I ran 'sh call_stream.sh stop'
- I added square brackets in the start and end of 'halifax_raw_stream.json' file and saved it as 'consolidated_streaming.json' to make it a json file. I used 'replace' function in python to include commas in between the json objects defined in 'clean_stream_data.py'
- I removed all special characters, emojis and URLs from 'consolidated_streaming.json' using a python script 'clean_stream_data.py'[6][7].
- The output of this script will be stored in 'cleanStreamFile.csv'. It has only necessary fields like id, user_name,tweet_text,created_at and retweet_count.
- I stored clean and transformed 'cleanStreamFile.csv' file in MongoDB as mentioned in StepA-3

C. Data Processing (Map Reduce)

I passed 'cleanSearchFile.csv' and 'cleanStreamFile.csv' as arguments to my mapreduce.py[8][9][10] code by running 'python mapreduce.py classSearchFile.csv'. This starts a Spark session and creates required tuples. It gives two outputs: tuples of matching strings and the most frequently used word.

D. Answers

- The most frequently used 'search' word is 'Expensive', where as the most frequently used 'stream' word in 'Bus'.
- A Standard Twitter User can gather data from the past 7 days. I think that there was some trend going on for 'expensive' items in the history data when I ran my code, but the trend changed while streaming it. 'Friendly' word count reduced by one for Streaming data.
- The streaming words are Friendly, Bus and Park and the search words are Friendly, Expensive and Pollution.
- Output: Image [Below], Files [search output, stream output]



```
ubuntu@ip-172-31-16-183:~/assign2/assign_final$ sudo python mapreduce.py cleanSearchFile.csv
2019-02-28 00:28:01 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
The keywords matching with my data: [('EXPENSIVE', 3), ('FRIENDLY', 2), ('POLLUTION', 1)]
The maximum occurred word: ('EXPENSIVE', 3)
ubuntu@ip-172-31-16-183:~/assign2/assign_final$ sudo python mapreduce.py cleanStreamFile.csv
2019-02-28 00:28:16 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
The keywords matching with my data: [('FRIENDLY', 1), ('BUS', 3), ('PARK', 2)]
The maximum occurred word: ('BUS', 3)
ubuntu@ip-172-31-16-183:~/assign2/assign_final$
```

References

[1] https://dal.brightspace.com/d2l/le/content/86639/viewContent/1246765/View
[2] https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html
[3] https://dal.brightspace.com/d2l/le/content/86639/viewContent/1251920/View
[4] https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html
[5] http://docs.tweepy.org/en/v3.4.0/streaming_how_to.html
[6] https://docs.python.org/3/library/re.html
[7] https://docs.python.org/3/library/csv.html
[8] https://docs.python.org/3/howto/functional.html
[9] https://spark.apache.org/examples.html
[10]https://dal.brightspace.com/d2l/le/content/86639/viewContent/1251920/View